Models and algorithms for the real-time railway and air traffic flow management problems

Marcella Samà

May 4, 2016

Contents

Chapter 1

Introduction

1.1 A general overview about transport

Transport is a important and valuable aspect of the everyday life. In Europe families spend 13.5% of their income on transport-related goods and service. Furthermore, 90% of the European Union foreign export trades relies on shipping carries [EU Explained - Transport (2014)]. In [EU Statistical Pocketbook (2014)] it has been estimated that in 2012 total passenger transport activities within the EU-28 using any motorized means of transport amounted to an average of 12700 km per person. Private citizens covered 72% of this distance by car, still the most common mode of transport for private individuals, 9% by air transport, a cumulative 8% share by buses and coaches, 6% by railways, around a 2% each by powered two-wheel vehicles and trams/metros and, last, less than 1% by sea travel. Regarding freight transport, it has been estimated that in 2012 total goods transport activities within the EU-28 amounted to 3768 billion tkm. Almost 45% has been transported by road vehicles, followed by a 37% by seagoing ships, 11% by railways, 4% by inland waterways and 3% by oil pipelines. Air cargo is rated last with less than a 0.1% share.

This high usage of road both for freight and passengers translates into a series of issues spanning from pollution, e.g., road traffic is accountable for the largest share in the transport section with over 70% of CO_2 emission caused by it, to costs, e.g., road congestions costs an equivalent of 1% of the all EU GDP every year [EU Explained - Transport (2014)].

The need for people and goods mobility has steadily grown worldwide in the last decades. This trend will probably continue in the future. Public transport systems with high capacity and high speed, such as railways and air traffic, address this mobility need, bundling passenger and goods demand from dispersed origin-destination pairs to a set of interconnecting lines, reducing congestion on roads, especially on highways

and in densely populated areas. Railways in particular provides a more eco-friendly and sustainable way of transport.

Increasing the use of public transport requires that a better quality of service has to be provided, in terms of frequencies, comfort, accessibility and reliability of services, along with information availability over traveling time and routing alternatives. This growth in mobility demand and the difficulties in building new infrastructures due to economic and physical constraints translate into the need of utilizing the existing infrastructures at full capacity at all times, including peak hours. This highlights the importance of developing optimization tools for improving the performance of the existing infrastructures.

In order to achieve good performance for railways and air traffic, the activities considered the most important are usually the careful design of effective timetables, done off-line and months in advance, and real-time traffic management policies. During the daily operations, disturbances may occur, creating delays in the infrastructure. Even if buffer times are provided, time-overlapping requests for the same resources might be made by multiple vehicles, making the timetables not feasible to implement. Traffic controllers monitor the flow of vehicles and change the dynamics in the presence of disturbances, answering these overlapping requests while maintaining the safety constraints intact. Currently, the decisions are manually made, not allowing controllers to fully evaluate their effects due to the limited time available. In practice, the dispatching decisions are often sub-optimal, and leave room for improvement. Some decisions may in fact lead to the propagation of the delays to previously not-affected vehicles, which in turn may create new conflicting requests, causing in a worst case scenario cancellations. Optimization is thus necessary, minimizing the delay propagation and consequentially minimizing the worsening of the quality of the service offered.

Effective Decision Support Systems (DSSs) should be available to the controllers. These DSSs must be able to present to the dispatchers conflict-free schedules, in which a routing and start times in the resources belonging to this routing are assigned to each vehicle traveling in the infrastructure during an analyzed time horizon. Furthermore, these DSSs should consider the minimization of the delays occurring in the network. The main pre-requisite of a good DSS is the real-time ability to deal with actual traffic conditions and safety rules for practical networks. In other words, the solution provided by a DSS must be feasible in practice, since the human dispatcher may have not enough time to check and eventually adjust the schedule suggested by the DSS.

A series of European projects have been dedicated to the application of optimization tools to railways, prompted by the introduction of the European Railway Traffic Management System (ERTMS), a new block signaling standard. The COMBINE (COntrol center for a Moving Block sIgNalling systEm) project, who took place in the years 1999–2001, focused on the real-time optimization of rail traffic in areas equipped with

this new standard [Giannettoni and Savio (2002),Mascis et al., (2008),Pellegrini and Savio (2000)]. Following the conclusion of this project, in the years 2002–2003, a second one, called COMBINED2, focused on the study of networks equipped instead with mixed signaling systems [Giannettoni and Savio (2004),Mazzarello and Ottaviani (2007)]. Furthermore, in 2014 ended the ON-TIME (Optimal Networks for Train Integration Management across Europe) project, whose purpose was to increase railway capacity by reducing delays and improving traffic fluidity. As a result, a framework for the automatic real-time management of railway traffic was developed [Quaglietta et al., (2016)].

On a similar note, in 2009 started for the air sector the European project SESAR (Single European Sky ATM Research). This project aims to improve the performance of Air Traffic Management (ATM). Among the solutions proposed are reducing the controllers workload using integrated automation support and improving arrival/departure management through air traffic control intervention.

The EU has compiled a document [EU White Paper (2011)] promulgating a series of goals to achieve a more competitive and sustainable transport system in the union, including but not limited to:

- the shift of 30% of road freight over 300 km to other modes such as rail or waterborne transport by 2030, with the goal of reaching more than 50% by 2050;
- the completion by 2050 of a European high-speed rail network. This will be achieved triplicating the existing high-speed infrastructure by 2030, maintaining a dense railway network in all the EU states, and shifting the majority of medium-distance passenger transport by 2050 to railways;
- the connection of all the major airports to the (preferably high-speed) railway network;
- the deployment of modernized traffic management for the existing infrastructure, currently often used at their full capacity, in all different modes of transport, in particular in the air sector. Improving the efficiency of aircraft and traffic management operations would in fact also answer the need for airport capacity to be optimized, while not compromising the EU role of a global aviation hub.

The PhD thesis here presented follows a series of pre-existing works concerning the application of operations research techniques to the real-time optimization of transport management. In particular, at Roma Tre University a solver called AGLIBRARY has been developed, which is a set of OR-based models and algorithms for complex practical scheduling problems. The solver is based on the alternative graph model [Mascis and Pacciarelli (2002)] and on the following framework: initially, a good quality solution for the scheduling problem with fixed routings is computed using heuristics and a (truncated) branch-and-bound algorithm [D'Ariano et al., (2007a)]; afterwards, a new improved solution for the scheduling problem with flexible routings is computed by applying metaheuristics [Corman et al., (2010)]. The PhD theses of [D'Ariano

(2008), Corman (2010)] have developed and applied these models and algorithms to the management of railway traffic in real-time. In particular, they improve and incorporate AGLIBRARY as the main solution engine of the ROMA (Railway traffic Optimization by Means of Alternative graph) railway dispatching support system, developed at Delft University of Technology. However, a series of relevant algorithmic issues still need to be analyzed. This PhD thesis stems from the need to fill the existing gaps in this stream of research, concentrating on the development of further metaheuristics for the scheduling problem with flexible routings and providing simplification techniques for cases difficult to solve with the instruments previously available. Furthermore, the thesis aims to study and apply the successful models and algorithms to a different problem, such as the air traffic flow management.

In the rest of this first Chapter, Sections 1.2 and 1.3 provides more insights on the two field of application chosen, Section 1.4 presents the research objectives tackled while Section 1.5 defines the outline of the other Chapters of this thesis.

1.2 Railway traffic management

In Europe, the deregulation of the railway sector, achieved with the European Directive 2001/14/EC, resulted into the separation between railway companies and railway infrastructure managers. Railways companies operate in the market and their interest lies into gaining market shares or fulfilling contract agreements with public authorities. The infrastructure managers, instead, have the only objective of managing the infrastructures in terms of construction and utilization, providing non-discriminated access to railway capacity to all the interested railway companies in order to provide transport services to passengers and freight, ensuring the safety and the quality of the service.

These actors, representing some of the main stakeholders in the railway sector, are experiencing difficulties in ensuring a good quality of service while facing the ever increasing transport demand (Figure 1.1 shows the worldwide expected growth until 2023). This, added to the limited space and funds available to build new infrastructures in bottleneck areas, is stimulating the interest for new effective operations research solutions.

Instead of tackling an overall and highly complicated planning process, a hierarchical decision-making approach is adopted when planning railway operations, resulting in a series of tractable problems. As shown in Figure 1.2, these problems can be grouped in three levels: strategic, tactical and operational [Lusby et al., (2011)].

The problems dealing with network and line planning belong to the strategic level. These problems involve the construction of infrastructures and/or the upgrading and modification of already existing ones, assessing in particular the impact on the capacity of the overall system [Bussieck (1998), Goossens et al.,

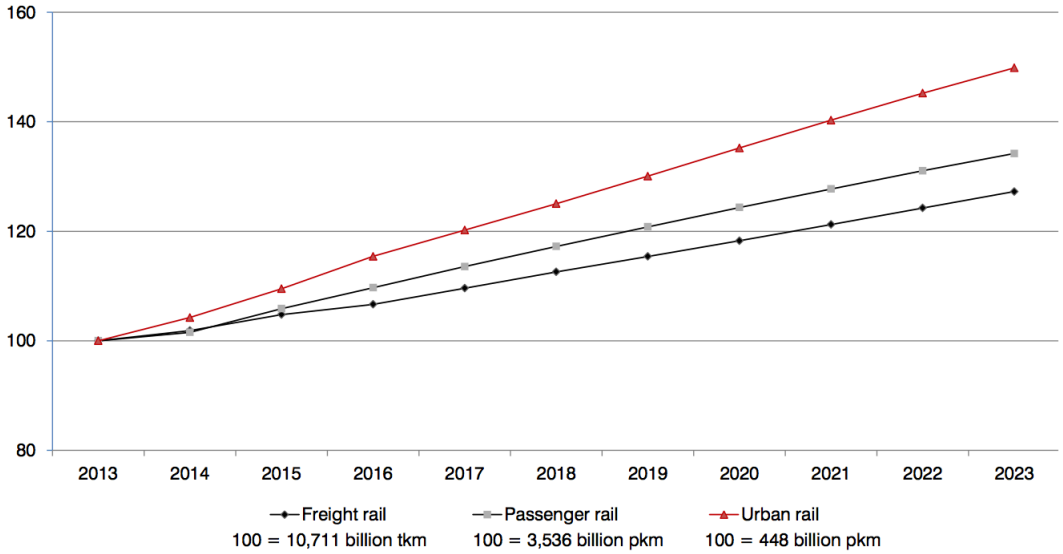


Figure 1.1: Worldwide rail transport performance 2013-2023 [Index 100 = 2013] (*Source:* [SCI "Rail Transport Markets – Global Market Trends (2014-2023"])

(2005),School (2005)]. In its basic form a rail infrastructure is composed of stations, links and block sections separated by signals. For safety reasons, signals, interlocking and Automatic Train Protection systems control the train traffic by imposing minimum safety separations between trains. To this end, railway tracks are divided into block sections or track-circuits, which may be occupied by no more than one train at a time. A detailed description of different aspects of railway signaling systems and traffic control regulations can be found in the handbook of [Hansen and Pachl (2014)]. Once an infrastructure is built, optimizing its use is a key aspect to meet the transport growth demand and is achieved through the problems tackled in the other two levels.

The problems dealing with the creation of plans for the utilization of the infrastructure and the required resources belong to the tactical level. Railway operations usually follow a timetable, carefully designed in advance in terms of routings assigned to each train, orders between trains on common tracks and timings on the required resources. Operations research techniques have been studied in the literature with large benefit to timetable generation [Cacchiani and Toth (2012),Carey and Crawford (2007),Sels et al., (2014)], rolling stock [Fioole et al., (2006), Vaidyanathan et al., (2008)] and crew scheduling problems [Ernst et al., (2004)]. Robustness considerations are often introduced in the timetables to prevent that unexpected perturbation make these plans not implementable [Cicerone et al., (2008),Liebchen et al., (2009),Schöbel and Kratz (2009)]. However, even a robust timetable may not be able to cope with perturbations. Unreliability exists in a railways system and disturbances may arise, due for example to extended dwell times at stations,

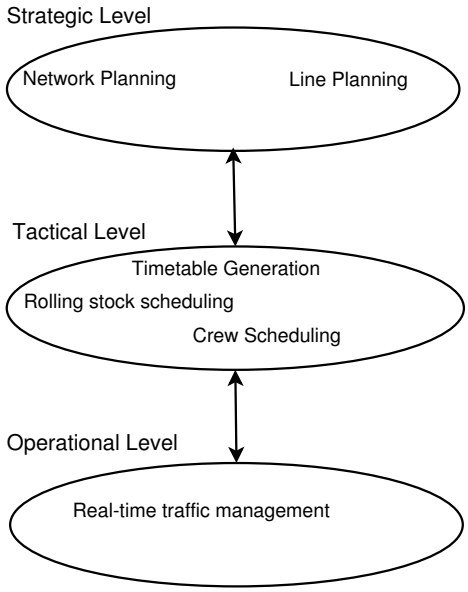


Figure 1.2: The three levels problems

infrastructure or rolling stock disruptions, incidents and so on. These disturbances create delays, which may propagate in the network, affecting other trains in a domino effect, worsening the overall quality of service offered. Even the presence of buffer times to reduce the effects of such delays may not be enough.

The problem dealing with the creation of recovery plans in real-time when such disturbances arises belongs to the operational level. When the tactical timetable becomes infeasible a new working timetable is required [Cacchiani et al., (2014), Corman and Meng (2015), Fang et al., (2015)]. Operational traffic management nowadays is limited to reacting to the disturbances. The dispatchers are often provided with limited aid from automated systems, such as graphical interfaces and automatic routing setting systems. Due to their limited time, dispatchers only change the routing setting plan when trains have a considerable delay, and network traffic controllers become active only when train traffic is already highly disrupted. A more proactive approach instead would predict the potential conflicts, preventing delays and their propagation altogether.

This thesis tackles one of the main operational level problems. Known as Conflict Detection and Resolution (CDR) or real-time Railway Traffic Management Problem (rtRTMP), the problem consists in detecting and solving conflicting track requests done by multiple trains during disturbed operations. Online adjustments regarding train routings, timings and orders are performed with the goal of achieving a feasible new timetable, i.e., each infrastructure resource is utilized by no more than one train at a time and no deadlock exist in the network, while minimizing the impact of the delay propagation.

While many models and algorithms for train rescheduling have been proposed in the literature, few have been successfully applied in practice. In AGLIBRARY the use of the alternative graph model allows to model individual and global train schedule feasibility considering the blocking time theory, enabling the detection of microscopic track conflicts in a general railway network with mixed traffic for a given look-ahead horizon, even in presence of heavy disturbances and network disruptions.

However, this is only one of the several approaches that can be studied in the literature. Other approaches model the problem in various ways, i.e., constraint programming [Rodriguez (2007)], mixed-integer programming [Dessouky et al., (2006), Lamorgese and Mannino (2015), Liu and Kozan (2009), Pellegrini et al., (2014), Törnquist and Persson (2007)], multicommodity flow models [Caimi et al., (2011)], set-packing inspired models [Lusby et al., (2013)], time-index models [Meng and Zhou (2014)], and each differs from the others also on the level of detail considered and on the solving approaches developed.

1.3 Air traffic flow management

As for the railway case previously described, an increase in transport demand is also expected in the air sector. The International Air Transport Association (IATA), together with Tourism Economics, released in 2014 a report of the passenger growth forecast for the next 20 years on an origin-destination basis [IATA (2014)]. They analyzed the air passenger flows across 4000 country pairs, using as demand drivers living standards, population and demographics, price and availability. The forecast states that the number of air passengers may reach 7.3 billion by 2034, more than doubling the 3.3 billion passengers of 2014, with a 4.1% average annual growth in demand for air connectivity. However, air traffic in peak hours is getting closer to the capacity of the Terminal Control Areas (TCAs), at least in the major European airports where there is limited possibility of creating new infrastructure. Aviation authorities are thus seeking methods to better use the existing infrastructures and to better manage aircraft movements in the proximity of airports, improving aircraft punctuality and respecting all safety regulations. The SESAR project is pushing for the introduction of automated support systems to help controllers in their jobs. However, the controllers still have to fine tune the outputs of the systems themselves due to their difficulty to take into account the more finer details of the aircraft movements.

In the literature, several studies propose the application of operations research tools to Air Traffic Flow Management (ATFM) related problems [Allahverdi et al., (2008), Ball et al., (2007), Pellegrini and Rodriguez (2013)]. These studies can be classified in the ATFM literature based on the the granularity of the problem taken into consideration, the type of information used and the algorithmic approaches utilized to solve the problem.

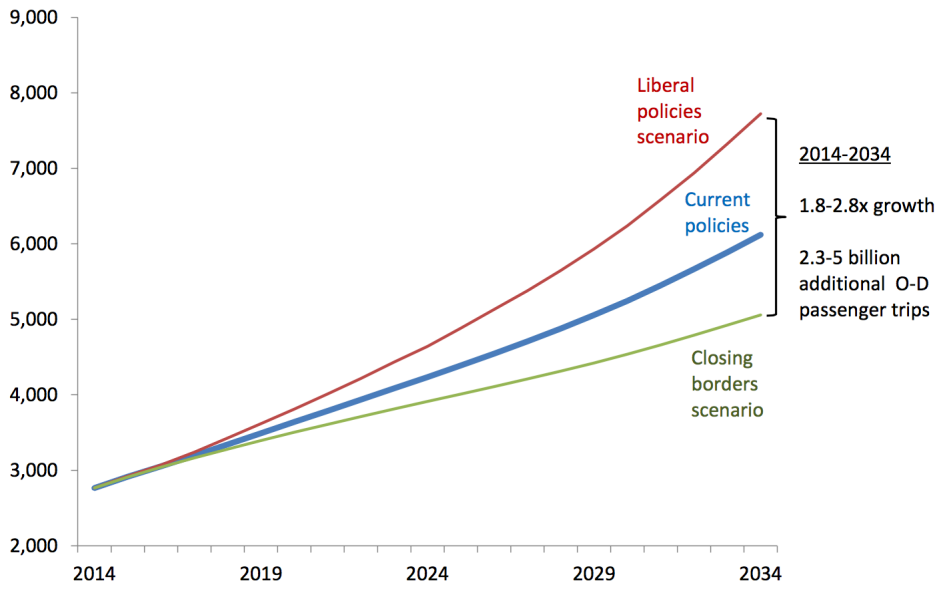


Figure 1.3: Outlook for worldwide origin-destination passenger trips, expressed in million (*Source:* [IATA (2014)])

Considering the granularity of the problem, two streams of research can be recognized: problems dealing with air traffic control between airports [Bertsimas et al., (2011a), Castelli et al., (2011)], and problems dealing with air traffic control in the TCA of a single airport [Bianco et al., (2006)]. When dealing with multiple airports the model used are often macroscopic, which drastically simplifies the characteristics of each single airport infrastructure, and flight path aggregation is usually implemented, making potential conflicts at runway level mostly undetectable. When dealing with a single airport instead, microscopic models are often considered in order to incorporate the finer information compliant with safety regulations, which instead allows the identification and resolution of potential aircraft conflicts even at runway level.

In this second stream different sub-categories can be recognized, due to the different aspect air traffic control has to deal with in a TCA. Some works focus only on the airport ground movements, examining: i) the ground delay program, applicable in case a severe reduction of landing capacity is forecast at a given airport, requiring to delay some of the expected aircraft before they depart from their origin airport [Ball et al., (2010), Kotynek and Richetta (2006)]; ii) the flight gate scheduling, in which aircraft serving a flight are assigned to distinct gates or aircraft stands in specific time windows [Dorndorf et al., (2007)], iii) the taxi planning, dealing with the aircraft routing and scheduling on the airport ground, considering the movements from the gates/stands to the runways and vice-versa [Clare and Richards (2011), Marin (2006)]. Other works focus instead on the optimization of air resources necessary for the landing and taking-off procedures,

considering only the runways [Balakrishnan and Chandran (2010), Soomer and Franx (2008)] or also the other airborne resources, such as holding circles and air segments [Artiouchine et al., (2008), Eun et al., (2010)]. In both cases, a diversification of the traffic studied may exists, such that arrival [Hu and Di Paolo (2009)] and departure schedules [Atkin et al., (2012)] are examined separately.

Considering the type of information used as input for the problem, two types are considered: static information, in which position and speed of all aircraft are known for the whole time horizon of traffic prediction at the beginning of the optimization phase [Ernst et al., (2004), Psaraftis (1980)] and dynamic information, in which every time a new incoming aircraft is detected it is required to recompute the aircraft schedule [Beasley et al., (2004), Hu and Chen (2005)].

Considering the algorithmic approaches, both heuristics and exact approaches have been used to solve ATFM related problems, minimizing a series of different objective functions due to the lack of a generally recognized performance indicator.

The problem tackled in this thesis deals with the optimization of resources necessary for both landing and taking-off procedures. The problem looks at the scheduling and re-routing of multiple aircraft in a single airport TCA when disturbances affect the normal course of daily operations. The model proposed has a microscopic granularity and it is based on the alternative graph [Mascis and Pacciarelli (2002)]. Algorithmic approaches are developed and included in the AGLIBRARY solver. The information utilized is mainly static information, with all data known before the optimization begins. However, it is also studied how the proposed approaches can be included in a dynamic system that iteratively solves the problem with static information.

1.4 Research objectives considered

The PhD thesis here presented investigates how operations research models and algorithms can bring benefits to real-time traffic control and management, looking specifically at the railway and air sector cases. It deals with and analyzes challenging instances, evaluating different large test cases in terms of network size, time of prediction assessed, number of available alternative routings, initial delays and disruptions considered. This PhD thesis follows the theses of [D'Ariano (2008), Corman (2010)] and has been carried out in a laboratory environment, extending or modifying the pre-existing modules or designing new components to be added to the whole system there introduced. A brief discussion on what this two theses have already tackled and developed is now presented.

D'Ariano applies the alternative graph model to the CDR problem with fixed train routings, developing a suite of rescheduling algorithms which includes heuristics and a truncated branch-and-bound. The branch-

and-bound initial solution is computed using some of the developed heuristics. The branching rule considers all the unselected alternative pairs and identifies the one to which the arc that, if selected, would create the highest delay belongs. The arc actually selected in the solution is its alternative one. The branch-and-bound stops if the optimal solution is found or if a previously set time limit is reached. Furthermore, D'Ariano's thesis analyzes train speed profiles, which can be considered fixed or can be dynamically computed using a train speed coordination procedure, then puts the basis for taking into consideration possible re-routing strategies for the CDR problem when flexible train routings are considered.

Corman enlarges D'Ariano's work, focusing on the management of difficult instances featuring highly-complex and large-sized networks, including a new degree of freedom in railway operations given by re-routing possibilities, typically considered to balance the use of critical resources. The CDR is tackled by a tabu search based meta-heuristic. The tabu search initial solution is computed using the previously described branch-and-bound. For the trains having at least one operation belonging to the critical path on the alternative graph of the initial solution, alternative routings that potentially lead to better quality solution are searched among a limited and precomputed set of possible candidates. Furthermore, a distributed approach to control large dispatching areas is proposed, which divides the whole area in smaller, local ones, each controlled by a single dispatcher. A coordination procedure is then proposed to help the coordination between pairs of local areas.

The PhD thesis here presented starts from these two works on the railway traffic management case, addressing some algorithmic issues left open. In particular, the thesis tackles the re-routing problem, considering possible different algorithmic schemes, evaluating the impact the number of available alternative routings has on the computation of a solution in a limited computation time.

Looking instead at the studied air traffic problem, some parallels may be drawn with the CDR. For example, both problems require a microscopic level of detail in order to incorporate constraints regarding safety regulations and to identify potential conflicts between multiple vehicles requiring the same resources. Furthermore, both may also take into consideration routing and scheduling flexibility. Due to the good performance obtained by the alternative graph model and the optimization algorithms on the railway case, this thesis attempts their application to the air traffic case, taking into consideration how the two problems possess quite different structures and thus carefully adapting both the model and the algorithms.

Here follows a short presentation of each Research Objective pursued in this work. The objectives can be devided in two categories, based on the case tackled. In particular, Research Objectives I–IV deal with the railway case, while Research Objectives V–VIII with the air traffic case.

Research Objective I: *Evaluate different search strategies and alternative solution methods that might out-*

perform the tabu search algorithm, quantifying the extend of the improvements based on the computation time required and the quality of the solution provided.

The Tabu Search (TS) makes extensive use of memory through a tabu list and implements aspiration criteria to guide the search and to avoid remaining trapped in local minima. Three routing neighbourhoods based on the (ramified) critical path on the graph of the current solution were investigated in [Corman (2010), Corman et al., (2010)], together with five neighbourhood strategies restricting the set of moves to be explored. These works demonstrated the great potential of including re-routing possibilities in the optimization and the TS there proposed is a state-of-the-art, high performing algorithm for the re-routing problem. However, the existence of alternative solution schemes with possible better performance were not investigated.

In the AGLIBRARY solver, the TS is used in a re-routing module to change the routing to a single train at each iteration, with the only exception of restart strategies, which randomly change the routings to multiple trains in order to escape a local minimum. The effect of changing multiple train routings at each step is analyzed, together with taking into consideration different neighbourhoods than the ones based on the critical path.

Thus, a series of algorithms based on the variable neighbourhood search metaheuristic schemes [Hansen et al., (2008)] are implemented, which consider routing neighbourhoods that differ from each other for the set of candidate trains to be re-routed in each move. The proposed algorithms are then evaluated over various real-world infrastructures, featuring different railway network characteristics and traffic flows but generally characterized by busy traffic, multiple delayed trains and temporarily disrupted railway resources. The performance obtained are compared with the ones of the TS algorithm and of a commercial solver. The algorithms proposed are able to improve both on the commercial solver and on the TS performance, with particular consideration on the time required to compute a good quality solution. However, the TS and the variable neighbourhood search based algorithms presents as stopping criteria the analysis of all possible routings combinations (rarely obtained) and the reaching of a previously set time limit.

Currently, if the value of the optimal solution is not previously known or if the lower bound for the problem has not been computed in parallel using a commercial solver, it is not possible to evaluate in AGLIBRARY the optimality gap of the solution found. This leads to the second research objective this thesis addresses.

Research Objective II: *Certify the quality of a solution found through the implementation and computation of an effective lower bound for the overall scheduling and routing problem.*

The computation of an effective lower bound for the job shop scheduling problem with flexible routings

modeled using the alternative graph, to the best of the author knowledge, has not yet been directly tackled. The train scheduling and routing problem is NP-hard. A big-M Mixed Linear Integer Programming (MILP) formulation of the problem can be obtained taking inspiration from the alternative graph model. Binary variables are introduced to model both routing and scheduling decisions. In particular, a binary variable is associated to each possible train routing and to each possible ordering decision. Among the alternatives only one routing per train can be selected, and for each alternative pair only one arc. Commercial solvers return poor quality lower bounds, since the solvers usually base their computation on a linear relaxation of the big-M MILP formulation.

The proposed lower bound computation begins with the construction of a dummy routing for each train. This dummy routing is built as follows: i) it includes the resources that are common among all alternative routings available to the train, i.e., the resources that are certainly going to be used by the train in its travel. If a train has one and only one routing available, the dummy routing corresponds to this routing, otherwise ii) between two (groups of) common resources it is inserted a fictitious resource. The travel time required by the train to travel this dummy resource is computed as the shortest time path among all possible alternatives considering the train running at free-net on it, i.e., in absence of conflicts. The alternative graph considering for each train its dummy routing is then built, representing the lower bound alternative graph. This lower bound graph differs from the instance alternative graph because it does not include alternative pairs for the fictitious resources and so no ordering decision are taken on them. The scheduling problem for a so built alternative graph is optimally solved and the value of the solution obtained is the lower bound for the overall scheduling and routing problem.

Furthermore, the lower bound thus computed is exploited in a constructive metaheuristic for the combined scheduling and routing problem. Starting from the lower bound solution, the constructive metaheuristic iteratively replaces the dummy routing with an actually available routing alternative. The routing included is evaluated using a fast heuristic that solves the alternative pairs added in the graph where actual resources have replaced the fictitious ones.

Research Objective III: *Investigate how the number of available alternative routings for each train affects the real-time solution process and how to select the best subset of routings for each train among these alternatives in order to simplify it.*

As previously said, the train scheduling and routing problem is NP-hard. Finding a good lower bound for the problem helps the solution process. However, the characteristics of the instancesconsidered, in the form of railway infrastructure and traffic flows, strongly affect the amount of time required to compute good quality solutions. The number of alternative routings for each train, when translated into variables and

constraints of the problem, appears to significantly enlarge the search space to examine. Obtaining a good quality solution in a reasonable amount of time appears to be an extremely difficult task to achieve. In fact, most of the solution approaches existing in the literature, instead of considering all possible available routings, limit the number of alternative routings for each train to a smaller subset, usually composed by the ones suggested by the dispatchers and/or similar to the one defined in the timetable. Thus, the problem solved is a simplified routing problem, with the issue that the routings proposed are often too general and that finding near-optimal solutions for a simplified problem does not necessarily correspond to finding near-optimal solutions for the original problem.

The real-time Train Routing Selection Problem (rtTRSP) is the problem of selecting, in a limited computation time, a routing subset for each train among all its possible alternatives without having to simultaneously consider timing and ordering decisions. Feasible combinations of train routings must exist and the routing subsets should lead to good quality solutions. The subsets obtained as result of the rtTRSP are then used during the rtRTMP solution process.

To stress how the difficulty to solve the rtRTMP when all routing alternatives are considered is a general issue present in the literature, it was decided to examine this Research Objective using, instead of AGLIBRARY, the RECIFE-MILP solver [Pellegrini et al., (2015)], developed at the Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux (IFSTTAR) in Lille-Villeneuve d'Ascq (France).

An integer linear programming formulation is proposed for the rtTRSP. The objective of the problem is finding a good quality routing subset for each train, where the quality of the subset is defined as a function of the interactions among the routings selected for the trains traveling the infrastructure. An algorithm based on Ant Colony Optimization [Dorigo and Stützle (2004)] is proposed to solve the rtTRSP.

Research Objective IV: *Due to the lack of a generally recognized objective function for the real-time railway traffic management problem, analyze the case in which more than one criterion is used to evaluate a proposed solution.*

In the literature there is no recognized objective function for the real-time railway traffic management problem. Typically, the objectives of any practitioner are to find a good schedule in a reasonably short amount of time while trying to avoid negative long-term effects of the rescheduling decisions. This has been translated into an array of different objective functions, each one valid since it considers a particular performance indicator that may interest dispatchers when performing real-time adjustments to an unfeasible timetable or that may focus on the maximization of the quality of service perceived by passengers.

This Research Objective starts from evaluating how different solutions, obtained by looking at different

performance indicators, can be compared to each others. Furthermore, the case in which the dispatchers consider interesting more than one performance indicator is evaluated. This problem is here addressed by developing a multi-criteria DSS to help dispatchers in taking more informed decisions. The scope of this Research Objective is developing a general and valid methodology. The most used objectives in the related literature have been selected as performance indicators. These indicators are not an exhaustive set of the whole range of practically relevant objectives but have been selected because prominent in the literature. Using different performance indicators would not require changes in the proposed methodology. Furthermore, to simplify the computation and the correlated discussion, the study is limited to the scheduling problem, with the train routings previously fixed as the one chosen in the tactical timetable.

To compare the best solutions obtained when looking at each performance indicator, it is necessary an automatic quantitative technique that selects the efficient ones and gives suggestions for improving the inefficient ones. This is achieved by Data Envelopment Analysis (DEA), a benchmarking, non parametric technique that assesses the relative efficiency of different solutions [Charnes et al., (1978)]. DEA defines an efficient frontier based on the relative best performance of the units compared (in our case the formulations differing from each others in the objective function). The frontier is made up of the units in the data set which are most efficient in transforming their inputs (computational resources) into outputs (optimization results). For each unit an efficiency score is computed based on its distances from the efficient frontier. Thus, a unit is defined as efficient if it lays on the efficient frontier. For each inefficient unit, an improvement target is individuated, based on the efficient units.

An iterative DEA-based procedure is proposed to select efficient solutions, leaving the final selection of the train schedule to the dispatchers.

Research Objective V: *Adapt the alternative graph model to the aircraft scheduling problem to include real-world constraints, increasing the level of accuracy of already existing models, and presenting a system which allows the solution of large time horizons of traffic prediction.*

In general, in this thesis, the aircraft scheduling problem in a TCA includes mixed-traffic runways, a degree of flexibility on scheduling decisions and, in some works, on routing decisions. Furthermore, not only runways but also airborne resources such as holding circles and air segments are modeled. A microscopic level of detail is thus required.

For example, for the holding circles the constraints should specify the maximum number of round trips an aircraft is allowed to perform, together with the required traveling times. Furthermore, air segments may host multiple aircraft simultaneously, but minimum longitudinal and diagonal distances between consecutive aircraft must be respected. This makes necessary accounting for aircraft sequencing in each air segment

and for feasible travel times required by the aircraft to traverse it. When the aircraft scheduling problem is modeled as job shop scheduling problem, the use of the alternative graph allows the required microscopic level of the detail. Specifically, the alternative graph can model any 4-dimensional routing for the aircraft in the TCA, while most of the works in the literature assume 3-dimensional routings are fixed and only optimize the timing of runway operations.

The existing algorithms in the AGLIBRARY solver are adapted to the problem. Compared to the railway case however, the routings available to each aircraft are quite limited in number. The size of the search space mostly depends on the number of aircraft considered, which in turn depends on the time horizon of traffic prediction examined. Thus, as a first step, the problem has been tackled looking only at the aircraft scheduling decisions in the TCA resources, considering the routings fixed under traffic regulation constraints.

A rolling horizon framework has been developed to restricting the search space. This framework allows a temporal decomposition of the problem. The overall time of prediction is divided in smaller time horizons. An optimization phase is required for each time horizon, and its feasible solution represents a possible scheduling plan. With respect to a time horizon just solved, the subsequent time horizon has a start time shifted a-head of a quantity of time called roll period. The roll period also represent the part of the scheduling plan fixed and to not recompute in the subsequent time horizon.

Such framework can be considered as a step toward a dynamic system since it allows to refresh the input data of every smaller time horizon solved. Thus, it can be used to manage dynamic information in a system that, at every stage, solves the problem using static information.

Research Objective VI: *Enlarge the previous Research Objective by including routing flexibility and investigate the potential benefit obtained even in presence of severe disturbances, including disrupted runways.*

Once proven the potentiality of the model and the solution framework applied to the aircraft scheduling problem, it is necessary to investigate their limits and possible improvement strategies, enlarging the problem to include alternative routings for each aircraft.

In general, every model reproduces a reality simplified. A too rich model in fact may need excessive computational resources, in terms of time and/or memory needed, requiring to find a trade-off between the reality-compliance of the model and its computational complexity. In order to evaluate possible limits of the alternative graph when applied to Air Traffic Control in a TCA, a number of model variants are proposed, each one focussed on different objective functions and user requirements. The model is also extended to include temporary unavailability of TCA resources. Furthermore, it is also analyzed the degree of flexibility added by including routing decisions, where an origin-destination routing for each aircraft has to be chosen regarding air segments and runway.

As in Research Objective II, an alternative graph inspired MILP formulation is proposed, which takes into consideration simultaneously scheduling and routing decisions. The MILP formulation is solved using commercial solver. The First In First Out rule (otherwise known as First Come First Served) is used as a surrogate for the dispatchers behaviour, together with various configuration of the AGLIBRARY solver for scheduling problem either with fixed or flexible routings.

The different solution approaches and the model variants are compared through a computational phase, including different disturbed cases and time horizons of traffic prediction, and evaluated in terms of computation time indicators, number of optimal solutions, number of constraint violations and aircraft delay minimization. The aim of the analysis is to show the benefits and limits of each model variation, the benefits of using routing flexibility and of ad hoc optimization-based approaches compared with the FIFO rule.

Research Objective VII: *Research algorithmic improvements in the AGLIBRARY solver to raise the possibility of finding quickly good quality solutions.*

This Research Objectives is an immediate follow-up of the previous one but also looks back at the results obtained with Research Objective I. On the one hand the Research Objective looks at the benefits of considering routing flexibility, on the other hand at the improvements over the TS obtained by implementing algorithms based on variable neighbourhood search schemes. In particular, the benefits of these new algorithms appeare to be time related.

To achieve an improvement in the quality of the solution found, the strengths of the tabu search and the variable neighbour search are combined. The variable neighbourhood search explores different neighbourhoods of the solution currently considered, allowing multiple routing modifications. The TS avoids remaining trapped in local minima and cycling during the solution process through the use of a list of tabu moves. An hybrid metaheuristic for solving the air traffic control problem in a TCA is implemented, named Variable Neighbourhood Tabu Search (VNTS). The VNTS mixes intensification strategies for the exact exploration of a restricted aircraft re-routing neighbourhood, diversification strategies for aircraft re-routing in different runway and air segment resources, and restart techniques based on a list of potentially useful routing alternatives for each aircraft.

An extensive computational phase is performed both to tune the parameters of the VNTS and to asses its performance, in terms of quality of the solution reached and computational time required, comparing the results with what has been obtained during the investigation of Research Objective VI.

Research Objective VIII: *Due to the lack of a generally recognized objective function for air traffic flow management problem, analyze the behaviour and the trade-off of literature proposed objective functions and*

then find a good compromise.

As seen in Research Objective IV for the railway case, also for the air traffic case there is no generally recognized performance indicator to assess its solutions. The quality of a solution in fact involves several performance indexes reflecting the interests of the different actors involved in air traffic management. Usually the solution process concentrates on the optimization of a single objective function, taking advantage of the fact that single objective optimization approaches are generally faster than multi-objective optimization ones. Defining an acceptable objective is thus very important, together with analyzing the influence its optimization has when the solution provided is also examined from a different point of view.

Differently than in Research Objective IV, this Research Objective deals with investigating the existence of gaps between optimal solutions when a single objective is optimized. As an initial analysis, only two functions are taken into consideration: the minimization of the largest delay due to potential aircraft conflicts, and the minimization of the total travel time spent in the TCA as a surrogate of energy consumption. However, these objectives have been selected because believed of particular practical interest. An analysis of a possible combination of the two in order to identify a reasonable balance is also performed.

1.5 Thesis outline

This thesis has been structured as a collection of articles, already published or submitted for publication. Each article shows how each of the Research Objectives previously presented has been approached. Looking at the topics tackled by the articles, four main areas of research can be recognized:

- **Algorithmic schemes:** this area deals with developing or improving algorithms in the AGLIBRARY solver. In particular, for the re-routing module of the solver a series of algorithms based on the variable neighbourhood search metaheuristic or on the hybridization of this metaheuristic with the tabu search are proposed. Furthermore, a lower bound and a heuristic based on its computation are also introduced in a new module for the solution of the overall scheduling and routing problem;

- **Search space restriction and decomposition:** in order to tackle large instances with large search spaces, characterized in particular by a high number of routing variables and vast time horizons of traffic prediction, this area investigates the possibility of restricting the search space by making an initial selection on the routing variables and by decomposing the traffic prediction in smaller time horizons;

- **Model Development:** this area deals with the application of the alternative graph to the air traffic control problem in a TCA. Possible model variants are also investigated;

Figure 1.4: Thesis outline

- **Objective function analysis:** lacking generally recognized objective functions both for the railway and the air traffic cases, the gaps between solutions optimized and evaluated with performance indicators prominent in the literature are analyzed. Ideas on how to deal with multiple performance indicators are also proposed.

Figure 1.4 provides a schematic outline of the articles. In general, the articles on the left column deal with the railway traffic management, while the articles on the right column with the air traffic flow management. Then, the articles are also grouped based on the four research areas described. Each chapter contains one article and has been organized has a self-contained unit, with its own abstract, introduction and conclusions.

Chapter 2 deals with Research Objective I. This chapter presents the comparative results for the optimization of the CDR when algorithms based on variable neighbourhood search or the TS are used.

Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D., (2016). A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers and Operations Research* Special Issue *"Variable Neighborhood Search methods in business and engineering"* doi:10.1016/j.cor.2016.02.008 **in press**.

Chapter 3 deals with Research Objective II. This chapter provides the procedure to compute a lower bound for the overall routing and scheduling problem when using the alternative graph. Furthermore, a fast heuristic that computes a feasible solution starting from the lower bound computation is presented.

Samà, M., D'Ariano, A., Pacciarelli, D., Corman, F. Lower and upper bound algorithms for the real-time train scheduling and routing problem in a railway network. Accepted for the preceeding of the *14-th IFAC Symposium on Control in Transportation Systems. 18–20 May, 2016 Istanbul, Turkey*, **to appear**.

Chapter 4 deals with Research Objective III. This chapter introduces the problem of selecting in real-time a subset of train routings (the rtTRSP) to pass to the rtRTMP. Furthermore, the chapter proposes an algorithm based on Ant Colony Optimization to solve the rtTRSP and evaluates its impact on the overall rtRTMP solution process, when the problem is modeled and solved using RECIFE-MILP [Pellegrini et al., (2015)].

Samà, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., Pacciarelli, D., (2016). Ant colony optimization for the real-time train routing selection problem. *Transportation Research Part B* , **85 (1)** 89–108.

Chapter 5 deals with Research Objective IV. This chapter proposes a multi-criteria decision support system to obtain and select solutions deemed efficient, relatively to the performance indicators taken into consideration for the train scheduling problem.

Samà, M., Meloni, C., D'Ariano, A., Corman, F., (2015). A multi-criteria decision support methodology for real-time train scheduling. *Journal of Rail Transport Planning & Management*, **5 (3)** 146–162.

Chapter 6 deals with Research Objective V. This chapter presents an alternative graph model applied to the aircraft scheduling problem in a TCA and a time decomposition based on a rolling horizon framework. A tuning phase for the selection of the best time horizon and roll period for the framework is included, together with the application of the framework in case of dynamic information.

Samà, M., D'Ariano, A., Pacciarelli, D., 2013. Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. *Transportation Research Part E* **60 (1)** 140–155.

Chapter 7 deals with Research Objective VI. This chapter adds to the model of the air scheduling problem flexibility on routing decisions and evaluates model variants. An extensive test analysis has been provided to evaluate the rolling horizon framework compared to a centralized solving approach, considering in both cases optimization tools and a possible dispatching rule.

Samà, M., D'Ariano, A., D'Ariano, P., Pacciarelli, D., (2014). Optimal aircraft scheduling and routing at a terminal control area during disturbances. *Transportation Research Part C* **47 (1)** 61–85.

Chapter 8 deals with Research Objective VII. This chapter is a follow-up of the previous one. The chapter provides a better solving approach in terms of quality of the solution provided and required computation time. An hybrid heuristic is proposed and evaluated both in the rolling horizon and in the centralized frameworks.

Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D. Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas. Submitted for publication.

Chapter 9 deals dealing with Research Objective VIII. This chapter presents an initial analysis on existing gaps between solutions computed with different objective functions, together with a possible combination strategy to find if a balance may exists.

Samà, M., D'Ariano, A., D'Ariano, P., Pacciarelli, D., (2015). Air Traffic Optimization Models for Aircraft Delay and Travel Time Minimization in Terminal Control Areas. *Public Transport* **7 (3)** 321–337.

Chapter 2

A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations

Abstract: *This chapter focuses on the development of metaheuristic algorithms for the real-time traffic management problem of scheduling and routing trains in complex and busy railway networks. This key optimization problem can be formulated as a mixed integer linear program. However, since the problem is strongly NP-hard, heuristic algorithms are typically adopted in practice to compute good quality solutions in a short computation time. This paper presents a number of algorithmic improvements implemented in the AGLIBRARY optimization solver in order to improve the possibility of finding good quality solutions quickly. The optimization solver manages trains at the microscopic level of block sections and at a precision of seconds. The solver outcome is a detailed conflict-free train schedule, being able to avoid deadlock situations and to minimize train delays. The proposed algorithmic framework starts from a good initial solution for the train scheduling problem with fixed routes, obtained via a truncated branch-and-bound algorithm. Variable neighbourhood search or tabu search algorithms are then applied to improve the solution by re-routing some trains. The neighbourhood of a solution is characterized by the set of candidate trains to be re-routed and the available routes. Computational experiments are performed on railway networks from different countries and various sources of disturbance. The new algorithms often outperform a state-of-the-art tabu search algorithm and a commercial solver in terms of reduced computation times and/or train delays.*

2.1 Introduction

In the last years, European railway companies are experiencing increasing difficulties to face the ever increasing transport demand while ensuring good quality of service to passengers, also due to the limited space and funds to build new infrastructure in bottleneck areas. These facts stimulated the interest for new effective Operations Research (OR) solutions for real-time train scheduling. This problem is faced by dispatchers, which have to modify orders, passing times and routes of trains (on-line train dispatching problem) in order to counter delays and keep traffic smooth.

In the train scheduling literature, there is a well-known difference between the level of sophistication of the theoretical results and algorithms and that of the methods that are employed in practice. While the theory typically address simplified problems, achieving optimal or near-optimal performance, the practice must face all the complexity of real-time operations, often with little attention to the performance level. This difference is especially evident for real-time scheduling, and train scheduling is not an exception. As a result, the poorly performing scheduling methods that are used in practice has a direct impact on the quality of service offered to the passengers, and the negative effects of disruptions on the regularity of railway traffic may last for hours after the end of the disruption [Kecman et al., (2013)]. However, there are recently many signals that the scheduling gap could be drastically reduced in the next few years. On the theoretical side, recent approaches to train scheduling tend to incorporate an increasing level of detail and realism in the models while keeping the computation time of the algorithms at an acceptable level. On the practical side, the railway industry is interested in assessing the suitability of these methods to the practical needs of real-time railway traffic management.

The design and implementation of advanced mathematical models is a pre-requisite to the development of innovative decision support systems for solving the on-line train dispatching problem. This paper is concerned with the modeling of the conflict detection and resolution (CDR) problem for railway networks. The CDR problem is the real-time problem of computing a conflict-free and deadlock-free schedule compatible with the actual status of the network and such that the circulating trains arrive and depart with the smallest possible delay. To solve the CDR problem, a number of algorithmic improvements are implemented in the AGLIBRARY solver, a set of OR-based models and algorithms for complex practical scheduling problems developed at Roma Tre University. This solver is the main solution engine of the ROMA dispatching support system [D'Ariano et al., (2008)], used for instance in the EU project ON-TIME [Corman and Quaglietta (2015)]. The solver is based on the alternative graph model introduced by [Mascis and Pacciarelli (2002)] and on the following framework: a good initial solution for the scheduling problem with fixed routes is computed by the (truncated) branch-and-bound algorithm in [D'Ariano et al., (2007a)]. Metaheuristics are

then applied to improve the solution by re-routing some trains. This action corresponds to the concept of a move, from a metaheuristics perspective. In [Corman et al., (2010)], a tabu search algorithm has been applied to solve practical-size railway instances for a Dutch test case in the Netherlands.

Previous research left open a number of relevant algorithmic issues. The first issue concerns the extent at which different search strategies and alternative solution methods might outperform the tabu search algorithm. A second issue is to quantify the algorithmic improvements, looking at a reduction of the computation time and at an improvement of solution quality. Both these issues motivate the development of the new metaheuristics proposed in this paper. The paper contributions are next outlined:

- We present routing neighbourhoods that differ from each other for the set of candidate trains to be re-routed in each move and for the available routing alternatives.

- We alternate the search for promising moves via systematic changes of a combination of neighbourhood structures, similarly to [Moreno Pérez et al., (2003)], and present strategies for searching within these neighbourhoods based on variable neighbourhood search schemes [Hansen et al., (2008)].

- We use fast train scheduling heuristics for the evaluation of each neighbour.

- We apply the proposed algorithms to the management of complex CDR problems, characterized by busy traffic, multiple delayed trains and temporarily disrupted railway resources. The new metaheuristics are compared with a state-of-the-art tabu search algorithm [Corman et al., (2010)] and with a commercial solver. Significantly better results are obtained in terms of a reduced time to compute the best-known (sometimes proven optimal) solutions, and for some CDR instances also in terms of an improved solution quality.

- We evaluate the algorithms over various real-world test cases, which feature different railway network characteristics and traffic flows.

Section 2.2 gives an overview of the literature related to the real-time railway traffic management. Section 2.3 formally defines the CDR problem and Section 2.4 presents mathematical formulations for this problem. Section 2.5 describes the algorithms of AGLIBRARY and the new metaheuristics proposed in this paper. Section 2.6 reports on the performance of the algorithms on various practical case studies from Italy, the Netherlands and UK. Section 2.7 summarizes the main paper findings and outlines future research directions. An appendix illustrates the neighbourhoods investigated in this work with a numerical example.

2.2 Literature review

The study of real-time train scheduling and routing problems received increasing attention in the literature in the last years. Early approaches (starting from the pioneering work of [Szpigel (1973)]) tend to solve very simplified problems that ignore the constraints of railway signalling, and that are only applicable for specific traffic situations or network configurations (e.g. a single line or a single junction), see the literature reviews in the following papers: [Ahuja et al., (2005), Cacchiani et al., (2014), Cordeau et al., (1998), Fang et al., (2015), Hansen and Pachl (2014), Lusby et al., (2013), Meng and Zhou (2011), Pellegrini and Rodriguez (2013), Pellegrini et al., (2014), Törnquist and Persson (2007)]. Among the reasons for this gap between early theoretical works and practical needs are the inherent complexity of the real-time process and the strict time limits for taking and implementing decisions, which leave small margins to a computerized Decision Support System (DSS).

Effective DSSs must be able to provide the dispatcher with a conflict-free disposition schedule, which assigns a travel path and a start time to each train movement inside the considered time horizon and, additionally, minimizes the delays (and possibly the main broken connections) that could occur in the network. The main pre-requisite of a good DSS is the real-time ability to deal with actual traffic conditions and safety rules for practical networks. In other words, the solution provided by a DSS must be feasible in practice, since the human dispatcher may have not enough time to check and eventually adjust the schedule suggested by the DSS. A recognized approach to represent the feasibility of a railway schedule is provided by the blocking time theory, acknowledged as standard capacity estimation method by UIC in 2004 [Hansen and Pachl (2014)], which represents a safe sequence of train movements in the railway network with the so-called blocking time stairways.

With the blocking time theory approach, the schedule of a train is individually feasible if a blocking time stairway is provided for it, starting from its current position and leaving each station (or each other relevant point in the network) not before the departure time prescribed by the timetable. A set of individually feasible blocking time stairways (one for each train) is globally feasible if no two blocking time stairways overlap. The timetable prescribes the set of trains that are expected to travel in the network within a certain time window, the stops for each train and a pair of (arrival, departure) times for each train and each stop. At other relevant points (e.g. at the exit from the network or specific relevant points between two consecutive stations) can be defined minimum and/or maximum pass through times.

Many models and algorithms for train re-scheduling have already been proposed in the literature, but only a few of them with successful application in practice. So far, the most successful attempt in the literature to incorporate the blocking time theory in an optimization model is based on the alternative graph model

introduced by [Mascis and Pacciarelli (2002)]. This model is a generalization of the disjunctive graph for job shop scheduling, in which each operation denotes the traversal of a resource of the network by a job (train). Effective applications to real-time train scheduling are described in [D'Ariano et al., (2007a), Mannino and Mascis (2009), Mazzarello and Ottaviani (2007)]. However, other promising approaches have been provided in the literature, either based on mathematical formulations [Cadarso and Marín (2014), Caimi et al., (2012), Lamorgese and Mannino (2013), Pellegrini et al., (2014), Rodriguez (2007), Şahin (1999), Törnquist and Persson (2007), Wegele et al., (2007)] or on algorithmic approaches [Almodóvar and García-Ródenas (2013), Cai and Goh (1994), Cheng (1998), Chiu et al., (2002), Liu and Kozan (2009), Törnquist (2012), Wegele and Schnieder (2004)]. Another important aspect when dealing with rail operations is the passenger behaviour [Cadarso et al., (2013), Corman et al., (2015a), Dollevoet et al., (2014), Kroon et al., (2010)], even if this latter aspect is not considered explicitly in this paper.

The alternative graph model allows to directly model the individual and global train schedule feasibility concepts expressed by the blocking time theory. This enables the detailed recognition of timetable conflicts in a general railway network with mixed traffic for a given look-ahead horizon, even in presence of heavy disturbances and network disruptions. Several later studies have confirmed the ability of the model to take into account different practical needs, such as train priorities [Corman et al., (2011a)], energy consumption issues [Corman et al., (2009a)], passenger and rolling stock transfer connections [Corman et al., (2012b), D'Ariano et al., (2008)], train re-routing [Corman et al., (2010), D'Ariano et al., (2008)], management of complex and busy stations [Corman et al., (2011b), Corman et al., (2009b)], traffic coordination between dispatching areas [Corman et al., (2012a), Corman et al., (2014a)]. Clearly, an alternative graph model of the CDR problem can be easily translated into a mixed integer program, and then solved with a commercial or academic software. However, the CDR problem is inherently strongly NP-hard, so it is often not possible to compute a feasible solution quickly via any commercial software for practical-size CDR instances. Such NP-hard problems are typically solved via heuristic algorithms, that enable the computation of good quality solutions in a computation time compatible with real-time operations.

This paper presents a number of algorithmic improvements implemented in the AGLIBRARY optimization solver in order to improve the possibility of finding good quality solutions quickly. A set of specialized algorithms based on the alternative graph model is included in AGLIBRARY. This re-scheduling system includes solution algorithms ranging from fast heuristic procedures that can be chosen by the user to sophisticated branch-and-bound algorithms for train scheduling [D'Ariano et al., (2007a)] or metaheuristics for train re-routing [Corman et al., (2010), Corman et al., (2014a), D'Ariano et al., (2008)]. AGLIBRARY has been tested on various railway networks managed by the Dutch infrastructure manager ProRail (the railway networks Leiden-Schiphol-Amsterdam; Utrecht-Den Bosch; Utrecht-Den Bosch-Nijmegen-Arnhem), by the

British infrastructure manager NetworkRail (part of the east coast mainline nearby London) and by the Italian infrastructure manager RFI (the regional line Campoleone-Nettuno), even if in principle the software can tackle any national or international traffic management system standard. AGLIBRARY has also been used in other application contexts, including steelmaking-continuous casting production scheduling [Pacciarelli and Pranzo (2004)], real-time air traffic scheduling and routing at a terminal control area [D'Ariano et al., (2015), D'Ariano et al., (2012b), Samà et al., (2014), Samà et al., (2013a)], real-time management of containers at a container terminal [Corman et al., (2015b), Xin et al., (2015)].

The new AGLIBRARY algorithms proposed in this paper can be potentially applied to improve the results obtained both in railway traffic management and in the other application contexts. However, the algorithmic structures and parameters would need to be customized for each particular railway network and test case from other application fields. This work is focused on the customization and application of the new algorithms to solve the CDR problem in various railway networks, including issues related to the management of complex station areas, connection constraints, train re-scheduling and re-routing variables. The new algorithms are compared with previously-published algorithms and with a commercial solver.

2.3 Problem definition

Signals, interlocking and Automatic Train Protection (ATP) systems control the train traffic by imposing safety constraints between trains, setting up train routes and enforcing speed restrictions on running trains. Fixed block ATP systems ensure safety through the concept of *block section*, a part of the infrastructure that is exclusively assigned to at most one train at a time. Train movements can be modeled by a set of characteristic times, as follows. The *running time* of a train on a block section starts when its head (the first axle) enters the block section and ends when the head of the train reaches the end of the block section.

Safety regulations impose a minimum separation between consecutive trains traveling on the same block section, which translates into a minimum *headway time* between the start of the running times of two consecutive trains on the same block section. This time depends on the length of the block section, as well as on other factors like the speed and length of the trains and includes the time between the entrance of the train head in a block section and the exit of its tail (the last axle) from the previous one, plus additional time margins to release the occupied block section and to take into account the sighting distance.

Proactive re-scheduling of railway traffic must take into account several facts. The network is composed by block sections and platform stops at stations. A train is not allowed to depart from a platform stop before its scheduled departure time and is considered late if arriving at the platform later than its scheduled arrival time. At a platform stop, the scheduled stopping time of each train is called *dwell time*.

Disturbances affect rail traffic. We can distinguish between light traffic *perturbations* from neighbouring dispatching areas and heavy traffic *disruptions*. The former are light disturbances caused by a set of delayed trains in a dispatching area, while the latter are much stronger disturbances of the scheduled times and routes (e.g. due to some block sections being unavailable for a certain amount of time). Other kinds of disturbances include extensions to dwell times due to passengers boarding, connection constraints, or technical problems; and running time prolongation because of headway conflicts between trains or technical failures.

Moreover, the railway infrastructure is increasingly becoming utilised, towards a saturation level. This results in a strong sensitivity to initial delays, which are due to breakdowns, failures, extended dwell time at stations due to passengers; those phenomena are almost unavoidable. In saturated networks, those initial delays are particularly hard to be managed, and easily generate knock-on (or consecutive) delays which spread over the network in time and space, affecting more trains.

Delays propagate between trains when solving potential conflicting routes. Namely, a *potential conflict* between two trains arises if the trains request a same block section within a time interval smaller than the minimum time headway between them, which is needed for safety reasons and smooth running. The solution of the potential conflict is to fix the order of trains over the block section; in that case, one of the approaching trains might be forced to decelerate and thus experiencing a knock-on delay. Unscheduled braking and stopping of trains increases the running time and may cause an additional delay. Similarly, trains can be held at stations due to unavailability of outbound routes, or conversely prevented to enter stations as far as platforms and inbound routes are not available. In general, delays may propagate to other trains causing a domino effect of increasing traffic disturbances.

The conflict detection and resolution (CDR) problem studied in this paper can be defined as follows: given a railway network, a set of train routes and passing/stopping times at each relevant point in the network, and the position and speed of each train being known at a given starting time t_0 of traffic prediction, find an optimized plan of operations that solves all potential conflicts between trains, does not result in deadlock situations (i.e. a set of trains that are circularly waiting for each other, making any planned movement impossible), it is compatible with initial positions of trains, and such that the selected train timing, ordering and routing decisions are feasible, no train appears in the network before its expected entrance time (including the initial delays), no train departs from a relevant point before its scheduled departure time, and trains arrive at the relevant points with the smallest possible consecutive delay.

2.4 Problem formulation

The CDR problem is characterized by train routing and scheduling decisions. The general problem can thus be divided into two sub-problems: (i) the selection of a route for each train, and (ii) the scheduling decisions once the routes have been fixed for each train. This section describes the problem decomposition in scheduling and routing variables. The general alternative graph model is given for the CDR problem with fixed routes. A Mixed Integer Linear Programming (MILP) formulation is proposed for the alternative graph model. An extended MILP formulation is then given for the overall CDR problem. Binary variables are introduced both for the train scheduling and routing decisions. An illustrative example of the alternative graph model is reported in the appendix of this paper.

2.4.1 Alternative graph model

The alternative graph (AG) generalizes the classical disjunctive graph in order to take into account constraints arising in real-world scheduling applications. Regarding the CDR problem, AG allows to easily and efficiently check the feasibility of a train scheduling solution (all operations are to be processed with no deadlock and conflict situations), as well as the quality of a train scheduling solution (the maximum consecutive delay collected at each relevant location). The CDR problem is based on fixed and alternative constraints.

Fixed constraints model the individual feasibility of a train schedule, i.e. the blocking time stairway. A timing variable is associated to the entrance of each train in each resource (block section, platform of station route). The schedule is individually feasible if the entrance in a resource is at least a sufficient amount of time after the entrance in the former resource, respecting all the safety and operational constraints. Assuming that the route of a train has been fixed, if at the current time the train occupies a certain resource, it cannot enter the next resource in its route before the time needed to traverse the remaining part of the current resource. Since the timetable prescribes a departure (or a pass through) time for the train at each relevant point in its route, the train cannot enter the next resource of the route before a minimum prescribed time.

Alternative constraints model the global feasibility of a set of blocking time stairways (one for each circulating train). Given a resource traversed by two trains, the second train cannot enter the resource before the entrance time of the previous train plus its blocking time, i.e. the time interval in which the resource is reserved for the first train. If a precedence constraint has not been fixed between the two trains on that resource (either by the timetable, or the dispatcher, or the physical network topology), then two orderings are possible and one of them has to be chosen in a train scheduling solution. This fact is represented in the alternative graph by a pair of alternative constraints, one of which must be chosen in a solution.

The AG formulation of the CDR problem with fixed routes (i.e. in which the route is prescribed and cannot be changed) is a triple $G = (N, F, A)$ where $N = \{0, 1, ..., n-1, n\}$ is a set of $n+1$ nodes, F is a set of *fixed* directed arcs and A a set of pairs of *alternative* directed arcs.

Each node, except the start 0 and end n nodes, is associated with the start of an operation krj, where k indicates the train, r the route chosen and j the resource it traverses. The start time t_{krj} of operation krj is the entrance time of train k with route r in resource j.

The fixed arcs are used to model running, dwell, connection, arrival, departure, and pass through times of trains. Let the resources p and j be two consecutive resources processed by train k with route r, the fixed arc $(krp, krj) \in F$ models a job constraint between the nodes krp and krj. The weight $w^F_{krp\_krj}$ represents a minimum time constraint between t_{krp} and t_{krj}: $t_{krj} - t_{krp} \geq w^F_{krp\_krj}$. A fixed arc $(umv, krz) \in F$ is used to enforce a connection constraint between train k with route r and train u with route m, i.e. $t_{krz} - t_{umv} \geq w^F_{umv\_krz}$.

The alternative arcs are used to model the headway times between two consecutive trains. Each pair of alternative arcs $((krj, ump), (umi, krp)) \in A$ models train ordering decisions between train k with route r and train u with route m on resource p. Note that j [respectively i] is the next resource processed by train k [u] when using route r [m]. The two arcs of the pair are associated with the weights $w^A_{krj\_ump}$ and $w^A_{umi\_krp}$. In any solution, only one arc of each pair can be selected. If alternative arc (krj, ump) [(umi, krp)] is selected in a solution, the constraint $t_{ump} - t_{krj} \geq w^A_{krj\_ump}$ [$t_{krp} - t_{umi} \geq w^A_{umi\_krp}$] has to be satisfied. This corresponds to fixing the order of trains, first k and then u [first u and then k].

A solution to the CDR problem with fixed routes is represented by the following graph structure. A graph selection S is a set of alternative arcs obtained by selecting exactly one arc from each alternative pair in A and such that the resulting graph $\mathcal{G}(F, S) = (N, F \cup S)$ does not contain positive weight cycles. This allows to associate train orders and times to all operations.

The objective function is the minimization of the maximum consecutive delay, i.e. the largest positive deviation from the scheduled times at relevant locations. In the alternative graph, the maximum consecutive delay minimization is measured as a makespan minimization. Given a selection S and any two nodes krp and uml, we let $l^S(krp, uml)$ be the weight of the longest path from krp to uml in $\mathcal{G}(F, S)$. By definition, the start time t_{krp} of $krp \in N$ is the quantity $l^S(0, krp)$, which implies $t_0 = 0$ and $t_n = l^S(0, n)$.

To summarize, the alternative graph model corresponds to the following mathematical formulation:

$$\begin{array}{lll} \min & t_n & \\ s.t. & t_{krj} - t_{krp} \geq w^F_{krp\_krj} & (krp, krj) \in F \\ & t_{krz} - t_{umv} \geq w^F_{umv\_krz} & (umv, krz) \in F \\ & (t_{ump} - t_{krj} \geq w^A_{krj\_ump}) \vee (t_{krp} - t_{umi} \geq w^A_{umi\_krp}) & ((krj, ump), (umi, krp)) \in A \end{array} \tag{2.1}$$

2.4.2 MILP formulations

A natural mathematical formulation of the CDR problem with fixed routes can be obtained from the alternative graph formulation (2.1) by translating each alternative pair into a pair of constraints and by introducing a binary variable representing the choice of one of the two constraints. The CDR problem with fixed routes can be viewed as a particular *disjunctive program*.

$$\begin{array}{lll} \min & t_n & \\ s.t. & t_{krj} - t_{krp} \geq w^F_{krp\_krj} & (krp, krj) \in F \\ & t_{krz} - t_{umv} \geq w^F_{umv\_krz} & (umv, krz) \in F \\ & t_{ump} - t_{krj} \geq w^A_{krj\_ump} + Mx_{(krj,ump),(umi,krp)} & ((krj, ump), (umi, krp)) \in A \\ & (t_{krp} - t_{umi} \geq w^A_{umi\_krp} + M(1 - x_{(krj,ump),(umi,krp)}) & ((krj, ump), (umi, krp)) \in A \\ & x_{(krj,ump),(umi,krp)} \in \{0, 1\} & \end{array} \tag{2.2}$$

The variables are the following: $|N|$ real variables t_{krj} associated to the start time of each operation $krj \in N$, and $|A|$ binary variables $x_{(krj,ump),(umi,krp)}$ associated to each alternative pair $((krj, ump), (umi, krp)) \in A$. The constant M is a sufficiently large number, e.g. the sum of all arc weights.

We model the variables and constraints of the CDR problem for the different routes of each train as follows. The formulation (2.2) can be extended to the problem with routing flexibility by enlarging sets N, F and A to contain all possible train routes. In addition to the $|N| + |A|$ variables of the CDR problem, $|C|$ binary variables y are associated to the routes of the set of trains considered. The CDR problem with routing flexibility can also be viewed as a particular *disjunctive program*.

In the CDR problem formulation (2.3), Z is the number of trains, and R_b the number of routes for each train $b = 1, ..., Z$. The binary variable y_{ab} indicates if route a is chosen (1) or not (0) for train b. For each train b, only a single route, among the R_b routes, can be chosen in any CDR solution. The following constraint holds for train b: $\sum_{a=1}^{R_b} y_{ab} = 1$.

When a route r is chosen for train k (i.e. $y_{kr} = 1$), each fixed constraint related to route r and train k must be satisfied. For each fixed arc $((krp, krj)) \in F$, $t_{krj} - t_{krp} \geq w^F_{krp\_krj}$ must hold. A fixed arc

$(umv, krz) \in F$ enforces a connection constraint between train k with route r and train u with route m.

$$
\begin{aligned}
&\min\ t_n \\
&s.t. \\
&t_{krj} - t_{krp} \geq w^F_{krp\_krj} + M(1 - y_{kr}) && (krp, krj) \in F \\
&t_{krz} - t_{umv} \geq w^F_{umv\_krz} + M(2 - y_{um} - y_{kr}) && (umv, krz) \in F \\
&t_{ump} - t_{krj} \geq w^A_{krj\_ump} + M(2 - y_{um} - y_{kr}) + M x_{(krj,ump),(umi,krp)} && ((krj, ump), (umi, krp)) \in A \\
&t_{krp} - t_{umi} \geq w^A_{umi\_krp} + M(2 - y_{um} - y_{kr}) + M(1 - x_{(krj,ump),(umi,krp)}) && ((krj, ump), (umi, krp)) \in A \\
&\textstyle\sum_{a=1}^{R_b} y_{ab} = 1 && b = 1, ..., Z \\
&y_{ab} \in \{0, 1\} \\
&x_{(krj,ump),(umi,krp)} \in \{0, 1\}
\end{aligned}
\tag{2.3}
$$

Regarding the alternative constraints, if $y_{um} = y_{kr} = 1$ and the routes of trains u and k use the same infrastructure resource p, a potential conflict exists on that resource and an ordering decision has to be taken. This is modelled by introducing the binary variable $x_{(krj,ump),(umi,krp)}$ for the alternative pair $((krj, ump), (umi, krp)) \in A$. There are two possible scheduling decisions for each alternative pair $((krj, ump), (umi, krp)) \in A$: if $x_{(krj,ump),(umi,krp)} = 1$ then $t_{krp} - t_{umi} \geq w^A_{umi\_krp}$ must be satisfied (i.e. $(umi, krp) \in S$); if $x_{(krj,ump),(umi,krp)} = 0$ then $t_{ump} - t_{krj} \geq w^A_{krj\_ump}$ must be satisfied (i.e. $(krj, ump) \in S$).

2.5 Train scheduling and re-routing algorithms

This section describes the algorithmic approaches proposed in this paper to solve the CDR problem. Section 2.5.1 presents the general framework of the AGLIBRARY solver that is based on a combination of train scheduling and re-routing algorithms. Section 2.5.2 presents the algorithms used to compute a train schedule for given routes. Section 2.5.3 describes the neighbourhoods for the search of new train routes starting from a routing and scheduling solution. Section 2.5.4 is devoted to the scheduling heuristic procedures used to evaluate the neighbours (the new routing combinations). The routing neighbourhoods and the scheduling algorithms are then used in Sections 2.5.5 and 2.5.6 that describe the former and new re-routing metaheuristics of the AGLIBRARY solver.

2.5.1 Solution framework

Figure 2.1 shows the architecture of the AGLIBRARY solver. The input data are given via an XML file, defining the timetable of scheduled arrival and departure times, the current status of the infrastructure

components (block sections and platforms), the running time of each train, an off-line defined default route and a set of re-routing options for each train, and a set of disturbances (initial delays and, eventually, disruptions). Given the input data, the AGLIBRARY solver iterates between the computation of a train schedule for a given set of routes, and the selection of a new set of routes. The basic idea is to first compute a train scheduling solution given fixed routes, and then search for better train routes. The solver solution is provided with another XML file, describing the CDR solution in terms of train orders and routes.

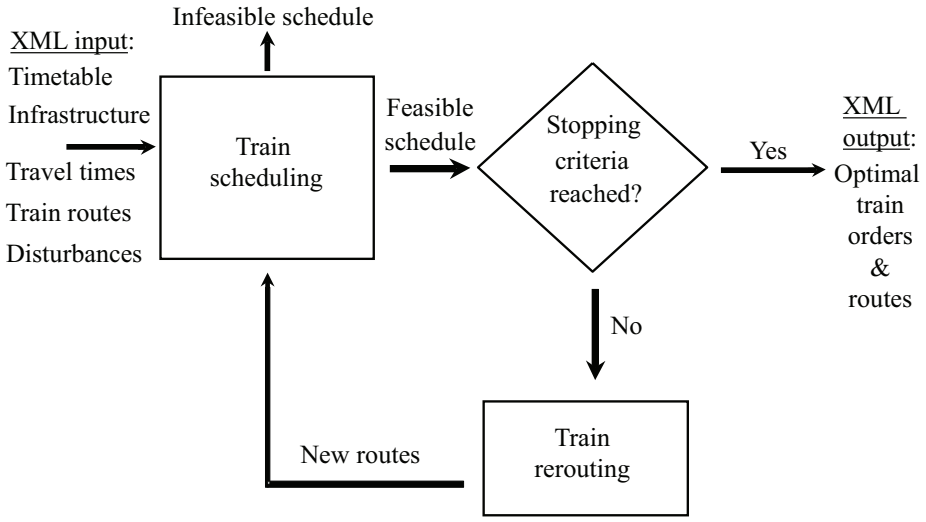


Figure 2.1: Architecture of the AGLIBRARY solver

If no feasible train schedule is found in a given computation time, the human dispatcher must recover infeasibility manually by taking some decisions that are not allowed to the solver, e.g. the cancelation of a train service. When a feasible train schedule is found, the train re-routing module verifies whether a new set of routes, leading to a potentially better solution, exists or not. Whenever re-routing is performed, the train scheduling module computes a new schedule. The iterative procedure continues till a stopping criterion is met and returns the best CDR solution. We next describe the algorithms we use for each module.

2.5.2 Branch-and-bound scheduling algorithm

The CDR problem with fixed routes is solved by the branch-and-bound (BB) algorithm of [D'Ariano et al., (2007a)], truncated at a given maximum computation time. A near-optimal solution is computed in a short time by this algorithm for practical-size instances. In particular, the algorithm is based on a binary branching scheme in which the branching decision is a sequencing order between two trains in a resource. In the alternative graph, this sequencing decision corresponds to the selection of an alternative arc from each

pair $((krj, ump), (umi, krp)) \in A$. The branching decision is thus on the arcs (krj, ump) and (umi, krp).

2.5.3 Routing neighbourhoods

Metaheuristic algorithms are generic solution procedures based on exploring the solution space by means of considering an incumbent solution and iteratively changing it in favor of a new incumbent solution. This action corresponds to the concept of a *move* from a solution to a possibly better one, and it is in general guided by some approximation or evaluation of the objective value, and/or properties of the solution. Commonly more tentative solutions are considered, and a single one is chosen as incumbent. The *neighbourhood* describes the moves that will be considered, based on a certain incumbent solution.

This subsection describes the neighbourhood structures used by the CDR algorithms presented in this paper. To this aim, we need to introduce the following notations. Let $S(F)$ be a CDR solution with the routes defined in F and the sequencing decisions defined in S, and let $\mathcal{G}(F, S)$ be the graph of this solution. The search for a better solution is based on the computation of a new graph $\mathcal{G}'(F', S')$. This graph differs from the former $\mathcal{G}(F, S)$ by a different route for some trains, and different orders and times of operations. This corresponds to a neighbour, in metaheuristics terms. The longest path in $\mathcal{G}'(F', S')$ is denoted as $l^{S'(F')}(0, n)$. We observe that F' improves over F in terms of the objective function value if $l^{S'(F')}(0, n) < l^{S(F)}(0, n)$.

The neighbourhoods studied in this paper are based on observations on the graph $\mathcal{G}(F, S)$ regarding the nodes that represent train operations delayed due to the resolution of potential conflicts between trains. These nodes are *critical* when they are on the longest path from the start node 0 to the end node n in the graph $\mathcal{G}(F, S)$, that is called the *critical path set* $\mathcal{C}(F, S)$. Given a solution $S(F)$, $krp \in N(F) \setminus \{0, n\}$ is a *critical node* of train k with route r if $l^{S(F)}(0, krp) + l^{S(F)}(krp, n) = l^{S(F)}(0, n)$. A critical node krp is a *waiting node* if $l^{S(F)}(0, krp) > l^{S(F)}(0, \nu(krp)) + w^F_{\nu(krp),krp}$, where the node $\nu(krp)$ precedes the node krp on route r. For each waiting node krp, there is at least one *hindering node* $\eta(krp)$ in $\mathcal{G}(F, S)$, different from node $\nu(krp)$, such that $l^{S(F)}(0, krp) = l^{S(F)}(0, \eta(krp)) + w^F_{\eta(krp),krp}$.

Given a node $krp \in N(F) \setminus \{0, n\}$, we recursively define the *backward ramification* $R_B(krp)$ as follows. If node krp is waiting, then $R_B(krp) = R_B(\nu(krp)) \cup R_B(\eta(krp) \cup \{krp\}$, otherwise $R_B(krp) = R_B(\nu(krp)) \cup \{krp\}$. Similarly, we recursively define the *forward ramification* $R_F(krp)$ as follows. If node krp is the hindering of a waiting node abc, then $R_F(krp) = R_F(\sigma(krp)) \cup R_F(abc) \cup \{krp\}$, where the node $\sigma(krp)$ follows the node krp on route r. Otherwise, $R_F(krp) = R_F(\sigma(krp)) \cup \{krp\}$. By definition, $R_B(0) = R_F(0) = \{0\}$ and $R_B(n) = R_F(n) = \{n\}$. Given $\mathcal{C}(F, S)$, we define a *ramified critical path* set as $\mathcal{F}(F, S) = \bigcup_{krp \in \mathcal{C}(F,S)}[R_B(krp) \cup R_F(krp)]$, and a *backward ramified critical path* set as $\mathcal{B}(F, S) = \bigcup_{krp \in \mathcal{C}(F,S)}[R_B(krp)]$. We study the following five neighbourhood structures.

- *Complete K-Route neighbourhood* $\mathcal{N}_{CKR}$ contains all the feasible solutions to the CDR problem in which K trains follow a different route compared to the incumbent solution. To limit the number of neighbours to be evaluated, $\mathcal{N}_{CKR}$ is only partially explored as follows. A move is obtained by choosing different routes from the ones of the current solution at random (i.e. all alternative routes having the same probability) for K trains, until a number ψ (parameter) of alternative routing solutions is obtained.

- *Ramified Critical Path Operations neighbourhood* $\mathcal{N}_{RCPO}$ considers only the routing alternatives for the trains associated to the nodes in $\mathcal{B}(F,S)$ plus $\mathcal{F}(F,S)$. The idea is that the maximum consecutive delay of a solution to the CDR problem can be reduced by removing some train conflicts causing it. This requires either removing, anticipating or postponing some train operations from the ramified critical path set. The latter result can be obtained by re-routing the trains associated with the ramified critical path operations through different resources (i.e. by re-routing some trains associated to the nodes in $\mathcal{B}(F,S)$ or $\mathcal{F}(F,S)$), and then re-scheduling train movements.

- *Waiting Operations Critical Path neighbourhood* $\mathcal{N}_{WOCP}$ is a restriction of $\mathcal{N}_{RCPO}$ that considers the routing alternatives for the trains associated to the waiting nodes in $\mathcal{C}(F,S)$.

- *Delayed Jobs neighbourhood* $\mathcal{N}_{DJ}$ considers only the trains (jobs) that have a consecutive delay at some relevant locations on the incumbent solution.

- *Free-Net Waiting Operations Jobs neighbourhood* $\mathcal{N}_{FNWJ}$ considers only the trains (jobs) that have some waiting nodes in the graph of the incumbent solution in which all alternative arcs are unselected. The alternative graph with no alternative arc selected corresponds to the *free-net traffic situation* in which each train travels in the absence of conflicts.

The appendix of this paper will illustrate some neighbourhood structures for an illustrative example.

2.5.4 Heuristic evaluation of routing neighbours

The choice of a best neighbour in the neighbourhood requires the computation of a new CDR solution $S'(F')$ starting from an incumbent solution $S(F)$, that is characterized by the train routing decisions in F' and the train sequencing decisions in S'. To this aim, we use fast heuristics based on a two-step graph building procedure in which the graph $\mathcal{G}(F,S)$ is translated into the graph $\mathcal{G}'(F',S')$. In the first step, a sub-graph of $\mathcal{G}'(F',S')$ is generated by considering all the nodes in $N(F^I)$ associated to the routes modelled by the arcs in $F^I = F \bigcap F'$, all the fixed arcs $\in F^I$ and all the alternative arcs in $S(F)$ incident in nodes in $N(F^I)$. This corresponds to keeping a subset of decisions from the incumbent solution into the neighbour solution.

In the second step, the fixed arcs in $F^R = F' \setminus F^I$ and the nodes in $N(F^R)$ are added to the sub-graph. Finally, $\mathcal{G}'(F', S')$ is obtained by adding a selection of alternative arcs $S'(F^R)$ to the sub-graph.

The selection $S'(F^R)$ is computed by selecting the best solution among two greedy algorithms based on the idea of repeatedly enlarging a selection by choosing an unselected pair at a time from the set A and by selecting one of the two arcs until a feasible schedule is found or an infeasibility (i.e. a positive weight cycle in the graph) is detected [Pranzo et al., (2003)]. The first greedy algorithm AMSP (*Avoid Most Similar Pair*) chooses an unselected alternative pair $((krj, ump), (umi, krp)) \in A$ maximizing the quantity $l^{S'(F^R)}(0, krj) + w^A_{krj,ump} + l^{S'(F^R)}(ump, n) + l^{S'(F^R)}(0, umi) + w^A_{umi,krp} + l^{S'(F^R)}(krp, n)$. The other greedy algorithm AMCC (*Avoid Most Critical Completion Time*) chooses the alternative pair $((krj, ump), (umi, krp)) \in A$ such that the quantity $l^{S'(F^R)}(0, krj) + w^A_{krj,ump} + l^{S'(F^R)}(ump, n)$ is maximum among all the unselected alternative arcs. Both algorithms select the arc of the pair causing the minimum consecutive delay.

2.5.5 Tabu search re-routing algorithm

The *Tabu Search* (TS) is a deterministic metaheuristic based on local search, which makes extensive use of memory for guiding the search. A basic ingredient is the *tabu list*, that is used to avoid being trapped in local optima and revisiting the same solution. From the incumbent solution, non-tabu moves define a set of solutions, named the *incumbent solution neighbourhood*. At each step, the best solution in this set is chosen as the new incumbent solution. Some attributes of the former incumbent are then stored in the tabu list. The moves in the tabu list are forbidden as long as these are in the list, unless an aspiration criterion is satisfied. The tabu list length can remain constant or be dynamically modified during the search.

The Tabu Search (TS) used in this paper for the CDR problem is the algorithm of [Corman et al., (2010)]. Two neighbourhood strategies for the maximum consecutive delay minimization problem are investigated named *Restart* and *Complete*. Each neighbourhood strategy restricts the set of moves to be explored in order to speed up the search of the best move. In particular, the Complete strategy explores ψ (parameter) randomly chosen neighbours in $\mathcal{N}_{CKR}$ with $K = 1$, while the Restart strategy selects at most ψ promising moves in $\mathcal{N}_{RCPO}$, unless this neighbourhood is empty. When no potentially better solution is found on the incumbent solution neighbourhood, the search alternates the neighbourhood strategy with a diversification strategy, which consists of changing at random the route of μ (parameter) trains at the same time.

In this paper, all neighbours are evaluated via the scheduling heuristics of Section 2.5.4. The best neighbour is set as the move to be made, and re-evaluated via the branch-and-bound scheduling algorithm of Section 2.5.2; the resulting best CDR solution is set as the new incumbent solution. The inverse of the chosen move is stored in a tabu list of length λ (parameter). The moves in the tabu list are forbidden for λ iterations and no aspiration criteria is used. From the tuning performed in [Corman et al., (2010)], the best

exploration strategies have the following parameters: for the Complete strategy the best values of (ψ, λ, μ) are (8; 3; 5); for the Restart strategy the best values of (ψ, λ, μ) are (8; 27; 5).

2.5.6 Variable neighbourhood search re-routing algorithms

Variable Neighbourhood Search (VNS) metaheuristics are presented in order to efficiently solve the CDR problem. This type of metaheuristic is based on the combination of neighbourhood structures. Systematic changes of the neighbourhood structures are proposed both in a local search phase in order to compute a local minimum, and in a perturbation phase in order to escape from a local minimum [Hansen et al., (2008)].

In this work, the classic ingredients of the VNS algorithm are combined with new routing neighbourhood structures and new specific neighbourhood search strategies to search for better train routing combinations. A move is obtained by choosing a set of routes different from the ones of the incumbent solution. Various variable neighbourhood schemes from [Hansen et al., (2008)] and routing neighbourhoods of Section 2.5.3 are implemented, differing in the set of candidate trains that are re-routed in each move.

The choice of investigating these search methods is motivated by the following facts: (i) the train scheduling solution with fixed routes can be improved in terms of multiple train routing modifications, (ii) there is a need of improving upon the local minima found by local search. From (i), we need to explore possibilities to generate new solutions starting from some reference solutions. From (ii), we need to develop strategies to spread the search for better quality solutions. For these reasons, we investigate VNS intensification and diversification strategies. The main algorithmic ingredients are next introduced and optionally incorporated in various versions of the variable neighbourhood search algorithm.

Build Neighbourhood. Starting from an incumbent solution, the $\mathcal{N}_{CKR}$ neighbourhood is generated, in which exactly K trains are re-routed in the graph $\mathcal{G}(F, S)$ of the incumbent solution.

Shaking procedure. This is a typical diversification procedure that consists in changing the route of K trains randomly in the $\mathcal{N}_{CKR}$ neighbourhood of the incumbent solution ($IncSol$), and in computing a new incumbent solution ($IncSol'$) via the scheduling heuristics of Section 2.5.4 and the new set of routes.

Neighbourhood Search Strategy. This procedure is proposed in order to limit the local search to the evaluation of up to L neighbours in the current neighbourhood. Starting from an incumbent solution and the $\mathcal{N}_{CKR}$ neighbourhood of this solution, a restricted neighbourhood is generated by using a given neighbourhood structure $\mathcal{N}_i$. The selection of L neighbours is achieved in the following steps:

1. *train ranking* : Each train gets a score based on the criterion specified in $\mathcal{N}_i$. The train ranking is based

on one of the neighbourhood structures of Section 2.5.3. In $\mathcal{N}_{RCPO}$, each train on the ramified critical path gets a score based on the maximum value $l^{S'(F^R)}(0, krp) + l^{S'(F^R)}(krp, n) \; \forall \; (krp)$ in the ramified critical path of the graph of the incumbent solution. In $\mathcal{N}_{WOCP}$, each train gets a score based on the sum of the consecutive delays collected at each critical node in the graph of the incumbent solution. In $\mathcal{N}_{DJ}$, each train gets a score based on the maximum consecutive delay collected at some relevant locations for each job. In $\mathcal{N}_{FNWJ}$, each train gets a score based on the sum of the consecutive delays collected at each waiting node in the graph of the incumbent routing solution in which all alternative arcs are unselected (i.e. free-net traffic situation) but the one generating the waiting node. The scores are used to decide how many times each train has to be re-routed in the L neighbours.

2. *route ranking* : The routes of each train get a score based on the distance from the route of the incumbent solution. The larger is the difference between the routes, the higher is the score. The route ranking thus suggests for each train to select the most different routes.

3. *neighbour generation* : This is the assignment of the routes to the trains in each neighbour. The trains to be re-routed are selected via the train ranking and the train routes are selected via the route ranking. A combinatorial combination of the routes is used in order to generate L different neighbours. In each neighbour, exactly K trains are re-routed compared to the incumbent solution. The neighbours are ordered based on the train ranking, and in case of tie on the route ranking.

Numerical example regarding the Neighbourhood Search Strategy. Figure 2.2 presents a numerical example of the neighbourhood search strategy, in which four trains ($J = \{A, B, C, D\}$) can be re-routed in a railway network. Trains A and D have a default route and three alternative routes, while trains B and C have a default route and five alternative routes. As an example, the routes of B are named $B1$, $B2$, $B3$, $B4$, $B5$, $B6$, with the first one $B1$ being the default route. Detailed information regarding the traffic flows and the example network is reported in the paper appendix.

In the given incumbent CDR solution, the default route is used by all trains (i.e. the routes $A1, B1, C1, D1$). The parameters of the procedure are set to the following values: $L = 4$ and $K = 2$ (i.e. the neighbourhood is restricted to 4 neighbours and 2 trains are re-routed in each neighbour).

The train ranking procedure determines a score matrix in which each row represents a train and each column reports a score used to compute the number of times each train should be considered for re-routing. This score matrix is depicted in the left-hand side of Figure 2.2. Specifically, the first column reports the score for each train based on the neighbourhood structure $\mathcal{N}_{DJ}$ (e.g. the value 14 in this example is the maximum consecutive delay collected for $A1$, see Appendix). The other columns report the score of the first

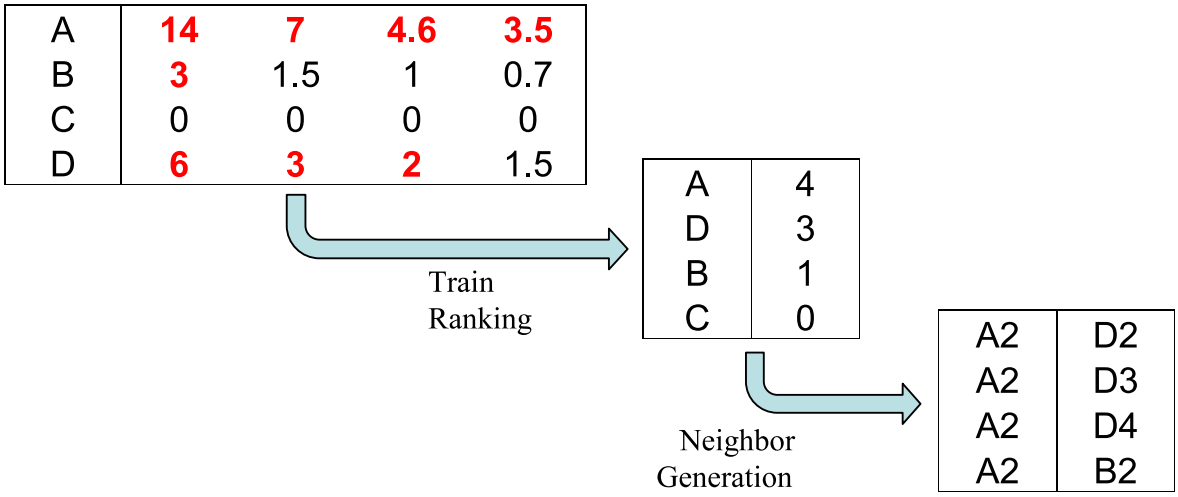


Figure 2.2: Example of neighbourhood search strategy with $|J| = 4$, $L = 4$ and $K = 2$

column divided by the number of column, e.g. $14/2 = 7$, $14/3 = 4.6$, $14/4 = 3.5$.

The procedure takes the highest KL scores that are provided in the score matrix (the scores in bold in Figure 2.2). In other words, from the largest values in the score matrix we obtain the number of times each train has to be re-routed in the four neighbours (e.g. train A will be re-routed in all four neighbours). This information is reported in the center table of Figure 2.2, in which the four trains are ordered according to the number of times they will be re-routed.

The route ranking orders the list of re-routing alternatives for each train based on maximizing the difference with the incumbent route, in terms of the number of different operations between the train routes. As an example, the route ranking of A is $A2$, $A3$, $A4$ and the most different route is $A2$; the route ranking of D is $D2$, $D3$, $D4$ and the most different route is $D2$; the most different route for train B is $B2$.

The neighbour generation procedure assigns the routes to the trains in each neighbour. In this example, train A appears in all the neighbours with one of its alternative routes; trains D and B appear with respectively three routes and one route. The neighbours are obtained as follows. We first re-route the two trains with higher train ranking (A and D) and then re-route the remaining train (B) with the train with the highest train ranking (A). Among the neighbours with the same re-routed trains, we first consider the current most different route for the train with the current higher train ranking, and then consider the current most different route for the next train with the current higher train ranking, and so on. Every row of the table in right-hand side of Figure 2.2 corresponds to a candidate move.

Best Improvement Strategy. This is a local search procedure in the restricted neighbourhood in

which all the candidate moves are evaluated via the train scheduling heuristics. This procedure lasts until all L neighbours in the restricted neighbourhood have been evaluated. At each step of the procedure, a neighbour is considered, a new graph is built with the new routes of the neighbour, and a new CDR solution is computed via the scheduling heuristics for the new set of routes. The best neighbour is set as the move to be made, and re-evaluated via the branch-and-bound scheduling algorithm of [D'Ariano et al., (2007a)]; the resulting best CDR solution is set as the new incumbent solution.

Move Or Not Procedure. This procedure is responsible for possibly performing a move. In case the best solution found in the neighbourhood is better than the incumbent, the resulting train scheduling problem is solved by the branch-and-bound algorithm of [D'Ariano et al., (2007a)], and the best solution is set as the new incumbent solution. Otherwise, the best solution in the neighbourhood is chosen as incumbent, or some diversification strategy is employed depending on the adopted VNS scheme.

Neighbourhood Change. This procedure is used to diversify the search by alternating K_{max} applications of the neighbourhood search strategy with $\mathcal{N}_1$ and K_{max} applications of the neighbourhood search strategy with $\mathcal{N}_2$.

The metaheuristics proposed in this paper are an adaptation of the VNS schemes described in [Hansen et al., (2008)]. Specifically, we consider the four algorithmic schemes named "VND", "General VNS", "Basic VNS", "Reduced VNS". The general structure of the studied metaheuristics is the following. The metaheuristics start from an incumbent solution $IncSol$ of the CDR problem, computed via the branch-and-bound scheduling algorithm of [D'Ariano et al., (2007a)] by assigning a default (off-line) route to each train. In each metaheuristic, a counter K is adopted to fix the number of trains that are re-routed in each move. The initial value of K is set to 1, i.e. a single train is re-routed in $IncSol$. The metaheuristics iterate the search for better solutions starting from $IncSol$ until a stopping criteria is reached.

The stopping criterion of all the metaheuristics is a maximum computation time T_{max} or the particular situation in which the maximum consecutive delay (i.e. the best objective function value $f(IncSol^*)$) is equal to 0. Additionally, VND stops the search when no improvement is obtained during a local search.

At each iteration, a neighbourhood of $IncSol$ is generated and a new solution $IncSol'$ is selected. Then, the *Move Or Not* function is performed as follows. In case an improving move $IncSol'$ is obtained via the local search (i.e. $f(IncSol') < f(IncSol)$), a new iteration is performed by setting $IncSol'$ as the new incumbent solution and K is set to 1. Otherwise, the parameter K is set to $K + 1$ and a new iteration is performed until $K \leq K_{max}$. When $K = K_{max}$ the algorithm diversifies the search with a change of neighbourhood structure (if the algorithm works with a single neighbourhood structure this step is not performed).

The general pseudo-code of the variable neighbourhood search algorithms is reported in Figure 2.3. Each metaheuristic returns the best CDR solution ($IncSol^*$) and the objective function value ($f(IncSol^*)$).

