Roma Tre University
Ph.D. in Computer Science and Engineering

Combinatorial structures for communication networks

Ezio Sperduto

Combinatorial structures for communication networks

A thesis presented by
Ezio Sperduto
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Informatics and Automation

March 2009

Committee:
Prof. Geppino Pucci
Prof. Francesco Scarcello
Prof. Riccardo Torlone

Reviewers:
Prof. Pitu Mirchandani
Prof. Federico Malucelli

Advisor:
Gaia Nicosia

... to my father.

Preface

This thesis deals with a class of theoretical problems arising in applications in communication networks. The dissertation is mainly divided in two parts.

In the first part, we attempt to solve a set of survivability network design problems. Network survivability refers to the guarantees that a communication network provides in the event that one or more failures occur. An attack or failure can significantly reduce the capability of the communication network to efficiently deliver basic services to users. In several cases, when a failure occurs, the network operators are interested in restoring traffic by re-routing it through different links. Since re-routing traffic can be rather expensive and may cause delays in transmissions, a key property is that of requiring that traffic which is not affected by the failure is not redirected in failure situations.
We study the problem of determining whether a given network, where the traffic is commonly routed on the edges of a shortest path tree (e.g. Ethernet networks with the Spanning Tree Protocol), may satisfy the above mentioned property. We provide computational complexity results for directed and undirected networks. In particular, for the directed case, we prove that such problem is in general NP-hard and that it remains NP-hard also in some special cases. Moreover, we show how to assign weights to the links of the network in order to configure a routing topology with the above mentioned property.

In the second part of the thesis, we deal with a problem regarding broadcasting in telecommunication networks. We investigate a new version of the well known *Minimum Broadcast Time problem* which has been deeply studied in the past, since broadcasting is a basic primitive in the communication networks area. Fundamental requirements for a broadcast process are that it completes in the quickest way and that, at the end of the procedure, all the peers in the network are informed. In this thesis we deal with an objective

function that takes into account the quality of the service associated with the broadcast, namely the minimization of the average broadcast time of the peers. We show that the considered version of the broadcast problem is an NP-hard problem. Indeed, the problem becomes polynomially solvable, if the instance graph is a tree. We also provide a distributed approximation algorithm for our version of the broadcast problem, in which every network node does not know the network topology.

The thesis is organized in the following way.

In Chapter 1, we introduce the reader to the communication network research area. We treat some of the most important topics regarding the communication network literature, and cite several fundamental results that are still now mile stones of network research field.

In Chapter 2, we provide an in-depth analysis of survivability network design issues.

In Chapter 3, we describe and discuss the complexity of a survivability network problem. Hence, we provide both the hardness result for the general setting and the proof of the polynomial solvability for some special cases.

In Chapter 4, we extend the survivability properties proved for the case of single link failure in Chapter 3 to multiple links failure.

In Chapter 5, we introduce the fundamental issues of network broadcasting. We also discuss the differences among the communication models used in broadcasting literature.

In Chapter 6, we deal with a variant of the minimum broadcast time problem: the *minimum service time broadcast problem*. We prove that the problem, in a centralized scenario, is in general NP-hard, but polynomial time solvable in special cases like in trees.

In Chapter 7, we analyze the *minimum service time problem* in a distributed scenario and provide a fast and simple approximation algorithm.

Acknowledgments

This work is the final result of a three years course that could not be possible without the support of many people. First of all, I need to thank my advisor Gaia Nicosia who guided me during these years, teaching how to deal with the difficulties of the research. I am very grateful to Prof. Gianpaolo Oriolo (University of Tor Vergata) for his helpful collaboration on the work on survivability networks. Further, I wish to express my gratitude to Prof. Angelo Monti (University of La Sapienza) for the precious time he dedicated to me. My sincere gratitude goes to Prof. Fernando Nicolò, who introduced me in the research group of Operations Research and Automation.

Among the people that I have to thank there are my Ph.D. colleagues: Michele, Luca, Annalisa, Andrea, Laura, Enrico, Yuri. They are very bright and capable and in many cases our ideas exchange enriched my life. During this period, I cannot forget to thank my friends Mario and Lucio, for their support, and Emanuela, who believed in me.