Algorithm Variable Neighbourhood Search
Input: $IncSol$ $\mathcal{G}(F,S)$, K_{max}, T_{max}, L, $\mathcal{N}_1$, $\mathcal{N}_2$
$\mathcal{N}_i \leftarrow \mathcal{N}_1$,
While $(T < T_{max})$ & $(f(IncSol) > 0)$ & $(OtherStoppingCriterion)$ **do**
  Begin
  $K \leftarrow 1$,
  While $(K \leq K_{max})$ **do**
    Begin
    $BuildNeighbourhood(IncSol, K)$,
    $IncSol' \leftarrow GenerateNewIncumbentSolution(IncSol, K, L, \mathcal{N}_i)$,
    $(IncSol, K) \leftarrow MoveOrNot(IncSol, IncSol', K)$,
    If $(K = K_{max})$ **do**
      Begin
      $\mathcal{N}_i \leftarrow NeighbourhoodChange(IncSol, \mathcal{N}_i, \mathcal{N}_1, \mathcal{N}_2)$,
      End
    $T \leftarrow$ CPU time()
    End
  End

Figure 2.3: General sketch of the metaheuristics

The iterative step of the metaheuristics studied in this paper differs in the choice of $IncSol'$:

- *VND*: This is a deterministic version of variable neighbourhood search in which a local search is performed in a restricted neighbourhood of $IncSol$, via the neighbourhood search strategy (for a given value of parameters L and $\mathcal{N}_i$) and the best improvement strategy. The best neighbour is set as $IncSol'$;

- *General VNS*: This is a generalization of the VND in which the neighbourhood of $IncSol$ is generated and a solution $IncSol''$ is selected via the shaking procedure (for a given value of parameter K), that takes a neighbour of $IncSol$ at random. The VND algorithm is applied starting from the solution $IncSol''$. The resulting solution is $IncSol'$;

- *Basic VNS*: This version of VNS combines the deterministic and random changes of neighbourhoods. This algorithm first performs the shaking procedure (for a given value of parameter K) on the incumbent solution $IncSol$, obtaining a solution $IncSol''$. Then, a local search is performed in a restricted neighbourhood of $IncSol''$, via the neighbourhood search strategy (for a given value of parameter L and neighbourhood structure $\mathcal{N}_i$) and the best improvement strategy. The best neighbour is set as $IncSol'$;

- *Reduced VNS*: This is a completely randomized version of the VNS metaheuristic. The shaking procedure (for a given value of parameter K) is performed starting from the incumbent solution $IncSol$. The resulting solution is set as $IncSol'$. This VNS can be viewed as a reduction of Basic VNS in which the local search is not performed. The rationale is to look at a larger number of neighbours compared to Basic VNS, even if these are randomly selected. However, we note that this metaheuristic scheme has a high risk of re-evaluating the same solutions several times, and thus being trapped into a local optimum.

2.6 Computational experiments

This section presents the experimental results on the TS, VND and VNS metaheuristics of Section 2.5.

Four practical railway test cases are investigated in a laboratory environment:

- an Italian single-track network (named "Italian (I) test case");
- a Dutch double-track network between Utrecht and Den Bosch (named "First Dutch (FD) test case");
- a Dutch busy and complex area around Utrecht central station (named "Second Dutch (SD) test case");
- a British double-track network nearby the city of London (named "British (B) test case").

All the studied test cases are modelled with a microscopic level of detail, which means that switches, signals, block sections, and track segments in complex station areas are considered (yielding several hundreds of resources per test case). Furthermore, train movements are described with a precision of seconds.

For each test case, we consider a set of 20 traffic disturbance instances, varying the initial delays of trains. The experiments are executed on a workstation Power Mac with processor Intel Xeon E5 quad-core (3.7 GHz), 12 GB of RAM. The algorithms are implemented in AGLIBRARY and use a total computation time $T_{max} = 180$ seconds. The MILP formulation of the CDR problem is solved by using the commercial solver: IBM LOG CPLEX MIP 12.0, that is executed with a time limit of 2 hours.

Table 2.1 presents the parameters studied for the variable neighbourhood search algorithms. Regarding the TS algorithm, the parameters are set as described in Section 2.5.5 (according to the parameter tuning in [Corman et al., (2010)]), expect for the BB computation time limit that is set as for the VND/VNS algorithms. In Table 2.1, the assessment of the new algorithms is based on a set of pilot CDR instances with $T_{max} = 180$ sec.

Regarding the information provided in Table 2.1, Column 1 reports the maximum computation time (in seconds), Column 2 the computation time given to the branch-and-bound scheduling algorithm ($BB\ Time$,

in seconds), Column 3 the number of trains that are rerouted in the current neighbourhood (K_{max}), Column 4 the size of the restricted neighbourhood (L). The best value of each parameter is reported in bold.

Table 2.1: Experimental setting of the algorithmic parameters

T_{max}	$BB\ Time$ (sec)	K_{max}	L
180	2/**4**/8/12	3/4/**5**/7	5/**10**/15/20

Table 2.2 presents the best configuration of each CDR algorithm for each test case. For the TS algorithm we report the best search strategy, while for the VND and VNS algorithms we report the best combination of the neighbourhood structures. In Table 2.2, the assessment of the TS, VND and VNS algorithms is performed with $T_{max} = 180$ sec, and is based on the set 20 traffic disturbance instances for each test case. We used the following two criteria in lexicographic order of importance to establish the best algorithmic configuration: 1) the average value of the objective function of the CDR problem; 2) the average time required to find the best CDR solution. Figure 2.4 presents these two key performance indicators that are used to evaluate logically the performance of the algorithms. For each test case we show the two best search strategies for TS, the best combinations of the neighbourhood structures for VND and VNS. Specifically, the best neighbourhood structures are: WOCP, DJ for the I test case, DJ+WOCP, WOCP+FNWJ, DJ+FNWJ, FNWJ for the FD test case, DJ, WOCP+DJ, FNWJ+DJ for the SD test case, WOCP, FNWJ, DJ for the B test case.

Table 2.2: Best configurations of the CDR algorithms for each test case

Test Case	Best TS	Best VND	Best VNS General	Best VNS Basic	Best VNS Reduced
Italian (I)	Complete	DJ	WOCP	WOCP	WOCP
First Dutch (FD)	Restart	WOCP+FNWJ	FNWJ	DJ+WOCP	DJ+FNWJ
Second Dutch (SD)	Complete	DJ	WOCP+DJ	DJ	FNWJ+DJ
British (B)	Restart	DJ	WOCP	WOCP	FNWJ

In the next subsections, we use the best configuration of each algorithm as indicated in Tables 2.1 and 2.2. Specifically, we present a detailed assessment of the computational results obtained for each CDR instance by the best TS, VND and VNS algorithms and by the commercial MILP solver. The computational experiments are reported per test case, together with additional information on the tested CDR instances. We conclude the section with some general discussion and observation on the obtained results.

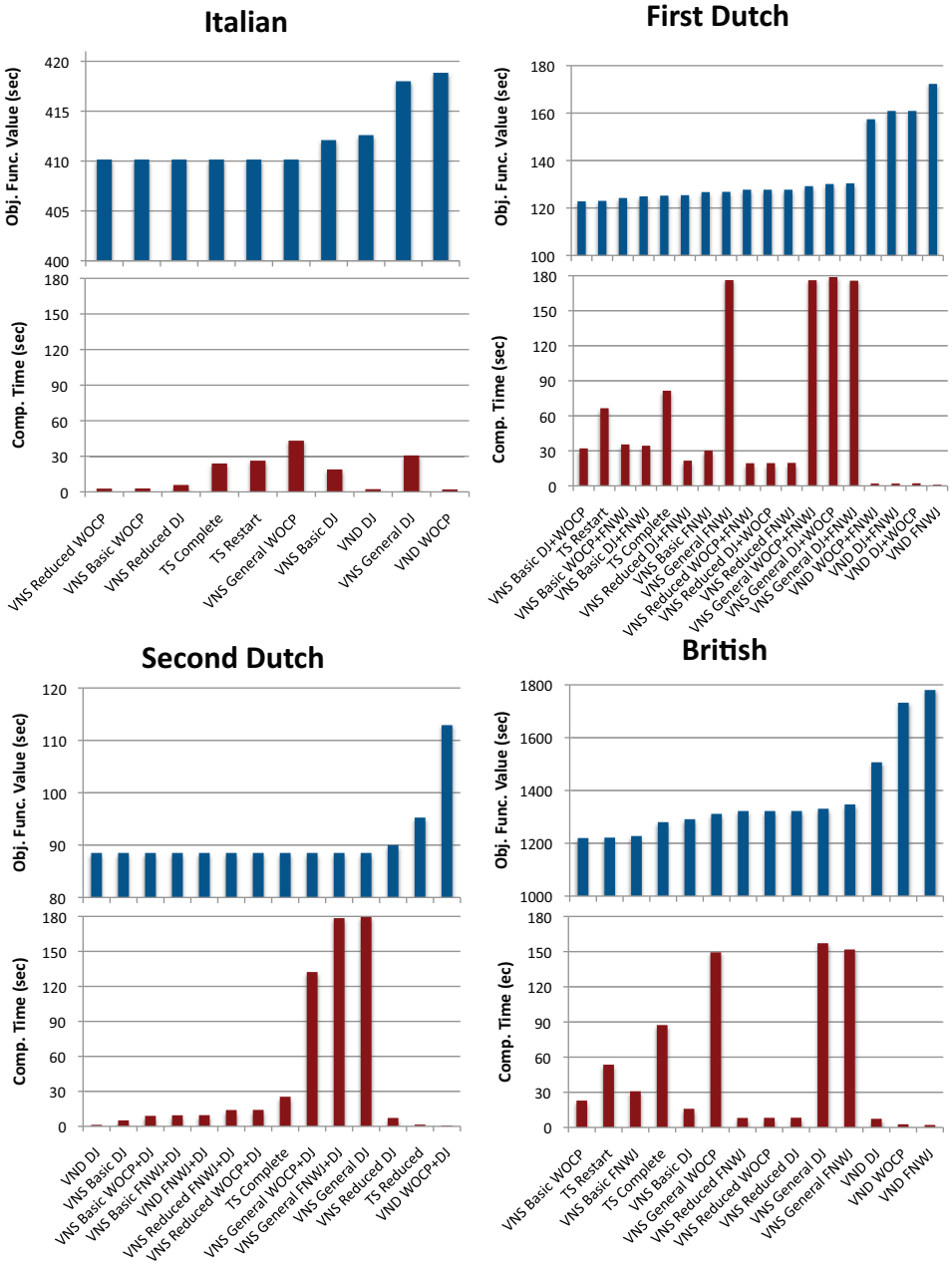


Figure 2.4: Quantitative comparison between configurations of the CDR algorithms.

2.6.1 Results on the Italian test case

Figure 2.5 shows a schematic view of the dispatching area of Campoleone-Nettuno, (i.e. the regional line FR8) with the amount of tracks per branch, and stations or minor stops. The railway network consists of 10 stations and 9 bidirectional single-track segments between stations. Most stations have two parallel platform tracks to allow re-routing and meet-pass operations. The railway infrastructure is around 26 km long. There are potential conflict points (where the dispatcher takes ordering and routing decisions) at the entrance and exit of each station. In Campoleone, there is a connection with the regional line FR7 and with the line towards Roma Termini station, the main station in Rome.

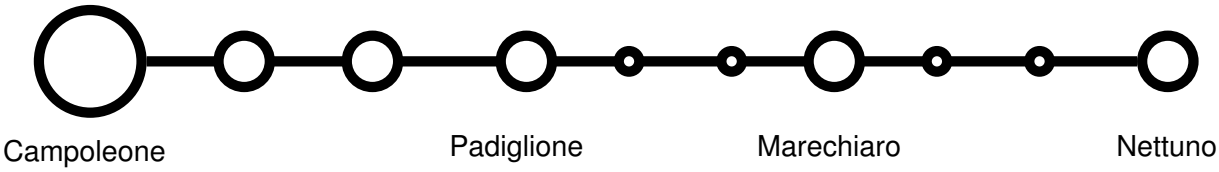


Figure 2.5: Italian test case - the Campoleone Nettuno line.

We consider a daily timetable which describes the movement of all trains running in the line during day hours, specifying, for each train, planned arrival/passing times at each station platform along its route. At stations, a train is not allowed to depart from a platform stop before its scheduled departure time and is considered late if arriving at the platform after its scheduled arrival time. The current daily timetable has 42 trains, with an average travel time for passengers of 42 minutes. Traffic disturbances are studied in which a set of trains is delayed at their entrance in the network.

Table 2.3 reports on the CDR instances of the Italian test case. Column 1 reports the time horizon of traffic prediction (in minutes), Column 2 the approximate length of the railway network (in kilometers), Column 3 the number of trains in the network during the entire horizon of traffic prediction, Column 4 the average number of routes assigned to each train (including the default route), Column 4 the average number of resources traversed by each train, Columns 5–7 the average size of the MILP formulation in terms of the number of timing variables (i.e. the set $|N|$), the number of train scheduling variables (i.e. the set $|A|$) and the number of train routing variables (i.e. the set $|C|$).

Table 2.3: Characteristics of the Italian test case instances

Time Horizon (min)	Network Length (km)	Num Trains	Num Routes Per Train	Num Resources Per Train	MILP Variables $\|N\|$	$\|A\|$	$\|C\|$
720	26	42	8	18	1094	752307	336

This single-track railway network presents several alternative routings, since each train can be routed in

several stations, where two parallel platform tracks are available.

Table 2.4 gives the computational results for 20 CDR instances of the Italian test case. In Column 1, each CDR instance is identified by a three-field code [α_β_γ], in which α identifies the network, β is the maximum initial delay (in seconds), and γ is the average initial delay (in seconds). We recall that the initial delay is caused by disturbances and cannot be recovered by re-scheduling train movements, except by using the available time reserves in the timetable. In the other columns, the performance of each algorithm/solver is presented in terms of the objective function value (i.e. the maximum consecutive delay, in seconds) and the time to compute the best solution (in seconds). The best average objective function values are reported in bold. For each CDR algorithm, we only present the results obtained for the best configuration of Table 2.2.

Table 2.4: Results obtained for the Italian (I) test case instances

CDR Instance	TS		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)
I_4000_986.3	237	0.6	237	0.6	237	89.3	237	0.6	237	4.6	79390	364.2
I_3864_965.6	262	0.6	262	0.6	262	0.6	262	0.6	262	0.6	458	7189.2
I_3883_983.3	270	0.6	270	0.6	270	0.6	270	0.6	270	0.6	83848	1042.1
I_4000_981.4	249	0.6	249	0.6	249	0.7	249	0.6	249	0.6	56463	6826.8
I_3976_1077.8	276	0.6	276	0.6	276	0.6	276	0.6	276	0.6	83471	1644.9
I_3646_949.3	261	0.6	261	0.6	261	164.6	261	0.6	261	0.6	62933	6023.6
I_4000_979.8	300	0.6	300	0.6	300	0.6	300	0.6	300	0.6	80187	4833.1
I_4000_971.8	297	0.6	297	0.6	297	0.6	297	0.6	297	0.6	84282	1325.7
I_3981_981.9	385	1.0	385	1.0	385	1.0	385	1.0	385	1.0	396	5537.1
I_10000_2618.3	436	7.2	436	13.9	436	129.1	436	3.8	436	5.9	64808	6828.3
I_10000_2629.8	605	23.2	605	0.7	605	0.6	605	0.6	605	0.7	76458	5547.3
I_9553_2579.1	539	67.6	539	7.7	539	9.0	539	4.6	539	8.0	83781	1175.5
I_10000_2688	577	135.2	626	3.8	577	174.4	577	19.9	577	7.9	51731	6457.6
I_10000_2600.1	467	0.6	467	0.7	467	7.6	467	0.6	467	0.7	66631	6059.3
I_9739_2689.4	345	0.6	345	0.7	345	0.6	345	0.6	345	0.7	55803	6552.6
I_9008_2504.5	787	33.8	787	0.6	787	0.6	787	0.6	787	0.7	59847	5822.8
I_9489_2607.9	638	7.1	638	3.6	638	147.4	638	2.8	638	13.9	84785	900.9
I_10000_2615.3	486	151.4	486	6.4	486	133.4	486	16.4	486	7.7	84002	1343.1
I_10000_2618.3	448	1.3	448	1.3	448	1.2	448	1.2	448	1.3	448	6456.1
I_10000_2584.3	338	0.7	338	0.7	338	0.6	338	0.6	338	0.8	344	7190.1
Avg Results	**410.2**	21.7	412.6	2.3	**410.2**	43.1	**410.2**	2.9	**410.2**	2.9	58003.3	4456.0

From the results of Table 2.4, VNS Basic and VNS Reduced are the best algorithms in terms of both indicators, while the other algorithms either present a larger delay or a larger computation time. VND is often the fastest algorithm but it does not provide the best-known solutions, while VNS Reduced uses VND combined with the shaking procedure and is able to compute the best-known solutions. TS and VNS General are, on average, much slower than VNS Basic and VNS Reduced. CPLEX is always outperformed by the CDR algorithms in terms of both indicators, expect for instance I_10000_2618.3. Furthermore, the lover bound returned by CPLEX is very low and does not certify the optimality of any CDR solution. For this set of CDR instances the low quality of CPLEX is probably due to the fact that the single-track train scheduling problem presents several infeasible train timing and ordering solutions, resulting in deadlock situations.

2.6.2 Results on the first Dutch test case

Figure 2.6 presents the Utrecht - Den Bosch dispatching area, that consists of 191 block sections and 21 platforms. The railway network is around 50 km long, connecting the cities of Utrecht and Den Bosch. There are two main tracks, a long corridor for each traffic direction, a dedicated stop for freight trains nearby Zaltbommel and 7 passenger stations.

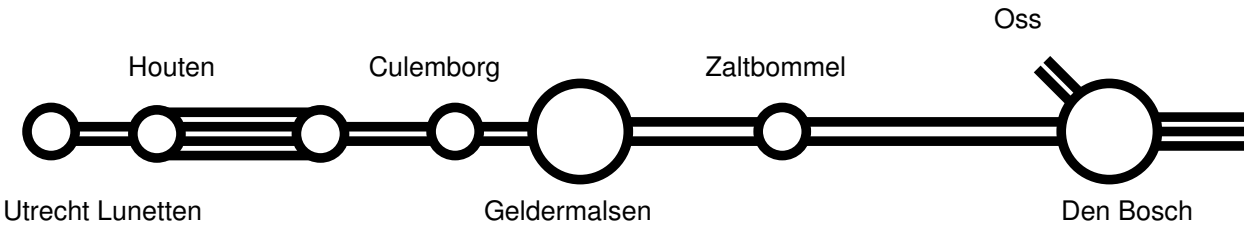


Figure 2.6: First Dutch test case - the Utrecht - Den Bosch area.

The infrastructure offers some possibility of train re-ordering and re-routing. Each train has a default route and a set of local re-routing options. Re-routing options can be applied along corridors or within a station, in which a train may be allowed to stop at different nearby platforms. Only standard train routes are considered and some less important switches have been omitted. Considering all possible alternative re-routing options yields a set of 356 routes. Figure 2.6 shows the dispatching area considered with the amount of tracks per area, and the indication of minor/major stations.

We consider a provisional hourly timetable for 2007 extended to the entire railway area. During peak hours, 26 passenger and freight trains are scheduled, in both directions, for the area around Geldermalsen. A more complex situation occurs at Den Bosch station, where up to 40 trains are scheduled each hour.

Additional constraints are included on the minimum transfer time between connected train services. Connections of the rolling stock are provided in Zaltbommel and Den Bosch stations. Passenger connections are located at Den Bosch station for the traffic directions from Oss to Utrecht and vice versa. The minimum time for passenger transfer connections varies from two to five minutes, depending on the distance to be travelled between the arrival platforms.

Table 2.5 presents average information on the CDR instances of this test case, with trains subject to random initial delays. This network presents a huge number of $|A|$ variables, since these are defined for each pair of trains that have routes sharing resources of the 50-km-long railway network.

Table 2.5: Characteristics of the first Dutch test case instances

Time Horizon (min)	Network Lenght (km)	Num Trains	Num Routes Per Train	Num Resources Per Train	MILP Variables $\|N\|$	$\|A\|$	$\|C\|$
60	50	40	9	31	1615	1092557	356

Table 2.6 reports on the computational results for 20 CDR instances of the Utrecht - Den Bosch test case. The instance code is as for Table 2.4. The best average results are obtained for VNS Basic in terms of the objective function value (as reported in bold). Specifically the best-known solution for instance FD_738_256.1 is only computed by VNS Basic and VNS General. The computation time of VNS Basic is decreased by more than half compared to TS. However, TS, VND and VNS Reduced are faster to compute the best-known solution for some instances. VNS General is the slowest algorithm to compute the best-known solution.

Table 2.6: Results obtained for the First Dutch (FD) test case instances

CDR Instance	TS		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)
FD_770_273.4	151	1.5	191	4.2	151	175.8	151	10.4	151	82.3	3900	6836.1
FD_738_256.1	68	59.0	78	0.4	64	176.2	64	45.3	71	11.9	91	6972.5
FD_991_401.1	154	1.2	154	0.3	154	177.6	154	0.7	154	2.7	307	6954.0
FD_1356_495.7	106*	86.9	187	0.0	117	177.9	106*	63.1	106*	57.9	3552	7195.4
FD_1249_534.0	119	117.5	301	0.7	130	180.0	119	81.9	130	48.0	3630	1903.9
FD_972_306.8	202	164.6	257	0.1	225	179.2	202	172.8	208	104.6	4349	7047.0
FD_1321_412.1	68	28.1	113	1.3	68	177.7	68	54.1	68	26.6	197	5872.0
FD_1372_379.6	291	6.4	278	25.9	291	167.5	291	4.0	291	4.0	4239	6292.5
FD_1776_607.0	135	9.5	181	0.1	135	180.2	135	1.9	135	4.5	410	1545.4
FD_659_133.1	94	144.1	142	0.5	98	179.7	94	30.7	94	5.8	111	6047.0
FD_816_158.2	114*	56.0	137	0.3	114*	179.1	114*	17.0	114*	32.6	114*	5272.0
FD_977_191.4	91*	27.4	176	0.4	91*	178.8	91*	92.4	91*	8.9	211	7064.4
FD_1017_201.3	85	42.4	97	0.4	85	179.5	85	0.5	85	2.0	4539	7018.0
FD_1240_293.5	103	143.9	103	1.2	103	178.5	103	2.0	103	14.5	4577	1969.3
FD_1312_294.6	123	116.6	123	2.4	123	178.3	123	12.3	123	10.1	3847	6780.6
FD_888_84.9	71	112.1	116	0.6	102	176.7	71	41.7	98	11.1	916	7132.6
FD_872_117.2	148	0.0	148	0.0	148	158.6	148	0.0	148	0.1	148	4418.1
FD_1371_182.8	122	60.5	150	0.5	122	177.2	122	4.2	122	0.6	3630	1898.1
FD_1769_216.2	99	55.2	99	0.3	99	172.6	99	0.3	99	3.3	99	6661.7
FD_1788_313.3	116*	96.1	116*	0.3	116*	172.9	116*	6.4	116*	0.4	116*	5909.3
Avg Results	123.0	66.5	157.4	2.0	126.8	176.2	**122.8**	32.1	125.4	21.6	1949.2	5539.5

Regarding the results obtained by the CPLEX solver, the lower bound is used to certify the optimality of some CDR solutions, while the upper bound is sometime good, even if those bounds are computed in a very long computation time. In Table 2.6, four CDR instances are solved to proven-optimum by some algorithms. The optimal solutions are identified with an asterisk in the columns regarding the objective function value. Two optimal solutions are also computed by CPLEX, while the other two are only certified by CPLEX. Overall, the optimality of several CDR instances is not proved by CPLEX, since the overall problem has a huge number of train ordering ($|A|$) variables required when combining all possible routing alternatives.

2.6.3 Results on the second Dutch test case

This test case is based on the railway network around the central station of Utrecht, the busiest station in the Netherlands. Figure 2.7 shows the overall network layout that has a diameter of around 20 km and 600 block sections, with the amount of tracks per branch, and the indication of minor/major stations considered. The main station area (known as Utrecht Central) provides 20 platform tracks, more than 100 switches and around 200 block sections. There are 5 main traffic directions that are delimited by the following minor stations: Utrecht Overvecht (on the line towards Amersfoort), Driebergen-Zeist (on the line towards Arnhem and Germany), Culemborg (on the line towards Den Bosch), Maarssen (on the line towards Amsterdam), and Vleuten (on the line towards Rotterdam and The Hague).

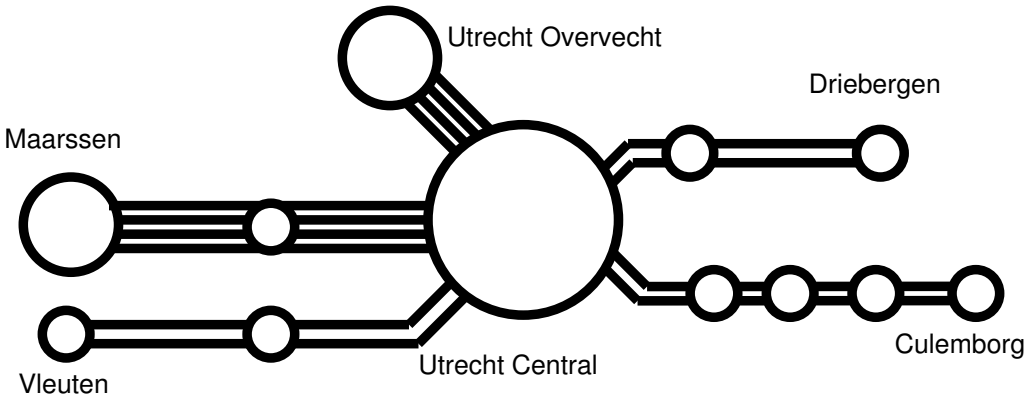


Figure 2.7: Second Dutch test case - the Utrecht Central Station area

The reference timetable is periodic with a cycle length of one hour. The timetable schedules 79 trains in a peak hour, with mixed passenger and freight traffic flows. The passenger trains are divided into International services going from the Netherlands to Germany and vice versa, Intercity services, Local trains and Sprinter services (faster local trains). The timetable provides connections between passenger services, coupling and splitting of rolling stock for intercity and local services coming from/going to Rotterdam, the Hague or Amersfoort, as well as re-use of rolling stock for commuter services towards Utrecht Overvecht and Culemborg. The alternative graph model of the traffic running on this complex station area is based on the aggregated formulation of the station routings at the interlocking areas described in [Corman et al., (2009b)].

Table 2.7 presents average information on the CDR instances tested for this network. For this network, the routing alternatives are platforming options within the main station area of Utrecht. We limit the number of stop platforms for all trains to a set of adjacent station platforms, thus limiting the connection time between two connected trains, i.e. the transfer time of passengers from one platform to another one.

Table 2.7: Characteristics of the second Dutch test case instances

Time Horizon (min)	Network Length (km)	Num Trains	Num Routes Per Train	Num Resources Per Train	MILP Variables $\|N\|$	$\|A\|$	$\|C\|$
75	20	79	3	22	2549	44730	228

The train delays are based on a statistical fitting procedure of the different train categories (similar to the one presented by Yuan [Yuan (2006)]), based on the arrival and departure data recorded by ProRail at Utrecht Central in April 2008. Here we consider a set of 20 timetable perturbation instances. For each instance, all trains suffer an entrance deviation, and multiple trains have a positive delay at their entrance in the network.

Table 2.8 reports on the results for 20 CDR instances of the Utrecht Central test case. The instance code is as for Table 2.4. For this set of instances, VND is the best algorithm since the best-known solution is, on average, computed in a shorter computation time (the computation time is up to 4 seconds) compared to the other algorithms. VNS Basic is, on average, the second best algorithm. Overall, the improvement of VND and VNS algorithms versus both the TS algorithm and the commercial solver is a strongly reduced computation time. Specifically, CPLEX is always outperformed by VND, VNS Basic and VNS Reduced. CPLEX is useful to certify the optimality for 8 CDR instances (see the values with asterisk). However, the optimality for the other 12 CDR instances is still not certified by CPLEX after 2 hours of computation.

Table 2.8: Results obtained for the Second Dutch (SD) test case instances

CDR Instance	TS		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)
SD_360_48.4	88	8.3	88	1.1	88	153.9	88	0.5	88	6.9	88	180.0
SD_510_48.8	95	25.1	95	0.9	95	175.7	95	26.9	95	3.7	95	168.0
SD_259_45.8	99	1.5	99	0.1	99	1.2	99	0.1	99	0.2	99	152.3
SD_253_42	90	135.4	90	2.0	90	174.7	90	2.1	90	51.4	90	255.9
SD_273_48.1	83	0.1	83	0.1	83	180	83	0.1	83	0.1	83	298.7
SD_382_48.6	51*	59.8	51*	1.2	51*	15.4	51*	0.5	51*	5.7	51*	263.5
SD_286_44.2	51*	0.1	51*	0.1	51*	2.1	51*	0.1	51*	0.1	51*	39.8
SD_502_37.2	51*	0.1	51*	0.1	51*	2.2	51*	0.1	51*	0.2	51*	155.3
SD_377_44.3	51*	158.2	51*	2.1	51*	174.3	51*	6.3	51*	27.9	51*	62.8
SD_310_29.7	51*	1.5	51*	1.0	51*	180	51*	0.6	51*	2.3	51*	148.2
SD_418_43.5	160	2.8	160	2.5	160	180	160	3.0	160	9.5	160	190.0
SD_528_34.3	232	4.0	232	4.0	232	180	232	4.0	232	4.0	232	402.5
SD_740_38.1	93	13.9	93	0.1	93	178.5	93	0.1	93	0.1	93	390.5
SD_493_32.3	66	0.2	66	0.2	66	1.2	66	0.2	66	0.2	66	37.9
SD_411_33.4	153	43.7	153	1.1	153	171.8	153	28.4	153	10.2	153	240.0
SD_1165_47.1	51*	3.1	51*	1.8	51*	179.5	51*	6.1	51*	3.2	51*	178.9
SD_493_23.4	51*	0.8	51*	0.7	51*	160.5	51*	0.4	51*	3.4	51*	43.4
SD_370_34.6	51*	30.4	51*	3.5	51*	180	51*	13.1	51*	30.3	51*	129.5
SD_675_28.6	70	12.1	70	2.1	70	175.2	70	6.0	70	113.1	70	619.5
SD_475_27.2	133	6.7	133	0.9	133	180	133	4.3	133	7.9	133	178.1
Avg Results	**88.5**	25.4	**88.5**	1.3	**88.5**	132.3	**88.5**	5.1	**88.5**	14.0	**88.5**	206.7

2.6.4 Results on the British test case

The test bed is a mixed-traffic railway network nearby the city of London, approximately from King's Cross station to Huntingdon station, on the East Coast Main Line of The United Kingdom. Figure 2.8 shows a layout of the studied network with the amount of tracks per branch, and the indication of minor/major stations. In this set of experiments the scheduler has to deal with strongly disrupted traffic situations in which some trains have speed restrictions and others are re-routed.

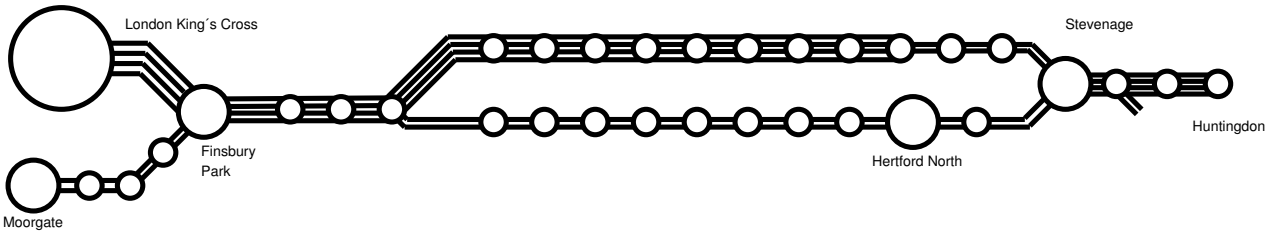


Figure 2.8: British test case - the East Coast Main Line nearby London

Table 2.9: Characteristics of the British (B) test case instances

Time Horizon (min)	Network Length (km)	Num Trains	Num Routes Per Train	Num Resources Per Train	MILP Variables $\|N\|$	$\|A\|$	$\|C\|$
60	80	90	2	69	5565	46219	128

Table 2.10: Results obtained for the British test case instances

CDR Instance	TS		VND		VNS General		VNS Basic		VNS Reduced		CPLEX	
	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)
B_4558_113.0	2236*	61.9	2236*	0.2	2236*	176.4	2236*	0.1	2236*	0.1	2236*	11.8
B_474_48.1	108*	6.3	108*	3.2	108*	180	108*	16.2	108*	4.6	108*	75.0
B_452_49.6	379*	4.8	979	1.5	379*	179.6	379*	45.2	379*	4.4	379*	111.8
B_2577_152.4	532*	114.0	813	1.9	532*	176.4	532*	6.4	532*	2.3	532*	350.0
B_852_67.4	412*	3.1	412*	1.3	672	179.3	412*	1.8	672	0.7	412*	25.5
B_716_59.9	277*	0.1	277*	0.1	277*	180	277*	0.1	277*	0.1	277*	4.1
B_437_55.1	3151*	118.0	3151*	0.1	3151*	0.1	3151*	0.1	3151*	0.1	3151*	79.9
B_520_54.1	353*	12.4	353*	4.7	353*	180	353*	4.3	353*	1.4	353*	13.6
B_556_56.9	451*	23.3	451*	0.9	901	178.5	451*	0.6	901	1.6	451*	22.0
B_2374_77.4	3199*	68.3	3199*	1.7	3199*	180	3199*	2.4	3199*	0.5	3199*	2410.3
B_1218_77.1	491	157.0	2626	2.1	408*	174.2	408*	70.3	408*	43.6	408*	2434.1
B_561_46.7	3499*	0.1	3499*	0.1	3499*	176.2	3499*	0.1	3499*	0.1	3499*	131.9
B_2194_59.2	1426*	35.3	1426*	0.1	1426*	0.1	1426*	0.1	1426*	0.1	1426*	2489.9
B_2255_65.1	1992*	80.3	1992*	0.2	1992*	0.2	1992*	0.2	1992*	0.2	1992*	2750.8
B_487_49.8	421*	25.4	421*	12.9	421*	153.8	421*	142.0	421*	3.5	421*	1667.3
B_452_46.2	408*	34.6	3030	0.1	1579	159.9	408*	118.7	1579	2.1	408*	352.2
B_1349_98.3	597*	14.6	597*	7.9	597*	180	597*	6.1	597*	6.6	597*	1751.1
B_512_52.9	2788*	103.0	2788*	1.9	2788*	177.6	2838	0.1	2788*	0.8	2788*	14.7
B_540_50.1	1202*	110.6	1266	98.6	1202*	170.5	1202*	40.3	1202*	82.5	2244	2263.0
B_556_52.0	508*	97.8	508*	8.6	508*	179.9	508*	3.1	723	7.1	508*	493.6
Avg Results	1221.5	53.6	1506.6	7.4	1311.4	149.4	**1219.9**	22.9	1322.2	8.1	1269.5	872.6

Table 2.9 presents information on the largest CDR instances tested for the British test case. For this set of CDR instances we only consider two routes per train, as provided by an industrial partner in [D'Ariano et al., (2014)].

Table 2.10 reports on the computational results for 20 CDR instances of the British railway network. The instance code is as for Table 2.4. For this set of instances, we know the optimal solution for all CDR instance, as certified by the lower bound of CPLEX. Regarding the performance of the various algorithms, VNS Basic is the best algorithm in terms of the average objective function value (as reported in bold), even if the other algorithms and the commercial solver compute a better solution for instance British_512_52.9. Furthermore, VNS Basic is often the fastest algorithm to compute the optimal solution. However, some CDR instances are solved to optimality in a shorter time by VND, VNS Reduced and TS.

2.6.5 Discussion on the obtained results

This section gives a brief overview of the average performance of the algorithms described in this paper.

- Comparing TS and VNS Basic, the latter outperforms the former in terms of the average objective function value and also provides an average strong reduction of the time to compute the best-known solutions. A motivation is that TS mostly performs single routing changes in RCPO, while VNS Basic is based on multiple simultaneous routing changes in a combination of neighbourhood structures.
- VNS Basic is the best variable neighbourhood search algorithm in terms of the average objective function value, since it combines the local search and the shaking procedures.
- VNS Basic is better than VNS Reduced, since the former is also guided by the local search procedure, helping to intensify the search of better quality solutions in specific regions of the search space.
- Comparing VND and VNS General, the latter algorithm sometimes improves the performance of VND via the shaking procedure, that can be a profitable attempt to escape from a local optimum.
- DJ, WOCP and their combination are the best neighbourhood structures when incorporated in VNS Basic. These neighbourhood structures are new promising ingredients to solve the CDR problem.
- Different CDR instances result in rather unsimilar average trends regarding the quality of the solutions computed by the various algorithms. In general a small amount of train traffic, in terms of trains/hour, makes the CDR algorithms rapidly converge to almost the same solution. When the traffic is more dense and multiple re-routing options are available, the CDR algorithms compute more diverse solutions.
- The computational speed of AGLIBRARY is mostly depending on the algorithmic structure and configuration, but it also depends on the railway infrastructure and traffic flow characteristics.

- The complexity of the CDR instance depends on the type of the re-ordering and re-routing alternatives available for each train. For instance, a train can avoid a conflict with another train by changing the stop platform in a station area, while a train can only slighly anticipate or posticipate a conflict with another train on a corridor by performing a local re-routing. In the latter case, the train ordering is the key decision variable to solve the conflicting traffic situation. In general, the interdependence between train scheduling and routing variables plays a key role in the resolution of the CDR problem.

- The commercial solver is not able to compute a good quality solution for most of the CDR instances in a short computation time, and therefore cannot be part of a DSS for real-time train traffic control.

2.7 Conclusions and future research

This paper proposes fast scheduling and routing metaheuristics for real-time railway traffic management in busy networks, with particular focus on the efficient control of strong traffic disturbances (such as multiple train delays and temporarily unavailable block sections). The CDR problem is modelled via the alternative graph, that is a generalization of the disjunctive graph, and as a MILP formulation for simultaneous train scheduling and routing. To solve the CDR problem, several algorithmic innovations are considered, which relate to design of effective metaheuristics based on a problem decomposition into train scheduling and routing decisions. Variable neighbourhood search schemes (VND, Reduced VNS, Basic VNS and General VNS) are proposed based on systematic changes of a combination of neighbourhood structures.

The new metaheuristic algorithms are benchmarked against a state-of-the-art tabu search algorithm [Corman et al., (2010)] and a commercial MILP solver. The evaluation is performed over multiple networks with varying traffic and infrastructure characteristics. The algorithms proposed in this paper, with various combinations of neighbourhood structures, improve the effectiveness of the previously developed CDR algorithms and outperform the commercial MILP solver. The main contributions of the variable neighbourhood search are a general significant reduction of the time required to compute good quality (sometimes proven optimal) solutions, and the computation of new best known solutions for some CDR instances.

Further research should be focused on a number of issues: the assessment of the proposed methodology to solve the CDR problem for different railway networks, traffic flows and types of demand or disturbance that could be described by some metric, in order to find an approximate relation between the instance characteristics and the expected algorithmic performance; the development and evaluation of alternative CDR heuristics, metaheuristics and exact approaches; the customization and application of the models and algorithms proposed in this paper to other transportation and logistics problems.

2.8 Appendix

A small illustrative example is proposed to explain the basic characteristics of the model, the neighbourhoods, the neighbours and their evaluations. We consider 4 trains (A, B, C, D) that are running on the railway network of Figure 2.9; those can be identified by different colors. Trains A and B cross the network from left to right, while trains C and D run in the opposite direction. Moreover, A and D are local services which stop at the stations R and Q, while the other trains have no planned stop in the network.

The network is composed by 8 resources (block sections), labeled from 1 to 8 in the top part of Figure 2.9. Signals delimit the block sections. Moreover, the two stations R and Q (respectively resources 2, 7, 8; and 4, 6) are associated with additional platform resources, reported as 2R and 8R (station R); and as 4Q and 6Q (station Q). Those resources model the dwell process, for the trains that have a scheduled stop at the stations (i.e. trains A and D).

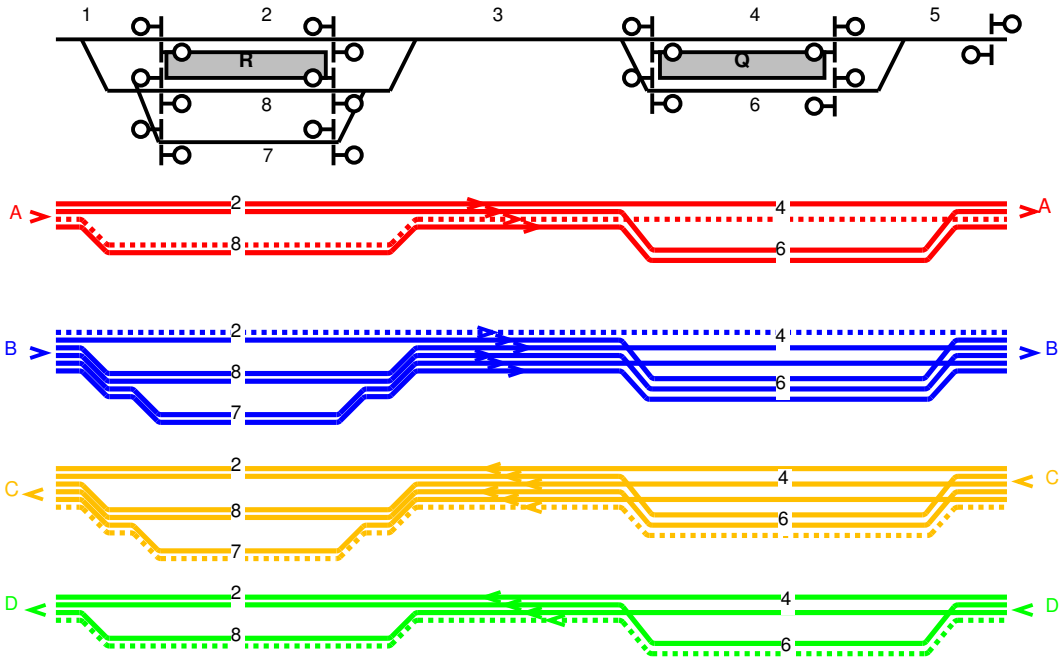


Figure 2.9: Railway network and the routing alternatives for the four trains

Each train has to be routed in the network and there are multiple alternative routes. Trains B and C can use any of the resources 2, 7, 8 in the area of station R, and any of the resources 4 and 6 in the area of station Q. As any route in the first station area can be combined with any route in the second station area, a total of 6 routing options are available for trains B and C. Due to the need to stop at station R, trains A and D can use only resources 2 and 8. They cannot use resource 7, since there is no platform available in

that resource. So trains A and D can only be routed between resources 2 and 8, and between resources 4 and 6, for a total of 4 routing options. The full set of train routes, including the default ones, is reported schematically on the bottom side of Figure 2.9. The default routes, reported in dotted lines, are as follows. Between brackets is the running time (dwell time for the station stop): **A1:** 1(1), 8(2), 8R(1), 3(3), 4(2), 4Q(1), 5(2); **B1:** 1(1), 2(2), 3(3), 4(2), 5(2); **C1:** 5(2), 6(3), 3(5), 7(3), 1(2); **D1:** 5(2), 6(3), 6Q(1), 3(5), 8(3), 8R(1), 1(2).

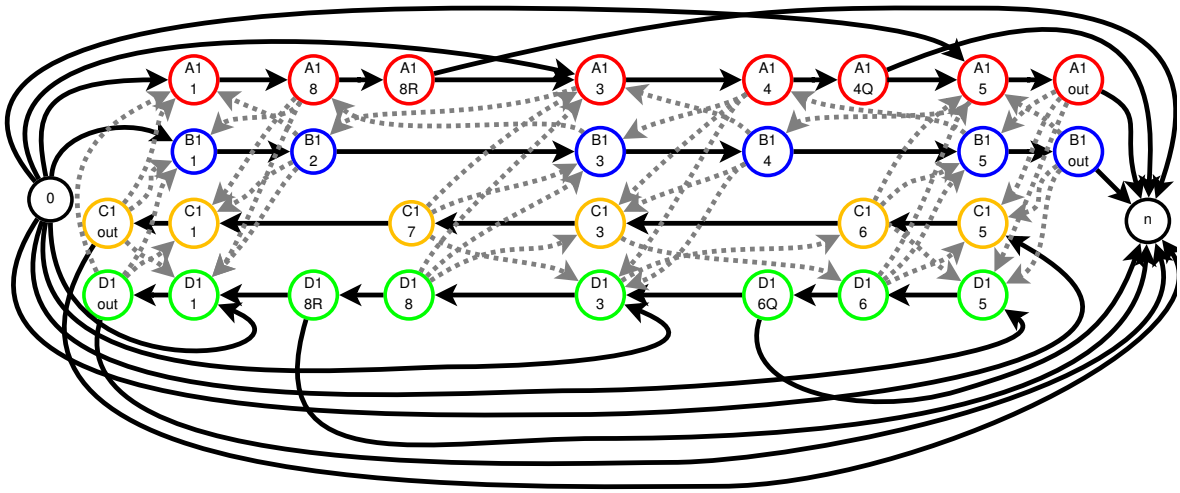


Figure 2.10: Alternative graph of the example with default train routes

Figure 2.10 shows the alternative graph model of the CDR problem with the default routes. Each train results in a sequence of nodes (operations) reported horizontally. Each node (e.g. A1-8) is identified by the name of the train and routing (e.g. A1) plus the resource over which the operation is performed (e.g. 8).

In the alternative graph model, the fixed arcs are reported in solid color, while the alternative arcs are reported in dotted gray. The latter arcs are given for each shared resource, namely resources 1, 3, 5 for all trains; resource 8 for trains A, D; resource 4 for trains A, B; resource 6 for trains C, D. Moreover, a number of fixed arcs exit node 0, in relation with the entrance time (including the initial delay) in the network and the minimum departure time from the station platforms. Analogously, a number of fixed arcs enter node n, in relation with the exit time from the network, and the arrival time at the station platforms. For this illustrative example, the headway time between consecutive trains is always equal to 1 time unit.

A solution to the CDR problem is reported in Figure 2.11 as a time distance path (top) and alternative graph (bottom). For the time distance graph, space is along the x-axis, time (increasing downwards) is along the y-axis. In this solution, train C goes first over resource 3, followed by trains B, A and D. The minimum headway time (one time unit) over the sections constrains the departure of train A from station R, as well as the departure time (in the opposite direction) of train D. Those trains are assumed to be waiting

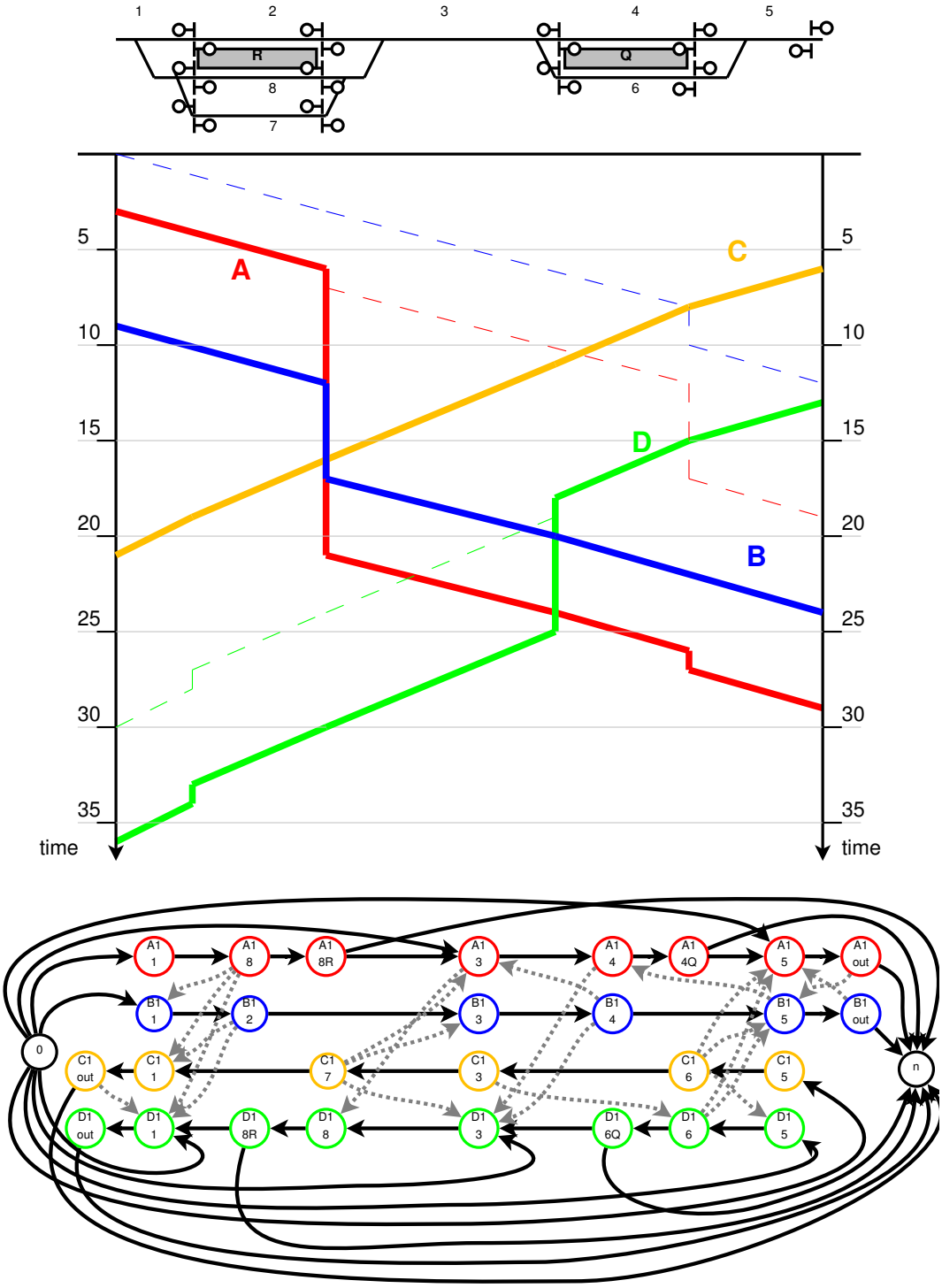


Figure 2.11: A CDR solution shown as a time-distance plot, and the associated alternative graph

at station R until the resource 3 becomes available. In the dotted lines of Figure 2.11, the original plan is reported. The train B suffers an initial delay of 9 time units, while the other trains have no initial delays. In Figure 2.11, the scheduled arrival times of the trains are: **A1:** 6 at station R (no delay); 12 at station Q (the delay is 14, as the realised arrival time is 26); 19 at exit of the network (the delay is 10, as the realised exit time is 29). **B1:** 12 at the exit of the network (the realised exit time is 24; the consecutive delay is 3). **C1:** 21 at the exit of the network (no delay); **D1:** 18 at station Q (no delay); 27 at station R (the delay is 6, as the realised arrival time is 33); 30 at the exit of network (the delay is 6 as the realised exit time is 36).

In the alternative graph model of the CDR solution reported at the bottom of Figure 2.11, exactly one arc from each alternative pair has been selected, i.e. a train ordering decision has been taken for each potential conflict. For instance, the chosen train order over resource 3 (i.e. C-B-A-D) results in the following arc selection: (C1-7, A1-3); (C1-7, B1-3); (C1-7, D1-3); (B1-4, A1-3); (B1-4, D1-3); (A1-4, D1-3). This is a scheduling solution with fixed routes, with a maximum consecutive delay of 14.

We next discuss the four neighbourhoods explained in Section 2.5.3 in terms of the train and route rankings.

Free-Net Waiting Operations Jobs neighbourhood $\mathcal{N}_{FNWJ}$: The $\mathcal{N}_{FNWJ}$ ranking is based on the sum of the consecutive delays collected at each waiting node in the graph of the incumbent routing solution in which all alternative arcs are unselected (i.e. free-net traffic situation) but the one generating the waiting node. For example, let's consider the pair ((B1-4, C1-3),(C1-7, B1-3)) that concerns the order of trains B and C over resource 3. We next refer to the time distance graph of Figure 2.12.

If the arc (B1-4, C1-3) is selected, the order reported in solid style in Figure 2.12 lines is implemented. Train C arrives at the end of resource 3 at time 11, and has to wait for the exit of train B from resource 3, that happens at time 16, plus 1 time unit of headway time making it 17. This results in a delay to train C of 6 time units. Train B suffers no additional delay. The max consecutive delay is thus 6.

If the other arc of the pair (C1-7, B1-3) is selected, the other train order is considered, as reported via the dotted lines in Figure 2.12. Train B has to wait from time 12, for the exit of train C from resource 3, that happens at time 16, plus 1 time unit of headway time. Train B then exits the network at time 24, with a consecutive delay of 3 time units. Differently, train C can enter resource 3 at time 11 without additional delay, and exit the network without any delay. The maximum consecutive delay is thus 3.

The $\mathcal{N}_{FNWJ}$ ranking computes the score as the minimum consecutive delay generated on the two waiting nodes related to each alternative pair, i.e. $\min(3, 6) = 3$ is assigned to B1 and C1 due to the alternative pair ((B1-4, C1-3),(C1-7, B1-3)). The complete score of each train is computed by summing up the minimum consecutive delay generated by each alternative pair. In this example, no other alternative pair generates any additional score, i.e. if only two trains at a time are considered in the network (free-net situation) no

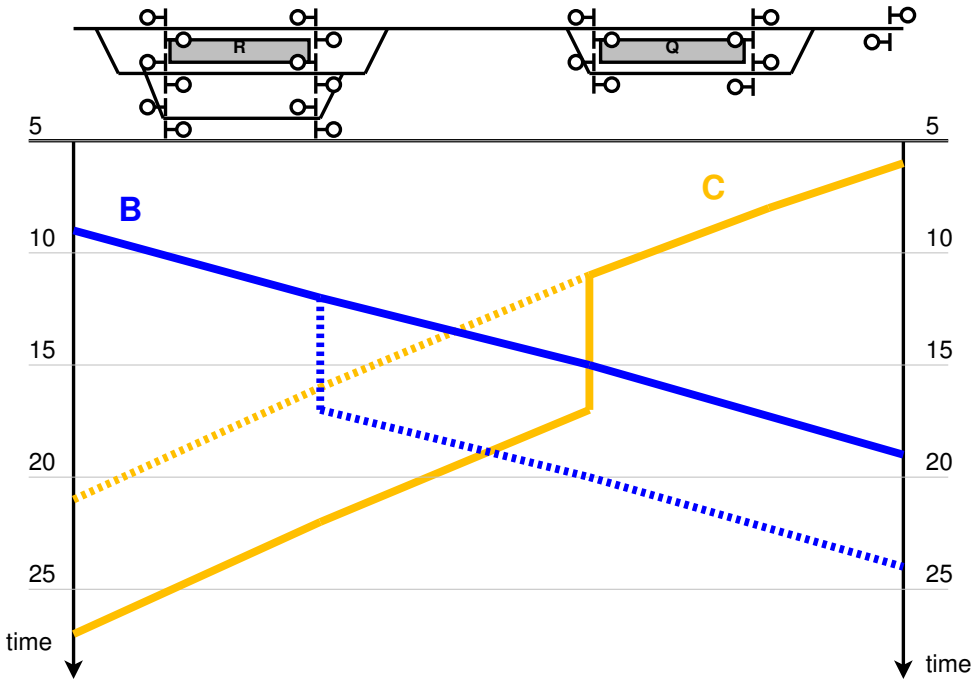


Figure 2.12: Free-net waiting operation - ranking computation

other conflict arises. The $\mathcal{N}_{FNWJ}$ ranking values are thus 3 for B1 and C1, 0 for the other trains.

Ramified Critical Path Operations neighbourhood $\mathcal{N}_{RCPO}$: For the solution of Figure 2.11, the critical path is highlighted in Figure 2.13 in color and black. The operations on the critical path are as follows: B1-1, B1-2, B1-3, B1-4, A1-3, A1-4, A1-4Q. The ramified critical path is an extension of the critical path including also the waiting operations preceding or following the ones on the critical path. In this case the ramified critical path corresponds to all operations of trains A1 and B1, highlighted in the thicker lines (gray or black). In $\mathcal{N}_{RCPO}$, the ranking values for each train are computed as the maximum value $l^{S'(F^R)}(0, krp) + l^{S'(F^R)}(krp, n)\ \forall\ (krp)$ in the ramified critical path of the graph of the incumbent solution. This value corresponds exactly to the makespan, i.e. the maximum consecutive delay of 14, for all nodes in the critical path. For each train route, the maximum of its ranking values is the ranking score. The score of A1 and B1 is 14, the soore for C1 and D1 is 0.

Waiting Operations Critical Path neighbourhood $\mathcal{N}_{WOCP}$: The $\mathcal{N}_{WOCP}$ ranking is based on the sum of the consecutive delays collected on the waiting operations on the critical path of the incumbent solution. We refer to Figure 2.14 for a graphical illustration. In this example, there is a single waiting operation (A1-3) in which a consecutive delay of value 14 time units is collected for A1. The consecutive delay (14) of A1-3 is computed as follows: the weight of the path $l^{S'(F^R)}(0, B1 - 4) = 20$, corresponding to the time

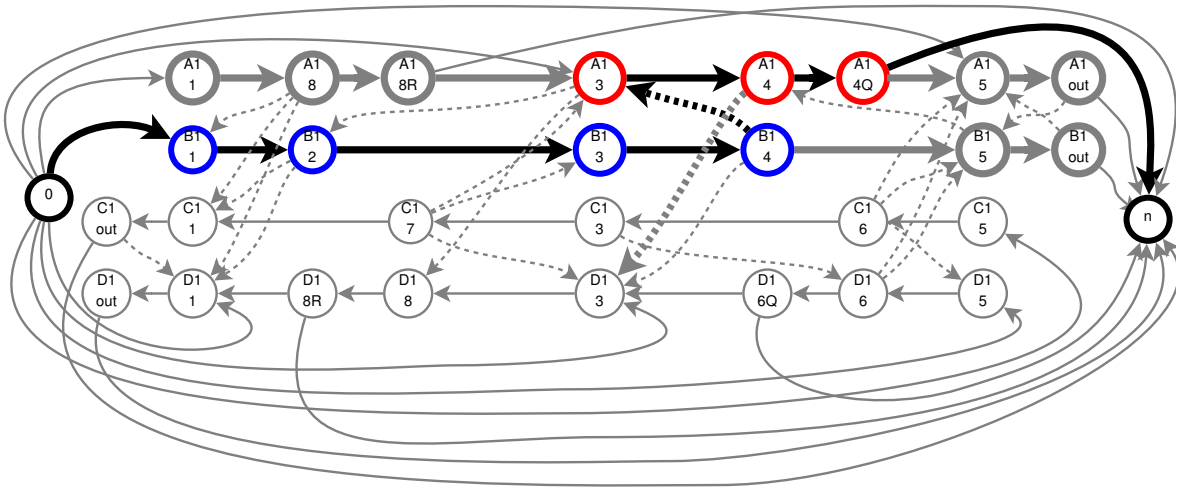


Figure 2.13: Ramified critical path

needed by train B1 to exit B1-3 (and to enter B1-4); plus the weight $w^F_{(B1-4),(A1-3)} = 1$, corresponding to the headway time; minus the weight of $l^{S'(F^R)}(0, B1-3)) = 7$, corresponding to the time at which train A1 would be able to enter A1-3, without any other potential conflict. This yields a score of 14. There is no other waiting operation on the critical path. The ranking of $\mathcal{N}_{WOCP}$ is thus 14 for train A1, 0 for the others.

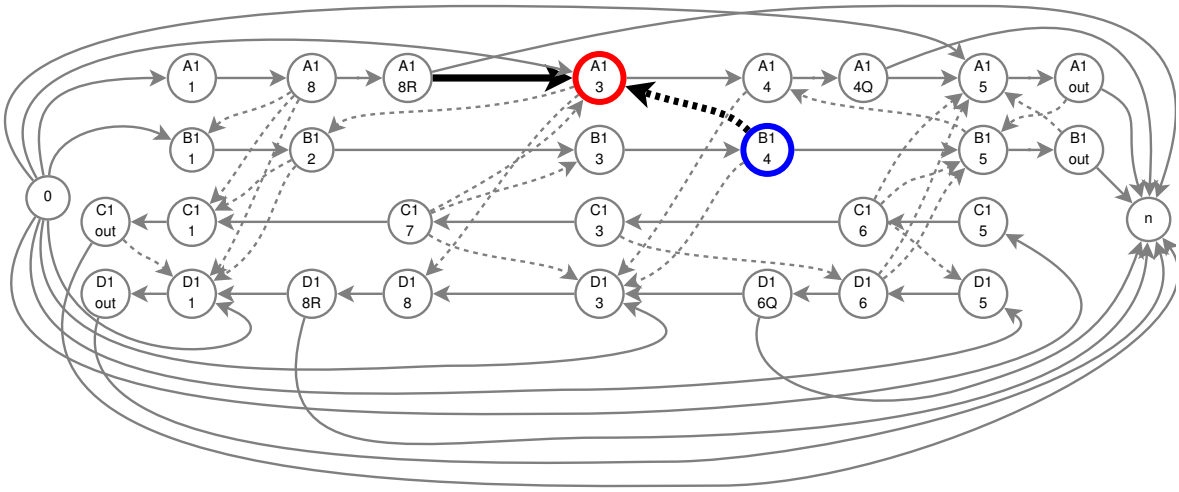


Figure 2.14: Waiting operations on the critical path

Delayed Jobs neighbourhood $\mathcal{N}_{DJ}$: The $\mathcal{N}_{DJ}$ ranking is based on the maximum consecutive delays at some relevant locations for each train. Figure 2.15 highlights the fixed arcs going into node n, in which some consecutive delays are experienced. Train C1 is not delayed and its score is thus 0. The other trains have

the following consecutive delays: A1-8R: 0 A1-4Q: 14; A1-out: 10; B1-out: 3; D1-4Q: 0; D1-8R:6; D1-out: 6. For each job, the largest delay is taken, which corresponds to a rank of 14 for train A1, 3 for B1, 0 for C1, 6 for D1. Figure 2.15 highlights the arcs going into node n in which consecutive delays are collected.

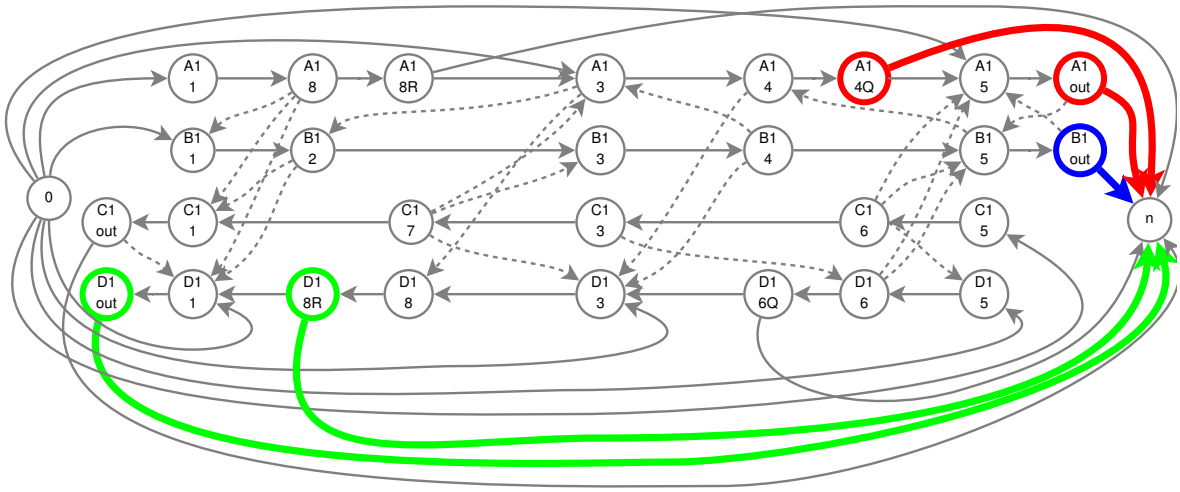


Figure 2.15: Delayed jobs

Acknowledgements

We acknowledge the support from Ing. Giacomo Zaninotto and Ing. Alessandro Toli. We also would like to thank the anonymous reviewers for their helpful and constructive remarks. The CDR instances are available upon request for research purposes by sending an email to the corresponding author of this paper.

Chapter 3

Lower and upper bound algorithms for the real-time train scheduling and routing problem in a railway network

Abstract: *This paper focuses on the development of new algorithms for the real-time train scheduling and routing problem in a complex and busy railway network. Since this is a strongly NP-hard problem and practical size instances are complex, simple heuristics are typically adopted in practice to compute feasible but low quality schedules in a short computation time. In order to compute good quality solutions, we consider a mixed-integer linear programming formulation of the problem and solve it with a commercial solver. However, the resolution of this formulation by a commercial solver often takes a too long computation time. Therefore, a new methodology based on the relaxation of some train routing constraints in the formulation is proposed for the quick computation of a good quality lower bound. The lower bound solution is then transformed via a constructive metaheuristic into a feasible schedule, representing a good quality upper bound to the problem. Computational experiments are performed on several disturbed traffic situations for two practical case studies from the Dutch and British railways. The results show that the new lower and upper bounds are computed in a few seconds and are often of similar quality to the ones computed by the commercial solver in hours of computation.*

3.1 Introduction

This work addresses the *Real-Time Train Scheduling and Routing Problem* (RTTSRP), i.e., the problem of computing in real time a conflict-free schedule for a set of trains circulating in a network within a time window $W = [t, t + \delta]$, given the position of the trains at time t and the status of the network in W. The objective function is the minimization of train delays. A schedule is conflict-free if it satisfies the railway traffic regulations, which prescribe a minimum separation between consecutive trains on a shared resource in order to ensure the safety of train movements and to avoid deadlock situations in the network.

The study of real-time train scheduling and routing problems received increasing attention in the literature in the last years. Early approaches (starting from the pioneering work of [Szpigel (1973)]) tend to solve very simplified problems that ignore the constraints of railway signalling, and that are only applicable for specific traffic situations or network configurations (e.g., a single line or a single junction), see, e.g., the literature reviews in the following papers: [Ahuja et al., (2005), Cacchiani et al., (2014), Cordeau et al., (1998), Corman and Meng (2015), Fang et al., (2015), Hansen and Pachl (2014), Lusby et al., (2013), Meng and Zhou (2011), Pellegrini and Rodriguez (2013), Pellegrini et al., (2014), Törnquist and Persson (2007)]. Among the reasons for this gap between early theoretical works and practical needs are the inherent complexity of the real-time process and the strict time limits for taking and implementing decisions, which leave small margins to a computerized Decision Support System (DSS).

The *alternative graph* of [Mascis and Pacciarelli (2002)] is among the few models in the literature that incorporate, within an optimization framework, the microscopic level of detail that is necessary to ensure the fulfillment of traffic regulations. This model generalizes the job shop scheduling model in order to deal with additional constraints. Each operation denotes the traversal of a resource (block/track section or station platform) by a job (train route).

A big-M Mixed Linear Integer Programming (MILP) formulation of the RTTSRP can be obtained from the alternative graph model by introducing a binary variable for each train ordering decision (i.e. an *alternative pair*) and a binary variable for each routing decision [D'Ariano et al., (2014)]. The resulting problem is strongly NP-hard [Mascis and Pacciarelli (2002)].

This paper reports on recent improvements implemented in the state-of-the-art optimization solver AGLIBRARY [D'Ariano et al., (2008)]. The solver includes a branch and bound algorithm for scheduling trains with fixed routes [D'Ariano et al., (2007a)] and a tabu search metaheuristic for re-routing trains [Corman et al., (2010)].

Previous research left open two relevant issues. The first issue is how to certify the quality of the RTTSRP solutions in a short computation time by means of effective lower bounds. This issue is made difficult due to

the poor quality of the lower bounds computed by MILP solvers, that are usually based on a linear relaxation of the big-M MILP formulation of the RTTSRP.

The second issue concerns with the computation of effective upper bounds through the development of new solution methods. Both these issues motivate this paper, whose contribution consists of the following algorithms.

A first algorithm is proposed for the computation of a lower bound for the RTTSRP. This is obtained by the construction of a particular alternative graph for a relaxed RTTSRP in which each train route is composed by two types of components: (i) *real operations* that are in common with all alternative routes of the associated train; (ii) *fictitious operations* that represent the shortest path between two real operations that can be linked by different routing alternatives for the associated train. For the latter type of component, no train ordering decision is modeled, disregarding the potential conflicts between trains. The resulting alternative graph is then solved to optimality by the branch and bound algorithm in [D'Ariano et al., (2007a)].

A second algorithm is a constructive metaheuristic proposed in order to optimize the selection of default routes. This metaheuristic starts from the optimal solution obtained for the alternative graph of the relaxed RTTSRP problem and iteratively replaces the fictitious operations of each train with a particular routing alternative. The selection of the routing alternative is based on the evaluation of the insertion of various train routes via the construction of the corresponding alternative graph and the computation of train scheduling solutions via fast heuristics.

Computational experiments are performed on practical-size instances from the Dutch and British railways. The new algorithms often compute good quality lower and upper bounds to the optimal RTTSRP solutions in a shorter computation time compared to a commercial MILP solver.

3.2 Problem description

The RTTSRP must consider the following issues. The railway network is made by track sections and platform stops at stations. A train is not allowed to depart from a platform stop before its scheduled departure time and is considered late if arriving at the platform later than its scheduled arrival time. At a platform stop, the scheduled stopping time of each train is called *dwell time*.

Signals, interlocking and Automatic Train Protection (ATP) systems control the train traffic by imposing safety regulations between trains, setting up train routes and enforcing speed restrictions on running trains. Fixed block ATP systems ensure safety through the concept of block section, a part of the infrastructure that is exclusively assigned to at most one train at a time. Train movements can be modeled by a set of

characteristic times, as follows. The *running time* of a train on a block section starts when its head (the first axle) enters the block section and ends when the train reaches the end of the block section.

Safety regulations impose a minimum separation between consecutive trains traveling on the same block section, which translates into a minimum *headway time* between the start of the running times of two consecutive trains on the same block section. This time depends on the length of the block section, as well as on other factors like the speed and length of the trains and includes the time between the entrance of the train head in a block section and the exit of its tail (the last axle) from the previous one, plus additional time margins to release the occupied block section and to take into account the sighting distance.

Disturbances affect train traffic flows in a railway network. We can distinguish between light traffic *perturbations* from neighbouring dispatching areas and heavy traffic *disruptions*. The former are light disturbances caused by a set of delayed trains in a dispatching area, while the latter are much stronger disturbances of the scheduled times and routes (e.g. due to some block sections being unavailable for a certain amount of time). Other kinds of disturbances include extensions to dwell times due to passengers boarding, connection constraints between trains, or technical problems; and running time prolongation because of headway conflicts between trains or technical failures.

Initial delays propagate between trains when solving potential conflicting routes. Namely, a *potential conflict* between two or more trains arises if the trains request the same resource within a time interval smaller than the minimum headway time between them. The solution of the potential conflict is to fix the order of trains over the resource; in that case, some of the approaching trains might be forced to decelerate and thus experiencing a consecutive delay. Those delays may propagate to other trains causing a domino effect of increasing traffic disturbances [Kecman et al., (2013)]. In this paper, the objective function is the minimization of the maximum consecutive delay: the largest positive deviation from the scheduled times at relevant locations.

3.3 Problem formulation

This section describes our formulation of the RTTSRP. The RTTSRP can be divided into two sub-problems: (i) the selection of a route for each train, and (ii) the train scheduling decisions once the routes have been fixed. We first provide a brief description of the alternative graph model for sub-problem (ii), more information can be found e.g. in [Corman et al., (2011a), Corman et al., (2014a), Corman et al., (2009b)] and in [D'Ariano et al., (2007a), D'Ariano et al., (2007b), D'Ariano et al., (2008)]. We then present a big-M MILP formulation for the overall scheduling and routing problem.

3.3.1 Alternative graph model

The alternative graph (AG) model for sub-problem (ii) of the RTTSRP is a triple $G = (N, F, A)$ where $N = \{0, 1, ..., n-1, n\}$ is a set of nodes, F is a set of *fixed* directed arcs, and A a set of pairs of *alternative* directed arcs.

Each node, except the start 0 and end n nodes, is associated with the start of an operation krj, where k indicates the train, r the route chosen and j the resource it traverses. The start time t_{krj} of operation krj is the entrance time of train k with route r in resource j.

The fixed arcs are used to model running, dwell, connection, arrival, departure, and pass through times of trains. Let the resources p and j be two consecutive resources processed by train k with route r, the fixed arc $(krp, krj) \in F$ models a job constraint between the nodes krp and krj. The weight $w^F_{krp\_krj}$ represents a minimum time constraint between t_{krp} and t_{krj}: $t_{krj} \geq t_{krp} + w^F_{krp\_krj}$. A fixed arc $(umv, krz) \in F$ enforces a connection constraint between train k with route r and train u with route m.

The alternative arcs are used to model the headway times between two consecutive trains. Each pair of alternative arcs $((krj, ump), (umi, krp)) \in A$ models train sequencing decisions between train k with route r and train u with route m on the common resource p. Note that j [respectively i] is the next resource processed by train k [u] when using route r [m]. The two arcs of the pair have associated the weights $w^A_{krj\_ump}$ and $w^A_{umi\_krp}$. In any solution, only one arc of each pair can be selected. If alternative arc (krj, ump) [(umi, krp)] is selected in a solution, the constraint $t_{ump} \geq t_{krj} + w^A_{krj\_ump}$ [$t_{krp} \geq t_{umi} + w^A_{umi\_krp}$] has to be satisfied. This corresponds to fixing the order of trains, first k and then u [first u and then k].

A solution for sub-problem (ii) of the RTTSRP is represented by the following graph structure. Selection S is a set of alternative arcs obtained by selecting exactly one arc from each alternative pair in A and such that the resulting graph $\mathcal{G}(F, S) = (N, F \cup S)$ does not contain positive weight cycles. The graph selection allows to associate feasible train orders and times to all operations. The objective function is measured as a makespan minimization. Given S and any two nodes krp and uml, we let $l^S(krp, uml)$ be the weight of the longest path from krp to uml in $\mathcal{G}(F, S)$. The start time t_{krp} of $krp \in N$ is the quantity $l^S(0, krp)$, which implies $t_0 = 0$ and $t_n = l^S(0, n)$.

3.3.2 MILP formulation

A MILP formulation for sub-problem (ii) of the RTTSRP can be obtained from the alternative graph model by representing each fixed arc in F as a linear constraint, by translating each alternative pair in A into a pair of linear constraints, and by introducing a binary variable to model the choice between one of the paired constraints. The variables of sub-problem (ii) are the following: $|N|$ real variables t_{krj} associated to the start

time of each operation $krj \in N$, and $|A|$ binary variables $x_{(krj,ump),(umi,krp)}$ associated to each alternative pair $((krj, ump), (umi, krp)) \in A$.

We next extend the MILP formulation for sub-problem (ii) to the overall RTTSRP formulation. The constraint sets F and A are enlarged in order to contain all possible train routing combinations. $|C|$ new binary variables y are associated to the set of routes of the considered train, in addition to the $|N| + |A|$ variables of sub-problem (ii).

The RTTSRP is formulated as the *disjunctive program* (3.1) with the following notation. M is a sufficiently large number, e.g. the sum of all arc weights. Z is the number of trains, R_b the number of routes for each train $b = 1, ..., Z$. For each train b, only one among the R_b routes can be chosen in any RTTSRP solution. The binary variable y_{ab} indicates if route a is chosen (1) or not (0) for train b.

The following constraint holds for train b: $\sum_{a=1}^{R_b} y_{ab} = 1$.

When a route r is chosen for train k (i.e. $y_{kr} = 1$), each fixed constraint in F related to route r and train k must be satisfied. For each fixed arc $(krp, krj) \in F$, $t_{krj} - t_{krp} \geq w^F_{krp\_krj}$ must hold. A fixed arc $(umv, krz) \in F$ enforces a connection constraint between train k with route r and train u with route m (if $y_{um} = y_{kr} = 1$).

$$
\begin{aligned}
&\min\ t_n \\
&s.t. \\
&\textstyle\sum_{a=1}^{R_b} y_{ab} = 1 && b = 1, ..., Z \\
&t_{krj} - t_{krp} \geq w^F_{krp\_krj} + M(1 - y_{kr}) && (krp, krj) \in F \\
&t_{krz} - t_{umv} \geq w^F_{umv\_krz} + M(2 - y_{kr} - y_{um}) && (umv, krz) \in F \\
&t_{ump} - t_{krj} \geq w^A_{krj\_ump} + M(2 - y_{kr} - y_{um}) + Mx_{(krj,ump),(umi,krp)} && ((krj, ump), (umi, krp)) \in A \\
&t_{krp} - t_{umi} \geq w^A_{umi\_krp} + M(2 - y_{kr} - y_{um}) + M(1 - x_{(krj,ump),(umi,krp)}) && ((krj, ump), (umi, krp)) \in A \\
&y_{ab} \in \{0, 1\} \\
&x_{(krj,ump),(umi,krp)} \in \{0, 1\}
\end{aligned}
\tag{3.1}
$$

Regarding the alternative constraints in A, if $y_{um} = y_{kr} = 1$ and the routes m and r of trains u and k use the same resource p of the network, a potential conflict exists on that resource and an ordering decision has to be taken. This is modelled by introducing the binary variable $x_{(krj,ump),(umi,krp)}$ for the alternative pair $((krj, ump), (umi, krp)) \in A$, related to trains u and k travelling on resource p. There are two possible scheduling decisions for each alternative pair $((krj, ump), (umi, krp)) \in A$: if $x_{(krj,ump),(umi,krp)} = 0$ then $t_{ump} - t_{krj} \geq w^A_{krj\_ump}$ must be satisfied (i.e. $(krj, ump) \in S$); if $x_{(krj,ump),(umi,krp)} = 1$ then $t_{krp} - t_{umi} \geq$

$w^A_{umi\_krp}$ must be satisfied (i.e. $(umi, krp) \in S$).

3.4 The lower bound algorithm

We propose a new method for the computation of a good quality lower bound for the RTTSRP. The basic idea is to reduce the sub-problem (i) to a single route for each train, and solve the resulting sub-problem (ii) to optimality.

Solving the sub-problem (i) requires the construction of the alternative graph $G' = (N', F', A')$ as follows.

The set N' contains node 0, node n and the nodes of all jobs. For each train, the corresponding job (train route) starts from node 0 and ends in node n. The intermediate nodes are either real or fictitious operations. We recall that a real operation for a train is an operation in common with all its alternative routes, i.e. an operation to be performed by the train in any solution of sub-problem (i). A fictitious operation is the shortest path between two real operations that can be linked by different routing alternatives.

F' is the set of fixed arcs in G'. In general, the weight $w^{F'}_{krp\_krj}$ of each arc $(krp, krj) \in F'$ corresponds to the less stringent constraint between nodes krp and krj when looking at all routing alternatives of k. In particular, if krp is a fictitious operation and krj is a real operation, the weight $w^{F'}_{krp\_krj}$ is equal to the minimum travel time between the start of krp and the start of krj, taken from the shortest path among the routing alternatives of k.

The weight $w^{F'}_{umv\_krz}$ corresponds to the less stringent connection constraint between the nodes umv and krz when looking at all routing alternatives of u and k.

$$
\begin{aligned}
&\min\ t_n \\
&s.t. \\
&t_{krj} - t_{krp} \geq w^{F'}_{krp\_krj} && (krp, krj) \in F' \\
&t_{krz} - t_{umv} \geq w^{F'}_{umv\_krz} && (umv, krz) \in F' \\
&t_{ump} - t_{krj} \geq w^{A'}_{krj\_ump} + M x_{(krj,ump),(umi,krp)} && ((krj, ump), (umi, krp)) \in A' \\
&(t_{krp} - t_{umi} \geq w^{A'}_{umi\_krp} + M(1 - x_{(krj,ump),(umi,krp)}) && ((krj, ump), (umi, krp)) \in A' \\
&x_{(krj,ump),(umi,krp)} \in \{0, 1\}
\end{aligned}
\tag{3.2}
$$

A' is the set of alternative pairs in G'. In each pair $((krj, ump), (umi, krp)) \in A'$, ump and krp are necessarily real operations, since the lower bound method only takes scheduling decisions between conflicting resources in all train routing combinations. In other words, if a potential conflict can be avoided by some routing combinations, this conflict is disregarded in the lower bound method.

Sub-problem (ii) requires the computation of the optimal graph selection S^* for G'. This corresponds to solving the following *disjunctive program* (3.2). The lower bound value is the optimal solution of sub-problem (ii). In this paper, we solve sub-problem (ii) via the branch and bound algorithm in [D'Ariano et al., (2007a)], truncated at a given time limit of computation. In case the optimal solution cannot be found, the algorithm always returns a valid lower bound value.

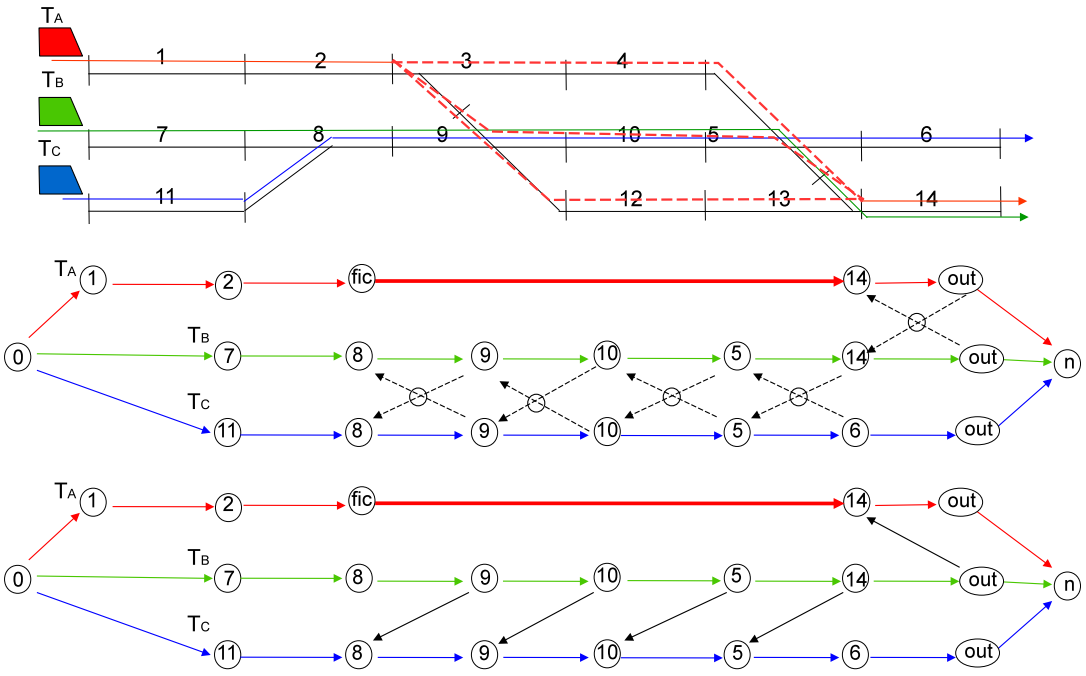


Figure 3.1: An example of lower bound calculation

Figure 3.1 provides an illustrative example of the lower bound method for a small network with 14 block sections. There are 3 trains: T_A has 3 alternative routes: [1-2-3-4-5-14], [1-2-3-9-12-13-14], [1-2-3-9-10-5-14]; T_B has the route [7-8-9-10-5-6]; T_C has the route [11-8-9-10-5-6].

Figure 3.1 (top) shows the alternative graph G' used by the lower bound method. The jobs (trains) can be identified as T_A, T_B and T_C. For the sake of simplicity, the nodes of each job are only identified with the infrastructure resource number, except for the nodes representing the exit from the network (i.e. the nodes named *out*) and the node fic. The latter node corresponds to a fictitiuos operation for T_A, that is used in G' to represent that there are multiple alternative routings between resources 2 and 14 for T_A. Each fixed arc can be identified with a solid arrow and with the colour of the corresponding train, while each alternative arc can be identified with a dashed black arc and the pair with a circle between the two alternative arcs.

Arc $(fic, 14)$ is shown with a bold solid arrow and represents the shortest path between from 2 to 14 for T_A.

The lower bound value is obtained by computing an optimal selection S^* and $l^{S^*}(0, n)$ in G', as shown in the bottom graph of Figure 3.1.

3.5 The upper bound algorithm

We propose a constructive metaheuristic for the computation of a good quality upper bound for the RTTSRP. This metaheuristic starts from the alternative graph G' and a selection S generated by the lower bound method, and iteratively trasforms each job (train route) with some fictitious operations into a new job with real operations only. At each iteration, the train to be trasformed is chosen from a train list, according to a sort criteria defined in advance (in this paper we order the trains on a first come first served basis). The selection of a real route for the train is based on a local search procedure in which we evalute the insertion in G' of all its potential real routes by an alternative graph greedy heuristic from [Pranzo et al., (2003)]. After the local search, the real route generating less consecutive delays is inserted in G', and the chosen alternative graph heuristic updates the selection S of G'. When all trains with some fictitious operations have been processed, the train list is empty and the iterative process ends. The metaheuristic returns $l^S(0, n)$ obtained in G' at the last iteration, that corresponds to a solution to the RTTSRP. Figure 3.2 reports a flow chart of the algorithm.

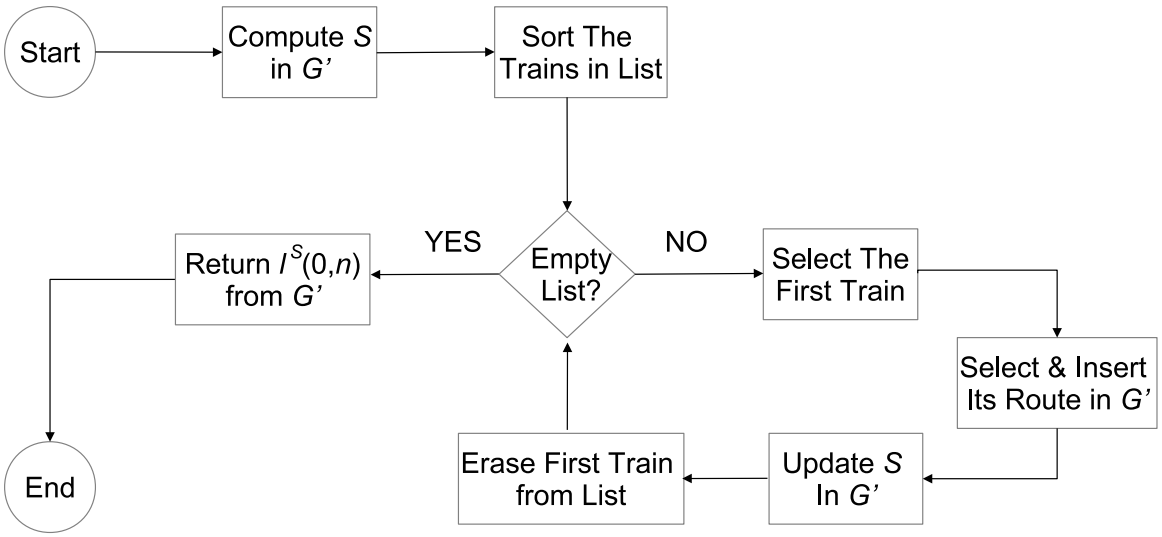


Figure 3.2: Flow chart of the upper bound method

Figure 3.3 shows a RTTSRP solution for the illustrative example of Figure 3.1. The route chosen for

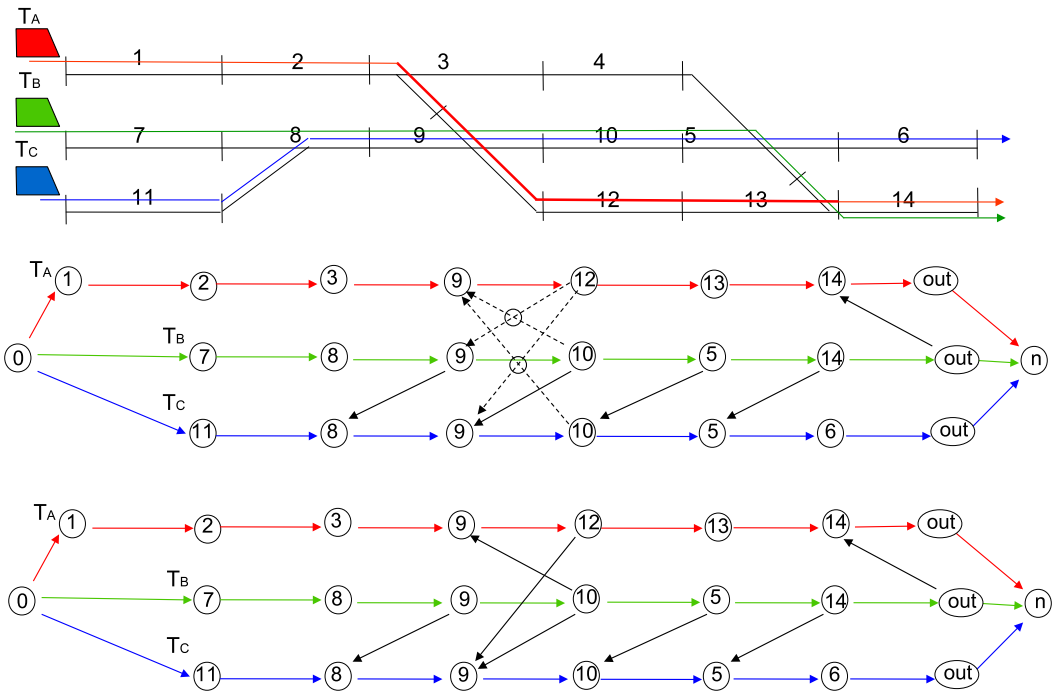


Figure 3.3: An example of upper bound calculation

T_A is [1-2-3-9-12-13-14], while the train ordering decisions are the following: T_B proceeds T_A and T_C in all common resources, T_A proceeds T_C in resources 9 and 14. The graph of the RTTSRP solution is shown in the bottom of Figure 3.3. The top graph of Figure 3.3 is the alternative graph G' obtained after the route insertion. In G', some alternative pairs are selected as in the selection S obtained by the lower bound method, while other alternative pairs are generated by the insertion of a real route for T_A. The greedy heuristic from. [Pranzo et al., (2003)] updates S by selecting the two remaining alternative pairs.

3.6 Computational experiments

This section presents the experimental results on the two practical case studies of the Dutch and British railways. Both the test cases are modelled with a microscopic level of detail, which means that switches, signals, block sections, and track segments in complex station areas are included (yielding several hundreds of resources). Also, train movements are described with a precision of seconds.

For each case study, we consider a set of 15 RTTSRP instances, varying the initial delays of trains. The experiments are executed on a workstation Power Mac with processor Intel Xeon E5 quad-core (3.7 GHz), 12 GB of RAM. We compare the lower and upper bounds obtained by using the methods of Sections 3.4 and 3.5 implemented in AGLIBRARY, and the MILP formulation of Section 3.3.2 solved by IBM LOG CPLEX MIP 12.0.

In the lower bound method, we use the Branch and Bound (BB) algorithm of [D'Ariano et al., (2007a)] truncated at 1 hour of computation. However, for all the instances tested in this paper the time limit of BB is never reached.

The CPLEX solver is truncated at 2 hours of computation. The time limit of CPLEX is reached for 16/30 instances.

3.6.1 Dutch test case

Figure 3.4 presents the Utrecht - Den Bosch railway network that is around 50 km long and consists of 191 block sections and 21 platforms. There are two main tracks, a long corridor for each traffic direction, a dedicated stop for freight trains nearby Zaltbommel and 7 passenger stations.

Table 3.1 presents average information on the 15 RTTSRP instances of this test case. This network presents a huge number of $|A|$ variables, since these are defined for each pair of trains sharing a resource in one hour traffic prediction horizon and for all train routing combinations.

Table 3.2 present the average results on the 15 RTTSRP instances in terms of Lower Bound (LB) and Upper Bound (UB) values (in seconds) obtained by the two solvers, and the time (in seconds) required to

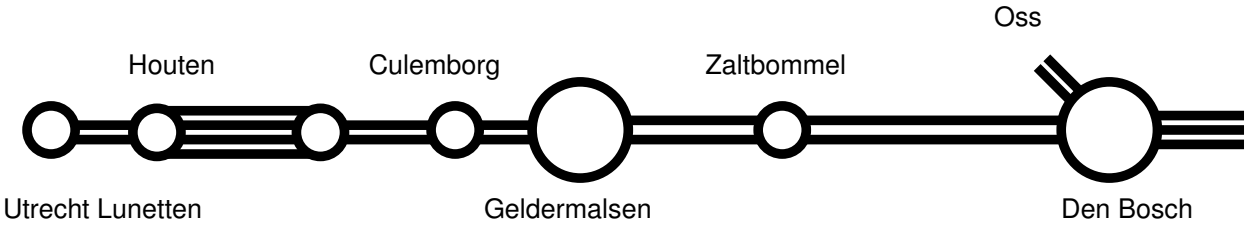


Figure 3.4: The Utrecht - Den Bosch railway network

Table 3.1: Characteristics of the instances

Num Trains	Num Routes Per Train	Num Resources Per Train	MILP Variables $\|N\|$	$\|A\|$	$\|C\|$
40	9	31	1615	1092557	356

compute those values.

Table 3.2: Average computational results

Solver	LB		UB	
	Value	Comp. Time	Value	Comp. Time
AGLIBRARY	94.1	0.1	179.4	11.5
CPLEX	58.7	244.9	1259.9	5399.6

The results of Table 3.2 show that AGLIBRARY, on average, outperforms CPLEX both in terms of LB quality, UB quality and the related computation times.

3.6.2 British test case

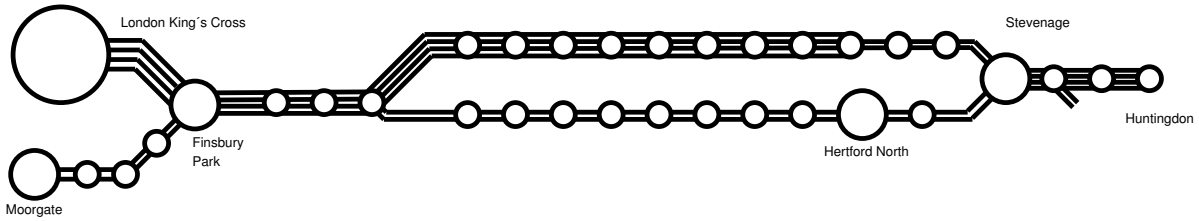


Figure 3.5: The East Coast Main Line nearby London

Figure 3.5 shows a layout of a part of the East Coast Main Line of The United Kingdom, approximately 80 km from King's Cross station to Huntingdon station. In this set of experiments the scheduler has to deal with strongly disrupted traffic situations on a mixed-traffic railway network, on which some trains have speed restrictions and other trains need to be re-routed.

Table 3.3 presents average information on the 15 RTTSRP instances of this test case. Each RTTSRP instance is a hour traffic prediction horizon with 90 trains. For this set of instances we only consider two

routes per train, as provided by an industrial partner in [D'Ariano et al., (2014)], and thus less $|A|$ and $|C|$ variables compared to the Dutch case study.

Table 3.3: Characteristics of the instances

Num Trains	Num Routes Per Train	Num Resources Per Train	MILP Variables $\|N\|$	$\|A\|$	$\|C\|$
90	2	69	5565	46219	128

Table 3.4 present the average results on the 15 RTTSRP instances in terms of the same indicators of Table 3.2.

Table 3.4: Average computational results

Solver	LB		UB	
	Value	Comp. Time	Value	Comp. Time
AGLIBRARY	1023.9	4.0	1472.1	7.8
CPLEX	1013.3	469.9	1062.8	872.5

From the results in Table 3.4, the LB quality of AGLIBRARY is, on average, superior to the one of CPLEX, while the constructive metaheuristic is not always able to compute an UB value close to the one obtained by the other solver. However, AGLIBRARY always requires a few seconds to compute the LB and UB values, while CPLEX requires several minutes of computation and is thus not applicable to recover real-time traffic disturbances.

3.7 Conclusions and future research

This paper introduces new methods for the computation of good quality LB and UB for the RTTSRP. The LB value is computed by relaxing some constraints in the MILP formulation of the RTTSRP and solving the relaxed formulation via the BB algorithm of [D'Ariano et al., (2007a)]. The UB value is obtained by transforming the LB solution in a RTTSRP solution via the greedy heuristics of [Pranzo et al., (2003)]. The computational results are promising, since both the new LB and UB values are computed very quickly and are often of good quality compared to the LB and UB values computed with CPLEX in hours of computation.

Future research should be dedicated to incorporating the new LB and UB methods in advanced heuristic, metaheuristic and exact algorithms for the RTTSRP.

Chapter 4

Ant colony optimization for the real-time train routing selection problem

Abstract: *This chapter deals with the real-time problem of scheduling and routing trains in a railway network. In the related literature, this problem is usually solved starting from a subset of routing alternatives and computing the near-optimal solution of the simplified routing problem. We study how to select the best subset of routing alternatives for each train among all possible alternatives. The real-time train routing selection problem is formulated as an integer linear programming formulation and solved via an algorithm inspired by the ant colonies' behaviour. The real-time railway traffic management problem takes as input the best subset of routing alternatives and is solved as a mixed-integer linear program. The proposed methodology is tested on two practical case studies of the French railway infrastructure: the Lille terminal station area and the Rouen line. The computational experiments are based on several practical disturbed scenarios. Our methodology allows the improvement of the state of the art in terms of the minimization of train consecutive delays. The improvement is around 22% for the Rouen instances and around 56% for the Lille instances.*

4.1 Introduction

In the last decade, traffic demand in transports in general and railways in particular has significantly grown. To appropriately address this growth, railway undertakings and infrastructure managers face the challenge of expanding their offer. The difficulties in building new infrastructures due to high costs or physical obstacles translate into the need to utilize the already existing ones at their full capacity. To maintain a satisfactory quality of service and reduce passengers' inconvenience [Ginkel and Schöbel (2007)], it is necessary to manage precisely and efficiently the railway traffic in case of unexpected disturbances and disruptions that may affect the normal course of daily operations. Still, few decision support tools are available to help dispatchers in minimizing the impact of these events. Currently, dispatchers intervene manually in real-time, sometimes using pre-made contingency plans that, while helpful, cannot cover all the potential scenarios that may occur. Indeed, it is not easy to immediately judge the effects a particular decision may have. Solving a single event as best as possible could actually not be the most appropriate decision when considering traffic with a broader perspective. In fact, such a decision may indirectly generate further disturbances, propagating the delay in a snow ball effect.

The real-time Railway Traffic Management Problem (rtRTMP) is the problem of detecting and solving conflicting track requests by multiple trains during disturbed operations [Pellegrini et al., (2014)]. The rtRTMP is an NP-Hard problem where routing, ordering and timing decisions are made simultaneously. Moreover, the characteristics of the railway infrastructure and of the traffic flows, and in particular the number of routing alternatives for each train, strongly affect the time required to compute good quality solutions [Pellegrini et al., (2015)].

Recent literature reviews show a rich stream of research focused on the rtRTMP [Cacchiani et al., (2014), Corman and Meng (2015), Fang et al., (2015)]. Most of the existing approaches do not consider all set of possible routing alternatives present in the infrastructure for each train. A subset of routing alternatives is very often considered in the rtRTMP optimization. This subset is usually defined by taking into account the suggestions given by the dispatchers and by considering alternative routings similar to the one defined in the timetable. This is a reasonable assumption since the rtRTMP needs to be solved in real-time. However, the resulting rtRTMP optimization is a simplified routing problem and the computation of near-optimal solutions for the simplified problem do not necessarily correspond to near-optimal solutions for the original rtRTMP.

This paper studies the under-investigated problem of selecting the best subset of routing alternatives for each train among all possible alternatives. We call this problem the real-time Train Routing Selection Problem (rtTRSP). Figure 4.1 shows a SysML activity diagram (SysML, 2015) in which the relation between

the rtRTMP and rtTRSP solution activities is depicted. An activity diagram describes the sequence of actions to be performed to transform inputs into outputs. These inputs and outputs are represented as rectangles spanning the activity frame boundary. The actions are represented as rounded rectangles. An action begins as soon as all its required inputs are provided. In general, the rtTRSP is a subproblem of the rtRTMP: the rtTRSP solution allows the reduction of the size of the rtRTMP search space, which shall then be solved on the reduced space through one of the approaches existing in the literature. The input required to solve the rtRTMP are data on the infrastructure, the timetable and the traffic state at a reference time. The data on the network and the timetable are used to initialize both the "solve the rtTRSP" and the "solve the rtRTMP with the routing subsets" actions. After the initialization, the traffic state at a reference time is provided in order to perform the "solve the rtTRSP" activity by taking into account the current traffic information. This input is transmitted concurrently to the "solve the rtTRSP" and the "solve the rtRTMP with the routing subsets" actions. However, the latter can start only after the end of the former, since the required input "routing subsets" needs to be computed beforehand. The final output produced is a new working timetable, which includes all routing, ordering and timing information to be used in the operations.

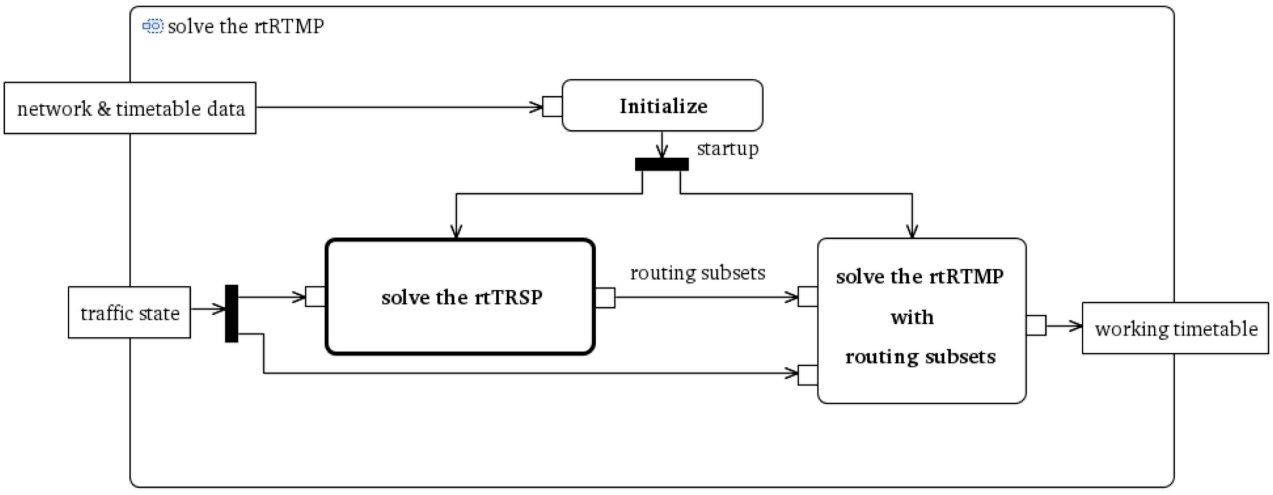


Figure 4.1: SysML activity diagram representing how to solve the rtRTMP

The common practice of solving the rtTRSP is to follow some pre-determined directives given by the infrastructure managers on which routing subset should be considered for each train in real-time in order to help dispatchers when dealing with traffic disturbances. These directives are too general, while a case-to-case solution is required during operations. A dynamic selection of the routing subsets may intuitively improve the quality of the rtRTMP solution. However, the definition of the routing subsets requires to address the

following trade-off. On the one hand, considering small subsets of all possible train routing decisions would reduce the number of variables to be tackled in the rtRTMP, and increasing the chances of finding good quality solutions in a short computation time. On the other hand, considering large subsets would increase the probability of preserving the optimal mix of train routings in the search space of the rtRTMP, but finding the global optimum may require a too high computation time.

This paper presents a general methodology to tackle the rtTRSP based on the scheme of Figure 4.1. We propose an integer linear programming formulation of the rtTRSP with the objective of finding a good quality routing subset for each train, where the subset quality is defined as a function of the interactions among the routings selected for trains traveling in the infrastructure. Since a solution to the rtTRSP needs to be computed very quickly and the search space contains a very large number of solutions, we solve the rtTRSP with an Ant Colony Optimization (ACO) meta-heuristic inspired by the foraging behaviour of ant colonies [Dorigo and Stützle (2004)]. The ACO meta-heuristic has already been applied to a number of complex problems in the railway domain, including rolling stock circulation [Tsuji et al., (2012)], timetabling [Huang (2006)] and re-scheduling [Fan et al., (2012)] problems.

In this work, the rtRTMP is modeled with the state-of-the-art formulation of [Pellegrini et al., (2014)] and solved with the exact and heuristic approaches proposed in [Pellegrini et al., (2014), Pellegrini et al., (2015)]. This is one of the most accurate formulations for the rtRTMP, in which the infrastructure is modeled at the level of track-circuit and of the interlocking system actually used in the practice. Several alternative routings are available for each train and the resulting rtTRSP is a challenging problem to be solved.

A thorough experimental analysis on two French case studies (the Lille terminal station area and the Rouen line) is proposed in order to assess the benefits of solving the rtTRSP to define the search space of the rtRTMP. Since practical-size rtRTMP instances present a too large number of variables, it is too time-consuming to consider all the routing alternatives for each train in the optimization model. For this reason, we analyze the performance of the heuristic approach *RECIFE-MILP* [Pellegrini et al., (2015)] when using various routing subsets:

- all routing alternatives: this case corresponds to remove the rtTRSP from the solution process;
- a random selection of the routing subsets: we take a random solution of the rtTRSP;
- an optimized selection of the routing subsets: we take the best rtTRSP solution computed by ACO.

The computational results show that solving the rtTRSP is very useful, since reducing the number of routing subsets to be managed in the rtRTMP strongly improves the performance of the heuristic approach RECIFE-MILP. The improvement is larger when using the ACO meta-heuristic. All heuristic approaches are evaluated with a short computation time compatible with real-time application. We also compare the

heuristic solutions with the global optimum computed by solving the formulation of [Pellegrini et al., (2014)] with a commercial solver, and evaluate the solutions computed by the heuristic approaches when varying the size of the routing subsets.

The paper is structured as follows: Section 4.2 presents a review of the rtRTMP literature. Section 4.3 defines the rtRTMP and the rtTRSP. Section 4.4 briefly reports the rtRTMP formulation (additional details are reported in Appendix) and introduces the rtTRSP formulation. Section 4.5 proposes methods to assign costs to the routing subsets in the rtTRSP. Section 4.6 describes the ACO meta-heuristic developed to solve the rtTRSP. Section 4.7 gives the results of the computational analysis on the two investigated test cases. Section 4.8 summarizes the conclusions of this work and outlines directions for further research on the rtTRSP and rtRTMP.

4.2 Literature review

Railway scheduling and routing problems deal with the efficient allocation of track capacity at strategic, tactical and operational levels [Lusby et al., (2011)]. On a strategic level, the problems deal with line planning and infrastructure construction and/or modification [Bussieck (1998), Goossens et al., (2005), School (2005)]. On a tactical level, the problems deal with the creation of plans for the utilization of infrastructure and human resources, including timetable generation and platform planning in station areas [Cacchiani and Toth (2012), Carey and Carville (2003), Carey and Crawford (2007), Sels et al., (2014), Zwaneveld et al., (2011)]. On an operational level, the problems deal with the recovery of the plans on the available infrastructure when disturbances and/or disruptions make them infeasible [Cacchiani et al., (2014), Corman and Meng (2015), Fang et al., (2015)]. This section discusses recent state-of-the-art approaches on the rtTRSP and rtRTMP that are problems to be solved at the operational level.

The rtRTMP can be formulated in several ways: alternative graph model [Corman et al., (2009b), Corman et al., (2010), Corman et al., (2011a), Corman et al., (2014a), D'Ariano et al., (2007a), D'Ariano et al., (2008)], constraint programming [Rodriguez (2007)], mixed-integer programming [D'Ariano et al., (2014), Dessouky et al., (2006), Lamorgese and Mannino (2015), Liu and Kozan (2009), Meng and Zhou (2011), Pellegrini et al., (2014), Törnquist and Persson (2007)], multicommodity flow models [Caimi et al., (2011), Caimi et al., (2012)], set-packing inspired models [Lusby et al., (2013)], time-index models [Meng and Zhou (2014)]. Further modeling approaches are discussed in the survey papers of [Cacchiani et al., (2014), Corman and Meng (2015), Fang et al., (2015)] and the recent book of [Hansen and Pachl (2014)]. The various approaches differs in (i) the level of detail the authors formulate the rtRTMP and in (ii) the way their algorithmic methods manage the variables during the solution process.

Regarding (i), the main difference is the granularity used to model the infrastructure and traffic flows. Two main possibilities exist: *macroscopic* approaches model the resources as groups of block-sections [Dessouky et al., (2006),Lamorgese and Mannino (2015),Mu and Dessouky (2011),Törnquist and Persson (2007)], while *microscopic* approaches model each resource as a single block-section [Corman et al., (2010),D'Ariano et al., (2007a),D'Ariano et al., (2008)], or a single track-circuit [Caimi et al., (2012),Pellegrini et al., (2014),Rodriguez (2007)]. Considering track-circuits allows the modeling of two variants of interlocking systems: a route-lock route-release interlocking, in which the utilization of a block-section locks all block-sections sharing a track-circuit with it independently on the actual position of the train within the block-section itself; a route-lock sectional-release interlocking, in which the utilization of a block-section locks the block-section sharing with it a not-yet-released track-circuit. This second interlocking system can be considered only for microscopic models where a resource represents a single track-circuit. This paper deals with two French infrastructures, where the route-lock sectional-release interlocking is used in practice. For this reason, we model the rtTRSP with the approach of [Pellegrini et al., (2014)], i.e., through a microscopic modelling of the track-circuits and the route-lock sectional-release interlocking system.

Regarding (ii), the key variables of the rtRTMP concern the routing alternatives for each train, its arrival and departure time at each station, the train ordering decisions on shared infrastructure resources. In general, the number of variables may be huge, so the computation time and memory usage to solve the rtRTMP may also be very high. To limit the complexity of the problem, different solution approaches have been considered. In [Törnquist and Persson (2007)] the ordering decisions concerning trains which are far away from each other, in terms of expected crossing time of common track sections, are considered fixed. In [Caimi et al., (2011), Rodriguez (2007)], the rtRTMP is solved in two steps: first the routing to assign to each train is investigated and then the ordering decisions are taken. In [Corman et al., (2011a),Corman et al., (2014a),D'Ariano et al., (2007a),D'Ariano et al., (2014),Liu and Kozan (2009),Meng and Zhou (2011), Törnquist (2012)] the routing alternatives are not taken into consideration, setting to each train its timetable routing, and thus limiting the problem to timing and ordering decisions. In other approaches instead, the timetable routings are used to compute a first improvable solution. In [Corman et al., (2010),D'Ariano et al., (2008)], this solution is bettered through an iterative procedure that, at each step, changes the routing for a certain number of trains and then finds new timings and orderings for the trains affected. The alternative routings are given as input to the iterative procedure (i.e., the rtTRSP is solved in a preliminary step). In [Pellegrini et al., (2014),Pellegrini et al., (2015)], the objective function value of the initial solution is used as un upper bound to the problem. Other approaches instead look at different means to simplify their search for a solution. In [Dessouky et al., (2006)], deadlock avoidance checks are adopted to reduce the search space. In [Caimi et al., (2012)], the timing and routing decisions are combined

through the definition of blocking-stairways, each one combining a routing and a speed-profile, selecting then few among a finite number of alternatives for each train. In [Lamorgese and Mannino (2015)] instead, the problem is tackled with a decomposition algorithm where a microscopic model is used in stations while a macroscopic one for the overall network. In [Mu and Dessouky (2011)], a simultaneous freight train routing and scheduling problem is formulated as a mixed integer programming model based on binary variables to explicitly model train ordering decisions and solved via some heuristic procedures based on clustering trains according to their entrance time in the network. A major drawback of this approach is the simplification of the topological structure of the network, since tracks are modeled macroscopically and can only allow trains traveling in a predefined traveling direction. In the rtRTMP, we need to deal with a microscopic representation of the network that allows all possible train routing alternatives. We also use a mathematical model based on binary variables for the train ordering and routing decisions. However, our main focus is not the modeling of the rtRTMP but the study of which train routing alternatives are more promising in order to efficiently solve the rtRTMP. In [Meng and Zhou (2014)], an alternative mathematical model is proposed in which the usage of resources is managed via cumulative flow variables instead of binary variables. This modeling approach has the advantage to limit the number of train ordering and routing variables. Another advantage compared to citeMuDessouky:2011 is the capability to deal with any kind of network structure. The model of [Meng and Zhou (2014)] is solved via heuristics based on a problem decomposition in single train optimization sub-problems. This is a very efficient approach since each sub-problem has a relatively small size and time-dependent shortest path algorithms can be utilized to find good quality lower bounds. We believe that the identification of promising routing alternatives could be useful to further improve the performance of the proposed heuristics. However, a main drawback of the approach proposed by [Meng and Zhou (2014)] is the granularity of time in the timespace discretization of track occupancy. This approach applied to the rtRTMP studied in our paper could result in a huge variable space, since a level of detail of seconds is mandatory to model the occupation of each track-circuit.

Despite the effort done in decomposing the problem and developing strategies to efficiently manage the problem variables, the rtRTMP is still difficult to solve in a short computation time. Most of the existing approaches are able to provide good quality solutions only for simplified infrastructure configurations, a small number of trains, and simple traffic patterns. Therefore, there is a still need for developing more sophisticated and efficient methodologies to improve the solution process. This paper addresses one of the most under-investigated problems in the related literature that is the definition of a subset of routing alternatives for each train in the rtRTMP (i.e., the rtTRSP). In most of the approaches previously described, this is the starting point of the optimization process. However, the rtTRSP is often solved in a very simplified way, e.g., a number of alternatives is selected based on a-priori [Caimi et al., (2011)] or random decisions [Pellegrini

et al., (2015)].

4.3 Problem description

A rail infrastructure is composed of sequences of *track-circuits*, which are grouped into *block-sections*. A track-circuit is the single part of an infrastructure where the presence of a train is automatically detected. A block-section is a sequence of track-circuits between two consecutive signals. These signals are used to communicate to train drivers the speed profile to adopt based on the status of the block-sections ahead. To provide clear signal to the driver, all the involved track-circuits must be reserved for the train itself before it can enter a sequence of block-sections, also allowing some additional time for preparing the routing, i.e., *routing formation*. After a train exits a track-circuit, its reservation is still active for the so-called *release time*, i.e., the time in which a track-circuit is released.

The travel of a train through a track-circuit or a block-section, according to the granularity of the model considered, is called *operation*. Depending on the infrastructure and the rolling stock composition, an operation requires a *travel time* to be completed, i.e., the time to process the infrastructure resource without encountering any other traffic. The clearing time is the time necessary for the tail of the train to clear the resource after the head has exited it. We name *utilization time* the sum of reservation, travel time and clearing time by a train on an infrastructure resource. If a train starts its routing at null speed, its travel time on the first track-circuit is accounted only from the time at which it starts moving. Its staying still on the track-circuit before that time is represented through reservation.

We call *network* the part of the infrastructure under study. A number of distinct alternatives of *routing* may exist in a network. A routing is a sequence of subsequent block-sections that leads from an entry point (i.e., the first resource of the routing) to an exit point (i.e., the last resource of the routing) in the network. A routing may include intermediate scheduled stops at stations. The travel times for the routing with intermediate stops includes the appropriate deceleration and acceleration times.

In a timetable, a particular routing is assigned to each train and it is called *default* or *timetable routing*. For each train, the scheduled departure and arrival times are defined for all the stations it stops at. A train is not allowed to depart from a station before its scheduled departure time, and it is late if arriving later than its scheduled arrival time. If the origin of the train is out of the network, its entrance time in the network can be computed based on the scheduled speed profile, and the train cannot enter the network earlier than this time. Similarly, if the destination of the train is out of the network, its exit time is computed and it is late if arriving at the limit of the network after it. Buffer times are carefully included in the timetable in order to absorb small perturbations. Even so, the timetable may become infeasible due to the arising of unexpected

events during operation, which may cause disturbances, e.g., delays on the entrance times of trains in the network, and disruptions, e.g., the unavailability of one or more track sections of the infrastructure. Due to these events, two or more trains might generate a *conflicting request* for the same track-circuit or the same sequence of tracks-circuits, i.e., they may ask the same resource(s) at the same time. Each conflicting request causes unscheduled waiting times and thus longer travel times. To minimize the impact of disturbances and disruptions, an efficient working timetable must be quickly computed and implemented in real-time.

We call *primary delays* the delays directly caused by the unexpected events arising during rail operations. The propagation of the primary delays may generate additional delays, called *secondary delays*, due to conflicting requests of shared resources. Part of the latter delays does not depend on the decisions made during the computation of the working timetable for the following reasons. Given a network, a train entering with a delay may be unable to recover all or part of it before its scheduled arrival times or its exit. This is the case when a train is late even if it travels at its maximum speed and it is ordered before all the other trains on common track sections. The delay suffered by the train in this case is named *unavoidable delay*. We name *consecutive delay* of a train the difference between its secondary delay and its unavoidable delay. The consecutive delay is caused by the solution of conflicting requests, and is thus connected to the routing, ordering, timing decisions taken in the working timetable.

4.3.1 The real-time railway traffic management problem

We call rtRTMP [Pellegrini et al., (2014)] the detection and solution of train conflicting requests during operations. Train routing, timing and ordering decisions are the problem variables to be fixed in order to achieve a feasible working timetable. A working timetable is feasible when a precedence between trains is chosen for each train conflicting request such that each resource is utilized by at most a train at a time and no deadlocks emerge in the network, i.e., there are no trains circularly waiting for each other in the network.

The rtRTMP considers a railway network and a set of trains required to traverse it in a given time period. For each train a single routing must be selected among a set of possible alternatives. Alternative routings must have the same entry and exit points unless they correspond to a station platform. In this case the other platforms of the same station may be alternative entry/exit points. Moreover, it is generally assumed that all alternative routings have to pass in the stations where the train is required to stop, according to its timetable. Each station operation has to be scheduled such that the train departure time is respected. The scheduling decisions are the train timing and ordering decisions.

Regarding the objective of the rtRTMP, train delays are usually minimized. However, this can be translated into different objective functions. While not the only ones existing, the objective functions related to train delays can be classified based on i) the type of delays considered, e.g., secondary [Meng

and Zhou (2014)] or consecutive [Corman et al., (2010), D'Ariano et al., (2007a)]; ii) what function of the delay is minimized, e.g., maximum [Corman et al., (2014a), D'Ariano et al., (2014)] or total [Pellegrini et al., (2015), Rodriguez (2007)]; iii) the consideration of weights for the different types of trains, [Lamorgese and Mannino (2015), Törnquist (2012)]. The rtRTMP studied in this paper minimizes the total consecutive delays with no distinction between train delays. These delays are influenced by the train scheduling and routing decisions, i.e., by the routing assigned to each train and the ordering and timing decisions on each resource.

4.3.2 The real-time train routing selection problem

We now introduce some preliminary definitions. A *train routing assignment* is the assignment of a routing to a train among the existing routing alternatives. The train routing assignments for a pair of trains t and v are *coherent* in two cases: i) there are no rolling stock constraints between the two trains; ii) the two trains have a rolling stock constraint and the train routing assignment of the first train has the exit point in the same resource in which the train routing assignment of the second train has the entry point. The latter case models, e.g., turnaround, join or split constraints. A combination of train routing assignments is *feasible* if the train routing assignments for each pair is coherent. We call Γ the set of all feasible combinations of train routing assignments.

The real-time Train Routing Selection Problem (rtTRSP) is the problem of defining for each train a subset of routings selected among the existing routing alternatives available for each train. Each train routing assignment in the subset must be part of a feasible combination of train routing assignments. An rtTRSP solution S is therefore contained in Γ.

We let p be the maximum cardinality of the subset of routings selected for each train, with p predefined parameter. In this paper, the best value of p is investigated for each network under study. To assess the influence of the selection of p routing alternatives for each train on the original rtRTMP, the rtTRSP evaluates the subsets of routing alternatives by using the concept of *potential delays*: they are the estimation of the propagation of train delays in the network for the given train routing assignments. These estimations consider implicitly the train timing and ordering decisions. Finding the optimal solution S^* of the rtTRSP consists of selecting the train routing subsets with minimal potential delays.

The potential delays shall be defined in such a way that they take into account as much as possible the objective function of the rtRTMP, since the rtTRSP solution is an input data for the rtRTMP (as illustrated in the activity diagram of Figure 4.1). Therefore, the objective function considered in the rtRTMP shall guide the choice of the objective function for the rtTRSP. In this work, potential delays are based on consecutive delays and the rtTRSP objective is the minimization of their sum.

4.4 Problem formulation

This section is divided into two parts. The first part presents a brief description of the mathematical formulation of the rtRTMP; a more detailed description is provided in Appendix. The second part introduces the mathematical formulation of the rtTRSP and provides an illustrative example.

4.4.1 The rtRTMP formulation

We consider the rtRTMP modeled using the MILP formulation presented in [Pellegrini et al., (2014),Pellegrini et al., (2015)]. The MILP formulation models the network microscopically where each resource represents a single track-circuit. A route-lock sectional-release interlocking system is employed. Timing decisions are modeled using non-negative continuous variables. In particular, they account for the start and end times of track-circuit utilization by the trains and the minimum and actual travel times of a track-circuit for a train along a particular routing. Binary variables model train routing decisions, as well as train scheduling decisions. For the routing decisions, the binary variables model the choice of a particular routing for a train, among its alternatives. For the scheduling decisions, the binary variables model the order in which two trains utilize a track-circuit belonging to both their routings.

The MILP formulation presents a series of constraints which can be classified in three distinct groups:

1. Constraints concerning the traveling of the trains in the network. Supposed that a train uses exactly one of its routing alternatives to traverse the network, these constraints impose the minimum entrance times in the network and the minimum departure times from stations, considering the scheduled times and possible primary delays. They also impose the minimum travel times on all track-circuits;

2. Constraints concerning the change of rolling stock configuration. In this model, turnaround, join or split of trains are taken into consideration. This translates into sets of constraints that impose time and space coherence between trains concerned by these changes of rolling stock configuration. In particular these constraints model the minimum separation time required between the trains arrival and departure. Both have to happen on the same track-circuit;

3. Constraints concerning the capacity of the network. These constraints model the route-lock sectional-release interlocking system. We remind that only one train at a time can utilize a block-section, unless the trains are involved in a rolling stock configuration change within the block-section itself.

All three groups of constraints depend on the routing binary variables. In the first group, a set of constraints models how each train uses exactly one routing to travel in the network. Furthermore, the travel

time on a track-circuit depends not only on the track-circuit itself, but also on the routing along which it is used. For example, a train with a particular routing could be required to brake on a track-circuit due to a switching requirement, while this may not happen for all the other routings of this train, even if they include the same track-circuit. Thus, the binary variables indicating the routing decision appear in the constraints concerning the travel times, setting them to 0 along all the not chosen routings.

In the second group of constraints, the relation between the routings of the arrival and the departure trains has to be satisfied. In particular, the last track-circuit of the arriving train routing has to be the same of the first track-circuit of the departing train routing.

In the third group of constraints, the use of the binary variables stating the routing decision is due to the fact that, indeed, the utilization time is a function of the travel time, which necessitates of these variables.

Since the routing binary variables appear in all groups of constraints, it becomes evident how diminishing the number of routing decisions strongly affects the size of the overall problem: not only the number of routing variables decreases, but also the variables of all timing decisions pertaining the discarded routings can be erased, together with the constraints on all three groups in which only those variables would have appeared.

The objective function considered in this paper for the rtRTMP is the minimization of the total consecutive delays suffered by trains at the exit points.

4.4.2 The rtTRSP formulation

We model the rtTRSP by means of a *construction graph* $G = (C, L)$, in which C is the set of components and L is the set of links. In Figure 4.2, each component is depicted as a black circle, while each link is depicted as a non-oriented line between two black circles.

The set of components C is divided into n disjoint subsets of components. Each subset of components $T_t \subset C$ represents the routing alternatives for a train t and there are n trains in the network. A component $c_i \in T_t$ is a train routing assignment for train t, where $i = 1, ..., |T_t|$.

Let us consider two components $c_i \in T_t$ and $c_j \in T_v$, with $t, v \in \{1, ..., n\}$ and $t \neq v$, a link $\{c_i, c_j\} = l_{ij} \in L$ (highlighted in green in Figure 4.2) exists if and only if the corresponding train routing assignments are coherent.

A clique s of size n in G represents a feasible combination of train routing assignments. The set of all possible cliques of size n in G models the set Γ of all feasible combinations of train routing assignments. A subset $S \subset \Gamma$ is an rtTRSP solution.

A clique $s \in \Gamma$ is defined as follows:

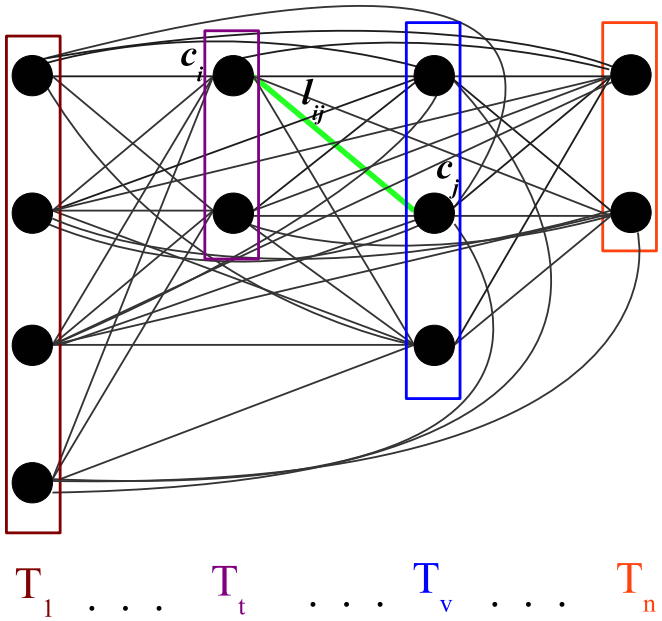


Figure 4.2: Example of a construction graph $G = (C, L)$

$$s = \left\{ \begin{array}{l} r \in \{0,1\}^{|C|} \\ z \in \{0,1\}^{|L|} \end{array} : \begin{array}{lll} \sum_{c_i \in T_t} r_i = 1 & \forall T_t \subset C & (1.a) \\ \sum_{c_i \in C} z_{ij} = (n-1)\ r_j & \forall c_j \in C & (1.b) \\ z_{ij} = z_{ji} & \forall l_{ij} \in L & (1.c) \end{array} \right\} \tag{4.1}$$

where r_i is a binary variable associated with component $c_i \in C$ and stating whether c_i has been selected in a feasible combination of train routing assignments ($r_i = 1$) or not ($r_i = 0$); z_{ij} is a binary variable associated with each link $l_{ij} \in L$ and stating whether link l_{ij} are selected ($z_{ij} = 1$) or not ($z_{ij} = 0$).

Constraints $(1.a)$ ensure the selection of exactly one train routing assignment for each train t among its $|T_t|$ routing alternatives. Constraints $(1.b)$ ensure the selection of exactly $n-1$ links incident to component c_j (i.e., $\sum_{c_i \in C} z_{ij} = n-1$) if $r_j = 1$, no link incident to component c_j is selected (i.e., $\sum_{c_i \in C} z_{ij} = 0$) if $r_j = 0$. Constraints $(1.c)$ ensure that all links are not-oriented.

The clique cost f_s related to a feasible combination of train routing assignments s is computed as follows:

$$f_s = \sum_{c_i \in C} u_i\ r_i + \sum_{l_{ij} \in L} w_{ij}\ z_{ij} \tag{4.2}$$

where u_i is a non-negative cost associated with component c_i and w_{ij} is a non-negative cost associated with

link l_{ij}. The costs u_i and w_{ij} indicate the undesirability of selecting the particular component c_i (i.e., a particular routing assignment for a train) respectively for itself or in combination with c_j. We will investigate different strategies to compute the costs associated with components and links in Section 4.5.

The rtTRSP corresponds to the following formulation:

$$\begin{aligned} & min \sum_{s \in \Gamma} f_s \; q_s \\ & s.t. \\ & \sum_{s \in \Gamma} q_s = m \end{aligned} \tag{4.3}$$

where q_s is a binary variable associated to a clique $s \in \Gamma$ and stating whether a clique s is part of an rtTRSP solution S ($q_s = 1$) or not ($q_s = 0$). We recall that p is the maximum cardinality of the subset of routings selected for each train. The constraints ensure that there are exactly m different feasible combinations of train routing assignments in S, where $m \geq p$ in order to cover multiple selections of the same component in different $s \in S$. An rtTRSP solution S thus corresponds to a set of m cliques in the construction graph G.

An example of rtTRSP

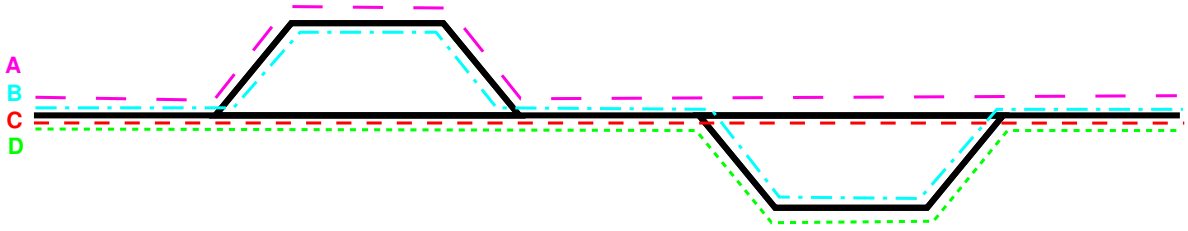


Figure 4.3: A toy network

We present an illustrative example with four trains (t_1, t_2, t_3, t_4) and four routings (A, B, C and D) available in the network (see Figure 4.3). However, there is a limited set of routing alternatives for each train: t_1 can use all four routings, t_2 and t_4 only routings C and D and t_3 routings B, C, D.

The construction graph G associated with the example has $\{c_1, c_2, c_3, c_4\} \in T_1$, $\{c_5, c_6\} \in T_2$, $\{c_7, c_8, c_9\} \in T_3$, $\{c_{10}, c_{11}\} \in T_4$. All train routing assignments are coherent, thus $|L| = 44$.

Table 4.1 presents an rtTRSP solution (rows 2, 3 and 4) and the relative input for the rtRTMP (row 5) for $m = 3$ and $p = 2$. For instance, a feasible combination of train routing assignments is shown in row 2 (e.g., $s_1 = \{c_1, c_5, c_7, c_{10}\}$). Each column shows the component chosen for the relative train in each of the m feasible combinations of train routing assignments among the $|\Gamma| = 48$ possible assignments. Regarding the input for the rtRTMP, the routing subset selected from S for t_1 contains only the routings A and B

(corresponding to the selected components c_1 and c_2), thus limiting the number of routing alternatives to be considered by the rtRTMP algorithm. A similar reasoning is applied to the other three trains.

Table 4.1: Example of constructing an rtTRSP solution

S	T_1	T_2	T_3	T_4
s_1	$\mathbf{c_1}$	$\mathbf{c_5}$	$\mathbf{c_7}$	$\mathbf{c_{10}}$
s_2	c_1	$\mathbf{c_6}$	$\mathbf{c_8}$	c_{10}
s_3	$\mathbf{c_2}$	c_5	c_8	$\mathbf{c_{11}}$
	$\mathbf{c_1, c_2}$	$\mathbf{c_5, c_6}$	$\mathbf{c_7, c_8}$	$\mathbf{c_{10}, c_{11}}$

Figure 4.4 depicts the three feasible combinations of train routing assignments of Table 4.1. The selected components of each feasible routing assignment for each train (e.g., s_1 is shown in the construction graph G on the left of Figure 4.4) are colored in black, while the other components are colored in grey. Only the links between the selected components are depicted.

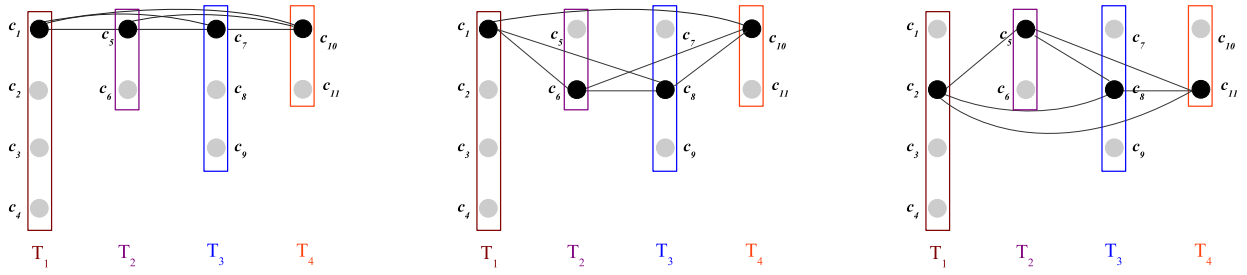


Figure 4.4: Feasible combination of train routing assignments in G for s_1 (left), s_2 (centre), s_3 (right)

4.5 The rtTRSP cost definition

This section describes the methods developed to define the strategies to compute the costs associated with components and links in the construction graph. These costs are based on the potential delay computed for each train routing assignment. We define the potential delays based on train routing and scheduling decisions. These two types of decisions contribute independently on the definition of the potential delays.

The potential delay due to a train routing assignment is computed as the non-negative difference between the travel time of the train when it uses the assigned routing and when it uses the timetable routing. The cost u_i of component c_i represents the potential delay due to the corresponding train routing assignment.

The potential delay due to train scheduling is based on the train ordering decisions between pairs of coherent train routing assignments. For each pair of coherent train routing assignments c_i and c_j, the cost w_{ij} of link l_{ij} represents the potential delay due to the corresponding train ordering decision. This potential

delays is computed based on the consecutive delays measured when the train using c_i follows the train using c_j or vice versa. Specifically, we distinguish between two cases: the two trains travel in the same direction or in opposite directions. If the two trains travel in opposite directions, we identify the train conflicting request (if any) with the highest number of common resources. On this train conflicting request, we order the two trains such that the consecutive delay is minimum. The latter delay represents the potential delay due to this scheduling decision.

If the two trains travel in the same direction, the following two steps are required for the computation of the potential delay: (i) a selection of the set of common resources between the two train routing assignments, (ii) an estimation of the potential delay on the selected set of common resources.

For step (i), we conceive the following alternative strategies based on the train conflicting requests between the two train routing assignments:

- **All** - the set contains all the resources for which train conflicting requests exist;

- **Min** - the set contains the resource k in each conflicting request with the *minimum maximum utilization time*, computed as follows. For each conflicting request, we first identify the resources k^{c_i} and k^{c_j} requiring the maximum utilization time for train routing assignments c_i and c_j respectively. If $k^{c_i} = k^{c_j}$, this resource is included in the set. Otherwise, $k^{c_i} \neq k^{c_j}$ and the resource with the minimum utilization time between k^{c_i} and k^{c_j} is included in the set;

- **Max** - the set contains the resource k in each conflicting request with the *maximum maximum utilization time*. This strategy differs from the previous one only if $k^{c_i} \neq k^{c_j}$, in which case the resource with the maximum utilization time between k^{c_i} and k^{c_j} is included in the set.

For step (ii), we use the set of resources selected in step (i) and estimate the potential delay on those resources for each pair of coherent train routing assignments by one among these three strategies:

- **Train** - We first define the potential delay for each train of the pair, and then define the potential delay of the pair as follows. For each selected resource, we compute the consecutive delay of each train when it waits for the other train. The potential delay for each train is the maximum among its consecutive delays over all the selected resources. The potential delay of the pair is the minimum between the potential delays of the two trains;

- **ResMin** - We first define the potential delay for each resource in the set of selected resources, then define the potential delay of the pair as follows. For each selected resource, we compute the consecutive delay of each train when it waits for the other train. The potential delay for each resource is the

maximum between the consecutive delays of the two trains on that resource. The potential delay of the pair is the minimum among the potential delays for all selected resources;

- **ResMax** - The same as for ResMin, except that the potential delay for each resource is the minimum between the consecutive delays of the two trains on that resource, and the potential delay of the pair is as the maximum among the potential delays for all selected resources.

For $l_{ij} \in L$, let $\tilde{w}_{ij}$ be the potential delay due to train ordering decisions computed as described above, and let k_{ij} be a resource where the potential delay is estimated for the components c_i and c_j. If $\tilde{w}_{ij} > 0$, we set $w_{ij} = \tilde{w}_{ij}$ in the construction graph G. Otherwise, we distinguish the two cases in which c_i and c_j do or do not have common resources. If c_i and c_j have at least one common resource, some unpredicted train conflicting requests could arise when solving the rtRTMP. We therefore set $w_{ij} = 1$. If the train routing assignments do not share resources, there will never be a conflicting request and $w_{ij} = 0$.

In Section 4.7, we will combine the three strategies developed for step (i) with the three strategies developed for step (ii) and select the best rtTRSP cost definition through experimental analysis.

4.5.1 An example of the rtTRSP cost definition

This section shows how to define rtTRSP costs for the illustrative example of Section 4.4.2 by means of the two-step procedure of Section 4.5. Specifically, we compute the three strategies of steps (i) and (ii) for the two trains (t_1 and t_3) traversing the network in the same direction. Figure 4.5 shows the routing assignments of the two trains ($c_1 \in T_1$ and $c_7 \in T_3$), that would generate two conflicting requests: the first one on the resources res_1, res_2 and res_3, the second one on res_5. For the sake of clarity, we assume in this example that all reservation times and clearing times are null. Figure 4.5 shows the start of the utilization time of c_1 (c_7) on the top (bottom) left of each resource.

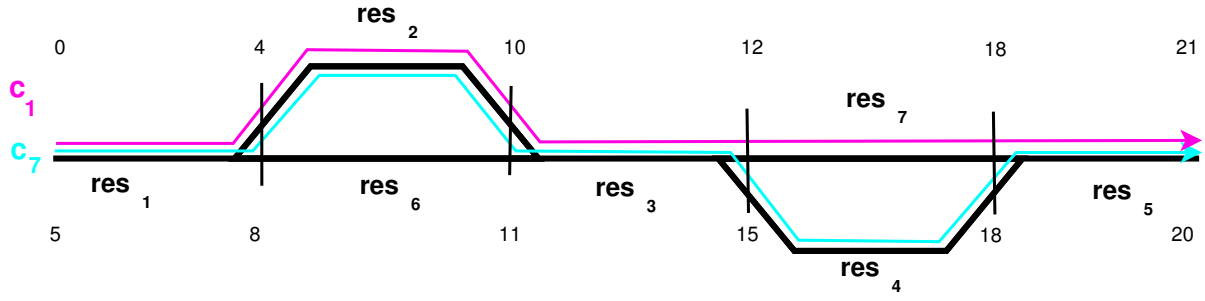


Figure 4.5: Example of two train routing assignments for the example of Section 4.4.2

The cost w_{17} of the link l_{17} connecting the components c_1 and c_7 is computed as follows. Step (i) requires

the computation of the following quantities with respect to the three proposed strategies:

- **All** - The set of the selected resources is formed by res_1, res_2, res_3 and res_5;
- **Min** - We analyze each of the two conflicting requests. For the first one, the resource with the maximum utilization time is: res_2 for c_1 ($k^{c_1} = \text{res}_2$, with a duration of $10 - 4 = 6$); res_3 for c_7 ($k^{c_7} = \text{res}_3$, with a duration of $15 - 11 = 4$). Being $k^{c_1} \neq k^{c_7}$, we select the resource with the minimum utilization time, i.e., $k = \text{res}_3$. For the second conflicting request, the only conflicting resource is res_5, therefore $k = \text{res}_5$. The set of the selected resources is formed by res_3 and res_5;
- **Max** - As for the Min strategy, $k^{c_1} = \text{res}_2$ and $k^{c_7} = \text{res}_3$. In addition, $k_{c_1} \neq k_{c_7}$. Between the two train conflicting resources we select the one with the maximum utilization time, i.e., $k = \text{res}_2$. The set of selected resources is formed by res_2 and res_5.

For the explanation of step (ii), we consider the set of selected resources computed by the Min strategy. The potential delay of l_{17} can be computed by one of the three proposed strategies:

- **Train** - For t_1, the consecutive delay generated when waiting t_3 is 5 $(15 - 10)$ on res_3 and 2 $(20 - 18)$ on res_5. The potential delay of t_1 is thus 5 on res_3. For t_3, the consecutive delay generated when waiting t_1 is 1 $(12 - 11)$ on res_3 and 3 $(21 - 18)$ on res_5. The potential delay of t_3 is thus 3 on res_5. The potential delay of the pair is the minimum, i.e., $\tilde{w}_{17} = 3$ and $k_{17} = \text{res}_5$;
- **ResMin** - For res_3, t_1 would have a consecutive delay of 5, t_3 of 1. The potential delay on res_3 is 5. For res_5, t_1 would have a consecutive delay of 2, t_3 of 3. The potential delay on res_5 is 3. The potential delay of the pair is $\tilde{w}_{17} = 3$ and $k_{17} = \text{res}_5$;
- **ResMax** - For res_3, t_1 would have a consecutive delay of 5, t_3 of 1. The potential delay on res_3 is 1. For res_5, t_1 would have a consecutive delay of 2, t_3 of 3. The potential delay on res_5 is 2. The potential delay of the pair is $\tilde{w}_{17} = 2$ and $k_{17} = \text{res}_5$;

In all three cases of step (ii) $\tilde{w}_{17} > 0$, thus $w_{17} = \tilde{w}_{17}$. In the whole example, all train routing assignments share at least one resource, therefore each link with $\tilde{w}_{ij} = 0$ has value $w_{ij} = 1$ in the construction graph.

4.6 The ACO-rtTRSP algorithm

This section presents a meta-heuristic algorithm which tackles the rtTRSP. The proposed algorithm is inspired by an ACO algorithm originally developed for the maximum clique problem [Solnon and Bridge (2006a)]. The reasons behind the choice of the ACO meta-heuristic are the two-fold: 1) the rtTRSP is

similar to a subset selection problem and the ACO algorithm performs well for this type of problem when compared with other heuristic approaches [Solnon and Bridge (2006b)]; 2) ACO fits particularly well the real-time and combinatorial nature of the rtTRSP and computes multiple good quality solutions in a very short computation time.

ACO is a meta-heuristic inspired by the foraging behaviour of ant colonies. Solutions are incrementally constructed by each of the $nAnts$ ants of a colony. At each step of the solution construction process, an ant a selects a new solution component probabilistically using the random proportional rule, which is based on *pheromone trails* and *heuristic information* (i.e., the colony's shared knowledge and a greedy measure on the quality of a component which may be added to the current partial solution s_a). Once all the ants of the colony have built a feasible solution, the best one is stored and the pheromone trails are updated accordingly. This process is repeated iteratively until the available computation time has elapsed.

We now describe the ACO algorithm developed for the rtTRSP, in which we apply the following slight abuse of terminology in order to be consistent with the literature on ACO algorithms. An ACO-rtTRSP solution is a set of n coherent train routing assignments, while an rtTRSP solution is derived from the best m ACO-rtTRSP solutions as follows. The routing subset of each train includes the timetable routing plus the $p-1$ routing alternatives belonging to the best ACO-rtTRSP solution. Since the routing subset must not include duplicates, we replace the routings which appear more than once in these best solutions with randomly chosen ones.

The ACO algorithm works on the construction graph G as follows. For each ant a, the first component $c_i \in C$ is randomly selected and is inserted in s_a. The set of *candidates* among which the ant can choose the next component $c_j \in C$ includes all the components linked to c_i, i.e., all coherent train routing assignments. The component c_j is chosen with a probability computed via the random proportional rule, that is based on the following pheromone trails and heuristic information. The value of the heuristic information associated to each $l_{ij} \in L$ is $\eta(l_{ij}) = \frac{1}{1+w_{ij}+u_j+IN_{ij}}$, stating the desirability of selecting c_j when considering c_i and IN_{ij}. IN_{ij} is a counter of the number of links which connect the pairs of components (c_h, c_z) for which $k_{hz} = k_{ij}$ (the resources where the potential delays is estimated on both links is the same) in the current partial solution s_a. We remark that the higher is IN_{ij}, the smaller is $\eta(l_{ij})$. The use of IN_{ij} in the formula favors the use of links with no new conflicting requests on the resources having already a high number of conflicting requests in s_a.

After each addition of a component c_j in the partial solution, the set of candidates is updated by removing both c_j and all components $c_h \in C : \nexists l_{jh} \in L$, i.e., all train routing assignments not coherent with the train routing assignment corresponding to c_j. The solution construction process terminates when the set of candidates is empty. A complete solution s_a is feasible only if it includes n components, where n is

the number of trains in the network.

Among the $nAnts$ solutions found in the current iteration, let $BestS$ be the feasible solution with minimum cost. After each iteration, the following local search is applied to $BestS$ to try to improve it: a solution component c_d in $BestS$ is replaced by another component c_e such that $c_d, c_e \in T_t$ and c_e has a link with each of the other components in $BestS$, and it minimizes the clique cost $CostBestS$ of $BestS$. To select the component c_d, we consider three possible local search strategies ($localSearchStr$):

- **Random** - a random selection;
- **Cost** - the selection of the component having $max\{u_d + \sum_{b \in BestS} w_{db}\}$, i.e., the one impacting most on $CostBestS$;
- **k** - the selection of the component that maximizes the number of links l_{db} such that $k_{db} = \hat{k}$, with $\hat{k}$ being the resource having the highest number of conflicting requests in the clique.

At the end of each iteration, an update of the pheromone trails is required. The ACO-rtTRSP algorithm is based on the MAX-MIN Ant System (Stützle and Hoos, 2000), in which upper and lower bounds are imposed on the pheromone trails. According to [Solnon and Bridge (2006a)], we use $\tau Min = 0.01$ and $\tau Max = 6$. During the update of the pheromone trails, some pheromone evaporates from all links of G and some additional pheromone is deposited on the links belonging to $BestS$, following the *clique pheromone strategy* [Solnon and Bridge (2006a)].

The m best ACO-rtTRSP solutions are stored in an ordered solution set $PoolBestSol$ as follows. The position of a solution in $PoolBestSol$ can be based on two ordering criteria: 1) its clique cost, being the best solution the one with minimum clique cost; 2) a lexicographic bi-objective function: the first objective is the minimization of the clique cost, the second is the minimization of the number of conflicting requests between train routing assignments on a resource, being the best solution the one with minimum number.

The algorithm terminates the search when a given time limit of computation $time_{max}$ seconds is reached or when all m solutions have a null cost. The pseudo-code of the ACO-rtTRSP algorithm is reported in Figure 4.6.

The ACO-rtTRSP parameters are next summarized:

- α, the influence of pheromone trails in the random proportional rule;
- β, the influence of heuristic information in the random proportional rule;
- $nAnts$, the number of ants in the colony;
- ρ, the pheromone evaporation rate;

ACO-rtTRSP ALGORITHM
Input: G, α, β, $nAnts$, ρ, $localSearchStr$, $count$, $biObj$, $time_{max}$, n
Set pheromone trails $\tau(l_{ij}) = \tau_{Max} \quad \forall l_{ij} \in L$
while $(time \leq time_{max})$ **do**
  $CostBestS \leftarrow$ MAX INT
  for $(a = 0; a < nAnts; a{+}{+})$ **do**
    Randomly choose $c_i \in C$
    $s_a \leftarrow \{c_i\}$
    $Candidates \leftarrow \{c_j \in C : \exists l_{ij} \in L\}$
    while $(Candidates \neq \emptyset)$ **do**
      choose $c_j \in C$ with probability $\frac{\tau(c_j,s_a)^\alpha \eta(c_j,s_a)^\beta}{\sum_{c_h \in Candidates} \tau(c_h,s_a)^\alpha \eta(c_h,s_a)^\beta}$
      $s_a \leftarrow s_a \cup \{c_j\} \cup \{l_{ij} : c_i \text{ in } s_a\}$
      $Candidates \leftarrow$ Remove $c_j \cup \{c_h \in C : \nexists l_{jh} \in L\}$
    end while
    if $(f_{s_a} < CostBestS$ & s_a feasible$)$ **then**
      $BestS \leftarrow s_a$
      $CostBestS \leftarrow f_{s_a}$
    end if
  end for
  Apply LocalSearch($localSearchStr$) to $BestS$
  Update pheromone trails $\tau(l_{ij})$ in G by using *clique pheromone strategy*
  Store $BestS$ in the ordered solution set $PoolBestSol$
end while
Output: Return $s_a \in PoolBestSol$ with $a = 1, ..., m$

Figure 4.6: Pseudo-code of the ACO-rtTRSP algorithm

- $localSearchStr$, the strategy used to select the component on which the local search is performed;
- $count$, determining if the counter IN is used (T, True) or not (F, False) in the computation of heuristic information;
- $biObj$, determining if $PoolBestSol$ is generated by using the ordering criterion 1 (F) or 2 (T).

The next section will describe how we set these parameters based on a stat-of-the-art automatic tuning procedure.

4.7 Computational experiments

This section presents the computational results on the rtTRSP and rtRTMP. The rtTRSP is solved by the ACO-rtTRSP algorithm of Section 4.6 or by a random approach, while the rtRTMP is solved by exact and heuristic approaches of [Pellegrini et al., (2014), Pellegrini et al., (2015)]. All tests are performed with a Quad-Core Intel Xeon E5 3.7GHz processor with 32 GB RAM, under OS X 10.10.3.

The computational experiments have been performed in a laboratory environment for two French case

studies based on real-world data. The first case study concerns the 27-km-long railway line around the city of Rouen, shown in Figure 4.7, with 190 track-circuits, 189 block sections, and 11347 routings. This line presents interesting practical aspects: there are several intermediate stations, each one with up to six platforms; a part of the line is equipped to be traversed on both directions.

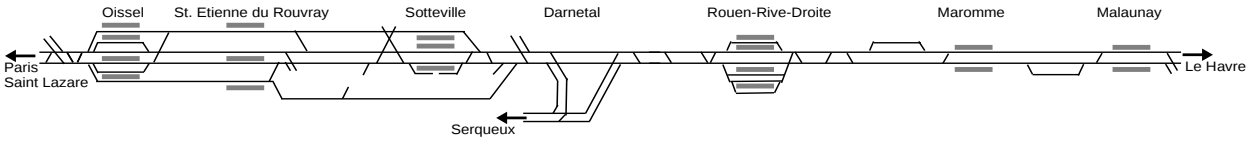


Figure 4.7: Rouen network

The second case study is the 12-km-long Lille station area shown in Figure 4.8, with 299 track-circuits, 734 block sections, and 2409 routings. The Lille station is a terminal station linked to national and international lines, with 17 platforms used by local, intercity and high speed trains.

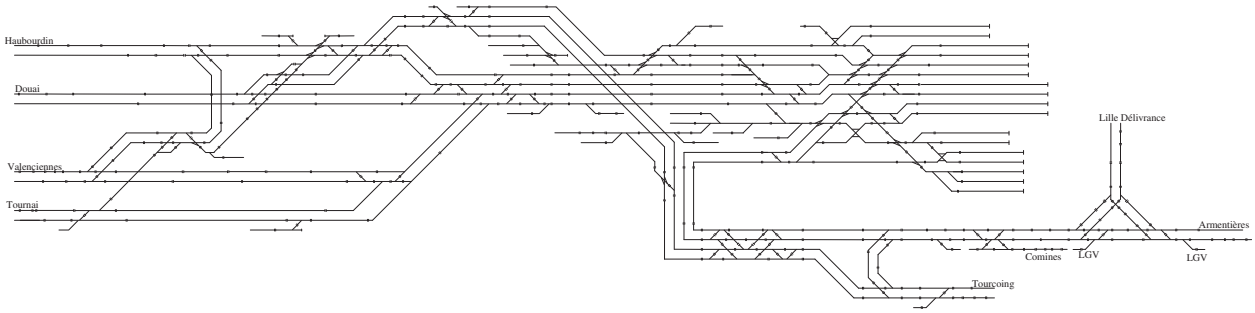


Figure 4.8: Lille network

A typical daily timetable has 186 trains for the Rouen case study and 509 for the Lille one. In our computational experiments, we deal with 21 scenarios for the Rouen case study and 15 scenarios for the Lille case study that are generated from a typical daily timetable as follows. In each scenario, 20% of the trains, randomly selected, are affected by a random delay between 5 and 15 minutes applied at their entry point. For each scenario, we generate 10 rtRTMP instances by considering all the trains that enter in the network within an hour, starting from 10 different time instants randomly taken during the traffic peak-hours (i.e., in the time periods 7:30-9:00 and 18:30-20:00). In total, we obtain 210 rtRTMP instances for the Rouen and 150 for Lille case study. In the Rouen instances, each train can have a maximum of 192 routing alternatives and the average number of trains is 13. In the construction graph G, the average number of components and links is $|C| = 597$ and $|L| = 152441$. In the Lille instances, each train can have a maximum of 100 routing alternatives and the average number of trains is 40. In the construction graph G, the average number of components and links is $|C| = 3881$ and $|L| = 6902867$ for the Lille instances.

In the remaining of this section, we present an assessment of the strategies to select the rtTRSP costs (Section 4.7.1), of the ACO-rtTRSP parameters (Section 4.7.2), of the ACO-rtTRSP algorithm (Section 4.7.3), of the algorithms to solve the rtTRSP and rtRTMP (Section 4.7.4).

4.7.1 Selection of the rtTRSP costs

This section reports a computational analysis on 30 Rouen instances dedicated to the selection of the best rtTRSP cost definition of Section 4.5. For each Rouen instance and for each cost definition, we generate 50 different rtTRSP solutions by using the ACO-rtTRSP algorithm for $m = 1$. In total, we have 1500 rtTRSP solutions for each cost definition. Starting from each of the 1500 rtTRSP solutions, we compute the optimal solution for the corresponding rtRTMP.

The 1500 rtTRSP and rtRTMP assessments are compared as follows. For each Rouen instance, we compute two rankings of the 50 solutions: one is based on the rtTRSP solutions and on the related objective function; another is based on the rtRTMP solutions and on the related objective function. The two rankings are compared in order to find the cost definition that presents the minimum absolute difference with respect to the rtRTMP assessment. This difference is obtained through the Wilcoxon rank-sum test with a confidence level of 0.95. We observe that the two rankings are significantly different for all nine cost definitions.

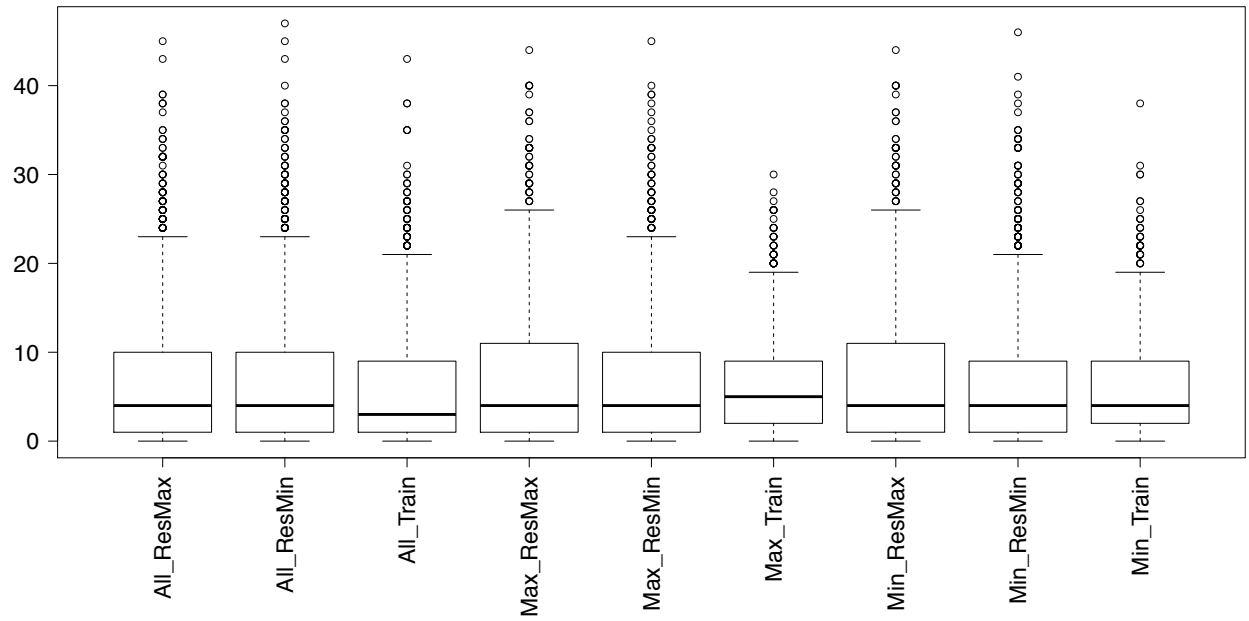


Figure 4.9: Boxplots of the difference in absolute value between the rtTRSP and rtRTMP rankings

Figure 4.9 shows a boxplot comparison of the rtTRSP cost definitions of Section 4.5 in terms of the of the

absolute value of the difference between the rtTRSP and the rtRTMP rankings. Each rtTRSP cost definition is named according to the scheme $\langle set \rangle$_$\langle estimation \rangle$. This scheme reflects the two steps of Section 4.5 used to estimate the potential delays. Specifically, $\langle set \rangle$ refers to the strategy used for step (i), i.e., the criterion (All, Max, Min) to select the set of common resources to be considered in the estimation; $\langle estimation \rangle$ refers to the strategy used for step (ii), i.e., the strategy (Train, ResMin, ResMax) to estimate the potential delay on the selected common resources. The thick horizontal line in the boxplot represents the median of the distribution, the extremes of the boxplot are the first and third quartiles, the dots are the outliers, and the whiskers show the smallest and the largest non-outliers in the data-set.

In addition to the boxplot comparison, we consider the value of the *pseudo-median* (i.e., an estimation of the population's location parameter that is closely related to its median) of the distributions: it gives an indication of what would be the median of the distribution of the absolute difference of the two solution rankings. We use this indication in combination with the boxplots in order to establish which is the best rtTRSP cost definition.

The cost definitions sharing the best minimum value (6) of the pseudo-median are: All_Train, Min_Train and Max_Train. We note that a pseudo-median of 6 means that we can expect to obtain a ranking which differs of at most 6, in absolute terms, in 50% of the cases. In order to select one among the three best cost definitions, we look at Figure 4.9. The best cost definition appears to be All_Train, since it has a median of 3 against 4 of Min_Train and 5 of Max_Train and is the best until the 75th percentile of the distribution, where the three best cost definitions are equivalent with an absolute difference of 9. In the next subsections, we consider All_Train set as the rtTRSP cost definition.

4.7.2 Tuning of the ACO-rtTRSP parameters

This section reports a computational analysis on 30 Rouen instances, different from the ones tested in Section 4.7.1, dedicated to the selection of the best combination of the ACO-rtTRSP parameters. The rtTRSP related to each Rouen instance is solved by the ACO-rtTRSP algorithm with a computation time limit of 30 seconds.

The ACO-rtTRSP parameters are tuned by using IRACE (Iterated Racing for Automatic Algorithm Configuration, [López-Ibáñez et al., (2011)]), in which the best parameter setting is selected through an iterated racing procedure. We set to 35000 the maximum number of runs to be performed in IRACE. Table 4.2 presents the settings tested for each parameter. We highlight in bold the settings selected.

Table 4.2: Parameter values considered during the tuning

α	β	$nAnts$	ρ	$localSearchStr$	$count$	$biObj$
1, **2**, 3, 4	**2**, 3, 4, 5	25, 50, 100, **150**, 200	0.02, **0.05**, 0.2, 0.7	**cost**, random, k	T, **F**	**T**, F

4.7.3 Convergence of the ACO-rtTRSP algorithm

This section gives information on the rapidity of the ACO-rtTRSP algorithm to converge to the best-known solution. We present average results on the remaining 150 Rouen instances and on the 150 Lille instances when varying the percentage of iterations performed by the algorithm in the given time limit of computation. We use the percentage of iterations because the time required by the algorithm to perform each iteration varies from instance to instance, and from test case to test case. Figure 4.10 shows the average variation of the objective function value computed as follows: (objective function value) - (best objective function value) / (best objective function value), expressed in percentage.

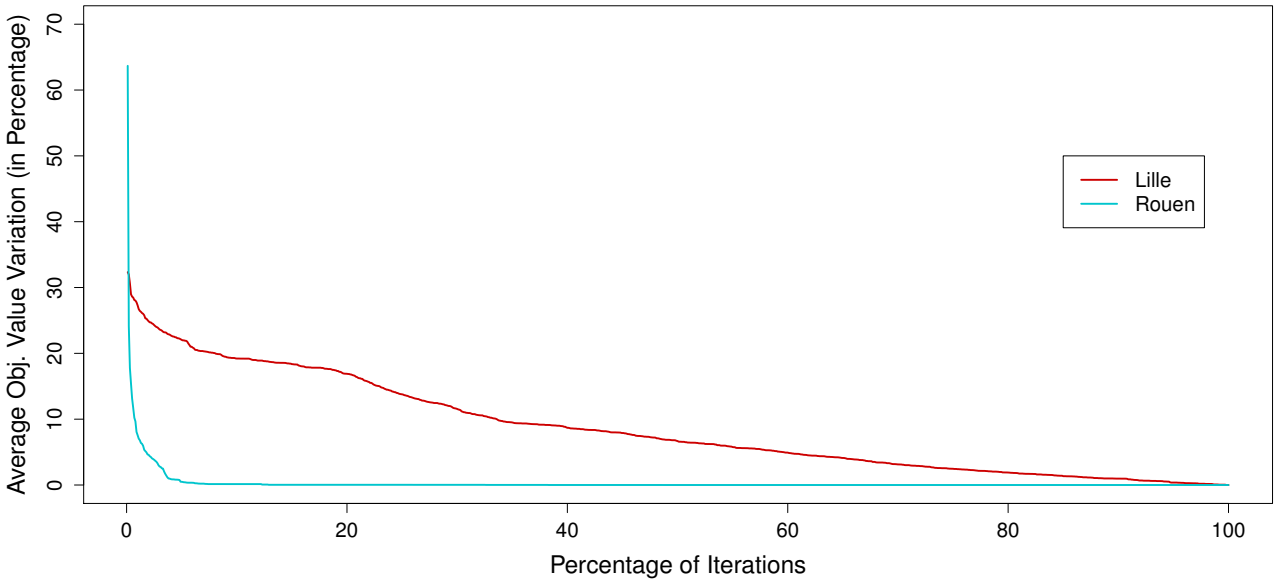


Figure 4.10: ACO-rtTRSP best solution value when varying the number of iterations

Comparing the results obtained in Figure 4.10 for the two test cases, the average convergence of the algorithm is significantly faster for the Rouen instances, since $|C|$ and $|L|$ are smaller in the corresponding construction graph G and the algorithm performs a high number of iterations in the given computation time. For the Lille instances, the algorithm takes a longer computation time per iteration and sometimes uses all the given time in order to find the best-known solution.

4.7.4 Comparison between algorithms

This section presents a computational comparison of rtTRSP and rtRTMP algorithms on the 150 Rouen instances and on the 150 Lille instances, also used in Section 4.7.3. Specifically, we use the ACO-rtTRSP

algorithm with the best value for each parameter of Section 4.7.2, the exact and heuristic rtRTMP approaches of [Pellegrini et al., (2014),Pellegrini et al., (2015)]. The two rtRTMP algorithms require a starting routing alternative for each train, that is used to compute an initial upper bound for the rtRTMP, and they may deal with a set of routing alternatives. In this experiments, the rtTRSP and rtRTMP algorithms are used to generate a working timetable via the following approaches:

1. *ALL ROUTINGS HEURISTIC*: All routing alternatives are given in input to the best heuristic algorithm in [Pellegrini et al., (2015)] that returns the best rtRTMP solution found within 180 seconds. The timetable routing is set as the starting routing alternative for each train;

2. *RND-rtTRSP*: An rtTRSP solution with p routing alternatives for each train is generated by fixing the starting routing alternative for each train as its timetable routing and by selecting randomly the remaining $p-1$ routing alternatives. This rtTRSP solution is given in input to the same algorithm of approach 1 that returns the best rtRTMP solution found within 180 seconds;

3. *ACO-rtTRSP*: An rtTRSP solution with p routing alternatives for each train is computed by the ACO-rtTRSP algorithm with a time limit of 30 seconds. This rtTRSP solution is given in input to the same algorithm of approach 1 that returns the best rtRTMP solution found within 150 seconds. The best ACO-rtTRSP routing for each train is set as its starting routing alternative;

4. *ALL ROUTINGS OPTIMUM*: All routing alternatives of each train are given in input to the exact approach in [Pellegrini et al., (2014),Pellegrini et al., (2015)] that returns the optimal rtRTMP solution. The timetable routing is set as the starting routing alternative for each train.

For the ALL ROUTINGS OPTIMUM approach, we used the exact approach in [Pellegrini et al., (2014), Pellegrini et al., (2015)] with no time limit of computation. All the 150 Rouen instances and 91/150 Lille instances were solved to optimality between around 1 and 3 hours of computation. Regarding the other 59 Lille instances, the algorithm was interrupted due to a lack of memory and returned the best solution found.

Figure 4.11 shows a barplot comparison between the above introduced approaches for the Rouen and Lille case studies. We consider the best average rtRTMP solution found by the ALL ROUTINGS HEURISTIC approach as the benchmark value, since this is our reference state-of-the-art approach to solve the rtRTMP with a computation time compatible with real-time operations. Each bar gives the percentage improvement of the average objective function value with respect to the benchmark value. For the RND-rtTRSP approach we consider the best values of p, i.e., 70 (10) for the Rouen (Lille) case study. For the ACO-rtTRSP approach we consider the cases: (i) the best values of p, i.e., 40 (10) for the Rouen (Lille) case study; (ii) the worst values of p, i.e., 192 (90) for the Rouen (Lille) case study; (iii) the average results obtained when varying

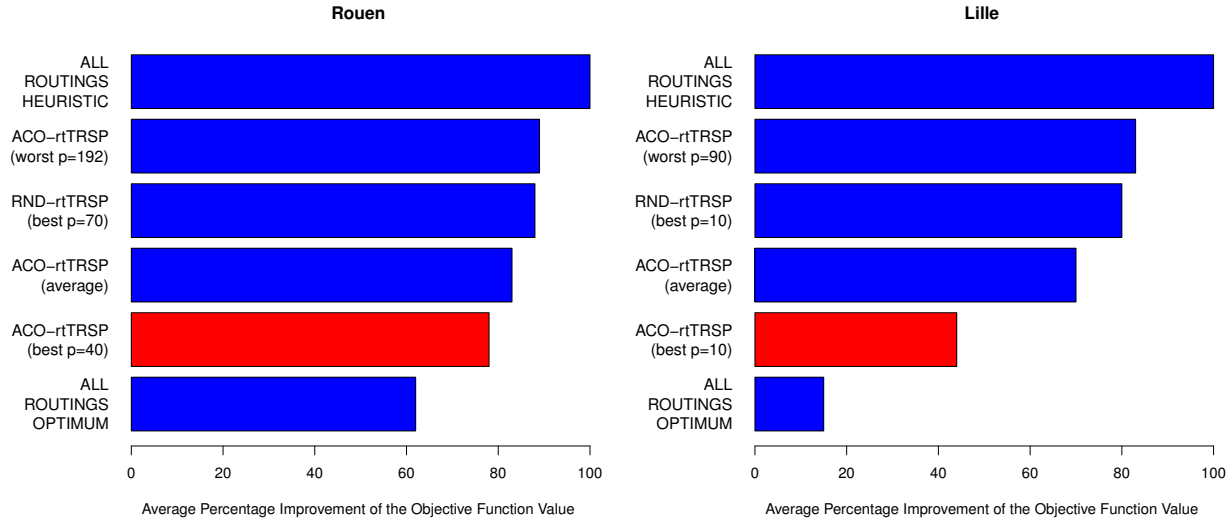


Figure 4.11: Barplots of the average percentage improvement with respect to the benchmark value

the values of p. The best and worst p values are obtained for the 300 instances of the two case studies, when varying p in the range [10, 20, 30,..., 170, 180, 190, 192] ([10, 20, 30,..., 80, 90, 100]) for the Rouen (Lille) instances. Tables 4.3 and 4.4 will give further information on the results with different p values.

The results of Figure 4.11 illustrate the improvement achievable when using the rtTRSP solutions as input for the rtRTMP heuristic RECIFE-MILP [Pellegrini et al., (2015)]. An improvement of around 12% (20%) is already obtained for the Rouen (Lille) instances when using random rtTRSP solutions, since the heuristic improves on average its performance on the reduced rtRTMP. A further improvement of around 10% (36%) is obtained for the Rouen (Lille) instances when using the ACO-rtTRSP algorithm for the best p values rather than the random approach, since the heuristic works on the reduced rtRTMP with an optimized subset of routing alternatives for each train. However, a potential improvement of around 16% (29%) can still be achieved for the Rouen (Lille) instances when looking at ALL ROUTINGS OPTIMUM. The potential improvement is larger for the Lille instances, since the central part of the Lille station is densely used in both traffic directions, many trains share the same track-circuits and the primary delay quickly propagates.

The results obtained by the ACO-rtTRSP algorithm for the worst p values and the average results obtained by the ACO-rtTRSP algorithm when varying the values of p show that even a *wrong* choice of the parameter p would still give an improvement over the ALL ROUTINGS HEURISTIC approach.

Tables 4.3 and 4.4 report a detailed comparison between the ACO-rtTRSP and RND-rtTRSP approaches on the 150 Rouen and Lille instances, respectively. The comparison is reported, on average, for each value of p in terms of the number of rtRTMP solutions that are optimal when considering all train routing alternatives,

Table 4.3: Comparison between ACO-rtTRSP and RND-rtTRSP on the Rouen results

p	# Opt. Sol. ACO-rtTRSP (/150)	# Opt. Sol. RND-rtTRSP (/150)	Obj. Value ACO-rtTRSP (sec)	Obj. Value RND-rtTRSP (sec)	Comp. Time ACO-rtTRSP (sec)	Comp. Time RND-rtTRSP (sec)
10	88	28	178.1	220.4	62.4	54.7
20	90	53	173.3	203.1	83.7	82.1
30	91	70	176.0	217.6	101.1	101.8
40	100	67	**168.3**	196.9	113.4	116.1
50	98	77	173.0	216.4	119.9	118.6
60	100	76	170.9	223.3	126.8	127.2
70	99	80	173.0	**189.2**	127.0	130.1
80	99	90	176.2	201.1	133.6	128.4
90	99	85	168.5	202.3	135.9	137.8
100	101	90	181.0	205.4	137.5	139.9
110	98	79	182.6	226.3	136.4	143.0
120	95	93	181.8	220.1	138.4	138.4
130	96	93	178.9	206.0	138.8	137.0
140	97	93	190.1	203.8	140.3	137.4
150	98	92	178.9	223.9	138.3	143.8
160	97	94	169.2	205.4	138.9	139.7
170	96	88	191.3	204.0	141.4	141.8
180	99	90	193.1	246.0	143.0	141.5
190	96	87	183.4	221.8	141.2	144.2
192	95	91	193.2	216.1	143.2	143.8

the objective function value and the computation time. For each case study and rtTRSP approach, the best average value of the objective function and the corresponding p value are highlighted in bold.

Table 4.4: Comparison between ACO-rtTRSP and RND-rtTRSP on the Lille results

p	# Opt. Sol. ACO-rtTRSP (/91)	# Opt. Sol. RND-rtTRSP (/91)	Obj. Value ACO-rtTRSP (sec)	Obj. Value RND-rtTRSP (sec)	Comp. Time ACO-rtTRSP (sec)	Comp. Time RND-rtTRSP (sec)
10	6	7	**615.1**	**1111.0**	160.4	162.8
20	10	6	731.9	1280.6	175.9	180.0
30	6	3	879.5	1375.5	180.0	180.0
40	6	3	953.5	1388.4	180.0	180.0
50	8	1	1008.4	1398.5	180.0	180.0
60	7	3	1047.0	1428.1	180.0	180.0
70	5	0	1062.4	1407.7	180.0	180.0
80	5	2	1077.1	1424.2	180.0	180.0
90	6	2	1145.1	1402.4	180.0	180.0
100	6	0	1098.7	1383.0	180.0	180.0

From the results of Tables 4.3 and 4.4 we have the following observations:

- The best p values are small values (especially for the ACO-rtTRSP approach): in the available com-

putation time, RECIFE-MILP does not manage to improve its performance when there is a too large number of routing alternatives for each train, even if the optimal routing alternatives are more likely to be considered for large p values;

- The ACO-rtTRSP approach is better than the RND-rtTRSP approach for any row of Tables 4.3 and 4.4. This result highlights the importance of optimizing the selection of the routing subset for each train.

- Looking at the case when all routing alternatives are considered (i.e., the last row of Tables 4.3 and 4.4), the ACO-rtTRSP approach is better than the RND-rtTRSP approach, and it properly chooses the starting routing alternative for each train. For the Rouen (Lille) case study, the starting rtRTMP upper bound is 233 (1493) when using the ACO-rtTRSP approach, while 1469 (1550) when using the RND-rtTRSP approach. Another added value of the ACO-rtTRSP approach for better solving the rtRTMP is therefore the selection of a good quality starting routing alternative for each train, that results to be better than the timetable routing in the studied perturbed scenarios and network configurations.

- The problem of finding a good quality rtRTMP solution in a short time becomes particularly challenging for large values of p, since the number of rtRTMP variables significantly increases. This trend motivates the focus of this paper in searching for the best selection of alternative routings for each train.

4.8 Conclusions and future research

This paper investigates systematically the improvement achievable in the solution of the rtRTMP when optimizing the selection of the starting routing alternative and the alternative routing subset for each train. This problem is named the rtTRSP and is important for the following reasons: 1) the dispatcher needs indications on how to re-optimize the routing of trains during disturbed operations; 2) the identification of the potential best routing alternatives for each train can improve the existing rtRTMP algorithms. Due to the real-time and combinatorial characteristic of the rtTRSP, this problem is modeled as an integer linear programming formulation and efficiently solved by an ACO algorithm in a short computation time. The computational results on two real-wold case studies of the French railways confirm the advantage of using the rtTRSP solutions, as input data for the solution of the rtRTMP, compared to a state-of-the-art rtRTMP algorithm. The largest advantage (up to 22% for the Rouen case study and up to 56% for the Lille case study) is obtained when solving the rtTRSP seeking a small number of suitable alternative routings for each train.

Future research shall be dedicated to further improve the quality of the rtRTMP solutions. This can be achieved by developing better quality combinations between rtRTMP and rtTRSP approaches, rtRTMP heuristics, rtTRSP cost definitions and ACO algorithmic settings. Other research directions shall be dedicated to the generalization of the ACO-rtRTMP approach to different objective functions and constraints, that may be be more appropriate when solving the rtRTMP for other types of traffic flows, disturbances and infrastructure configurations.

4.9 Appendix

In this Appendix, we detail the MILP formulation for the rtRTMP, introduced in [Pellegrini et al., (2014), Pellegrini et al., (2015)]. In the MILP formulation, we use the following notation:

- $tc_0 \equiv$ entry location in the network considered;
- $tc_\infty \equiv$ exit location from the network considered;
- $T \equiv$ set of trains;
- $ty_t \equiv$ type (rolling stock characteristics) of train t;
- $init_t \equiv$ earliest time at which train t can be operated, including the entrance delay if any;
- $exit_t \equiv$ maximum between the scheduled exit time of train t from the network and the earliest exit time computed by considering the given $init_t$, the timetable routing and the scheduled stops;
- $i(t', t) \equiv$ indicator function: 1 if t' and t use the same rolling stock and t results from the turnaround or join or split of t', 0 otherwise;
- $ms_{t',t} \equiv$ minimum separation time between the arrival and departure of two trains using the same rolling stock ($i(t', t) = 1$);
- $R_t \equiv$ set of alternative routings for train t;
- $TC^r \equiv$ set of track-circuits composing routing r;
- $OTC_{ty,r,tc} \equiv$ set of track-circuits occupied by a train t of type ty along r if the head of train t is at the end of track-circuit tc ($\emptyset$ if t is shorter than tc);
- $TC(tc, tc', r) \equiv$ set of track-circuits between tc and tc' along r;
- $p_{r,tc}, s_{r,tc} \equiv$ track-circuits preceding and following tc along r;

- $rt_{ty,r,tc}, ct_{ty,r,tc} \equiv$ travel and clearing time of tc along r for a train of type ty in absence of disturbances;
- $ref_{r,tc} \equiv$ reference track-circuit for the reservation of tc along r (it depends on the interlocking system);
- $bs_{r,tc} \equiv$ block-section including track-circuit tc along routing r;
- $for_{bs}, rel_{bs} \equiv$ formation time and release time for block-section bs;
- $e(tc, r) \equiv$ indicator function: 1 if track-circuit tc belongs to either the first block-section or the last block-section of r, 0 otherwise;
- $S_t, TCS_{t,s} \equiv$ set of stations where t has a scheduled stop, and set of track-circuits that can be used by t for stopping at s;
- $dw_{t,s}, d_{t,s} \equiv$ minimum dwell time and scheduled departure time for train t at station s;
- $\hat{TC}_{t,t',tc} \equiv$ set of track-circuits tc' which may be used by both t and t' such that if t precedes ($\prec$) t' on tc, $t \prec t'$ on tc', and so on (e.g., if the track-circuits follow each other on a straight track segment). $\hat{TC}_{t,t',tc} = \{tc\}$ if $tc \in \cup_{r \in R_t} TC^r \cap \cup_{r \in R_{t'}} TC^r$ and no implied precedence relation links tc to other track-circuits. $\hat{TC}_{t,t',tc} = \emptyset$ if $\exists tc' \in \cup_{r \in R_t} TC^r \cap \cup_{r \in R_{t'}} TC^r$ such that $tc \in \hat{TC}_{t,t',tc'}$, i.e., each track-circuit belongs to one and only one set $\hat{TC}_{t,t',tc}$;
- $M \equiv$ large constant.

The formulation uses non-negative **continuous variables**:

- for all triplets of $t \in T$, $r \in R_t$ and $tc \in TC^r$:

$$o_{t,r,tc} : \text{time in which } t \text{ starts the occupation of } tc \text{ along } r,$$
$$l_{t,r,tc} : \text{longer stay of } t\text{'s head on } tc \text{ along } r, \text{ due to dwell time and scheduling decisions (i.e., a consecutive delay);}$$

- for all pairs of $t \in T$ and $tc \in \cup_{r \in R_t} TC^r$:

$$sU_{t,tc}, eU_{t,tc} : \text{time in which } t \text{ starts and ends } tc \text{ utilization;}$$

- for all $t \in T$:

$$D_t : \text{consecutive delay suffered by train } t \text{ when exiting the network.}$$

In addition, the formulation includes **binary variables**:

- for all pairs of $t \in T$ and $r \in R_t$:

$$x_{t,r} = \begin{cases} 1 & \text{if } t \text{ uses } r, \\ 0 & \text{otherwise,} \end{cases}$$

- for all triplets of $t, t' \in T$ such that the index t is smaller than the index t', and $tc \in \cup_{r \in R_t} TC^r \cap \cup_{r \in R_{t'}} TC^r$ such that $\hat{TC}_{t,t',tc} \neq \emptyset$:

$$y_{t,t',tc} = \begin{cases} 1 & \text{if } t \text{ utilizes } tc \text{ before } t' \ (t \prec t'), \\ 0 & \text{otherwise } (t \succ t'). \end{cases}$$

The objective is the minimization of the total consecutive delays suffered by trains at their exit points:

$$\min \sum_{t \in T} D_t. \tag{4.4}$$

This objective function must be minimized while respecting the following constraints:

$$o_{t,r,s_{r,tc_0}} \geq init_t \, x_{t,r} \qquad \forall t \in T \quad r \in R_t \tag{4.5}$$

$$o_{t,r,tc} \leq M x_{t,r} \qquad \forall t \in T \quad r \in R_t \quad tc \in TC^r \tag{4.6}$$

$$o_{t,r,tc} = o_{t,r,p_{r,tc}} + l_{t,r,p_{r,tc}} + rt_{r,ty_t,p_{r,tc}} x_{t,r} \qquad \forall t \in T \quad r \in R_t \quad tc \in TC^r \setminus \{tc_0\} \tag{4.7}$$

$$o_{t,r,s_{r,tc}} \geq d_{t,s} x_{t,r} \qquad \begin{array}{r} \forall t \in T \quad r \in R_t \quad s \in S_t \\ tc \in TCS_{t,s} \cap TC^r \end{array} \tag{4.8}$$

$$l_{t,r,tc} \geq dw_{t,s} x_{t,r} \qquad \begin{array}{r} \forall t \in T \quad r \in R_t \quad s \in S_t \\ tc \in TCS_{t,s} \cap TC^r \end{array} \tag{4.9}$$

$$\sum_{r \in R_t} x_{t,r} = 1 \qquad \forall t \in T \tag{4.10}$$

$$D_t \geq \sum_{r \in R_t} o_{t,r,tc_\infty} - exit_t \qquad \forall t \in T \tag{4.11}$$

$$\sum_{\substack{r \in R_t, tc \in TC^r: \\ p_{r,tc} = tc_0}} o_{t,r,tc} \geq \sum_{\substack{r \in R_{t'}, tc \in TC^r: \\ s_{r,tc} = tc_\infty}} o_{t',r,tc} + (ms_{t',t} + rt_{r,ty_{t'},tc}) x_{t',r} \qquad \forall t, t' \in T : i(t',t) = 1 \tag{4.12}$$

$$\sum_{r\in R_t: tc\in TC^r} x_{t,r} = \sum_{r\in R_{t'}: tc\in TC^r} x_{t',r} \qquad \begin{array}{r} \forall t,t' \in T : i(t',t) = 1 \\ tc \in TC : \exists r \in R_t, tc = s_{r,tc_0} \end{array} \tag{4.13}$$

$$\sum_{\substack{tc\in\cup_{r\in R_t} TC^r: \\ \exists r\in R_t, p_{r,tc}=tc_0}} sU_{t,tc} \leq \sum_{\substack{tc\in\cup_{r\in R_{t'}} TC^r: \\ \exists r\in R_{t'}, s_{r,tc}=tc_\infty}} eU_{t',tc} \qquad \forall t,t' \in T : i(t',t) = 1 \tag{4.14}$$

$$sU_{t,tc} = \sum_{r\in R_t: tc\in TC^r} \left(o_{t,r,ref_{r,tc}} - for_{bs_{r,tc}} x_{t,r}\right) \qquad \begin{array}{r} \forall t \in T \quad tc \in \cup_{r\in R_t} TC^r : \\ (\nexists t' \in T : i(t',t) = 1) \vee (\forall r \in R_t : ref_{r,tc} \neq s_{r,tc_0}) \end{array} \tag{4.15}$$

$$sU_{t,tc} \leq \sum_{r\in R_t: tc\in TC^r} \left(o_{t,r,ref_{r,tc}} - for_{bs_{r,tc}} x_{t,r}\right) \qquad \begin{array}{r} \forall t \in T \quad tc \in \cup_{r\in R_t} TC^r : \\ (\exists t' \in T : i(t',t) = 1) \wedge (\exists r \in R_t : ref_{r,tc} = s_{r,tc_0}) \end{array} \tag{4.16}$$

$$\begin{aligned} eU_{t,tc} = \sum_{\substack{r\in R_t: \\ tc\in TC^r}} o_{t,r,ref_{r,tc}} + \sum_{tc'\in TC(ref_{r,tc},tc,r)} (rt_{r,ty_t,tc'} x_{t,r} + l_{t,r,tc'}) + \\ + \sum_{\substack{tc'\in\cup_{r\in R_t} TC^r: \\ tc\in OTC_{ty_t,r,tc'}}} l_{t,r,tc'} + (ct_{r,ty_t,tc} + rel_{bs_{r,tc}}) x_{t,r} \end{aligned} \qquad \forall t \in T \quad tc \in \cup_{r\in R_t} TC^r \tag{4.17}$$

$$eU_{t,tc} - M(1 - y_{t,t',tc'}) \leq sU_{t',tc} \qquad \begin{array}{r} \forall t,t' \in T \quad \text{index } t < \text{index } t' \\ tc, tc' \in \cup_{r\in R_t} TC^r \cap \cup_{r\in R_{t'}} TC^r : \\ tc \in \hat{TC}_{t,t',tc'} \\ i(t,t') \sum_{r\in R_t} e(tc,r) = 0 \ \wedge i(t',t) \sum_{r\in R_{t'}} e(tc,r) = 0 \end{array} \tag{4.18}$$

$$eU_{t',tc} - M y_{t,t',tc'} \leq sU_{t,tc} \qquad \begin{array}{r} \forall t,t' \in T \quad \text{index } t < \text{index } t' \\ tc, tc' \in \cup_{r\in R_t} TC^r \cap \cup_{r\in R_{t'}} TC^r : \\ tc \in \hat{TC}_{t,t',tc'} \\ i(t,t') \sum_{r\in R_t} e(tc,r) = 0 \ \wedge i(t',t) \sum_{r\in R_{t'}} e(tc,r) = 0 \end{array} \tag{4.19}$$

Following the classification used in Section 4.4.1, constraints (4.5)-(4.11) concern the traveling of the trains in the network, constraints (4.12)-(4.14) the change of rolling stock configuration and constraints (4.15)-(4.19) the capacity of the infrastructure. In more detail, these constraints impose the following conditions.