I owe special thanks to my family that permitted me to complete my studies.

Contents

Chapter 1

Introduction

The field of *Network Design*, with its many variants, is one of the most active research areas in computer science involving researchers from graph theory, combinatorial optimization, game theory, system and information theory. Mathematical modeling of networks plays a vital role in the understanding of computer and telecommunication networks and provides questions and solutions regarding the allocation of network resources, analysis and effects of competitive and/or cooperative agents, Internet protocols, wireless network protocols, network dynamics, queuing systems, performance optimization, and network traffic and topology. These models explain fundamental performance limits and trade-offs in practical scenarios. In addition, new problems in this area are constantly proposed by practitioners working in various aspects of network design such as construction, routing and staged deployment. Furthermore, many new design paradigms such as ATM, Ad hoc, Optical and Wireless networking add rich new flavors to existing problems. On the other hand, many of the key algorithmic challenges in the context of the internet, the largest network in the world, require considering the objectives and interests of the different participants involved. These include problems ranging from pricing goods and resources, to improving search, to routing, and more generally to understanding how participants can support the improving of the overall system. Recent results show a strong relation between network design and combinatorial optimization, and techniques from each seem well-poised to help with key problems of the other.

Network design area is too big to be treated in a comprehensive way in this thesis, so we limit to describe several intresting aspects and problems. In

particular, we introduce network design by a more theoretical point of view: we use graph theory to model network design problems. Therefore, in network design, we can identify a division in smaller basic subareas, among which we list the following:

- spanning trees problems;
- routing problems;
- survivability network problems;
- broadcast problems.

Usually, real applications in network design are a complex mix of basic questions and constrains. Further, if we attempt to solve a real network problem, we must analyze in a deep way the network architectures, so that we may take advantage of their features. In this discussion we skate over technical issue and deal only with theoretical questions.

1.1 Spanning tree problems

Given a connected, undirected graph, a spanning tree of that graph is a subgraph which is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a weight to each edge, which is a number representing how unfavorable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. A minimum spanning tree or minimum weight spanning tree is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of minimum spanning trees for its connected components. Problem to find a minimum spanning tree, over a weighted graph, was shown in the past to be solvable in polynomial time. We can solve the problem using the well known algorithms of Kruskal [56] or Prim [68]. But there are several versions of the minimum spanning tree problem that are under attention of many researchers, and that are more complex than the previous one.

The problem of computing the minimum tree spanning any k vertices can be viewed in the general setting of finding a "best" subgraph spanning k vertices and satisfying certain property. Clearly, all problems for which the properties can be checked in polynomial time are polynomially solvable, for fixed k, by

examining all k vertex subsets. In particular if we require that the subgraph be connected and of the least possible weight, then we are really looking for the minimum weight tree that spans k vertices. Thus, given an undirected graph G, a weight function $w : V \to Z_+$, and an integer $k \leq |V|$, we want to find a spanning tree over G, that spans only k vertices, such that its total weight is lesser or equal than each other spanning tree that spans k vertices over G. This problem was shown to be NP-hard by Ravi *et al.* in [73]. A $3\sqrt{k}$-approximation algorithm for this problem was first given in [73], and this has been, after, improved to a $O(\log^2 k)$-approximation algorithm by Awrebuch, Azar, Blum and Vempala [9]. The best approximation algorithm, with an approximation ratio of 3, for the minimum k-spanning tree was provided by Garg (see [40]).

An other minimum spanning tree problem with constraint is the the minimum degree spanning tree (MDST) problem, in which we find a spanning tree for a graph G with n vertices whose maximal degree is the smallest among all spanning trees of G. This problem was easily shown to be NP-hard. Let T^* be an optimal spanning tree, whose maximal degree is Δ^*. Fürer and Raghavachari [35] have given a parallel approximation algorithm which produces a spanning tree of degree $O(\Delta^* \log n)$. The same authors, in [36], prove that the algorithm provided in the paper guarantees an approximation ratio, for the MDST problem, of $\Delta^* + 1$.