A train t cannot be operated earlier than $init_t$ (4.5). The start time of track-circuit occupation along a routing is zero if the routing itself is not used (4.6). A train starts occupying track-circuit tc along a routing after spending in the preceding track-circuit its longer stay and its travel time, if the routing is used (4.7). A train t with a scheduled stop at station s and using routing r does not enter the track-circuit following tc before the scheduled departure time from s if tc is in $TCS_{t,s}$ (4.8), and t has a longer stay in tc of at

least $dw_{t,s}$ (4.9). A train t must use exactly one routing (4.10). The value of delay D_t at least equals the difference between the actual and the scheduled arrival times at the exit of the infrastructure (4.11).

If trains t' and t use the same rolling stock and t results from t', a time at least equal to $ms_{t',t}$ must separate their respective arrival and departure times (4.12). Moreover, the track-circuit tc where the turnaround, join or split takes place (4.13) must be utilized for the whole time between t''s arrival and t's departure. Thus, tc starts being reserved by t at the latest when t' ends its utilization (4.14). Here, the inequality must be imposed since, in case of a join, two trains arrive and are connected to become a single departing one. The utilization of the departing train must then immediately follow the utilization of the first arriving train, its arriving being strictly smaller than the one of the second arriving train.

A train's utilization of a track-circuit tc starts as soon as the train occupies the track-circuit $ref_{r,tc}$ along one of the routing alternatives including it, minus the formation time (4.15). These constraints are imposed as inequalities (4.16) when they concern a track-circuit of the first block-sections of the routing ($ref_{r,tc} = s_{r,tc_0}$) and the train t results from the turnaround, join or split of one or more other trains. This is a consequence of the need of keeping platforms utilized. Indeed, if t results from t', constraint (4.14) ensures that the track-circuit where the turnaround takes place starts being reserved by t as soon as t' arrives. However, t needs to wait at least for a time $ms_{t',t}$ before departing. The occupation of the track-circuit by t is however starting from its actual departure, for guaranteeing the coherence of the occupation variables and the travel time (see constraint (4.7)). Hence, t's reservation starts much earlier than its occupation. Furthermore, the utilization of a track-circuit lasts till the train utilizes it along any routing, plus the release time (4.17). The utilization time includes: the travel time of all track-circuits between $ref_{r,tc}$ and tc, the longer stay of the train's head on each of these track-circuits and the clearing time of tc. Moreover, the utilization time includes the longer time on all track-circuits tc' such that $tc \in OTC_{ty_t,r,tc'}$. As reported in the notation list, if the head of train t is on one of the track-circuits in $OTC_{ty_t,r,tc'}$, then its tail has not yet exited tc: train t is longer than tc', or of the sequence of track-circuits between tc and tc'. Hence, if train t suffers a longer stay when its head is on one of the track-circuits in $OTC_{ty_t,r,tc'}$, such a longer stay must be counted in the utilization time of tc.

Finally, the track-circuit utilizations by two trains must not overlap (4.18, 4.19).

Acknowledgment

This work was partially supported by the "Ambassade de France en Italie" with a three-month MAE mobility scholarship for PhD candidates. We would like to thank the anonymous reviewers for their helpful and constructive remarks.

Chapter 5

A multi-criteria decision support methodology for real-time train scheduling

Abstract: *This work addresses the real-time optimization of train scheduling decisions at a complex railway network during congested traffic situations. The problem of effectively managing train operations is particularly challenging, since it is necessary to incorporate the safety regulations into the optimization model and to consider key performance indicators. This paper deals with the development of a multi-criteria decision support system to help dispatchers in taking more informed decisions when dealing with real-time disturbances. As a novel idea in the literature, optimal train scheduling solutions are computed with high level precision in the modelling of the safety regulations and with consideration of state-of-the-art performance indicators. Mixed-integer linear programming formulations are proposed and solved via a commercial solver. For each problem instance, an iterative procedure is executed to establish an efficient-inefficient classification of the best solutions provided by the formulations via a well-established non-parametric benchmarking technique: data envelopment analysis. Based on this classification, the procedure improves inefficient formulations with generation of additional linear constraints. Computational experiments are performed for practical-size instances from a Dutch railway network with mixed traffic and several disturbances. The procedure converges after a limited number of iterations, and returns a set of efficient solutions and the relative formulations.*

5.1 Introduction

A key problem in real-time railway traffic management is to efficiently reschedule trains during operations [Cacchiani et al., (2014)]. In presence of initial delays, this problem requires to detect potential conflicts between two or more trains in each resource and to globally solve them by taking into account the propagation of consecutive delays [D'Ariano (2009), Kecman et al., (2013), Corman et al., (2014b)]. Due to the limited time available to take real-time decisions, train dispatchers usually have a limited view on the effects of conflict resolution methods and are not able to compare alternative solutions in terms of various performance indicators. In practice, the dispatching measures are often sub-optimal, and leave room for improvement. Moreover, there is no clear agreement in literature on the objective functions to be used as many should be considered, due to different stakeholders and operational aspects.

This paper deals with the development of a multi-criteria decision support methodology to help dispatchers in taking more informed decisions when dealing with real-time disturbance management. As a novel idea in the literature, we compute optimal train scheduling solutions with high level precision in the modelling of the safety regulations and with consideration of state-of-the-art performance indicators to be optimized in a given computation time period. To achieve this aim, the following scientific issues have to be addressed:

- Microscopic optimization models for the real-time train scheduling problem are required, each one taking into account multiple performance indicators, either in the objective function or in the problem constraints;
- The models are to be automatically created and solved in a short computation time, and to be assessed in terms of the various performance indicators;
- A quantitative technique is needed to automatically compare the best solutions computed for the different models, to select efficient solutions and to give suggestions for improving inefficient ones;
- A comprehensive computational analysis is required to perform the performance assessment and to provide a pool of good quality solutions.

The first issue is addressed by the development of *Mixed-Integer Linear Programming* (MILP) formulations based on the *alternative graph model* of the real-time train scheduling problem [D'Ariano et al., (2007a), D'Ariano et al., (2014)]. Since there is no generally recognized indicator, the investigation of suitable objectives to optimize is very important. We study a selection of the most used objectives in the related literature, including the minimization of the maximum (initial plus consecutive) delay [Mazzarello and Ottaviani (2007)], the maximum consecutive delay [D'Ariano et al., (2007a), Pellegrini et al., (2014)], the maximum consecutive delay with consideration of train priorities [Corman et al., (2011a)], the cumula-

tive consecutive delays [Pellegrini et al., (2014)], the weighted sum of deviations from the arrival/departure scheduled times [Caimi et al., (2012)], the sum of all completion times [Dessouky et al., (2006)], the sum of delays [Meng and Zhou (2011)], a weighted sum of delays [Higgins et al., (1996)] with penalties when exceeding a threshold [Törnquist and Persson (2007)], the travel time of trains as a surrogate of energy consumption [Corman et al., (2009a)].

All the objectives studied in this paper are *operational-centric* since they focus on the minimization of railway operations objectives with a train point of view, whereas other approaches are *passenger-centric* since they focus on the maximization of the quality of service perceived by the passengers (the latter approaches are investigated e.g. in [Binder et al., (2014), Cacchiani et al., (2014), Caprara et al., (2006), Kanai et al., (2011), Sato et al., (2013), Tomii et al., (2005), Takeuchi et al., (2006)]. However, the methodology proposed in this paper remains valid even if different types of objectives are proposed.

The second issue is organized in the construction of the MILP formulations via the alternative graph model, the resolution of each formulation via a MIP solver, the investigation of the resulting solutions via a post-processing analysis. The latter procedure assesses the quality of each solution in terms of the various performance indicators.

The third issue is addressed via a useful benchmarking technique, named *Data Envelopment Analysis* (DEA), for assessing the relative efficiency of the different solutions [Charnes et al., (1978)]. It uses linear programming (LP) to determine the relative efficiencies of a set of homogeneous (comparable) units. This is a non-parametric technique, used for performance measurement and benchmarking the model solutions computed by the MIP solver (viewed as units). The analysis is based on the determination of the technical efficiency frontier as the frontier (envelope) representing the best performance and is made up of the units (formulations) in the data set which are most efficient in transforming their inputs (computational resources) into outputs (optimization results). The interested reader is referred to [Charnes et al., (1994), Cooper et al., (2007)] regarding DEA fundamentals.

In DEA analysis, the performance of a formulation on a particular problem instance is calculated by comparing it to the efficiency frontier directly determined from the data. An efficiency score is thus computed for each solution based on its distance from the efficient frontier determined by the envelope of data of all solutions. This analysis is based on the BCC DEA model [Banker et al., (1984)] which assumes variable returns to scale input/output relationships, and produces, for each unit, an efficiency score and additional information.

For each inefficient unit, improvement targets are individuated using the concept of composite unit. The attributes of a composite unit (which is a hypothetical efficient unit) are determined by the projection of an inefficient unit to the efficiency frontier. The attributes are formed as a combination of specific efficient units,

in the proportions indicated by the results of DEA analysis. Based on an analysis of the MIP solver solutions computed for each formulation, we first classify the solutions into efficient and inefficient in terms of the given set of performance indicators. Then, for all the inefficient solutions, we propose an iterative procedure in order to improve their performance. This procedure, in a first step, generates additional constraints and then calls for the MIP solver in order to solve the modified formulations. In a second step, a further DEA analysis is devoted to establish a new efficiency ranking. The iterative procedure ends after a stopping criteria. The efficient solutions obtained for the modified formulations and the values for each performance indicator are given to the dispatchers for the final selection of the train schedule to be implemented.

The fourth issue is carried out on a busy and complex Dutch railway network. The investigated network is a central part of the Dutch railway network, including the Utrecht Central station area. This station area is the most complex in the Netherlands since the station layout presents the largest amount of parallel tracks. Dense mixed local and international passenger traffic traverses the network. In the problem formulation, different train types are considered when modeling the objective function.

The computational experiments are based on the evaluation of several alternative problem formulations. Specifically, we study which formulations generate inefficient solutions or can be modified, with the proper addition of specific constraints. This would quantify the interplay and cross-performance of the different objective functions, and allows for an efficient multi-criteria decision analysis.

This paper is organized as follows. Section 5.2 provides a description of the investigated problem. Section 5.3 presents the studied formulations with a common solution space and different objective functions and Section 5.4 the methodology proposed to evaluate the performance of the best solutions provided by these formulations. Section 5.5 gives a description of the computational analysis performed on real-world Dutch railway instances. Section 5.6 describes the paper conclusions and lines for further research.

5.2 Problem description

In its basic form a rail network is composed of stations, links, block sections and track-circuits. For safety reasons, signals, interlocking and Automatic Train Protection (ATP) systems control the train traffic by imposing a minimum safety separation between trains. To this end, railway tracks are divided into block sections, which may host at most one train at a time. A detailed description of different aspects of railway signalling systems and traffic control regulations can be found e.g. in the handbook of [Hansen and Pachl (2014)].

A timetable describes the movements of all trains running in the network, specifying, for each train, the planned arrival/passing times at a set of relevant points (such as stations, junctions, and the exit point of the

network). Train movements can be formally described as follows. The passage of a train through a particular block section is named an *operation* i. A *route* is an ordered sequence of operations to be processed during the train service. The *timing* of a route specifies the start time t_i of each operation.

Each operation requires a *running time*, which depends on the speed profile followed by the train while traversing the block section. A speed profile is constrained by the rolling stock characteristics (maximum speed, acceleration and braking rates), physical infrastructure characteristics (maximum allowed speed and signalling system) and driver behaviour when approaching variable signals aspects. Safety regulations impose a minimum headway separation between the trains running in the network, which translates into a minimum *headway time* between consecutive trains. Railway timetable design usually includes *recovery times* and *buffer times* between the train routes. The recovery time is an extra time added to the travel time of a train between two stations, which corresponds to planning train speed smaller than maximum; those times can be utilized in real-time to recover from delays by running trains at maximum speed. Instead, the buffer is an extra time inserted in the timetable between consecutive train paths to prevent or reduce the propagation of train delays in the network.

Trains have planned stops at stations. The published stopping time of each train is named *dwell time*. A train is not allowed to depart from a platform stop before the scheduled departure time published in the timetable. Trains result in delay when arriving at the platform later than their scheduled arrival time as planned in the timetable.

Timetables are designed to satisfy all traffic regulations. However, unexpected events occur during operations. An *entrance perturbation* is a set of delayed trains at their entrance in the network, due to the propagation of delays from previous areas. The latter delays are called *initial delays* and can only be partially recovered by exploiting available recovery times. The initial delays are used to compute the *release time* of each train, i.e., the expected time the train enters the network under the current perturbed traffic conditions.

In disturbed operations, conflicting situations might be solved. A *potential conflict* arises when two or more trains claim the same block section at the same time. A decision on the train ordering has to be taken and one of the trains involved has to change running, departure, and/or passing times according to the constraints of the signalling system.

A delayed train might cause many potential conflicts, as the available infrastructure has very limited possibility for overtaking, normally only at major stations, thus many trains might be slowed down or held since the subsequent block section is occupied by another train and cannot be entered. This phenomenon is known as *delay propagation*; the related propagated delays spreads the impact of a disturbance to other trains in the network, affecting greatly traffic in time and space. In general, the main goal of real-time train

scheduling is to minimize train delays while satisfying traffic regulation constraints and guaranteeing the compatibility with the actual position of each train.

5.2.1 Investigated performance indicators

There is no general agreement in the literature on which the objective functions should to be adopted for formulating the real-time train scheduling problem. This problem copes with temporary infeasibility by adjusting the schedule of each train, and the newly determined schedule should optimize some performance measure, related to delay propagation. We next describe some of the objective function mostly used in literature, as previously introduced in Section 5.1:

- *Tardiness* is the difference between the estimated train arrival time and the scheduled arrival time at a relevant point in the networks. This difference can be computed with or without the so-called *unavoidable delays*, that are the initial delays that cannot be recovered. The optimization problem with unavoidable delays is named the *total delay minimization*, while the optimization problem without unavoidable delays is named the *consecutive delay minimization*. The latter problem focuses on the delays caused by the resolution of potential train conflicts only.
- *Priority tardiness* takes train classes into account, assigning suitable weights in the tardiness objective function. Tardiness is a special case of priority tardiness in which all trains have the same priority. However, setting different train weights in the objective function often means computing different optimal train schedules.
- *Punctuality* reports the number of trains that do not arrive at their final destination within a given threshold. In our experiments we study the special case in which the threshold is zero;
- *Schedule deviation* is computed to evaluate the difference between the off-line schedule and the schedule computed in real-time. This is adopted in order to try to limit the deviation from the off-line planned departure and arrival times. We observe that tardiness is a special case of schedule deviation in which only delays linked to the due dates are considered (early arrival and departure times are not penalized). However, it is again often the case that different solutions are computed for these two objective functions.
- *Total completion* is related to the arrival time of all trains at their destination.
- *Travel time* is a measure of the time spent by all trains in the system.

All indicators related to tardiness and punctuality measures can be computed either at all stations or at end (destination) stations only, in terms of consecutive or total delays. In general, we selected the above

set of performance indicators since they are representative of commonly used objectives in the literature and their optimization produces different train schedules. However, these indicators do not represent an exhaustive set of practically relevant objectives. Further research can be directed to evaluate combinations of the proposed objectives or different sets of objectives.

5.2.2 Assumptions and limitations

We model the real-time train scheduling problem by introducing the following assumptions and limitations that are the hypothesis at its basis:

- *Fixed train speed profile*: The computation of an optimized train schedule is based on a fixed-speed model, that does not allow a dynamic adjustment of running and headway times. In other words, the fixed-speed model corresponds to a fixed speed profile for each train, that does not consider the impact of braking and re-acceleration when facing the yellow and red signal aspects of the Dutch signaling system. However, the solutions computed for the fixed-speed model can be modified in a later stage as described e.g. in [D'Ariano et al., (2007b), Corman et al., (2009a)].
- *Fixed train routing*: Each train has a prescribed sequence of operations in the network, which corresponds to a fixed routing. We fix a commonly used routing for each train based on suggestions from the infrastructure managers. The combined train scheduling and routing problem has been studied e.g. in [Corman et al., (2010)].
- *Fixed release time*: At the start time of the traffic optimization, the initial position of each train in the network is a deterministic information computed and released by the traffic controllers of the neighbouring dispatching areas. The coordination of multiple dispatching areas has been investigated e.g. in [Corman et al., (2014a)].
- *Fixed dwell time*: The dwell time is considered as a deterministic information. The impact of variable dwell times has been assessed e.g. in [Larsen et al., (2014)].
- *Interlocking system*: Track circuits in complex interlocking areas are aggregated into station routings with a good approximation of sectional-release route-locking operations, as shown in [Corman et al., (2009b), Corman et al., (2011b)].
- *Rolling-stock re-utilization*: The constraints on the re-use of the same rolling stock for different train trips are ignored in this work. The modeling of this type of constraints has been proposed e.g. in [Corman et al., (2009a)].

The above assumptions and limitations do not impact the general validity of the methodology provided in this paper. However, the train schedules computed by the scheduler would need to be adjusted before final implementation. We also observe that introducing additional constraints would increase the complexity to compute a feasible train schedule, while introducing additional variables would increase the time to compute the optimal solution.

5.3 Mathematical model

The train scheduling problem can be modelled as a job shop scheduling problem with additional constraints, in which a *job* is a sequence of operations performed by a train while respecting all operational constraints. We next represent it via a MILP based on the alternative graph model of [Mascis and Pacciarelli (2002)]. Let $G = (N, F, A)$ be the alternative graph composed of the following sets:

- $N = \{0, 1, ..., n, *\}$ is the set of *nodes*, where nodes 0 and $*$ represent the start and the end operations of the schedule, while the other nodes are related to the operations in the schedule. To each node $i \in N$ is associated a start time t_i of operation i. By definition, the start time of the schedule is a known value, e.g. $t_0 = 0$, and the end time of the schedule is a variable t_*.

- F is the set of *fixed directed arcs* that model the sequence of operations to be executed by trains. Let $\sigma(i)$ be the operation succeeding i, each fixed directed arc $(i, \sigma(i)) \in F$ has a length $w^F_{i\sigma(i)}$, representing the minimum running, dwell or release time of operation i before operation $\sigma(i)$ can begin, such that and $t_{\sigma(i)} \geq t_i + w^F_{i\sigma(i)}$. In particular, the release times are represented by an arc $(0, i) \in F$.

- A is the set of *alternative pairs* of directed arcs that model the train sequencing decision when a potential conflict arises. To each alternative pair $((i, j), (h, k)) \in A$ belongs two arcs whose lengths are w^A_{ij} and w^A_{hk}. The alternative arc length w^A_{ij} represents a minimum headway time between the start time t_i of i and the start time t_j of j. In particular, w^A_{ij} (w^A_{hk}) can be sequence-dependent.

A *selection* S is a set of alternative arcs, at most one from each pair. A solution is a *complete* selection S^c, where an arc for each alternative pair of A is selected, and it is feasible if the connected graph (N, F, S^c) has no positive length cycles. Given a feasible schedule S^c, a timing t_i for operation i is the length of a longest path from 0 to i ($l^{S^c}(0, i)$).

The alternative graph can be viewed as a *disjunctive program*. We let X be the set:

$$X = \left\{ \begin{array}{ll} t \geq 0 \;\; x \in \{0,1\}^{|A|} : & \\ t_{\sigma(i)} - t_i \geq w^F_{i\sigma(i)} & \forall (i,\sigma(i)) \in F,\; \sigma(i) \neq * \\ t_j - t_i + M(1 - x_{ijhk}) \geq w^A_{ij} & \forall ((i,j),(h,k)) \in A \\ t_k - t_h + M x_{ijhk} \geq w^A_{hk} & \end{array} \right\} \tag{5.1}$$

The variables of the problem are the following: $|N|$ real variables t_i associated with the start time of each operation $i \in N$ and $|A|$ binary variables x_{ijhk} associated with each alternative pair $((i,j),(h,k)) \in A$. The variable x_{ijhk} is 1 if $(i,j) \in S$, and $x_{ijhk} = 0$ if $(h,k) \in S$. The constant M must be a sufficiently large number.

In the following subsections, a set of MILP formulations with different objective functions is presented and an illustrative numerical example is reported.

5.3.1 Formulations

The modelling of the different objective functions requires the introduction of *due date* constraints in the MILP formulation of Section 5.3. In order to measure the delay of a train we use *due date arcs* in the alternative graph model. In case of total delays, a due date arc $(k,*) \in F$ is used to measure the train delay with respect to the time at which the operation k is scheduled, i.e. the arrival of a train at a scheduled stop in a station or its scheduled exit from the network. When the goal is the minimization of consecutive delays (delays caused by the real-time train scheduling decisions necessary to solve the potential train conflicts in the network), due date arcs measures the delay with respect to the time at which the operations are scheduled but without considering the initial delays.

The length d_k of a due date arc $(k,*) \in F$ is next described. We let k be the arriving of a train at a relevant location where the delay is measured, δ_k its scheduled (off-line) arrival time and τ_k its earliest possible arrival time. In case the train has no intermediate stops with scheduled (off-line) arrival and departure times, the latter time is computed as the sum of the release time of the first operation of the train plus the minimum running and dwell times in all block sections processed by the train before operation k. The other case requires to take the maximum between the scheduled departure time and the earliest possible arrival time at each intermediate stop computed as for the previous case. The computation of the maximum between those two quantities is necessary, since the trains cannot depart before their scheduled departure time.

When we fix the due date $d_k = \delta_k$, we compute a feasible real-time train schedule and measure the total delay of the train at operation k as $max\{0, t_k - d_k\}$, in which t_k is the start time of operation k, i.e. the actual arrival time in the real-time train schedule. When we fix the due date $d_k = max\{\tau_k, \delta_k\}$, the

consecutive delay of the train at operation k can be computed as $max\{0, t_k - d_k\}$, while its initial delay is $max\{0, \tau_k - \delta_k\}$.

In what follows, we model MILP formulations with different objective functions, and we study the performance indicators introduced in Section 5.2.

The **MAXIMUM TARDINESS - MT** (9.2) minimizes the maximum (consecutive or total) delay at all relevant locations.

$$\begin{aligned} &\min t_* \\ &s.t. \\ &t_* - t_k \geq -d_k \quad \forall (k,*) \in F \\ &\{x,t\} \in X \end{aligned} \tag{5.2}$$

The **CUMULATIVE TARDINESS - CT** (5.3) is the minimization of the sum of (consecutive or total) delays at all relevant locations:

$$\begin{aligned} &\min \textstyle\sum_{k=1}^{|K|} z_k \\ &s.t. \\ &z_k - t_k \geq -d_k \quad \forall (k,*) \in F \\ &z_k \geq 0 \qquad\quad \forall k \in K \\ &\{x,t\} \in X \end{aligned} \tag{5.3}$$

where $K \subset N$ is the set of nodes from which a due date arc starts and z_k is a real positive variable associated with the due date arc $(k,*) \in F$.

A variant of cumulative tardiness is to compute the (consecutive or total) delay of each train with respect to the due date time of its last operation only. The last operation is associated either to the exit from the network or to a scheduled stop at the ending station. The resulting formulation is named **CUMULATIVE TARDINESS END - CTE** (5.4):

$$\begin{aligned} &\min \textstyle\sum_{e=1}^{|E|} z_e \\ &s.t. \\ &z_e - t_e \geq -d_e \quad \forall (e,*) \in F \\ &z_e \geq 0 \qquad\quad \forall e \in E \\ &\{x,t\} \in X \end{aligned} \tag{5.4}$$

where $E \subseteq K$ is the set of nodes from which the due date arc associated with the last operation of each train starts and z_e is a real positive variable associated with the due date arc $(e,*) \in F$.

We now consider a punctuality measure by counting the number of trains that are delayed (i.e. have a

consecutive or total delay) with respect to a threshold P on the due date time related their last operation. Considering $P = 0$, we call it **PUNCTUALITY - P0** (5.5):

$$\begin{aligned} & \min \textstyle\sum_{e=1}^{|E|} v_e \\ & s.t. \\ & Mv_e - t_e \geq -d_e - P \quad \forall (e,*) \in F \\ & v \in \{0,1\}^{|E|} \\ & \{x,t\} \in X \end{aligned} \tag{5.5}$$

where v_e is a binary variable indicating if a train is delayed ($v_e = 1$) or not ($v_e = 0$).

In some approaches different classes of trains have different priorities (see e.g. [Carey and Kwiecinski (1995),Gorman (2009)]), considering the fact that the same delay could have a different cost impact depending on the train class, e.g. international passenger, local passenger or freight trains. The next two formulations take into account priorities.

The **PRIORITY CUMULATIVE TARDINESS END - PCTE** (5.6) is the minimization of the sum of weighted delays with respect to the due date time of the last operations only. The weights depend on the train class:

$$\begin{aligned} & \min \textstyle\sum_{e=1}^{|E|} f_e z_e \\ & s.t. \\ & z_e - t_e \geq -d_e \quad \forall (e,*) \in F \\ & z_e \geq 0 \quad \forall e \in E \subset K \\ & \{x,t\} \in X \end{aligned} \tag{5.6}$$

where f_e is the cost associated with the due date arc $(e,*) \in F$.

The **PRIORITY CUMULATIVE TARDINESS END COST - PCTEC** (5.7) extends the previous formulation with the introduction of a threshold to the delay of each train and a related penalty cost:

$$\begin{aligned} & \min \textstyle\sum_{e=1}^{|E|} (f_e z_e + c_e v_e) \\ & s.t. \\ & z_e - t_e \geq -d_e \quad \forall (e,*) \in F \\ & z_e \geq 0 \quad \forall e \in E \\ & Mv_e - t_e \geq -d_e - u_e \quad \forall (e,*) \in F \\ & v \in \{0,1\}^{|E|} \\ & \{x,t\} \in X \end{aligned} \tag{5.7}$$

where f_e is the weight associated with the due date arc $(e,*) \in F$, u_e the delay threshold related to operation

e, c_e the related penalty cost and v_e is a boolean variable indicating if the associated train is delayed ($v_e = 1$) or not ($v_e = 0$) with respect to the threshold.

We consider three different classes of passenger trains: high speed, intercity and local. For all due dates belonging to the operations of each train class, we assign a weight and a threshold. In particular, high speed trains have $f^{HS} = 20/3600$ and $c^{HS} = 30$ minutes, intercity $f^{IC} = 10/3600$ and $c^{IC} = 2$ hours, local trains $f^{L} = 5/3600$ and $c^{L} = 2$ hours. Those thresholds correspond to the minimum time resulting in a monetary compensation for passengers.

All the previous formulations take into account train delays. Another interesting formulation is related to the assessment of positive and negative deviations. The two deviations correspond to study how the new schedule differs from the off-line schedule. Clearly, ahead trains have a different cost impact than delayed trains, so different costs are associated in objective function with the two kinds of deviation. The following formulation, named **SCHEDULE DEVIATION - SD** (5.8), measures the positive and negative deviations with respect to due date times and the positive deviations with respect to release times:

$$
\begin{aligned}
&\min \textstyle\sum_{k=1}^{|K|}(az_k - bp_k) + \sum_{r=1}^{|R|} aq_r \\
&s.t. \\
&z_k - t_k \geq -d_k && \forall (k,*) \in F \\
&z_k \geq 0 && \forall k \in K \\
&p_k - t_k \leq -d_k && \forall (k,*) \in F \\
&p_k \leq 0 && \forall k \in K \\
&t_r - q_r \leq w_{sr} && \forall (0,r) \in F \\
&q_r \geq 0 && \forall r \in R \\
&\{x,t\} \in X
\end{aligned}
\tag{5.8}
$$

where a and b are the costs associated with the kind of deviation considered (in our experiments $a = 1/180$ and $b = 1/360$), $K \subset N$ is the set of nodes from which due date arcs start, $R \subset N$ is the set of nodes from which release arcs start, z_k (p_k) is a real positive (negative) variables associated with each due date $(k,*) \in F$, and q_r is a real variable associated with each release arc $(0,r) \in F$.

The **TOTAL COMPLETION - TC** (5.9) models the minimization of the sum of the arrival times of all trains at their exit from the network. This objective function can be viewed as the maximization of throughput. This formulation requires to fix $d_k = 0 \; \forall k \in K$.

$$
\begin{aligned}
&\min \textstyle\sum_{e=1}^{|E|} z_e \\
&s.t. z_e - t_e \geq 0 && \forall (e,*) \in F \\
&z_e \geq 0 && \forall e \in E \\
&\{x,t\} \in X
\end{aligned}
\tag{5.9}
$$

We also consider the minimization of the travel time of all trains in the network as a surrogate for the minimization of the energy consumption. This formulation is named **TRAVEL TIME - TT** (9.3):

$$\begin{aligned} &\min \textstyle\sum_{g=1}^{|G|}(t_l^g - t_f^g) \\ &s.t \\ &\{x, t\} \in X \end{aligned} \tag{5.10}$$

where G is the set of trains in the network, t_f^g and t_l^g and the start times of the first and the last operations (f and l) in the schedule for a train g, and $(t_l^g - t_f^g)$ is the travel time spent by train g in the network.

We are interested in minimizing the above-presented objective functions, both in case of consecutive and total delays. However, the distinction between the two cases only stands for formulations (9.2) and (5.5), for which the following four distinct cases must be considered: **MT *CONSECUTIVE* (MT^C)** and **MT *TOTAL* (MT^T)** for formulation (9.2); **P0 *CONSECUTIVE* (P0^C)** and **P0 *TOTAL* (P0^T)** for formulation (5.5). Regarding the formulations (5.3), (5.4), (5.6), (5.7), (5.8), (5.9) and (9.3), a distinction between the consecutive and total delay minimization is not necessary, since the only difference in the objective function value would be a constant factor (i.e. the unavoidable delay).

5.3.2 A numerical example

This section illustrates an example of traffic situation on a network of six block sections (numbered 1–6), three of which (3, 4, 5) can be traversed in both directions, and a station (Q) with a single stop platform. In Figure 5.1, we have 4 trains (named A, B, C, D): trains A and B follow the same route (even if they enter the area from different entrance points) traversing resources 1, 2, stopping at station Q, and traversing resource 3; train C follows the route traversing resources 4, 5, 3; train D follows the route traversing resources 3, 5, 4.

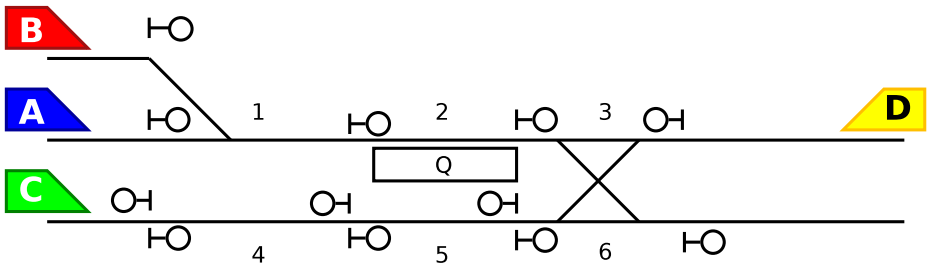


Figure 5.1: Example with four trains running in a small network

Table 5.1 presents the numerical data of the example. Row 1 specifies which train the value refers to, Row 2 the release time at which each train enters the network, Rows 3-7 the minimum running time on the

various block sections, Row 8 the minimum dwell time in station Q, Row 9 the minimum departure time from station Q, Rows 10-11 the due date times of arrival at station Q for the total and consecutive delay minimization cases, Rows 12-13 the due date times of exiting the network for the total and consecutive delay minimization cases. All trains must keep a minimum headway time of 1 unit on all resources.

Table 5.1: Numerical Example : Data

Train		A	B	C	D
Release Time	Entrance	60	45	45	60
Running Time	Block Section 1	10	10		
Running Time	Block Section 2	10	10		
Running Time	Block Section 3	15	15	15	5
Running Time	Block Section 4			25	15
Running Time	Block Section 5			25	15
Dwell Time	Station Q	20	20		
Release Time	Station Q	70	110		
Total Due Date Time	Station Q	40	65		
Cons. Due Date Time	Station Q	80	65		
Total Due Date Time	Exit	105	125	110	45
Cons. Due Date Time	Exit	115	125	110	95

For sake of simplicity, the illustrative example only considers four formulations of Section 5.3: MT^C, MT^T, $P0^T$, PCTE with $w_c = 10$ and $w_a = w_b = w_d = 1$. Table 5.2 reports on the four optimal solutions obtained by the optimization solver for each formulation. Row 1 indicates which objective function has been minimized, Row 2 the total computation time (in seconds), Rows 3-4 if the solution is optimal (1) or not (0) and the relative gap of optimality (in percentage), Rows 5-8 the value for each performance indicator (in bold the one optimized in the corresponding formulation), Row 9 the train ordering in each solution.

Table 5.2: Numerical Example : Results

Formulation	MT^C	MT^T	$P0^T$	PCTE
Comp. Time (s)	0.01	0.01	0.01	0.01
Num. Opt. Sol.	1	1	1	1
Opt. Gap %	0	0	0	0
MT^C	**51**	52	67	63
MT^T	101	**52**	117	101
$P0^T$	3	4	**3**	3
PCTE	106	541	110	**106**
Trains Orders	C-D-B-A	D-A-B-C	C-D-A-B	C-B-D-A

The optimal resolution of each formulation gives a different solution and no solution outperforms the others in terms of all performance indicators. The four solutions should therefore be given to the dispatcher. But how can a subset of solutions be selected? And, for each problem instance, how can the formulations be combined in order to obtain better quality solutions (if any exists)?

The following section proposes a new methodology for the systematic evaluation of performance of the different formulations and for the enhancement of their performance.

5.4 Performance evaluation methodology

The approach proposed in this work focuses on identifying relatively efficient formulations among a set of available formulations for the real-time train scheduling problem. The core of a new Decision Support System (DSS) is developed for this purpose. In addition, the proposed methodology enables the possibility to enhance the available formulations on the basis of an efficiency assessment conducted applying an iterative procedure based on DEA. We next describe the DSS and the developed procedure.

5.4.1 Decision support system

Each formulation is considered as a *decision-making unit* (DMU) with multiple inputs and outputs. Then, DEA is used to evaluate the relative efficiency of the different optimization formulations. The DEA-based approach differs from methods based on statistical tests in that multiple inputs and outputs are simultaneously considered in an integrated framework for the evaluation of unit efficiency. In addition, the proposed approach gives a support in order to improve the configurations showing inefficiency. In fact, besides the identification of relatively efficient and inefficient DMUs (i.e. formulations with respect to a given instance) and the efficiency ranking of the formulations under study, DEA helps to identify also the sources and the level of inefficiency for each of the considered inputs and outputs. This translates into an iterative procedure for the generation of improved formulations via the addition of specific constraints on a number of inefficient performance indicators.

Figure 5.2 describes the DSS scheme, which it is applied to a set of formulations $F_1, ..., F_q$. Each formulation F_i is characterized by a set of inputs I_i (representing the use of resources) and a set of outputs O_i (representing the achieved results) used by the DEA Module to elaborate the efficiency assessment. In the DEA methodology, the measurement of relative efficiency in the presence of multiple inputs and outputs is addressed by assigning weights so that the overall relative efficiency score is actually a ratio of the weighted sum of the outputs to the weighted sum of the inputs. The result of this analysis includes: a) a relative efficiency measure for each DMUs; b) the individuation of a set of inefficient DMUs; c) the determination of a *virtual composite* efficient DMU for each inefficient DMU; d) indications on how to drive improvements for the inefficient DMUs.

These results are the inputs for another DSS module, namely the Formulation Enhancement Module which suggests to modify the set of constraints of the evaluated formulations in order to improve their performance.

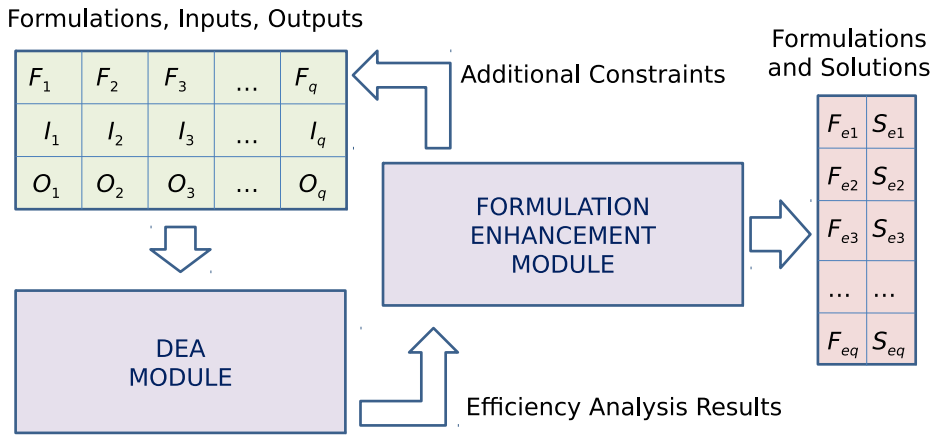


Figure 5.2: The general scheme of the DSS

The set of DMUs is updated with the new enhanced formulations and proposed to the DEA Module for a new efficiency analysis. At end of this iterative process, a final set of enhanced formulations $F_{e1}, ..., F_{eq}$ and the correspondent solutions $S_{e1}, ..., S_{eq}$ are returned by the DSS. A detailed description of the DEA literature, of the DEA model adopted for the DEA Module and of the phases of the overall process of Figure 5.2 is the argument of the following sections.

5.4.2 DEA evaluation

Setting the best formulations in use should be considered as an integral part of an advanced optimization system and plays a crucial role in the success of a modern planning and scheduling system. For the sake of simplicity and without loss of generality, we consider a formulation of high quality if it effectively and efficiently solves the problem of interest. To help to decide efficient configurations of optimization approaches, the literature (see e.g. [Cooper et al., (2007), Lu and Yu (2012), Lu and Wu (2014)]) recently proposed to use the well-known DEA method, which is based on LP and has been widely adopted in the non-parametric performance evaluation of several systems and organizations. Previous DEA-based studies measured empirically the production efficiency of a set of DMUs with multiple inputs and outputs. Also, the feature of handling the multiple inputs and outputs of DMUs makes DEA an attractive alternative for evaluating the algorithmic performance of different algorithmic configurations, because researchers and practitioners may desire to compare several inputs and outputs at the same time in order to assess the relative efficiency of each algorithm or configuration.

Given selected inputs and outputs, DEA analysis requires to handle a set of LP problems which can be solved by standard optimization solvers. Other empirical techniques for analysis of the performance of algorithms (typically based on design of experiments (DOEs), analysis of variance (ANOVA), and paired statistical tests) are also based on standard software and help to individuate best configurations arriving at conclusions that have a statistical meaning. However, these techniques are often based on parametric methods separately and sequentially compared. In general, conducting the analysis could become difficult when the numbers of distinct parameter values and evaluation criteria are large.

The proposed DEA approach differs from classical empirical analysis methods in that multiple inputs and outputs of the unit configurations under evaluation are simultaneously taken into account in an integrated framework. This paper considers each formulation as a DMU and applies DEA to assess the relative efficiency of a set of formulations tackling the same optimization problem under investigation.

Different DEA models have been proposed in the literature (see e.g. [Charnes et al., (1994),Cooper et al., (2007)]). Based on the assumption of *constant returns-to-scale* (CRS), [Charnes et al., (1978)] developed a DEA model using LP, named the CCR (*Charnes Copper Rhodes*) model. That model is based on the choice of input and output weight for each DMU. An optimal solution to the CCR model assigns the input and output weights which maximize the efficiency of the evaluated DMU, relative to the other DMUs. The model is solved in turn once for each DMU, to obtain the relative efficiencies of all the DMUs under evaluation. To relax the strict CRS assumption, [Banker et al., (1984)] proposed a generalization of the CCR model named the BCC model, which allows *variable returns-to-scale* (VRS) on production frontiers. Both CCR and BCC models can be adapted to measure the relative efficiency of different unit configurations.

[Lu and Wu (2014)] adopted the CCR model of DEA to evaluate relative efficiencies of a set of algorithms for a variant of the vehicle routing problem. [Dellino et al., (2009)] use the same approach in the configuration of a metamodel-assisted engineering design optimization. The CCR model is often criticized due to the assumption of CRS, which might not always hold to the inputs to and/or outputs from an algorithm for solving optimization problems. For instance, more computational time for executing an algorithm does not guarantee to obtain a better objective value. In the evaluation of algorithmic efficiency, [Lu and Wu (2014)] applied the BCC model which allows VRS on production frontiers. Their work shows how the relative efficiency of each configuration can be determined using DEA, helping to easily determine the best configuration or the set of efficient configurations of an algorithm. In addition, it compares the evaluation results of the CCR and BCC models for determining efficient algorithmic configurations. In general, BCC seems to be more adequate for the decision-maker who needs to analyze the performance of optimization approaches, to identify efficient unit configurations, to rank them effectively, and to give indications on how to improve inefficient cases.

5.4.3 The adopted DEA model

The CCR model [Charnes et al., (1978)] is considered as the origin of many following ideas and models in DEA literature. The mathematical formulations and summaries adopted in this paper are based on [Cooper et al., (2007)]. We suppose that there are q DMUs representing q different optimization problem formulations: DMU_1, DMU_2, ..., DMU_q. Suppose there are m inputs and s outputs for each one of them. For DMU_j the inputs and outputs are represented by $(w_{1j}, w_{2j}, \ldots, w_{mj})$ and $(y_{1j}, y_{2j}, \ldots, y_{sj})$, respectively. For each DMU_o, DEA basically tries to maximize the ratio θ_o representing the efficiency, according to the following fractional model:

$$\begin{aligned} &\max \theta_o = \tfrac{(u_1 y_{1o} + \ldots + u_s y_{so})}{(v_1 w_{1o} + \ldots + v_m w_{mo})} \\ &s.t. \\ &\tfrac{(u_1 y_{1o} + \ldots + u_s y_{so})}{(v_1 w_{1o} + \ldots + v_m w_{mo})} \leq 1 \\ &v_1, v_2, ..., v_m \geq 0; u_1, v_2, ..., u_s \geq 0 \end{aligned} \tag{5.11}$$

For each DMU involved in the performance evaluation process, DEA provides an efficiency score between 0 and 1. This efficiency score for a DMU is determined by computing the ratio of cumulative weighted outputs to cumulative weighted inputs for it. DEA enables variable weights, which are calculated in such a way that the efficiency score for the DMU is maximized. For a DEA analysis with q different DMUs, q different optimization problems are solved to compute the efficiency scores of each of the DMUs. The basic (fractional) formulation (5.11) can be easily transformed into a LP model, while the input and output data can be arranged in matrix notation W and Y, respectively. Based on these input and output matrices, the DEA model can be rewritten in the following *envelopment form* which is expressed with a variable θ and a non-negative vector of $\mathbf{h} = (h_1, ..., h_q)^T$:

$$\begin{aligned} &\min \theta \\ &s.t. \\ &\theta \mathbf{w_o} - W\mathbf{h} \geq \mathbf{0} \\ &Y\mathbf{h} \geq \mathbf{y_o} \\ &\mathbf{h} \geq \mathbf{0} \end{aligned} \tag{5.12}$$

In model (5.12), the objective is to guarantee at least the output levels expressed by $\mathbf{y_o}$ of DMU_o in all dimensions while reducing the input vector $\mathbf{w_o}$ proportionally as much as possible. An efficient DMU_o has a θ_o equal to 1, while for an inefficient DMU_o, its *reference set* RS_o is defined as follows: $RS_o = \{j | h_j^* >$

$0\}(j = 1, \ldots, q)$.

The rationale behind DEA technique is that if a given DMU is capable of producing a level of outputs using a certain amount of inputs, then other DMUs should also be able to do the same if they were to operate efficiently. The DMUs can then be combined to form a target DMU with composite inputs and composite outputs. Such an unit does not necessarily exist and it is typically indicated as virtual composite DMU (DMU_{VC}). The heart of the analysis lies in finding the *best virtual unit* for each real DMU. If DMU_{VC} is better than the original DMU under study by either making more output with the same input or making the same output with less input then the original DMU is inefficient. Even more importantly, considering the envelopment form, at the optimum, the constraints give indication about both the input excesses and the output shortfalls, in such a way an inefficient DMU can be improved correspondingly. The reference set allows to determine the DMU_{VC} obtained as a linear combination of the elements in RS_o using $\mathbf{h}^*$ as coefficients.

The presented CCR models deal with trying to reduce input variables while attaining at least the provided output levels. There is another type of model aiming to maximize output levels while spending no more than existing resources or inputs. This kind of models is referred as output-oriented. A CCR output-oriented model is formulated (in its envelopment form) as follow: In DEA, θ^* indicates the input reduction rate whereas η^* represents the output enlargement rate [Cooper et al., (2007)]. It is clear that the less the η^* value, the more efficient the DMU, or vice versa. In order to obtain an efficiency score of between 0 and 1 and to relate the efficiency scores with the input oriented model, $1/\eta^*$ is used to express the efficiency score of the DMU in an output oriented model.

The model orientation does not change the set of efficient DMUs; however the possible improvements and the composite virtual DMUs are calculated with different $\mathbf{h}$ values. The CRS assumption of CCR models typically is oversimplified for several real world problems and specifically for optimization approaches. In fact, in the practice of algorithmic design it is difficult to obtain an increase/decrease in outputs proportional to an augmentation/reduction in input variables. To this aim, the BCC model firstly proposed by [Banker et al., (1984)] allows the consideration of more useful VRS. The BCC model differs from the presented CCR model for a convexity condition in its constraints. Moreover, in order to analyze the performance of different formulations, the output-orientation appears to be more adequate. We adopt the following envelopment form, where $\mathbf{e}$ indicates the unit vector:

$$
\begin{aligned}
&\max \eta \\
&s.t. \\
&\mathbf{w_o} - W\mathbf{h} \geq \mathbf{0} \\
&\eta\mathbf{y_o} - Y\mathbf{h} \leq \mathbf{0} \\
&\mathbf{eh} = 1 \\
&\mathbf{h} \geq \mathbf{0}
\end{aligned}
\tag{5.13}
$$

In order to prepare data for the DEA analysis proposed in this paper, the DSS scheme contains a *pre-processing phase* based on [Sarkis (2007)], including scaling and normalization data and a correlation analysis.

5.4.4 Formulation enhancement procedure

For the real-time train scheduling problem under study, each considered formulation F_i is associated to a DMU in the DEA analysis. The set of inputs I_i associated to a F_i is limited to the computation time, while the set of outputs O_i includes: a) the value of the performance index related to the best solution S_i attained by F_i; b) the gap of optimality showed by F_i; c) the optimality of the solution S_i (through a binary index); d) the $q-1$ evaluations of S_i with the performance indicators (i.e. objective functions) of formulations F_j with $j \neq i$. The DEA analysis gives for each DMU (i.e. each formulation F_i) an assessment of its relative efficiency represented by η (or θ). If F_i is not fully efficient, the method individuates a reference set RS_i of efficient DMUs which can be used to obtain a virtual composite unit as a benchmark for F_i. This DMU_{VC} is determined as a linear combination of the units in RS_i, being $\mathbf{h}^*$ the vector of coefficients. In more detail, this linear combination involves the outputs of the reference being $\mathbf{y_{VC}} = \mathbf{h}^{*T}Y$.

For each inefficient F_i a formulation enhancement procedure is executed. This procedure compares each output of F_i (i.e. $\mathbf{y_i} = (y_{1i}, y_{2i}, \ldots, y_{si})$) with the correspondent output of the DMU_{VC} and individuates output improvements to drive F_i towards efficiency. The output improvements refer to the evaluation of S_i with the different performance indicators and are used to modify F_i by introducing constraints on the inefficient performance indicators. Each additional constraint requires to F_i an improvement with respect to a specific performance index (i.e. to increase the correspondent output). These constraints are determined by all the relations in $\mathbf{y_i} \leq \mathbf{y_{VC}}$ violated in the evaluations of the solution S_i (i.e. the train order obtained by F_i) with respect to the performance indicators of F_j with $j \neq i$. The procedure is executed iteratively by updating at each step the set of formulations via the improvements suggested by the DEA analysis. The stopping criteria is reached either when all DMUs are fully efficient or when inefficient DMUs are not improvable.

5.4.5 Numerical example

We use the example of Section 5.3.2 to better explain how the DSS work. We evaluate the four formulations with the DEA Module, considering inputs and outputs according to Section 5.4.4. As a result, the solutions found by $P0^T$ and PCTE are shown to be inefficient and the DMU_{VC} is represented by MT^C in both cases (i.e. the reference set contains MT^C only).

Comparing the values of each performance indicator, the solution resulting from MT^T is efficient, since MT^T is by far the most difficult indicator to take into account indirectly. The Formulation Enhancement Module suggests the following additional constraints: PCTE is required to add a constraint regarding MT^C (≤ 51); and $P0^T$ three constraints regarding MT^C (≤ 51), MT^T (≤ 101) and PCTE (≤ 106). The solutions returned by the enhanced PCTE and $P0^T$ formulations are the same found by MT^C. All four formulations are now efficient and the final solutions presented by the DSS to the dispatcher are the trains orders of MT^C and MT^T solutions.

5.5 Computational experiments

This section presents the computational results on the formulations and DEA techniques introduced in this paper. The tests have been performed in a laboratory environment on 30-minute instances of a Dutch railway network. The optimal MILP solutions are computed via the solver IBM ILOG CPLEX MIP 12.0. The experiments are executed on a workstation Power Mac with processor Intel Xeon E5 quad-core (3.7 GHz), 12 GB of RAM.

5.5.1 Test case description

The infrastructure considered is a large part of the railway network in the east of the Netherlands. As shown in Figure 5.3, the network is composed of four major stations with complex interlocking systems and dense traffic: Utrecht Central, Nijmegen, Arnhem and Den Bosch. Other 36 minor stations in the network are also considered in our model.

The two main traffic directions are served by the line between Utrecht and Arnhem (towards Germany) and the line between Utrecht and Den Bosch (from Amsterdam towards Eindhoven and the southern part of the country). The network comprises a combination of single/double-tracks of different length, with a maximum distance between area borders of around 300 km. In total, there are more than 1000 block sections and 200 stopping locations (i.e. platforms actually used at stations).

For the computational experiments, we use the 2008 timetable that is cyclic with a cycle length of one

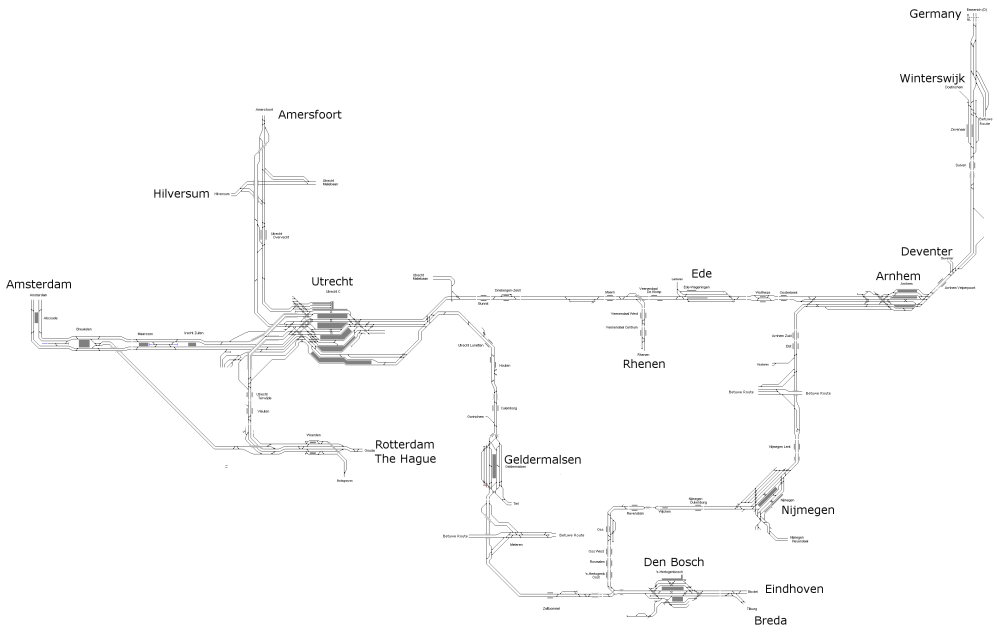


Figure 5.3: The considered Dutch railway network

hour, though most intercity and local trains services operate up to 4 times per hour per direction. Figure 5.4 gives a graphical view of line frequencies, each thick line (solid or dotted) represents a train line with 4 trains running per hour per direction, each thin line represents a train line with 2 trains running per hour per direction.

The real-life stochasticity of train operations can be modeled by Weibull distributions. In this work, a random variation of the release time is applied to all trains in the network, and initial delays are generated for multiple trains. From the Weibull distribution in [Corman et al., (2011b)], two sets of entrance perturbation instances are generated: 5 “normal” perturbations that corresponds to the scale, shift and shape parameters in that paper; 5 “increased” perturbations that are obtained by using the same scale and shift parameters and doubling the shape parameter. The latter perturbations result in an increased variability of train delays.

For each entrance perturbation, we deal with two types of instances: 30-minute (60-minute) traffic optimization with 99 trains (154 trains). In the latter instances the following trains traverse the network: 80 local trains, 70 intercity trains and 4 high speed trains. For the latter instances, we use CPLEX with one-hour computation time in order to compute near-optimal train schedules.

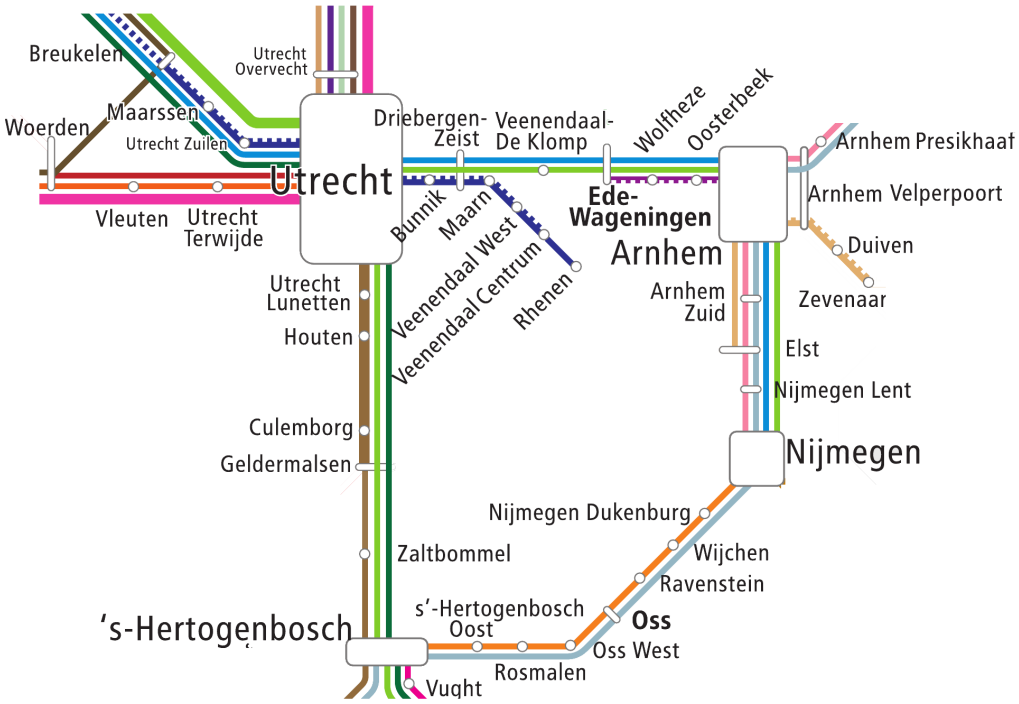


Figure 5.4: Line frequencies on the network (source: treinreiziger.nl)

5.5.2 Performance evaluation

Tables 5.3 and 5.4 report on the first and last iterations of the DEA framework for the 30-minute and 60-minute instances, respectively. In each table, we report 110 runs of CPLEX per iteration: the 10 entrance perturbation instances of Section 5.5.1 multiplied by the 11 formulations of Section 5.3.1. Specifically, each row of the tables presents average results on the 10 entrance perturbation instances.

The results of the first/last iterations are provided in Tables 5.3 and 5.4 with a 14-row information: Row 1 gives the average computation time (in seconds), Row 2 the number of problems that were solved to optimality by CPLEX, Row 3 the average optimality gap (in percentage), Rows 4-14 the average value of each performance indicator. For each column (i.e. for each formulation), the average value of the optimized performance indicator is highlighted in bold, while the improvements of the last iteration versus the first iteration are highlighted in italic.

For the 30-minute instances of Table 5.3, CPLEX takes on average up to 15 seconds to compute the optimal solution for each formulation and the DSS requires at most 3 iterations in order to converge, i.e. no more constraints can be added to improve efficiency. Overall, the DSS requires at most 3 iterations in order to converge, and is therefore very quick to compute efficient solutions. Regarding the 60-minute instances of

Table 5.3: Computational results for 30-minute instances

Formulation	MT^C	CT	CTE	$P0^C$	PCTE	PCTEC	SD	TC	TT	MT^T	$P0^T$
First Iteration											
Comp. Time (s)	3.3	11.9	8.9	5.3	9.1	7.6	9.1	7.3	14.0	1.8	3.5
Num. Opt. Sol.	10	10	10	10	10	10	10	10	10	10	10
Opt. Gap %	0	0	0	0	0	0	0	0	0	0	0
MT^C	**220**	290	245	530	278	284	287	264	264	414	604
CT	4485	**2486**	3032	5519	3668	3696	2568	3244	3237	7537	9300
CTE	1559	1116	**954**	1735	1058	1058	1207	1032	1032	2481	2917
$P0^C$	16.9	14.5	14.5	**11.7**	14.6	14.6	14.7	15.0	15.0	18.3	16.5
PCTE	3.4	2.8	2.3	3.7	**2.2**	2.2	3.1	2.5	2.5	5.0	6.2
PCTEC	3.4	2.8	2.3	3.7	2.2	**2.2**	3.1	2.5	2.5	5.0	6.2
SD	192	171	179	203	185	185	**170**	181	181	222	242
TC	513825	513449	513320	514166	513440	513440	513606	**513247**	513247	514716	515084
TT	91422	91046	90917	91763	91037	91037	91203	90844	**90844**	92313	92681
MT^T	881	917	881	1011	893	893	917	881	881	**872**	1033
$P0^T$	70	70	70	69	70	70	70	70	70	71	**66**
Last Iteration											
Comp. Time (s)	3.4	11.9	8.9	5.3	9.1	8.3	9.1	7.3	14.3	2.9	3.5
Num. Opt. Sol.	10	10	10	10	10	10	10	10	10	10	10
Opt. Gap %	0	0	0	0	0	0	0	0	0	0	0
MT^C	**220**	290	245	530	278	*275*	287	264	*263*	*313*	604
CT	*4381*	**2486**	3032	5519	3668	*3629*	2568	3244	*3229*	*6125*	9300
CTE	*1512*	1116	**954**	1735	1058	1058	1207	1032	1032	*2051*	2917
$P0^C$	*16.5*	14.5	14.5	**11.7**	14.6	14.6	14.7	15.0	15.0	*16.5*	16.5
PCTE	3.4	2.8	2.3	3.7	**2.2**	2.2	3.2	2.5	2.5	*4.2*	6.2
PCTEC	3.4	2.8	2.3	3.7	2.2	**2.2**	3.2	2.5	2.5	*4.2*	6.2
SD	*191*	171	179	203	185	*184*	**170**	181	181	*208*	242
TC	*513780*	513449	513320	514166	513440	*513438*	513606	**513247**	513247	*514350*	515084
TT	*91377*	91046	90917	91763	91037	*91035*	91203	90844	**90844**	*91947*	92681
MT^T	881	917	881	1011	893	893	917	881	881	**872**	1033
$P0^T$	70	70	70	69	70	70	70	70	70	*70*	**66**

Table 5.4, CPLEX is not always able to compute an optimal solution within 1 hour of computation, except for MT^C, MT^T and $P0^T$. For the latter formulations, the lower bound value is, on average, closer to the value of the optimal solution compared to the other formulations.

When looking at the number of efficient solutions for each instance, the DEA Module delivers, on average, 8 different efficient solutions. The Formulation Enhancement Module is able to improve several performance indicators, but some other indicators decrease their performance, thus yielding to trade-off solutions. In some particular cases, the addition of constraints deteriorates the performance of some formulation in terms of its objective function, because the set of feasible solutions decreases for the enhanced formulation and the optimal solution of the original formulation is discarded by the solver.

Table 5.5 reports average information on the DEA evaluation performed by the DSS. The evaluation is given both for 30-minute and 60-minute instances. Each row presents the average results obtained on the 10 entrance perturbation instances. Row 1 gives the cumulative number of instances for which each formulation results to be efficient at the first, second and third iterations; Row 2 counts how many times a DMU is in some reference set RS_i, i.e. how many times it has been used in a DMU_{VC} (at first, second and third

Table 5.4: Computational results for 60-minute instances

Form.	MT^C	CT	CTE	$P0^C$	PCTE	PCTEC	SD	TC	TT	MT^T	$P0^T$
First Iteration											
Comp. Time (s)	914.4	1022.6	1182.3	411.8	924.1	900.6	1376.9	874.8	1160.9	85.0	365.5
Num. Opt. Sol.	10	8	7	9	8	7	7	8	7	10	10
Opt. Gap %	0	10.3	18.9	0.4	11.7	11.1	11.4	0	0.3	0	0
MT^C	**313**	389	418	840	499	497	441	424	427	595	733
CT	14789	**7760**	10079	16556	12495	12085	8368	9971	10143	21393	24697
CTE	5178	3081	**2619**	5484	2981	2906	3406	2673	2699	6121	7115
$P0^C$	37.0	28.6	28.1	**22.6**	29.3	28.9	29.4	28.4	28.9	37.8	33.6
PCTE	12.2	8.6	6.8	14.5	**6.1**	6.0	9.5	6.9	6.8	13.2	15.6
PCTEC	12.2	8.6	6.8	14.5	6.1	**6.0**	9.5	6.9	6.8	13.2	15.6
SD	437	370	399	458	423	419	**371**	397	399	505	539
TC	965034	962888	962433	965175	962840	962813	963308	**962327**	962384	965918	966716
TT	186553	184407	183952	186695	184360	184332	184827	183846	**183903**	187437	188235
MT^T	1017	1034	1028	1349	1057	1057	1088	1039	1039	**1004**	1221
$P0^T$	112	109	109	107	109	110	110	109	109	112	**104**
Last Iteration											
Comp. Time (s)	920.5	1238.8	1243.9	416.4	1644.1	930.3	1850.2	1025.8	918.7	121.6	365.5
Num. Opt. Sol.	10	8	5	8	8	7	6	7	7	10	10
Opt. Gap %	0	12.2	20.4	2.7	11.2	11.8	7.9	0	0.2	0	0
MT^C	**313**	*383*	*403*	*734*	*498*	*491*	*435*	425	*415*	*553*	733
CT	*14638*	**7894**	10154	*15465*	*12047*	12120	*7635*	10012	*9023*	*18680*	24697
CTE	*5075*	*3067*	**2650**	*4974*	*2945*	2913	*2997*	*2668*	*2433*	*5495*	7115
$P0^C$	*36.3*	29.0	*28.0*	**23.1**	29.3	28.9	*28.7*	28.5	*28.0*	*34.8*	33.6
PCTE	*12.1*	*8.4*	6.8	*13.4*	**6.1**	6.0	*8.4*	*6.8*	*6.5*	*12.3*	15.6
PCTEC	*12.1*	*8.4*	6.8	*13.4*	6.1	**6.0**	*8.4*	*6.8*	*6.5*	*12.3*	15.6
SD	437	371	400	*449*	*418*	419	***360***	398	*382*	*479*	539
TC	*964915*	962910	962455	*964680*	*962811*	*962804*	*962624*	**962336**	*961883*	*965274*	966716
TT	*186434*	184429	183974	*186199*	*184330*	*184323*	*184453*	183855	***183712***	*186794*	188235
MT^T	1017	*1022*	*1022*	*1249*	*1051*	*1051*	*1060*	*1034*	*1011*	**1004**	1221
$P0^T$	112	110	109	107	109	110	110	109	109	112	**104**

iterations); Row 3 returns the average number of constraints added to each formulation at each iteration.

Table 5.5: Cumulative DEA evaluation

Formulation	MT^C	CT	CTE	$P0^C$	PCTE	PCTEC	SD	TC	TT	MT^T	$P0^T$
30-Minute Instances											
Efficient cases	9|10|10	6|6|6	8|8|8	10|10|10	9|9|9	4|4|5	9|9|9	8|8|8	2|3|3	6|9|9	10|10|10
$\in RS_i$	0|0|0	1|1|1	4|4|6	1|0|0	11|9|5	3|1|1	6|3|3	6|5|5	0|0|0	0|0|2	1|0|0
Added Constr.	8|0|0	0|0|0	0|0|0	0|0|0	0|0|0	3|2|0	0|0|0	0|0|0	2|0|0	9|0|0	0|0|0
60-Minute Instances											
Efficient cases	8|9|9	8|8|8	6|6|6	8|8|8	7|8|8	4|6|6	7|7|7	6|7|7	4|4|4	6|9|9	10|10|10
$\in RS_i$	2|0|0	4|3|4	4|2|2	1|1|1	5|4|4	9|5|5	6|6|6	8|8|8	1|1|1	8|8|8	2|0|0
Added Constr.	7|0|0	6|0|0	3.3|3.5|3.5	9|9|0	2.7|7|7	3.3|3|3	4.3|8|6.5	2.3|4|4	2.3|3.3|3	9.5|9|9	0|0|0

From Table 5.5, we have the following observations. The efficiency found at the first iteration is often improved at the next iterations (4 formulations are improved for the 30-minute instances and 5 formulations are improved for the 60-minute instances). This is a confirmation that the additional constraints are very useful to improve the formulations and their solution quality in terms of multiple performance indicators.

Regarding the specific formulations, $P0^T$ is already efficient for all the instances, since the other formulations have a poor performance for this performance indicator. TT has the smallest number of efficient solutions, since competitive solutions are found by the other formulations in a shorter computation time.

Also, MT^T has a high number of inefficient solutions, however these are often improved by the iterative procedure. This is due to the nature of the formulation: only a small number of trains have effects in this objective function and different optimal solutions exist. Furthermore, MT^T is quite often improved and presents the largest number of average constraints added during the iterative process. The improvement of MT^T is also visible when looking at the specific performance indicators in Tables 5.3 and 5.4. The other formulations are improved for similar reasons.

5.6 Conclusions and future research

This paper presents a methodology for the development of a multi-criteria decision support system to help dispatchers in taking more informed decisions when dealing with real-time traffic disturbances. We are particularly interested in the computation of near-optimal solutions in terms of a set of performance indicators. This is achieved by the development of MILP formulations based on the alternative graph model, each one taking into account multiple performance indicators, either in the objective function or in the problem constraints. An iterative DEA-based procedure is proposed to establish an efficient-inefficient classification of the formulations and to improve inefficient formulations. For each tested instance, the procedure converges after a limited number of iterations and returns a set of efficient formulations and their best solutions. The final selection of the train schedule to be implemented is left to the dispatchers.

Experiments are performed for 30-minute and 60-minute instances from a Dutch railway network with mixed traffic and multiple delayed trains. We study which formulations are efficient or can be modified to be efficient, with the proper addition of selected linear constraints. The proposed methodology is shown to be able to improve the inefficient formulations, and to deliver of a pool of improved formulations and their solutions in a short computation time for 30-minute instances.

The insights of the proposed approach regard the investigation of different objective functions and constraints and the identification of the most representative formulations for the computation of good trade-off solutions. On-going research is dedicated to a more comprehensive evaluation of alternative formulations and DEA classifications. The long term contribution to the practice of dispatching is the possibility to automatically generate a combination of objective functions and constraints such that the most efficient solutions can be quickly identified, classified, visualized and delivered to the dispatchers.

Chapter 6

Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports

Abstract: *This paper addresses the real-time problem of scheduling aircraft in a terminal control area. We formulate this problem via the alternative graph formulation. A rolling horizon framework is introduced to manage busy traffic situations with a large number of delayed aircraft. As scheduling algorithms, we compare a Branch and Bound (BB) algorithm with a First Come First Served (FCFS) rule. The algorithms are evaluated on practical size instances from Roma Fiumicino and Milano Malpensa. Experimental results demonstrate that BB better minimizes aircraft delays and travel times compared to FCFS. BB also requires less frequent changes of aircraft scheduling decisions.*

6.1 Introduction

Due to the increasing traffic demand and the limited availability of airport resources, aviation authorities are seeking methods to better use the existing infrastructure during operations and to better manage aircraft movements in the proximity of airports, trying to improve punctuality while maintaining the required safety level.

From a logical point of view, it is possible to divide Air Traffic Control (ATC) decisions in a Terminal Control Area (TCA) into: (i) Routing decisions, where an origin-destination route for each aircraft has to be chosen regarding air segments and runways; (ii) Scheduling decisions, where feasible aircraft sequencing and timing have to be determined in each air segment, runway and (eventually) holding circle, satisfying scheduling regulation and giving optimized solutions. In this work, a traffic control system is developed in order to support traffic controllers in taking optimal scheduling decisions for given aircraft routes. Aircraft re-routing decisions are considered in other studies, we refer the interested reader e.g. to [D'Ariano et al., (2010), D'Ariano et al., (2012b), D'Ariano et al., (2015)].

We study the aircraft scheduling problem (ASP) that is one of the challenging problems in air traffic control during disturbed traffic situations. Given a set of landing/take-off aircraft, and given for each aircraft its approach/leaving path in the TCA, its current position, its scheduled runway occupancy time, and a time window to accomplish the landing (departing) procedures, our problem definition is to assign the passing time to each aircraft at all resources of its path (holding circles, air segments, common glide paths, runways) in such a way that all aircraft potential conflicts are solved, all the constraints on safety separation distances are satisfied, the available airport capacity is efficiently used and aircraft delays due to conflicting paths are significantly reduced.

Recent literature reviews of [Ball et al., (2007), Barnhart et al (2012), Bennell et al., (2011), D'Ariano et al., (2012b), Eun et al., (2010), Kuchar and Yang (2000)] presented at least two types of ASP classification.

First, ASP models can be grouped as *basic* or *detailed*. Basic models include only the runways in the TCA, while detailed models also schedule aircraft on other relevant TCA resources. Most of the early papers on ASP fall in the former category. This choice is motivated by the fact that the runways are often the bottleneck of the TCA. With a basic model the ASP is typically formulated as a single/parallel machine scheduling problem, while detailed models of the ASP are formulated as a job shop scheduling problem. In later level of modelling, a significant number of real-world constraints and characteristics of the ASP can be included. In this paper a detailed ASP model is presented based on the alternative graph formulation of [Mascis and Pacciarelli (2002)] that is able to enrich the model of [Bianco et al., (2006)] by including additional real-world constraints, such as holding circles, time windows for aircraft travel times, multiple

capacities of air segments and blocking constraints at runways. This optimization model has been already successfully adopted to manage rail and air transportation problems [Corman et al., (2012a), D'Ariano et al., (2007a), D'Ariano et al., (2010), D'Ariano et al., (2012b)].

Second, ASP approaches can be classified as *static* or *dynamic*. In the static ASP, landing/departing aircraft must be sequenced when all information is known, whereas in the dynamic ASP aircraft enter the TCA during a current time period of traffic prediction, and a new sequence of take-off/landing aircraft has to be recomputed for every time period in which new incoming aircraft positions are known. In this paper static ASP algorithms are utilized, provided that they are able to compute a feasible schedule, in a dynamic environment in which the information on the landing/departing aircraft is updated during the traffic prediction [Abdelghany et al., (2008), Clarke and Solak (2009), Steria et al., (2011)].

The main objective of this work is the investigation of potential benefits of an advanced rolling horizon approach for the management of aircraft traffic flows at busy airports. Rolling horizon approaches have already been applied to transportation problems in which data variability is inherently associated with incomplete information on the real-time operations [Hu and Chen (2005), Lai et al., (2008), Meng and Zhou (2011), Zhan and Zhang (2010)]. The proposed rolling horizon approach enables the dynamic management of aircraft ingoing and outgoing the TCA. Compared to the centralized approach in which all aircraft are managed in one step for the entire time horizon of traffic prediction, the rolling horizon approach divides the problem in multiple steps, enabling the dynamic management of aircraft for large time horizons.

The originality of this approach is the design and implementation of a dynamic setting for air traffic control based on a detailed problem modelling and on the adoption of heuristic and exact ASP algorithms. With a special focus on mathematical modelling and design of advanced algorithmic strategies, this research contribution evaluates the quality of ASP solutions in case of various sources of disturbance. A rolling horizon framework is developed based on alternative graphs, taking care of uncertainty of the predicted operational conditions over time, and solved by heuristic and exact ASP algorithms. Specifically, we address the relevant questions "how does the traffic control system react when disturbances arise", "when and how is it more convenient to reschedule aircraft in the TCA", "which algorithm performs best in terms of delay and travel time minimization", "which algorithm is the less sensitive to disturbances".

The architecture of our optimization-based traffic control system with rolling horizon mechanism works as follows. The aircraft scheduling procedure considers a roll period and a look-ahead period (i.e., a time horizon of traffic prediction). Given currently available and predicted information on the operational conditions, the optimization algorithm adopted for the current ASP problem delivers the scheduling plan periodically for every roll period. At the beginning of each roll period, the operational conditions are predicted for a time period in the future. The look-ahead period is defined as the time period from the end of the roll period

to the end of the current traffic prediction. This dynamic mechanism complicates the scheduling aspect compared to single optimization run, due to overlapping time periods and to additional constrains from current operations.

At each stage of the rolling horizon approach, we run a scheduler plus eventually a pre-processing phase, in which holding decisions are taken for incoming aircraft in order to avoid potential conflicts between the aircraft from the previous time period still occupying resources in the TCA and next incoming aircraft. We then apply the following scheduling algorithms for solving the ASP of the current time period of traffic prediction. As rule-based method, first come first served (FCFS) is evaluated, taking ASP decisions one at a time by assigning each conflicting resource to the first aircraft requiring it. This local rule requires no look-ahead control action but ignores useful information on the actual traffic conditions since aircraft run in the airport area on the basis of their actual order of arrival at each air segment or runway. According to [Bennell et al., (2011)] FCFS is a common dispatching rule, even if human controllers may adjust the FCFS sequence in order to recover possible infeasibilities. As optimization method, the truncated branch and bound (BB) algorithm of [D'Ariano et al., (2010), D'Ariano et al., (2012b)] is used to solve the ASP to near-optimality.

The ASP instances are generated by varying the following factors: (i) the two main Italian airports (Roma Fiumicino and Milano Malpensa, the airport models are shown in [D'Ariano et al., (2015)]) are investigated, having different infrastructures and traffic flows; (ii) delay configurations with multiple ingoing and outgoing delayed aircraft; (iii) various level of uncertainty regarding the aircraft entry times in the TCA.

For all the proposed ASP instances, the ASP algorithms are evaluated on the following key performance indicators: (1) System configurations: roll and look-ahead periods are critical components to be assessed in the rolling horizon approach [Hozak and Hill (2009)]. From the one hand, high rescheduling frequency may be connected with updated information on the traffic flow status. From on the other hand, low rescheduling frequency may cause less nervous changes of the scheduling plans over time due to minor disturbances. The length of the look-ahead period has clearly impact on the BB performance. (2) The computation time of the algorithms needs to be evaluated for each system setting. A dynamic approach may not be usable on-line only if the CPU requirement is too high. This may be a limitation on the number of viable system settings. (3) Delay propagation and disturbance robustness of the two ASP algorithms: entrance and exit aircraft delays are to be quantified. The question is how much could be the gain of BB compared to FCFS for the different airports (i) and for the various sources (ii) and (iii) of traffic disturbance. To answer the question, we will compare the results obtained for static information, i.e., there is perfect knowledge on the aircraft entry time in the TCA, and for dynamic information, i.e., the knowledge on the aircraft entry time is updated at each run of the algorithm.

The paper is organized as follows. Section 6.2 describes the ASP problem and introduces definitions of constraints, objectives, disturbances and airport structures. Section 6.3 presents the alternative graph formulation of the ASP. Section 6.4 describes the new rolling horizon framework and gives an illustrative example based on alternative graphs. Section 6.5 reports on the tested instances for the two Italian airports. Computational results are reported for different system configurations and algorithms. Key performance indicators (1-3) are assessed. Section 6.6 gives conclusions and further research directions for practical implementation of the proposed system.

6.2 Aircraft scheduling problem

In the TCA, landing aircraft move along predefined routes from an entry point to a runway, following a standard descent profile. A minimum separation between every pair of consecutive aircraft, depending on the type and relative positions of the two aircraft (at the same or different altitude), must be guaranteed during all the approaching phases. The safety distance can be translated in a separation time (*setup time*) by considering the different aircraft speeds. Following the same concept, departing aircraft leave the runway moving towards the assigned exit fix along an ascent profile, also respecting the separation standards. The runways can be occupied by only one aircraft at a time each, and a separation time should be ensured between any pair of aircraft.

Once a take-off/landing aircraft enters the TCA it should proceed to the runway. For each aircraft, a *processing time* on the runway and on the air segments before/after the runway is computed according to the ascent/descent profile. The processing time of each air segment varies between minimum and maximum possible values, since the speed of each aircraft flying in the TCA must vary in a pre-defined window. Additionally, *holding time* can be used to make aircraft wait at ground level or in flight, in specific resources named *holding circles*, until they can be guided into the take-off/landing sequence. Holding in flight can be achieved by letting aircraft flying in circle in specific areas named holding circles. Once an aircraft enters the holding pattern, it must fly at fixed speed for a number of half circles as prescribed by the air traffic controller. Departing aircraft can wait on the ground just before entering the runway. The schematic structure of the TCA for the two case studies of this paper, namely Roma Fiumicino (FCO) and Milano Malpensa (MXP), is next reported.

6.2.1 Terminal control area

Figure 6.1 shows a TCA scheme of the FCO airport in which there are three runways (RWY 16L, RWY 16R, RWY25), but two (RWY 16R and RWY 25) cannot be used simultaneously and are thus considered as one.

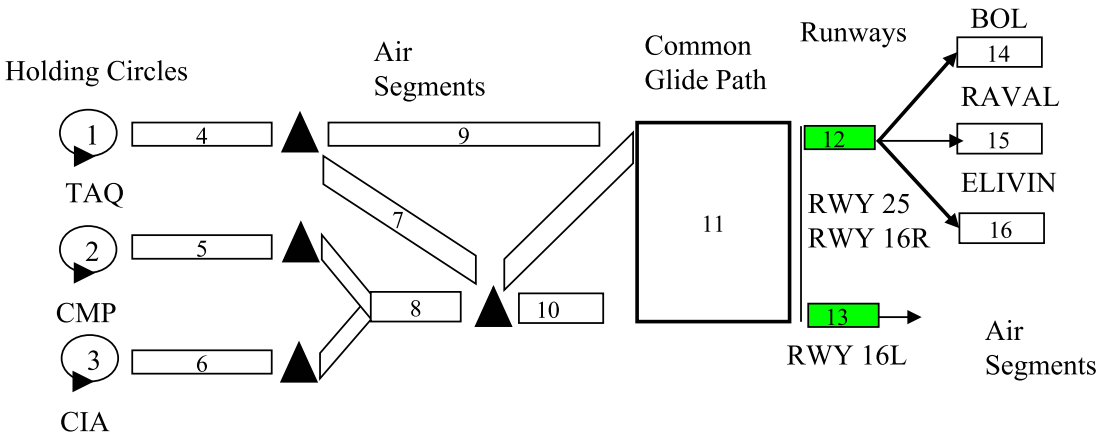


Figure 6.1: Schematic view of Roma Fiumicino TCA

These runways can be used for departing or landing aircraft. The airport resources are 3 airborne holding circles (CIA, CMP, and TAQ, numbered 1-3), 7 air segments for landing procedures (4-10), three for take-off procedures (14-16), two runways (12-13) and a common glide path (11). The latter resource (resource 11) includes two parallel air segments before the two runways for which, besides a minimum longitudinal distance between aircraft, traffic regulations also impose a minimum diagonal distance.

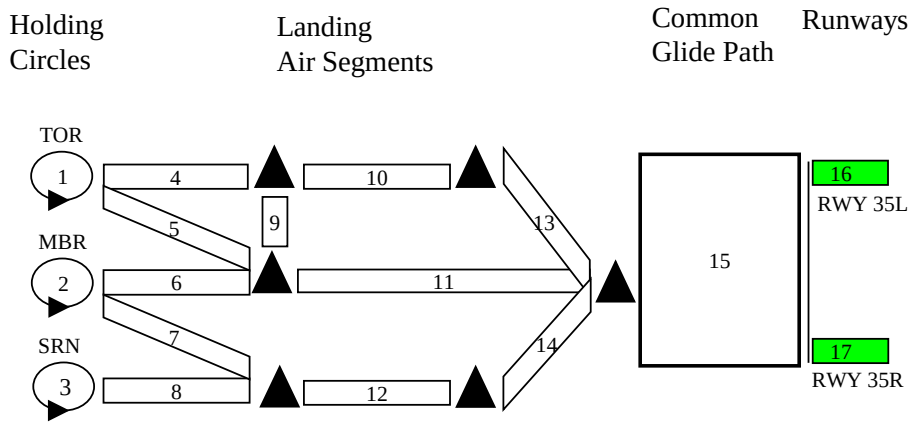


Figure 6.2: Schematic view of Milano Malpensa TCA

Figure 6.2 shows a TCA scheme of the MXP airport in which there are two runways (RWY 35L, RWY 35R), which can be used both for departing and landing procedures. The airport resources are 3 airborne holding circles (TOR, MBR and SRN, numbered 1-3), 11 air segments for landing procedures (4-14), two

runways (16-17) and a common glide path (15).

6.2.2 Traffic disturbances

Real-time traffic control copes with conflicting aircraft routes at air segments and runways by adjusting the aircraft schedule in terms of new processing, setup and holding times. Specifically, a *potential conflict* occurs whenever aircraft traversing the same resource (air segment or runway) do not respect the minimum separation time required for safety reasons. Separation times depend not only on the aircraft times but also on the route chosen for consecutive aircraft. For this reason, the setup times are sequence-dependent. In addition to the traffic regulations constraints, the real-time position of each aircraft must be satisfied. The latter information enables the computation of the *release time*, which is the expected time at which each aircraft will enter the TCA.

The main goal of the aircraft rescheduling process is to reduce aircraft delays that are late arrival times at the entrance/exit of the TCA. Aircraft delays are computed in the following way. A departing aircraft is supposed to take-off within its assigned time window and is late whenever it is not able to accomplish the departing procedure within its assigned time window. Following the procedure commonly adopted by air traffic controllers, we consider a time window for take-off between 5 minutes before and 10 minutes after the Scheduled Take-off Time (STT). A departing aircraft is late if leaving the runway after 10 minutes from its STT. Arriving aircraft are late if landing after their Scheduled Landing Time (SLT). We name *total exit delay* the delay of each aircraft at the runway. This value is partly a consequence of the late entrance in the TCA (which causes an *unavoidable initial delay* at the runway) and partly due to the additional delay caused by the resolution of potential aircraft conflicts, which is named *consecutive delay* [D'Ariano et al., (2007b), D'Ariano et al., (2008), D'Ariano et al., (2010), D'Ariano et al., (2012b)].

Two types of entrance delay will be analyzed. In a first set of experiments the arrival time of each aircraft at the TCA is known at time 0. Besides the entrance time value computed off-line, a forecast of the entrance delay of each aircraft is computed at time 0 (average and maximum values are reported in Table 6.1) and kept fixed during the overall traffic prediction. In a second set of experiments the entrance delay is estimated at time 0 and updated along the time (the range of the additional variations is reported in Table 6.6) and finally fixed only when the aircraft actually enters the TCA. In both sets of experiments, once the aircraft enters the TCA we assume that its behavior is deterministic. However, besides the entrance delays that are generated as input data, the scheduler may require additional (consecutive) delays both at the entrance of the TCA and at the runway, in order to solve aircraft conflicts. We thus measure a *total entrance delay* and a *total exit delay* for each aircraft.

6.3 Alternative graph formulation

The ASP problem can be represented by an alternative graph that is a triple$G = (N, F, A)$, where $N = 0, 1, \ldots, n, *$ is the set of nodes, F is a set of directed arcs, called *fixed arcs*, and A is a set of pairs of directed arcs called *alternative arcs*. Each node $1 \ldots n$ is associated to the start time t_i of operation i, i.e., to the entrance of the associated aircraft in the associated resource. Additional nodes 0 and $*$ are used to model the start of the schedule, i.e., the schedule start time t_0 and its completion. In fact, t_* models the objective function of the ASP, which is the maximum consecutive delay, as described later in this section.

Arcs in the set F, fixed, model the aircraft routes. Given two consecutive aircraft operations i and $\sigma(i)$, the fixed arc $(i, \sigma(i))$ of the set F is associated to the processing of operation i and has weight $f_{i\sigma(i)}$ (the time spent on the associated resource).

Alternative arcs in the set A model aircraft sequencing and holding decisions. If $(< i, j >, < h, k >) \in A$, arc $< i, j >$ is the alternative of arc $< h, k >$ and vice versa. A selection S is a set of alternative arcs. S is a *partial selection* if at most one arc is selected from each pair in A, and it is *complete* if exactly one arc is chosen from each pair in A. A complete selection S is a *feasible schedule* (i.e., a solution to the ASP problem) if the connected graph $(N, F \cup S)$ has no positive length cycles.

Given a feasible schedule S, the length $l^S(0, i)$ of a longest path from 0 to i in the graph $(N, F \cup S)$ defines the minimum feasible start time t_i of operation i associated to S. We call this quantity a timing $t_i = l^S(0, i)$. A feasible schedule S is optimal if $t_* = l^S(0, *)$ is minimum over all feasible schedules. We next describe the problem constraints, the objective function formulation and a mathematical program for the ASP.

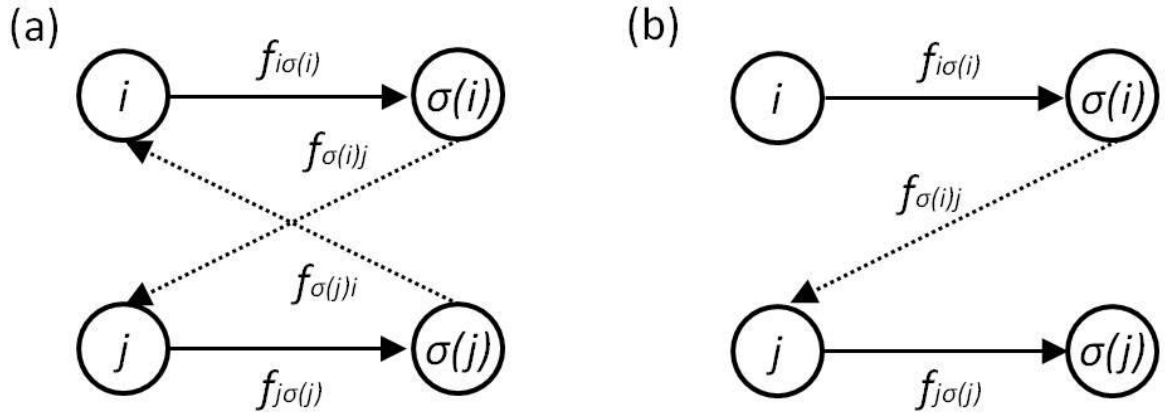


Figure 6.3: Two aircraft approaching the same runway (a) and an ordering decision (b)

In Figure 6.3, i and j are the operations associated the entrance of two aircraft in a runway (blocking resource). In Figure 6.3 (b), i precedesj (arc $< \sigma(i), j >$ is selected) and the aircraft associated to j must respect the minimum separation constraint with the other aircraft: $t_j \geq t_{\sigma(i)} + f_{\sigma(i)j}$. Similarly, if j precedes i: $t_i \geq t_{\sigma(j)} + f_{\sigma(j)i}$.

In Figure 6.4, i and j are the operations corresponding to the entrance of two aircraft in an air segment. The aircraft have a minimum (m) and maximum (z) processing times for each air segment. Given two consecutive operations i and $\sigma(i)$, the fixed arcs $(i, \sigma(i))$ and $(\sigma(i), i)$ model the traversing time constraints: $t_i + f_{i\sigma(i)} \leq t_{\sigma(i)} \leq t_{\sigma(i)} + f_{\sigma(i)i}$. Arc $(i, \sigma(i))$ is weighted m and arc $(\sigma(i), i)$ is weighted $-z$.

The sequencing decisions for two aircraft on an air segment are shown in Figure 6.4. The minimum separation time (or setup time) between the two aircraft is obtained by adding two alternative pairs ($< i, j >, < \sigma(j), \sigma(i) >$) and ($< j, i >, < \sigma(i), \sigma(j) >$). The weights on the alternative arcs correspond to the setup time required between them. Since the entrance and exit order should be the same (no overtaking is allowed in air segments of the TCA), the only two feasible ordering decisions (no positive length cycles in the graph) are shown in Figure 6.4 (a): the selection of both arcs $< i, j >$ and $< \sigma(i), \sigma(j) >$ (entrance time distance f_{ij} and exit time distance $f_{\sigma(i)\sigma(j)}$), or the selection of both arcs $< j, i >$ and $< \sigma(j), \sigma(i) >$ (entrance time distance f_{ji} and exit time distance $f_{\sigma(j)\sigma(i)}$).

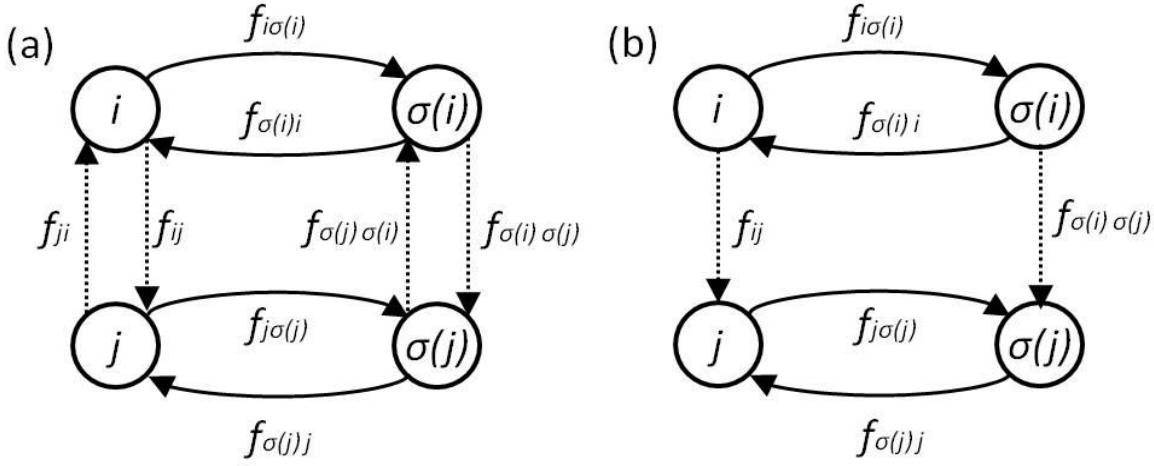


Figure 6.4: Two aircraft approaching the same air segment (a) and an ordering decision (b)

In Figure 6.4 (b), i precedes j (arcs $< i, j >$ and $< \sigma(i), \sigma(j) >$ are selected) and the separation constraint requires that the aircraft associated to j must respect the setup time (to respect the longitudinal distance) both at entrance and at the exit of the air segment: $t_j \geq t_i + f_{ij}$ and $t_{\sigma(j)} \geq t_{\sigma(i)} + f_{\sigma(i)\sigma(j)}$. In case of a common glide path resource, the separation constraint considers the maximum between the longitudinal

and the diagonal distances.

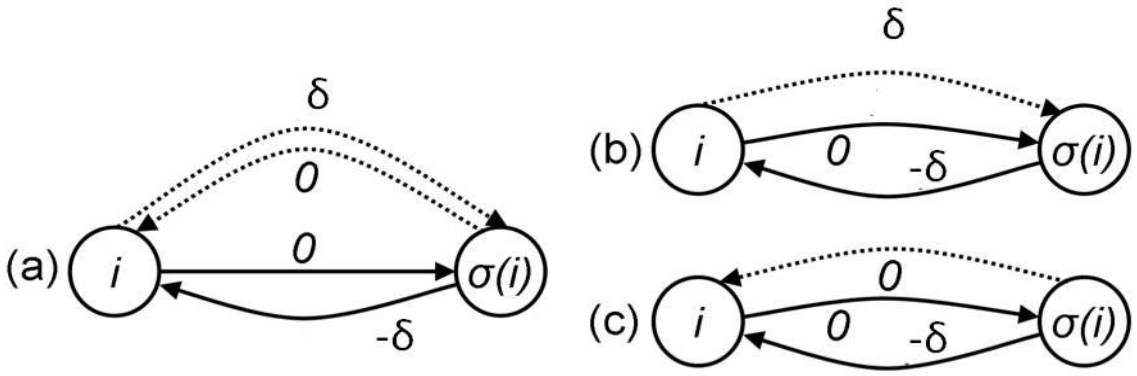


Figure 6.5: Airborne holding with one possible circle (a) and the case in which the circle is selected (b) or is not selected (c)

Figure 6.5 (a) shows the formulation of holding circles on the alternative graph. Let i be the entrance of the aircraft in the holding and $\sigma(i)$ the following operation. On the graph there is a pair of fixed arcs $(i, \sigma(i))$ and $(\sigma(i), i)$, and one alternative pair. The length of $(i, \sigma(i))$ is 0. The length of $(\sigma(i), i)$, instead, is $-\delta$, where δ is the time to perform the holding circle. The alternative pair $((i, \sigma(i)), (\sigma(i), i))$ has arc weights δ and 0, respectively. Figure 6.5 (b) represents the selection of the holding circle while Figure 6.5 (c) is the case with no holding circle.