Given an arbitrary weighted graph with a distinguished vertex subset, the *Steiner Tree Problem* asks for a minimum-cost subtree spanning the distinguished vertices. Steiner trees are important in various applications such as VLSI routing, wirelength estimation, phylogenetic tree reconstruction in biology, and network routing. The Steiner Tree Problem is NP-hard even if we consider the instance graph in the Euclidean or rectilinear metrics [39]. Arora established that Euclidean and rectilinear minimum-cost Steiner trees can be efficiently approximated arbitrarily close to optimal [5]. On the other hand, unless $P = NP$, the Steiner Tree Problem in general graphs cannot be approximated within a factor of $1 + \epsilon$ for sufficiently small $\epsilon > 0$ [13, 20]. For arbitrary weighted graphs, the best Steiner approximation ratio achievable within polynomial time was gradually decreased from 2 to $1,59$ in a series of works [82, 91, 12, 92, 69, 50, 48]. Actually, the best approximation algorithm for the Steiner Tree Problem guarantees an approximation ratio of $1+(\ln 3)/2$, [75].

1.2 Routing problems

In the field of network design, a routing process is the procedure of moving information across an interconnected system, from a source to a destination. Usually, along the way, there are at least one network peer, so that the routing protocol uses metric to identify which path is the *best* for the information to travel. In order to dispatch the information, by a more technical point of view, a routing protocol need to organize the information, so that different kinds of network architectures can communicate among them.

In the area of network routing, there is an enormous number of kinds of routing problems. Many works, in literature, treat the performance evaluation of routing algorithms. Here, for the sake of shortness, we cite only some combinatorial problems, related to routing problems.

In a theoretical scenario, in which the communication network is modeled as a graph, the best known problem is the *Traveling Salesperson Problem*. In a TSP instance, we are given a number of cities (the network peers) and a distance function over each pair of cities. The question is to find the smallest distance round-trip (cycle) that visits each city exactly once, and that returns, at the end, to the starting city. Since TSP has a mass of application in the real world, this problem has been well studied by many researchers belonging to the operations research and discrete combinatorial areas. There is a huge number of papers that treat the TSP issue, for an intriguing history we refer to [57], but in this discussion we limit to describe only some peculiar aspects. In 1979, TSP was shown to be NP-complete, [39], and, in 1987, Orponen and Mannila, [64], proved that TSP can not have a constant factor approximation algorithm (is NPO-complete). Even if this simple problem is so hard to approximate, many TSP variants have been studied alot in the past, with better approximation results. We cite, among them, the metric TSP and the Euclidean TSP. The metric TSP is the version of the TSP in which the distance function among the vertices agrees with the triangular inequality. The metric TSP (also known as Δ-TSP) was shown to be APX-complete. The Christofides approximation algorithm (see [18]) guarantees an approximation ratio of 3/2 for this TSP variant. A more restrictive version of the TSP is the Euclidean TSP. In this problem, we are given a set of points in the Euclidean two-dimensional space, so that the distance among the nodes agrees with all the geometric property of the plane. In 1996, Sanjeev Arora proved the best result about the approximation for the Euclidean TSP (or geometric TSP). He has shown, in [5], that the geometric TSP has a polynomial time approximation scheme (PTAS).

1.3 Survivability network problems and broadcasting problems

Since this two areas, in network design, are of special interest for this thesis, we prefer to treat them apart from the general introduction. We analyze the survivability network issue in Chapter 2, and we describe the broadcast problems in Chapter 5.