Figure 6.6 illustrates the remaining constraints of our ASP formulation. Given an operation i, the aircraft entry time is given by arc f_{0i} while the aircraft delay is measured by arc f_{i*}.

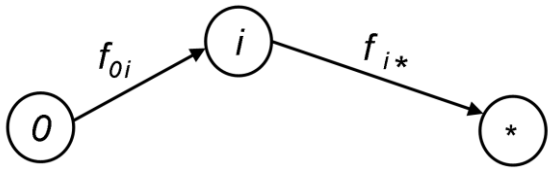


Figure 6.6: Release and due date arcs

Given a forecast r_i on an aircraft entry time in the TCA, with i being the operation associated to the entrance of the aircraft in the TCA, we have the *release arc* $(0, i)$ with weight $f_{0i} = r_i$. For an operation i can be also defined a *due date time* d_i for the associated aircraft. In this case, the *initial delay* I_i of operation i is defined as: $I_i = \max\{0, l^{\emptyset}(0, i) - d_i\}$ and we have a *due date arc* $(i, *)$ with weight $f_{i*} = -d_i - I_i$. Note that the quantity $l^{\emptyset}(0, i)$ represents a lower bound on the start time of operation i in any schedule and therefore the initial delay I_i is unavoidable for operation i. Given any other selection $S \neq \emptyset$, the *total delay*

of operation i is the quantity $\max\{0, l^S(0,i) - d_i\}$ and the *consecutive delay* of operation i is defined as:

$$\max\{0, l^S(0,i) - d_i\} - I_i \tag{6.1}$$

The consecutive delay is the only part of the total delay which depends on the scheduling decisions taken in the TCA. We take into account the consecutive delay of operation i by introducing a fixed arc $(i, *)$ with weight $f_{i*} = -d_i - I_i$, so that the quantity $t_* = l^S(0, *)$ represents the *maximum consecutive delay* associated to schedule S. In this paper the consecutive delay of each landing aircraft is measured both at the entrance and at the exit of the TCA. Therefore, in the remainder of this paper we call *entrance due date* and *exit due date* the due dates associated to the first and to the landing operation of each landing aircraft, respectively. Since we measure the consecutive delay only at the runway for departing aircraft, we associate only the exit due date to these aircraft. Similarly, we call *entrance* or *exit delay* the delay of an aircraft with respect to entrance or exit due dates.

A mathematical program for the ASP can be defined starting from the alternative graph $G = (N, F, A)$. The variables of the problem are $|N|$ real variables t_i equal to the start times of each operation $i \in N$ and $|A|$ binary variables x_{ij} associated to each alternative pair $(< i, j >, < h, k >) \in A$, with:

$$x_{ij} = \begin{cases} 1 & if \quad < i, j > \in S \\ 0 & if \quad < h, k > \in S \end{cases} \tag{6.2}$$

The mathematical program of the ASP is then:

$$\begin{aligned} &\min\ t_* \\ &s.t. \\ &t_q \le t_p + f_{pq} && \forall (p,q) \in F \\ &t_j \le t_i + f_{ij} - M(1 - x_{ij}) && \forall (< i, j >, < h, k >) \in A \\ &t_k \le t_h + f_h k - M x_{ij} \end{aligned} \tag{6.3}$$

In which M is a sufficiently large constant, for example:

$$M = \sum_{(p,q) \in F} \max\{0, f_{pq}\} + \sum_{(<i,j>, <h,k>) \in A} (\max\{0, f_{ij}\} + \max\{0, f_{hk}\}) \tag{6.4}$$

6.4 Rolling horizon framework

This section describes the traffic control system we developed for solving the ASP for a large time horizon of traffic prediction. Since we are dealing with a large number of aircraft, we decompose the problem as follows. The overall time horizon is divided in smaller time horizons of prediction with different start times. The difference between two consecutive start times is named *roll period*. Figure 6.7 shows an example with two consecutive time horizons of prediction. For each time horizon, we run a *single stage optimization* that delivers a scheduling plan based on the currently available and predicted information on the operational conditions. The traffic prediction is repeated till the computation of the overall traffic prediction (*multi-stage optimization*).

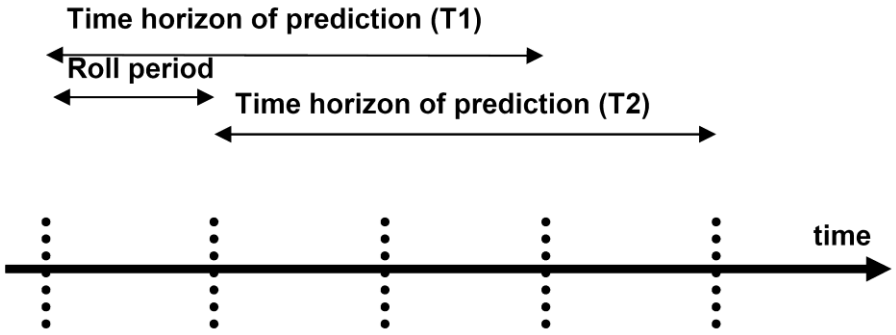


Figure 6.7: Rolling horizon approach: two time horizons of prediction and the corresponding roll period

Figure 6.8 introduces the optimization-based scheduler. Given the airport structure and resources, the current status of traffic, a route (sequence of operations with processing times, setup times, holding times, release times and scheduled arrival times) for each aircraft, the time horizon h of traffic prediction and a start time t_0 of the traffic prediction, an instance generator module creates the xml file containing all the necessary information to compute the alternative graph model of the ASP. Then, the single stage scheduler computes a solution to the ASP in the interval $[t_0, h]$. Solving the problem with a centralized approach, i.e., in a single step, can be achieved by fixing $t_0 = 0$ and $h = tot$, being tot the overall time horizon of traffic prediction.

Figure 6.9 presents the architecture of the multi-stage scheduler. The transition between two consecutive stages is managed by the rolling horizon approach introduced in Figure 6.7. At each stage, a start time of the current time horizon is set according to the roll period. Since at the end of the previous time horizon there may be a number of TCA resources still occupied by aircraft that have not completely processed them, the next time horizon of traffic prediction includes two set of aircraft: (i) all aircraft of the previous steps that are still not fully processed at the start time of the current time horizon (the other aircraft have already

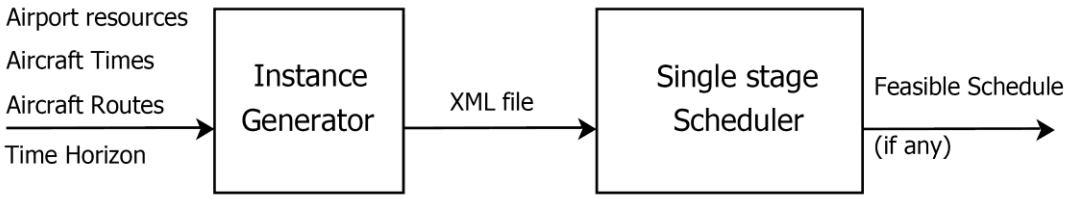


Figure 6.8: Single stage scheduler

finished their landing/take-off procedure at the beginning of the new time horizon, so these are considered completely processed) and (ii) all aircraft entering in the network during the current time horizon. The instance generator procedure thus creates the xml file of the current stage by considering the aircraft from both sets. Specifically, the aircraft of set (i) have shorter routes since we reschedule them starting from the last operation still not fully processed in the previous time horizon, so this operation is considered as the first operation of the shorter route. New (updated) data regarding the aircraft entry time in the TCA are collected in the transition between consecutive stages and used for the generation of the current ASP instance. When the entry time of an aircraft is updated, the release time and the entrance due date are updated accordingly. The single stage solver of Figure 6.8 is then called to solve the current ASP instance.

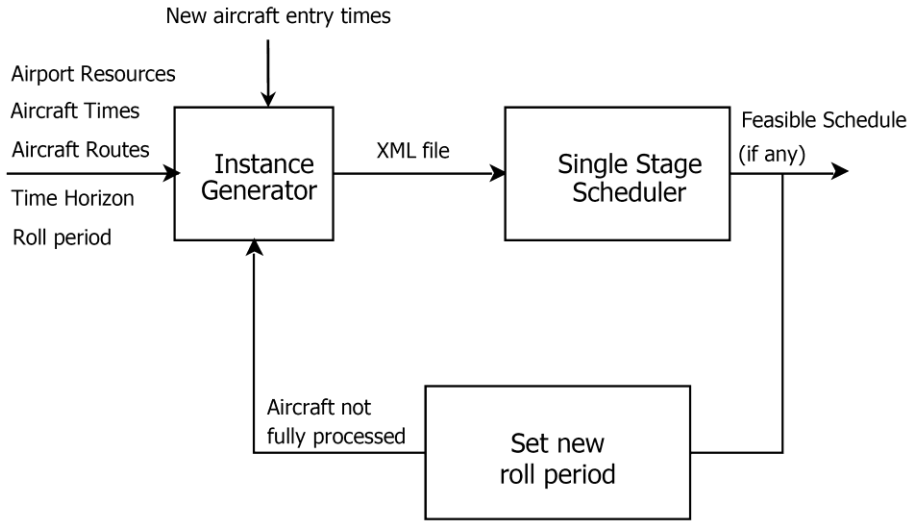


Figure 6.9: Multi-stage scheduler based on the rolling horizon approach

We also analyze the performance of an optional *pre-processing phase* before the single stage scheduler in which we schedule the use of holding circles as follows. If there are aircraft not yet fully processed in TCA, aircraft arriving in the TCA after five minutes from the first operation beginning in the new stage will be

forced to perform holding circles. Potential conflicts between aircraft from the previous stages and aircraft entering shortly in time in the current stage are thus preliminarily solved. The aim of this pre-processing

---

Procedure Multi-stage scheduler (tot, h, r)
Input: Set $tot :=$ overall time of prediction
Set $h :=$ chosen time horizon of prediction
Set $r :=$ chosen rolling timeSet pheromone trails
Begin
Set $t_0 := 0$
while $(t_0 < tot)$ **do**
Begin
Set $P :=$ set of aircraft scheduled in the previous stage
Set $Z := \emptyset$ (set of aircraft not fully processed)
If $(t_0 > 0)$ **then**
Begin
Given the solution found in the previous stage $[t_0 - r, t_0]$
Set $Z :=$ Set new roll period $(P, t_0 - r, r)$
End
Generate the new XML file of the instance with Instance generator (t_0, Z, h)
Build the alternative graph of the instance for the single stage from the XML file (with or without *pre-processing phase*)
Solve the instance by using the Single stage scheduler (BB or FCFS)
Let $t_0 := t_0 + r$
End
End

Procedure Set new roll period (P, t_0, r)
Input:
$P :=$ set of aircraft scheduled in the previous stage
$t_0 :=$ start time of the prediction of the previous stage
$r :=$ chosen rolling time
Begin
Set $w := 0$
for each aircraft of P **do**
Begin
Verify if aircraft A has finished his landing/take-off procedure before $t_0 + r$
If aircraft A is not fully processed **then**
Begin
Consider the first operation i, if any, of aircraft A such that $t_i \leq t_0 + r$ and $t_i + f_{i\sigma(i)} > t_0 + r$
(i.e., i starts before $t_0 + r$ and ends after $t_0 + r$)
If operation i exists (i.e., aircraft A entered the TCA within $t0 + r$) **then**
Begin
Set a new sub-route for aircraft A starting from operation i to the last operation of its route
Set entry time of aircraft A at t_i (i.e., set arc $(0, i)$ with weight $f_{0i} := t_i$)
Add aircraft A to W
End
End
End
Return W
End

Procedure Instance generator (t_0, Z, h)
Input:
t_0 := start time of traffic prediction
Z := set of aircraft not fully processed
h := chosen time horizon of prediction
Begin
for each aircraft entering the TCA after t_0 **do**
    Begin
    Get new entry times in the TCA
    If $t_0 \leq$ aircraft entry time $\leq t_0 + h$ **then**
        Begin
        Add the new aircraft to Z
        End
    End
for each aircraft A of Z **do**
    Begin
    Let i be the first operation of aircraft A
    Set aircraft release time := entry time in the TCA (i.e., set arc $(0, i)$ with weight f_{0i} := aircraft entry time)
    Set aircraft entrance due date := −release time (i.e., set arc (i, *) with weight f_{i*} := aircraft entry time)
    Set the route of aircraft A defining processing, setup and holding times of each operation of A
    Set aircraft exit due date : = −scheduled arrival time
    End
Build XML file for the aircraft in the set Z Return XML file
End

Figure 6.10: Sketch of the rolling horizon framework

phase is to increase the feasibility rate of the single stage scheduler, since a partial schedule is computed beforehand.

The multi-stage procedure lasts till the end of the overall traffic prediction or till infeasibility is found by the single stage scheduler at an intermediate stage. When the automated scheduler does not find a feasible schedule, human traffic controllers are asked to take some actions (possibly interacting with the airlines) that the automated system is not allowed to take, such as the re-routing of incoming aircraft to a different airport, the cancellation of departing flight, or the increase of the maximum time that aircraft can spend in holding circles.

Figure 6.10 presents a detailed pseudo-code of the rolling horizon procedure implemented in this work. Here the main code is the Multi-stage scheduler, which builds the alternative graph model of each single stage instance, by calling the two procedures Set new roll period and Instance generator, and then solves each instance by calling Procedure Single stage scheduler. The aim of Set new roll period procedure is to select the aircraft partially scheduled in the roll period of length r preceding the current stage and to reschedule partially in the current stage. Procedure Instance generator selects the aircraft to schedule in the current stage that are still outside the TCA at the beginning of the current stage, for which the exact

entrance time in the TCA have to be updated at the beginning of the stage (i.e., at time t_0). As for Procedure Single stage scheduler we make use of two algorithms, namely the first come first served (FCFS) rule, often considered a good surrogate for the dispatchers behaviour [Bennell et al., (2011), Bianco et al., (2006)] and the Branch and Bound (BB) algorithm illustrated in [D'Ariano et al., (2010), D'Ariano et al., (2012b)].

6.4.1 Illustrative example

Figure 6.11 gives an illustrative example of the rolling horizon framework for an instance of MXP. The instance is modelled using the alternative graph formulation of Section 6.3. Figure 6.11(a) presents the traffic situation in which there are two landing aircraft (A and B) and one taking-off aircraft (C). Aircraft A arrives from the entry point TOR (resource 1), processes air segments 4, 10, 13, 15 (common glide path) and lands on runway 16. In the graph, the route of aircraft A is depicted as the following list of nodes: A1, A4, A10, A13, A15, A16, Aout. Aircraft B arrives from entry point SRN (resource 3), processes air segment 8, 12, 14, 15 and land on runway 17; aircraft C takes-off from runway 17. For this set of routes, we have two potential conflicts on resources 15 and 17. Decisions need also to be taken regarding the crossing of holding circles TOR and SRN.

Figure 6.11(b) shows the alternative graph at the first stage of the rolling horizon procedure. In the graph, we show the route of each aircraft with a different colour (A red, B blue and C green). We also depicted the release times $(\alpha_A, \alpha_B, \alpha_C)$, the entrance due dates $(\beta_A, \beta_B, \beta_C)$ and the exit due dates $(\gamma_A, \gamma_B, \gamma_C)$. In this time horizon of prediction, i.e., time interval $[0, t]$, only aircraft A and B are modelled in the graph since their entrance in the TCA is scheduled during the time horizon considered, i.e., $0 \leq \alpha_A \leq t$ and $0 \leq \alpha_B \leq t$, while the entrance of aircraft C is scheduled after the end of the current time horizon, i.e., $\alpha_C > t$. Between aircraft A and B we have two alternative pairs representing the ordering decision to be taken at the entrance and exit of air segment 15. Both aircraft A and B have one alternative pair modelling the holding circle decision (see resources A1 and B3).

Figure 6.11(c) illustrates the alternative graph of the second stage of the rolling horizon procedure. Given the roll period r, the second time horizon of traffic prediction is in the time interval $[r, t + r]$. In this graph, aircraft A needs again to be completely processed since $\alpha_A > r$. However, we observe that α_A may change whether the entry time of aircraft A is updated due to new real-time data collected between the first and the second time horizons. If this would have been the case, α'_A = updated entry time of aircraft A and $\beta'_A = -\alpha'_A$.

Differently from aircraft A, at time r aircraft B has already started the landing procedure in the TCA: operations B3, B8 and B12 are completely processed, operation B14 has not been fully processed and operations B15, B17 are still to be processed. Consequently, the route of aircraft B has to be partially

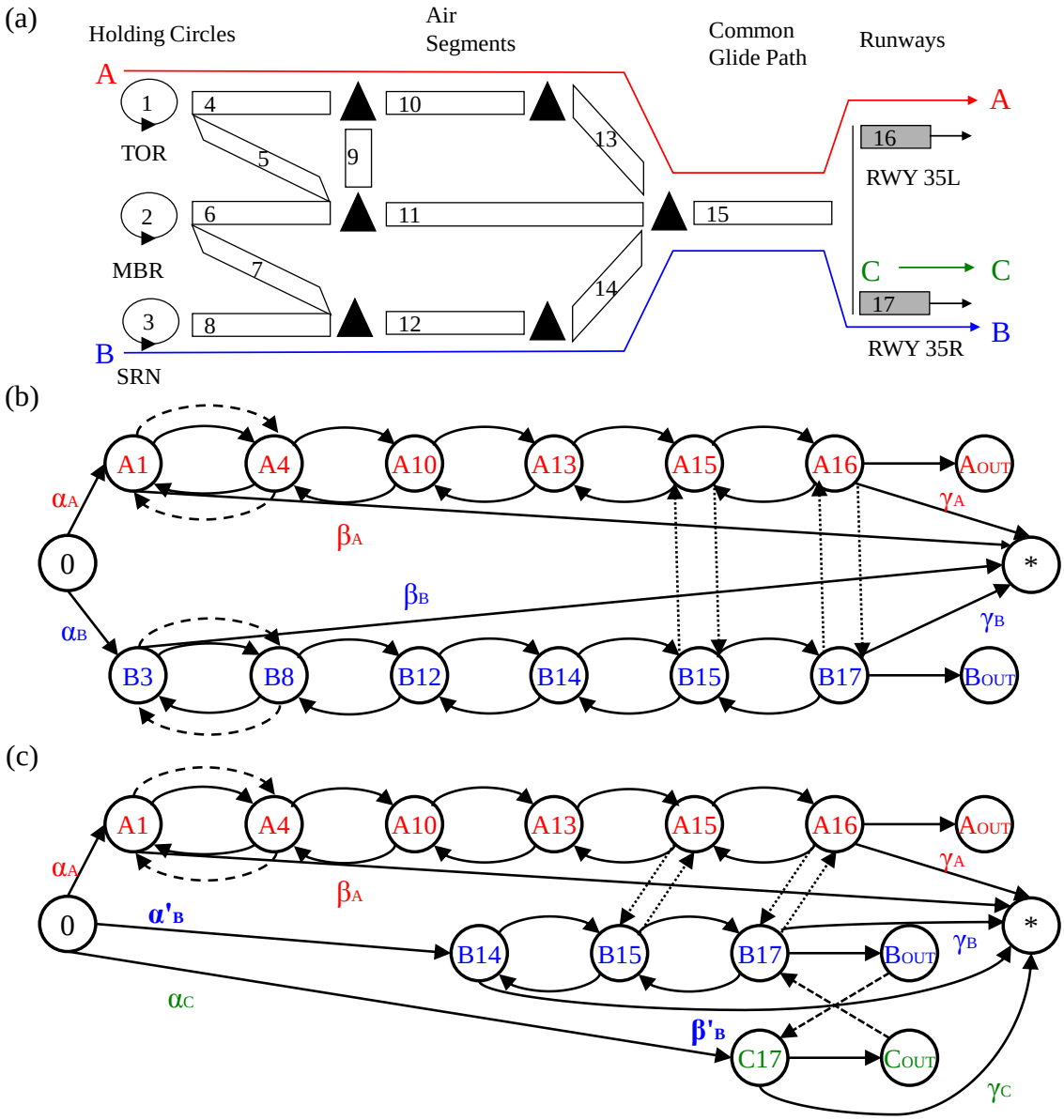


Figure 6.11: A traffic situation at MXP airport (a) and the alternative graph formulation of consecutive stages (b-c)

modelled (B14, B15 and B16) in the graph of the second stage. The release time α_B and the entrance due date β_B are replaced by α'_B = start time of operation B14 in the previous schedule (i.e., start time of node B14 in the solution of the previous stage), and $\beta'_B = -\alpha'_B$. Note that γ_B never changes, since this is the off-line scheduled exit time. Aircraft C enters the network during the second time horizon, i.e., $r \leq \alpha_C \leq t + r$.

Since C is a taking-off aircraft, we only model the exit due date γ_C (the entrance due date is equal to the exit due date). Between aircraft B and C we have one alternative pair representing the ordering decision to be taken at runway 17. The other scheduling decisions need to be taken on the holding circle resource 1 and on the air segment 15.

In this ASP example, we do not constraint the scheduling decisions between stages. Considering resources 1 and 15, aircraft A may be scheduled first on runway 15 (first A then B) and may not run holding circle 1 at the first stage, while A may be schedule after B on runway 15 and may run holding circle 1 at the second stage.

6.5 Experimental Results

We tested the traffic control system on a laboratory environment with Linux operating system. The instances are based on real traffic data of the two airports. Specifically, the experiments of Sections 6.5.1, 6.5.2 and 6.5.3 use static information (all aircraft information is known at time 0) and are executed on a processor Intel i7 (2.84 GHz), 8 GB Ram. The experiments of Sections 6.5.4 use dynamic (updated) information and are executed on a processor Intel Dual Core (3 GHz), 2 GB Ram. The algorithms are implemented in *AGLibrary*, software developed by the Operations Research group of Roma Tre University for solving complex scheduling and routing problems.

The next subsections will provide a description of the ASP instances for each airport, computational experiments for various settings of the rolling horizon framework and a centralized system developed in [D'Ariano et al., (2010), D'Ariano et al., (2012b)]. Also, a detailed quantitative comparison between the branch and bound (BB) algorithm and the first come first served (FCFS) rule will be shown for various sources of disturbances.

6.5.1 ASP instances

The instances considered in this paper are derived from real scheduling data regarding traffic flows in Rome Fiumicino (FCO) and Milan Malpensa (MXP) terminal control areas. Each instance simulates a 3 hour time horizon of traffic prediction. We consider 20 ASP instances for each airport: 10 instances have light disturbances (up to 5 minute entrance delays are generated randomly for 1 to 5 aircraft every 15 to 60 minutes) and the other 10 have heavy disturbances (up to 15 minute random entrance delays distributed similarly to light disturbances).

Table 6.1 gives further information on the 20 ASP instances. For each instance, we simulate 3 hours of traffic flow prediction. Column 1 reports the airport considered. Columns 2-3 give the number of taking-off

and landing aircraft. Columns 4-6 present average information on the maximum and average entrance delays (in seconds), and the total travel time spent by all aircraft in the free-network (*free-net* TTS, in seconds). We compute free-net TTS as the sum of travel times when each aircraft processes all resources of its route without consecutive delays.

Table 6.1: ASP instances of Roma Fiumicino (FCO) and Milano Malpensa (MXP) airports for 3 hours of traffic prediction

Airport	Taking-off Aircraft	Landing Aircraft	Max Entrance Delay (sec)	Avg Entrance Delay (sec)	Free-Net TTS (sec)
FCO	48	96	585	8	76002
MXP	51	66	895	53	58518

From Table 6.1, we observed that the FCO instances present a lower value of the entrance delays than the MXP instances. However, the aircraft require a longer travel time in the FCO terminal control area and runways are more densely used at FCO airport due to a higher traffic demand.

6.5.2 System Configurations

Table 6.2 shows the results obtained for various multi-stage configurations of the rolling horizon approach and for the centralized approach (only one single stage). For both approaches, we use the branch and bound (BB) of [D'Ariano et al., (2010), D'Ariano et al., (2012b)] as scheduler. In the rolling horizon approach, BB is truncated after 60 seconds of computation, while in the centralized approach after 800 seconds of computation.

Each row of Table 6.2 gives average results at the end of 3 hours of traffic flow prediction for the 40 ASP instances of Section 6.5.1. Column 1 indicates the airport considered: FCO or MXP. Column 2 gives the solution approach: rolling horizon or centralized. Column 3 and 4 report the time horizon of traffic prediction (look ahead period) and the roll period (in minutes). We consider 9 combinations of the two parameters: 3 time horizons representing a small, medium or large period of traffic prediction (15, 30, 45 minutes) and 3 roll periods representing a 2/3, 1/3 or no overlap of take-off and landing procedures between two consecutive time horizons (5, 10, 15 minutes). For example, for the 15 minute time horizon and the three roll periods, we have overlaps of 10, 5 and 0 minutes, respectively. For fixed values of time horizon (Column 3) and roll period (Column 4), Columns 5 gives the number roll periods in the rolling horizon procedure (i.e., the number of calls to the single stage scheduling procedure performed by the multi stage scheduler).

Columns 6-11 present the computational results. For each column, the best configuration is shown in bold. In Columns 6, the average computation time (in seconds) for the single stage scheduler is reported while in Column 7 the multi-stage approach at the end of the overall traffic prediction. Columns 8-11 report

information on aircraft delays and travel times computed at time tot. Specifically, Column 8 gives the maximum consecutive delay, Columns 8 and 9 the average total delay at entrance and exit of the TCA (all delays are in seconds). In the multi-stage approach the aircraft delays are collected at the last stage in which each aircraft is scheduled in the TCA.

Table 6.2: Assessment of various configurations of the rolling horizon framework

Airport	Approach	Time Horizon (min)	Roll Period (min)	Number of Roll Periods	Computation Time (sec) Single Stage	Computation Time (sec) Multi Stage	Maximum Consecutive Delay (sec)	Avg Total Entrance Delay (sec)	Avg Total Exit Delay (sec)	Delta TTS (sec)
FCO	Rolling	15	5	34	19.6	668.1	**813**	**112**	**120**	**8376**
	Horizon	15	10	18	23.4	421.7	881	117	211	9253
		15	15	12	25.2	302.5	886	119	132	8586
		30	10	16	52.3	836.6	864	118	194	9043
		30	20	9	52.7	474.7	879	122	128	8558
		30	30	6	50.1	300.9	879	124	207	8897
		45	15	10	59.5	594.8	881	121	127	8932
		45	30	6	59.1	354.8	879	125	158	8872
		45	45	4	58.7	**234.9**	884	122	131	7567
	Centralized	180	-	1	800	-	1135	130	223	14968
MXP	Rolling	15	5	34	10.3	350.3	504	89	68	7334
	Horizon	15	10	18	**6**	**108.6**	**498**	**87**	**66**	7646
		15	15	12	12.1	145.2	511	90	69	7811
		30	10	16	52.1	832.9	511	89	68	6865
		30	20	9	54	485.8	515	90	69	7376
		30	30	6	51.9	311.5	512	90	68	6864
		45	15	10	53.1	531.4	501	89	67	7209
		45	30	6	52.2	313.4	512	90	69	6892
		45	45	4	57.1	228.4	501	89	68	7150
	Centralized	180	-	1	800	-	505	91	70	8604

Column 11 gives the total increase (*delta TTS*, in seconds) of travel time spent by all aircraft in the TCA besides their minimum traversing time (free-net TTS). We compute the delta TTS for an aircraft a landing at a runway r as follows. Let t_{ar} and t_{ae} denote the actual landing time at r and the actual entry time in the TCA, respectively. Let also τ_{ar} and τ_{ae} denote the minimum landing time at r and the minimum entry time in the TCA, computed disregarding the presence of the other aircraft. Then, the delta TTS of aircraft a is $(t_{ar} - t_{ae}) - (\tau_{ar} - \tau_{ae})$. The computation is similar for departing aircraft. The delta TTS indicator is interesting for energy consumption reasons, since the smaller is the delta TTS of aircraft in the TCA, the smaller is the overall energy consumption.

From the results obtained in Table 6.2 regarding the computation time, the single stage scheduler (Column 6) is faster for small time horizons compared to large time horizons, since there are less aircraft in the corresponding alternative graphs and thus less aircraft ordering decisions to be taken. We recall that BB algorithm has up to 60 seconds to compute its best scheduling solution, so a smaller computation time means that BB found the optimal solution before the given time limit of computation.

The multi-stage scheduler (Column 7) is often faster for larger roll periods, because less calls (Column 5) to the single stage scheduler are required in order to complete the forecast of the overall time horizon of traffic prediction. Another important reason is that for large roll periods there is less overlap between consecutive time horizons, and thus less aircraft that are not fully processed at the end of each stage and need to be rescheduled in the successive stages. As a result, when comparing instances with the same time horizon, the alternative graphs of large roll periods have, on average, less jobs of previous time horizons (i.e., partial aircraft routes) compared to the graphs of small roll periods. In general, we obtain a different best configuration for each airport.

Comparing the results obtained in the last four columns of Table 6.2, we observe that the rolling horizon approach outperforms the centralized approach. This result is more evident for the FCO instances where the exit total delays and travel times are more than 40% less when using the rolling horizon approach. For both TCAs, the maximum consecutive delay (i.e., the objective function of the centralized approach) is also better minimized by the rolling horizon approach, since the centralized one is not able to find better solutions within 800 seconds.

In the rolling horizon approach, the smallest time horizon yields the best results for all instances (shown in bold in Table 6.2). This behaviour is likely due to two main reasons. One reason is the limited computation time allowed to the BB algorithm to compute a solution for the single stage scheduler. As a consequence, larger instances have fewer chances to be solved at optimality. This behaviour is especially evident for the centralized approach compared to all rolling horizon approaches. The second reason is the very constrained structure of ASP, so that each aircraft likely interacts with few preceding/following aircraft. In this context, minimizing the maximum consecutive delay for an instance with a large time horizon may lead to increase the average total delay of some aircraft in the first roll period in order to minimize the maximum consecutive delay of the following aircraft. However, only the schedule of the first roll period will be actually implemented, while the following aircraft will be rescheduled in the next stage with other incoming aircraft. Therefore, increasing the time horizon may cause a worsening of the average total delay of the overall multistage scheduler.

When comparing the different rolling horizon configurations in terms of the total increase of travel times in the TCA (delta TTS, Column 11), there is no full correspondence with the best configurations for delay minimization. The different behaviour is due to the following reason: the scheduled arrival time of each aircraft at each runway is computed by using the maximum processing times in each air segment, while during operations we assume that the aircraft process all air segments with minimum processing time in order to recover from possible delays. As a result, some aircraft can arrive before scheduled time and thus the aircraft is considered on-time (consecutive and total delays are zero) even if there is an increase of the

delta TTS.

The differences between average entrance delays and average total delays are next discussed. For both airports, there is a large total entrance delay in Table 6.2 compared to the entrance delay of Table 6.1. In fact, aircraft are largely delayed at their entrance in the TCA in order to avoid potential conflicts on the first air segment of their route in the TCA. At FCO, comparing Columns 9 and 10, the total delay increases from the entrance to the exit of the TCA due to further conflicts in other TCA resources. Differently, at MXP the total delay decreases, since there are a few conflicts in the TCA and aircraft traverse the TCA with their minimum processing times.

6.5.3 FCFS versus BB: Static information

This section compares the FCFS heuristic with the branch and bound (BB), both enrolled in the rolling horizon framework. The tests are based on the 40 ASP instances of Table 6.1. We assume that aircraft information is known at time 0. For each airport, we consider the best rolling horizon configurations of Table 6.2 with respect to the maximum delay minimization and the total entrance/exit delay minimization: 15 minute time horizon and 5 minute roll period for FCO and 15 minute time horizon and 10 minute roll period for MXP.

Table 6.3 compares FCFS and BB with (ON) or without (OFF) pre-processing (see Column 3). Column 4 gives the computation time of the multi stage procedure (in seconds), Column 5 the maximum consecutive delay (in seconds), Columns 6 and 7 the average total entrance and exit delay (in seconds), Column 8 the delta TTS (in seconds). For Columns 5-8, the best configuration for each airport and algorithm is reported in bold.

Table 6.3: Comparison of four configurations of the rolling horizon framework

Airport	Scheduler	Pre Processing Phase	Multi Stage Computation Time (sec)	Maximum Consecutive Delay (sec)	Avg Entrance Entrance (sec)	Avg Total Exit Delay (sec)	Delta TTS (sec)
FCO	FCFS	ON	0.34	**1480**	**166**	**610**	**75528**
	FCFS	OFF	-	-	-	-	-
	BB	ON	646	898	116	216	25754
	BB	OFF	668.1	**813**	**112**	**120**	**8376**
MXP	FCFS	ON	1.09	**1126**	96	**381**	**52633**
	FCFS	OFF	0.99	1148	**93**	385	53468
	BB	ON	180.3	528	98	90	**6903**
	BB	OFF	108.6	**498**	**87**	**66**	7646

For this set of ASP instances, FCFS with the pre-processing procedure gives better results than without

the pre-processing for most of delay and travel time indicators, since this procedure helps in the resolution of potential conflicts on the first air segment of the TCA. At FCO airport, FCFS without pre-processing is not able to compute feasible schedules for more than half of the delay configurations. At MXP, FCFS is always able to compute a feasible schedule since this network is less dense of traffic. Regarding the optimization algorithm, we obtain that the use of pre-processing improves the BB results in terms of delta TTS at MXP airport only, it is better not to use pre-processing with BB for all the other indicators. We conclude that BB is better without pre-processing while FCFS is better with pre-processing. This is due to the myopic rescheduling strategy of FCFS.

Overall, the rolling horizon framework is very fast (less than 1 second) when FCFS is used in the single stage scheduler. However, BB clearly outperforms FCFS in terms of delay and travel time minimization, but at the cost of a greater computation time. A smaller time limit could be given to BB at each stage of the rolling horizon framework in order to reduce the multi stage time.

Table 6.4: Analysis of the number of changed scheduling decisions between consecutive stages

Airport	Scheduler	Time Horizon (min)	Roll Period (min)	Schedule Changes	Holding Circles	Air Segments	Runways
	FCFS	15	5	60.65	53.8	2.35	4.5
	BB	15	5	27.35	2.35	25	0
		15	10	18.5	0.25	18.25	0
		15	15	14.65	0.2	14.45	0
		30	10	11.45	0.8	10.65	0
		30	20	12.15	0.05	12.1	0
		30	30	9.35	0.05	9.1	0.2
		45	15	9.45	0.1	9.35	0
		45	30	9.35	0.05	9.1	0.2
		45	45	**2.25**	**0**	**2.25**	**0**
FCO	FCFS	15	10	8.45	8.15	0.3	0
	BB	15	5	11.7	3	8.7	0
		15	10	5.9	2.45	3.45	0
		15	15	5.6	2.4	3.2	0
		30	10	5.55	2.3	3.25	0
MXP		30	20	**0.4**	**0.3**	**0.1**	**0**
		30	30	5.25	2.05	3.2	0
		45	15	3.7	1.45	2.25	0
		45	30	5.25	2.05	3.2	0
		45	45	1.95	0.9	1.05	0

Table 6.4 shows the number of schedule changes between consecutive stages of the rolling horizon framework. We measure this value for both airports and scheduling algorithms. Each row gives average results over the 40 ASP instances at the end of the three-hour traffic flow prediction. Specifically, Column 1 indicates the airport, Column 2 the scheduling algorithm, Columns 3 and 4 the chosen time horizon and roll

period (both in minutes), Column 5 the overall number of changed scheduling decisions between consecutive stages. Columns 6, 7 and 8 report the number of changed decisions at the holding circles, air segments and runways. For FCFS, we only report the best system configuration since this heuristic rule does not always compute a feasible schedule.

Table 6.5: Comparison of the solution approaches in terms of the alternative graph size

Airport	Approach/ Scheduler	Time Horizon (min)	Roll Period (min)	Graph Nodes	Graph Arcs	Alternative Pairs
FCO	Centralized/any			773	54265	26385
	Rolling Hor./FCFS	15	5	141	2201	964
	Rolling Horizon/ BB	15	5	102	1193	499
		15	10	97	1113	464
		15	15	105	1281	540
		30	10	165	2844	1263
		30	20	158	2603	1150
		30	30	173	3147	1408
		45	15	234	5449	2499
		45	30	211	4559	2077
		45	45	224	5044	2306
MXP	Centralized/any			557	28027	13482
	Rolling Hor./FCFS	15	10	102	1274	539
	Rolling Horizon/ BB	15	5	78	764	308
		15	10	75	713	286
		15	15	80	805	327
		30	10	126	1760	760
		30	20	122	1703	735
		30	30	122	1633	701
		45	15	173	3074	1372
		45	30	162	2704	1198
		45	45	170	2986	1331

The results of Table 6.4 show that FCFS is more nervous than BB in terms of changed scheduling decisions. In fact, FCFS gives precedence to the aircraft entering first the TCA, while BB is based on look-ahead decisions. This result is more evident at FCO airport, where most of the changes regard the holding circle resources. In fact, delaying aircraft before entering the TCA is easier since the travel times in the TCA are strongly constrained by minimum and maximum processing times. Regarding BB combined the different rolling horizon framework configurations, we obtained that BB for small time horizons and roll periods often generated more changes of scheduling decisions, because a larger number of single stage iterations are required. The largest number of schedule changes is detected on the air segments, since these are the largest group of resources in the TCA.

From the results of Tables 6.2 and 6.4 (best configurations are shown in bold), we conclude that a trade-off exists between the best rolling horizon configurations in terms of delay minimization (obtained for small

time horizons and roll periods, see Table 6.2) and the less nervous configurations in terms of number of changed scheduling decisions (obtained for large time horizons and roll periods, see Table 6.4).

Table 6.5 presents the average size of the alternative graphs generated for the resolution of the 40 ASP instances of Table 6.1. We analyze the FCO and MXP airports, the centralized and rolling horizon approaches and the two scheduling algorithms (BB and FCFS). For each configuration, we report the number of nodes (Column 5), the number of arcs (Column 6) and the number of alternative pairs (Column 7). As for Table 6.4, we give the best rolling horizon configuration for FCFS and compare all the rolling horizon configurations for BB.

Evaluating the different graph sizes in Table 6.5, the centralized approach requires a huge number of scheduling decisions compared to the rolling horizon approaches (one decision is required for each alternative pair). This is due to the global conflict resolution approach proposed by the centralized approach. Regarding the rolling horizon configurations, the number of scheduling decisions increase (on average) with the increase of the size of the time horizon, since there is a significant larger number of aircraft entering the system every 15 minutes. When the length of the roll period is increased, the resulting graph does not necessarily present a larger number of alternative pairs (i.e., scheduling decisions). In fact, the number of scheduling decisions taken in each stage depends mostly on the time horizon rather than on the rescheduling frequency.

The scheduler used in the rolling horizon framework has a clear impact on the graph size. Both at FCO and MXP TCAs, when comparing FCFS and BB for the same rolling horizon configurations, FCFS requires managing alternative graphs with a larger number of nodes and arcs. This fact is motivated by the behaviour of FCFS, which takes myopic decisions and delays more aircraft than BB. The delayed aircraft need to be rescheduled in multiple stages and, thus, there are additional (partially processed) jobs at each stage.

6.5.4 FCFS versus BB: Dynamic information

This section evaluates the performance of FCFS and BB when the entrance delay of each incoming aircraft is updated during the approach phase and finally fixed only when the aircraft actually enters the TCA. The set of experiments presented in this section (as shown later in Table 6.6) is obtained by from the 40 ASP instances of Table 6.1 (20 per TCA) by considering the first hour of traffic prediction only. For each instance, we analyze a number of cases with dynamic information, relevant for real-time applications, in which the arrival time of each aircraft at the TCA is a stochastic variable. In the dynamic setting, a small random variation is generated after every 60 seconds in a range $[-\delta; +\delta]$, where δ indicates the maximum deviation for all incoming and outgoing aircraft. In this setting, the expected value estimated at time 0 is thus updated along the time and finally fixed when the aircraft enters the TCA.

The choice of a shorter time horizon of traffic prediction (1 hour instead of 3 hours) compared to the

Table 6.6: ASP instances of Roma Fiumicino (FCO) and Milano Malpensa (MXP) airports for 1 hour of traffic prediction

Airport	Taking-off Aircraft	Landing Aircraft	Max Entrance Delay (sec)	Avg Entrance Delay (sec)	Free-Net TTS (sec)
FCO	32	16	582	7	25334
MXP	23	17	895	63	19506

instances of Table 6.1 is motivated by the need of studying traffic flows of interest for real-time operations, in which the traffic forecast is usually limited to short time horizons of traffic prediction.

Table 6.7: Comparison of algorithms when dealing with or without deviation of the entry time at FCO airport

Scheduler	Entry Time Deviation δ (sec)	Total Computation Time (sec)	Schedule Changes	Maximum Consecutive Delay (sec)	Avg Total Entrance Delay (sec)	Avg Total Exit Delay (sec)	Delta TTS (sec)
FCFS	0	0.01	11.9	1246	104	516	23812
	[- 3; 3]	0.01	22.5	1208	125	520	22843
	[- 6; 6]	0.01	26.1	1210	130	514	22123
BB	0	24.5	3.6	872	104	107	2402
	[- 3; 3]	28	9.7	905	130	136	2927
	[- 6; 6]	28.4	10.6	906	134	138	3041

Starting from the 40 ASP instances of Table 6.6 with static information, we generated 800 ASP instances as follows. For each static instance, 10 dynamic instances are generated with random variations in the range [-3 sec; +3 sec] and other 10 dynamic instances with random variations in the range [-6 sec; +6 sec]. In Table 6.6, we report the same type of information as for Table 6.1.

Regarding the scheduling algorithms, the rolling horizon approach is combined either with FCFS or BB truncated after 60 seconds of computation. For each airport, we consider the best rolling horizon configuration reported in Table 6.2. Since this set of experiments deal with one hour traffic predictions, the rolling horizon framework requires 10 (6) roll periods to solve each ASP instance for the TCA of FCO (MXP).

Tables 6.7 and 6.8 give the computational results obtained for each airport. Each instance is solved by the rolling horizon approach and by the two algorithms: FCFS and BB. For each algorithm, we tested the 40 static instances of Table 6.6 and the 800 dynamic instances described above. Differently from BB, FCFS does not always compute a feasible schedule for the FCO instances. Therefore, Table 6.7 reports the average results on the 391 (up to 420) FCO instances solved by FCFS, while Table 6.8 reports the average results over all the 420 MXP instances. Tables 6.7 and 6.8 are organized as follows. Column 1 presents the scheduling algorithm, Column 2 the [min, max] range of entry time deviation, Column 3 the total computation time of

Table 6.8: Comparison of algorithms when dealing with or without deviation of the entry time at MXP airport

Scheduler	Entry Time Deviation δ	Total Computation Time	Schedule Changes	Maximum Consecutive Delay	Avg Total Entrance Delay	Avg Total Exit Delay	Delta TTS
FCFS	0	0.01	7.7	723	97	271	13645
	[- 3; 3]	0.01	8.8	799	115	294	13850
	[- 6; 6]	0.01	8.7	810	118	296	13848
BB	0	4	6.2	498	97	75	2751
	[- 3; 3]	7.4	6.4	505	111	81	2870
	[- 6; 6]	8.6	6.2	505	113	82	2906

each algorithm, Column 4 the number of schedule changes between consecutive time horizons, Columns 5-7 the consecutive and total delays, Column 8 the travel time spent in the TCA. All values but the schedule changes are reported in seconds.

From the results of Tables 6.7 and 6.8, we observe that both algorithms are very fast to compute a feasible solution (if any in case of FCFS). Regarding the static instances (the entry time is fixed at time 0 for all aircraft), BB clearly outperforms FCFS in terms of all performance indicators with the exception of the average total entrance delay. For the latter indicator, we have similar results for the two algorithms, but BB slightly increases the entrance delay of some aircraft in order to significantly reduce exit delays. Comparing the results obtained for the static and dynamic cases, FCFS presents, as expected, a slightly larger variability of the number of schedule changes compared to BB, and both algorithms are sensitive to random variability of aircraft entry times in the TCA. However, the quality gap between FCFS and BB is stable for all performance indicators.

6.6 Conclusions

This paper investigates the potential of a dynamic aircraft traffic control system for dealing with disturbed traffic situations at the two major Italian TCAs. The main contribution of this paper to the related literature is an innovative combination of the rolling horizon concept (enabling the dynamic management of aircraft ingoing and outgoing the TCA, in which data variability is inherently associated with incomplete information on the real-time operations [Hu and Chen (2005), Lai et al., (2008), Meng and Zhou (2011), Zhan and Zhang (2010)], a detailed mathematical model (enriching the job shop scheduling model of [Bianco et al., (2006)] by including additional real-world constraints) and a branch and bound algorithm (introduced in [D'Ariano et al., (2010), D'Ariano et al., (2012b)] for the static ASP). Specifically, the new rolling horizon framework makes use of alternative graphs for the detailed formulation of the static and dynamic ASP, solved via

heuristic and exact algorithms. Computational experiments, carried out on practical instances with static and dynamic information, show the remarkable potential of this combination for the purpose of controlling air traffic in complex and busy TCAs.

Computational results demonstrate the effectiveness of our rolling horizon approach compared to a centralized approach. Regarding the scheduling algorithms, we quantified the advantage of using BB compared with FCFS, in terms of delay and travel time minimization. A remarkable result is that the rolling horizon approach solved by BB achieved better results than FCFS both when aircraft entry times are fixed at time 0 (static case) or are updated during the time (dynamic case). Furthermore, BB requires less changes of scheduling decisions than FCFS during consecutive look-ahead periods and it is more robust to disturbances of aircraft entry times. Regarding the computation time, the rolling horizon configurations with FCFS are very fast to compute a feasible solution (if any) compared to BB, even for the three-hour ASP instances. However, one-hour ASP instances are quickly solved by the BB-based rolling horizon approach also.

Further research should be dedicated to the development of a link with real-time information coming from the traffic control system to close the loop with operations, as well as to the implementation of different system settings (e.g. dynamic roll periods) and aircraft reordering and rerouting algorithms for handling even more severe traffic disturbances. An interesting modification of the mathematical formulation would be the modification of the objective function, e.g. by inserting different weights for the aircraft delays, in order to take into account that the variability of aircraft entry times increase with the distance of the aircraft from the TCA.

Acknowledgment

This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB “Advanced tracking system in intermodal freight transportation”.

Chapter 7

Optimal aircraft scheduling and routing at a terminal control area during disturbances

Abstract: *This paper addresses the real-time problem of aircraft scheduling and routing in terminal control area. A main task of traffic controllers is to mitigate the effects of severe traffic disturbances on the day of operations in the Terminal Control Area (TCA) of an airport. When managing disturbed take-off and landing operations, they need to minimize the delay propagation and, in addition, to reduce the aircraft travel time and energy consumption. The paper tackles the problem of developing effective decision support tools for air traffic monitoring and control in a busy TCA. To this purpose, centralized and rolling horizon traffic control paradigms are implemented and compared. The mathematical formulation is a detailed model of air traffic flows in the TCA based on alternative graphs, that are generalized disjunctive graphs. As for the aircraft scheduling and (re-)routing approaches, the First-In-First-Out (FIFO) rule, used as a surrogate for the behavior of air traffic controllers, is compared with various optimization-based approaches including a branch and bound algorithm for aircraft scheduling with fixed routes, a combined branch and bound and tabu search algorithm for aircraft scheduling and re-routing, and a mixed integer linear programming formulation for simultaneous scheduling and routing. Various hypothetical disturbance scenarios are simulated for a real-world airport case study, Milano Malpensa, and the proposed timing and routing approaches are compared in terms of their performance in the different scenarios. The disturbed traffic situations are generated by simulating multiple delayed arriving/departing aircraft and a temporarily disrupted runway. In general, the optimization approaches are found to improve the solutions significantly compared to FIFO, in terms of aircraft delay minimization. However, there are some trade-offs involved in picking the right approach and paradigms for practical implementations.*

7.1 Introduction

An increasing problem that air traffic controllers have to face is the growth of traffic demand while the availability of new airport resources is very limited. Aviation authorities are thus seeking methods to better use the infrastructure and to better manage aircraft movements in the proximity of airports, improving aircraft punctuality and respecting all safety regulations [Pellegrini and Rodriguez (2013)].

This paper deals with the development of advanced optimization approaches for improving the real-time management of severely disturbed aircraft operations at busy airports. Terminal area operations are usually considered under the umbrella of Air Traffic Control (ATC) because they are managed by local airport controllers. From a logical point of view, ATC decisions in a Terminal Control Area (TCA) can be broadly divided into: (i) Routing decisions, where an origin-destination route for each aircraft has to be chosen regarding air segments and runways; (ii) Timing decisions, where routes are fixed under traffic regulation constraints and an aircraft passing timing has to be determined in each air segment, runway and (possibly) holding circle. In practice, routing and timing decisions in a TCA are taken simultaneously and a given performance index is optimized. The main objective of routing decisions is typically to balance the use of critical resources while that of the whole process is to limit the propagation of disturbances across flight legs either due to aircraft, crew, or passenger considerations [Ball et al., (2007)].

Decision Support Systems (DSSs) based on optimization may help to exploit to the fullest the capacity available in a TCA during operations. The improvement of take-off/landing operations is an important factor related to the performance of the entire ATC system. However, ATC decisions are still mainly taken by human controllers with only a limited aid from automated systems [Djokic et al., (2010), Kim et al., (2009), Prevot et al., (2011)]. In most cases, computer support only consists of a graphical view of the current aircraft position and speed. As a result, delays are not effectively limited during landing and take-off operations. The optimization-based DSS developed in this work may support controllers to dynamically exploit at most the capacity available in the TCA during severe disturbances and busy traffic.

Landing aircraft move along predefined routes from an entrance point in the TCA to a runway following a standard descent profile. During all the approach phases, a minimum separation between every pair of consecutive aircraft must be guaranteed. This standard separation depends on the types and relative positions of the two aircraft (at the same or different altitude). By considering the different aircraft speeds, the safety distance can be translated in a separation time. Similarly, departing aircraft leave the runway moving towards the assigned exit point from the TCA along an ascent profile, respecting separation standards. The runway can be occupied by only one aircraft at a time, and a separation time should be ensured between any pair of aircraft. Once a landing/take-off aircraft enters the TCA it should proceed to the runway. However,

airborne holding circles (ground holding) can be used to make aircraft wait in flight (at ground level) until they can be guided into the landing (take-off) sequence. Real-time traffic management copes with potential aircraft conflicts by adjusting the off-line plan in terms of re-timing, re-ordering, re-routing and holding actions. A potential conflict occurs whenever aircraft traversing the same resource (i.e. air segment or runway) do not respect the minimum separation time required for safety reasons. Separation times depend not only on the aircraft sequence but also on the route chosen for consecutive aircraft in each TCA resource and the aircraft types (we consider three aircraft categories: small, medium and large).

The problem of reacting to disturbed traffic conditions is a key issue in air traffic control practice [Prevot et al., (2011),Taylor and Wanke (2011)]. This paper focuses on the real-time control problem to provide optimal conflict-free airborne decisions at the TCA. Similar problems are also studied in railway transportation field for re-ordering and re-routing problems [D'Ariano et al., (2008),Pellegrini and Rodriguez (2013)]. However, the two types of problems have a quite different structure and require careful adaptation of existing solution frameworks, mathematical models and algorithmic methods.

In previous works of our research group, we developed a branch and bound algorithm for the Air Traffic Control problem in a Terminal Control Area (ATC-TCA) problem with fixed routes, in which aircraft routes are decided at preliminary step [D'Ariano et al., (2010)]. In a recent work, we developed an iterative approach for solving the ATC-TCA problem with flexible routes [D'Ariano et al., (2012b), D'Ariano et al., (2012a)]. Given a route for each aircraft, a scheduling approach takes the aircraft sequencing decisions and assigns the start time to each operation. A re-routing approach then searches for better aircraft routes. From our previous research [D'Ariano et al., (2012a)], a better performance has been observed when using runway re-routing compared to the re-routing of other TCA resources.

The objective of this work is to investigate the potential delay reduction achievable by optimization-based solvers with respect to the most common approach used in practice for real-time aircraft scheduling and routing at a busy and complex TCA, in presence of severe disturbances and even for large time horizons of traffic predictions. To this aim, this paper presents a number of modeling and algorithmic contributions. The original contributions are the next summarized:

- A new formulation is proposed for the simultaneous aircraft scheduling and routing problem.
- The problem is solved via various solution approaches: 1. a commercial solver, 2. an optimization solver based on a problem decomposition in re-timing, re-ordering and re-routing decisions, 3. a temporal decomposition of the overall problem, 4. a number of combinations of the proposed approaches. Specifically, the approach 2 is based on the algorithms developed by D'Ariano et al. [D'Ariano et al., (2010), D'Ariano et al., (2012b)], and is extended in the current paper by means of a procedure to speed-up the search of a feasible schedule. The approach 3 was presented in Samà et al. [Samà et al.,

(2013a)]. Approach 4 extends the approach 3 to deal with routing flexibility.

- Three model variants are proposed to study different objective functions and user requirements.
- The three model variants and the various solution approaches are compared in the computational results section. We tested 80 practical-size ATC-TCA instances of the Milano Malpensa airport (including various sources of disturbance and traffic predictions of increasing length). Each instance has been tested for the three model variants and for the various solution approaches.
- The computational results show the high potential of the optimization-based approaches compared to the FIFO rule. The optimization procedures are evaluated in terms of computation time indicators, number of optimal solutions, number of constraint violations and aircraft delay minimization.

The paper is organized in five sections. Section 7.2 gives a brief literature review on aircraft scheduling and routing models related to the present work. Section 7.3 presents our problem formulation via alternative graphs. Section 7.4 describes the approaches proposed to solve the problem. Section 7.5 provides a set of model variants, and describes the ATC-TCA instances and the computational results. The experiments are shown for various time horizons of traffic prediction with multiple delayed aircraft and a temporary disruption due to severe weather issues. Section 7.6 concludes the paper and outlines research directions dealing with the traffic control at busy TCAs.

7.2 Review of the related literature

The Air Traffic Flow Management (ATFM) problem has been the subject of several studies on the development of mathematical formulations and heuristic/exact algorithms [Balakrishnan and Chandran (2010), Ball et al., (2007), Barnhart et al (2012), Bennell et al., (2011), Bertsimas et al., (2011a), Bertsimas et al., (2011b), Churchill et a., (2010), Eun et al., (2010), Kuchar and Yang (2000), Pellegrini et al., (2012)]. Recent studies on the ATFM problem are dedicated to the development of several components, e.g. ground-delay programs, airspace flow programs in large networks connecting multiple airports, en-route vectoring and speed adjustments. For these approaches, various kinds of irregular operations are investigated, including lack of critical resources (aircraft and crew members) in order to cover services [Abdelghany et al., (2008), Clausen et al., (2010), Kohl et al., (2007)]. Other research directions focus on a market for arrival and departure slots at airport [Pellegrini et al., (2012), Castelli et al., (2011), Corolli et al., (2014)].

In our view, the mathematical structure of prior works on the ATFM problem is similar to the ATC-TCA problem studied in this paper, even if the scope moves from planning to re-planning in real-time. The underlying problems can be broadly classified into two groups of formulations: *aggregated* or *detailed*.

In aggregated models, the characteristics of the airport infrastructure are simplified and the flight paths are often aggregated. The aircraft scheduling and routing problem of a TCA is often viewed as a runway scheduling problem. With an aggregated model, the runway scheduling problem is typically formulated as a single/parallel machine scheduling problem [Balakrishnan and Chandran (2010), Bertsimas et al., (2011b), Sölveling et al., (2010), Tavakkoli-Moghaddam et al., (2012)]. Most of the early papers on ATFM falls in this category and the choice is motivated by the fact that the runways are often the TCA bottleneck.

In detailed models, individual aircraft are managed on each relevant TCA resource. The aircraft scheduling and routing problem is viewed as a job shop scheduling, in which a job is a sequence of TCA operations and timing constraints related to an aircraft. With a detailed model, additional constraints and problem characteristics can be included [Bennell et al., (2011), Bianco et al., (2006), D'Ariano et al., (2010), D'Ariano et al., (2012b)].

In general, aggregated models are more tractable than detailed models. They can be useful to get insights on the runway selection and the airport flow balancing. At the same time, they are less realistic, since potential conflicts are not detected and solved at the level of individual aircraft and TCA resources.

In this paper, the ATC-TCA problem is modeled as a generalized job shop scheduling problem and is formulated via alternative graphs [Mascis and Pacciarelli (2002)], that are able to enrich the model of [Bianco et al., (2006)] by including additional real-world constraints, such as holding circles, time windows for aircraft travel times, air segments that can host multiple aircraft simultaneously and runways that can host at most one aircraft at a time. A single capacity resource is also named a *blocking* resource, since there must be at most one aircraft at a time processed on this type of resource. This formulation allows accurate modeling of future air traffic flows on the basis of the actual aircraft positions and speeds, and safety constraints. In addition, we use the alternative graph to obtain a new Mixed Integer Linear Programming (MILP) formulation for the ATC-TCA problem, including all scheduling and routing alternatives. The proposed approach is used to study severely disturbed traffic situations, such as multiple delayed landing and/or take-off aircraft and temporarily disrupted runway.

7.3 Mathematical formulations

The ATC-TCA problem can be divided into two sub-problems: (i) the selection of a route for each aircraft and (ii) the scheduling decisions once the routes have been fixed for each aircraft. This section describes this problem decomposition approach via alternative graphs, including a numerical example. Then a mathematical (MILP) formulation is proposed for the overall problem, in which binary variables for route selection are introduced within the scheduling optimization model based on alternative graphs.

7.3.1 Alternative graph formulation of the aircraft scheduling problem

In our formulation of the ATC-TCA problem, an *operation* is the traversing of a resource (air segment, holding circle or runway) by an aircraft. The sequence of operations of an aircraft represents its *route*. Once a route has been assigned to each aircraft (routing problem), the ATC-TCA problem is reduced to the Aircraft Scheduling Problem (ASP), which can be modeled via the alternative graph of [Mascis and Pacciarelli (2002)].

An alternative graph is a triple $G = (N, F, A)$, where $N = \{s, 1, ..., n, t\}$ is a set of n nodes plus the two additional nodes s and t, representing the start and the end operations of the schedule; F is a set of fixed directed arcs, representing precedence constraints between operations; and A is a set of pairs of alternative directed arcs, representing the possible alternative scheduling decisions. In a feasible schedule exactly one alternative arc for each pair has to be selected such that all problem constraints are satisfied.

Each node, except s and t, is associated with the start of an operation kp, where k indicates the aircraft and p the resource it traverses. The start time h_{kp} of operation kp is the entrance time of k in p.

Each fixed arc $(kp, kj) \in F$ has associated the weight $w^F_{kp\_kj}$ that represents a minimum time constraint between h_{kp} and h_{kj}, i.e. $h_{kj} \geq h_{kp} + w^F_{kp\_kj}$, in other words the minimum time required by aircraft k to traverse resource p and arrive at the following resource j.

The alternative pairs $((kp, dj), (ul, vi)) \in A$ model aircraft sequencing and holding decisions, each one with its associated weight $w^A_{kp\_dj}$ and $w^A_{ul\_vi}$, respectively. If alternative arc (kp, dj) is selected in a solution, the additional constraint $h_{dj} \geq h_{kp} + w^A_{kp\_dj}$ is inserted. Otherwise, alternative arc (ul, vi) is selected, and the additional constraint $h_{vi} \geq h_{ul} + w^A_{ul\_vi}$ is added.

A partial selection S is a set of arcs in A obtained by selecting at most one arc from each alternative pair in A. A partial selection is feasible if the graph $(N, F \cup S)$ does not contain positive weight cycles. Otherwise, some of the problem constraints are not satisfied in the schedule. A solution to the ATC-TCA problem is a complete feasible selection, i.e. a selection in which exactly one arc from each alternative pair in A is selected. Given a selection S and any two nodes kp and dj, we let $l^S(kp, dj)$ be the weight of the longest path from kp to dj in the graph $(N, F \cup S)$. By definition, the start time h_{kp} of node $kp \in N$ is the quantity $l^S(s, kp)$, which implies $h_s = 0$ and $h_t = l^S(s, t)$. Note that if S is a complete feasible selection, then h_{kp} is a feasible start time of operation kp. If S is a partial selection, then h_{kp} is a lower bound on the start time of operation kp in any complete feasible selection including S. In fact, adding arcs to the selection cannot decrease the value h_{kp}.

Given the operations s and t, their start times h_s and h_t are used to model the objective function of the ATC-TCA problem as follows. Let h_{kp} be the actual start time of aircraft k in resource p, α_{kp} the scheduled start time and η_{kp} the earliest possible start time, computed by letting the aircraft traveling at

maximum speed and disregarding possible conflicts with other aircraft. We define the *total delay* of k in p as $\max\{0, h_{kp} - \alpha_{kp}\}$. The total delay can be divided into:

- the *unavoidable delay*, which represents the delay that cannot be recovered by rescheduling the aircraft movements and is modeled as $\beta_{kp} = \max\{0, \eta_{kp} - \alpha_{kp}\}$;

- the *consecutive delay*, which represents the delay required to solve potential aircraft conflicts and is modeled as $\gamma_{kp} = \max\{0, h_{kp} - \alpha_{kp} - \beta_{kp}\}$.

We minimize the maximum consecutive delay γ_{kp} by fixing the weight of the fixed arc (kp, t) equal to $w^F_{kp\_t} = -\alpha_{kp} - \beta_{kp}$. The minimization of the maximum consecutive delay can be expressed as $h_t - h_s$ (see equation 7.1). A feasible schedule S is *optimal* if the longest path between s and t in the connected graph (i.e. $l^S(s,t)$) is minimum over all the solutions [D'Ariano et al., (2007a)]. This objective function is used to minimize the aircraft delay propagation, that is an important indicator in the real-time traffic control context. The maximum consecutive delay minimization is inspired on the makespan minimization, and it has the advantage to compute a more compact schedule compared to the minimization of the total consecutive delay.

The alternative graph of the ATC-TCA problem can be formulated as follows. Since fixed and alternative arcs are used to model different types of constraints we partition sets F and A into several subsets: F_{rt} is the set of release time constraints, F_{dt} is the set of due date time constraints, F_{Dt} is the set of deadline time constraints, F_{HC} is the set of holding circle constraints, F_{AS} is the set of air segment constraints, F_{RW} is the set of runway constraints. Similarly, A_{HC}, A_{AS} and A_{RW} are the corresponding sets of alternative arcs.

$$\min \; h_t - h_s \tag{7.1}$$

s.t.

$$h_{kj} - h_s \geq w^{F_{rt}}_{s\_kj} \quad \forall (s, kj) \in F_{rt} \tag{7.2}$$

$$h_t - h_{kj} \geq w^{F_{dt}}_{kj\_t} \quad \forall (kj, t) \in F_{dt} \tag{7.3}$$

$$h_s - h_{kj} \geq w^{F_{Dt}}_{kj\_s} \quad \forall (kj, s) \in F_{Dt} \tag{7.4}$$

$$\begin{array}{l} h_{kj} - h_{kp} \geq w^{F_{HC}}_{kp\_kj} \quad \forall (kp, kj) \in F_{HC} \\ h_{kp} - h_{kj} \geq w^{F_{HC}}_{kj\_kp} \quad \forall (kj, kp) \in F_{HC} \end{array} \tag{7.5}$$

$$h_{kj} - h_{kp} \geq w^{A_{HC}}_{kp\_kj} \quad \vee \quad h_{kp} - h_{kj} \geq w^{A_{HC}}_{kj\_kp} \quad \forall ((kp, kj), (kj, kp)) \in A_{HC} \tag{7.6}$$

$$\begin{aligned} h_{kl} - h_{km} &\geq w^{F_{AS}}_{km\_kl} \quad \forall (km, kl) \in F_{AS} \\ h_{km} - h_{kl} &\geq w^{F_{AS}}_{kl\_km} \quad \forall (kl, km) \in F_{AS} \end{aligned} \tag{7.7}$$

$$\begin{aligned} h_{um} - h_{km} \geq w^{A_{AS}}_{km\_um} \quad \vee \quad h_{kl} - h_{un} \geq w^{A_{AS}}_{un\_kl} \quad \forall ((km, um), (un, kl)) \in A_{AS} \\ h_{km} - h_{um} \geq w^{A_{AS}}_{um\_km} \quad \vee \quad h_{un} - h_{kl} \geq w^{A_{AS}}_{kl\_un} \quad \forall ((um, km), (kl, un)) \in A_{AS} \end{aligned} \tag{7.8}$$

$$h_{ko} - h_{kj} \geq w^{F_{RW}}_{kj\_ko} \quad \forall (kj, ko) \in F_{RW} \tag{7.9}$$

$$h_{uj} - h_{ko} \geq w^{A_{RW}}_{ko\_uj} \quad \vee \quad h_{kj} - h_{ug} \geq w^{A_{RW}}_{ug\_kj} \quad \forall ((ko, uj), (ug, kj)) \in A_{RW} \tag{7.10}$$

Constraints 8.4 model the fixed *release* arcs $(s, kj) \in F_{rt} \subset F$. The weight $w^{F_{rt}}_{s\_kj}$ is the minimum (release) time to start processing operation kj. A release arc (s, kj) represents the earliest possible arrival time of aircraft k in resource j.

Constraints 8.6 model the fixed *due date* arcs $(kj, t) \in F_{dt} \subset F$. The weight $w^{F_{dt}}_{kj\_t} = -\alpha_{kj} - \beta_{kj}$ is the scheduled (due date) time to start processing operation kj. A due date arc (kj, t) represents the scheduled arrival time of aircraft k in resource j.

Constraints 8.5 model the fixed *deadline* arcs $(kj, s) \in F_{Dt} \subset F$. The weight $w^{F_{Dt}}_{kj\_s}$ is the maximum (deadline) time to start processing operation kj. A deadline arc (kj, t) represents the latest possible arrival time of aircraft k in resource j.

Constraints 7.5 and 7.6 are used to formulate the holding circle resources. We recall that holding circles are used by traffic controllers to let arriving aircraft wait before the start of their landing procedure when the TCA is congested. Let kp/kj be the entrance/exit of aircraft k in/from the holding circle. Constraints 7.5 model the fixed arcs (kp, kj) and $(kj, kp) \in F_{HC} \subset F$ of weights $w^{F_{HC}}_{kp\_kj} = 0$ and $w^{F_{HC}}_{kj\_kp} = -\phi$, where ϕ is the time required to perform a circle. Constraints 7.6 model pairs of alternative arcs $((kp, kj), (kj, kp)) \in A_{HC} \subset A$, of weight $w^{A_{HC}}_{kp\_kj} = \phi$ and $w^{A_{HC}}_{kj\_kp} = 0$. Here the selection of an arc models the decision of letting an aircraft enter the holding circle or not. The formulation of multiple circles can be easily done in a similar way.

Constraints 7.7 and 7.8 are used to formulate the air segment resources. The travel time of an aircraft in the air segment must be included in a processing time range $[w_{min}, w_{max}]$. Let km/kl be the entrance/exit of aircraft k in/from the air segment. Constraints 7.7 model two fixed arcs (km, kl) and $(kl, km) \in F_{AS} \subset F$ of weights $w^{F_{AS}}_{km\_kl} = w_{min}$ and $w^{F_{AS}}_{kl\_km} = -w_{max}$.

In each landing/take-off air segment, air traffic regulations impose a minimal longitudinal and diagonal separation distance between consecutive aircraft, that varies according to the aircraft category (e.g. small, medium and large aircraft). Since an overtake between two aircraft k and u in the same air segment is not

allowed for safety reasons, the entrance and exit orders between the aircraft must be the same. Constraints 7.8 model two pairs of alternative arcs that represent the minimum separation time between two aircraft in the air segment m as a sequence-dependent setup time: $((km, um), (un, kl))$ and $((um, km), (kl, un)) \in A_{AS} \subset A$. We observe that l and n are the successive resources for aircraft k and u, respectively. The selection of the alternative pairs models the order in which the aircraft enter/exit the air segment.

Constraints 7.9 and 7.10 are used to formulate the runway resources as a blocking resource. Constraints 7.9 model the fixed arc $(kj, ko) \in F_{RW} \subset F$, of weight $w^{F_{RW}}_{kj\_ko}$ equal to the processing time of runway j by aircraft k. Constraints 7.10 model a pair of alternative arcs $((ko, uj), (ug, kj)) \in A_{RW} \subset A$, of weights $w^{A_{RW}}_{ko\_uj}$ and $w^{A_{RW}}_{ug\_kj}$ equal to the sequence-dependent setup time between aircraft k and u in runway j. The selection of the alternative pair models the order in which the aircraft use the runway. We observe that o and g are the successive resources for aircraft k and u, respectively.

A detailed illustration of the formulation for each specific TCA resource can be found in [D'Ariano et al., (2012b)]. We next provide an illustrative example of the alternative graph formulation for a numerical case study.

7.3.2 A numerical example

Figure 7.1 presents a hypothetical traffic situation with three aircraft, two arriving (A and B) and one departing (C), traversing the TCA of Milan Malpensa (MXP) airport. The TCA resources are: three airborne holding circles (named TOR, MBR and SRN, numbered 1-3); eleven air segments for landing procedures (numbered 4-14); two runways (named RWY 35L, RWY 35R, numbered 16-17), which can be used for aircraft take-off and landing procedure; a common glide path (numbered 15). The latter resource is a special type of landing air segment in which traffic regulations impose a minimum diagonal distance between aircraft, in addition to a minimum longitudinal distance. Overtaking is permitted at triangular locations, while it is forbidden within air segments and runways.

The routes used for the three aircraft are the following. Two routes are considered for aircraft A. Its entrance point in the TCA is TOR (resource 1) and it can either process air segments 4, 10, 13, 15 and land on runway 16 (default route) or process air segments 5, 11, 15 and land on runway 17 (alternative route). Only one route is considered for aircraft B and C. Aircraft B arrives from entry point SRN (resource 3), processes air segments 8, 12, 14, 15 and lands on runway 17, while aircraft C is a departing aircraft and takes-off from runway 16.

Figure 7.2 (a) shows the alternative graph formulation of this example when the default route is chosen for aircraft A. In the graph, each node models the start of an operation, e.g. node $A4$ represents aircraft A on resource 4. The default route for aircraft A is thus expressed by the following chain of nodes $A1$, $A4$,

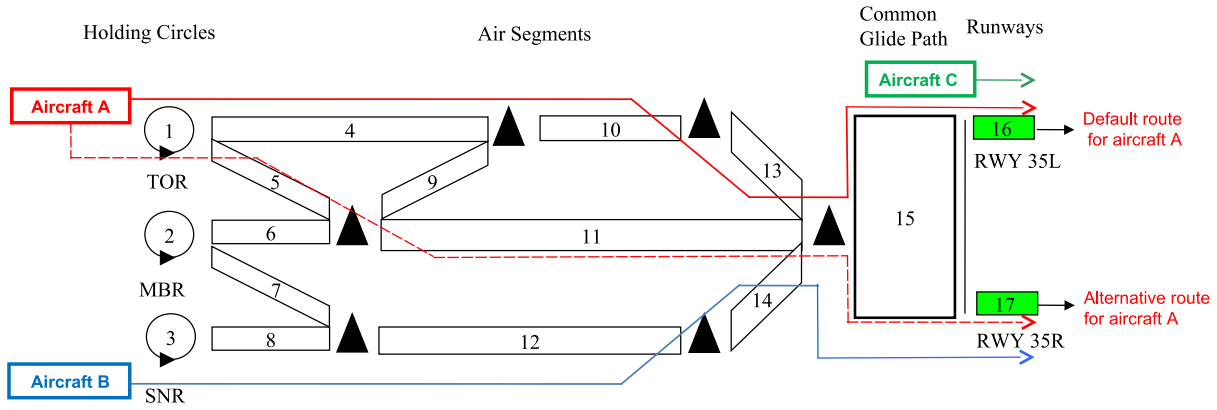


Figure 7.1: A hypothetical traffic situation at the Milan Malpensa (MXP) airport

$A10$, $A13$, $A15$, $A16$, $Aout$, and associated start times. Similarly, for aircraft B (nodes $B3$, $B8$, $B12$, $B14$, $B15$, $B17$, $Bout$) and C (nodes $C16$, $Cout$). Fixed (alternative) arcs are depicted with solid (dotted and colored) arrows. We assume that the start time of traffic prediction is $h_s = 0$.

We now describe the default route of aircraft A in all its components. Applying the same logic, we also describe the operations of aircraft B and C. Each arc weight represents time units. The fixed arc $(s, A1) \in F_{rt}$ is a release arc and its weight $w^{F_{rt}}_{s\_A1} = 0$ is a release time, modeling the expected time at which aircraft A should enter the TCA from the entry point TOR. The fixed arc $(A1, t) \in F_{dt}$ is a due date arc and its weight $w^{F_{dt}}_{A1\_t} = 0$ is a due date time, that allows us to measure the possible delay of aircraft A in entering the TCA.

Aircraft A can (possibly) wait in the holding circle resource before being guided into its landing sequence. The fixed and alternative arcs between nodes $A1$ and $A4$ depict how holding circles are modeled. The weights $w^{F_{HC}}_{A1\_A4} = 0$ and $w^{F_{HC}}_{A4\_A1} = -240$ of the two fixed arcs ($(A1, A4)$ and $(A4, A1) \in F_{HC}$) represent respectively the minimum and maximum time that can be spent by aircraft A in holding circle resource 1. Each alternative pair $((A1, A4), (A4, A1)) \in A_{HC}$ models a specific waiting time in the holding circle (0, 180 or 240). Specifically, we have three alternative feasible situations: selecting arc $(A4, A1)$ of weight $w^{A_{HC}}_{A4\_A1} = 0$ and arc $(A4, A1)$ of weight $w^{A_{HC}}_{A4\_A1} = -180$ means 0 waiting time, selecting arc $(A1, A4)$ of weight $w^{A_{HC}}_{A1\_A4} = 180$ and arc $(A4, A1)$ of weight $w^{A_{HC}}_{A4\_A1} = -180$ means 180 waiting time, selecting arc $(A1, A4)$ of weight $w^{A_{HC}}_{A1\_A4} = 240$ and arc $(A1, A4)$ of weight $w^{A_{HC}}_{A1\_A4} = 180$ means 240 waiting time.

Once out of the holding circle resource, aircraft A processes a number of air segments. The travel time on each air segment is computed according to the speed profile of A and varies between minimum and maximum possible values, since the trajectory of each aircraft flying in the TCA must vary in a pre-defined window.

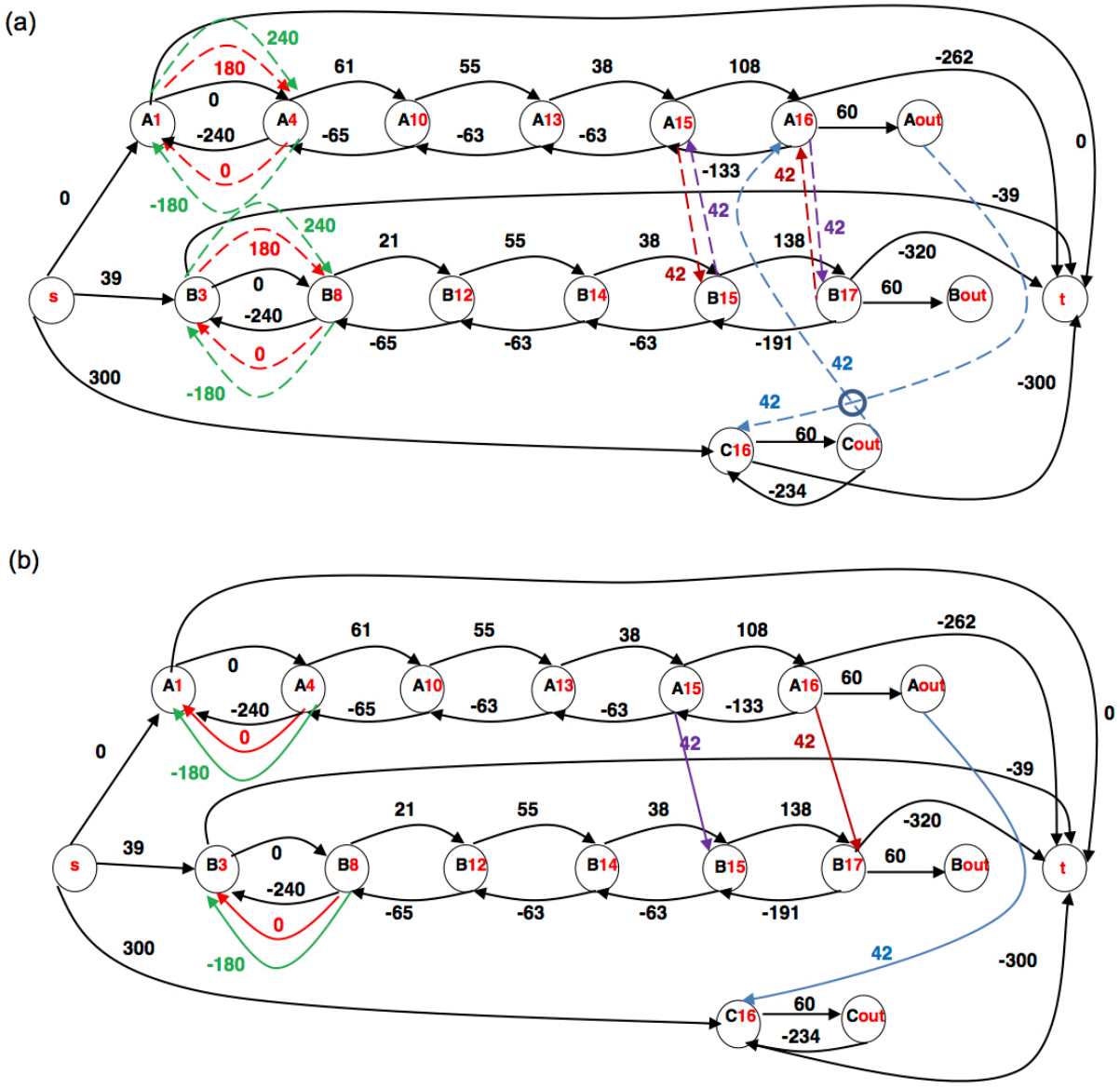


Figure 7.2: Alternative graph for the default routes (a) and a solution (b)

This time window is modeled though a pair of fixed arcs. In particular, the minimum travel time of aircraft A on air segment 4 is given by the weight $w^{F_{AS}}_{A4\_A10} = 61$ of the fixed arc $(A4, A10) \in F_{AS}$, while the maximum travel time is given by the weight $w^{F_{AS}}_{A10\_A4} = -65$ of the fixed arc $(A10, A4) \in F_{AS}$, expressed as a negative value in the formulation. The processing of the other air segments is modeled in a similar way.

The last operation of aircraft A in the TCA is the landing operation on runway 16, depicted by using

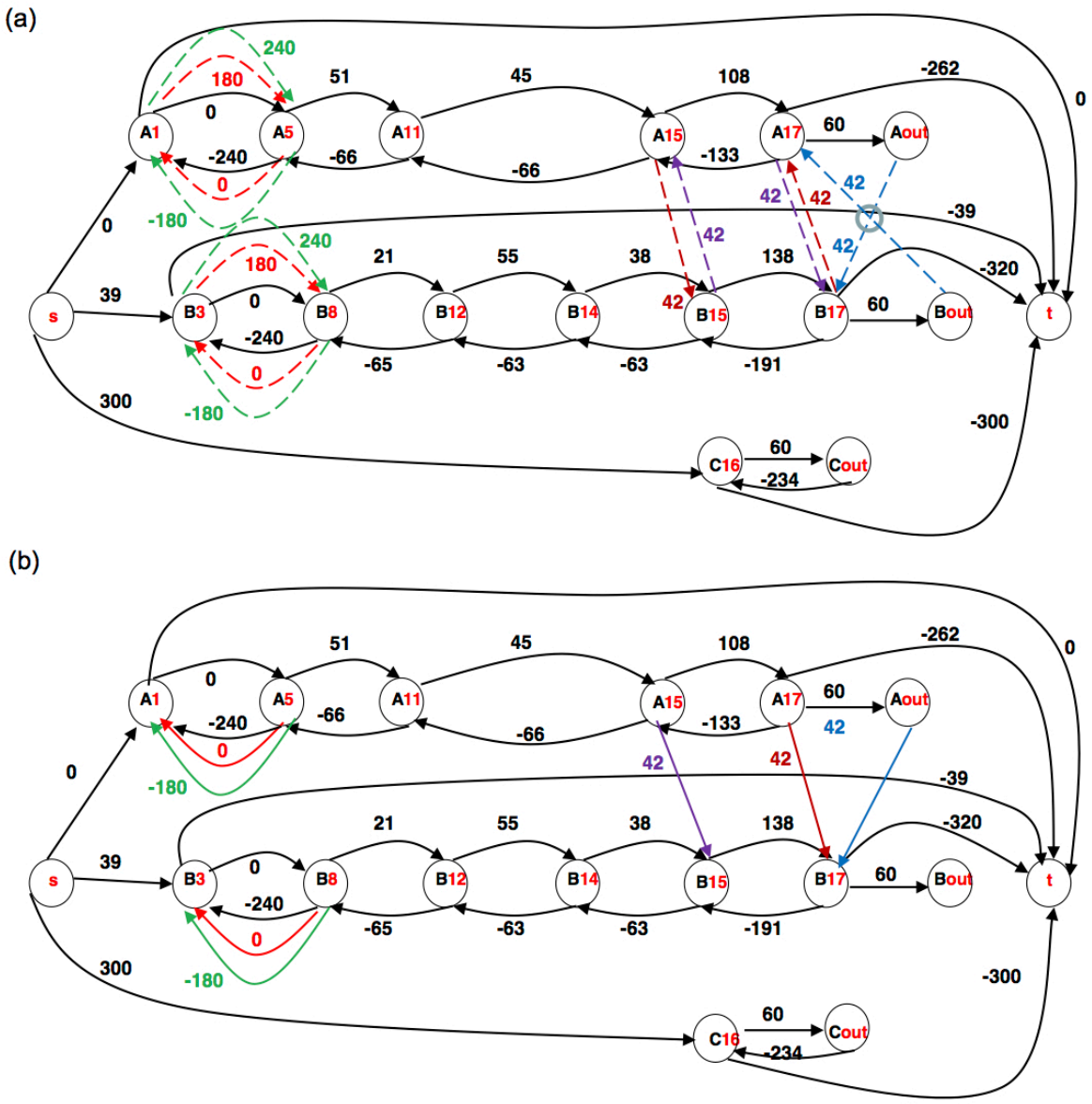


Figure 7.3: Alternative graph with an alternative route for aircraft A (a) and a solution (b)

fixed arc $(A16, Aout) \in F_{RW}$ of weight $w^{F_{RW}}_{A16\_Aout} = 60$. Also, in order to measure the possible consecutive delay of aircraft A at runway 16, a due date arc $(A16, t)$ of weight $w^{Fdt}_{A16\_t} = -262$ is used.

Having more than one aircraft scheduled in each TCA resource, potential conflicts between aircraft routes must be modeled and managed such that minimum separation times are always respected and consecutive delays are minimized. In Figure 7.2, aircraft A and B have a potential conflict on the common glide path

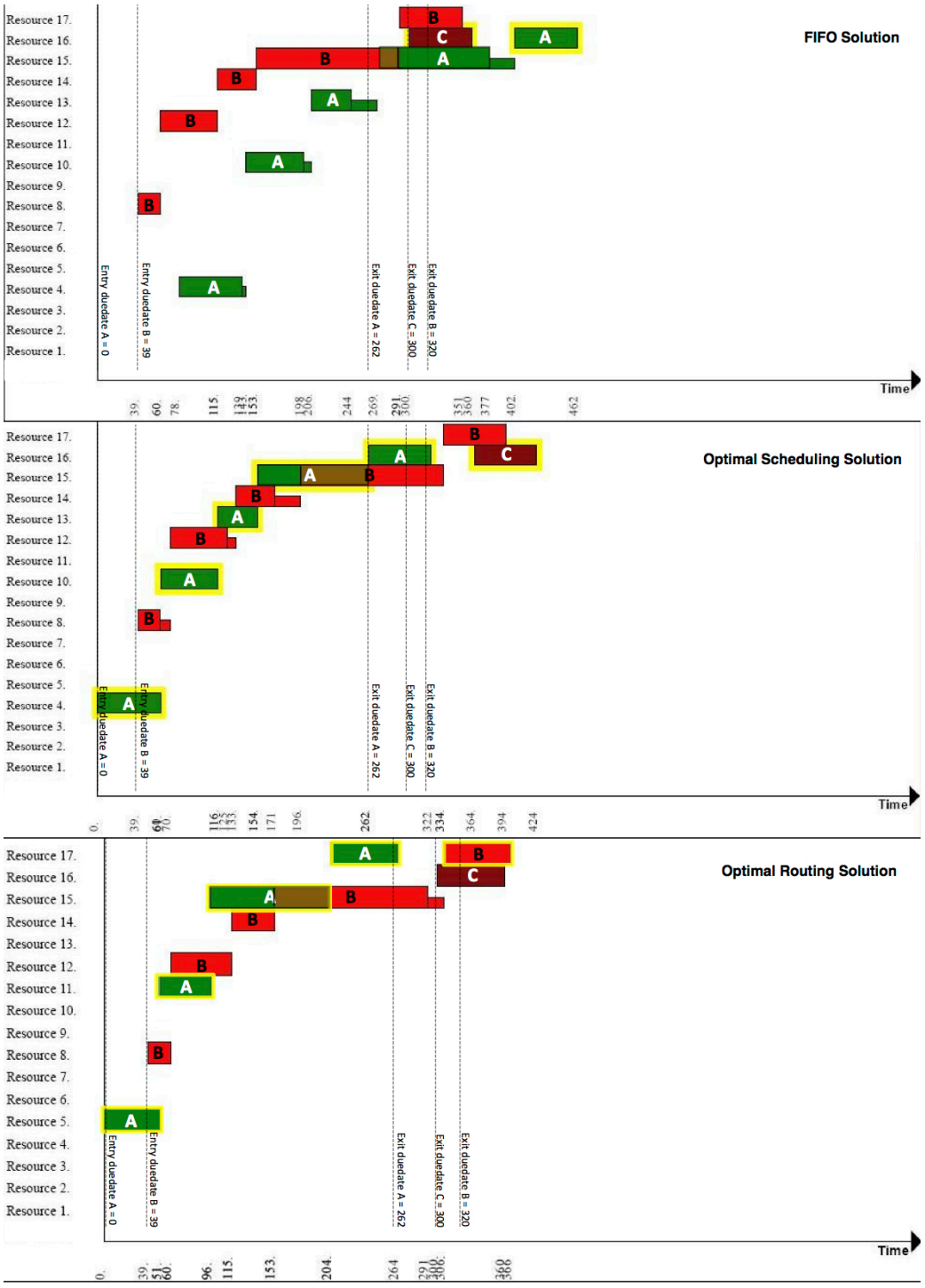


Figure 7.4: Gantt diagrams: FIFO, optimal scheduling and optimal routing solutions

(resource 15), and, since an overtaking is not possible within any air segment (including resource 15), the same sequencing decision must be taken at the entrance and exit of resource 15 between the two aircraft. This is modeled by two pairs of alternative arcs $((A15, B15),\ (B17, A16))$ and $((B15, A15),\ (A16, B17)) \in A_{AS}$, weighted with the minimum separation times required at the entrance $w^{A_{AS}}_{A15\_B15} = w^{A_{AS}}_{B15\_A15} = 42$ and exit $w^{A_{AS}}_{B17\_A16} = w^{A_{AS}}_{A16B17} = 42$ of the air segment. The order in which the two aircraft enter resource 15 is given by which alternative arcs are selected. If alternative arc $(A15, B15)$ is selected in a solution, then aircraft A enters the common glide path before B. Consequently, alternative arc $(A16, B17)$ must be selected, because the same order is required at entrance and exit, otherwise, the selection of alternative arc $(B17, A16)$ would cause a positive weight circle in the graph. Instead, if arcs $(B15, A15)$ and $(B17, A16)$ are selected, B enters resource 15 before A.

Another potential conflict exists in Figure 7.2 between A and C on runway 16. Since the runway is considered as a blocking resource, this conflict is modeled by the alternative pair $((Aout, C16),\ (Cout, A16)) \in A_{RW}$. The weight $w^{A_{RW}}_{Aout\_C16} = w^{A_{RW}}_{Cout\_A16} = 42$ of arcs $(Aout, C16)$ and $(Cout, A16)$ represents the minimum separation time required between the processing of A and C on runway 16. Selecting one arc of the pair models the order in which the aircraft will use the runway, e.g. the selection of alternative arc $(Aout, C16)$ means that A is processed before C on the runway.

Figure 7.2 (b) reports the optimal solution the scheduling problem. Both aircraft A and B have no waiting time in the holding circle resources, A is scheduled before B in the common glide path, A is sequenced before C on runway 16. The maximum consecutive delay is 64, which is the delay of aircraft C at resource 16.

Figure 7.3 (a) shows the alternative graph in which the alternative route for aircraft A is fixed. The new nodes of A are $A1$, $A5$, $A11$, $A15$, $A17$, $Aout$. With the new route for A, there is no more potential conflict between aircraft A and C on runway 16, however there is a new potential conflict between A and B on runway 17, which is modeled by the alternative pair $((Aout, B17), (Bout, A17)) \in A_{RW}$.

Figure 7.3 (b) presents the optimal solution of the ATC-TCA problem (with flexible routes). Neither A nor B have a waiting time in the holding circle resources, A is scheduled before B in the common glide path and on runway 17. The maximum consecutive delay is 0.

Figure 7.4 shows the Gantt diagram of the FIFO, the optimal scheduling, the optimal routing and scheduling solutions. Each diagram reports the following information: a dotted vertical line for each due date arc, a thick rectangle for the minimum processing time of each aircraft in each resource, a thin rectangle for the additional processing time of some aircraft in some resources (due to the resolution of potential conflicts). The critical path is highlighted in yellow color. The overlapping rectangles model the situation in which multiple aircraft are assigned to the same air segment resources.

In the FIFO solution both aircraft A and B have no waiting time in the holding circle resources, B

is scheduled before A in the common glide path (resource 15), and C is sequenced before A on runway 16. The maximum consecutive delay is 140, which is the delay of aircraft A at the runway (resource 16). Comparing the FIFO solution with those produced with the other approaches, the minimization of the maximum consecutive delay helps in finding more compact solutions, and thus to limit the propagation of delays to forthcoming aircraft.