Chapter 2

Introduction to survivability problems

With the increase in size, complexity, and speed of communication, network performance under failures or attacks has become a great concern in the telecommunication industry. A failure or attack can significantly reduce the capability of the communication network to efficiently deliver service to users. This scenario contributed to the growth of the discipline named survivability network design. Network survivability refers to the trustworthy of the overall network to provide communication in the event of one or more failures in the network occur. The term fault-tolerant is often used to refer to how reliable a particular component of a network is (e.g., a computer, a switch or a router). The term fault-tolerant network, on the other hand, refers to how resilient the network is against the failure of a component. There are several techniques to project a failure resilient network. Depending on the type of the network we want design, there is a different strategy. Also, the survivability of a network is concerned with the ability of the network to provide a defined degree of assurance that the system will continue to function during and after a natural or man-made disturbance. In some cases, the survivability of a network is the capability of the network to remain connected after the failures. In more applicative cases, survivability is a more complex concept, that is perceived as a composite measure consisting of both network failure duration and failure impact on the network. Since several researchers, in their papers, have used different definition of survivability, we refer to the tech report of Knight and Sullivan (see [52]), that is a good dissertation on the meaning of survivability.

In that work, they attempt to find a general survivability definition that can be applied to any system.

But, the different kinds of network architecture have led to many different applications in survivability network design. A lion's share of the network survivability literature tells about wireless networks. The motivation of this literature trend, as well as in many other computer science fields, is imputable to the fall of the price of the wireless transmitters and, as a consequence, the dissemination of them in a mass of products. The wireless communication scenario has worsened a lot the failure survivability issues. The mobility of the peers and the variability of other connecting factors result in a network with a potentially rapid and unpredictable changing topology (*ad-hoc* network). In [80], the authors explain several options that the connection providers for mobile devices must consider to decrease the number of network failures and to cope with failures when they occur. In particular, they discuss strategies to improve network survivability and classify them into three categories:

- Prevention,
- Network design and capacity allocation,
- Traffic management and restoration.

Regarding mobile communication, in wireless network area, there is a well known network architecture, named Personal Cellular Service (PCS) network. We refer to [71, 84] for an in-depth examination of survivability issues in PCS networks. Moreover, in [71], the authors, with a simulation model, study a variety of failure scenario over this type of mobile networks. In an ad-hoc wireless network there are many parameters that combine to bring about the network functionality, e.g. transmission range, average velocity of the peers, power consumption. Very important is also the density of the peers over the terrain. A experimental study of which combinations of these parameters guarantee that network works, is done in the paper of Paul *et al.* [66]. Furthermore, in [60], we can find a framework for the maximum survivability routing, that, considering parameter specificactions, permits to evaluate the limit until network connectivity is preserved. After the break of the network connectivity, there is the impossibility to route messages. Thus, it becomes a fundamental issue, in survivability area, to develop good restoration strategies. For instance, we refer to the work of Wang *et al.* [86], that proposes two failure-handling schemes to recover connectivity in ATM networks. Every restoration scheme works under suitable hypothesis, first of all, the type of failure that it attempts to handle.

Therefore, it is important to classify failures, so that it is possible to quantify the survivability of the network. Considering wireless ad-hoc networks, in [16] there is an extensive discussion about the network survivability performance evaluation by a quantitative measure.

While the previous works on network survivability provide a good description of the concept of survivability, they do not have the mathematical issue to lead to a topological property in network design. We know that, if a network (a graph) is k-edge-connected, after the failure of $k-1$ links, the network remains functional. Analogously, if a network is k-vertex-connected, after the failure of $k-1$ peers, the network remains functional. In the first case, the problem reduces to solve an instance of the well known minimum cut problem. The algorithmic problem of testing k-connectivity has been well studied in the past (e.g. in [88, 51, 49, 67]). Considering the theoretical side of the survivability issue, we must cite the survivability telecommunication network design problem. Since we prefer to discuss this problem apart from the other topics, we deepen it in the next section.

2.1 Survivability and the telecommunication network design problem

Interesting in network survivability issue, is the application to the telecommunication network design problem. Given an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of links, the *telecommunication network design problem* is to choose integer multiples of a capacity unit on the links and route all the traffic demands so that the total capacity installation cost is minimized. In many networks (e.g. ATM networks), the capacity is considered undirected even if flow is directed. With respect to telecommunication network design problem, a network is said to be survivable if all of the demands can be routed after the failure of any one of its links. Usually, a link failure is considered as the event of decreasing the capacity of the link to zero. Since in telecommunication networks the probability of two components failing simultaneously is very small, designing a network protected against single component failures is considered satisfactory. In this case 2-edge-connectivity of the underlying graph G is a necessary condition for the survivability of the network, but it is clearly not sufficient. In order to ensure that the flow on the network can be rerouted in case of a failure, sufficient spare (excess) capacity must be available on the working links of the network (see [2, 70]).

The capacitated survivable network design problem can be formulated, with

integer linear programming, as a multicommodity network flow problem, for each failure scenario, linked by integral capacity variables [3]. However, there are at least two reasons as to why such a model is not used in practice. The first one is that it has a cubic number of variables and constraints in the number of edges for a complete network and is, therefore, impractical for designing networks except for very small instances. The second reason is that its solution involves globally rerouting flow in the network whether the flow of a commodity is disrupted by the failed edge or not. Solutions with minimal changes to no-failure flows are preferred because it is highly undesirable in practice to manipulate unaffected flow while restoring affected flows. Therefore, a number of practical models and strategies have been developed for designing survivable networks that admit *local* rerouting of flow on a failed edge; see for instance [4, 10, 43, 90]. We refer the reader to [81] for a well done survey on the survivable network design problem.

Traditionally, in order to design and implement survivable networks, one uses some variant of the following two different strategies: *protection* or *restoration*. Protection techniques completely identify ahead time the routes that disrupted flows will take and the capacities that will be used. Restoration techniques determine which available capacity will be used for a specific failure (and the routes that will be used for each affected demand) when the failure occurs. *Dedicated protection* techniques [4, 19, 43] install and assign spare capacity specifically for each commodity to protect it against the different possible failures, i.e., spare capacity is dedicated to a particular commodity. A significant reduction in the amount of spare capacity can be achieved by using a *shared* protection strategy [3, 4, 47] when dealing with failures. In a shared protection scheme, instead of preassigning spare capacity to protect each commodity of the network independently, spare capacity is shared by more than one commodity, and used as required to restore the disrupted flow. Survivable networks design strategies can also be broadly classified into frameworks: *hierarchical* (non-joint) and *integrated* (joint). The hierarchical one (for instance [47, 76]) involves a two-stage approach; first, no-failure routing and working capacities are determined, followed by rerouting of disrupted flows and spare capacities. Solving these two problems simultaneously in an integrated framework provides significant savings in installed capacity [61].

In [7], a heuristic method to solve survivable network design problem is proposed. It consists of a scheme that explicitly introduce spare capacities on directed cycles of the network, which are used to reroute disrupted flow under failure.

A study of the polyhedra of splittable and unsplittable single arcset relax-

ations of multicommodity flow capacitated network design problems is developed in [8].

2.2 Local restoration problems

The Spanning Tree Protocol (STP) and its several variants, are a family of routing techniques used in network architectures like Ethernet. An STP routing protocol routes the traffic on a *spanning tree*, namely, the shortest path tree for some special node r of the network, the *root*, with respect to suitable arc costs set by network operators. In the context of STP family protocols, in the last years efficient protocols have been standardized, such as Rapid Spanning Tree Protocol (RSTP) and Multiple Spanning Tree Protocol (MSTP)[1], so that the design of networks based on such routing protocols is receiving a lot of attention (see e.g. [53, 65, 78]).

The problem of designing tree-based routing strategies that are strongly resilient concerns, essentially, the maintenance of the connectivity of the network. Since the tree-based routing topology uses just a small set of the communication links of the underlying network, in these cases a restoration strategy consists, in the event of a link failure, in a procedure that finds a set of new links that restore topology connectivity. In this scenario, *fault-tolerant* communication networks are required, to guarantee that traffic demands can be re-routed even in failure situations. In particular, since re-routing of traffic can be rather expensive and/or may cause long delay in a message's transmission [31, 62], in order to meet QoS constraints, a key property of the re-routing strategy is the so-called *strong resilience* [14] or *minimum service disruption property* [22], requiring that traffic, which is not affected by the failure, is not redirected. In other words, under the minimum service disruption property, the number of traffic demands that are re-routed is minimized. Further, in [22], the authors study the relationship between load balance and service disruption. They propose a MSTP links weights assignment that implements any desired spanning tree, minimizing the number of extra links needed to re-route traffic demands affected by links failures.