7.3.3 Formulation of the aircraft scheduling and routing problem

We present a MILP formulation of the ATC-TCA problem that considers the scheduling and routing decisions simultaneously. This can be obtained from the alternative graph formulation, enlarging sets F and A in order to contain the fixed and alternative arcs related to all possible aircraft routes. Each fixed arc translates into a constraint, while each alternative pair into a pair of alternative constraints, introducing a binary variable representing the choice made. The start time of each operation is a non-negative real variable. In addition, a variable is adopted for each alternative route. The resulting formulation is:

$$\min \; h_t - h_s \tag{7.11}$$

s.t.

$$h_{krj} - h_s + M(1 - y_{kr}) \geq w^{F_{rt}}_{s\_krj} \quad \forall (s, krj) \in F_{rt} \tag{7.12}$$

$$h_t - h_{krj} + M(1 - y_{kr}) \geq w^{F_{dt}}_{krj\_t} \quad \forall (krj, t) \in F_{dt} \tag{7.13}$$

$$h_s - h_{krj} + M(1 - y_{kr}) \geq w^{F_{Dt}}_{krj\_s} \quad \forall (krj, s) \in F_{Dt} \tag{7.14}$$

$$\begin{aligned} h_{krj} - h_{krp} + M(1 - y_{kr}) &\geq w^{F_{HC}}_{krp\_krj} \quad \forall (krp, krj) \in F_{HC} \\ h_{krp} - h_{krj} + M(1 - y_{kr}) &\geq w^{F_{HC}}_{krj\_krp} \quad \forall (krj, krp) \in F_{HC} \end{aligned} \tag{7.15}$$

$$\begin{aligned} &h_{krj} - h_{krp} + Mx^{krj\_krp}_{krp\_krj} + M(1 - y_{kr}) \geq w^{A_{HC}}_{krp\_krj} \\ &h_{krp} - h_{krj} + M(1 - x^{krj\_krp}_{krp\_krj}) + M(1 - y_{kr}) \geq w^{A_{HC}}_{krj\_krp} \quad \forall ((krp, krj), (krj, krp)) \in A_{HC} \end{aligned} \tag{7.16}$$

$$\begin{aligned} h_{krl} - h_{krm} + M(1 - y_{kr}) &\geq w^{F_{AS}}_{krm\_krl} \quad \forall (krm, krl) \in F_{AS} \\ h_{krm} - h_{krl} + M(1 - y_{kr}) &\geq w^{F_{AS}}_{krl\_krm} \quad \forall (krl, krm) \in F_{AS} \end{aligned} \tag{7.17}$$

$$\begin{aligned} &h_{uim} - h_{krm} + Mx^{uin\_krl}_{krm\_uim} + M(2 - y_{kr} - y_{ui}) \geq w^{A_{AS}}_{krm\_uim} \\ &h_{krl} - h_{uin} + M(1 - x^{uin\_krl}_{krm\_uim}) + M(2 - y_{kr} - y_{ui}) \geq w^{A_{AS}}_{uin\_krl} \quad \forall ((krm, uim), (uin, krl)) \in A_{AS} \\ &h_{krm} - h_{uim} + Mx^{krl\_uin}_{uim\_krm} + M(2 - y_{kr} - y_{ui}) \geq w^{A_{AS}}_{uim\_krm} \\ &h_{uin} - h_{krl} + M(1 - x^{krl\_uin}_{uim\_krm}) + M(2 - y_{kr} - y_{ui}) \geq w^{A_{AS}}_{krl\_uin} \quad \forall ((uim, krm), (krl, uin)) \in A_{AS} \end{aligned} \tag{7.18}$$

$$h_{kro} - h_{krj} + M(1 - y_{kr}) \geq w_{krj\_kro}^{F_{RW}} \quad \forall (krj, kro) \in F_{RW} \tag{7.19}$$

$$\begin{aligned} &h_{uij} - h_{kro} + M x_{kro\_uij}^{uig\_krj} + M(2 - y_{kr} - y_{ui}) \geq w_{kro\_uij}^{A_{RW}} \\ &h_{krj} - h_{uig} + M(1 - x_{kro\_uij}^{uig\_krj}) + M(2 - y_{kr} - y_{ui}) \geq w_{uig\_krj}^{A_{RW}} \quad \forall ((kro, uij), (uig, krj)) \in A_{RW} \end{aligned} \tag{7.20}$$

$$\sum_{r=1}^{R_k} y_{kr} = 1 \quad k = 1, ..., Z \tag{7.21}$$

$$x_{krp\_krj}^{krj\_krp} \in \{0, 1\} \quad \forall ((krp, krj), (krj, krp)) \in A_{HC} \tag{7.22}$$

$$\begin{aligned} x_{krm\_uim}^{uin\_krl} &\in \{0, 1\} \quad \forall ((krm, uim), (uin, krl)) \in A_{AS} \\ x_{uim\_krm}^{krl\_uin} &\in \{0, 1\} \quad \forall ((uim, krm), (krl, uin)) \in A_{AS} \end{aligned} \tag{7.23}$$

$$x_{kro\_uij}^{uig\_krj} \in \{0, 1\} \quad \forall ((kro, uij), (uig, krj)) \in A_{RW} \tag{7.24}$$

$$y_{kr} \in \{0, 1\} \quad k = 1, ..., Z \ ; \ r = 1, ..., R_k \tag{7.25}$$

Operations are represented by the triple aircraft, route and resource, e.g. operation krp models the processing of aircraft k on resource p when using route r. The binary variable $x_{krp\_krj}^{krj\_krp}$ models the scheduling decision on the alternative pair $((krp, krj), (krj, krp))$; the binary variable y_{kr} models the possible selection of route r by aircraft k; the non-negative real variable h_{krp} models the start time of operation kp for route r. Furthermore, Z is the number of aircraft, and R_k the number of routes for aircraft k. M is a very large constant, e.g. equal to the sum of all arc weights. The objective function in 7.11 is the same as in 7.1.

This big-M formulation can be seen as an extended version of the formulation with fixed route of Section 7.3.1, since scheduling and routing decisions are taken simultaneously. In particular, the fixed constraints 7.12, 7.13, 7.14, 7.15, 7.17, 7.19 correspond to the fixed constraints 8.4, 8.6, 8.5, 7.5, 7.7, 7.9 when aircraft k uses route r.

Constraints 7.16 are the MILP equivalents of the alternative constraints 7.6. In case of a single holding pattern, binary variable $x_{krp\_krj}^{krj\_krp}$ models whether the holding circle procedure (modeled by operations krp and krj) is performed ($x_{krp\_krj}^{krj\_krp} = 0$) or not ($x_{krp\_krj}^{krj\_krp} = 1$) by aircraft k when using route r.

Constraints 7.18 are the MILP equivalents of the alternative constraints 7.8. Binary variables $x_{krm\_uim}^{uin\_krl}$ and $x_{uim\_krm}^{krl\_uin}$ model the sequencing decisions in the air segment m between aircraft k with route r and aircraft u with route i. There are two feasible sequencing solutions. If $x_{krm\_uim}^{uin\_krl} = 0$ and $x_{uim\_krm}^{krl\_uin} = 1$ aircraft k enters and exits air segment m before aircraft u. Alternatively, if $x_{krm\_uim}^{uin\_krl} = 1$ and $x_{uim\_krm}^{krl\_uin} = 0$ aircraft u enters and exits air segment m before aircraft k.

Constraints 7.20 are the MILP equivalents of the alternative constraints 7.10. Binary variable $x_{kro\_uij}^{uig\_krj}$

models the sequencing decision in the runway j between aircraft k when using route r and aircraft u when using route i. If $x^{uig\_krj}_{kro\_uij} = 0$ aircraft u follows k, otherwise if $x^{uig\_krj}_{kro\_uij} = 1$ aircraft k follows u.

Constraints 7.21 model the fact that an individual route has to be fixed for each aircraft, e.g. if $y_{kr} = 1$ aircraft k uses route r among its set of R_k routes.

The MILP formulation of the example of Figure 7.1 is shown in the paper appendix.

7.4 Solution methods

This section describes the approaches proposed to solve the aircraft scheduling and routing problem at a busy TCA. We use a commercial solver to solve the MILP formulation, and propose two decomposition frameworks. The problem decompositions are motivated by the fact that the studied aircraft scheduling and routing problem is very difficult to solve in real-time, specially for large time horizons and disrupted traffic situations.

A temporal decomposition, named *rolling horizon* framework, is proposed to divide the problem into time horizons of traffic predictions of a reasonable size. This approach is compared with the centralized framework, that solves the overall problem in one step. In this work, we assume that information is fully known for all approaches at time 0 of the traffic prediction (before the solvers begin solving the problem). In this way, the rolling horizon approach is consistent with the centralized approach.

The second framework decompose the problem variables into two sets: (i) routing and (ii) scheduling variables. The solution algorithm iteratively solves the scheduling problem with fixed routes and then changes the route of some aircraft given the scheduling solution. The two sub-problems are solved with the solver AGLIBRARY, developed at Roma Tre University.

Clearly, the optimal solution of the problem can only be certified by solving the whole MILP formulation, while the decomposition approaches allow the computation of good quality solutions in a short computation time.

In general, whenever a solver does not provide a feasible schedule, the infeasibility is reported to the human traffic controllers. They (in cooperation with the airlines) are asked to take some actions that the automated system is not allowed to take, such as re-routing some aircraft to another airport or changing the maximum time that aircraft can spend in the holding circles.

7.4.1 Scheduling and re-routing decomposition

Figure 7.5 illustrates AGLIBRARY's architecture in case of flexible routes. The basic idea is to first compute an aircraft scheduling solution, given a route for each aircraft, and then search for better aircraft routing

solutions. The aircraft re-routing module verifies if new routes, leading to a potentially better solution, exist. Whenever re-routing is performed, the aircraft scheduling module computes a new schedule of aircraft movements using the new chosen routes. The iterative procedure returns the best aircraft times, orders and routes after a stopping criteria is reached, i.e. a given time limit of computation is reached or there are no more aircraft routes available.

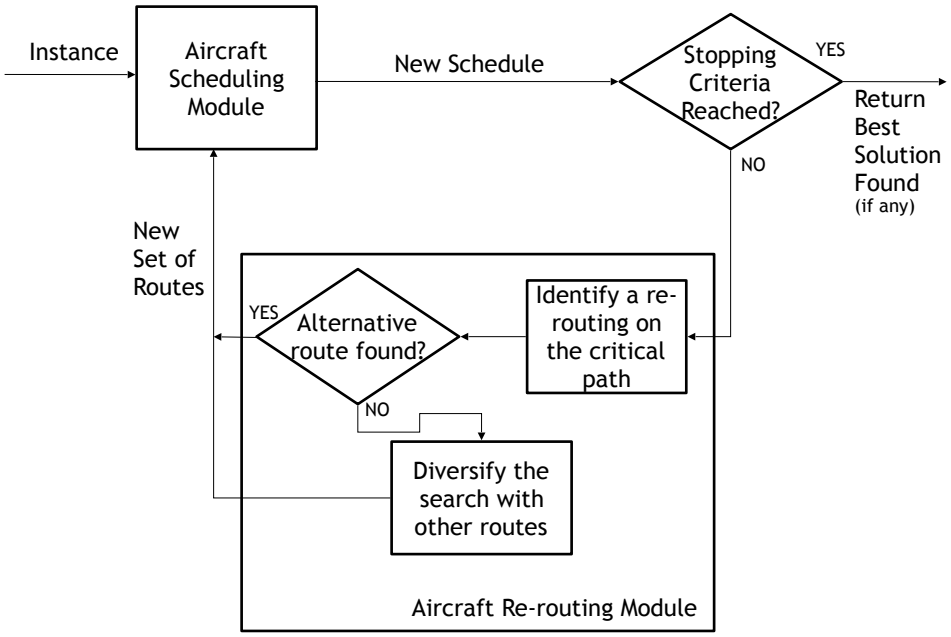


Figure 7.5: AGLIBRARY solver based on iterative aircraft scheduling and re-routing

At the first run of the iterative procedure, an aircraft scheduling module computes aircraft orders and times given a default (off-line) route for each aircraft. The aircraft scheduling problem is first solved by heuristic algorithms, then a branch and bound algorithm, truncated after a given time limit of computation, is executed to possibly improve the heuristic solutions. As rule-based method, First-In-First-Out (FIFO) is used, taking aircraft ordering decisions one at a time by assigning each conflicting resource to the first aircraft requiring it. According to Bennell et al. [Bennell et al., (2011)], this is a commonly used rule, even if human controllers may adjust the FIFO sequence in order to recover possible infeasible decisions. As optimization method, the truncated branch and bound (BB) algorithm of D'Ariano et al. [D'Ariano et al., (2010)] is adopted to solve the ASP to near-optimality. In particular, a binary branching scheme is implemented in order to take one scheduling decision at a time. This is implemented by selecting an alternative pair $((kp, dj), (ul, vi)) \in A$ and branching on the arcs (kp, dj) and (ul, vi) of the pair. We branch

with priority on the alternative pairs associated to the runway resources, i.e. on the set A_{RW}. This strategy is motivated by the fact that the runways are often the bottleneck of the TCA, as reported by many studies discussed in the literature review.

The algorithm used in this paper for the iterative scheduling and re-routing approach is the Tabu Search (TS) of D'Ariano et al. [D'Ariano et al., (2012a), D'Ariano et al., (2012b)]. In particular, the neighbourhood strategy used by TS explores several neighbors based on the concept of ramified critical path, that is an extension of the classical critical path method. The key idea is that changing the route of a single aircraft may avoid traversing a resource on the critical path, which in turn may result in a new solution with smaller value of the objective function. At each iteration, the best neighbor is chosen as the new reference routing solution. When no potentially better solution is found on the critical path neighbourhood, the search alternates this neighbourhood strategy with a diversification strategy, which consists of changing at random the route of a set of aircraft at the same time. The parameters of the tabu search algorithm have been tuned in [D'Ariano et al., (2012a)]. In the remaining of this paper we indicate with BB+TS the tabu search algorithm for the re-routing module combined with the BB algorithm for the scheduling module.

Deadline implications

This subsection describes some useful techniques to speed up our solution algorithms, developed in the areas of scheduling theory and constraint programming. The first concept we use is named *implication* [D'Ariano et al., (2007a)]. The idea is to prove that, given a partial selection S and an unselected pair of alternative arcs $((kp, dj), (ul, vi)) \in A$, no feasible schedule exist if arc (kp, dj) is selected. In such case, we say that arc (ul, vi) is *implied* by S and can be immediately added to S without the need for branching.

Implication rules are a key property to solve the scheduling problem to optimality [D'Ariano et al., (2007a), D'Ariano et al., (2010)]. Besides the implications used in [D'Ariano et al., (2010)], in this work we add a further implication rule particularly useful in the presence of deadline constraints, that we call *deadline implication*.

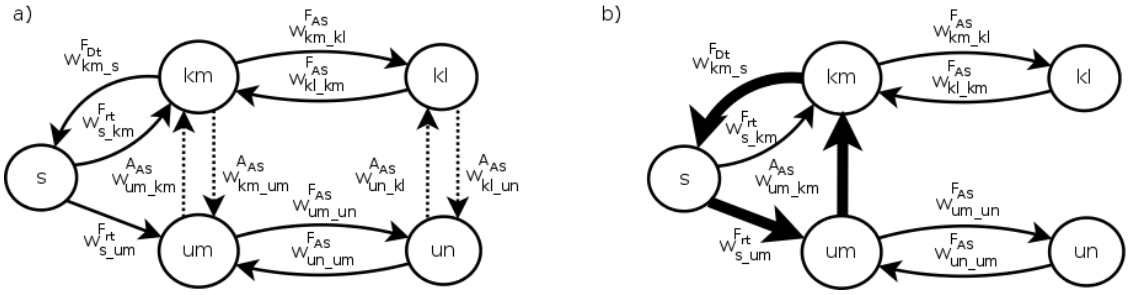


Figure 7.6: Example of alternative graph with a deadline arc

Proposition 7.4.1. *Consider a graph (N, F, A) with F containing deadline arcs and a selection S. Let (mn, s) be a deadline arc and $((kp, dj), (ul, vi))$ an unselected alternative pair. If $-w^{F_{Dt}}_{mn\_s} < h^S_{kp} + w^A_{kp\_dj} + l^S(dj, mn)$ then (ul, vi) is implied by S, since arc (kp, dj) is forbidden in any feasible schedule including S.*

Proof. The result follows from the observation that the quantities h^S_{kp} and $l^S(dj, mn)$ equal the weights of two (longest) paths from s to kp and from dj to mn respectively. These paths form a cycle with arcs (kp, dj) and (mn, s). From the hypothesis it follows that the weight of this cycle is $h^S_{kp} + w^A_{kp\_dj} + l^S(dj, mn) + w^{F_{Dt}}_{mn\_s} > 0$ and therefore the selection $S \cup (kp, dj)$ is infeasible. □

Figure 7.6 (a) shows an example of a small alternative graph with deadline arc (km, s). In the example, given the empty selection $S = \emptyset$ the start time h_{um} of node um is equal to the weight of the fixed arc (s, um), i.e. $h_{um} = w^{F_{rt}}_{s\_um}$. Hence, if $w^{F_{rt}}_{s\_um} + w^{A_{AS}}_{um\_km} > -w^{F_{Dt}}_{km\_s}$ the cycle in Figure 7.6 (b) has positive weight, the infeasibility being caused by the violation of the deadline constraint. If this is the case, in any feasible schedule the alternative arc (um, km) is forbidden and its alternative arc (kl, un) is implied.

We observe that proposition 7.4.1 specifies which alternative arc is forbidden in a feasible schedule, with the consequence that the other arc in the pair is implied by S, e.g. arc (kl, un) in the example in Figure 7.6. Therefore, if both arcs in an alternative pair are forbidden, no feasible schedule exists given the selection S.

The following proposition is derived from constraint programming techniques applied to the job shop scheduling problem [Sadeh et al., (1995), Sadeh et al., (1996)]. The idea is to define for each node $kp \in N$ an interval $[h_{kp}, h^{max}_{kp}]$ of [minimum, maximum] start time for any complete feasible selection $\hat{S} \supseteq S$. By definition, the interval associated to the start node s is $[0, 0]$ and $h^{max}_t = +\infty$. For any other node $kp \in N$ the value h^{max}_{kp} can be recursively defined as follows:

$$h^{max}_{kp} = min_{(kp,xy)\in F\cup S}\{h^{max}_{xy} - w_{kp\_xy}\}. \tag{7.26}$$

Proposition 7.4.2. *Consider an alternative graph (N, F, A) with F containing deadline arcs and a selection S. Let $h_{kp} = l^S(s, kp)$, and let h^{max}_{kp} be defined according to (7.26). In any complete feasible selection $\hat{S} \supseteq S$, and for each node $kp \in N$, it holds $h_{kp} \leq l^{\hat{S}}(s, kp) \leq h^{max}_{kp}$. Moreover, given an unselected alternative pair $((kp, dj), (ul, vi))$, if $h^{max}_{dj} < h_{kp} + w^A_{kp\_dj}$ then arc (kp, dj) is forbidden and (ul, vi) is implied by S.*

Proof. The proof follows from Proposition 7.4.1. In fact, if $h^{max}_{kp} < \infty$ holds for some node $kp \in N \setminus \{s\}$ it follows that there is a path from kp to s of weight $-h^{max}_{kp}$. Clearly, adding arcs in $\hat{S} \setminus S$ to the graph $(N, F \cup S)$ cannot remove the previous path. Assume by contradiction that a complete feasible selection $\hat{S} \supseteq S$ exists such that $l^{\hat{S}}(s, kp) > h^{max}_{kp}$. This means that in the graph $(N, F \cup \hat{S})$ there is a path from s to kp of weight $l^{\hat{S}}(s, kp)$ and a path from kp to s of weight $-h^{max}_{kp}$, i.e. there is a positive weight cycle in the

graph, a contradiction. Similarly, there is a positive weight cycle if arc (kp, dj) is selected, and this concludes the proof. □

Pre-processing procedure

Proposition 7.4.2 is used at the root node of our branch and bound algorithm or before executing a heuristic scheduling algorithm to reduce the cardinality of set A and to early detect the infeasibility on an ATC-TCA instance when the deadline constraints are incompatible with each other.

The pre-processing procedure starts with the computation of h_{kp} and h_{kp}^{max} for each node $kp \in N$. Then, we consider all the alternative pairs $((kp, dj), (ul, vi)) \in A$ such that $h_{dj}^{max} < +\infty$ or $h_{vi}^{max} < +\infty$, and check whether one of the following three cases occurs:

- $h_{dj}^{max} < h_{kp} + w_{kp\_dj}^{A}$ and $h_{vi}^{max} < h_{ul} + w_{ul\_vi}^{A}$,
- $h_{dj}^{max} < h_{kp} + w_{kp\_dj}^{A}$ and $h_{vi}^{max} \geq h_{ul} + w_{ul\_vi}^{A}$,
- $h_{dj}^{max} \geq h_{kp} + w_{kp\_dj}^{A}$ and $h_{vi}^{max} < h_{ul} + w_{ul\_vi}^{A}$.

In the first case the instance infeasibility is detected. In the second [third] case (ul, vi) is implied [(kp, dj) is implied] and the values h_{kp} and h_{kp}^{max} are updated for each node $kp \in N$. The procedure is repeated until no new arc is implied or until an infeasibility is detected. If the pre-processing procedure ends with a feasible partial selection, a scheduling algorithm is called in order to select the remaining alternative arcs.

Figure 7.7 presents an example of this procedure. Aircraft A and B are two landing aircraft, each with a deadline arc on the first operation, bounding its entrance in the TCA. Each node $kp \in N$ is labeled with the values $(h_{kp}|h_{kp}^{max})$. Figure 7.7 (a) refers to the first phase of the procedure, when h_{kp} and h_{kp}^{max} are computed and S is an empty selection. Figure 7.7 (b) shows the changes occurred during the second phase. The analysis of the alternative pair $((B15, A15), (A16, B17))$ leads to the implication of arc $(A16, B17)$, since arc $(B15, A15)$ would cause a positive weight cycle in the graph. Similarly, the analysis of the alternative pair $((A15, B15), (B17, A16))$ leads to the implication of arc $(A15, B15)$, alternative of arc $(B17, A16)$ causing another positive weight cycle. In both cases, the update of h_{kp} and h_{kp}^{max} for each node $kp \in N$ does not change any value and the procedure terminates with the partial selection $\{(A16, B17), (A15, B15)\}$.

7.4.2 Temporal decomposition

The aircraft scheduling and routing problem can be solved via the following temporal decomposition, in which the overall time horizon of traffic prediction is divided into consecutive traffic forecast intervals, named look-ahead periods. This decomposition is implemented via the rolling horizon framework introduced by Samà

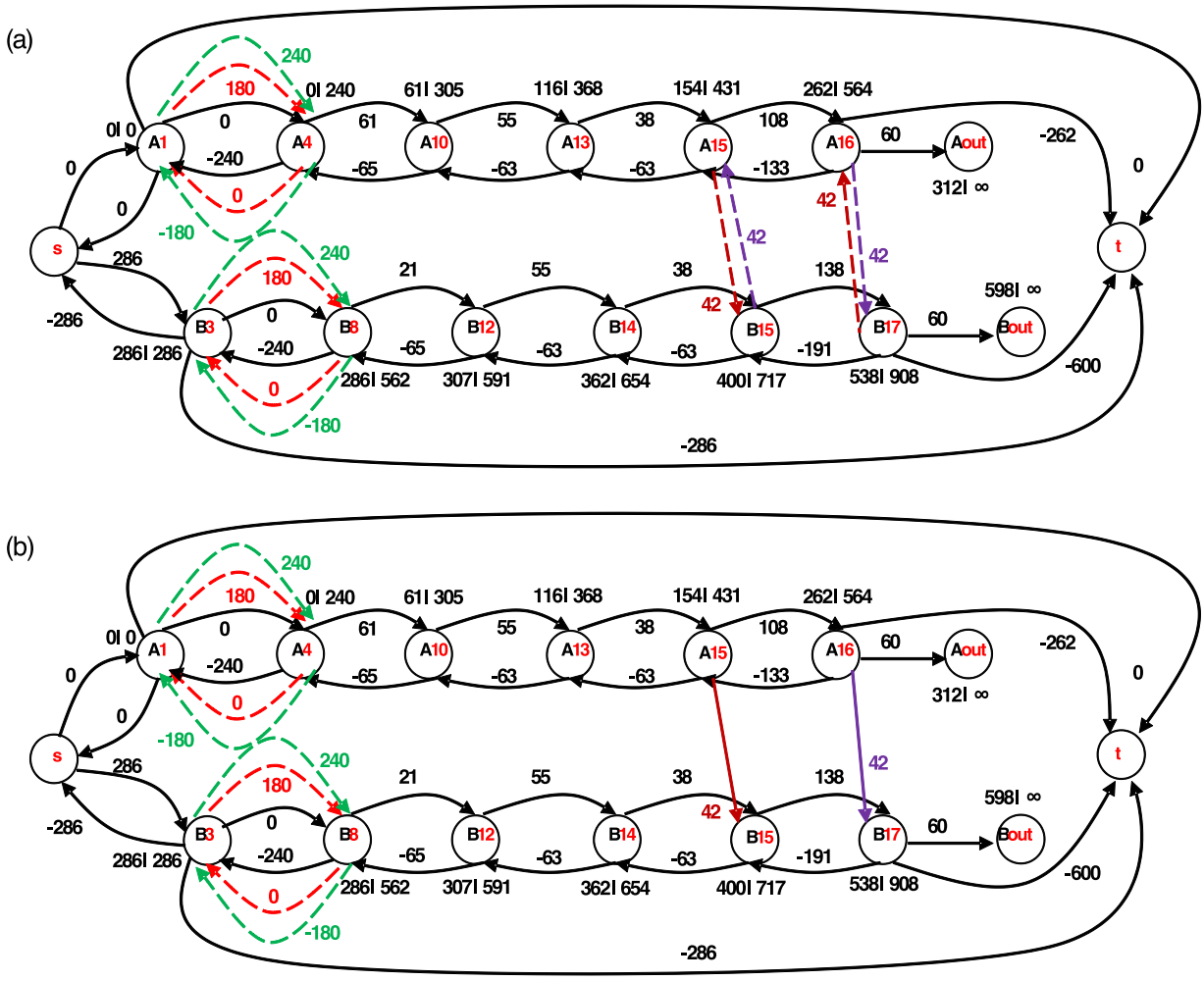


Figure 7.7: Example of deadline implications during the pre-processing procedure

et al. [Samà et al., (2013a)] for the aircraft scheduling problem formulated via alternative graphs. We next describe an extended version of this framework that deals with aircraft scheduling and routing variables.

For a given time horizon of prediction, we use a set of look-ahead periods (T_i with $i = 1, ..., u$). The start time of each look-ahead period is set via a fixed roll period, i.e. a non-overlapping period between the previous and the next look-ahead periods. The roll periods themselves are non-overlapping with each other.

Given the predicted information on the current operational conditions, the rolling horizon mechanism computes a scheduling and routing plan periodically, one for every roll period. This iterative mechanism complicates the optimization aspect compared to a single solver execution, due to the partial overlap of consecutive look-ahead periods and to the operational constraints related to this transition.

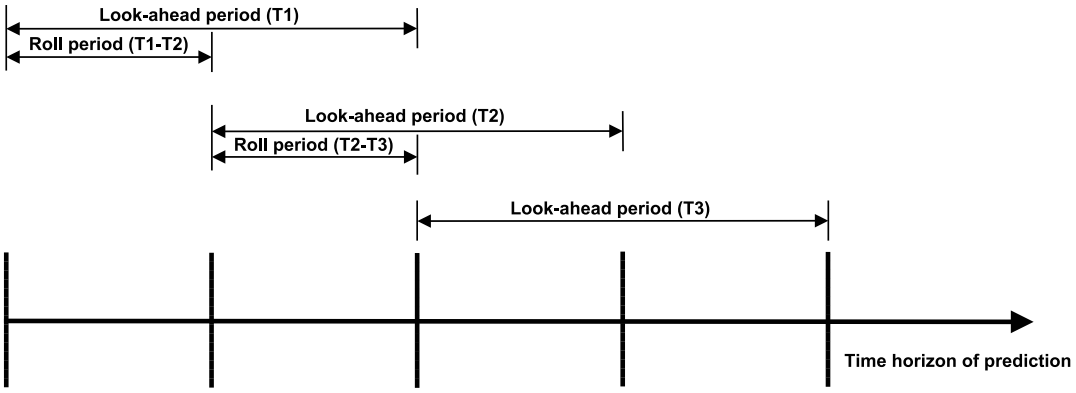


Figure 7.8: Example of rolling horizon approach with two roll periods and three look-ahead periods

The rolling horizon framework performs u iterations. At iteration i, the current look-ahead period T_i is solved. At the next iteration $i+1 \leq u$, the new look-ahead period T_{i+1} is solved. The latter look-ahead period starts after the roll period T_i-T_{i+1}. Figure 7.8 gives the structure of the rolling horizon framework in case of three look-ahead periods (T1,T2 and T3) and two roll periods (T1-T2 and T2-T3).

Figure 7.9 presents the flowchart of the proposed rolling horizon framework. The lengths of the time horizon of traffic prediction, the look-ahead periods and the roll periods are input data. At each iteration i of the procedure, we generate the current look-ahead period T_i, taking into account the following two aircraft sets: the aircraft entering the TCA during the look-ahead period T_i plus the aircraft that have not yet completed their operations in the look-ahead period T_{i-1}. For the latter set of aircraft, we compute the set of feasible routes as follows. An aircraft route is feasible for the look-ahead period T_i if the route includes the last operation performed by the aircraft in the look-ahead period T_{i-1}. Then, the look-ahead period T_i is solved by a specific solver. If a feasible schedule is found and the end of the traffic prediction is not reached, the next look-ahead and roll periods are set. The rolling horizon procedure lasts till the resolution of the overall time horizon, or till an infeasibility is found at an intermediate iteration.

The centralized framework can be seen as a special case of the rolling horizon procedure, where the length of the look-ahead period is the same as that of the overall time horizon of prediction. In this case, a single look-ahead period is thus generated and the procedure performs a single iteration.

7.5 Experiments

This section presents the experimental assessment performed by using the different formulations, frameworks and algorithms of Sections 7.3 and 7.4. The test bed is the Milan Malpensa terminal control area (MXP).

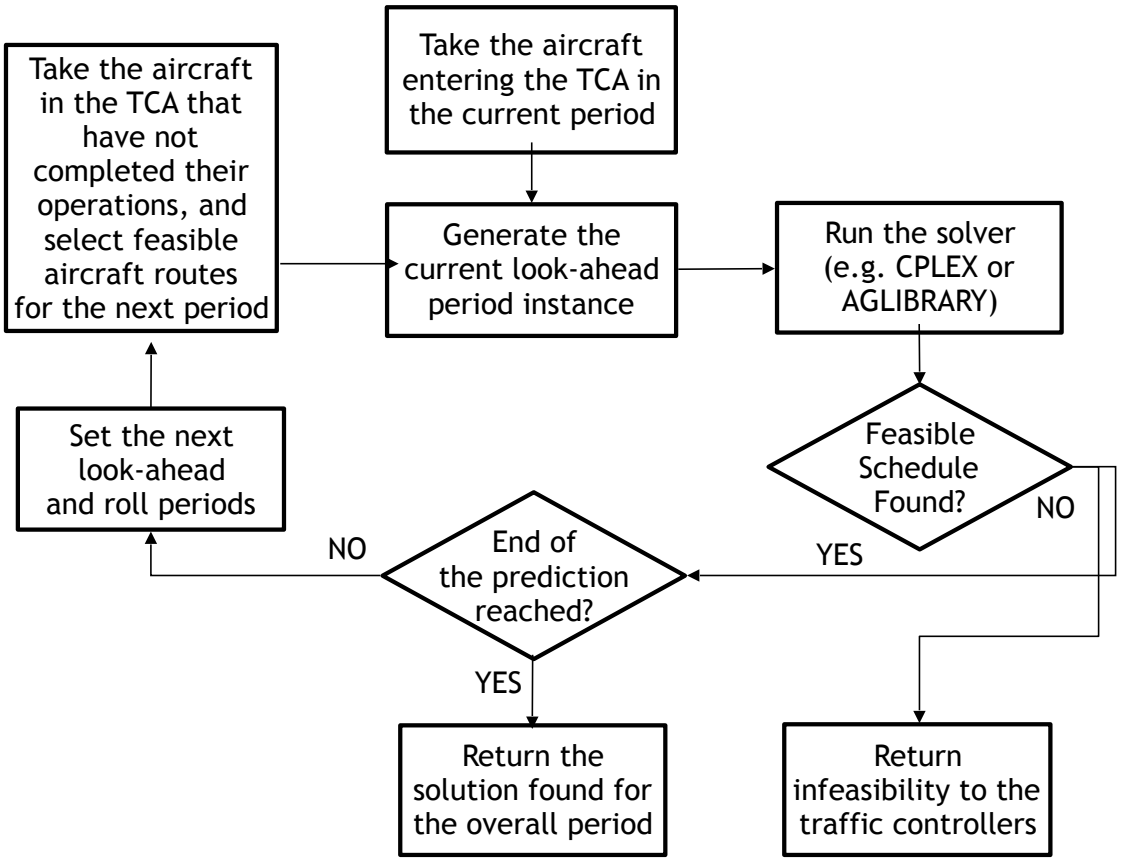


Figure 7.9: A flowchart for the rolling horizon framework

7.5.1 Model variants

This subsection presents the three model variants, M1, M2 and M3, studied in this paper. We use them to study different objective functions and user requirements. Figure 7.10 illustrates the difference between the models for a landing aircraft. In all cases, Model 1 (M1) is adopted for a departing aircraft. We next describe the general characteristics of each variant for landing and take-off aircraft.

Model 1 (M1) measures aircraft delays only at the runways, that are frequently the bottleneck of the whole TCA. This is achieved by inserting a due date arc (kj, t) measuring the delay of each landing/take-off aircraft k at the entrance of scheduled runway j. The weight $w_{kj\_t}^{F_{dt}}$ is set as described in Section 7.3.

Model 2 (M2) measures the delays of arriving aircraft both at the runways and at the entrance of the TCA, while the delay of departing aircraft is measured at the runways only. This is achieved by inserting a due date arc (ki, t) measuring the delay of each landing aircraft k at the entrance of air segment i. The

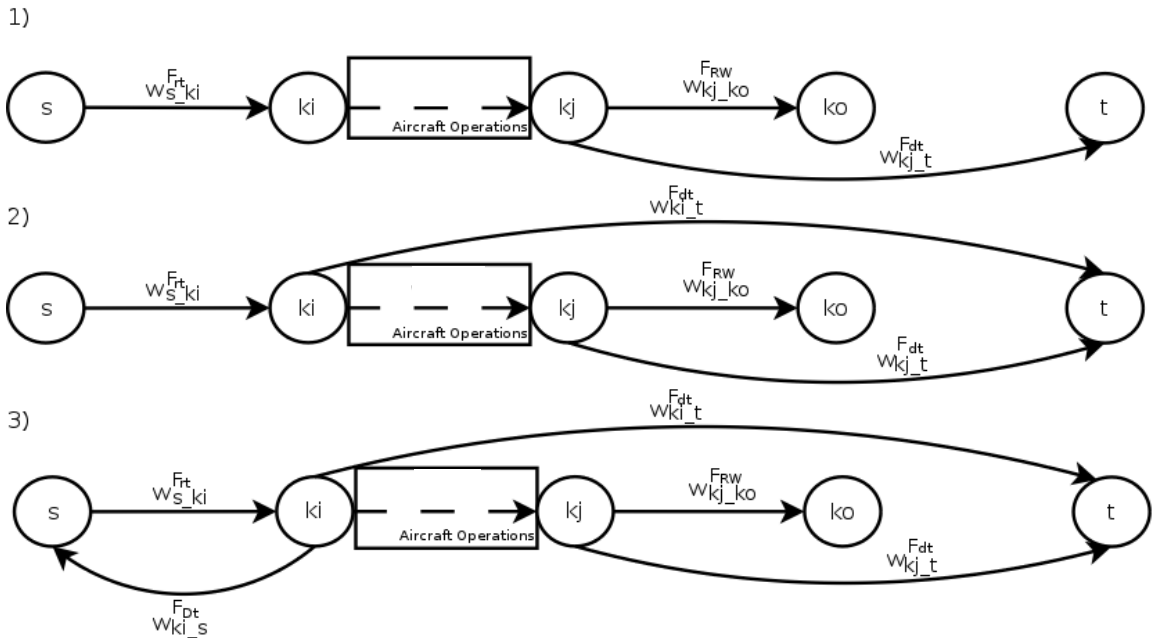


Figure 7.10: Illustration of model variants (M1, M2 and M3)

weight $w_{ki\_t}^{F_{dt}}$ is set equal to $-w_{s\_ki}^{F_{rt}}$. We penalize a late entrance in the TCA, since it may have an impact on the management of neighboring areas.

Model 3 (M3) is an extension of M2 with the addition of deadline constraints for landing aircraft at the entrance of the TCA, which limit their maximum possible entrance delay. This is achieved by inserting a deadline arc (ki, s) constraining the start of operation i of each landing aircraft k to be up to a maximum time. The weight $w_{ki\_s}^{F_{Dt}}$ is the maximum possible entry delay for aircraft k. This additional constraint may represent important practical differences between ground and airborne holding, the latter being more expensive and constrained. In fact, in some cases airborne holding is not even an option, e.g., due to limited fuel availability or to environmental policies aiming at the reduction of pollution and/or noise in the TCA. With deadline constraints, if a delayed aircraft cannot enter the TCA within its maximum feasible entrance time the solver returns an infeasibility related to that aircraft. In the computational experiments of this paper, the maximum possible entry delay for each landing aircraft has been fixed equal to the scheduled entrance time in the TCA plus fifteen minutes.

In all three models, following a common slot policy, the departure of an aircraft is considered on time within the first ten minutes after the planned departure time. In other words, the first ten minutes of delay of each departing aircraft are not considered by our objective function.

7.5.2 Disruption formulation

The management of a temporary unavailable resource can be modeled in terms of alternative graphs via a fictitious job with suitable timing constraints related to the disrupted resource.

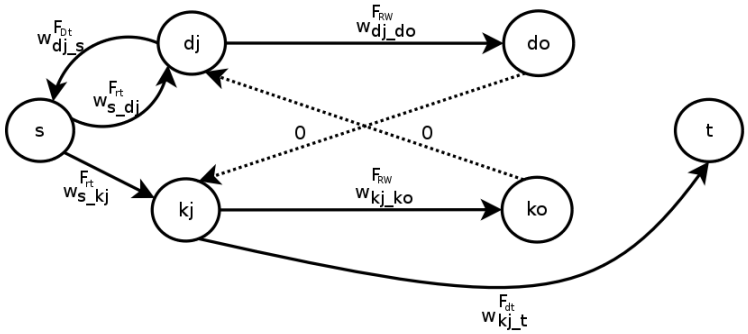


Figure 7.11: Examples of modeling of a disruption via alternative graphs

Figure 7.11 shows the case of a runway d disrupted in the deterministic time window [disruption start, disruption end], and a departing aircraft k (a similar example can be considered for a landing aircraft). The fictitious job of the disrupted runway is made by a start operation dj and an end operation do. The necessary time constraints are the following: the duration of the disruption (modeled by arc (dj, do) with $w^{F_{RW}}_{dj\_do}$ = disruption end - disruption start), the start time of the disruption (modeled by release arc (s, dj) with $w^{F_{rt}}_{s\_dj}$ = disruption start), the end time of the disruption (modeled by deadline arc (dj, s) with $w^{F_{Dt}}_{dj\_s}$ = - disruption start). The scheduling of aircraft k in runway d requires to consider its temporary unavailability. This is modeled by the fictitious alternative pair $((do, kj), (ko, dj))$ with $w^{A_{RW}}_{do\_kj} = w^{A_{RW}}_{ko\_dj} = 0$. If arc (do, kj) is selected, aircraft k enters the runway after the end time of the disruption in any feasible schedule. If arc (ko, dj) is selected, aircraft k leaves the runway before the start time of the disruption in any feasible schedule. The latter constraint is forced by the deadline arc (dj, s), since a start time of $h_{ko} > |w^{F_{Dt}}_{dj\_s}|$ would cause the positive weight cycle $s - kj - ko - dj - s$ of weight $h_{ko} + w^{F_{Dt}}_{dj\_s}$ in the graph.

7.5.3 ATC-TCA delay instances

The ATC-TCA instances considered in the experiments are taken from real aircraft routing and scheduling data collected for the Milano Malpensa TCA, in Italy. Disturbed traffic conditions are simulated in a given time period of traffic prediction. Three time horizons of different length are considered: 30 minutes, 60 minutes and 180 minutes. The entry delays are chosen randomly according to a uniform distribution and up to a given maximum value. These delays are generated each 15 minutes and are applied to 1 to 5 aircraft

entering the network in the first half of the time horizon under examination. For each time horizon, 20 ATC-TCA delay instances are analyzed: 10 light disturbances (the random delays are up to 5 minutes), and 10 heavy disturbances (the random delays are up to 15 minutes).

Table 7.1: Characteristics of the ATC-TCA delay instances of Milano Malpensa

Time Horizon Length (min)	Tot Num of Landing/Take-off Aircraft	Avg Entry Delay (sec)	Max Entry Delay (sec)	Avg Num of Delayed Aircraft	Tot Num of Landing/Take-off Aircraft Routes	Avg Num of Landing/Take-off Aircraft Routes
30	14/6	40	441	3.3	28/6	2/1
60	23/16	63	530	7.5	46/16	2/1
180	69/48	103	637	36.5	318/96	4.6/2

Table 7.1 gives further information on the 60 ATC-TCA delay instances. Each row reports average data over the 20 instances of a specific time horizon. Column 1 gives the time horizon length (in minutes), Column 2 the number of landing and take-off aircraft, Columns 3–4 the average and maximum entry delays (in seconds), Column 5 the average number of delayed aircraft (i.e. aircraft entering the network after their release time), Columns 6 the total number of landing/take-off aircraft routes given as input to the solver, Column 7 the average number of routes generated for landing/take-off aircraft. The last two columns characterize the complexity of the routing problem.

Instances can be divided into two groups: those with a time horizon of 30 and 60 minutes are of practical interest for air traffic control, while the second group of larger instances with a time horizon of 180 minutes has been added to stress the solvers. In the first group the routes of each arriving aircraft are the most promising only, i.e., the shortest route from the entrance point of each aircraft toward each runway. In fact, as observed in a previous work [D'Ariano et al., (2012a)], these routes are more promising compared to re-routing in air segments. In the tested instances there are only two runways and therefore the number of routes evaluated per arriving aircraft is two. The second group contains more challenging instances, that are proposed to evaluate the computational behavior of the solvers. In this second group we increased both the number of aircraft and the number of routes per aircraft, allowing re-routing of arriving aircraft both in air segments and runways, while the departing aircraft have routing variables at runways only, since we do not include departing air segments in our formulations.

Table 7.2 presents a detailed illustration of the difference between the various sets of ATC-TCA instances. Each row presents the average number of fixed constraints (Column 4), alternative constraints (Column 5) and MILP variables (Columns 6–8) related to the 20 instances of a specific time horizon of traffic prediction (Column 1), type of problem (Column 2) and model variant (Column 3). The model variants differ in terms of fixed constraints, since different sets of due date and deadline arcs are used. The scheduling problem presents a smaller number of fixed and alternative constraints compared to the scheduling and routing problem, since

Table 7.2: Size of the alternative graph and the MILP formulations

Time Horizon	Type of Problem	Model Variant	Num of Fixed Constraints	Num of Alternative Constraints	MILP Variables h	x	y
30 min	Scheduling	1	248	1786	145	893	-
		2	262	1786	145	893	-
		3	276	1786	145	893	-
	Routing & Scheduling	1	436	2362	186	1181	34
		2	464	2362	186	1181	34
		3	478	2362	186	1181	34
60 min	Scheduling	1	332	3042	187	1521	-
		2	355	3042	187	1521	-
		3	378	3042	187	1521	-
	Routing & Scheduling	1	728	11526	264	5763	62
		2	751	11526	264	5763	62
		3	774	11526	264	5763	62
180 min	Scheduling	1	994	26964	557	13482	-
		2	1063	26964	557	13482	-
		3	1132	26964	557	13482	-
	Routing & Scheduling	1	4331	456476	871	228238	414
		2	4649	456476	871	228238	414
		3	4967	456476	871	228238	414

one route per aircraft is modeled in the former problem and all available routes of each aircraft are modeled in the latter problem. The 180-minute traffic predictions present a large increase of the number of alternative arcs, since multiple routes are available for each landing and take-off aircraft.

7.5.4 Setting of the solvers

This subsection discusses the settings of the two solvers. Regarding the rolling horizon framework, we fix the roll period to 10 minutes and the look-ahead period to 15 minutes, since this is the best configuration obtained from a previous set of experiments. Samà et al. [Samà et al., (2013a)] tested 180-minute instances of the MXP airport with similar aircraft delays for a job shop scheduling model with fixed routing. The selection of the best configuration was mainly based on the following performance indicators: the maximum consecutive delay minimization and the speed of producing a feasible aircraft scheduling solution.

Table 7.3 shows the computation time (in seconds) used to test the various solution approaches. Each row refers to a specific time horizon of traffic prediction and presents the same total computation time for each approach (i.e. 120 seconds for 30-minute instances, 240 seconds for 60-minute instances, 720 seconds for 180-minute instances). The MILP formulation is solved by the IBM ILOG CPLEX MIP 12.0 solver, while BB and BB+TS are solved by the Roma Tre AGLIBRARY solver. In Table 7.3 we report two limits of total computation time when using CPLEX. The first time limit (named MILP) is needed to report the results after the same computation time to the other solution approaches. The second larger time limit (named MILP*) is used to certify optimality for all 30-minute and 60-minute instances. All the experiments

but MILP* are executed on a PC with processor Intel Core 2 Duo E6550 (2.33 GHz), 2 GB of RAM and Windows XP operative system. Experiments under MILP* are executed on a workstation Power Mac with processor Intel Xeon E5 quad-core (3.7 GHz), 12 GB of RAM.

For the rolling horizon approach, the time limit of computation for each look-ahead period to solve is the same for all time horizons of traffic prediction, since we use a look-head period of fixed length. For both the centralized and rolling horizon approaches, we tuned the time limit of computation of BB in the BB+TS algorithm in order to obtain a feasible schedule for all instances.

Table 7.3: Time limit for each combination of the solution approaches

Time Horizon Length	Centralized Framework					Rolling Horizon Framework					
	BB	MILP	MILP*	BB + TS		BB	MILP	BB + TS		Num Periods	Time Tot
				BB	TS			BB	TS		
30 min	120	120	3600	4	120	40	40	4	40	3	120
60 min	240	240	3600	4	240	40	40	4	40	6	240
180 min	720	720	–	30	720	40	40	4	40	18	720

7.5.5 Computational results

This subsection presents the computational assessment on the three model variants (M1, M2 and M3) of Section 7.5.1. The model variants are solved by the two frameworks (centralized and rolling horizon) and the four approaches (FIFO, BB for scheduling with default routes, BB+TS for scheduling and re-routing, the MILP formulation for simultaneous scheduling and routing) of Sections 7.3 and 7.4.

The instances generated for each model variant are the 60 ATC-TCA delay instances of Section 7.5.3, plus the 20 ATC-TCA disruption instances generated as follows: we inserted a temporary disruption in the 20 ATC-TCA delay instances with 180-minute time horizon of traffic prediction. Specifically, runway 16 is disrupted in the time window [t_0 + 60 min, t_0 + 120 min] (all aircraft have to be scheduled on the other runway available in that time window). This traffic disruption is modeled via the alternative graph of Section 7.5.2. For the 20 ATC-TCA disruption instances, we use the same time limit of computation shown in row 3 of Table 7.3 for each combination of the solution approaches (see Section 7.5.4).

This subsection is organized as follows. Section 7.5.5 compares the results obtained when using the optimization algorithms of AGLIBRARY with/without pre-processing for the models with deadline constraints. For each solution approach, Section 7.5.5 presents the computational results for the 60 ATC-TCA delay instances, while Section 7.5.5 shows the results for the 20 ATC-TCA disruption instances. A detailed comparison of the various solution approaches will be reported in terms of the following indicators:

- Optimality: this is the number of instances, out of 20 that were simulated, for which a specific approach was able to find the optimal solution for the ASP (*Num Optim Sched*) and for the ATC-TCA problem

(*Num Optim Rout*). The value of the optimal ASP solutions is found by BB, while the value of the optimal ATC-TCA solutions is certified by CPLEX. Besides CPLEX and BB, optimality is certified whenever a solution is found with maximum consecutive delay equal to zero.

- Computation time (in seconds): we use a number of indicators that are the average time (*Avg Best*) and maximum time (*Max Best*) required to find the best solution, plus the average total time (*Avg Tot*) and the maximum total time (*Max Tot*) required by a specific approach.

- Violation: this is the number of violations (*Num Viol*) of the deadline constraints related to the maximum possible entry delay of each landing aircraft in the TCA.

- Delay minimization (in seconds): the objective function is expressed in terms of the maximum consecutive delay minimization. Specifically, M2 and M3 measure this objective at the entrance resources of the TCA (*Max Entry Delay*) and at the runway resources (*Max Runway Delay*), while M1 measures this objective at the runway resources only. Furthermore, the maximum between Max Entry Delay and Max Runway Delay is reported as *Max Max Delay*. For each set of studied instances, the best value of the objective function of each model variant is reported in bold.

Results with/without pre-processing

The aim of this study is to assess the effectiveness of using the pre-processing procedure. Table 7.4 reports a computational analysis of the model variant with deadline constraints (M3). Each row gives the average results on the 20 ATC-TCA instances of a specific time horizon of traffic prediction (Column 1). As solution approaches, we compare the results with (*On*) and without (*Off*) pre-processing (Column 2) when using the centralized framework and the optimization algorithms of AGLIBRARY (Column 3).

Table 7.5 reports a computational analysis on the 20 ATC-TCA instances with temporary runway disruption, formulated via the addition of a fictitious job with deadline constraints (see Section 7.5.2). Each row gives the average results obtained for a specific model variant. We compare the same algorithms of Table 7.4.

From the results of Tables 7.4 and 7.5, we have the following observations. The pre-processing procedure is useful to compute better quality solutions, since the case *On* always gives equal or better quality solutions that the case *Off*. The improvement is more evident for BB+TS that applies the pre-processing procedure at each run of the aircraft scheduling module. Also, one additional optimal solution for a 30-minute delay instance of M3 is found by BB and by BB+TS. However, the case *On* sometimes takes a slightly longer computation time to find the best solution. In the following subsection, we always use the case *On* for BB and BB+TS.

Table 7.4: Assessment with/without pre-processing for the 30-,60-,180-minute delay instances

Time Horizon (min)	Pre-processing	Approach	Num Optim Sched	Num Optim Rout	Avg Best (sec)	Max Best (sec)	Avg Tot (sec)	Max Tot (sec)	Objective Function (sec)
Centralized Framework - Model Variant M3									
30	Off	BB	1	-	3.7	4.2	116.0	120.0	35.7
30	Off	BB+TS	-	19	12.7	55.7	120.0	120.0	1.7
30	On	BB	2	-	3.3	3.5	115.8	120.0	35.7
30	On	BB+TS	-	20	11.0	40.3	120.0	120.0	**1.5**
60	Off	BB	1	-	1.3	2.3	228.1	240.0	40.7
60	Off	BB+TS	-	13	71.8	207.8	240.0	240.0	**10.8**
60	On	BB	2	-	7.3	134.0	228.0	240.0	38.0
60	On	BB+TS	-	13	71.9	208.1	240.0	240.0	**10.8**
180	Off	BB	0	-	19.9	86.7	720.0	720.0	99.5
180	Off	BB+TS	-	0	131.9	720.0	720.0	720.0	76.7
180	On	BB	0	-	20.4	94.6	720.0	720.0	99.5
180	On	BB+TS	-	0	152.8	720.0	720.0	720.0	**75.7**

Table 7.5: Assessment with/without pre-processing for the 180-minute disruption instances

Pre-processing	Approach	Num Optim Sched	Num Optim Rout	Avg Best (sec)	Max Best (sec)	Avg Tot (sec)	Max Tot (sec)	Objective Function (sec)
Centralized Framework - Model Variant M1								
Off	BB	0	-	8.8	10.3	720.0	720.0	**951.8**
Off	BB+TS	-	0	224.5	720.0	720.0	720.0	957.6
On	BB	0	-	10.5	16.3	720.0	720.0	**951.8**
On	BB+TS	-	0	203.3	706.9	720.0	720.0	954.0
Centralized Framework - Model Variant M2								
Off	BB	0	-	9.4	11.0	720.0	720.0	953.8
Off	BB+TS	-	0	226.3	720.0	720.0	720.0	939.5
On	BB	0	-	9.3	11.3	720.0	720.0	951.8
On	BB+TS	-	0	290.5	720.0	720.0	720.0	**925.3**
Centralized Framework - Model Variant M3								
Off	BB	0	-	161.1	488.0	720.0	720.0	**1000.8**
Off	BB+TS	-	0	260.0	720.0	720.0	720.0	1027.7
On	BB	0	-	156.2	472.3	720.0	720.0	**1000.8**
On	BB+TS	-	0	207.3	720.0	720.0	720.0	1023.4

Results for the entry delays

Tables 7.6, 7.7 and 7.8 show the computational results for the 30-minute, 60-minute and 180-minute delay instances of Table 7.1. Each row of the tables presents average results over the 20 ATC-TCA delay instances of a time horizon of traffic prediction for a specific combination of model variant (M1, M2 and M3), framework (centralized and rolling horizon) and approach (FIFO, BB, BB+TS and MILP). Under MILP* we report the number of optimal solutions and their values that are certified by CPLEX within a computation time of 3600 seconds. We report the results obtained for each combination, except the MILP approach combined with the centralized framework for the 180-minute delay instances. In fact, the latter approach did not obtain a feasible schedule for any of the 20 instances. Similarly to the previous subsection, we assess the

various combinations in terms of the indicators related to optimality, computation time, violations and delay minimization. We next discuss the results of Tables 7.6, 7.7 and 7.8 when comparing the 3 model variants, the 2 frameworks, and the 4 approaches.

Table 7.6: Assessment for the 30-minute delay instances

Approach	Num Optim Sched	Num Optim Rout	Avg Best (sec)	Max Best (sec)	Avg Tot (sec)	Max Tot (sec)	Num Viol	Max Max Delay (sec)	Max Entry Delay (sec)	Max Runway Delay (sec)
Centralized Framework - Model Variant M1										
FIFO	15	-	0.3	0.3	0.3	0.3	0	114.1	106.7	36.2
BB	20	-	0.3	0.9	6.9	46.2	0	116.0	116.0	14.9
BB+TS	-	19	13.9	118.0	120.0	120.0	0	83.6	83.6	1.6
MILP	-	20	1.5	4.2	1.6	4.3	0	75.9	75.9	**1.1**
Rolling Horizon Framework - Model Variant M1										
FIFO	15	-	-	-	0.8	0.9	0	114.1	106.7	36.2
BB	20	-	-	-	0.9	3.4	0	116.2	116.2	14.9
BB+TS	-	18	-	-	120.0	120.0	0	85.0	85.0	2.0
MILP	-	20	-	-	2.5	4.5	0	33.6	33.5	**1.1**
Centralized Framework - Model Variant M2										
FIFO	0	-	0.3	0.5	0.3	0.5	0	114.1	106.7	36.2
BB	7	-	0.3	0.9	96.8	120.0	0	26.0	8.4	25.1
BB+TS	-	19	14.5	106.7	120.0	120.0	0	2.0	1.1	1.6
MILP	-	20	1.9	4.9	2.0	5.0	0	**1.5**	1.3	1.1
Rolling Horizon Framework - Model Variant M2										
FIFO	0	-	-	-	0.8	0.8	0	114.1	106.7	36.2
BB	6	-	-	-	52.5	120.0	0	30.1	8.4	28.6
BB+TS	-	15	-	-	120.0	120.0	0	5.4	4.2	4.0
MILP	-	19	-	-	2.6	5.7	0	2.0	1.1	1.6
Centralized Framework - Model Variant M3										
FIFO	0	-	0.2	0.2	0.2	0.2	0	114.1	106.7	36.2
BB	2	-	3.3	3.5	115.8	120.0	0	35.7	11.2	31.3
BB+TS	-	20	11.0	40.3	120.0	120.0	0	**1.5**	1.1	1.1
MILP (MILP*)	-	18 (20)	16.7	21.3	31.9	120.0	0	**1.5** (**1.5**)	1.3	1.1
Rolling Horizon Framework - Model Variant M3										
FIFO	0	-	-	-	0.8	0.8	0	114.1	106.7	36.2
BB	3	-	-	-	52.5	120.0	0	30.1	8.4	28.6
BB+TS	-	17	-	-	120.0	120.0	0	4.3	1.1	3.9
MILP	-	20	-	-	2.8	10.3	0	**1.5**	1.1	1.1

The solutions of the model variants obtained with an optimization approach differ from each other in terms of the indicators related to the maximum consecutive delay minimization. By construction we have that the M1 solutions better minimize the maximum runway delay, while the M2 and M3 solutions better minimize the maximum entry delay. The traffic controllers should therefore consider that a trade-off exists between the two indicators, e.g. selecting a solution with smaller runway delays may cause a later entrance in the TCA for some aircraft.

Regarding the impact of the deadline constraints, there is no violation in all ATC-TCA solutions computed for the delay instances. However, when dealing with the 180-minute delay instances the best solutions are obtained for M3, in terms of the various delay indicators. For this set of instances, limiting the entry delay of the most delayed aircraft may help to minimize the maximum consecutive delay.

Table 7.7: Assessment for the 60-minute delay instances

Approach	Num Optim Sched	Num Optim Rout	Avg Best (sec)	Max Best (sec)	Avg Tot (sec)	Max Tot (sec)	Num Viol	Max Max Delay (sec)	Max Entry Delay (sec)	Max Runway Delay (sec)
Centralized Framework - Model Variant M1										
FIFO	8	-	0.4	0.5	0.4	0.5	0	156.6	145.7	71.8
BB	9	-	0.5	1.2	192.1	240.0	0	140.5	140.5	30.8
BB+TS	-	15	51.3	222.8	240.0	240.0	0	126.5	126.5	7.8
MILP (MILP*)	-	19 (20)	50.2	240.0	50.4	240.0	0	112.9	112.9	7.1 (**4.0**)
Rolling Horizon Framework - Model Variant M1										
FIFO	9	-	-	-	1.3	1.4	0	155.3	143.5	71.2
BB	10	-	-	-	35.4	76.5	0	125.0	125.0	29.8
BB+TS	-	13	-	-	240.0	240.0	0	97.4	97.4	9.8
MILP	-	20	-	-	7.1	28.1	0	88.4	88.4	**4.0**
Centralized Framework - Model Variant M2										
FIFO	0	-	0.4	0.5	0.4	0.5	0	156.6	145.7	71.8
BB	2	-	7.2	132.4	228.0	240.0	0	38.0	18.2	36.5
BB+TS	-	15	64.9	232.8	240.0	240.0	0	9.7	1.6	9.3
MILP (MILP*)	-	19 (20)	44.9	240.0	45.1	240.0	0	7.5 (**7.45**)	3.4	7.1
Rolling Horizon Framework - Model Variant M2										
FIFO	0	-	-	-	1.4	1.6	0	155.3	143.5	71.2
BB	1	-	-	-	71.1	122.5	0	49.8	31.3	36.8
BB+TS	-	10	-	-	240.0	240.0	0	16.4	4.5	14.3
MILP	-	13	-	-	10.4	48.9	0	12.0	4.4	10.1
Centralized Framework - Model Variant M3										
FIFO	0	-	0.4	0.4	0.4	0.4	0	156.6	145.7	71.8
BB	2	-	7.3	134.0	228.0	240.0	0	38.0	18.2	36.5
BB+TS	-	13	71.9	208.1	240.0	240.0	0	10.8	2.1	10.4
MILP (MILP*)	-	19 (20)	10.1	59.8	46.2	240.0	0	8.8 (**7.45**)	5.3	8.4
Rolling Horizon Framework - Model Variant M3										
FIFO	0	-	-	-	1.3	2.0	0	155.3	143.5	71.2
BB	2	-	-	-	70.9	122.5	0	49.8	31.3	36.8
BB+TS	-	9	-	-	240.0	240.0	0	21.9	4.5	19.8
MILP	-	16	-	-	8.7	21.3	0	10.2	4.8	8.3

The selection of the most suitable framework depends on the type of instance. For the 30-minute and 60-minute delay instances, the centralized framework often offers the best results in terms of the objective function of the three model variants. A different trend is found for the 180-minute delay instances in which the rolling horizon framework outperforms the centralized framework.

The MILP approach is always the best solution approach. For the 30-minute and 60-minute delay instances, the MILP [respectively, the MILP*] approach in the centralized framework certifies optimality in 58/60 [60/60] and 52/60 [60/60] cases. The MILP approach in the rolling horizon framework finds an optimal solution in 59/60 and 49/60 cases, respectively. For the 180-minute delay instances, the latter approach is always the best approach, however we have no information on the value of the optimal solutions.

The time to compute the best solution of the MILP approaches is always below 22 seconds for the 30-minute delay instances, while up to around 240 seconds and 500 seconds are required for the 60-minute and 180-minute delay instances. Considering the time to compute good quality solutions, the MILP approach combined with the rolling horizon framework takes up to 11 seconds for 30-minute delay instances and up

Table 7.8: Assessment for the 180-minute delay instances

Approach	Num Optim Sched	Num Optim Rout	Avg Best (sec)	Max Best (sec)	Avg Tot (sec)	Max Tot (sec)	Num Viol	Max Max Delay (sec)	Max Entry Delay (sec)	Max Runway Delay (sec)
Centralized Framework - Model Variant M1										
FIFO	0	-	4.9	5.5	4.9	5.5	0	204.2	158.3	201.3
BB	0	-	27.8	86.2	684.3	720.0	0	171.7	171.7	99.0
BB+TS	-	0	175.8	666.2	720.0	720.0	0	156.0	156.0	76.9
Rolling Horizon Framework - Model Variant M1										
FIFO	0	-	-	-	14.5	15.2	0	208.5	158.2	205.8
BB	0	-	-	-	418.7	567.5	0	153.4	153.4	99.4
BB+TS	-	0	-	-	720.0	720.0	0	139.8	139.8	50.4
MILP	-	0	-	-	273.7	463.6	0	180.7	180.7	**31.2**
Centralized Framework - Model Variant M2										
FIFO	0	-	4.7	5.8	4.7	5.8	0	204.2	158.3	201.3
BB	0	-	17.2	83.4	684.3	720.0	0	99.5	57.9	99.5
BB+TS	-	0	188.1	629.7	720.0	720.0	0	75.9	47.9	75.5
Rolling Horizon Framework - Model Variant M2										
FIFO	0	-	-	-	14.6	15.0	0	208.5	158.2	205.8
BB	0	-	-	-	465.1	540.5	0	113.4	66.8	111.9
BB+TS	-	0	-	-	720.0	720.0	0	67.6	41.7	66.5
MILP	-	0	-	-	348.4	472.1	0	**43.0**	33.7	31.4
Centralized Framework - Model Variant M3										
FIFO	0	-	5.0	5.8	5.0	5.8	0	204.2	158.3	201.3
BB	0	-	20.4	94.6	720.0	720.0	0	99.5	57.9	99.5
BB+TS	-	0	152.8	720.0	720.0	720.0	0	75.7	49.5	75.4
Rolling Horizon Framework - Model Variant M3										
FIFO	0	-	-	-	14.4	15.3	0	208.5	158.2	205.8
BB	0	-	-	-	539.1	540.0	0	113.4	66.8	111.9
BB+TS	-	0	-	-	720.0	720.0	0	67.8	43.6	65.4
MILP	-	0	-	-	388.3	499.1	0	**25.7**	15.6	20.5

to 49 seconds for the 60-minute delay instances. This combination can thus be considered as applicable for the real-time management of practical-size instances.

Regarding the other approaches, BB+TS better minimizes the objective function of each model variant compared to BB and FIFO, but it is less competitive than the MILP approach. The FIFO rule is the fastest approach, however it is also the worst approach in terms of aircraft delay minimization.

Results for the entry delays plus a disrupted runway

Table 7.9 shows average results obtained for the 20 ATC-TCA 180-minute disruption instances. Some approaches are not reported for the following reasons. FIFO very often failed to compute a feasible schedule. The MILP approach in the centralized framework was not able to find a feasible solution in any of the 20 instances we tested under the disrupted runway scenario. The problem of computing a feasible schedule in presence of a disruption becomes more difficult than for the delay instances, since a runway is disrupted and all aircraft have to be scheduled in the only available runway. We next compare the results of the three model variants, the two frameworks and the feasible approaches in terms of the indicators related to optimality,

computation time, violations and delay minimization.

From the computational results of Table 7.9, the M1 and M2 solutions present a number of violations of the deadline constraints. However, a feasible schedule is always found for M3 by delaying different aircraft (i.e. by computing a different sequencing and routing solution) compared to the M1 and M2 solutions. Specifically, M2 has the best performance in terms of the Max Max Delay minimization at the cost of 11 violations. The maximum consecutive delay is often due to late arrivals at the only available runway resource.

For this set of instances, BB+TS with the rolling horizon framework outperforms the other approaches in terms of delay minimization, even if this combination requires the largest computation time.

Table 7.9: Assessment for the 180-minute delay instances plus the temporary runway disruption

Approach	Num Optim Sched	Num Optim Rout	Avg Best (sec)	Max Best (sec)	Avg Tot (sec)	Max Tot (sec)	Num Viol	Max Max Delay (sec)	Max Entry Delay (sec)	Max Runway Delay (sec)
Centralized Framework - Model Variant M1										
BB	0	-	10.5	16.3	720.0	720.0	19	956.1	605.0	951.8
BB+TS	-	0	203.3	706.9	720.0	720.0	19	958.3	642.1	954.0
Rolling Horizon Framework - Model Variant M1										
BB	0	-	-	-	515.3	578.9	11	892.2	439.0	890.6
BB+TS	-	0	-	-	720.0	720.0	11	844.6	396.6	**844.6**
MILP	-	0	-	-	458.1	655.2	10	853.0	431.8	852.8
Centralized Framework - Model Variant M2										
BB	0	-	9.3	11.3	720.0	720.0	18	951.8	595.1	951.8
BB+TS	-	0	290.5	720.0	720.0	720.0	17	925.3	556.3	925.3
Rolling Horizon Framework - Model Variant M2										
BB	0	-	-	-	551.8	639.7	13	853.4	401.8	853.4
BB+TS	-	0	-	-	720.0	720.0	11	**770.9**	366.9	770.9
MILP	-	0	-	-	511.0	590.3	12	835.7	453.1	835.7
Centralized Framework - Model Variant M3										
BB	0	-	156.2	472.3	720.0	720.0	0	1000.8	578.0	1000.8
BB+TS	-	0	207.3	720.0	720.0	720.0	0	1023.4	577.0	1023.4
Rolling Horizon Framework - Model Variant M3										
BB	0	-	-	-	548.8	636.9	0	908.0	405.2	908.0
BB+TS	-	0	-	-	720.0	720.0	0	**872.3**	344.6	872.3
MILP	-	0	-	-	504.9	595.6	0	910.0	411.6	905.1

7.6 Conclusions and further research

This paper investigates the potential of using optimization-based approaches as decision support for air traffic control at a busy TCA, including the management of strong traffic disturbances (such as multiple aircraft delays and a temporarily disrupted runway). Centralized and rolling horizon frameworks are evaluated on a Italian practical case study, i.e. Milano Malpensa (MXP). For both frameworks, we analyzed the performance of a commonly used rule (i.e. the FIFO rule), a branch and bound algorithm for aircraft scheduling, a tabu search algorithm for iterative aircraft scheduling and re-routing, and a new MILP formulation for simultaneous aircraft scheduling and routing. We also compared the performance of the various approaches

for three model variants, with some differences in the objective function and in the set of constraints.

From the computational results on the MXP TCA case study, we obtained the following insights:

- The solutions obtained for the three model variants show that there is a trade-off between the minimization of the entry delays and the runway delays. The minimization of the former delays is necessary for optimizing the coordination with the management of neighboring areas, while the minimization of the latter delays is necessary for the minimization of the passenger/freight delays and the delays of ground operations. Another comparison of the model variants is shown in terms of deadline constraint violations. For the disruption instances, a further trade-off is quantified between limiting the number of violations and the maximum consecutive delay. The resolution of the various model variants may thus give support to the traffic controllers in order to take more informed decisions.

- In the centralized framework, optimal solutions are computed for the MILP formulation when dealing with 30-minute and 60-minute delay instances. Most of them are found within a small computation time, but all the 40 instances are solved to the proven optimum within 1 hour of computation. However, CPLEX was not able to find a feasible solution in any of the 40 instances we tested with the 180-minute delay instances within a computation time of 720 seconds. Feasible solutions can be obtained, however, within a larger computation time. This result is due to the large number of scheduling and routing variables in the 180-minute instances. Further research should be dedicated to the development of exact methods and good lower bounds for the latter type of instances.

- The rolling horizon framework is the most robust approach, since a feasible schedule is always obtained when using the proposed optimization-based approaches for all tested instances. Furthermore, when dealing with 180-minute instances the rolling horizon framework outperforms the centralized framework in terms of the objective function minimized by each model variant.

- MILP is the best approach for the 30-minute, 60-minute and 180-minute delay instances, either in the centralized or rolling horizon framework. Up to the 60-minute delay instances, the MILP approach in the rolling horizon framework always computes a feasible schedule within less than one minute, so this combination can be potentially used for real-time traffic control. However, BB+TS in the rolling horizon framework is the best approach for the 180-minute disruption instances. The optimization algorithms of AGLIBRARY are thus competitive with the MILP approach. This result is more evident for the instances with deadline constraints, for which AGLIBRARY uses the pre-processing procedure. Comparing the AGLIBRARY approaches, BB+TS often presents better results than BB, since the use of re-routing adds value when optimizing the aircraft flows in the TCA.

- The poor results obtained by FIFO confirm that a potential improvement is achievable by adopting intelligent aircraft scheduling and routing techniques as decision support for traffic controllers. In fact, FIFO is not able to compute a feasible schedule for the disruption instances and is outperformed by all other approaches for the delay instances. Overall, an average improvement (in percentage) of around 64%, 88% and 93% is obtained when using BB, BB+TS and MILP respectively.

There are a number of research directions that deserve further investigation. The model could be improved by taking explicitly into account the issue of the perceived fairness of the solutions with respect to the different airlines, which has a direct impact on the possible acceptance of the optimization procedure by airlines in competition.

The solution approach could be improved along different directions. For example, the study of closed-loop decision support systems for traffic control would increase the accuracy of the models, since information is updated in real-time. It would be interesting to study approaches in which the length of the roll and look-ahead periods changes dynamically. Other interesting research directions should focus on the development of more efficient/effective re-ordering and re-routing algorithms, and on the investigation of multi-objective optimization approaches.

7.7 Appendix

The appendix presents the MILP formulation of the simultaneous aircraft scheduling and routing problem for the example of Section 7.3. The variables used in the formulation are the following:

- h_t is a non-negative real variable indicating the start time of the last operation;
- $h_{K.r.p}$ is a non-negative real variable indicating the start time of the node corresponding to the operation of aircraft K on resource p when using route r (route 1 being the default (off-line) route);
- $x_{K.r.p\_X.i.j\_Y.g.l\_Z.m.n}$ is a binary variable associated with the alternative pair (($K.r.p$, $X.i.j$), ($Y.g.l$, $Z.m.n$)), where each triple $K.r.p.$ indicates the node corresponding to the operation of aircraft K on resource p when using route r;
- $y_{K.r}$ is a binary variable indicating whether route r is chosen ($y_{K.r} = 1$) or not by aircraft K (specifically, $y_{K.1} = 1$ if the default route of aircraft K is selected).

The value of h_s is set to the start time of the first operation. For the numerical example, $h_s = 0$.

Minimize $h_t - h_s$

Subject To

Start time of source node:

$h_s = 0$

Fixed arcs for aircraft A:

$h_{A.1.out} - h_{A.1.16} - My_{A.1} \geq -M + 60$

$h_t - h_{A.1.16} - M\ y_{A.1} \geq -M - 262$

$h_{A.1.16} - h_{A.1.15} - M\ y_{A.1} \geq -M + 108$

$h_{A.1.15} - h_{A.1.16} - M\ y_{A.1} \geq -M - 133$

$h_{A.1.15} - h_{A.1.13} - M\ y_{A.1} \geq -M + 38$

$h_{A.1.13} - h_{A.1.15} - M\ y_{A.1} \geq -M - 63$

$h_{A.1.13} - h_{A.1.10} - M\ y_{A.1} \geq -M + 55$

$h_{A.1.10} - h_{A.1.13} - M\ y_{A.1} \geq -M - 63$

$h_{A.1.10} - h_{A.1.4} - M\ y_{A.1} \geq -M + 61$

$h_{A.1.4} - h_{A.1.10} - M\ y_{A.1} \geq -M - 65$

$h_{A.1.1} - h_{A.1.4} - M\ y_{A.1} \geq -M - 240$

$h_{A.1.4} - h_{A.1.1} - M\ y_{A.1} \geq -M$

$h_t - h_{A.1.1} - M\ y_{A.1} \geq -M$

$h_{A.1.1} - h_s - M\ y_{A.1} \geq -M$

$h_{A.2.out} - h_{A.2.17} - M\ y_{A.2} \geq -M + 60$

$h_t - h_{A.2.17} - M\ y_{A.2} \geq -M - 262$

$h_{A.2.17} - h_{A.2.15} - M\ y_{A.2} \geq -M + 108$

$h_{A.2.15} - h_{A.2.17} - M\ y_{A.2} \geq -M - 133$

$h_{A.2.15} - h_{A.2.11} - M\ y_{A.2} \geq -M + 45$

$h_{A.2.11} - h_{A.2.15} - M\ y_{A.2} \geq -M - 66$

$h_{A.2.11} - h_{A.2.5} - M\ y_{A.2} \geq -M + 51$

$h_{A.2.5} - h_{A.2.11} - M\ y_{A.2} \geq -M - 66$

$h_{A.2.1} - h_{A.2.5} - M\ y_{A.2} \geq -M - 240$

$h_{A.2.5} - h_{A.2.1} - M\ y_{A.2} \geq -M$

$h_t - h_{A.2.1} - M\ y_{A.2} \geq -M$

$h_{A.2.1} - h_s - M\ y_{A.2} \geq -M$

Fixed arcs for aircraft B:

$h_{B.1.out} - h_{B.1.17} - M\ y_{B.1} \geq -M + 60$

$h_t - h_{B.1.17} - M\ y_{B.1} \geq -M - 320$

$h_{B.1.17} - h_{B.1.15} - M\ y_{B.1} \geq -M + 138$

$h_{B.1.15} - h_{B.1.17} - M\ y_{B.1} \geq -M - 191$

$h_{B.1.15} - h_{B.1.14} - M\ y_{B.1} \geq -M + 38$

$h_{B.1.14} - h_{B.1.15} - M\ y_{B.1} \geq -M - 63$

$h_{B.1.14} - h_{B.1.12} - M\ y_{B.1} \geq -M + 55$

$h_{B.1.12} - h_{B.1.14} - M\ y_{B.1} \geq -M - 63$

$h_{B.1.12} - h_{B.1.8} - M\ y_{B.1} \geq -M + 21$

$h_{B.1.8} - h_{B.1.12} - M\ y_{B.1} \geq -M - 65$

$h_{B.1.3} - h_{B.1.8} - M\ y_{B.1} \geq -M - 240$

$h_{B.1.8} - h_{B.1.3} - M\ y_{B.1} \geq -M$

$h_t - h_{B.1.3} - M\ y_{B.1} \geq -M - 39$

$h_{B.1.3} - h_s - M\ y_{B.1} \geq -M + 39$

Fixed arcs for aircraft C:

$h_{C.1.out} - h_{C.1.16} - M\ y_{C.1} \geq -M + 60$

$h_t - h_{C.1.16} - M\ y_{C.1} \geq -M - 300$

$h_{C.1.16} - h_s - M\ y_{C.1} \geq -M + 300$

Alternative arcs for the holding circle resource of aircraft A:

$h_{A.1.4} - h_{A.1.1} - M\ x_{A.1.1\_A.1.4\_A.1.4\_A.1.4\_A.1.1} - M\ y_{A.1} \geq -2M + 240$

$h_{A.1.1} - h_{A.1.4} + M\ x_{A.1.1\_A.1.4\_A.1.4\_A.1.4\_A.1.1} - M\ y_{A.1} \geq -M - 180$

$h_{A.1.4} - h_{A.1.1} - M\ x_{A.1.4\_A.1.1\_A.1.1\_A.1.1\_A.1.4} - M\ y_{A.1} \geq -2M + 180$

$h_{A.1.1} - h_{A.1.4} + M\ x_{A.1.4\_A.1.1\_A.1.1\_A.1.1\_A.1.4} - M\ y_{A.1} \geq -M$

$h_{A.2.5} - h_{A.2.1} - M\ x_{A.2.1\_A.2.5\_A.2.5\_A.2.5\_A.2.1} - M\ y_{A.2} \geq -2M + 240$

$h_{A.2.1} - h_{A.2.5} + M\ x_{A.2.1\_A.2.5\_A.2.5\_A.2.5\_A.2.1} - M\ y_{A.2} \geq -M - 180$

$h_{A.2.5} - h_{A.2.1} - M\ x_{A.2.5\_A.2.1\_A.2.1\_A.2.1\_A.2.5} - M\ y_{A.2} \geq -2M + 180$

$h_{A.2.1} - h_{A.2.5} + M\ x_{A.2.5\_A.2.1\_A.2.1\_A.2.1\_A.2.5} - M\ y_{A.2} \geq -M$

Alternative arcs for the holding circle resource of aircraft B:

$h_{B.1.8} - h_{B.1.3} - M\ x_{B.1.3\_B.1.8\_B.1.8\_B.1.3} - M\ y_{B.1} \geq -2M + 240$

$h_{B.1.3} - h_{B.1.8} + M\ x_{B.1.3\_B.1.8\_B.1.8\_B.1.3} - M\ y_{B.1} \geq -M - 180$

$h_{B.1.8} - h_{B.1.3} - M\ x_{B.1.8\_B.1.3\_B.1.3\_B.1.8} - M\ y_{B.1} \geq -2M + 180$

$h_{B.1.3} - h_{B.1.8} + M\ x_{B.1.8\_B.1.3\_B.1.3\_B.1.8} - M\ y_{B.1} \geq -M$

Alternative arcs for the air segment resources:

$h_{B.1.15} - h_{A.1.15} - M\ x_{A.1.15\_B.1.15\_B.1.17\_A.1.16} - M\ y_{A.1} - M\ y_{B.1} \geq -3M + 42$

$h_{A.1.16} - h_{B.1.17} + M\ x_{A.1.15\_B.1.15\_B.1.17\_A.1.16} - M\ y_{A.1} - M\ y_{B.1} \geq -2M + 42$

$h_{A.1.15} - h_{B.1.15} - M\ x_{B.1.15\_A.1.15\_A.1.16\_B.1.17} - M\ y_{B.1} - M\ y_{A.1} \geq -3M + 42$

$h_{B.1.17} - h_{A.1.16} + M\ x_{B.1.15\_A.1.15\_A.1.16\_B.1.17} - M\ y_{B.1} - M\ y_{A.1} \geq -2M + 42$

$h_{B.1.15} - h_{A.2.15} - M\ x_{A.2.15\_B.1.15\_B.1.17\_A.2.17} - M\ y_{A.2} - M\ y_{B.1} \geq -3M + 42$

$h_{A.2.17} - h_{B.1.17} + M\ x_{A.2.15\_B.1.15\_B.1.17\_A.2.17} - M\ y_{A.2} - M\ y_{B.1} \geq -2M + 42$

$h_{A.2.15} - h_{B.1.15} - M\ x_{B.1.15\_A.2.15\_A.2.17\_B.1.17} - M\ y_{B.1} - M\ y_{A.2} \geq -3M + 42$

$h_{B.1.17} - h_{A.2.17} + M\ x_{B.1.15\_A.2.15\_A.2.17\_B.1.17} - M\ y_{B.1} - M\ y_{A.2} \geq -2M + 42$

Alternative arcs for the runway resources:

$h_{C.1.16} - h_{A.1.out} - M\ x_{A.1.out\_C.1.16\_C.1.out\_A.1.16} - M\ y_{A.1} - M\ y_{C.1} \geq -3M + 42$

$h_{A.1.16} - h_{C.1.out} + M\ x_{A.1.out\_C.1.16\_C.1.out\_A.1.16} - M\ y_{A.1} - M\ y_{C.1} \geq -2M + 42$

$h_{B.1.17} - h_{A.2.out} - M\ x_{A.2.out\_B.1.17\_B.1.out\_A.2.17} - M\ y_{A.2} - M\ y_{B.1} \geq -3M + 42$

$h_{A.2.17} - h_{B.1.out} + M\ x_{A.2.out\_B.1.17\_B.1.out\_A.2.17} - M\ y_{A.2} - M\ y_{B.1} \geq -2M + 42$

Constraints on the route selection:

$y_{C.1} = 1$

$y_{B.1} = 1$

$y_{A.1} + y_{A.2} = 1$

Bounds

$h_{k.r.p} \geq 0$

$h_t \geq 0$

Binary

x, y

Chapter 8

Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas

Abstract: *Intelligent decision support systems for the real-time management of landing and take-off operations can be very effective in helping traffic controllers to limit airport congestion at busy terminal control areas. The key optimization problem to be solved to this aim can be formulated as a mixed integer linear program. However, since this problem is strongly NP-hard, heuristic algorithms are typically adopted in practice to compute good quality solutions in a short computation time. This paper presents a number of algorithmic improvements implemented in the AGLIBRARY solver in order to improve the possibility of finding good quality solutions quickly. The proposed framework starts from a good initial solution for the scheduling problem with fixed routes, obtained via a truncated branch-and-bound algorithm. A metaheuristic is then applied to improve the solution by re-routing some aircraft. The new metaheuristics are based on variable neighbourhood search, tabu search and hybrid schemes. The neighbourhoods differ from each other in terms of the aircraft that are re-routed in each move. Computational experiments are performed on an Italian terminal control area under various types of disturbances, including multiple aircraft delays and a temporarily disrupted runway. The metaheuristics achieve solutions of remarkable quality, within a small computation time, compared with a commercial solver and with the previous versions of AGLIBRARY.*

8.1 Introduction

8.1.1 The investigated problem

In the last years, air traffic controllers are experiencing increasing difficulties to manage the ever increasing transport flows while ensuring safety and efficiency of operational schedules. This is partially due to the limited space and funds to build new infrastructure in bottleneck areas, but also due to the limited support offered by the actual traffic control systems. One typical bottleneck of the entire air traffic system is the Terminal Control Area (TCA). During operations, aircraft delays are considered to cause a substantial cost from both airlines and passengers' points of view. The computation of optimal aircraft landing and take-off schedules is thus one of the most relevant operational problems. These facts stimulated the interest for effective intelligent transport system solutions that can show how to better use the existing resources [Zhang (2008)].

This paper aims to develop good quality solutions for the Air Traffic Control in a Terminal Control Area (ATC-TCA) problem. The investigated problem consists of simultaneously determining the routing (i.e. the resources to be traversed), sequencing (i.e. the orders between aircraft in each resource) and timing of landing and take-off aircraft on the TCA resources, which may include several runways and air segments. The objective is to minimize the maximum positive deviation from the target landing and take-off times.

Generally, the resolution of air traffic control problems requires to consider several factors related to safety, efficiency and equity. In this work, safety requires the careful modelling of practical TCA constraints, efficiency consists of reducing aircraft delays with global conflict detection and resolution approaches, equity is achieved by minimizing the largest delay due to the resolution of conflicts. These factors require to consider the routing, sequencing and timing of all aircraft moving in the network during the studied traffic horizon. Further relevant safety, efficiency, equity, and even environmental impacts are addressed, e.g., in [Soomer and Franx (2008), Soomer and Koole (2008), Solveling et al., (2011)].

8.1.2 The related literature

In the aircraft scheduling literature, it is often mentioned a big gap existing between the level of sophistication of published results and algorithms, and the simple methods that are employed in practice. One motivation reported for this gap is that the theory typically addresses very simplified problems for which optimal or near-optimal performance can be achieved, while the practice must face all the complexity of real-time operations, often with little regard to the performance level. However, poorly performing aircraft scheduling and routing methods that are used in practice directly impact on the quality of service offered to the passengers, the

effect being more evident as traffic density gets close to saturation. In fact, in this case any small disturbance related to few aircraft may propagate to the other aircraft, altering the regularity of air traffic even some hours after the end of the original disturbance.

There is a recent trend of research to incorporate more practical objectives and constraints in the detailed (microscopic) models that have not been adequately captured in published models, since too simplified (macroscopic) models may have a limited impact on the practice of air traffic control. In view of the extensive reviews reported in [Ball et al., (2007), Barnhart et al (2012), Bennell et al., (2011), Kohl et al., (2007), Kuchar and Yang (2000), Pellegrini and Rodriguez (2013)], we limit our review of the recent related literature to two streams of research: (i) the development of microscopic models for the management of aircraft flows in terminal control areas, (ii) the development of algorithmic methods in air traffic control.

Regarding stream (i), there are a few microscopic models that are able to incorporate the detailed information that is compliant with the safety regulations of the TCA, including the characteristics of the airport infrastructure and of the individual flight paths. Such a level of detail is required to safely detect and solve potential conflicts at the level of runways, ground and air segments of the TCA. The most detailed model used in this context is the job shop scheduling model in which each operation denotes the traversal of a resource (air/ground segment, runway) by a job (aircraft). The variables are the start time of each operation to be performed by an aircraft on a specific resource. A *no-wait* version of this model has been firstly proposed in [Bianco et al., (2006)] and successively extended in [D'Ariano et al., (2015),D'Ariano et al., (2010), D'Ariano et al., (2012b), Mascis and Pacciarelli (2002), Samà et al., (2014)] as a *blocking and no-wait* version. In the latter approach, air segment resources are treated as no-wait resources with time windows for modelling minimum/maximum aircraft travel times, while runway resources are treated as blocking resources which can host at most one aircraft at a time. Objective functions are based on the makespan minimization.

Regarding stream (ii), exact and heuristic algorithms have been proposed for the ATC-TCA problem. Among the literature on exact algorithms, Balakrishnan and Chandran [Balakrishnan and Chandran (2010)] propose a combination of the constrained position shifting approach (originally introduced by [Dear (1976)]) and of the dynamic programming approach (originally introduced by [Psaraftis (1980)]) to runway scheduling, D'Ariano et al. [D'Ariano et al., (2015)] describe a branch and bound algorithm for the aircraft scheduling problem with fixed routes, Faye [Faye (2015)] present a dynamic constraint generation algorithm for an aircraft landing problem. However, exact algorithms can quickly compute near-optimal solutions only for quite small instances or simplified problems. Consequently, numerous metaheuristics have been recently proposed to search for good quality solutions in a short computation time, the most used being the following: genetic algorithms [Beasley et al., (2001), Hu and Chen (2005), Hu and Di Paolo (2008), Hu and Di Paolo (2009), Hu and Di Paolo (2011)], scatter search [Pinol and Beasley (2006)], tabu search [Atkin et al., (2007),

D'Ariano et al., (2012b), Furini et al., (2015, Samà et al., (2014)], ant colony [Bencheikh et al., (2011), Jiang et al., (2014), Zhan and Zhang (2010)], simulated annealing [Hancerliogullari et al., (2013), Salehipour et al., (2013)], variable neighbourhood search [Alonso-Ayuso et al., (2015), Salehipour et al., (2013), Salehipour et al., (2009)]. Several of the proposed algorithms have also been hybridized in order to combine interesting properties and to take the best from each of them. Furthermore, some approaches (e.g., [Furini et al., (2015, Hu and Chen (2005), Hu and Di Paolo (2008), Hu and Di Paolo (2009), Murça and Müller, Samà et al., (2014), Zhan and Zhang (2010)]) have been implemented in a rolling horizon (or receding horizon) control framework in order to solve large instances in a short computation time compatible with real-time applications, and to deal with the dynamic and uncertain nature of the ATC-TCA problem. All these approaches have proposed significant improvements compared to the commonly used air traffic control rules, such as the first-in-first-out rule. In fact, the usual control rules take a few sequencing and routing decisions at a time in a myopic fashion, ignoring the propagation of delays to other aircraft in the network [Hu and Di Paolo (2008), Nosedal et al., (2015)]. The proposed neighbourhood research methods are well focused on the minimization of the propagation of aircraft delays or other relevant factors, however the majority of the works focuses on the development of good neighbourhood search capabilities for aircraft sequencing or scheduling problems, while there is still a lack of advanced algorithmic contributions on the simultaneous aircraft scheduling and routing problem on the TCA resources, that is the main subject of this paper.

In view of the above discussion of the recent literature regarding the management of landing and take-off operations, there is a clear need to incorporate an increasing level of detail and realism in the models while keeping the computation time of the algorithms at an acceptable level. Furthermore, the ATC-TCA problem is well-known to be NP-hard, requiring the use of advanced heuristics, especially when solving complex instances with multiple delayed aircraft and severe resource capacity deficiencies. This paper deals with the real-world instances of Samà et al. [Samà et al., (2014)], with up to more than 200000 scheduling variables and more than 400 routing variables. Since it is not possible to solve these instances with an exact method in a reasonable amount of time, we focus our work to the development of hybrid metaheuristics (based on tabu search and variable neighbourhood search schemes) to derive good quality solutions in a short time.

8.1.3 The paper contribution

A recent stream of research on detailed ATC-TCA problem formulations focuses on the Alternative Graph (AG) of Mascis and Pacciarelli [Mascis and Pacciarelli (2002)]. This graph has been first successfully applied to manage other transportation and production problems [Corman et al., (2010), D'Ariano et al., (2007a), Pacciarelli and Pranzo (2004)]. In this paper, the ATC-TCA problem is modelled as a generalized job

shop scheduling problem via alternative graphs, enriching the model of [Bianco et al., (2006)] by addressing the real-world constraints and the objective function proposed in [D'Ariano et al., (2015), D'Ariano et al., (2010), D'Ariano et al., (2012b), Samà et al., (2013a), Samà et al., (2014)]. This graph allows a more accurate modelling of relevant TCA aspects and safety constraints, such as holding circles, waiting in flight before landing, travelling in feasible time windows, hosting multiple aircraft simultaneously in air segments and individual aircraft in runways. Specifically, the alternative graph can model any 4-dimensional route for the aircraft in the TCA, while most of the related work done assumes 3-D routes are fixed and only optimizes the timing of runway operations. In order to include the routing flexibility in the AG model, we make use of the Mixed Integer Linear Programming (MILP) formulation of [Samà et al., (2014)]. The MILP formulation can been efficiently solved by the rolling horizon framework of [Samà et al., (2013a)]. However, the latter approach requires a large computation time when dealing with complex ATC-TCA instances.

This paper presents a number of algorithmic improvements implemented in the AGLIBRARY solver, a set of optimization algorithms for complex job shop scheduling problems developed at Roma Tre University. The solver is based on the following framework: a good initial solution for the scheduling problem with fixed routes is computed by the (truncated) branch-and-bound algorithm in [D'Ariano et al., (2015), D'Ariano et al., (2010)]. Metaheuristics are then applied to improve the solution by re-routing some aircraft. This action corresponds to the concept of a move, from a metaheuristics perspective. In [D'Ariano et al., (2012b)], a tabu search algorithm has been applied to solve practical-size instances for small disturbances. Previous research left open the following two relevant issues. The first issue concerns the extent at which different solution methods might outperform the tabu search algorithm and the rolling horizon framework. A second issue is to study algorithmic improvements, in order to reduce the time to compute good quality solutions. Both these issues motivate the development of the new metaheuristics proposed in this paper. The paper contributions are next outlined:

- We present new routing neighbourhoods that differ from each other in terms of the aircraft that are re-routed in each move and for the set of candidate aircraft routes;

- We alternate the search for promising moves in neighbourhoods of different size, similarly to [Moreno Pérez et al., (2003)], adopt fast re-scheduling heuristics for the neighbour evaluation, and present strategies for searching within these neighbourhoods based on variable neighbourhood search, tabu search and hybrid schemes;

- We apply the proposed algorithms to the management of complex disturbed situations, including multiple delayed landing and/or take-off aircraft and a temporarily disrupted runway. The situations tested are the most complex instances in [Samà et al., (2014)]. The new metaheuristics are compared

with the other existing methods based on the AG model, and with solutions computed with a commercial MILP solver. Significantly better results are obtained in terms of an improved solution quality and/or a reduced computation time with respect to both the MILP solver and the previous versions of AGLIBRARY.

Section 8.2 formally describes the ATC-TCA problem and the MILP formulation. Section 8.3 presents the metaheuristic algorithms proposed in this paper. Section 8.4 reports the performance of the algorithms on the Milan Malpensa terminal control area (MXP) instances of Samà et al. [Samà et al., (2014)]. Section 8.5 summarizes the paper results and outlines future research directions. An appendix illustrates the neighbourhoods used in this paper with a numerical example.

8.2 Problem definition and formulation

8.2.1 The ATC-TCA problem

Landing aircraft move in the landing air segments of the TCA, following a standard descent profile, from an air entry point to a common glide path, that is the final landing air segment before the runway. Take-off aircraft move in the ground resources till they get access to the runway and finally fly toward their assigned exit point via take-off air segments.

A minimum longitudinal and diagonal safety separation distance between every pair of consecutive aircraft must be always respected, depending on their type, altitude and relative positions. This minimum distance can be translated into a *minimum separation time* that is sequence-dependent, since it depends on the relative processing order of the common resources by the different aircraft categories.

Each aircraft has a *processing time* on each TCA resource, according to its landing/take-off profile. On the air segments, the processing time varies between minimum and maximum feasible values.

Each landing/take-off aircraft has a minimum entrance time into the TCA, *release time*, according to its current position and speed. Landing aircraft can also be constrained to have a maximum entrance time, *deadline time*, into the TCA, e.g., due to limited fuel availability.

All aircraft have scheduled times, *due date times*, to start processing some TCA resources. A departing aircraft is supposed to take-off within its assigned time window and is late whenever it is not able to accomplish the departing procedure within its assigned time window. Following the procedure commonly adopted by air traffic controllers, we consider a time window for take-off between 5 minutes before and 10 minutes after the *Scheduled Take-off Time* (STT). A departing aircraft is considered delayed in exiting the TCA if leaving the runway after 10 minutes from its STT. Arriving aircraft are late if landing after their

Scheduled Landing Time (SLT).