Many authors have dealt with problems regarding the links weight setting on communication networks. When dealing with OSPF networks this problem is usually called *traffic engineering* and consists, roughly speaking, in determining a link weight system for a given objective. Most of the papers on this topic aim at balancing the network load by minimizing certain cost functions (see for instance [32, 33, 63, 87]). In particular, in [32] the authors show that

optimizing link weights to balance load for a given traffic matrix is NP-hard and they propose a local search heuristic. In [87] the authors prove that there exists a set of weights in OSPF such that the resulting shortest paths are the same as the optimal routes derived from the optimization model where traffic can be split among equal cost paths. Such a set of link weights can be derived by solving the dual of a linear programming formulation. Finally, in [63] the same problem is address however, here the authors consider also the possibility that a link may fail and therefore try to choose weights so that overloads during transient link failures are minimized.

In case of single link failure, the problem of finding the best link which reconnects the communication topology (*swap arc*) with efficient techniques was studied by Nardelli *et al.* [62] and, with distributed algorithms, by Flocchini *et al.* [31]. In particular, the problem of minimum service disruption in networks using the Multiple Spanning Tree Protocol has been addressed by De Sousa *et al.* in [22]. In the next two chapter we examine, in a deeper way, network design techniques to provide a quick restoration procedure in case of single or multiple link failures.

Chapter 3

Single link failure restoration

In network survivability, one of the most common failure situation that can affect the functioning of an interconnected system, is a link break. As we formerly said, a link failure may interrupt the delivery of basic services, specially in SPT routing protocols. Thus, we are interested in developing a survivable network design strategy for SPT protocol family.

Since in these routing protocols there is a good performance implementation of the shortest path tree routine, the standard way to use an SPT protocol is to assign a weight to each edge in the network, so that the routine builds the communication topology depending on the operator weights assignment. Therefore, the engineers set low weights for suitable edges and high weights for bad (expensive to use) edges. Furthermore, we are interested in planning a network such that, there exists a weight assignment that allows to build a "robust" tree routing topology. With the term robust we mean that the new routing topology built after a link failure, differs from the first one for at most one edge. Hence, we formally provide, by proving several properties, a criterion to identify whether a network has this survivability property or not.

We consider the following setting. We are given a communication network $G = (V, E)$, that we first suppose to be undirected. Consider $|E| + 1$ *scenarios*, each corresponding either to the failure of some link $e \in E$, the *back-up scenario*, or to the no-failures case, the *primary scenario*. For each scenario, as in SPT protocols, traffic will be routed on a spanning tree of the network that is operational in that scenario. Therefore, in the primary scenario, traffic will be routed on a spanning tree $T(\emptyset)$ of G; in the back-up scenario corresponding to the failure of link $e \in E$, traffic will be routed on a spanning tree $T(e)$ of

$G' = (V, E \setminus \{e\})$. (In the following, with an abuse of notation, we indicate with $G \setminus \{e\}$ the subgraph $G' = (V, E \setminus \{e\})$.)

By SPT protocols setting, we have a communication network G and a root node r, so that, by our choice of the *cost function* $w : E \mapsto \mathcal{Z}_+$, we get $T(\emptyset)$ as a shortest path tree of G, while $T(e)$, $e \in E$, as a shortest path tree of $G \setminus \{e\}$. The spanning tree $T(\emptyset)$ is called the *primary tree*; the paths in the primary tree are called *primary paths* and the primary path between u and v is denoted by $P_{uv}(\emptyset)$.