Before entering the TCA, landing aircraft can fly in *holding circles* that are air segments dedicated to accumulating aircraft delays during the flight. In each holding circle, landing aircraft must fly at a fixed speed for a number of half circles, as prescribed by the air traffic controller. Departing aircraft instead can be delayed in entering the TCA at ground level, i.e. before entering the runway.

We use the following notation for aircraft delays. *Entrance delay* (*exit delay*) is the delay of an aircraft on the entrance to (the exit from) the TCA. The exit value is partly a consequence of a possible late entrance, which causes an *unavoidable delay*, and partly due to additional delays caused by the resolution of potential aircraft conflicts in the TCA, which is the *consecutive delay*. In this paper, we minimize the maximum consecutive delay that is an equitable approach for the minimization of aircraft delay propagation.

The next section will show a model of ATC-TCA problem in which a route is assigned to each aircraft. This assumption will be then relaxed in order to deliver a general optimization model.

8.2.2 The AG model

This subsection presents the alternative graph for the ATC-TCA problem with pre-defined routes. This graph is a triple $G(N, F, A)$: $N = \{s, 1, ..., n-2, t\}$ is the set of *nodes*, where nodes s and t represent the start and the end operations of the schedule, while the other $n-2$ nodes are related to the start of the other operations; F is the set of *fixed directed arcs* that model the pre-defined aircraft routes; A is the set of *alternative pairs* that model the sequencing decisions. Each pair is composed of two directed arcs.

Each node, except s and t, is associated with the start of an operation krj, where k indicates the aircraft, r the route chosen and j the resource it traverses. The start time h_{krj} of operation krj is the entrance time of aircraft k in resource j when using route r.

Each fixed directed arc $(krp, krj) \in F$ is identified by the two nodes krp and krj that are connected, and it has associated the weight $w^F_{krp\_krj}$, representing a minimum time constraint between h_{krp} and h_{krj} (i.e. $h_{krj} - h_{krp} \geq w^F_{krp\_krj}$). The set F is the union of the disjoint sets: F_{rt} is the set of release time constraints, F_{dt} is the set of due date time constraints, F_{Dt} is the set of deadline time constraints, F_{HC} is the set of holding circle constraints, F_{AS} is the set of air segment constraints, F_{RW} is the set of runway constraints. A detailed description of the various sets of fixed directed arcs can be found in [Samà et al., (2014)].

Each alternative pair $((kro, uim), (uig, krl)) \in A$ models an aircraft holding (aircraft k = aircraft u and route r = route i) or sequencing (aircraft $k \neq$ aircraft u) decision. The two arcs of the pair have associated the weights $w^A_{kro\_uim}$ and $w^A_{uig\_krl}$. In any solution, only one arc of each pair in the set A can be selected. If alternative arc (kro, uim) $[(uig, krl)]$ is selected in a solution, the constraint $h_{uim} - h_{kro} \geq w^A_{kro\_uim}$ $[h_{krl} - h_{uig} \geq w^A_{uig\_krl}]$ has to be satisfied. The set A is composed by the subsets: A_{HC} for holding decisions,

A_{AS} and A_{RW} for sequencing decisions at air segments and runways. A detailed description of A_{HC}, A_{AS} and A_{RW} can be found in [Samà et al., (2014)].

A selection S is a set of alternative arcs obtained by selecting exactly one arc from each alternative pair in A and such that the resulting graph $\mathcal{G}(F,S) : (N, F \cup S)$ does not contain positive weight cycles. A solution to the ATC-TCA problem with pre-defined routes is a selection S. This allows to associate orders and times to all operations. The minimization of the maximum consecutive delay is measured as a makespan minimization. Given a selection S and any two nodes krp and uml, we let $l^S(krp, uml)$ be the weight of the longest path from krp to uml in $\mathcal{G}(F,S)$. By definition, the start time h_{krp} of $krp \in N$ is the quantity $l^S(s, krp)$, which implies $h_s = 0$ and $h_t = l^S(s,t)$.

8.2.3 The MILP formulation

The ATC-TCA problem with flexible routes is formulated as a particular *disjunctive program* [Samà et al., (2014)]. This is achieved via a MILP formulation in which the scheduling and routing decisions are considered simultaneously. The starting point is the alternative graph model for the ATC-TCA problem with pre-defined routes. The graph is formulated via a $big-M$ formulation enlarging the sets F and A in order to include the fixed and alternative arcs related to all possible aircraft routes. The advantage of this $big-M$ formulation is the exact correspondence between arcs and constraints: each fixed directed arc translates into a fixed constraint, while each alternative pair into a pair of alternative constraints. However, we observe that computing the optimal solution of $big-M$ formulations can be a time-consuming task for any solver.

We next give a compact $big-M$ formulation, while a detailed formulation is given in Samà et al. [Samà et al., (2014)]. For each alternative pair $((krp, dij), (uml, vnw)) \in A$ there is a binary variable $x^{uml,vnw}_{krp,dij}$ modelling the sequencing/holding decision. For each route r and aircraft k there is a binary variable y_{kr} modelling the route selection. For each operation krp there is a non-negative real variable h_{krp} modelling its start time.

$$\min \; h_t - h_s \tag{8.1}$$

$$\sum_{r=1}^{R_k} y_{kr} = 1 \qquad k = 1, ..., Z \tag{8.2}$$

$$h_{krj} - h_{krp} + M(1 - y_{kr}) \geq w^F_{krp\_krj} \qquad \forall (krp, krj) \in F \tag{8.3}$$

$$h_{krp} - h_s + M(1 - y_{kr}) \geq w^{F_{rt}}_{s\_krp} \qquad \forall (s, krp) \in F_{rt} \tag{8.4}$$

$$h_s - h_{krp} + M(1 - y_{kr}) \geq w^{F_{Dt}}_{krp\_s} \quad \forall (krp, s) \in F_{Dt} \tag{8.5}$$

$$h_t - h_{krj} + M(1 - y_{kr}) \geq w^{F_{dt}}_{krj\_t} \quad \forall (krj, t) \in F_{dt} \tag{8.6}$$

$$\begin{aligned} &h_{uim} - h_{kro} + M(2 + x^{uig\_krl}_{kro\_uim} - y_{kr} - y_{ui}) \geq w^{A}_{kro\_uim} \\ &h_{krl} - h_{uig} + M(3 - x^{uig\_krl}_{kro\_uim} - y_{kr} - y_{ui}) \geq w^{A}_{uig\_krl} \quad \forall ((kro, uim), (uig, krl)) \in A \end{aligned} \tag{8.7}$$

$$\begin{aligned} &h_{krj} - h_{krp} + M(1 + x^{krj\_krp}_{krp\_krj} - y_{kr}) \geq w^{A_{HC}}_{krp\_krj} \\ &h_{krp} - h_{krj} + M(2 - x^{krj\_krp}_{krp\_krj} - y_{kr}) \geq w^{A_{HC}}_{krj\_krp} \quad \forall ((krp, krj), (krj, krp)) \in A_{HC} \end{aligned} \tag{8.8}$$

$$x^{krj\_krp}_{krp\_krj} \in \{0, 1\} \quad \forall ((krp, krj), (krj, krp)) \in A \tag{8.9}$$

$$x^{uig\_krl}_{kro\_uim} \in \{0, 1\} \quad \forall ((kro, uim), (uig, krl)) \in A \tag{8.10}$$

$$y_{kr} \in \{0, 1\} \quad k = 1, ..., Z \ ; \ r = 1, ..., R_k \tag{8.11}$$

The objective function is reported in Equation (8.1). We next describe the ATC-TCA problem constraints.

Constraints (8.2) model the routing decision for each aircraft k among its set of R_k routes. The route r is chosen for aircraft k if and only if $y_{kr} = 1$. In total, there are Z aircraft.

Constraints (8.3) model the fixed directed arcs $(krp, krj) \in F$. An arc (krp, krj) is active (i.e. enforces $h_{krj} - h_{krp} \geq w^{F}_{krp\_krj}$) when the route r is chosen for aircraft k (i.e. $y_{kr} = 1$). Fixed directed arcs are used to model the release arcs in F_{rt}, the deadline arcs in F_{Dt} and the due date arcs in F_{dt}, as shown respectively in Constraints (8.4), Constraints (8.5) and Constraints (8.6), together with the holding arcs in F_{HC}, the air segment arcs in F_{AS} and the runway arcs in F_{RW}.

Constraints (8.7) model the alternative pairs $((kro, uim), (uig, krl)) \in A$. These alternative pairs model sequencing decisions between two aircraft at air segments A_{AS} and the sequencing decisions at runways A_{RW}, thus the arcs of each pair connect two operations of the different job. A pair $((kro, uim), (uig, krl))$ is active when both the following conditions hold: the route r is chosen for aircraft k (i.e. $y_{kr} = 1$) and the route i is chosen for aircraft u (i.e. $y_{ui} = 1$). The alternative arc (kro, uim) is active (i.e. enforces $h_{uim} - h_{kro} \geq w^{A}_{kro\_uim}$) when $x^{uig\_krl}_{kro\_uim} = 0$ while the alternative arc (uig, krl) is active (i.e. enforces $h_{krl} - h_{uig} \geq w^{A}_{uig\_kro}$) when $x^{uig\_krl}_{kro\_uim} = 1$.

Constraints (8.8) model the alternative pairs $((krp, krj), (krj, krp)) \in A_{HC}$. These alternative pairs model holding decisions of a landing aircraft. Differently than in the previous case, this implies that the arcs of each pair connect two operations of the same job. A pair $((krp, krj), (krj, krp))$ is active when the route r is chosen for aircraft k (i.e. $y_{kr} = 1$). When the pair is active, only one of its arc must be active in a solution. The activation of one arc for each pair is modelled as for Constraints (8.7). Selecting which alternative arc

is active in each alternative pair determines the number of circles performed during the holding procedure by each aircraft.

Constraints (8.9), (8.10) and (8.11) set x and y as binary variables.

8.3 Scheduling and re-routing algorithms

This section describes the algorithmic approaches proposed in this paper to solve the ATC-TCA problem. Section 8.3.1 presents the general framework of the solver that is based on a combination of aircraft scheduling and re-routing algorithms, Section 8.3.2 the neighbourhoods for the search of new aircraft routes starting from a routing and scheduling solution, Section 8.3.3 the scheduling heuristic procedure used to evaluate the neighbours (the new routing combinations). The routing neighbourhoods and the scheduling algorithms are used in Sections 8.3.4, 8.3.5, 8.3.6 that describe the metaheuristics developed and tested in this paper.

8.3.1 Solution framework

Figure 8.1 illustrates the general scheme of the solver. Since the ATC-TCA problem is an NP-hard problem, we adopt a temporal decomposition and a decomposition in routing and scheduling variables. The former is solved via the rolling horizon procedure in [Samà et al., (2013a)], while the latter is solved via the scheduling and re-routing algorithms of the AGLIBRARY solver. Specifically, we use the scheduling algorithms in [D'Ariano et al., (2015), D'Ariano et al., (2010)], the re-routing algorithms in [D'Ariano et al., (2012b)], the new scheduling and re-routing algorithms developed in this paper. The two decomposition frameworks can be further combined together.

The rolling horizon decomposition framework divides the ATC-TCA problem into time horizons of traffic predictions. Each time horizon is a sub-problem instance to be solved by the AGLIBRARY solver. We assume that all aircraft information is known at the start time t_0 of the traffic prediction. This rolling horizon framework corresponds to a centralized framework when the overall problem is solved with a single time horizon (i.e. when no temporal decomposition is performed).

The decomposition framework into routing and scheduling works instead as follows. The AGLIBRARY solver iterates between the computation of a new aircraft schedule for a set of routes, and the selection of a new set of routes. The basic idea is to first compute an aircraft scheduling solution given fixed (default) routes, and then search for better aircraft routes. The latter procedure is based on a local search for routing alternatives starting from the scheduling solution, and an iterative scheduling and re-routing technique to continue the search. The iterative procedure returns the best aircraft schedule and the best set of routes after a stopping criteria is reached. In this paper, the maximum computation time is a stopping criteria.

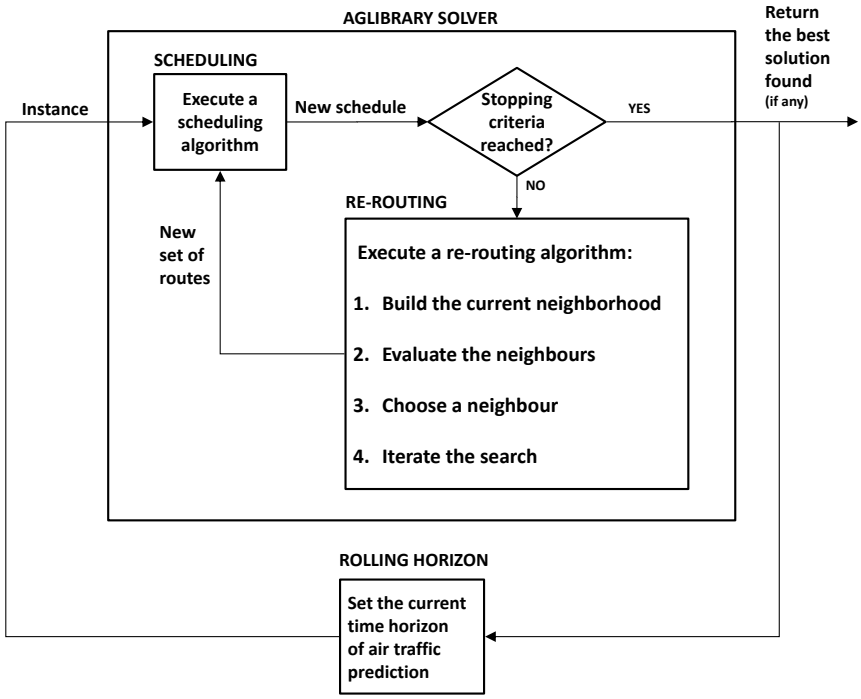


Figure 8.1: A general scheme of the solver

The overall framework returns a feasible aircraft schedule in which a route is fixed for each aircraft and all potential routing conflicts are solved. In case no feasible schedule is computed, the solver reports the conflicting routes via a detailed time-space diagram. Based on the information provided by the solver, the en-route/ground human traffic controllers could take suitable re-scheduling actions on the potential conflicts that are not allowed by the automated decision support system, including re-routing some aircraft to other resources in the same or other airports.

8.3.2 Routing neighbourhoods

This subsection describes the neighbourhood structures used in this paper. To this aim, we need to introduce the following notation. Let $S(F)$ be a ATC-TCA solution with the routes defined in F and the sequencing decisions defined in S, and let $\mathcal{G}(F, S)$ be the graph of this solution. The search for a better solution is based on the computation of a new graph $\mathcal{G}'(F', S')$ This graph differs from the former $\mathcal{G}(F, S)$ by a different route for some aircraft, and different orders and times of operations. This corresponds to a neighbour, in metaheuristics terms. The longest path in $\mathcal{G}'(F', S')$ is denoted as $l^{S'(F')}(s, t)$. We observe that F' improves over F in terms of the objective function value if $l^{S'(F')}(s, t) < l^{S(F)}(s, t)$.

The neighbourhoods studied in this paper are based on observations on the graph $\mathcal{G}(F,S)$ regarding the nodes that represent operations delayed due to the resolution of potential aircraft conflicts. These nodes are *critical* when they are on the longest path from the start node s to the end node t in $\mathcal{G}(F,S)$, that is called the *critical path set* $\mathcal{C}(F,S)$. Given a solution $S(F)$, $krp \in N(F) \setminus \{s,t\}$ is a *critical node* of aircraft k with route r if $l^{S(F)}(s,krp) + l^{S(F)}(krp,t) = l^{S(F)}(s,t)$. A critical node krp is a *waiting node* if $l^{S(F)}(s,krp) > l^{S(F)}(s,\nu(krp)) + w^F_{\nu(krp),krp}$, where the node $\nu(krp)$ precedes the node krp on route r. For each waiting node krp, there is at least one *hindering node* $\eta(krp)$ in $\mathcal{G}(F,S)$, different from node $\nu(krp)$, such that $l^{S(F)}(s,krp) = l^{S(F)}(s,\eta(krp)) + w^F_{\eta(krp),krp}$.

Given a node $krp \in N(F) \setminus \{s,t\}$, we recursively define the *backward ramification* $R_B(krp)$ as follows. If node krp is waiting, then $R_B(krp) = R_B(\nu(krp)) \cup R_B(\eta(krp) \cup \{krp\}$, otherwise $R_B(krp) = R_B(\nu(krp)) \cup \{krp\}$. Similarly, we recursively define the *forward ramification* $R_F(krp)$ as follows. If node krp is the hindering of a waiting node dij, then $R_F(krp) = R_F(\sigma(krp)) \cup R_F(dij) \cup \{krp\}$, where the node $\sigma(krp)$ follows the node krp on route r. Otherwise, $R_F(krp) = R_F(\sigma(krp)) \cup \{krp\}$. By definition, $R_B(s) = R_F(s) = \{s\}$ and $R_B(t) = R_F(t) = \{t\}$. Given $\mathcal{C}(F,S)$, we define a *ramified critical path* set as $\mathcal{F}(F,S) = \bigcup_{krp \in \mathcal{C}(F,S)}[R_B(krp) \cup R_F(krp)]$, and a *backward ramified critical path* set as $\mathcal{B}(F,S) = \bigcup_{krp \in \mathcal{C}(F,S)}[R_B(krp)]$. We study the five neighbourhood structures listed below.

- *Complete K-Route neighbourhood* $\mathcal{N}_{CKR}$: contains all the feasible solutions to the ATC-TCA problem in which K aircraft follows a different route compared to the incumbent solution. To limit the number of neighbours to be evaluated, $\mathcal{N}_{CKR}$ is only partially explored as follows. A move is obtained by choosing K routes different from the ones of the current solution at random (i.e. all alternative routes having the same probability), until a number ψ (parameter) of alternative routing solutions is obtained:

- *Ramified Critical Path Operations neighbourhood* $\mathcal{N}_{RCPO}$ considers only the routing alternatives for the aircraft associated to the nodes in $\mathcal{B}(F,S)$ plus $\mathcal{F}(F,S)$. The idea is that the maximum consecutive delay of an optimal solution to the ATC-TCA problem can be reduced by removing aircraft conflicts causing it. This requires either removing, anticipating or postponing some operations from the critical path set (i.e. re-routing the aircraft associated to the critical path on the graph $\mathcal{G}(F,S)$ of the incumbent solution). The latter result can be obtained by re-routing some aircraft represented by jobs with nodes in $\mathcal{B}(F,S)$ or $\mathcal{F}(F,S)$ and then re-scheduling aircraft movements;

- *Waiting Operations Critical Path neighbourhood* $\mathcal{N}_{WOCP}$ is a restriction of $\mathcal{N}_{RCPO}$ that considers the routing alternatives for the aircraft associated to the waiting nodes in $\mathcal{C}(F,S)$;

- *Delayed Jobs neighbourhood* $\mathcal{N}_{DJ}$ considers only the aircraft (jobs) that have a consecutive delay on some due date arcs in the graph $\mathcal{G}(F,S)$ of the incumbent solution;

- *Free-Net Waiting Operations Jobs neighbourhood* $\mathcal{N}_{FNWJ}$ considers only the aircraft (jobs) that have some waiting nodes in the alternative graph $G(N, F, A)$ of the incumbent solution. A waiting node is identified by computing the consecutive delay that would be associated in $G(N, F, A)$ by selecting an alternative arc (i.e. the one generating the waiting node) and by disregarding all the other arcs in A.

The appendix will illustrate some neighbourhood structures for an illustrative example.

8.3.3 Heuristic evaluation of routing neighbours

The choice of a best neighbour in the neighbourhood requires the computation of a new ATC-TCA solution $S'(F')$ starting for an incumbent solution $S(F)$, that is characterized by the routing decisions in F' and the sequencing decisions in S'. To this aim, we use fast heuristics based on a two-step graph building procedure in which the graph $\mathcal{G}(F, S)$ is translated into the graph $\mathcal{G}'(F', S')$. In the first step, a sub-graph of $\mathcal{G}'(F', S')$ is generated by considering all the nodes $\in N(F^I)$ associated to the routes modelled by the arcs $\in F^I = F \bigcap F'$, all the fixed directed arcs $\in F^I$ and all the alternative arcs in $S(F)$ incident in a node $\in N(F^I)$. This corresponds to keeping a subset of decisions from the incumbent solution into the neighbour solution. In the second step, the fixed directed arcs $\in F^R = F' \setminus F^I$ and the nodes $\in N(F^R)$ are added to the sub-graph. Finally, $\mathcal{G}'(F', S')$ is obtained by adding a selection of alternative arcs $S'(F^R)$ to the sub-graph. The selection $S'(F^R)$ is computed by taking the best solution among two greedy algorithms (i.e. the AMSP and AMCC algorithms) described in D'Ariano et al. [D'Ariano et al., (2015)].

8.3.4 Tabu search re-routing algorithm

The *Tabu Search* (TS) is a deterministic metaheuristic based on local search, which makes extensive use of memory for guiding the search [Glover (1986)]. A basic ingredient is the *tabu list*, that is used to avoid being trapped in local optima and revisiting the same solution. From the incumbent solution, non-tabu moves define a set of solutions, named the *incumbent solution neighbourhood*. At each step, the best solution in this set is chosen as the new incumbent solution. Some attributes of the former incumbent are then stored in the tabu list. The moves in the tabu list are forbidden as long as these are in the list, unless an aspiration criterion is satisfied. The tabu list length can remain constant or be dynamically modified during the search.

The Tabu Search (TS) of D'Ariano et al. [D'Ariano et al., (2012b)] is used in the iterative scheduling and re-routing framework. The neighbourhood strategy used by TS explores candidate solutions in $\mathcal{N}_{RCPO}$ unless this neighbourhood is empty. In the latter case, ψ consecutive moves are performed in $\mathcal{N}_{CKR}$ with $K = 1$ before searching again in $\mathcal{N}_{RCPO}$. All neighbours are evaluated via the scheduling heuristics of Section 8.3.3. The best neighbour is set as the move to be made, and evaluated via the branch-and-bound algorithm

in [D'Ariano et al., (2015), D'Ariano et al., (2010)]; the resulting best solution is set as the new incumbent solution. The inverse of the chosen move is stored in a tabu list of length λ (parameter). The moves in the tabu list are forbidden for λ iterations and no aspiration criteria is used. When no potentially better solution is found on the incumbent solution neighbourhood, the search alternates the above neighbourhood strategy with a diversification strategy, which consists of changing at random the route of μ (parameter) aircraft at the same time. From the tuning performed in [D'Ariano et al., (2012b)], the best overall exploration strategy has the following parameters $\psi = 10$, $\lambda = 32$ and $\mu = 5$.

8.3.5 Variable neighbourhood search re-routing algorithm

A *Variable Neighbourhood Search* (VNS) is proposed to efficiently solve the ATC-TCA problem. This metaheuristic is based on the combination of different neighbourhoods. Neighbourhood changes are proposed both in a local search phase in order to compute a local minimum, and in a perturbation phase in order to escape from a local minimum [Hansen et al., (2008)]. The intuition behind the choice of this search method for solving the ATC-TCA problem is as follows. The metaheuristic search starts from a reference ATC-TCA solution with fixed routes and needs to explore various possibilities to generate new solutions in the local search phase. Multiple aircraft are simultaneously re-routed and new aircraft scheduling solutions are computed in order to evaluate a different load of the TCA resources and to investigate the sequence flexibility when solving the potential aircraft conflicts. The choice of re-routing multiple simultaneous aircraft differs from the local search used in TS, in which a new solution is generated by changing the route of a single aircraft and by considering the aircraft on the ramified critical path only. The proposed VNS makes use of neighbourhoods based on different candidate aircraft and routing alternatives in the runway and/or air segment resources.

The VNS algorithm of Figure 8.2 is an adaptation of the basic VNS described in Hansen et al. [Hansen et al., (2008)]. This algorithm combines the classic ingredients of the VNS algorithm with the routing neighbourhood structures of Section 8.3.2 and new sophisticated neighbourhood search strategies to search for better aircraft routes.

The general structure of the VNS is the following. The algorithm starts from an incumbent solution of the ATC-TCA problem, named $IncSol$ $\mathcal{G}(F, S)$, computed via the branch-and-bound algorithm in [D'Ariano et al., (2015), D'Ariano et al., (2010)] given a default (off-line) route to each aircraft. A counter K is adopted to fix the number of aircraft that are re-routed in each move. The initial value of K is set to 1, i.e. a single aircraft is re-routed in $IncSol$ $\mathcal{G}(F, S)$. The metaheuristic iterates the search for better solutions starting from $IncSol$ until a maximum computation time T_{max} is reached or until the maximum consecutive delay is larger than 0. At each iteration, a neighbourhood of $IncSol$ is generated and a new solution $IncSol'$ is

selected via a shaking procedure. The search continues with the generation of a neighbourhood of $IncSol'$ and a local search based on best-improvement is performed in a restricted neighbourhood. Then, a *Move Or Not* function is performed as follows. In case an improving move $IncSol''$ is obtained via the local search (i.e. $f(IncSol'') < f(IncSol)$), a new iteration is performed by setting $IncSol''$ as the new incumbent solution and K is set to 1. Otherwise, the parameter K is set to $K + 1$ and a new iteration is performed until $K \leq K_{max}$. When $K = K_{max}$ the algorithm diversifies the search with a change of neighbourhood structure (if the algorithm works with a single neighbourhood structure this step is not performed). The metaheuristic returns the best ATC-TCA solution ($IncSol$) and the objective function value ($f(IncSol)$). The pseudo-code of the VNS is reported in Figure 8.2. We next describe the specific features of the VNS.

Build Neighbourhood. Starting from an incumbent solution, the $\mathcal{N}_{CKR}$ neighbourhood is generated, in which exactly K aircraft are re-routed in the graph $\mathcal{G}(F, S)$ of the incumbent solution.

Shake. This is a typical diversification procedure in metaheuristics that consists in changing the route of K aircraft randomly in the $\mathcal{N}_{CKR}$ neighbourhood of the incumbent solution ($IncSol$), and in computing a new incumbent solution ($IncSol'$) via the scheduling heuristics of Section 8.3.3 and the new set of routes.

Neighbourhood Search Strategy. This procedure is proposed in order to reduce the local search to the evaluation of up to L neighbours in the current neighbourhood. Starting from an incumbent solution and the $\mathcal{N}_{CKR}$ neighbourhood of this solution, a restricted neighbourhood is generated by using a given neighbourhood structure $\mathcal{N}_i$. The selection of L neighbours is achieved in the following steps:

1. *aircraft ranking* : Each aircraft gets a score based on the criterion specified in a neighbourhood structure $\mathcal{N}$. The score is used to decide how many times the aircraft has to be re-routed in the L neighbours; This ranking is based on one of the neighbourhood structures of Section 8.3.2. In $\mathcal{N}_{RCPO}$, each aircraft gets a score based on the maximum value $l^{S(F)}(s, krp) + l^{S(F)}(krp, t)\ \forall\ (krp)$ in the ramified critical path of the graph $\mathcal{G}(F, S)$ of the incumbent solution. In $\mathcal{N}_{WOCP}$, each aircraft gets a score based on the sum of the consecutive delays associated at each critical node in the graph $\mathcal{G}(F, S)$. In $\mathcal{N}_{DJ}$, each aircraft gets a score based on the maximum consecutive delay associated at its due date arcs of the graph $\mathcal{G}(F, S)$. In $\mathcal{N}_{FNWJ}$, each aircraft gets a score based on the sum of the consecutive delays associated at a restricted set of waiting nodes in the graph $G(N, F, A)$ of the incumbent solution. We only consider the alternative pairs in which both the alternative arcs generate a consecutive delay. For those alternative pairs, we take the alternative arc generating the largest consecutive delay in $G(N, F, A)$ and the corresponding waiting node.

2. *route ranking* : The routes of each aircraft get a score based on the distance from the route of the incumbent solution. The larger is the difference between the routes, the higher is the score. The route ranking thus suggests for each aircraft to select the most different routes. Among the selected routes,

the ranking gives precedence to the routing alternatives in which there is a change of runway, since this is often the conflicting resource of the TCA aircraft routes;

3. *neighbour generation* : This is the assignment of the routes to the aircraft in each neighbour. This is done by selecting the aircraft based on the aircraft ranking and by selecting the routes based on the route ranking. A combinatorial combination of the routes is used in order to generate L different neighbours. In each neighbour, exactly K aircraft are re-routed compared to the incumbent solution. The neighbours are ordered based on the aircraft ranking, and in case of tie on the route ranking.

Algorithm VNS
Input: $IncSol$ $\mathcal{G}(F,S)$, K_{max}, T_{max}, L, $\mathcal{N}_1$, $\mathcal{N}_2$
$\mathcal{N}_i \leftarrow \mathcal{N}_1$,
While $(T < T_{max})$ & $(f(IncSol) > 0)$ **do**
  Begin
  $K \leftarrow 1$,
  While $(K \leq K_{max})$ **do**
    Begin
    $BuildNeighbourhood(IncSol, K)$,
    $IncSol' \leftarrow Shake(IncSol, K)$,
    $BuildNeighbourhood(IncSol', K)$,
    $NeighbourhoodSearchStrategy(IncSol', K, L, \mathcal{N}_i)$,
    $IncSol'' \leftarrow FirstImprovement(IncSol')$,
    $(IncSol, K) \leftarrow MoveOrNot(IncSol, IncSol'', K)$,
    If $(K = K_{max})$ **do**
      Begin
      $\mathcal{N}_i \leftarrow NeighbourhoodChange(IncSol, \mathcal{N}_i, \mathcal{N}_1, \mathcal{N}_2)$,
      End
    $T \leftarrow$ CPU time()
    End
  End

Figure 8.2: Sketch of the VNS algorithm

Figure 8.3 presents a numerical example of the neighbourhood search strategy, in which four aircraft ($J = \{J1, J2, J3, J4\}$) can be re-routed in a TCA. More details are reported in the paper appendix. J1 and J4 have four alternative routes (e.g., the routes of $J1$ are: $J1$-1, $J1$-2, $J1$-3, $J1$-4), while J2 and J3 have two alternative routes each. In the incumbent solution, all aircraft use the first route ($J1$-1, $J2$-1, $J3$-1, $J4$-1). The parameters of the procedure are set to the following values: $L = 4$ and $K = 2$ (i.e. the neighbourhood is restricted to 4 neighbours and 2 aircraft are re-routed in each neighbour).

The aircraft ranking determines a score matrix in which each row represents an aircraft and each column the number of times each aircraft can be re-routed. This score matrix is depicted in the left-hand side of Figure 8.3. Specifically, the first column reports the score for each aircraft based on the neighbourhood structure $\mathcal{N}_{DJ}$ (e.g., the value 40 in this example is the maximum consecutive delay associated at aircraft

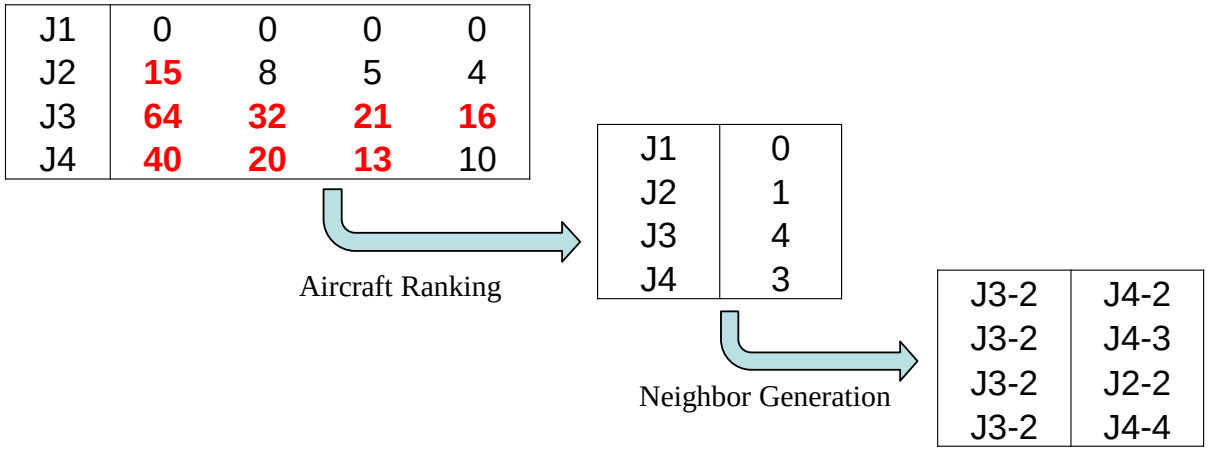


Figure 8.3: Example of neighbourhood search strategy with $|J| = 4$, $L = 4$ and $K = 2$

$J4$, see Appendix). The other columns report the score of the first column divided by the column number, e.g., $40/2 = 20$, $40/3 = 13$, $40/4 = 10$. By taking the highest KL scores in the score matrix (these values are reported in bold in Figure 8.3), we get the number of times each aircraft has to be re-routed in the four neighbours (e.g., $J4$ is re-routed in three neighbours). This is reported in the center table of Figure 8.3.

The route ranking orders the list of re-routing alternative of each aircraft based on maximizing the difference with the incumbent route and giving precedence to the routing alternatives in which there is a change of runway. In this case, the route ranking of $J4$ is $J4$-2, $J4$-3, $J4$-4 and the most different route with a change of runway is $J4$-2.

The neighbour generation procedure assigns the routes to the aircraft in each neighbour. In this example, $J4$ appears in the neighbours with its three different routes, while $J2$ and $J3$ have only a route to be chosen. Every row of the table in right-hand side of Figure 8.3 corresponds to a candidate move.

First Improvement. This is a local search procedure in the restricted neighbourhood in which the candidate moves are ordered based on the neighbour generation procedure. At each step of the procedure, the neighbour with the highest ranking in the restricted neighbourhood is considered, a new graph is build with the new routes of the neighbour, and a new solution is computed via the scheduling heuristics of Section 8.3.3 for the new set of routes. This procedure lasts until a better solution is obtained compared to the incumbent solution or until all L neighbours in the restricted neighbourhood have been evaluated.

Move Or Not. This procedure is responsible for possibly making a move. In case the best solution found in the neighbourhood is better than the incumbent, the resulting graph is solved by the branch-and-bound algorithm in [D'Ariano et al., (2015), D'Ariano et al., (2010)], and the best solution is set as the

new incumbent solution. Otherwise, the best solution in the neighbourhood is chosen as incumbent, or some diversification strategy is employed.

Neighbourhood Change. This procedure is used to diversify the search by alternating K iterations of the neighbourhood search strategy with $\mathcal{N}_1$ and K iterations of the neighbourhood search strategy with $\mathcal{N}_2$.

8.3.6 Hybrid search re-routing algorithm

We introduce a hybrid metaheuristic, named Variable Neighbourhood Tabu Search (VNTS), for solving the ATC-TCA problem. This algorithm consists of a combination of VNS and TS, as proposed earlier in other contexts by Moreno Pérez et al. [Moreno Pérez et al., (2003)]. The hybridization is proposed in order to take promising ideas from both the metaheuristics. The VNS is used in order to explore different neighbourhoods of the incumbent solution, such that the reference aircraft scheduling solution can be improved in terms of multiple routing modifications. The TS is used to avoid cycling and is combined with aircraft re-routing strategies to escape from local minimum. Specifically, the ingredients of the proposed hybrid metaheuristic are an intensification strategy based on the exact exploration of a restricted aircraft re-routing neighbourhood, and various diversification strategies for aircraft re-routing in different runway and air segment resources and on a restart technique based on a list of potentially useful routing alternatives for each aircraft.

Figure 8.4 introduces the pseudo-code of the hybrid metaheuristic. Starting from an incumbent solution *IncSol* $\mathcal{G}(F,S)$, a given neighbourhood structure $\mathcal{N}_i$ and a counter Q (initialized as 0), the metaheuristic iterates the search for better solutions in various neighbourhoods until a maximum computation time T_{max} is reached, as far as the maximum consecutive delay is larger than 0. Each iteration evaluates all the neighbours in a restricted neighbourhood of *IncSol* (i.e. L non-tabu neighbours, determined via a neighbourhood search strategy, analogously to the VNS procedure) and the best neighbour is implemented via the branch-and-bound algorithm in [D'Ariano et al., (2015),D'Ariano et al., (2010)]. The *Move Or Not* function is performed as for the VNS algorithm of Figure 8.2. However, when $K = K_{max}$ the algorithm diversifies the search as follows. If $Q < Q_{max}$ a change of neighbourhood structure is implemented, and Q is set to $Q+1$; otherwise a restart strategy based on a problem-specific memory structure is applied and Q is set to 0. Before a new iteration is performed, a tabu search memory and a time counter are updated. We next describe key VNTS features in more detail.

Best Improvement. Given a restricted neighbourhood of an incumbent solution, this procedure evaluates all neighbours via the scheduling heuristics of Section 8.3.3. The best neighbour is set as the move to be made, and evaluated via the branch-and-bound algorithm in [D'Ariano et al., (2010), D'Ariano et al., (2015)]; the resulting best solution is set as the new incumbent solution.

Algorithm VNTS
Input: $IncSol\ \mathcal{G}(F,S)$, K_{max}, T_{max}, L, TL_{max}, Q_{max}, $\mathcal{N}_1$, $\mathcal{N}_2$
$Q \leftarrow 0$,
$\mathcal{N}_i \leftarrow \mathcal{N}_1$,
While $(T < T_{max})$ & $(f(IncSol) > 0)$ **do**
  Begin
  $K \leftarrow 1$,
  While $(K \leq K_{max})$ **do**
    Begin
    $BuildNeighbourhood(IncSol,\ K)$,
    $TabuMemoryFilter(TL)$,
    $NeighbourhoodSearchStrategy(IncSol,\ K,\ L,\ \mathcal{N}_i)$,
    $IncSol' \leftarrow BestImprovement(IncSol)$,
    $(IncSol,\ K) \leftarrow MoveOrNot(IncSol,\ IncSol',\ K)$,
    If $(K = K_{max})$ **do**
      Begin
      If $(Q < Q_{max})$ **do**
        Begin
        $\mathcal{N}_i \leftarrow NeighbourhoodChange(IncSol,\ \mathcal{N}_i,\ \mathcal{N}_1,\ \mathcal{N}_2)$,
        $Q \leftarrow Q + 1$,
        End
      Else
        Begin
        $RestartStrategy(Q)$,
        $Q \leftarrow 0$,
        End
      End
    $TL \leftarrow TabuMemoryUpdate(TL,\ TL_{max},\ Q,\ Q_{max})$,
    $T \leftarrow$ CPU time()
    End
  End

Figure 8.4: Sketch of the VNTS algorithm

Neighbourhood Change. This procedure is used to diversify the search by alternating K iterations of the neighbourhood search strategy with $\mathcal{N}_1$ with K iterations of the neighbourhood search strategy with $\mathcal{N}_2$. In particular, when $Q < Q_{max}$, $K = K_{max}$ and the current neighbourhood structure $\mathcal{N}_i = \mathcal{N}_1$, the procedure sets $\mathcal{N}_i = \mathcal{N}_2$ and the search continues with K neighbourhood search strategy iterations with $\mathcal{N}_2$.

Restart Strategy. This is a typical diversification strategy that we implemented as follows. A restart memory structure is used to store the number of times each aircraft re-routing has been evaluated in any neighbour so far. From this structure, we compute an ordered list of routing alternatives for each aircraft from the less frequently used to the most frequently used. The restart strategy consists on the implementation of a move (disregarding the tabu memory) in which the less frequently used route is set for each aircraft. In case of tie, a random decision is taken for each aircraft among its routes being used the least. The restart memory structure is reset each time the restart strategy is used.

Tabu Memory. This technique is used to avoid cycling and revisiting the same solution. This is achieved by two functions named *tabu memory filter* and *tabu memory update*. The former function removes from the current neighbourhood all the tabu moves (no aspiration criteria is used), while the latter function updates the list of tabu moves during the search. The tabu list (TL) used in the VNTS is made by up to TL_{max} moves. For each move that has been implemented in a local search step, the following key information is stored: the re-routed aircraft of the move, and the old route chosen for each re-routed aircraft. For example, we have a problem with three aircraft and three routes for each aircraft. An incumbent solution is $J1-1, J2-1, J3-1$ and the move performed by a local search is $J1-1, J2-2, J2-3$. The tabu memory update will store in the tabu list the following information: $J2$ and $J3$ have been re-routed and the tabu-routes are $J2-1$ and $J3-1$. A move is tabu when the two routes $J2-1$ and $J3-1$ are chosen as the new routes of $J2$ and $J3$. The tabu list is reset when a restart move is performed (i.e. when $Q = Q_{max}$).

8.4 Computational experiments

This section presents the experimental assessment of the various metaheuristics of Section 8.3. The test bed is the Milan Malpensa terminal control area (MXP) and the instances are taken from Samà et al. [Samà et al., (2014)]. In particular, we evaluate the metaheuristics on the most complex instances with strong disturbances and a large number of aircraft. The objective of this evaluation is to report the marginal improvement achieved by the new metaheuristics compared with the approaches in [Samà et al., (2014)]. To this end, the experiments are executed on the same processor used in [Samà et al., (2014)], that is Intel Core 2 Duo E6550 (2.33 GHz), 2 GB of RAM, Windows XP. For all the metaheuristics tested in this paper, we adopt the deadline implications and the pre-processing procedure developed in [Samà et al., (2014)]. For all the approaches based on the rolling horizon framework, we use the same parameter setting in [Samà et al., (2014)] (i.e. we fix the roll period to 10 min and the look-ahead period to 15 min).

8.4.1 Description of the TCA

Figure 8.5 shows the TCA scheme of Malpensa (MXP) TCA. There are two runways (RWY 35L, RWY 35R), used both for departing and arriving procedures. The MXP resources are: three airborne holding circles (resources 1-3 in Figure 8.5, named TOR [Torino], MBR [Mebur], SRN [Saronno]), eleven air segments for arriving procedures (resources 4-14), a common glide path (resource 15) and two runways (resources 16-17). The common glide path resource includes two parallel air segments before the runways for which traffic regulations impose a minimum diagonal/longitudinal distance between landing aircraft.

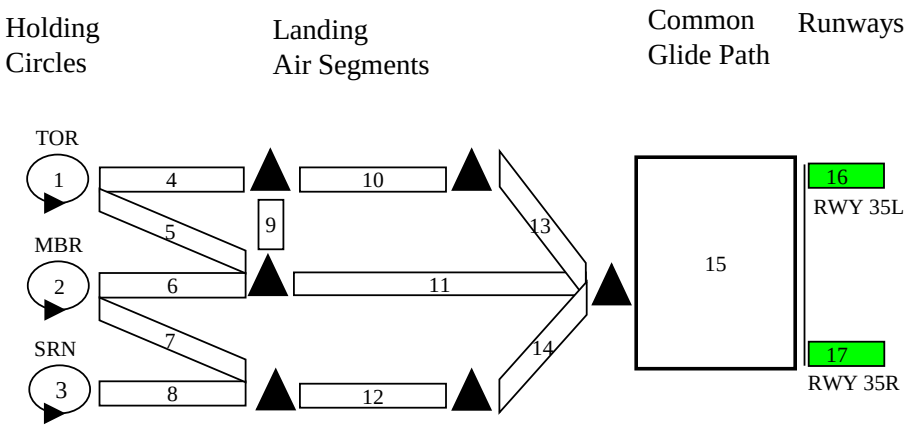


Figure 8.5: Malpensa (MXP) Terminal Control Area

8.4.2 Tested instances

The ATC-TCA instances are based on real data collected for the Milan Malpensa TCA. We consider a subset of the instances generated in [Samà et al., (2014)], that are obtained by varying the following parameters: (i) the model variant, (ii) the time horizon of traffic prediction, (iii) the aircraft delays, (iv) the disruption.

Model variant. Three variants are investigated to model objective functions and user requirements. Model 1 (M1) measures the delay of landing and take-off aircraft at the runways, that are the most used TCA resources. Model 2 (M2) modifies the objective function for landing aircraft by measuring their delay both at the runways and at the entrance of the TCA, penalizing a late entrance in the TCA. The latter model takes into consideration the extra work required to coordinate the solutions of TCA and en-route traffic controllers. Model 3 (M3) extends M2 with additional deadline constraints that limit the maximum possible entrance time of each landing aircraft in the TCA. These constraints are inserted in order to consider the limited possibility of airborne holding, being more expensive and constrained than ground holding.

Time horizon of traffic prediction. Two time horizons of different length are considered: 60 and 180 minutes. The shorter time horizon can be considered of practical interest for the management of light disturbances, while the longer time horizon can better assess the traffic control measures in terms of aircraft delay propagation. The latter time horizon is particularly relevant to study in case of disruptions.

Aircraft delays. Disturbed traffic conditions are generated by delaying the entrance time of some aircraft in the TCA. The entrance delays are randomly generated according to a uniform distribution, and are applied at some aircraft entering the TCA during the first half of the time horizon under examination. For each time horizon of traffic prediction, we consider 10 delay instances in which random delays are up to

5 minutes and other 10 delay instances in which random delays are up to 15 minutes.

Disruption. We consider the case in which one of the two runways of the TCA is unavailable in a time window, and all landing and take-off aircraft have to be scheduled in the only available runway during the disruption. The disrupted case is only studied for the 180-minute traffic predictions with a disrupted runway between the first and the second hour of traffic prediction.

In total, there are 3 groups of instances: two groups regard undisrupted operations with different time horizons (60 and 180 minutes), for a total of 120 delayed instances (i.e. for the 3 model variants, the 2 time horizons, and the 20 aircraft delays); one group of 60 disrupted instances (i.e. the 3 model variants, the 180-minute time horizon, and the 20 aircraft delays).

Table 8.1: Size of the MILP formulation for the various ATC-TCA instances

Time Horizon	Model Variant	Num of Fixed Constraints	Num of Alternative Constraints	MILP Variables			Num of Aircraft
				h	x	y	
60 min	M1	728	11526	264	5763	62	40
	M2	751	11526	264	5763	62	40
	M3	774	11526	264	5763	62	40
180 min	M1	4331	456476	871	228238	414	117
	M2	4649	456476	871	228238	414	117
	M3	4967	456476	871	228238	414	117

Table 8.1 gives average information on the aircraft delay instances regarding the MILP formulation; every row is an average over the 20 delay instances. Moreover, Column 1 reports the time horizon of traffic prediction, Column 2 the model variant, Column 3 the fixed constraints, Column 4 the alternative constraints, Column 5–7 the MILP variables, Column 8 the number of aircraft. The model variants differ in terms of the fixed constraints, since different due date and deadline arcs are used. Instead, disrupted instances have the same characteristics (concerning the variables in Table 8.1) than the corresponding undisrupted instances.

8.4.3 Assessment of the VNS and VNTS parameters

This section discusses the choice of alternative configurations for the VNS and VNTS algorithms, while the TS algorithm configurations were evaluated in previous works [D'Ariano et al., (2012b),Samà et al., (2014)]. The computational assessment is based on 20 pilot ATC-TCA instances with aircraft delays. Disrupted instances have not been considered since these present a reduced number of alternative routes. The assessment is based on the evaluation of the metaheuristics in the centralized framework with $T_{max} = 180$ seconds.

Table 8.2 shows the parameters considered in the algorithm assessment. The parameters used in both algorithms are the time given to the branch-and-bound algorithm (BB Time, in seconds), the number of aircraft that are rerouted in the current neighbourhood (K_{max}), the size of the restricted neighbourhood

Table 8.2: Experimental setting of algorithmic parameters

T_{max} (sec)	BB Time (sec)	K_{max}	L	TL_{max}	Q_{max}
180	4/**10**/20/30	2/**3**/4/5	5/**10**/20	0/8/15/**32**	2/**3**/4

$\mathcal{N}$	$\mathcal{N}_1 + \mathcal{N}_2$
DJ / RCPO / WOCP / FNWJ	**WOCP+FNWJ** / FNWJ+WOCP / DJ+WOCP / WOCP+DJ

(L), and the choice of the neighbourhood structure ($\mathcal{N}_1$ + $\mathcal{N}_2$ or $\mathcal{N}$ when the two coincide). Additional parameters of the VNTS are the tabu list length (TL_{max}) and the counter used for the diversification strategies (Q_{max}). The best value of each parameter is reported in bold and used in all the experiments.

8.4.4 Assessment of the solution quality

This section presents the results obtained for the best metaheuristics developed in this paper and compares them with the results obtained in Samà et al. [Samà et al., (2014)], that is used as a benchmark comparison for the newly developed algorithms. Table 8.3 gives an overview of the best ALGOrithm in [Samà et al., (2014)] (ALGO, Column 4), the best CEntralized MetaHeuristic (CE MH, Column 5) and the best Rolling Horizon MetaHeuristic (RH MH, Column 6) for the instances grouped by the time horizon of traffic prediction (Column 1), the type of disturbance (Column 2), the model variant (Column 3). Each row reports the best algorithm on the set of 20 ATC-TCA instances described in Section 8.4.2.

We now recall the different acronyms used: MILP refers to the Mixed-Integer Linear Programming formulation of the ATC-TCA problem solved by using the IBM ILOG CPLEX MIP 12.0 solver; VNS, VNTS and TS refer to the metaheuristics Variable Neighbourhood Search, Variable Neighbourhood Tabu Search and Tabu Search; DJ and WOCP+FNWJ refer to the restricted neighbourhood Delayed Job and the combined restricted neighbourhoods Waiting Operation Critical Path and Free-Net Waiting Operations. We next give detailed information on the performance of the various best algorithms reported in Table 8.3.

Table 8.3: Best algorithms for various groups of instances

Time Horizon	Disturbance Type	Model Variant	Best Algo [Samà et al., (2014)]	Best CE MH	Best RH MH
60 min	Normal	M1	RH MILP	VNTS DJ	VNS DJ
		M2	CE MILP	VNTS DJ	VNS DJ
		M3	CE MILP	VNTS DJ	VNS DJ
180 min	Normal	M1	RH MILP	VNTS DJ	VNS DJ
		M2	RH MILP	VNTS DJ	VNS DJ
		M3	RH MILP	VNTS DJ	VNS DJ
180 min	Disrupted	M1	RH TS	VNS WOCP+FNWJ	VNS DJ
		M2	RH TS	VNS WOCP+FNWJ	VNS DJ
		M3	RH TS	VNS WOCP+FNWJ	VNS DJ

Regarding CE MILP, RH MILP and RH TS, we use the same setting of the parameters and thus the

same results presented in Samà et al. [Samà et al., (2014)]. We recall that their time limit of computation is 240 (720) seconds for the 60-minute (180-minute) instances.

Regarding the metaheuristics developed in this paper, the parameters are set as described in Section 8.4.3. The best metaheuristics have been taken among VNS WOCP+FNWJ, VNS DJ, VNTS WOCP+FNWJ, VNTS DJ. We recall that the time limit of computation is 180 seconds for the CE MH. The RH MH have a time limit of 30 seconds (10 seconds) for each roll period of the 60-minute (180-minute) instances, since 6 (18) periods are required to solve the overall time horizon and the maximum computation time is also fixed equal to 180 seconds. The BB time limit for the CE MH (RH MH) is 10 seconds (4 seconds).

Table 8.4 provides the results obtained for the three groups of instances: the 60-minute delayed instances, the 180-minute delayed instances and the 180-minute disrupted instances. Column 1 presents a three-field code in order to identify each group of instances, where each code is structured as follows: [model variant]-[time horizon]-[NOrmal or DISrupted traffic]. Columns 2-3 report the average best objective function value (in seconds) obtained in [Samà et al., (2014)], and the average computation time (in seconds) at which that value was found. Columns 4-5 report the average best objective function value found by the best metaheuristics developed in this paper combined with the centralized and rolling horizon frameworks, and the average computation time. Columns 6-7 (8-9) (10-11) report the average best objective function value and the average computation time by the best algorithm in [Samà et al., (2014)] (for the best CEntralized MetaHeuristic) (for the best Rolling Horizon MetaHeuristic). The best average values are reported in bold regarding the best solution and the best algorithm columns. The results obtained for individual instances can be found in [Samà et al., (2015)].

Table 8.4: Average results for each type of instance

Instance Type	Best Sol		Best Sol MH		Best Algo		Best CE MH		Best RH MH	
	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)	Value (sec)	Time (sec)
M1-60-NO	**4.0**	7.1	**4.0**	27.0	**4.0**	7.1	4.4	19.1	5.7	180
M2-60-NO	**7.45**	44.9	**7.45**	34.3	7.5	44.9	**7.45**	34.3	8.5	180
M3-60-NO	**7.45**	22.1	**7.45**	37.7	8.7	10.1	**7.45**	37.7	8.1	180
M1-180-NO	**24.5**	330.9	**24.5**	133.4	**31.2**	273.7	41.4	89.0	36.5	180
M2-180-NO	39.2	348.2	**37.6**	147.5	42.9	348.4	43.9	99.6	**42.7**	180
M3-180-NO	**25.1**	403.8	37.8	138.2	**25.7**	388.3	43.9	99.8	44.2	180
M1-180-DIS	749.3	506.8	**699.7**	107.4	844.6	720	**756.2**	24.6	858.9	180
M2-180-DIS	**688.8**	641.9	691.2	100.9	770.8	720	**747.1**	40.4	842.7	180
M3-180-DIS	795.5	627.5	**699.0**	92.3	872.2	720	**806.4**	30.1	907.0	180

For the 60-minute instances of Table 8.4, the best solution found by the new metaheuristics (Column 4) is always equal to the best solution found by the algorithms in [Samà et al., (2014)] (Column 2), and all these solutions are proven optimal. When comparing the algorithms, the best CE MH is the best algorithm

in terms of the average objective function value, even if it often requires a longer computation time that the best algorithm in [Samà et al., (2014)]. The best RH MH is outperformed by the other best algorithms in terms of computation time, since the computation time of the rolling horizon framework is fixed to 180 seconds by construction.

For the 180-minute delayed instances of Table 8.4, the best CE MH and RH MH are significantly faster to compute their best solution compared to the best algorithm in [Samà et al., (2014)] (i.e. RH MILP). Regarding the solution quality, the best RH MH is better than the best CE MH. However, the best algorithm in [Samà et al., (2014)] gives the best average results, in a longer computation time, except for M2.

For the 180-minute disrupted instances of Table 8.4, the best CE MH outperforms the previously best known algorithm and the best RH MH in terms of both solution quality and time to compute the best solution. This strong improvement is due to the ability of VNS and VNTS to perform multiple simultaneous re-routing actions in the air segments of the TCA for landing aircraft. These re-routing actions are required in order to allow a better landing and take-off aircraft scheduling on the only available runway. The new best known solutions are found partly by the best RH MH and partly by the best CE MH, but the latter algorithm is quicker and has the best average performance in terms of the objective function value.

Table 8.5 presents the following aggregate comparisons: the best solutions computed via the new metaheuristics versus the best known solutions in [Samà et al., (2014)], the best centralized metaheuristic (CE MH) versus the best algorithm in [Samà et al., (2014)] (BA:2014), the best rolling horizon metaheuristic (RH MH) versus the best algorithm in [Samà et al., (2014)] (BA:2014).

Table 8.5: Performance comparison between algorithms for each type of instance

Instance Type	Comparison Type	Num Better Solut	Avg Value Reduction (sec)	Avg Time Variation (sec)	Num Equal Solut	Avg Time Variation (sec)	Num Worse Solut	Avg Value Increase (sec)	Avg Time Variation (sec)
M1-60-NO	Best Solut: MH vs BA:2014	0	-	-	20	+20.0	0	-	-
	Best Algo: CE MH vs BA:2014	0	-	-	18	+9.5	2	4.5	+35.2
	Best Algo: RH MH vs BA:2014	0	-	-	19	+174.0	1	34.0	+151.9
M2-60-NO	Best Solut: MH vs BA:2014	0	-	-	20	-10.6	0	-	-
	Best Algo: CE MH vs BA:2014	1	1.0	-125.1	19	-4.6	0	-	-
	Best Algo: RH MH vs BA:2014	0	-	-	17	+144.6	3	6.7	+81.3
M3-60-NO	Best Solut: MH vs BA:2014	0	-	-	20	+15.6	0	-	-
	Best Algo: CE MH vs BA:2014	1	26.0	+45.9	19	+26.6	0	-	-
	Best Algo: RH MH vs BA:2014	1	26.0	+180.0	17	+172.0	2	6.5	+146.2
M1-180-NO	Best Solut: MH vs BA:2014	6	5.5	-230.8	11	-183.1	3	11.0	-183.7
	Best Algo: CE MH vs BA:2014	5	29.8	-169.1	2	-129.8	13	27.2	-199.1
	Best Algo: RH MH vs BA:2014	6	25.3	-103.3	5	-62.3	9	28.8	-104.7
M2-180-NO	Best Solut: MH vs BA:2014	4	21.3	-283.1	11	-194.3	5	11.0	-149.0
	Best Algo: CE MH vs BA:2014	5	24.6	-235.8	5	-282.1	10	14.2	-238.6
	Best Algo: RH MH vs BA:2014	5	32.2	-172.7	6	-153.3	9	17.3	-176.0
M3-180-NO	Best Solut: MH vs BA:2014	6	17.5	-324.2	1	-207.9	13	27.8	-243.0
	Best Algo: CE MH vs BA:2014	4	10.0	-306.4	3	-334.1	13	31.1	-272.5
	Best Algo: RH MH vs BA:2014	5	15.8	-238.8	1	-207.9	14	32.1	-197.4
M1-180-DIS	Best Solut: MH vs BA:2014	10	160.4	-345.7	3	-397.6	7	87.4	-476.9
	Best Algo: CE MH vs BA:2014	13	197.5	-684.7	0	-	7	114.3	-715.3
	Best Algo: RH MH vs BA:2014	5	136.0	-540.0	7	-540.0	8	120.8	-540.0
M2-180-DIS	Best Solut: MH vs BA:2014	9	109.7	-571.7	1	-445.1	10	103.6	-523.0
	Best Algo: CE MH vs BA:2014	9	187.4	-683.1	0	-	11	110.2	-676.8
	Best Algo: RH MH vs BA:2014	3	291.3	-540.0	6	-540.0	11	210.2	-540.0
M3-180-DIS	Best Solut: MH vs BA:2014	20	96.6	-535.0	0	-	0	-	-
	Best Algo: CE MH vs BA:2014	13	162.6	-695.7	0	-	7	113.9	-679.3
	Best Algo: RH MH vs BA:2014	5	223.2	-540.0	4	-540.0	11	164.6	-540.0

In Table 8.5, Column 1 gives the three field instance code [model variant]-[time horizon]-[NOrmal or

DISrupted traffic]; Column 2 the type of comparison; Columns 3–5 the number of better solutions obtained by the new metaheuristics, the average reduction of the objective function value (in seconds), the average variation of the time (in seconds) to compute the best solution (we note that a negative variation means that the new metaheuristics are faster, while a positive variation means that the new metaheuristics are slower); Columns 6–7 the number of equal solutions obtained by the new metaheuristics, the average variation of the best solution computation time (in seconds); Columns 8–10 the number of worse solutions obtained by the new metaheuristics, the average increase of the objective function value (in seconds), the average variation of the best solution computation time (in seconds).

For the 60-minute instances, the number of equal solutions is very large for all models and there is a small time variation in computing the best known solutions by the different approaches. This is related to the relative low complexity of the 60-minute delayed instances, and to the fact that the best known solutions in [Samà et al., (2014)] were already optimal. Furthermore, the best algorithms in [Samà et al., (2014)] are often very fast to compute the optimal solution, leaving a relatively small margin for further improving the computational performance. On average, the new metaheuristics take a larger computation time. However, the best CE MH presents a small increase of the computation time. Regarding the objective function value, the new metaheuristics compute three better solutions than the best algorithm in [Samà et al., (2014)].

For the 180-minute delayed instances, for 16 out of 60 instances the new metaheuristics compute a new best known solution in a significantly smaller computation time compared to the one required to compute the previously best known solution. For other 23 instances, the best known solution is computed by new metaheuristics again in a smaller computation time compared to [Samà et al., (2014)], while for the remaining instances 180 seconds of computation time are not sufficient in order to find the best known solution. In general, the main advantage of the new metaheuristics is related to the quickness to compute their best solution.

For the 180-minute disrupted instances, a new best known solution is found by the new metaheuristics in 39 out of 60 cases, the same best known solution is found in 4 cases but with a strongly reduced computation time, and a worsening solutions is found with 180 seconds of computation in the remaining cases. For each row of the table, the average solution value increase is always smaller than the average solution value reduction. For all cases, a main advantage of the new metaheuristics (specially in the centralized framework) is again a strongly reduced time to compute the best solution, that is an important requirement to fit the real-time application of the proposed methodology even for the most difficult instances.

8.4.5 Assessment of the computational performance

This section presents a detailed performance assessment in the first 180 seconds of computation for a limited set of algorithms in the centralized framework: the best metaheuristics developed in this paper and the tabu search algorithm. The metaheuristics combined with the rolling horizon framework are not reported, since they deliver a feasible solution at the end of the given computation time (i.e. at 180 seconds). We also do not report the results obtained for the other approaches in [Samà et al., (2014)], since we limit our assessment to the only algorithms that are able to find a feasible solution for all ATC-TCA instances in the given maximum computation time. We note that the commonly used First-In-First-Out rule (FIFO) is very fast but it computes a feasible solution for the undisrupted instances only. The other algorithms in [Samà et al., (2014)] have a better feasibility rate than FIFO but they take a too long computation time for the 180-minute instances.

Figures 8.6, 8.7 and 8.8 report the plots of average best objective function values found at time $T = 10; 20; ...; 170; 180$ seconds, for the 60-minute delayed, the 180-minute delayed and the 180-minute disrupted instances. These figures show the results obtained for five metaheuristics: TS, VNS DJ, VNS WOCP+FNWJ, VNTS DJ, VNTS WOCP+FNWJ. Each point of the plots gives the average values obtained at a given computation time. In case some infeasibility is reported at an intermediate time point T_i for a metaheuristic, its average value at T_i considers the worst solutions obtained at T_i by the other metaheuristics.

From the plots in Figure 8.6, a number of observations follow for various time points. At 10 seconds of computation, the initial solution is provided, that is often the one computed by the branch-and-bound algorithm with default routes. At 20 seconds of computation, TS outperforms, on average, all the other metaheuristics. For large computation times, most of the new metaheuristics present significantly better results. This is probably due to the different neighbourhoods and search strategies used: TS mostly performs single routing changes on the ramified critical path neighbourhood, while the new metaheuristics are based on multiple simultaneous routing changes in air segment and runway resources. The stronger is the routing change, the larger is the potential to find a better quality aircraft scheduling in the TCA resources.

Among the new metaheuristics, VNS DJ and VNTS DJ are the best performing algorithms. It is remarkable that the solution improvement compared to TS is mainly in the first 40 seconds of computation, even if further marginal improvements are obtained till the time limit of computation. At 180 seconds of computation, the combination of VNS DJ and VNTS DJ delivers the optimal solution for all instances. Looking at the gap between the initial and final solutions, the reduction in objective function is quite impressive for all metaheuristics. The latter trend shows the importance of aircraft re-routing during disturbed operations.

Figure 8.7 shows a different performance pattern compared to Figure 8.6. The best solution is often improved during the 180 seconds of computation for the 180-minute delayed instances, since the solution

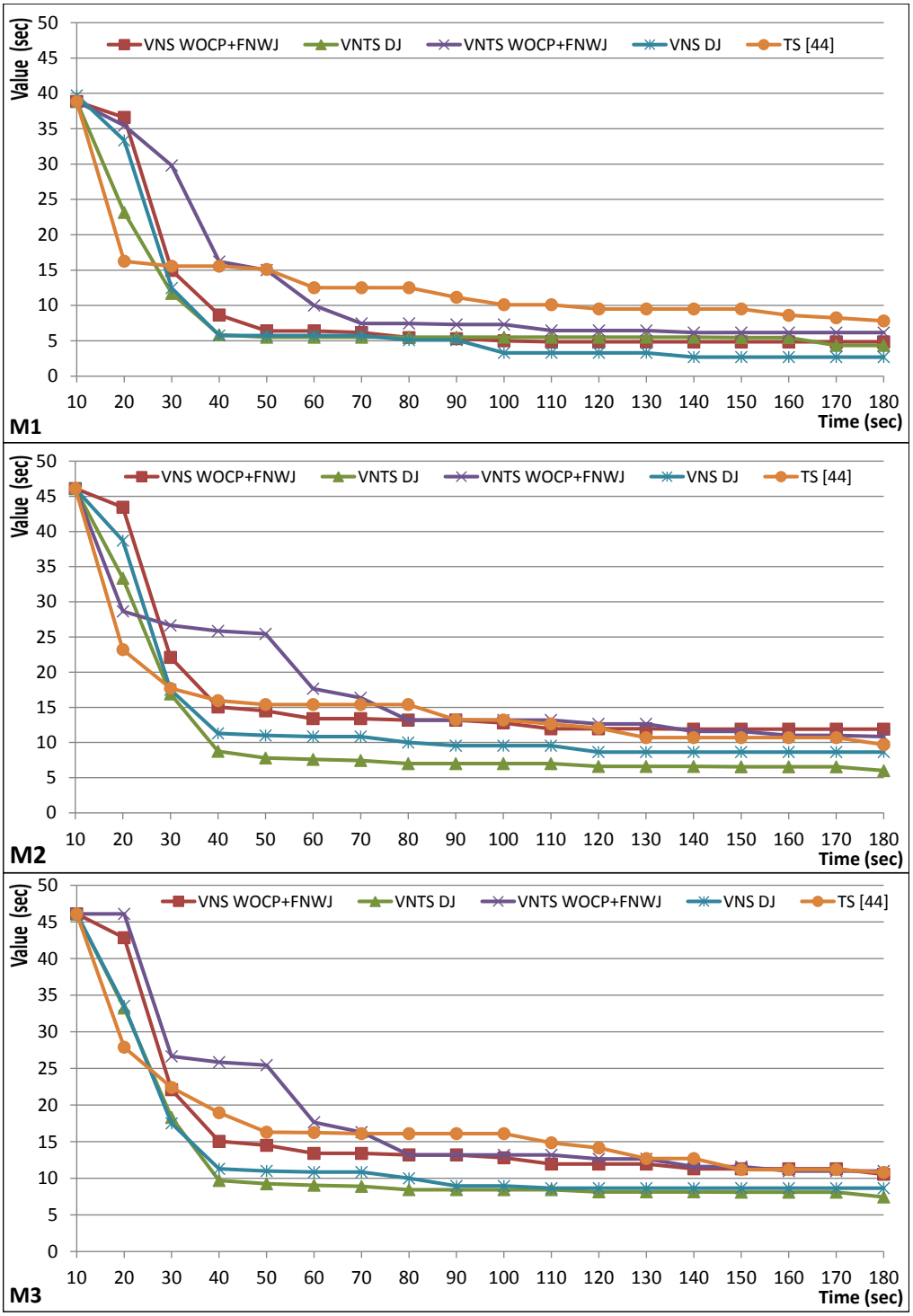


Figure 8.6: 60-minute delayed instances: results obtained for the metaheuristics

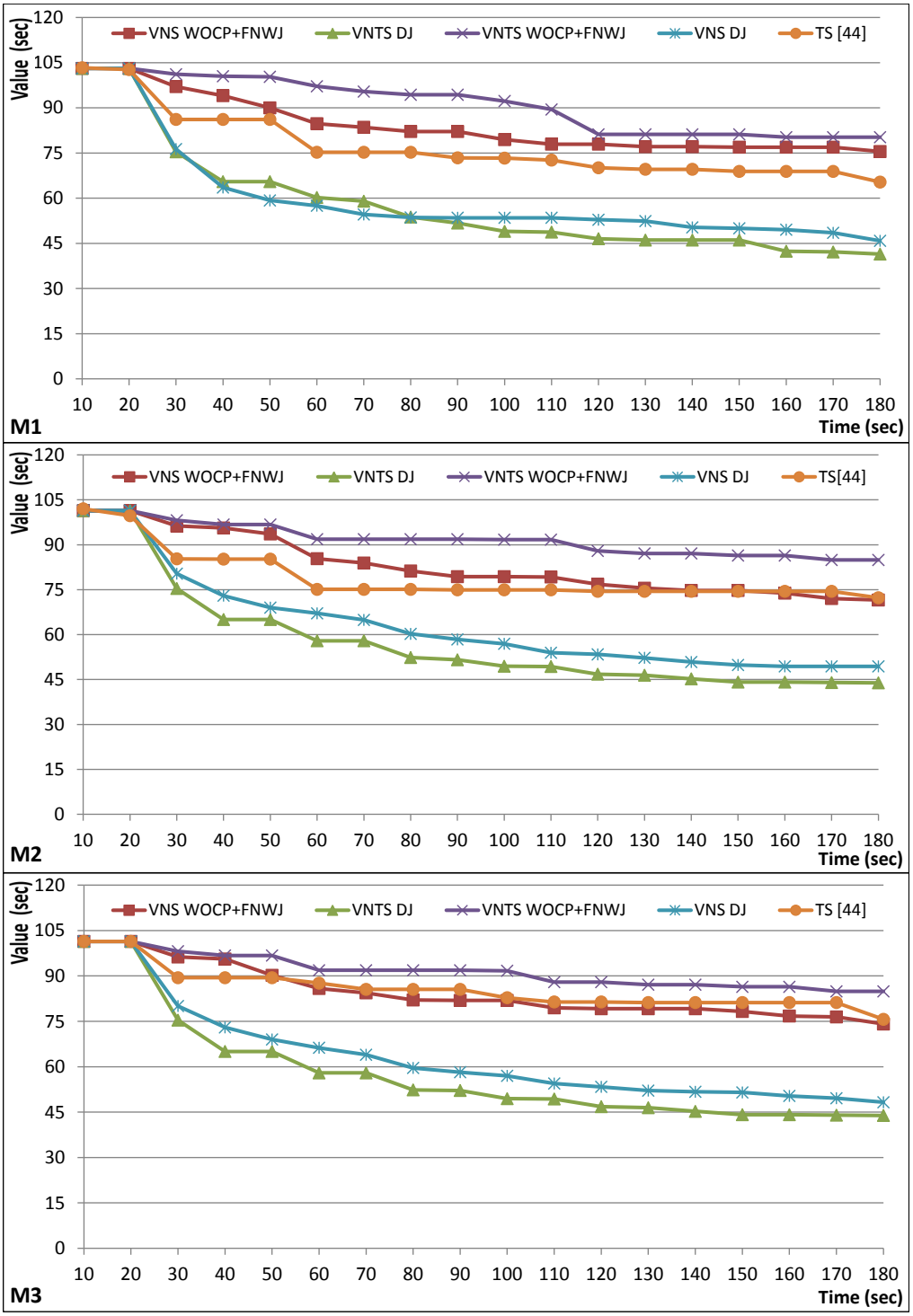


Figure 8.7: 180-minute delayed instances: results obtained for the metaheuristics

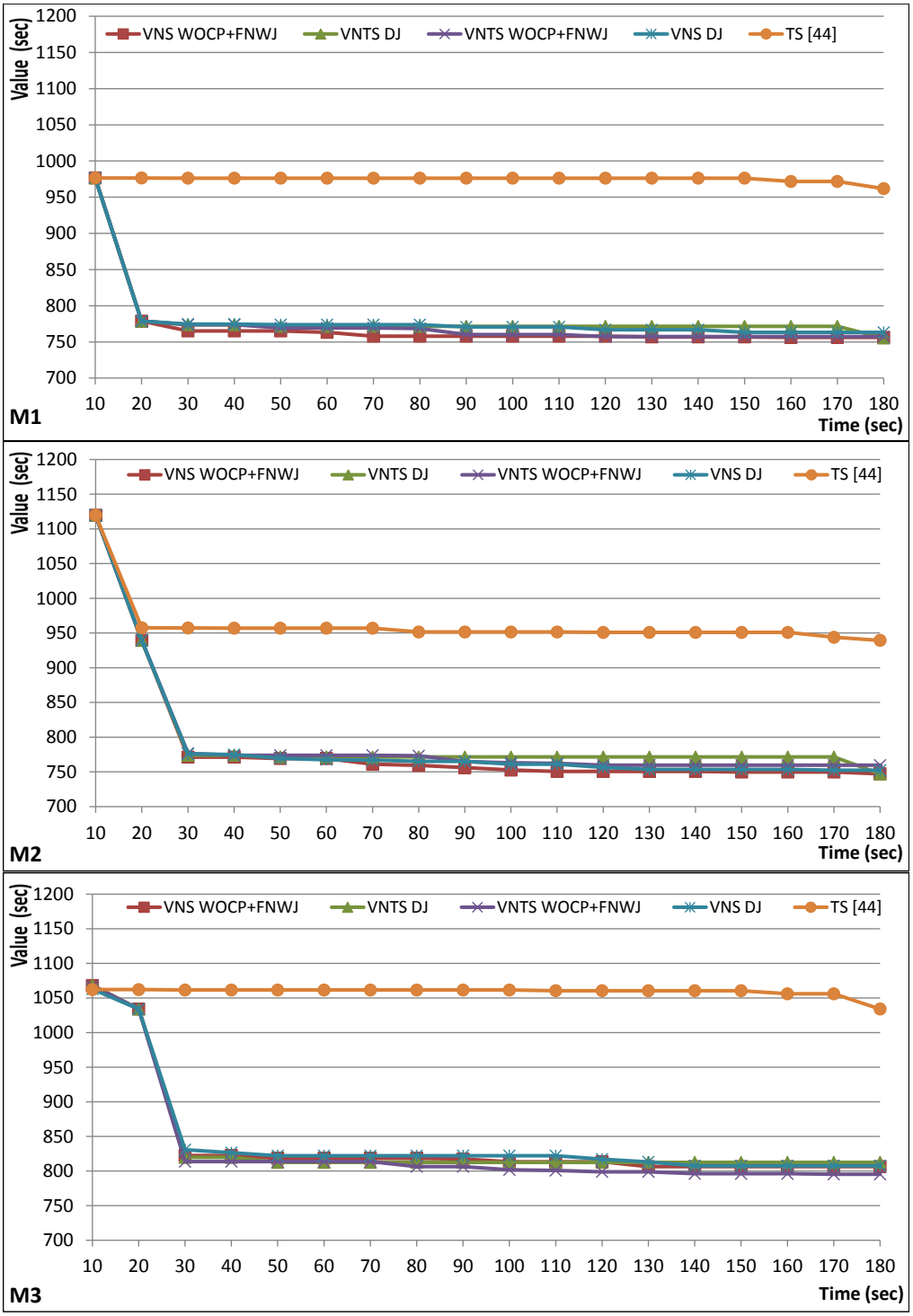


Figure 8.8: 180-minute disrupted instances: results obtained for the metaheuristics

space is very large and the computation of good quality solutions takes more time. From the results of Figure 8.7, VNTS DJ is very competitive during the entire search process and offers the best solutions at the end of the computation time. The second best metaheuristic is VNS DJ, remarking the very positive performance of the DJ neighbourhood. All metaheuristics present a good improvement of the initial solution after one minute of computation, and further marginal improvements are obtained in the other two minutes of computation.

Figure 8.8 presents a quite flat performance pattern compared to the other cases. Taking the average solution at $T = 10$ seconds as reference solution, a significant improvement is obtained at 20 seconds for TS in case of M2, while the other metaheuristics have their largest improvements during the first 30 seconds of computation. All the new metaheuristics have a similar performance, but VNS WOCP+FNWJ gives slightly better average results. Some marginal reduction of the objective function value is also obtained during the remaining computation time for all metaheuristics. In general, the flat performance pattern after 30 seconds is due to the limited number of feasible solutions available for aircraft re-routing and scheduling in case of disruption. However, for this set of instances the new metaheuristics provide their largest improvement of the objective function value compared to the previously best known solutions.

8.4.6 Discussion on the obtained results

Looking at the computational results, the main conclusions are next drawn. The new metaheuristics compute good quality solutions in much less time compared to state-of-the-art algorithms for the 180-minute traffic predictions. The results obtained by the new metaheuristics for the 60-minute traffic predictions are optimal, even if, for the latter instances, a slightly larger computation time is required compared with the best algorithms in [Samà et al., (2014)]. A new best known upper bound value is provided for 55 out of the 120 instances for which no optimal solution is yet proven. For the disrupted instances, the new neighbourhoods and metaheuristic schemes, on average, outperform the best results obtained with the TS algorithm (i.e. the only algorithm proposed in [Samà et al., (2014)] that is able to compute a feasible solution for all ATC-TCA instances during the first 180 seconds) within one minute of computation on a standard processor. The VNS and VNTS algorithms improve the results obtained by TS, since they explore multiple simultaneous aircraft re-routing actions and allow additional aircraft re-scheduling flexibility in the air segment and runway resources.

When comparing the centralized versus the rolling horizon frameworks, the metaheuristics are, on average, faster in the centralized framework. However, the solutions provided by the combination of the metaheuristics with the rolling horizon framework help to find new best known solutions for a significant number of instances. The strength of the rolling horizon approach is the problem decomposition in small time horizons, that has

the effect of simplifying the aircraft scheduling problem and thus enables a more effective evaluation of the potential routing moves during the neighbourhood search strategy.

The metaheuristics present a different performance for the different types of instances. Regarding the time horizon and type of disturbance, the 180-minute disrupted instances are more difficult to be solved compared to the other instances, since the 60-minute instances present a reduced number of variables and the 180-minute delayed instances have a reduced number of deadline constraints. Regarding the model variants, no significant performance different is found when comparing M1 and M2, even if the two models have a different number of due date constraints. On the contrary, M3 is more difficult to solve than the other models, since there are additional deadline constraints compared to M1 and M2. We recall that the presence of deadline constraints may strongly reduce the set of feasible schedules.

For the instances with deadline constraints the new metaheuristics are performing very well compared to the approaches in [Samà et al., (2014)], since these complex traffic situations require to perform simultaneous aircraft re-routing actions frequently in order to find good quality solutions. Another important reason is the use of the deadline implications and the pre-processing procedure developed in [Samà et al., (2014)], that help the metaheuristics to find good quality solutions and to compute better solutions than the commercial MILP solver.

8.5 Conclusions and future research

This paper proposes fast scheduling and routing metaheuristics for air traffic control at a busy TCA, with particular focus on the efficient control of strong traffic disturbances (such as multiple aircraft delays and a temporarily disrupted runway). To this end, several algorithmic innovations are considered, which relate to design of effective metaheuristics based on a decomposition of the problem into scheduling and routing decisions. A new set of neighbourhoods and new metaheuristic schemes are proposed on the basis of the hybridization between a variable neighbourhood search and a tabu search. Those algorithms are furthermore combined with a rolling-horizon based decomposition framework.

The algorithms and frameworks are evaluated on a Italian practical case study, i.e. the TCA of Milan Malpensa, and on a restricted group of instances that resulted the most challenging in [Samà et al., (2014)]. Those instances can be considered some of the most complex instances in the literature of the ATC-TCA problem, including a large number of aircraft and various sources of disturbances. We also compared the performance of the various approaches for three model variants, with differences in the set of constraints.

The new metaheuristics are benchmarked against state-of-the-art ATC-TCA approaches which include optimization algorithms developed in the AGLIBRARY solver and a MILP formulation for simultaneous

aircraft scheduling and routing solved by a commercial MILP solver. The algorithms proposed in this paper, with new neighbourhood structures and search strategies, improve the effectiveness of the previously developed approaches. The main contributions are next summarized: a general significant reduction of the time required to compute good quality solutions; a better performance in terms of solution quality and computation time, especially for the most difficult instances including disruptions; and the computation of new best known solutions for the several instances for which the optimal solution is not yet proven.

The average computational behaviour of the new metaheuristics is also analyzed in depth, resulting in a slightly slower descent compared to the TS (the best performing algorithm in [Samà et al., (2014)] during the first three minutes of computation) in the first 20 seconds. However, at around 1 minute of computation the new metaheuristics offer the possibility to strongly reduce the objective function value computed by TS.

Future efforts should be dedicated to a path towards the implementation of advanced ATC-TCA approaches into a dynamic air traffic management system. The optimization models, algorithms and frameworks presented in this work should be a core component of the intelligent transport system. Other issues are worthwhile being investigated, including the development of good quality lower bounds for the ATC-TCA problem, the extension of our methodology to better deal with aircraft speed variations in the arriving and departing procedures, the integration and coordination with large-scale air traffic control measures.

8.6 Appendix

A small illustrative example is presented in order to explain and illustrate the basic characteristics of the neighbourhoods, the neighbours and their evaluations. We refer the reader to Samà et al. [Samà et al., (2014)] for a detailed description of the alternative graph $G(N, F, A)$ formulation of the ATC-TCA problem.

The proposed illustrative example considers four aircraft (J1, J2, J3, J4) travelling on the network of Figure 8.9. The network is composed by seventeen resources: 1–3 are the three airborne holding circles, 4–14 are the eleven air segments for arriving procedures, 15 is the common glide path, 16–17 are the two runways. Aircraft J1, J2, J4 are arriving in the TCA, while aircraft J3 is departing from the TCA.

Each aircraft has a default route (numbered as route 1) and a set of routing alternatives. Figure 8.9 shows the default route assigned to each aircraft: **J1-1**: 1-4-10-13-15-16-out; **J2-1**: 3-8-12-14-15-17-out; **J3-1**: 16-out; **J4-1**: 1-4-10-13-15-17-out. With *out* we mean the first resource outside the network.

All the routing possibilities available for each aircraft are now described. The landing aircraft J1 and J4 have eight routing alternatives: all alternatives start from resource 1; from resource 1 to resource 15 the aircraft can follow four routes: 1-4-10-13-15, 1-4-9-11-15, 1-5-9-10-13-15, 1-5-11-15; from resource 15 the aircraft can land via resource 16 or 17. The landing aircraft J2 has only two routing alternatives: land via

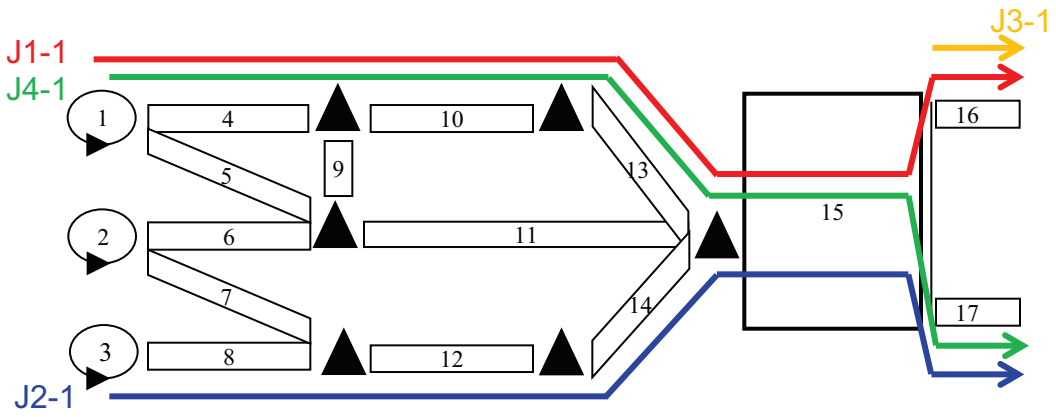


Figure 8.9: Illustrative example: the network and the default route of each aircraft

runway 16 or 17. The take-off aircraft J3 has also two routing alternatives: take-off via runway 16 or 17.

Figure 8.10 shows the alternative graph $G(N, F, A)$ of the example of Figure 8.9, in which the default route is considered for each aircraft. A different color is used to highlight each aircraft route that is identified by a sequence of nodes (J1-1 is red, J2-1 is blue, J3-1 is orange, J4-1 is green). Each node (operation) in N (except for the start node s and the end node t) is identified by the following three-field code: (aircraft-route, resource). For example, node (J2-1,3) means aircraft J2 with route 1 (i.e. the default route) traverses resource 3. Each fixed directed arc in F is identified by a solid arrow, while each alternative arc in A is identified by a dotted arrow. The weight of each arc is depicted nearby the corresponding arrow.

In the alternative graph $G(N, F, A)$ of Figure 8.10, the following aircraft scheduling decisions have to be taken: airborne holding circle decisions for J1, J2 and J4 on resource 1 (the time in the holding circle can be either 0, 180 or 240); a sequencing order between J1 and J4 at the entrance in and at the exit from resources 4, 10, 13; a sequencing order between J1, J2 and J4 at the entrance in and at the exit from resource 15; a sequencing order between J1 and J3 on resource 16; a sequencing order between J2 and J4 on resource 17. In this numerical example, the minimum separation time between aircraft is always set to 42.

Figure 8.11 presents the graph $\mathcal{G}(F, S)$ of a solution to the ATC-TCA problem with the default routes for the example of Figure 8.9. In $\mathcal{G}(F, S)$, exactly one arc is selected for each alternative pair. The following aircraft scheduling decisions have been taken: J1, J2 and J4 do not perform airborne holding circles; J1 precedes J4 at the entrance in and at the exit from resources 4, 10, 13, 15; the sequencing order at the entrance in and at the exit from resource 15 is J1, J2 and J4; J1 (J2) precedes J3 (J4) on resource 16 (17).

We next discuss four neighbourhoods explained in Section 8.3.2 in terms of the aircraft ranking.

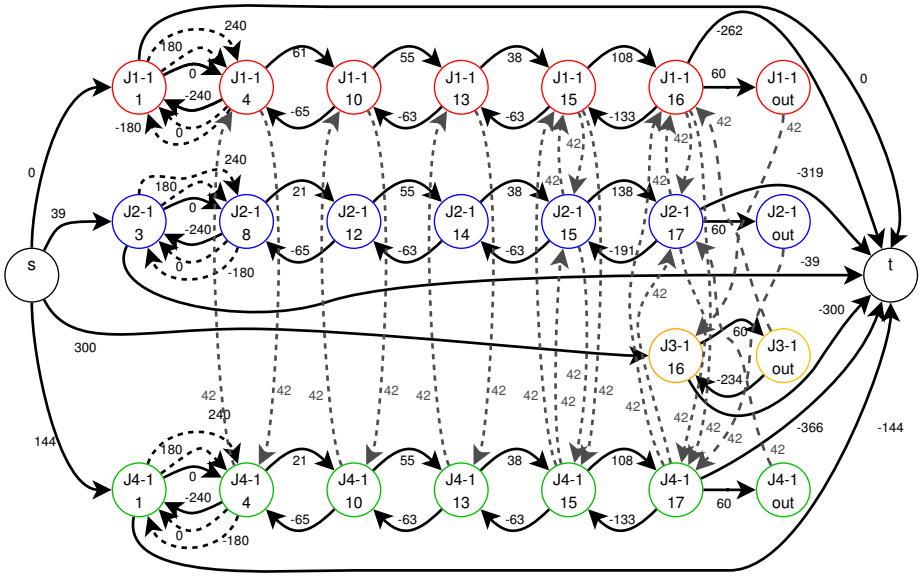


Figure 8.10: Alternative graph $G(N, F, A)$ for the example of Figure 8.9

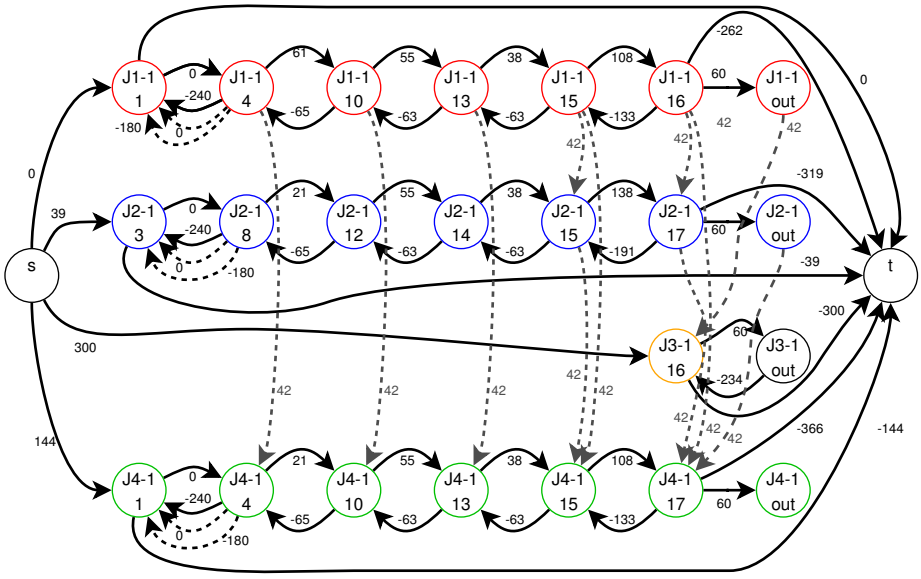


Figure 8.11: Graph $\mathcal{G}(F, S)$ of an ATC-TCA solution for the example of Figure 8.9

Free-Net Waiting Operations Jobs neighbourhood $\mathcal{N}_{FNWJ}$: The $\mathcal{N}_{FNWJ}$ ranking is computed on the alternative graph $G(N, F, A)$ of the incumbent solution as follows. We first identify each waiting node in $G(N, F, A)$ by selecting only the alternative arc generating the waiting node and by disregarding all the other alternative arcs in A. We then compute a restricted set of waiting nodes for each aircraft as follows. We only consider the alternative pairs in which both the alternative arcs generate a consecutive delay. For those alternative pairs, we take the alternative arc generating the largest consecutive delay in $G(N, F, A)$ and the corresponding waiting node. We finally set the score to each aircraft as the sum of the consecutive delays obtained on its restricted set of waiting nodes.

We now describe $\mathcal{N}_{FNWJ}$ for the alternative graph of Figure 8.10. In this example, the restricted set of waiting nodes for all aircraft is obtained by three alternative pairs as follows:

- (((J1-1,out), (J3-1,16)),((J3-1,out), (J1-1,16))) generate the waiting node (J1-1,16) with consecutive delay equal to $l^{S(F)}(s,(J3-1,out)) + w^{A}_{(J3-1,out),(J1-1,16)} + l^{S(F)}((J1-1,16),t) = 360 + 42 - 262 = 140$ when the only alternative arc ((J3-1,out), (J1-1,16)) is selected in S;
- (((J1-1,15), (J2-1,15)),((J2-1,17), (J1-1,16))) generate the waiting node (J1-1,16) with consecutive delay equal to $l^{S(F)}(s,(J2-1,17)) + w^{A}_{(J2-1,17),(J1-1,16)} + l^{S(F)}((J1-1,16),t) = 291 + 42 - 262 = 71$ when the only alternative arc ((J2-1,17), (J1-1,16)) is selected in S;
- (((J2-1,15), (J1-1,15)),((J1-1,16), (J2-1,17))) generate the waiting node (J1-1,15) with consecutive delay equal to $l^{S(F)}(s,(J2-1,15)) + w^{A}_{(J2-1,15),(J1-1,15)} + l^{S(F)}((J1-1,15),t) = 153 + 42 - 154 = 41$ when the only alternative arc ((J2-1,15), (J1-1,15)) is selected in S.

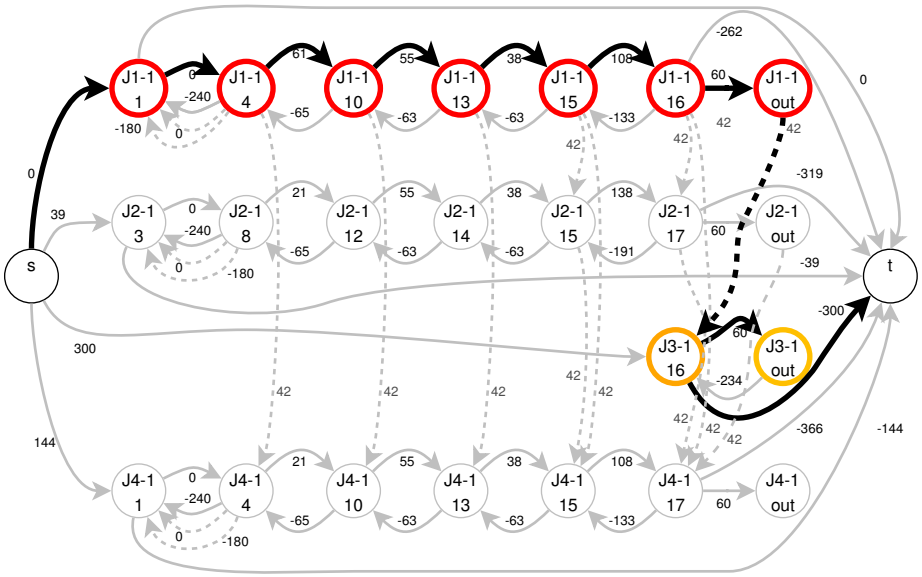
The ranking values for $\mathcal{N}_{FNWJ}$ are: J1-1: 252 (= 140+71+41); J2-1: 112 (= 71+41); J3-1: 140; J4-1: 0.