To clarify our way to proceed, we now explain the chapter organization. We, first, show that an undirected network G has the 1-restoration property *if and only if* it is 2-edge-connected. We also give a different proof that every spanning tree T of a 2-edge-connected undirected network G has the 1-restoration property: the proof includes a very simple procedure for assigning the costs on the edges so that T has the 1-restoration property. We then deal with directed networks. We show that deciding whether a directed network G has the 1-restoration property is NP-hard (therefore, this decision problem can unlikely be solved by an algorithm with time-complexity bounded by a polynomial in the size of the network), already when each node of G has distance at most 2 from the root r. On the sunny side, the problem is easy when each node has distance 1 from r and it is easy to check whether a given spanning arborescence T has the 1-restoration property.

To avoid any ambiguity, we explain here the notation used in the rest of the discussion. For the sake of simplicity, in the following, we will use the term 2-connected instead of 2-(edge)-connected. For a subset of nodes $S \subseteq V$ of an undirected network, we denote by $\delta_G(S)$ the set of edges in $E(G)$ with exactly one endpoint in S. For a subset of nodes $S \subseteq N$ of a directed network, we denote by $\delta^+_G(S)$ the set of edges in $A(G)$ outgoing S, and $\delta^-_G(S)$ the set of edges incoming S.

3.1 1-restoration property for undirected networks

In this section we deal with networks modeled as undirected graphs. We, formally, introduce a property that guarantees that the communication topology is resilient to single link failure, when we use SPT routing protocols.

Definition 1. *Given an undirected graph $G = (V, E)$ and T a spanning tree over G. We say that T has the* 1-restoration property*, with respect to G, if there exist a root vertex r and a cost function $w : E \mapsto Z_+$ such that:*

- *T is $T(\emptyset)$ (T is the unique shortest path tree over G respect to r and the weight function w);*
- *for each $e \in T$ there exists $f \in G \setminus T$ such that $T(e) = T \setminus \{e\} \cup \{f\}$ ($T \setminus \{e\} \cup \{f\}$ is the unique shortest path tree over $G \setminus \{e\}$ respect to r and the weight function w).*

Note that, given a network $G = (V, E)$ and a spanning tree routing topology T over G that has the 1-restoration property, for each link e in T we have that, for each pair of nodes u, v that do not use e in their primary path, the primary path is still operational when e fails, i.e. $P_{uv}(\emptyset) \subseteq T(e)$, if $e \notin P_{uv}(\emptyset)$. As a consequence, we define the network survivability in the event of single link failure.

Definition 2. *Given an undirected graph $G = (V, E)$, we say that G has the* 1-restoration property *if there exists a spanning tree T over G, such that T has the* 1-restoration property *respect to G.*

Now, we provide an equivalent property to the 1-restoration property that does not involve the definition of the root and the edges costs.

Definition 3. *Let $G = (V, E)$ be an undirected graph and T a spanning tree over G. We say that T has the* 1-exchange property *if, for each edge $e \in E(T)$, there exists $f \in E(G) \setminus E(T)$ such that $T(e) = (T \setminus \{e\}) \cup \{f\}$ is a spanning tree.*

The following lemma reduces the problem of checking whether a spanning tree T of a graph G has the 1-restoration property to checking whether T has the 1-exchange property:

Lemma 1. *Let $G = (V, E)$ be an undirected network. A spanning tree T has the 1-restoration property if and only if T has the* 1*-exchange property.*

Proof. Necessity is trivial.

Sufficiency. Choose r arbitrarily. Number arbitrarily all edges $f \in E(G) \setminus E(T)$ from 1 to $k = |E(G) \setminus E(T)|$ and let i_f be the index, or number, associated to edge f. Consider the following cost function on the edges:

$$w(e) = \begin{cases} 1 & \text{if } e \in E(T) \\ n \cdot i_e & \text{if } e \in E(G) \setminus E(T) \end{cases}$$

We first show that the following statements hold, with respect to the cost function w: (i) T is the unique shortest path tree rooted at r of G; (ii) for each $e \in E(T)$, there exists $f \in E(G) \setminus E(T)$ such that $(T \setminus \{e\}) \cup \{f\}$ is the unique shortest path tree rooted at r of the network $G \setminus \{e\}$.