Ramified Critical Path Operations neighbourhood $\mathcal{N}_{RCPO}$: The $\mathcal{N}_{RCPO}$ ranking is based on the computation of the value $\beta = \max\{\ 0, \max\{\ l^{S(F)}(s,krp) + l^{S(F)}(krp,t)\ \forall\ (krp)\ \}\ \}$ in the ramified critical path of the graph $\mathcal{G}(F,S)$ of the incumbent solution. The aircraft on the ramified critical path have the value β in the ranking, while the other aircraft have a value equal to 0.

Figure 8.12 highlights the ramified critical path for the graph $\mathcal{G}(F,S)$ of Figure 8.11. The nodes and arcs on the ramified critical path are reported in bold (the nine fixed directed arcs are depicted with solid black arrows, the alternative arc with a dotted black arrow, the nodes of J1 with red circles, the nodes of J3 with orange circles). The critical path is composed by the following operations: (J1-1,1), (J1-1,4), (J1-1,10), (J1-1,13), (J1-1,15), (J1-1,16), (J1-1,out) and (J3-1,16). The ramified critical path is composed by the operations on the critical path plus all the other operations of the aircraft involved on the critical path. In this case, the ramified critical path extends the critical path with the additional operation (J3-1,out).

We now compute the ranking values for each aircraft in $\mathcal{N}_{RCPO}$. All the operations on the critical path

have the value β equal to the longest path in the graph $\mathcal{G}(F,S)$. For example, for operation (J3-1,16) we have the following value: $l^{S(F)}(s,(J3{-}1,16)) + l^{S(F)}((J3{-}1,16),t) = 364$ - $300 = 64$. For the additional operation (J3-1,out) on the ramified critical path, we have a negative value: $l^{S(F)}(s,(J3-1,16)) + w^{F}_{(J3-1,16),(J3-1,out)}$ $+ w^{F}_{(J3-1,out),(J3-1,16)} + l^{S(F)}((J3-1,16),t) = 364 + 60$ - 234 - $300 =$ -100. Therefore, the score of the two aircraft on the ramified critical path is the same (i.e. J1-1: 64 and J3-1: 64), while the score of the other two aircraft is null (i.e. J2-1: 0 and J4-1: 0).



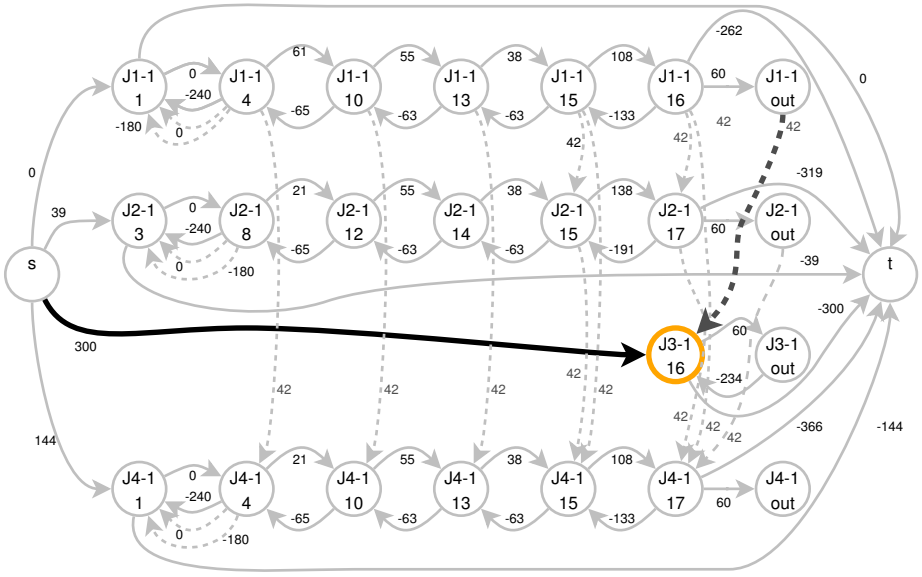
Aircraft	Ranking
J1-1	64
J2-1	0
J3-1	64
J4-1	0

Figure 8.12: The ramified critical path for the graph $\mathcal{G}(F,S)$

Waiting Operations Critical Path neighbourhood $\mathcal{N}_{WOCP}$: The $\mathcal{N}_{WOCP}$ ranking restricts the set of aircraft to be re-routed as follows. Each aircraft with some waiting nodes on the critical path $\mathcal{C}(F,S)$ of the graph $\mathcal{G}(F,S)$ has a score equal to the sum of the consecutive delays associated at these nodes. Each aircraft with no waiting node on the critical path $\mathcal{C}(F,S)$ of the graph $\mathcal{G}(F,S)$ has a null score.

We now describe $\mathcal{N}_{WOCP}$ for the graph $\mathcal{G}(F,S)$ of Figure 8.11. In this example, the only waiting node (waiting operation) on the critical path $\mathcal{C}(F,S)$ is (J3-1,16). Figure 8.13 highlights this waiting node with an orange circle, the alternative arc ((J1-1,out),(J3-1,16)) generating this waiting node with a bold dotted arrow, and the fixed arc (s, (J3-1,16)) entering this waiting node with a solid black arrow. The consecutive delay on the waiting node (J3-1,16) is computed as follows: the weight of the longest path $l^{S(F)}(s,(J1-1,out))$ plus the weight of the alternative arc $w^{A}_{(J1-1,out),(J3-1,16)}$ (42) minus the weight of the longest path $l^{S(F)}(s,(J3-1,16))$ (i.e. 322 + 42 - 300 = 64). Since (J3-1,16) is the only a waiting node in

graph $\mathcal{G}(F, S)$, the score of the aircraft J3 is 64, while the score of the aircraft J1, J2 and J4 is 0.

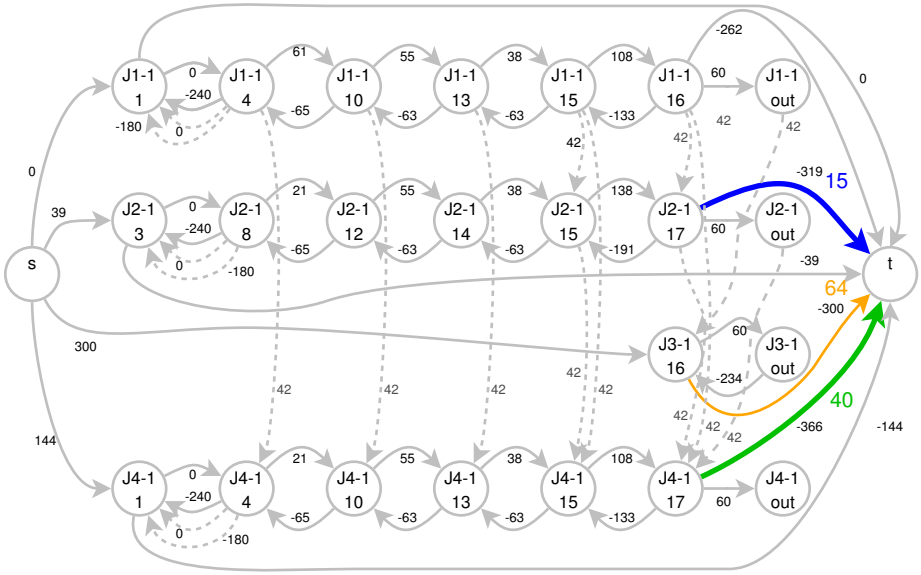


Aircraft	Ranking
J1-1	0
J2-1	0
J3-1	64
J4-1	0

Figure 8.13: Waiting operation on the critical path of the graph $\mathcal{G}(F, S)$

Delayed Jobs neighbourhood $\mathcal{N}_{DJ}$: The $\mathcal{N}_{DJ}$ ranking restricts the set of aircraft to be re-routed as follows. Each aircraft with a consecutive delay on its due date arcs in the graph $\mathcal{G}(F, S)$ has a score equal to the maximum consecutive delay generated on these due date arcs. Each aircraft without consecutive delays on its due date arcs in the graph $\mathcal{G}(F, S)$ has a null score. We recall that the consecutive delay on a due date arc ((Jx-y,z), t) is computed as the weight of the longest path $l^{S(F)}(s,(Jx\text{-}y,z))$ plus the weight of the due date arc ((Jx-y,z), t). For example, the consecutive delay on ((J2-1,17), t) is 334 - 319 = 15.

For the graph $\mathcal{G}(F, S)$ of Figure 8.11, we have the following $\mathcal{N}_{DJ}$ ranking. J1-1 is not delayed in any of its due date arcs, its score is thus 0. J2-1 has a consecutive delay equal to 15 on the due date arc ((J2-1,17), t), that is depicted with a blue solid arrow in Figure 8.14, its score is thus 15. J3-1 has a consecutive delay equal to 64 on the due date arc ((J3-1,16), t), that is depicted with an orange solid arrow in Figure 8.14, its score is thus 64. J4-1 has a consecutive delay equal to 40 on the due date arc ((J4-1,17), t), that is depicted with a green solid arrow in Figure 8.14, its score is thus 40.



Aircraft	Ranking
J1-1	0
J2-1	15
J3-1	64
J4-1	40

Figure 8.14: Delayed jobs in the graph $\mathcal{G}(F, S)$

Chapter 9

Air Traffic Optimization Models for Aircraft Delay and Travel Time Minimization in Terminal Control Areas

Abstract: *This paper addresses the real-time aircraft routing and scheduling problem at a busy Terminal Control Area (TCA) in case of traffic congestion. The problem of effectively managing TCA operations is particularly challenging, since there is a continuous growth of traffic demand and the TCAs are becoming the bottleneck of the entire air traffic control system. The resulting increase in airport congestion, economic and environmental penalties can be measured in terms of several performance indicators, including take-off and landing aircraft delays and energy consumption. This work addresses this problem via the development of mixed-integer linear programming formulations that incorporate the safety rules with high modeling precision and objective functions of practical interest based on the minimization of the total travel time and the largest delay due to potential aircraft conflicts. Computational experiments are performed on real-world data from Roma Fiumicino, the largest airport in Italy in terms of passenger demand. Traffic disturbances are generated by simulating sets of random landing/take-off aircraft delays. Near-optimal solutions of practical-size instances are computed in a short time via a commercial solver. The computational analysis makes possible the selection of those solutions offering the best compromise among the different objectives.*

9.1 Introduction

Aviation authorities are seeking intelligent decision support systems to effectively manage TCA operations in case of traffic congestion. The aircraft routing and scheduling problem is particularly relevant at a busy Terminal Control Area (TCA), since there is a continuous growth of traffic demand and a limited capacity of infrastructure resources. Several methods have been recently proposed in order to better use the available infrastructure and to improve the multifaceted characteristic of the air traffic management system (see e.g. the decision support tool developed in [Abdelghany et al., (2008)] to generate efficient airlines schedule recovery plans that minimize flight delays and cancellations; the system described in [Andreatta et al., (2014)] for intelligent monitoring and controlling ground vehicle movements; the modelling approaches proposed in [Atkin et al., (2009), Atkin et al., (2010), Ravizza et al., (2013)] for improving runway controller decisions with the view points of minimising delays, fuel consumption and treating aircraft equitably; the aircraft trajectory optimization models developed in [Palagachev et al., (2013)] to produce physically feasible trajectories for landing aircraft; the simulation-based studies presented in [Bubalo and Daduna (2011)] for analysing the impact of different capacity scenarios and increasing levels of demand on the runway throughput; the slot allocation mechanisms based on market principles provided in [Castelli et al., (2011), Pellegrini et al., (2012)]).

However, there is still a lack of contributions and practical systems on the effective and simultaneous control of landing and take-off operations. As reported e.g. in [Djokic et al., (2010),Kim et al., (2009),Prevot et al., (2011)], the development of advanced decision support systems for air traffic controllers require paying attention to multi-fold and simultaneous aspects of challenging operational problems:

- The modeling of TCA resources should be able to incorporate the TCA safety regulations, so that potential conflicts between single aircraft are visible, at least at the level of runways, ground and air segments of the TCA. This would require detailed information on the aircraft and the TCA resources, that is often neglected in macroscopic models with aggregated flight paths. The latter models are often utilized for air traffic management problems requiring the consideration of large networks with multiple airports (see e.g. the runway configuration management problem and the arrival/departure runway balancing problem [Bertsimas et al., (2011b)]; the problem of managing simultaneously the flight routing and sequencing in capacitated en route sectors and airports [Bertsimas et al., (2011a)]; the strategic planning problem of identifying various impacts of delay propagation at several airports [Churchill et a., (2010)]).

- The time available for the computation and implementation of an efficient aircraft routing and scheduling solution can be very limited. The decision support system should be able to compute a solution

each time there is a significant change of aircraft position and speed or a capacity drop of some TCA resources occur during operations.

- There is a shortage of generally recognized performance indicators in order to assess the solutions produced by the system, while human traffic controllers still develop feasible schedules based on their past experience and intuition. The resulting increase in airport congestion, economic and environmental penalties could thus be limited by the development of efficient solution methods in terms of economic and environmental penalties, including take-off and landing aircraft delays and energy consumption.

This paper addresses the three issues above. The first issue is approached by developing new mixed-integer linear programming (MILP) formulations for Air Traffic Flow Management in a Terminal Control Area (ATFM-TCA), taking into account the relevant TCA safety aspects with a high level of detail. We started from the approach of Bianco et al. [Bianco et al., (2006)] that is based on the no-wait job shop scheduling problem with aircraft routing and timing variables. Other approaches to solve job shop scheduling problems with additional constraints can be found e.g. in Will and Voss [Witt and Voss (2010)].

In this work, the alternative graph model of [Mascis and Pacciarelli (2002)] is used to increase the level of detail of the ATFM-TCA formulation based on job shop scheduling. Further relevant TCA aspects are modelled, such as holding circles, speed intervals for aircraft, capacitated use of air segments and no-store constraints at runways. Compared with published papers from our research group (a branch and bound algorithm for aircraft scheduling [D'Ariano et al., (2010), D'Ariano et al., (2015)], a tabu search algorithm for aircraft routing [Corman et al., (2010)], a rolling horizon approach for static and dynamic air traffic management [Samà et al., (2013a)]), this paper presents a simultaneous optimization of aircraft routing and scheduling decisions, and investigates the combination of two objective functions of practical interest.

The second and third issues require to test the optimization model with a short computation time and to pay special attention to the definition of the indexes. We follow a most common choice in the literature that is the use of a single objective function, typically as a combination of various performance indicators (see e.g. the air transport reviews regarding disruption management and other irregular operations [Ball et al., (2007), Clausen et al., (2010), Kohl et al., (2007)], demand and capacity management [Barnhart et al (2012)], scheduling of airport operations [Bennell et al., (2011)], methods for aircraft conflict detection and resolution [Kuchar and Yang (2000)], analysis of similarities and differences between the air and rail transportation systems [Pellegrini and Rodriguez (2013)]). However, single objective optimization approaches are generally faster than multi-objective optimization.

Computational experiments have been carried out on real-world data via a commercial solver. The test bed is the Italian airport of Roma Fiumicino (FCO). We consider practical-size instances with multiple delayed aircraft and solve the related ATFM-TCA problems. As for the indexes, we analyze two objective

functions of practical interest: the minimization of the largest delay due to potential aircraft conflicts, and the minimization of the total travel time spent in the TCA. A weighted sum of them is also investigated in order to identify a reasonable balance. Near-optimal solutions from the viewpoint of the two performance indicators are provided to the traffic controllers.

Section 9.2 formally describes the ATFM-TCA problem, including a description of the specific constraints and objectives, while Section 9.3 presents the mathematical formulations. Section 9.4 reports a campaign of experiments to assess the quality of the solutions computed by the solver, and its applicability to deal with real-time computing requirements. Section 9.5 summarizes the paper results and outlines directions for further research. An appendix reports numerical examples and Gantt charts of traffic flows in the FCO TCA.

9.2 Problem description

The problem of managing aircraft in a TCA can be divided into: (i) Routing decisions, where an origin-destination route for each aircraft has to be chosen; (ii) Timing decisions, where routes are fixed under traffic regulation constraints and aircraft passing timing have to be determined in each air segment, runway and (possibly) holding circle. In this work, routing (i) and timing (ii) decisions are taken simultaneously.

The ATFM-TCA problem is defined as follows: given a set of landing and take-off aircraft and, for each aircraft, a set of possible paths in the TCA, its current position, its scheduled runway occupation time and a time window to accomplish the arriving/departing procedures, assign an entry time to each aircraft in each resource (holding circles, air segments, runways) traversed by the aircraft in the chosen path in such a way that potential conflicts between single aircraft are efficiently solved in each TCA resource. The resolution of potential aircraft conflicts requires that aircraft respect the minimum longitudinal and diagonal safety distances.

In the TCA, each arriving aircraft moves from an entry point to a runway, following a standard descend profile, while maintaining a minimum safety distance with the other aircraft in air segments and at the runway, depending on their type and position. Similarly, each departing aircraft leaves the runway flying toward the assigned exit point along an ascent standard profile, still respecting separation safety distances. Since the variability of aircraft speed in the TCA is limited, this distance can be translated into a *setup time*. Setup times are sequence-dependent, since the minimum distance between heavy, medium or light aircraft depends on the relative order of processing of the common resources. Each aircraft has a minimum entry time in the TCA, named *release time*. Each aircraft has also scheduled times, *due date times*, to start processing some TCA resources.

The runway can be occupied by only one aircraft at a time, and each aircraft has a travel time on a runway and on the air segments before or after it, according to its landing/take-off profile. On the air segments, the travel time can vary within a pre-defined time window, due to the limited possibility of aircraft speed changes.

Landing aircraft enter the TCA, traverse air segments with a standard descend profile and proceed to the runway. However, in case of congestion, airborne holding circle resources can be used as entrance buffers until aircraft can be guided through their arriving procedure in the TCA. Once entered a holding circle, the landing aircraft must travel at a fixed speed for at least half circle. After performing half circle of a loop the landing aircraft can reverse direction in order to continue the arriving procedure. For each aircraft, there is a maximum number of allowed half circles, as prescribed by the air traffic controller. We assume that holding circle resources are uncapacitated.

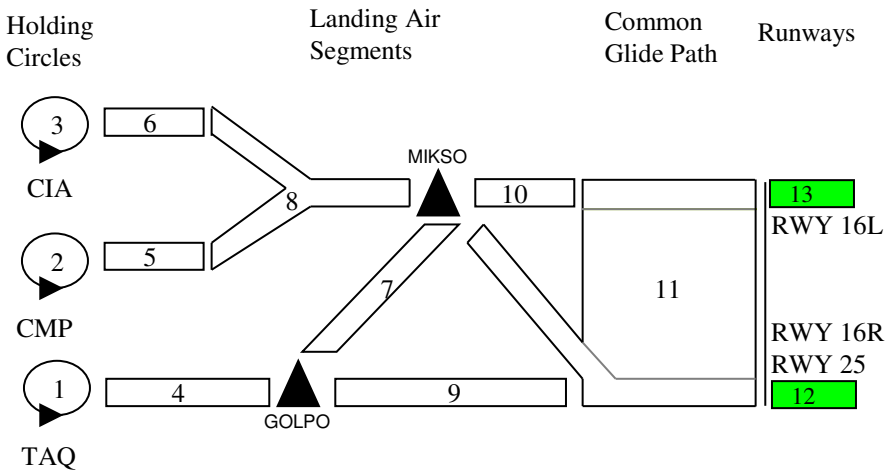


Figure 9.1: Fiumicino Terminal Control Area

Figure 9.1 presents the scheme of Fiumicino Terminal Control Area (FCO TCA). Three runways (RWY 16L, RWY 16R, RWY25) can be used for arriving and departing procedures, but two of them (RWY 16R and RWY 25) cannot be used at the same time and are thus considered as one. The airport resources are 3 airborne holding circles (CIA, CMP, TAQ, labelled 1-3), 7 air segments for arriving procedures (labelled 4-10) and 3 for departing ones (BOL, RAVAL, ELIVIN, labelled 14-16), 2 runways (RWY 16R and RWY 25 labelled 12, and RWY 16L labelled 13) and 1 common glide path (labelled 11). Black triangles represent merging points in the airspace (named GOLPO and MIKSO).

In the FCO TCA, arriving aircraft enter at the holding circles (resources 1–3), pass the landing air segments (resources 4–11) and land on a runway (resources 12–13). Departure aircraft enter the dedicated runway (resource 12) and leave the model through take-off air segments (that are not depicted in Figure 9.1).

Potential conflicts between arrival and departure flows are on resource 12.

Take-off aircraft are supposed to depart within their assigned time window and are late whenever they are not able to accomplish the departing procedure within the assigned time window. A take-off aircraft can be delayed in entering the TCA at ground level, before entering the runway. Following the procedure commonly adopted by air traffic controllers, we consider a time window for take-off between 5 minutes before and 10 minutes after the *Scheduled Take-off Time* (STT) in the timetable. A departing aircraft is considered late if leaving the runway after 10 minutes from its STT. So a ground delay does not necessarily cause a delay at the runway. Take-off aircraft are late if landing after their *Scheduled Landing Time* (SLT) in the timetable.

We use the following notation for the aircraft delays. *Entrance delay* is the delay of each aircraft at the entrance in the TCA. Specifically, landing aircraft enter the TCA at the holding circle resources, while take-off aircraft enter the TCA at the dedicated runway. Both for landing and take-off aircraft, *exit delay* is the delay of each aircraft measured at the runway. The latter value can be divided into an *unavoidable delay* at the runway, consequence of a late entrance in the TCA, and a *consecutive delay*, consequence of the resolution of potential aircraft conflicts [D'Ariano et al., (2007a),D'Ariano et al., (2010),D'Ariano et al., (2012b)].

Landing aircraft can have consecutive delays at the entrance, if other aircraft are scheduled before them in some TCA resources. Landing/take-off aircraft can have a consecutive delay at a runway, if they have to give precedence to other aircraft in some TCA resources.

Two objective functions are considered: the minimization of the maximum consecutive delay and of the total travel time spent by aircraft in the TCA. The former is the largest delay caused by the resolution of potential aircraft route conflicts in the TCA, while the latter can be considered as a surrogate for the energy consumption minimization.

9.3 Aircraft routing and scheduling formulations

The ATFM-TCA problem can be formulated as a job shop scheduling problem with additional practical constraints, in which an *operation* denotes the traversing of a resource (i.e. air segment, common glide path, runway, holding circle) by a job (i.e. aircraft). The sequence of operations of an aircraft is named *route*. Once a route has been assigned to each aircraft (routing problem), the ATFM-TCA problem (with fixed routes) is reduced to the Aircraft Scheduling Problem (ASP).

The variables of the ASP are the start time t_i of each operation i to be performed by an aircraft on a specific resource. For a given operation i of an aircraft, we denote with $\sigma(i)$ the operation following i on its

route, and with $w_{i\sigma(i)}$ a minimum separation time between the start time of operations i and $\sigma(i)$. A set of feasible route timings is *conflict-free* if, for each pair of operations associated to the same resource, the minimum separation constraints between aircraft in each resource of the TCA are satisfied.

The ASP is represented by an alternative graph as follows. Let $G = (N, F, A)$ be the graph composed by the following sets:

- $N = \{0, 1, ..., n-1, n\}$ is the set of *nodes*, where 0 and n represent the start and end operations of the schedule, while the other nodes are related to the start time of the other operations;
- F is the set of *fixed arcs* that model the sequence of operations to be executed by an aircraft (i.e. the chosen route for the aircraft);
- A is the set of *alternative pairs* that model the sequencing decisions. Each pair $((i,j),(h,k)) \in A$ is composed by two alternative arcs, either (i,j) or (h,k) must be selected in a feasible schedule.

Each node of the graph is thus associated to the start time t_i of operation i. By definition, the start time of the schedule is $t_0 = 0$. Each arc (i,j), either fixed or alternative, has a length w_{ij}, which indicates a minimum separation time between operations i and j, i.e. $t_j \geq t_i + w_{ij}$. A detailed description of the constraints related to the specific TCA resources can be found in [D'Ariano et al., (2010), D'Ariano et al., (2012b), D'Ariano et al., (2015), Samà et al., (2013a), Samà et al., (2013b)].

A *selection* S is a set of alternative arcs, at most one arc from each alternative pair. A selection is *complete* S^c if exactly one arc for each alternative pair is selected. A complete selection is a *feasible schedule* S^f if the connected graph $(N, F \cup S^f)$ has no positive length cycles.

Given a selection S and any two nodes i and j, we let $l^S(i,j)$ be the length of the longest path from i to j in the connected graph $(N, F \cup S)$. By definition, the timing t_i of node $i \in N$ is the quantity $l^S(0,i)$, which implies $t_0 = 0$ and $t_n = l^S(0,n)$.

The alternative graph with flexible routing can be viewed as a particular *disjunctive program*. We model the variables and constraints for the different routes of each aircraft as follows. Let X be the set of feasible ATFM-TCA solutions:

$$X = \left\{ \begin{array}{ll} t \geq 0, x \in \{0,1\}^{|A|}, y \in \{0,1\}^{|C|} : & \\ t_{\sigma(i)} - t_i + M(1 - y_{ab}) \geq w_{i\sigma(i)} & \forall (i,\sigma(i)) \in F \;\; with \;\; \sigma(i) \neq n \\ t_j - t_i + M(1 - x_{ijhk}) + M(1 - y_{ab}) + M(1 - y_{cd}) \geq w_{ij} & \forall ((i,j),(h,k)) \in A \\ t_k - t_h + M x_{ijhk} + M(1 - y_{ab}) + M(1 - y_{cd}) \geq w_{hk} & \\ \sum_{a=1}^{R_b} y_{ab} = 1 & b = 1, ..., Z \end{array} \right\} \quad (9.1)$$

The variables are the following:

- $|N|$ real variables t_i associated to the start time of each operation $i \in N$,
- $|A|$ binary variables x_{ijhk} associated to each alternative pair $((i,j),(h,k)) \in A$,
- $|C|$ real variables associated to the routes of the set of aircraft considered.

Further, Z is the number of aircraft, and R_b the number of routes for each aircraft $b = 1, ..., Z$. The constant M is a sufficiently large number, e.g. the sum of all arc lengths.

Variable $y_{ab}[y_{cd}] \in \{0,1\}$ indicates if route a $[c]$ is chosen (1) or not (0) for aircraft b $[d]$. Exactly one route for each aircraft must be selected, e.g. for aircraft b: $\sum_{a=1}^{R_b} y_{ab} = 1$. When a route a is chosen for aircraft b (i.e. $y_{ab} = 1$), each constraint related to the fixed arcs of route a and aircraft b must be satisfied. For the corresponding fixed arcs $(i, \sigma(i)) \in F$, $t_{\sigma(i)} - t_i \geq w_{i\sigma(i)}$ must hold.

If $y_{ab} = y_{cd} = 1$ and aircraft b and d are scheduled on a same resource of the TCA, a potential conflict exists on that resource and an ordering decision has to be taken. This is modelled by introducing the variable $x_{ijhk} \in \{0,1\}$ for the alternative pair $((i,j),(h,k)) \in A$, related to the two aircraft travelling on that specific resource. If $x_{ijhk} = 1$ then $t_j - t_i \geq w_{ij}$ must be satisfied (i.e. $(i,j) \in S$), otherwise $t_k - t_h \geq w_{hk}$ must be satisfied (i.e. $(h,k) \in S$).

In order to formulate the objective functions, we must introduce two types of due date arcs: *entrance due date arcs*, associated with the start time of the first operation of each landing aircraft to measure its entrance delay; *exit due date arcs*, associated with the start time of the runway operation of each aircraft to measure the consecutive delay in the TCA.

We let $j \in N$ be the first operation of a landing aircraft in the TCA, ϕ_j its scheduled entrance time and f_j its entrance delay, that we assume to be a constraint for the traffic controller. The length of the entrance due date arc $(j,n) \in F$ is $d_j = -\phi_j - f_j$. The consecutive delay at the entrance of the TCA is $max\{0, t_j + d_j\}$.

We let $i \in N$ be the arrival/departure operation at/from a runway r of a landing/take-off aircraft A, β_i its scheduled arrival/departure time and τ_i its earliest possible entrance time in the runway r. The latter time is computed as the sum of the release time plus the minimum travel time into the landing air segments. In case of a take-off aircraft, τ_i is the sum of the release time plus the scheduled travel time in the runway. The exit delay of A at r is $t_i - \beta_i$.

We recall that the exit delay at the runway is composed by the unavoidable delay (which cannot be recovered by aircraft rescheduling) plus the consecutive delay (required to solve potential aircraft conflicts in the TCA). In the alternative graph, these delays are computed as follows. Let's fix the length of the exit due date arc $(i,n) \in F$ as $d_i = -max\{\tau_i, \beta_i\}$, the unavoidable delay is $max\{0, \tau_i - \beta_i\}$, while the consecutive delay at the runway is $max\{0, t_i + d_i\}$.

The Maximum Tardiness **MT** corresponds to the minimization of the maximum consecutive delay [D'Ariano et al., (2007a), D'Ariano et al., (2010)]. In our case, MT is the largest positive deviation from the entrance and exit scheduled times. For this objective function, a feasible schedule S^f is optimal if $l^{S^f}(0,n)$ is minimum over all the solutions. The MT formulation is the following:

$$\begin{aligned} &\min t_n \\ &s.t \\ &t_n - t_k + M(1-y_{ab}) \geq d_k \quad \forall (k,n) \in F \\ &\{x,y,t\} \in X \end{aligned} \tag{9.2}$$

The minimization of the Total Travel Time Spent **TTTS** is the objective function we use as a surrogate for the energy consumption minimization in the TCA. For an aircraft a, let t_{af} be the end time of its last operation and let t_{ar} be its release time, the travel time spent in the TCA by the aircraft can be computed as $t_{af} - t_{ar}$. The TTTS formulation is the following:

$$\begin{aligned} &\min \textstyle\sum_{a=1}^{|Z|}(t_{af} - t_{ar}) \\ &s.t \\ &\{x,y,t\} \in X \end{aligned} \tag{9.3}$$

The two objective functions reported in (9.2)–(9.3) support specific aspects of the ATFM-TCA problem. We also study a convex combination of these two objective functions, named **MT-TTTS**:

$$\begin{aligned} &\min \alpha\ \delta\ t_n + (1-\alpha)\ \gamma \textstyle\sum_{a=1}^{|Z|}(t_{af} - t_{ar}) \\ &s.t. \\ &t_n - t_k + M(1-y_{ab}) \geq d_k \qquad\qquad \forall (k,n) \in F \\ &\{x,y,t\} \in X \end{aligned} \tag{9.4}$$

where $\delta = 1/t_n^*$, $\gamma = 1/\sum_{a=1}^{|Z|}(t_{af}^* - t_{ar}^*)$ and α is a value between 0 and 1. The latter value is used to balance the importance of each objective function. The values t_n^* and $\sum_{a=1}^{|Z|}(t_{af}^* - t_{ar}^*)$ are, respectively, the best known values for the MT and TTTS problems.

Since the $\alpha = 0$ and $\alpha = 1$ could give arbitrary values for the non-minimized objective, *modified MT-TTTS formulations* can be defined by fixing an upper bound value for the first objective while minimizing the secondary objective (similarly e.g. to [Samà et al., (2013b)]). For the TTTS formulation (i.e. the case with $\alpha = 0$) the constraint $t_n \leq t_n^*$ should be added, while for the MT formulation (i.e. the case with $\alpha = 1$)

the constraint $\sum_{a=1}^{|Z|}(t_{af} - t_{ar}) \leq \sum_{a=1}^{|Z|}(t^*_{af} - t^*_{ar})$ should be added.

It has to be observed that the optimal solutions of the modified MT-TTTS formulations are non-dominated (Pareto efficient) solutions only if t^*_n and $\sum_{a=1}^{|Z|}(t^*_{af} - t^*_{ar})$ are the optimal values for the MT and TTTS problems.

9.4 Experimental results

This section presents the computational results for the ATFM-TCA formulations of Section 9.3. The tests have been performed in a laboratory environment by using real-world data from FCO TCA. Specifically, the travel and setup times are computed for each aircraft category (small, medium, large) according to standard descend and ascent profiles, disregarding the actual aircraft passenger and freight load. The release and due date times are computed from a reference timetable. The ATFM-TCA solutions are computed by using CPLEX MIP solver 12.6. The experiments are executed on a workstation Power Mac with processor Intel Xeon E5 quad-core (3.7 GHz), 12 GB of RAM.

9.4.1 Test cases

Table 9.1 describes the studied ATFM-TCA instances. Each row presents average information on 100 randomly disturbed scenarios. In this set of experiments, entrance delays are uniformly distributed with respect to a nominal (undisturbed) scenario (i.e. a reference timetable). Entrance delays are up to 5 minutes, since we investigate how to solve potential aircraft conflicts in 30-minute traffic predictions. Column 1 reports the time period of traffic prediction (in minutes), Columns 2-3 the type of aircraft and the number of landing and take-off aircraft, Columns 4-5 the maximum and average entrance delays (in seconds), Columns 6-7 the maximum and average unavoidable delays (in seconds), and Column 8 the number of routes available for all aircraft. For each landing aircraft we consider two alternative routes, varying the chosen runway, while we do not examine rerouting measures for take-off aircraft.

Table 9.1: Fiumicino airport instances

Time Period	Aircraft Type	Num. of Aircraft	Max Entr. Delay (sec)	Avg Entr. Delay (sec)	Max Unav. Delay (sec)	Avg Unav. Delay (sec)	Aircraft Routes
0–30	Landing	16	183.36	22.37	89.39	8.68	32
0–30	Take-off	4	59.71	16.28	0	0	4

Table 9.2 gives the size of the MILP formulation in terms of the number of variables and constraints. We recall that $t \in X$ are the timing variables, $x \in X$ are the sequencing variables and $y \in X$ are the routing

Table 9.2: MILP formulation

ATFM-TCA Problem	Variables			Constraints	
	t	x	y	Fixed	Alternative
Scheduling	118	688	-	229	1376
Routing and Scheduling	154	2680	36	845	5360

variables. We divide the constraints in two sets: fixed and alternative constraints, corresponding to fixed and alternative arcs in the alternative graph.

We next report the results obtained for single and combined objective functions.

9.4.2 Single Objective Functions

Table 9.3 presents the average results on the 100 ATFM-TCA instances for Maximum Tardiness (MT) and Total Travel Time Spent (TTTS) formulations. Specifically, MT indicates the solutions obtained for Model (2), while TTTS indicates the solutions obtained for Model (3).

Table 9.3: ATFM-TCA solutions computed for single objective functions

Obj Funct	**Maximum Tardiness (MT)**		**Total Travel Time Spent (TTTS)**		
ATFM-TCA Problem	Scheduling	Routing and Scheduling	Scheduling	Routing and Scheduling	Routing and Scheduling
Comp time	1 minute	1 minute	1 minute	1 minute	1 hour
Num Opt Sol	100	100	100	0	20
Opt Gap (%)	0	0	0	6.80 (3.02#)	2.83#
UB (sec)	57.75	22.59	13161.09	12515.32	12491.93
LB (sec)	57.75	22.59	13161.09	11718.90	12148.53
MT (sec)	-	-	241.81	141.51	131.84
TTTS (sec)	13394.01	13215.96	-	-	-

In Table 9.3, Row 1 presents the adopted objective functions (*Obj Funct*), while Row 2 the studied ATFM-TCA problems. *Scheduling* is the case with aircraft routes fixed off-line (fixed e.g. in order that the workload of the runways is well balanced and there is no conflict at runways when aircraft are on time). *Routing and Scheduling* is the case with routing flexibility. Rows 4–9 give average information on the results obtained by CPLEX with the time limit of computation reported in Row 3 (the last column presents a larger computation time that allowed us to get further information on the optimal solutions). Row 4 presents the number of problems that were solved to optimality (*Num Opt Sol*) and Row 5 the optimality gap (*Opt Gap*, in percentage) computed as follows: (upper bound − lower bound) / lower bound. The symbol # indicates when the optimality gap is computed with the lower bound obtained at 1 hour of computation. Row 6 gives

the Upper Bound (UB, in seconds) and Row 7 the Lower Bound (LB, in seconds) to the optimal objective function value. Rows 8–9 show the average values obtained for the one indicator when optimizing the other.

When evaluating the solutions found in Table 9.3, optimal or near-optimal solutions are computed in a short time for all the proposed formulations. Precisely, all instances of the MT formulation are solved to optimality in one minute of computation. The same trend applies for the scheduling problem of the TTTS formulation, while optimality is not proven for several instances of the routing and scheduling problem (either in 1 minute or 1 hour of computation). This fact is motivated by the increased number of variables for the latter TTTS problem. Also, the optimal TTTS solutions are often influenced by a larger number of aircraft compared to the optimal MT solutions.

We now look at the secondary performance indicator for each formulation. The solutions found for the minimization of TTTS have poor performance in terms of MT, while the solutions found for MT have small optimality gaps (1.77% in the scheduling case and 8.79% in the routing and scheduling case) regarding TTTS. The low correlation between the travel time and the aircraft delay minimization can be justified by the following observations. In presence of disturbances, landing aircraft can increase their travel time in holding circle and air segment resources before landing. However, this increment does not necessarily cause aircraft delays, since there is a scheduled recovery time in the timetable. This effect is more evident for landing aircraft that have, by construction, a large recovery time in the assigned time window of departure in the timetable.

9.4.3 Combined Objective Functions

This subsection presents the results obtained for Model (4) when varying the parameter α in the window $[0; 0.1; ...; 0.9; 1]$. The values $\alpha = 1$ and $\alpha = 0$ correspond to the resolution of Models (2) and (3). For each value of α in the MT-TTTS formulation, we analyze the average results obtained for the 100 ATFM-TCA instances. We use the best-known values of δ and γ, as reported in Table 9.3. Table 4 gives the results obtained for the scheduling problem, while Table 5 the results obtained for the routing and scheduling problem. We set the solver with a computation time of 1 minute.

Each row of Tables 4 and 5 reports the average results (in seconds) obtained for a given α on the 100 ATFM-TCA instances. For each objective function, the results are divided into landing and take-off aircraft per resource 12 (runways RWY 16R and RWY 25) and resource 13 (runway RWY 16L) separately. The maximum tardiness is not reported for take-off aircraft since these are always scheduled in the assigned time window. The MT formulation is thus focused on the delay minimization of landing aircraft, while the TTTS formulation improves the performance of both types of aircraft. The last two columns of each table report the number of optimal solutions and the optimality gap computed by CPLEX for the corresponding

combined objective function.

Table 9.4: Results on the scheduling problem

Obj Funct	Maximum Tardiness (sec)			Total Travel Time Spent (sec)				Num	Opt
Aircraft	All	Landing		All	Landing		Take-off	Opt	Gap
Resource	12 & 13	12	13	12 & 13	12	13	12	Sol	(%)
$\alpha = 0.0$	253.31	251.77	43.69	13161.09	5596.62	5947.83	1616.64	100	0
$\alpha = 0.1$	58.71	51.15	45.48	13304.03	5308.61	6086.09	1909.33	100	0
$\alpha = 0.2$	58.09	50.39	46.64	13313.54	5301.45	6094.90	1917.19	100	0
$\alpha = 0.3$	57.77	49.46	45.62	13327.98	5303.08	6105.88	1919.02	100	0
$\alpha = 0.4$	57.76	49.64	46.35	13328.94	5297.79	6106.18	1924.97	100	0
$\alpha = \mathbf{0.5}$	57.75	49.51	46.53	13330.83	5296.56	6109.76	1924.51	100	0
$\alpha = \mathbf{0.6}$	57.75	49.48	46.66	13330.83	5300.19	6109.76	1920.88	100	0
$\alpha = \mathbf{0.7}$	57.75	49.51	46.71	13330.83	5296.56	6109.76	1924.51	100	0
$\alpha = \mathbf{0.8}$	57.75	49.48	46.96	13330.83	5309.87	6109.76	1911.20	100	0
$\alpha = \mathbf{0.9}$	57.75	49.36	46.95	13330.83	5305.03	6109.76	1916.04	100	0
$\alpha = 1.0$	57.75	49.02	48.23	13381.89	5301.78	6136.19	1943.92	100	0

Table 9.5: Results on the routing and scheduling problem

Obj Funct	Maximum Tardiness (sec)			Total Travel Time Spent (sec)				Num	
Aircraft	All	Landing		All	Landing		Take-off	Opt	Gap
Resource	12 & 13	12	13	12 & 13	12	13	12	Sol	(%)
$\alpha = 0.0$	141.51	103.46	111.2	12515.4	5646.29	5557.61	1311.5	0	3.02
$\alpha = 0.1$	23.9	21.65	20.09	12824.01	5376.32	5605.99	1841.7	72	2.76
$\alpha = 0.2$	22.65	20.47	19.31	12861.9	5415.7	5556.6	1889.6	95	0.39
$\alpha = 0.3$	22.59	20.44	18.58	12867.8	5419.43	5552.93	1895.5	95	0.66
$\alpha = 0.4$	22.59	20.21	18.75	12868.8	5420.88	5551.35	1896.51	98	0.50
$\alpha = 0.5$	22.59	20.41	18.93	12868.26	5406.08	5567.31	1894.87	98	0.20
$\alpha = 0.6$	22.59	20.38	19.36	12867.95	5414.43	5561.49	1892.03	98	0.20
$\alpha = \mathbf{0.7}$	22.59	20.42	19.04	12867.62	5415.62	5558.15	1893.85	100	0
$\alpha = \mathbf{0.8}$	22.59	20.51	19.29	12867.62	5419.84	5555.97	1891.81	100	0
$\alpha = 0.9$	22.59	20.53	19.2	12868.24	5417.93	5553.7	1896.61	100	0
$\alpha = 1.0$	22.59	20.98	17.51	13199.33	5329.39	5733.48	2136.46	100	0

We define an average best value α^* for the identification of the best MT-TTTS formulation(s). This value corresponds to the selection of the combined formulation that minimizes the quantity:

$$\left\{ \frac{t_n - t_n^*}{t_n^*} + \frac{\sum_{a=1}^{|Z|}(t_{af} - t_{ar}) - \sum_{a=1}^{|Z|}(t_{af}^* - t_{ar}^*)}{\sum_{a=1}^{|Z|}(t_{af}^* - t_{ar}^*)} \right\} \tag{9.5}$$

In Tables 4 and 5, the average best values α^* are reported in bold. For all the α^* values the related MT-TTTS problem is solved to near-optimality in one minute of computation. In both tables, several values of α return the optimal values of the MT indicator. However, the corresponding values of the TTTS indicator are

still sub-optimal (1.29% gap in the scheduling case and 5.92% gap in the routing and scheduling case). We conclude that the solutions resulting from the combination of single objective functions have the drawback to deteriorate the performance of at least one indicator.

9.4.4 Illustration of non-dominated solutions

Figure 9.2 shows a possible representation of a set of Pareto efficient solutions obtained for an ASP instance of FCO TCA. The horizontal (vertical) axis reports the value obtained for the TTTS (MT) problem. This kind of plot should be useful to support the decision makers, who can express their preferences on which solution should be implemented based on some knowledge regarding the quality of alternative efficient solutions. The traffic controllers can, e.g., define a maximum level of delay tolerance and thus exclude those solutions that do not respect that additional constraint.

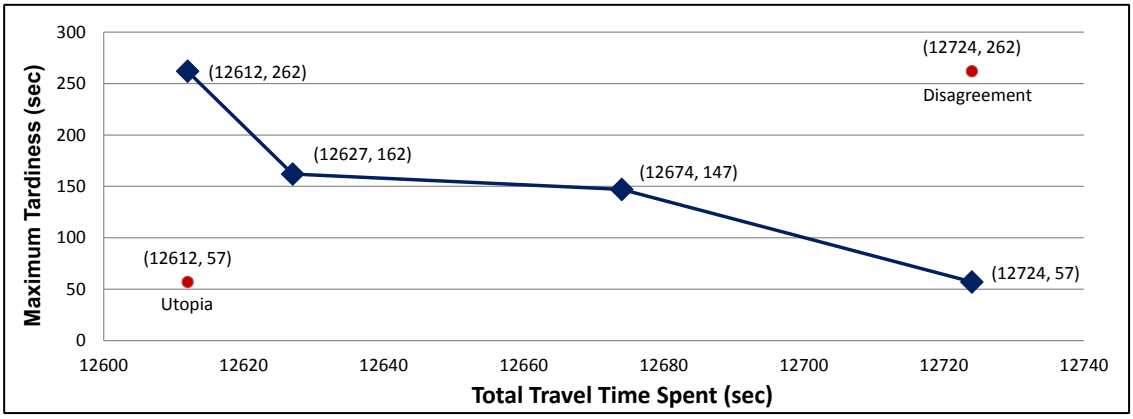


Figure 9.2: Non-dominated solutions for an aircraft scheduling instance

For the instance in Figure 9.2, an aggregated view is given from the viewpoint of four non-dominated solutions found by the following procedure:

1. Compute a first non-dominated solution by solving a single objective optimization problem.
2. Solve another single objective optimization problem by fixing the value of the objective function of the problem of step 1 as a constraint in the latter problem.
3. If the optimal solution has been found for the problem of step 2, a further non-dominated solution is found and go to step 4. Otherwise stop.
4. Repeat step 1 by adding the following constraint: the value of the objective function of the problem of step 2 must be lower than the optimal solution.

The procedure can also be executed by inverting objectives used at steps 1 and 2.

In addition to the four non-dominated solutions, we show the *utopia point*, that is the point whose coordinates are the best achievable for each objective function, and the *disagreement point*, that consists of setting for each coordinate the worst possible values. Based on those extreme points, classifications of non-dominated solutions can be identified according to bargaining theory [Agnetis et al., (2009), Kalai and Smorodinsky (1975)].

9.5 Conclusions and further research

This paper presents microscopic formulations of the ATFM-TCA problem with two objective functions of practical interest: minimization of the largest consecutive delay and the total travel time spent in the TCA. Computational results show the existence of relevant gaps between the (near)optimal solutions computed for one objective and the same performance indicator computed on the solutions optimized with the other objective. From the experiments on the combined objective functions, better trade-off solutions are obtained in terms of both the performance indicators.

In general, we believe that this work moves the interest of researchers and practitioners in considering efficient aircraft traffic control measures in order to recover from disturbances, and paying more attention to the inherent multi-objective nature of the ATFM-TCA problem.

Ongoing research is dedicated to the study of additional objective functions and more severe traffic disturbances, including temporary blocked runways and large aircraft delays caused e.g. by wake turbulences. A further interesting direction for further research is the study of different delay distributions. Future research will also be focused to the development of real-time aircraft scheduling, routing and trajectory optimization algorithms for multi-objective optimization models. The aim is again to compute efficient solutions in a short computation time.

9.6 Appendix

Figure 9.3 presents a numerical example of traffic flows for FCO TCA. We consider two landing aircraft (named A and B) and a taking-off aircraft (named C). The following routes are considered by the solver. Aircraft A has two routes: 3-6-8-10-11-13 is the *default* route, and 3-6-8-11-12 is the *alternative* route. Aircraft B has the default route 1-4-7-10-11-13, while aircraft C has the default route 12. The entrance (exit) due date of A is 40 (640), the entrance (exit) due date of B is 0 (640), the entrance (exit) due date of C is 630 (630). The release time of each aircraft is equal to the corresponding entrance due date time. The

travel time of each aircraft in each resource is reported in Figure 9.4.

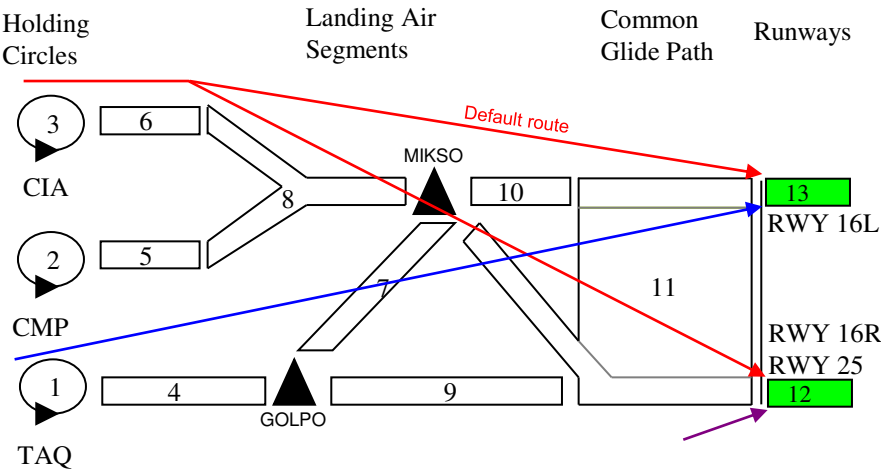


Figure 9.3: A numerical example for FCO TCA

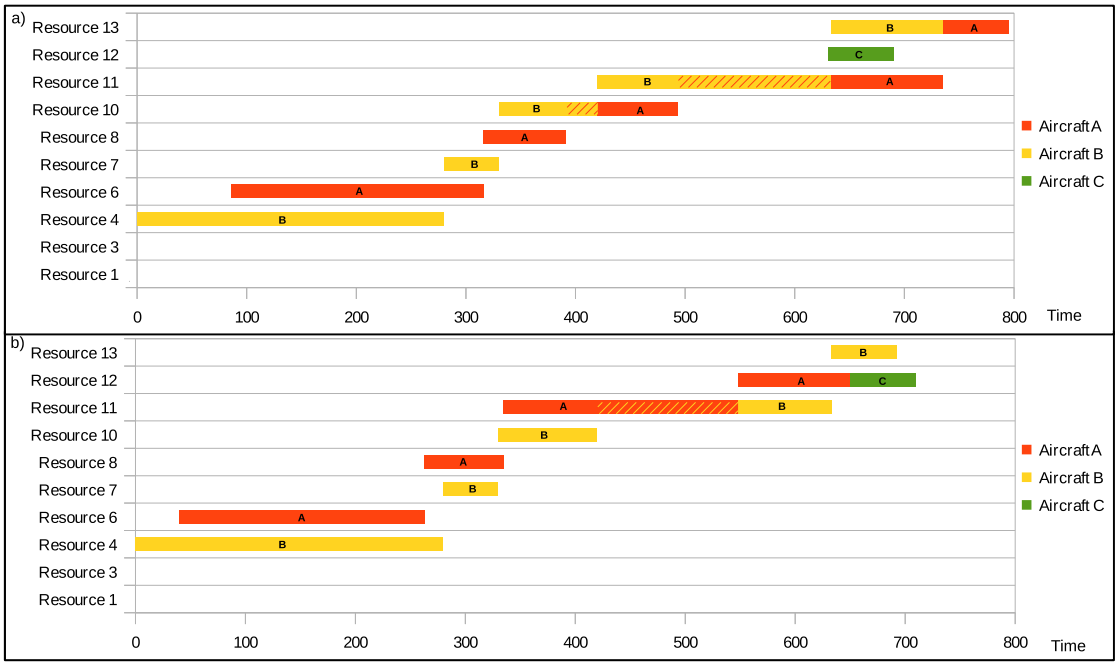


Figure 9.4: Gantt diagrams of the solutions with default routes (a) and alternative routes (b)

Figure 9.4 a (b) gives the Gantt diagram of the optimal ATFM-TCA solution for the default (alternative) routes. In the case with default routes, the routes of aircraft A and B are conflicting and the optimal sequencing order is first B and then A. The consecutive delay of A is 46 at the entrance (a too small delay

to consider half circles in the holding resource) and 95 at the runway. In the case with alternative routes, the routes of aircraft A and C are conflicting and the optimal sequencing order is first A and then C. Thus, the consecutive delay of C is 20.

Bibliography

[Abdelghany et al., (2008)] Abdelghany, K.F., Abdelghany, A.F., Ekollu, G., (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research* **185 (2)** 825-848.

[Agnetis et al., (2009)] Agnetis, A., De Pascale, G., Pranzo, M., (2009). Computing the Nash solution for scheduling bargaining problems. *International Journal of Operational Research* **6 (1)** 54–69.

[Ahuja et al., (2005)] Ahuja, R., Cunha, C., Şahin, G., (2005). Network models in railroad planning and scheduling. In: Greenberg, H., Smith, J., (Eds.), *TutORials in Operations Research*, **1**,54–101.

[Allahverdi et al., (2008)] Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y., (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* **187 (3)** 985–1032

[Almodóvar and García-Ródenas (2013)] Almodóvar, M., García-Ródenas, R., (2013). On-line reschedule optimization for passenger railways in case of emergencies. *Computers and Operations Research* **40 (3)** 725–736.

[Alonso-Ayuso et al., (2015)] Alonso-Ayuso, A., Escudero, L.F., Martín-Campo, F.J., Mladenović, N., (2015). A VNS metaheuristic for solving the aircraft conflict detection and resolution problem by performing turn changes. *Journal of Global Optimization* **63 (3)** 583–596.

[Andreatta et al., (2014)] Andreatta, G., Capanna, L., De Giovanni, L., Monaci, M., Righi, L., (2014). Efficiency and Robustness in Integrated Airport Apron, a Support Platform for Intelligent Airport Ground Handling. *Journal of Intelligent Transportation Systems* **18 (1)** 121–130.

[Artiouchine et al., (2008)] Artiouchine K., Baptiste P., Dürr C., (2008). Runway sequencing with holding patterns. *European Journal of Operational Research* **189 (3)** 1254–1266

[Atkin et al., (2007)] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D., (2007). Hybrid Metaheuristics to Aid Runway Scheduling at London Heathrow Airport. *Transportation Science* **41 (1)** 90–106.

[Atkin et al., (2009)] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., Reeson, D., (2009). An examination of take-off scheduling constraints at London Heathrow airport. *Public Transport* **1** 169–187.

[Atkin et al., (2010)] Atkin, J.A.D., Burke, E.K., Greenwood, J.S., (2010). TSAT allocation at London Heathrow: the relationship between slot compliance, throughput and equity. *Public Transport* **2 (3)** 173–198.

[Atkin et al., (2012)] Atkin, J.A.D., Maere, G.D., Burke, E.K., Greenwood, J.S., (2012). Addressing the pushback time allocation problem at Heathrow airport. *Transportation Science* **47 (4)** 584–602

[Balakrishnan and Chandran (2010)] Balakrishnan, H., Chandran, B., (2010). Algorithms for Scheduling Runway Operations under Constrained Position Shifting. *Operations Research* **58 (6)** 1650–1665

[Ball et al., (2007)] Ball, M.O., Barnhart, C., Nemhauser, G., Odoni, A., (2007). Air Transportation: Irregular Operations and Control. In: G. Laporte and C. Barnhart (Eds.). *Handbooks in Operations Research and Management Science* **14** 1–68

[Ball et al., (2010)] Ball, M.O., Hoffman, R., Mukherjee, A., (2010). Ground delay program planning under uncertainty based on the ration-by-distance principle. *Transportation Science* **44 (1)** 1–14

[Banker et al., (1984)] Banker, R., Charnes, A., Cooper, W.W., (1984). Some models for estimating technical and scale inefficiencies in data envelopment analysis. *Management Science* **30 (9)** 1078–1092.

[Barnhart et al (2012)] Barnhart, C., Fearing, D., Odoni, A., Vaze, V., (2012). Demand and capacity management in air transportation. *EURO Journal of Transportation and Logistics* **1 (1)** 135–155.

[Beasley et al., (2001)] Beasley, J.E., Sonander, J., Havelock, P., (2001). Scheduling aircraft landings at London Heathrow using a population heuristic. *Journal of the Operational Research Society* **52 (1)** 483–493.

[Beasley et al., (2004)] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., Abramson, D., (2004). Displacement problem and dynamically scheduling aircraft landings. *Journal of Operational Research Society* **55 (1)** 54–64

[Bencheikh et al., (2011)] Bencheikh, G., Boukachour, J., Hilali Alaoui, A.E., (2011). Improved ant colony algorithm to solve the aircraft landing problem. *International Journal of Computer Theory and Engineering* **3 (2)** 224–233.

[Bennell et al., (2011)] Bennell, J.A., Mesgarpour, M., Potts, C.N., (2011). Airport runway scheduling. *4OR – Quarterly Journal of Operations Research* **4 (2)** 115–138.

[Bertsimas et al., (2011a)] Bertsimas, D., Lulli, G., Odoni, A., (2011). An integer optimization approach to large-scale air traffic flow management. *Operations Research* **59 (1)** 211–227

[Bertsimas et al., (2011b)] Bertsimas, D., Frankovich, M., Odoni, A., (2011). Optimal Selection of Airport Runway Configurations. *Operations Research* **59 (6)** 1407–1419.

[Bianco et al., (2006)] Bianco, L., Dell'Olmo, P., Giordani, S., (2006). Scheduling models for air traffic control in terminal areas. *Journal of Scheduling* **9 (3)** 180–197

[Binder et al., (2014)] Binder, S., Chen, J., Bierlaire, M., (2014). Generation and evaluation of passenger-oriented railway disposition timetables, Proceedings of the 14th Swiss Transport Research Conference, Ascona, Switzerland.

[Bubalo and Daduna (2011)] Bubalo, B., Daduna, J.R., (2011). Airport capacity and demand calculations by simulation — the case of Berlin-Brandenburg International Airport. *NETNOMICS* **12 (3)** 161–181.

[Bussieck (1998)] Bussieck, M.,(1998). Optimal Lines in Public Rail Transport. *PhD thesis, Technische Universitat Braunschweig.*

[Cacchiani et al., (2014)] Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J., (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Science* **63 (1)** 15–37.

[Cacchiani and Toth (2012)] Cacchiani, V., Toth, P., (2012). Nominal and Robust Train Timetabling Problems. *European Journal of Operational Research* **219 (3)** 727–737.

[Cadarso and Marín (2014)] Cadarso, L., Marín, Á., (2014). Improving robustness of rolling stock circulations in rapid transit networks. *Computers and Operations Research*, **51 (1)**, 146–159.

[Cadarso et al., (2013)] Cadarso, L., Marín, Á., Maróti, G., (2013). Recovery of disruptions in rapid transit networks. *Transportation Research Part E* **53 (1)**, 15–33.

[Cai and Goh (1994)] Cai, X., and Goh, C.J., (1994). A fast heuristic for the train scheduling problem. *Computers and Operations Research* **21 (5)** 499–510.

[Caimi et al., (2011)] Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., Zenklusen, R., (2011). A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science* , **45 (2)** 212–227.

[Caimi et al., (2012)] Caimi, G., Fuchsberger, M., Laumanns, M., Lüthi, M., (2012). A model predictive control approach for discrete-time rescheduling in complex central railway station approach. *Computers and Operations Research* **39 (11)** 2578–2593.

[Caprara et al., (2006)] Caprara, A., Kroon, L.G., Monaci, M., Peeters, M., Toth, P., (2006). Passenger Railway Optimization. In: C. Barnhart & G. Laporte (Eds.), *Handbooks in Operations Research and Management Science*, **14** 129–187, Elsevier, Amsterdam, The Netherlands.

[Carey and Kwiecinski (1995)] Carey, M., Kwiecinski, A., (1995). Properties of expected costs and performance measures in stochastic models of scheduled transport. *European Journal of Operational Research* **83 (1)** 182–199.

[Carey and Carville (2003)] Carey, M., Carville, S., (2003). Scheduling and platforming trains at busy complex stations. *Transportation Research Part A* **37 (3)** 195–224.

[Carey and Crawford (2007)] Carey, M., Crawford, I., (2007). Scheduling trains on a network of busy complex stations. *Transportation Research Part B* **41 (2)** 159–178.

[Castelli et al., (2011)] Castelli, L., Pesenti, R., Ranieri, A., (2011). The design of a market mechanism to allocate Air Traffic Flow Management slots. *Transportation Research Part C* **19 (5)** 931–943

[Charnes et al., (1978)] Charnes, A., Cooper, W.W., Rhodes, E., (1978). Measuring the efficiency of decision making units. *European Journal of Operational Research* **2 (6)** 429–444.

[Charnes et al., (1994)] Charnes, A., Cooper, W.W., Lewin, A.Y., Seiford, L.M., (1994). Data Envelopment Analysis: Theory, Methodology, and Application. *Kluwer Academic Publishers, Dordrecht, The Netherlands.*

[Cheng (1998)] Cheng, Y., (1998). Hybrid simulation for resolving resource conflicts in train traffic rescheduling. *Computers in Industry* **35 (3)**, 233–246.

[Chiu et al., (2002)] Chiu, C. , C. Chou, J. Lee, H. Leung, Leung, Y., (2002). A constraint-based interactive train rescheduling tool. *Constraints* **7 (2)**, 167–198.

[Churchill et a., (2010)] Churchill, A.M., Lovell, D.J., Ball, M.O., (2010). Flight delay propagation impact on strategic air traffic flow management. *Transportation Research Records* **2177** 105–113.

[Cicerone et al., (2008)] Cicerone, S., Di Stefano, G., Schachtebeck, M., Schöbel, A., (2008). Dynamic algorithms for recoverable robustness problems. *ATMOS 2008 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems.*

[Clare and Richards (2011)] Clare, G., Richards, A.G., (2011). Optimization of taxiway routing and runway scheduling. *IEEE Transactions on Intelligent Transportation Systems* **12 (4)** 1000–1013

[Clarke and Solak (2009)] Clarke, J.P.B., Solak, S., (2009). Air traffic flow management in the presence of uncertainty. *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM (2009), Napa, California, USA.*

[Clausen et al., (2010)] Clausen, J., Larsen, A., Larsen, J., Rezanova, N.J., (2010). Disruption management in the airline industry - Concepts, models and methods. *Computers and Operations Research* **37 (5)** 809–821.

[Cooper et al., (2007)] Cooper, W.W., Seiford, L.M., Tone, K., (2007). Data Envelopment Analysis: A Comprehensive Text with Models, Applications, References and DEA-Solver Software, *2nd edn., Springer, New York, USA.*

[Cordeau et al., (1998)] Cordeau, J. F., Toth, P., Vigo, D., (1998). A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science* **32 (4)** 380–404.

[Corman (2010)] Corman, F., (2010). Real-time Railway Traffic Management: dispatching in complex, large and busy railway networks. *PhD Thesis, TRAIL Thesis Series T2010/14, The Netherlands.*

[Corman et al., (2009a)] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., (2009). Evaluation of green wave policy in real-time railway traffic management. *Transportation Research Part C* **17 (6)**, 607–616.

[Corman et al., (2009b)] Corman, F., Goverde, R.M.P., D'Ariano, A., (2009). Rescheduling dense train traffic over complex station interlocking areas. *Lecture Notes in Computer Science*, **5868**, 369–386.

[Corman et al., (2010)] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., (2010). A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B* **44 (1)** , 175–192.

[Corman et al., (2011a)] Corman, F., D'Ariano, A., Hansen, I.A., Pacciarelli, D., (2011). Optimal multi-class rescheduling of railway traffic. *Journal of Rail Transport Planning and Management*, **1 (1)** 14–24.

[Corman et al., (2011b)] Corman, F., D'Ariano, A., Pranzo, M., Hansen, I.A., (2011). Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transport. Planning and Technology* **34 (4)** 341–362.

[Corman et al., (2012a)] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., (2012). Optimal inter-area coordination of train rescheduling decisions, *Transportation Research Part E* **48 (1)** 71–88.

[Corman et al., (2012b)] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., (2012). Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C* **20 (1)**, 79–94.

[Corman et al., (2014a)] Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., (2014A). Dispatching and coordination in multi-area railway traffic management. *Computers and Operations Research* **44 (1)** 146–160.

[Corman et al., (2014b)] Corman, F., D'Ariano, A., Hansen, I.A., (2014B). Evaluating disturbance robustness of railway schedules, *Journal of Intelligent Transport Systems: Technology, Planning, and Operations* **18 (1)** 106–120.

[Corman et al., (2015a)] Corman, F., Pacciarelli, D., D'Ariano, A., Samà, M., (2015). Railway Traffic Rescheduling Taking into Account Minimization of Passengers Discomfort, In: F. Corman, S. Voss and R.R. Negenborn (Eds.), Computational Logistics, Springer. *Lecture Notes in Computer Science* **9335**, 602–616.

[Corman et al., (2015b)] Corman, F., Xin, J., Toli, A., Negenborn, R.R., D'Ariano, A., Samà, M., Lodweijks, G., (2015). Optimizing hybrid operations at large-scale automated container terminals, *Proceedings of the 4th International Conference on Models and Technology for Intelligent Transportation Systems*, Budapest, Hungary.

[Corman and Meng (2015)] Corman, F., Meng, L., (2015). A Review of Online Dynamic Models and Algorithms for Railway Traffic Management. *IEEE Transactions on Intelligent Transportation Systems* **16 (3)** 1274–1284.

[Corman and Quaglietta (2015)] Corman, F., Quaglietta, E., (2015). Closing the loop in railway traffic control: framework design and impacts on operations. *Transportation Research Part C* **54 (1)** 15–39.

[Corolli et al., (2014)] Corolli, L., Lulli, G., Ntaimo, L., (2014). The time slot allocation problem under uncertain capacity. *Transportation Research Part C* **46 (1)** 16–29.

[D'Ariano (2008)] D'Ariano, A., (2008). Improving Real-Time Train Dispatching: Models, Algorithms and Applications. *PhD Thesis, TRAIL Thesis Series T2008/6, The Netherlands.*

[D'Ariano et al., (2007a)] D'Ariano, A., Pacciarelli, D., Pranzo, M., (2007). A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research* **183 (2)** 643–657.

[D'Ariano et al., (2007b)] D'Ariano, A., Pranzo, M., Hansen, I.A., (2007). Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Trans. on Intelligent Transportation Systems* **8 (2)** (208–222.

[D'Ariano et al., (2008)] D'Ariano, A., Corman, C., Pacciarelli, D., Pranzo, M., (2008). Reordering and Local Rerouting Strategies to Manage Train Traffic in Real Time. *Transportation Science* **42 (4)** 405–419.

[D'Ariano (2009)] D'Ariano, A., (2009). Innovative Decision Support System for Railway Traffic Control. *IEEE Intelligent Transportation Systems Magazine* **1 (4)** 8–16.

[D'Ariano et al., (2010)] D'Ariano, A., D'Urgolo, P., Pacciarelli, D., Pranzo, M., (2010). Optimal sequencing of aircrafts take-off and landing at a busy airport. *Proceedings of the 13th International IEEE Annual Conference on Intelligent Transportation Systems*, pages 1569–1574. Madeira Island, Portugal.

[D'Ariano et al., (2012a)] D'Ariano, A., Pistelli, M., Pacciarelli, D., (2012). Effectiveness of optimal aircraft rerouting strategies for the management of disturbed traffic conditions. *Proceedings of the 91st Transportation Research Board Annual Meeting* 12–0428. Washington DC, USA.

[D'Ariano et al., (2012b)] D'Ariano, A., Pistelli, M., Pacciarelli, D., (2012). Aircraft retiming and rerouting in vicinity of airports. *IET Intelligent Transport Systems Journal* **6 (4)** 433–443.

[D'Ariano et al., (2014)] D'Ariano, A., Samà, M., D'Ariano, P., Pacciarelli, D., (2014). Evaluating the applicability of advanced techniques for practical real-time train scheduling. *Transportation Research Procedia*, **3 (1)** 279–288.

[D'Ariano et al., (2015)] D'Ariano, A. Pacciarelli, D., Pistelli, M., Pranzo, M., (2015). Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area. *Networks* **65 (3)** 212–227.

[Dear (1976)] Dear, R.G., (1976). *The Dynamic Scheduling of Aircraft in the Near Terminal Area.* Flight Transportation Laboratory (FTL). Report R76-9, Massachusetts Institute of Technology.

[Dellino et al., (2009)] Dellino G., Lino P., Meloni C., Rizzo A., (2009). Kriging metamodel management in the design optimization of a CNG injection system. *Mathematics and Computers in Simulation* **79 (8)** 2345–2360.

[Dessouky et al., (2006)] Dessouky, M., Lu, Q., Zhao, J., Leachman, R., (2006). An exact solution procedure to determine the optimal dispatching times for complex rail networks. *IIE Transactions*, **38 (2)** 141–152.

[Djokic et al., (2010)] Djokic, J., Lorenz, B., Fricke, H., (2010). Air traffic control complexity as workload driver. *Transportation Research Part C* **18(6)** 930–936.

[Dollevoet et al., (2014)] Dollevoet, T., Corman, F., D'Ariano, A., Huisman, D., (2014). An Iterative Optimization Framework for Delay Management and Train Scheduling. *Flexible Services and Manufacturing Journal* **26** (4), 490–515.

[Dorigo and Stützle (2004)] Dorigo, M., Stützle, T., (2004). Ant Colony Optimization, *MIT Press, Massachusetts Institute of Technology, Cambridge.*

[Dorndorf et al., (2007)] Dorndorf, U., Drexl, A., Nikulin, Y., Pesch, E., (2007). Flight gate scheduling: State-of-the-art and recent developments. *Omega* **35 (3)** 326–334

[Ernst et al., (2004)] Ernst, A., Jiang, H., Krishnamoorthy, M., Sier, D., (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research* , **153 (1)** 3–27.

[EU White Paper (2011)] European Commission, (2011). *White Paper. Roadmap to a Single European Transport Area – Towards a competitive and resource efficient transport system.*

[EU Statistical Pocketbook (2014)] European Commission, (2014). Statistical pocketbook (2014). *EU transport in figures*

[EU Explained - Transport (2014)] European Commission, (2014). Transport – Connecting Europe's citizens and businesses. *The European Union Explained*

[Eun et al., (2010)] Eun, Y., Hwang, I., Bang, H., (2010). Optimal Arrival Flight Sequencing and Scheduling Using Discrete Airborne Delays. *IEEE Transactions on Intelligent Transportation Systems* **11 (2)** 359–373

[Fan et al., (2012)] Fan, B., Roberts, C., Weston, P., (2012). A comparison of algorithms for minimising delay costs in disturbed railway traffic scenarios. *Journal of Rail Transport Planning & Management*, **2 (1–2)**, 23–33.

[Fang et al., (2015)] Fang, W., Yang, S., Yao, X., (2015). A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks. *IEEE Transactions on Intelligent Transportation Systems* . **16 (6)** 2997–3016.

[Faye (2015)] Faye, A., (2015). Solving the aircraft landing problem with time discretization approach. *European Journal of Operational Research* **242 (3)** 1028–1038.

[Fioole et al., (2006)] Fioole, P. J. , Kroon, L., Maróti, G., Schrijver, A., (2006). A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research* , **174 (2)** 1281–1297

[Furini et al., (2015] Furini, F., Kidd, M.P., Persiani, C.A., Toth, P., (2015). Improved rolling horizon approaches to the aircraft sequencing problem. *Journal of Scheduling* **18 (5)** 435–447.

[Giannettoni and Savio (2002)] Giannettoni, M., Savio, S., (2002). Traffic management in moving block railway systems: the results of the EU project COMBINE. *In J. Allen, R.J. Hill, C.A. Brebbia, G. Sciutto, S. Sone (Eds.), Computers in Railways VIII, 953–962. Southampton, UK: WIT Press.*

[Giannettoni and Savio (2004)] Giannettoni, M., Savio, S., (2004). The European Project COMBINE 2 to Improve Knowledge on Future Rail Traffic Management Systems. *In J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto, S. Sone (Eds.), Computers in Railways IX, 69–704. Southampton, UK: WIT Press.*

[Ginkel and Schöbel (2007)] Ginkel, A., Schöbel A., (2007). To wait or not to wait? The bi-criteria delay management problem in public transportation. *Transportation Science*, 41 (4), 527–538

[Glover (1986)] Glover, F., (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* **13 (5)** 533–549.

[Goossens et al., (2005)] Goossens, J. H. M., Van Hoesel, C. P. M., Kroon, L. G., (2005). On Solving Multi-type Railway Line Planning Problems. *European Journal of Operational Research* **168 (2)**, 403–424.

[Gorman (2009)] Gorman, M.F., (2009). Statistical estimation of railroad congestion delay. *Transportation Research Part E* **45 (3)** 446–456.

[Hancerliogullari et al., (2013)] Hancerliogullari, G., Rabadi, G., Al-Salem, A.H., Kharbeche, M., (2013). Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem. *Journal of Air Transport Management* **32 (1)** 39–48.

[Hansen et al., (2008)] Hansen, P., Mladenović, N., Moreno Pérez, J.A., (2008). Variable neighbourhood search: methods and applications. *4OR – Quarterly Journal of Operations Research* **6 (4)** 319–360.

[Hansen and Pachl (2014)] Hansen I.A., Pachl, J., (2014). Railway timetabling & operations: analysis, modelling, optimisation, simulation, performance evaluation. *Eurailpress, Hamburg, Germany.*

[Higgins et al., (1996)] Higgins, A., Kozan, E., Ferreira, L., (1996). Optimal scheduling of trains on a single line track. *Transportation Research Part B* **30 (2)** 147–158.

[Hozak and Hill (2009)] Hozak, K., Hill, J.A., (2009). Issues and opportunities replanning and rescheduling frequencies. *International Journal of Production Research* **47 (18)** 4955–4970.

[Hu and Chen (2005)] Hu, X.-B., Chen, W.-H., (2005). Receding horizon control for aircraft arrival sequencing and scheduling. *IEEE Transactions on Intelligent Transportation Systems* **6 (2)** 189–197

[Hu and Di Paolo (2008)] Hu, X.-B., Di Paolo, E., (2008). Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. *IEEE Transactions on Intelligent Transportation Systems* **9 (2)** 301–310.

[Hu and Di Paolo (2009)] Hu X.-B., Di Paolo E., (2009). An efficient genetic algorithm with uniform crossover for air traffic control. *Computers and Operations Research* **36 (1)** 245–259

[Hu and Di Paolo (2011)] Hu, X.B., Di Paolo, E., (2011). A Ripple-Spreading Genetic Algorithm for the Aircraft Sequencing Problem. *Evolutionary Computation* **19 (1)** 77–106.

[Huang (2006)] Huang, J., (2006). Using ant colony optimization to solve train timetabling problem of mass rapid transit. *Proceedings of the Joint Conference on Information Science (2006.*

[IATA (2014)] International Air Transport Association - IATA, (2014). Air Passenger Forecasts Global Report. *www.iata.org/pax-forecast*

[Jiang et al., (2014)] Jiang, Y., Xu, Z., Xu, X., Liao, Z., Luo, Y., (2014). A schedule optimization model on multirunway based on ant colony algorithm. *Mathematical Problems in Engineering*, Volume (2014, 11 pages, http://dx.doi.org/10.1155/2014/368208

[Kalai and Smorodinsky (1975)] Kalai, E., Smorodinsky, M., (1975). Other Solutions to Nash's Bargaining Problem. *Econometrica* **43 (3)** 513–518.

[Kanai et al., (2011)] Kanai, S., Shiina, K., Harada, S., Tomii, N., (2011). An optimal delay management algorithm from passengers viewpoints considering the whole railway network. *Journal of Rail Transport Planning & Management* **1 (1)** 25–37.

[Kecman et al., (2013)] Kecman P., Corman, F., D'Ariano, A., Goverde, R.M.P., (2013). Rescheduling models for railway traffic management in large-scale networks. *Public Transport: Planning and Operations* **5 (1)**, 95–123.

[Kim et al., (2009)] Kim, J., Kröller, A., Mitchell, J.S.B., Sabhnani, G.R., (2009). Scheduling Aircraft to Reduce Controller Workload. *Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems.* In J. Clausen. G. Di Stefano (Eds), Cop., Denmark.

[Kohl et al., (2007)] Kohl, N., Larsen, A., Larsen, J., Ross, A., Tiourine, S., (2007). Airline disruption management – Perspectives, experiences and outlook. *Journal of Air Transport Management* **13 (3)** 149–162.

[Kotynek and Richetta (2006)] Kotynek, B., Richetta, O., (2006). Equitable models for the stochastic ground-holding problem under collaborative decision-making. *Transportation Science* **40 (1)** 133–146

[Kroon et al., (2010)] Kroon, L.G., Maróti, G., Nielsen, L.K., (2014). Rescheduling of railway rolling stock with dynamic passenger flows. Erasmus Research Institute of Management, *Transportation Science* **49 (2)** 165–184.

[Kuchar and Yang (2000)] Kuchar, J.K., Yang, L.C., (2000). A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems* **4 (1)** 179–189.

[Lai et al., (2008)] Lai, Y.C., Ouyang, Y., Barkan, C.P.L., (2008). A rolling horizon model to optimize aerodynamic efficiency of intermodal freight trains with uncertainty. *Transportation Science* **42 (4)** 466-477.

[Lamorgese and Mannino (2013)] Lamorgese L., Mannino C., (2013). The track formulation for the Train Dispatching problem. *Electronic Notes in Discrete Mathematics* **41 (1)**, 559–566.

[Lamorgese and Mannino (2015)] Lamorgese, L., Mannino, C., (2015). An exact decomposition approach for the real-time Train Dispatching problem. *Operations Research*, **63 (1)** 48–64

[Larsen et al., (2014)] Larsen, R., Pranzo, M., D'Ariano, A., Corman, F., Pacciarelli, D., (2014). Susceptibility of Optimal Train Schedules to Stochastic Disturbances of Process Times. *Flexible Services and Manufacturing Journal* **26 (4)** 466–489.

[Liebchen et al., (2009)] Liebchen, C., Lbbecke, M., Möhring, R., Stiller, S., (2009). The concept of recoverable robustness, linear programming recovery, and railway applications, *Lecture Notes in Computer Science*, 5868, 1–27.

[Liu and Kozan (2009)] Liu, S.Q., Kozan, E., (2009). Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers and Operations Research* **36 (10)** 2840–2852.

[López-Ibáñez et al., (2011)] López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M., (2011). The irace package, Iterated Race for Automatic Algorithm Configuration. *Technical Report TR/IRIDIA/2011-004, IRIDIA, Université libre de Bruxelles, Belgium.*

[Lu and Yu (2012)] Lu, C.-C., Yu, V.F., (2012). Data envelopment analysis for evaluating the efficiency of genetic algorithms on solving the vehicle routing problem with soft time windows. *Computers & Industrial Engineering* **63 (2)** 520–529.

[Lu and Wu (2014)] Lu C.-C. J., Wu Y.-C. J., (2014). Evaluation of Heuristics Using Data Envelopment Analysis. *International Journal of Information Technology and Decision Making* **13 (4)** 795–810.

[Lusby et al., (2011)] Lusby, R.M., Larsen, J., Ehrgott, M., Ryan, D.M., (2011). Railway track allocation: models and methods. *OR Spectrum*, **33 (4)** 843–883.

[Lusby et al., (2013)] Lusby, R.M., Larsen, J., Ehrgott, M., Ryan, D.M., (2013). A set packing inspired method for real-time junction train routing. *Computers and Operations Research* **40 (3)** 713–724.

[Mannino and Mascis (2009)] Mannino, C., Mascis, A., (2009). Optimal real-time traffic control in metro stations. *Operations Research* **57 (4)** 1026–1039.

[Marin (2006)] Marin, A.G., (2006). Airport Management: Taxi Planning. *Annals of Operations Research* **143 (1)** 191–202

[Mascis and Pacciarelli (2002)] Mascis, A., Pacciarelli, D., (2002 Job shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* **143 (3)** 498–517.

[Mascis et al., (2008)] Mascis, A., Pacciarelli, D., Pranzo, M., (2008). Scheduling models for short-term railway traffic optimisation. *Lecture Notes in Economics and Mathematical Systems*, 600, 71–90.

[Mazzarello and Ottaviani (2007)] Mazzarello, M., Ottaviani, E., (2007). A traffic management system for real-time traffic optimisation in railways. *Transportation Research Part B* **41 (2)** 246–274.

[Meng and Zhou (2011)] Meng, L., Zhou, X., (2011). Robust single-track train dispatching model under a dynamic and stochastic environment: A scenario-based rolling horizon solution approach. *Transportation Research Part B* **45 (7)** 1080–1102.

[Meng and Zhou (2014)] Meng, L., Zhou, X., (2014). Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B* **67 (1)** 208–234.

[Moreno Pérez et al., (2003)] Moreno Pérez, J.A., Moreno-Vega, J.M., Rodríguez Martín, I., (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research* **151 (2)** 365–378.

[Mu and Dessouky (2011)] Mu, S., Dessouky, M., (2011). Scheduling freight trains traveling on complex networks. *Transportation Research Part B* **45 (7)** 1103–1123.

[Murça and Müller] Murça, M.C.R., Müller, C., (2015) Control-based optimization approach for aircraft scheduling in a terminal area with alternative arrival routes. *Transportation Research Part E* **73 (1)** 96–113.

[Nosedal et al., (2015)] Nosedal, J., Piera, M.A., Solis, A.O., Ferrer, C., (2015). An optimization model to fit airspace demand considering a spatio-temporal analysis of airspace capacity. *Transportation Research Part C* **61 (1)** 11–28.

[Pacciarelli and Pranzo (2004)] Pacciarelli, D., Pranzo, M., (2004). Production scheduling in a steelmaking-continuous casting plant. *Computers & Chemical Engineering* **28(12)**, 2823–2835.

[Palagachev et al., (2013)] Palagachev, K.D., Rieck, M., Gerdts, M., (2013). A Mixed-Integer Optimal Control Approach for Aircraft Landing Model, *Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transport Systems*. Dresden, Germany.

[Pellegrini and Savio (2000)] Pellegrini, F., Savio, S., (2000). Moving block and traffic management in railway applications: the EC project COMBINE. J. Allen, R.J. Hill, C.A. Brebbia, G. Sciutto, S. Sone (Eds.), Computers in Railways VII, WIT Press, Southampton (2000), 11–2.

[Pellegrini and Rodriguez (2013)] Pellegrini, P., Rodriguez, J., (2013). Single European sky and single European railway area: A system level analysis of air and rail transportation. *Transportation Research Part A* **57 (1)** 64–86.

[Pellegrini et al., (2012)] Pellegrini, P., Castelli, L., Pesenti, R., (2012). Metaheuristic algorithms for the simultaneous slot allocation problem. *IET Intelligent Transport Systems* **6 (4)** 453–462.

[Pellegrini et al., (2014)] Pellegrini, P., Marlière, G., Rodriguez, J., (2014). Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B* **59 (1)** 58–80.

[Pellegrini et al., (2015)] Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., (2015). RECIFE-MILP: An effective MILP-based heuristic for the real-time railway traffic management problem. *IEEE Transactions on Intelligent Transportation Systems* **16 (5)** 2609–2619.

[Pinol and Beasley (2006)] Pinol, H., Beasley, J.E., (2006). Scatter Search and Bionomic Algorithms for the aircraft landing problem. *European Journal of Operational Research* **171 (2)** 439–462.

[Pranzo et al., (2003)] Pranzo, M., Meloni, C., Pacciarelli, D., (2003). A new class of greedy heuristics for job shop scheduling problems. *Lecture Notes in Computer Science* **2647**, 223–236.

[Prevot et al., (2011)] Prevot, T., Homola, J., Martin, H., Mercer, J., (2011). Automated air traffic control operations with weather and time-constraints. *Ninth USA/Europe Air Traffic Management Research and Development Seminar*. Berlin, Germany.

[Psaraftis (1980)] Psaraftis, H.N., 1980. A dynamic programming approach for sequencing identical groups of jobs. *Operations Research* **28 (6)** 1347–1359

[Quaglietta et al., (2016)] Quaglietta, E., Pellegrini, P., Goverde, R.M.P., Albrecht, T., Jaekel, B., Marlière, G., Rodriguez, J., Dollevoet, T., Ambrogio, B., Carcasole, D., Giaroli, M., Nicholsonh, G., (2016). The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C* **63 (1)** 23–50.

[Ravizza et al., (2013)] Ravizza, S., Chen, J., Atkin, J.A.D., Burke, E.K., Stewart, P., (2013). The trade-off between taxi time and fuel consumption in airport ground movement. *Public Transport* **5 (1)** 25–40.

[Rodriguez (2007)] Rodriguez, J., (2007). A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B* **41 (2)** 231–245.

[Sadeh et al., (1995)] Sadeh, N., Sycara, K., Xiong, Y., (1995). Backtracking techniques for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence* **76 (1–2)** 455–480.

[Sadeh et al., (1996)] Sadeh, N., Foxb, M.S., (1996). Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence* **86 (1)** 1–41.

[Şahin (1999)] Şahin, İ., (1999). Railway traffic control and train scheduling based on inter-train conflict management. *Transportation Research Part B* **33 (7)** 511–534.

[Salehipour et al., (2009)] Salehipour, A., Moslemi Naeni, L., Kazemipoor, H., (2009). Scheduling aircraft landings by applying a variable neighborhood descent algorithm: Runway-dependent landing time case, *Journal of Applied Operational Research* **1 (1)** 39–49.

[Salehipour et al., (2013)] Salehipour, A., Modarres, M., Moslemi Naeni, L., (2013). An efficient hybrid meta-heuristic for aircraft landing problem. *Computers and Operations Research* **40 (1)** (207–213.

[Samà et al., (2013a)] Samà, M., D'Ariano, A., Pacciarelli, D., (2013). Rolling Horizon Approach for Aircraft Scheduling in the Terminal Control Area of Busy Airports. *Transportation Research Part E* **60 (1)** 140–155.

[Samà et al., (2013b)] Samà, M., D'Ariano, A., D'Ariano, P., Pacciarelli, D., (2013). *Scheduling models for optimal aircraft traffic control at busy airports: tardiness, priorities, equity and violations considerations*, In: Tech. Rep. RT-DIA-205-2013, Dipartimento di Ingegneria, Roma Tre University.

[Samà et al., (2014)] Samà, M., D'Ariano, A., D'Ariano, P., Pacciarelli, D., (2014). Optimal aircraft scheduling and routing at a terminal control area during disturbances. *Transportation Research Part C* **47 (1)** 61–85.

[Samà et al., (2015)] Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D., (2015). Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas. Technical Report RT-DIA-213-2015, pages 1–41, Roma Tre University, Department of Engineering, Section of Computer Science and Automation, Rome, Italy.

[Sarkis (2007)] Sarkis, J., (2007). Preparing Your Data for DEA, In: Joe Zhu, Wade D. Cook (Eds.). *Modeling Data Irregularities and Structural Complexities in Data Envelopment Analysis*, 305–320.

[Sato et al., (2013)] Sato, K., Tamura, K. Tomii, N., (2013). A MIP-based timetable rescheduling formulation and algorithm minimizing further inconvenience to passengers. *Journal of Rail Transport Planning Management* **3 (3)** 38–53.

[School (2005)] Scholl, S., (2005). Customer-Oriented Line Planning. *PhD thesis, University of Kaiserslautern*.

[Schöbel and Kratz (2009)] Schöbel, A., Kratz, A., (2009). A Bicriteria Approach for robust timetabling. *Lecture Notes in Computer Science*, 5868, 119–144.

[SCI "Rail Transport Markets – Global Market Trends (2014-2023"] SCI Verkehr GmbH, (2014). *Rail Transport Markets – Global Market Trends (2014-2023: Importance and Dynamism of the Worldwide Rail Transport Markets and Their Drivers*

[Sels et al., (2014)] Sels, P., Vansteenwegen, P., Dewilde, T., Cattrysse, D., Waquet, B., Joubert, A., (2014). The train platforming problem: The infrastructure management company perspective. *Transportation Research Part B* **61 (1)** 55–72.

[Sölveling et al., (2010)] Sölveling, G., Solak, S., Clarke, J.B., Johnson, E.L., (2010). Scheduling of runway operations for reduced environmental impact. *Transportation Research Part D* **16 (2)** 110–120.

[Soomer and Franx (2008)] Soomer, M.J., Franx, G.J., (2008). Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research* **190 (1)** 277–291

[Soomer and Koole (2008)] Soomer, M.J., Koole, G.M., (2008). *Fairness in the Aircraft Landing Problem.* Technical Report, Department of Mathematics, Vrije Universiteit Amsterdam.

[SysML] SysML Open Source Specification Project. Sysml.org, (2015).

[Solnon and Bridge (2006a)] Solnon, C., Bridge, D., (2006). An Ant Colony Optimization Meta-Heuristic for Subset Selection Problems. In *System Engineering using Particle Swarm Optimization.* Nova Science publisher, 7–29.

[Solnon and Bridge (2006b)] Solnon, C., Bridge, D., (2006). A study of ACO capabilities for solving the maximum clique problem. *Journal of Heuristics*, **12 (3)** 155–180.

[Solveling et al., (2011)] Solveling, G., Solak, S., Clarke, J.-P., Johnson, E., (2011). Scheduling of runway operations for reduced environmental impact. *Transportation Research Part D* **16 (2)** 110–120.

[Steria et al., (2011)] Steria, G.G., Allignol, C., Durand, N., (2011). The influence of uncertainties on traffic control using speed adjustments. *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM (2011), Berlin, Germany.*

[Stützle and Hoos (2000)] Stützle, T., Hoos, H.H., (2000). MAX-MIN ant system. *Future Generation Computer Systems* **16 (8)** 889–914.

[Szpigel (1973)] Szpigel, B., (1973). Optimal Train Scheduling on a Single Track Railway. *Operational Research* **72** 343–352

[Takeuchi et al., (2006)] Takeuchi, Y. Chikara, H. Tomii N., (2006). Robustness Indices based on passengers' utilities. *Proceedings of the 7th World Conference on Railway Research*, Montréal, Canada.

[Tavakkoli-Moghaddam et al., (2012)] Tavakkoli-Moghaddam, R., Yaghoubi-Panah, M., Radmehr, F., (2012). Scheduling the sequence of aircraft landings for a single runway using a fuzzy programming approach. *Journal of Air Transport Management* **25(1)** 15–18.

[Taylor and Wanke (2011)] Taylor, C. and Wanke, C., (2011). Dynamically Generating Operationally-Acceptable Route Alternative Using Simulated Annealing. *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM (2011).* Berlin, Germany.

[Tomii et al., (2005)] Tomii, N., Tashiro, Y., Tanabe, N., Hirai, C., Muraki, K., (2005). Train rescheduling algorithm which minimizes passengers' dissatisfaction. In: Ali, M., Esposito, F., (Eds.), Innovations in Applied Artificial Intelligence, *Lecture Notes in Artificial Intelligence*, **3533** 829-838

[Törnquist (2012)] Törnquist Krasemann, J., (2012). Design of an effective algorithm for fast response to re-scheduling of railway traffic during disturbances. *Transportation Research Part C* **20 (1)** 62–78.

[Törnquist and Persson (2007)] Törnquist, J., Persson, J., (2007). N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B* **41 (3)**, 342–362.

[Tsuji et al., (2012)] Tsuji, Y., Kuroda, M., Kitagawa, Y., Imoto Y., (2012). Ant Colony Optimization Approach for Solving Rolling Stock Planning for Passenger Trains. *(2012 IEEE/SICE International Symposium on System Integration (SII), Kyushu University, Fukuoka, Japan, December 16-18.*

[Vaidyanathan et al., (2008)] Vaidyanathan, B., Ahuja, R.K., Orlin, J.B., (2008). The locomotive routing problem. Transportation Science **42 (4)** 492–507.

[Wegele and Schnieder (2004)] Wegele, S., Schnieder, E., (2004). Automated dispatching of train operations using genetic algorithms. *Computers in Railways IX*. WIT Press, Southampton, UK, (2004, 775–784.

[Wegele et al., (2007)] Wegele, S., Slovák, R., Schnieder, E., (2007). Real-time decision support for optimal dispatching of train operation. In: I.A. Hansen, A. Radtke, J. Pachl, E. Wendler (Ed.), Proc. of the 2nd Internat. Conf. on Railway Operations Modelling and Analysis, Hannover, Germany.

[Witt and Voss (2010)] Witt, A., Voss, S., (2010). Job Shop Scheduling with Buffer Constraints and Jobs Consuming Variable Buffer Space. *Lecture Notes in Business Information Processing*, **46** 295–307. Editors: W. Dangelmaier, A. Blecken, R. Delius, S. Klöpfer. Book Title: Advanced Manufacturing and Sustainable Logistics. Publisher: Springer Berlin Heidelberg.

[Xin et al., (2015)] Xin, J., Negenborn, R.R., Corman, F., Lodewijks, G., (2015). Control of interacting machines in automated container terminals using a sequential planning approach for collision avoidance. *Transportation Research Part C* **60 (1)** 377–396.

[Yuan (2006)] Yuan, J., (2006). Stochastic Modelling of Train Delays and Delay Propagation in Stations. PhD Thesis, TRAIL Thesis Series T2006/6, Delft University of Technology, Delft, The Netherlands.

[Zhang (2008)] Zhang, Y., (2008). *Real-time Inter-modal Strategies for Airline Schedule Perturbation Recovery and Airport Congestion Mitigation under Collaborative Decision Making (CDM).* University of California Transportation Center, UCTC Dissertation No. 154.

[Zhan and Zhang (2010)] Zhan, Z-H., Zhang, J., Li, Y., Liu O., Kwok, S.K., Ip, W.H., Kaynak, O., (2010). An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem. *IEEE Transactions on Intelligent Transportation Systems* **11 (2)** 399–412.

[Zwaneveld et al., (2011)] Zwaneveld, P. J., Kroon, L. G, Van Hoesel, S. P. M., (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, **128 (1)**, 14–33.