Proof of (i). Assume that T is not the unique shortest path tree rooted at r of G. Then, there must exist a path from r to some node v that is no longer than the (r, v)-path in T and uses at least one edge f not belonging to $E(T)$. This yields to a contradiction, since such path has a cost greater than $n - 1$, while (r, v)-path in T has cost less or equal than $n - 1$.

Proof of (ii). Let e be an edge of T and let S_e and $\bar{S}_e$ be the nodes in the two components of $T \setminus \{e\}$, with $r \in S_e$. Let f be the minimum cost edge in $\delta_{G\{e\}}(S_e)$. We claim that $T(e) = (T \setminus \{e\}) \cup \{f\}$ is the unique shortest path tree rooted at r of $G \setminus \{e\}$. Suppose the contrary. Then, in $G \setminus \{e\}$ there must exist a path P from r to some node v that is no longer than the (r, v)-path in $T(v)$ and uses at least one edge h not belonging to $E(T(e))$. If $v \in S_e$, P has a cost greater than $n-1$ while the (r, v)-path in $T(e)$ has cost less than or equal to $n - 1$, a contradiction. Viceversa, if $v \in \bar{S}_e$, P must use at least one edge from $\delta_{G\setminus\{e\}}(S_e)$ and its cost is greater than $(i_f + 1)n - 1$: this is trivial if P uses some edge from $\delta_{G\setminus\{e\}}(S_e)$ different from f, else it follows from the fact that P uses f and an edge in $E(G) \setminus E(T)$. Since the (r, v)-path in $T(e)$ has cost less or equal than $i_f n + n - 1$, we have again a contradiction.

Finally, it is easy to see that (i) and (ii) imply that the pair (r, w) has the 1-restoration property, and since T is the primary tree defined by (r, w), also T has the 1-restoration property. □

Hence, Lemma 1 proves that the 1-restoration property is equivalent to the 1-exchange property.

An example of an undirected network (and a spanning tree) with the 1-restoration property is depicted in Figure 3.1.

The next lemma shows that the 1-exchange property holds indeed for *every* spanning tree T, when G is 2-connected. Also 2-connection of G is a necessary condition for the 1-exchange property to hold. We recall that checking whether a network is 2-connected can be easily performed in linear time with a Depth-First-Search of the network [83].

Lemma 2. *Let $G(V, E)$ be an undirected network. If G is 2-connected then each spanning tree T has the* 1*-exchange property. Vice versa, if there exists a spanning tree T, over G, that has the* 1*-exchange property, then G is 2-connected.*

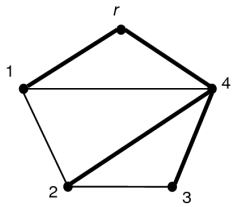

Figure 3.1: An undirected network with the 1-restoration property.

Proof. First suppose that G is 2-connected and let T be a spanning tree of G. Let e be an edge of T and $(S_e, \bar{S}_e)$ the subsets of nodes in the two components of $T \setminus \{e\}$. Since G is 2-connected, there exists $f \in \delta_G(S_e)$, $f \neq e$. Then, $T(e) = (T \setminus \{e\}) \cup \{f\}$ is a spanning tree and we are done.

Now suppose that there exists a spanning tree T, over G, that has the 1-exchange property. If G is not 2-connected, then there exists an edge $e \in E$ such that $G \setminus \{e\}$ is not connected. Therefore, e must belong to T. Trivially, T can not have the 1-exchange property, a contradiction. □

If we put together Lemma 1 and Lemma 2 we get the main result of this section:

Theorem 1. *An undirected network has the 1-restoration property if and only if is 2-connected. Moreover, each spanning tree of a 2-connected network has the 1-restoration property.*

We want to emphasize that the proof of Lemma 1 shows a simple procedure for endowing, in 2-connected networks and through the SPT protocol, a spanning tree T with the 1-restoration property. We recap this procedure in the following.

First, we arbitrarily choose a root r. Then we arbitrarily number all edges $f \in E(G) \setminus E(T)$ from 1 to $k = |E(G) \setminus E(T)|$ and let i_f be the number associated to edge f. Finally, we define the following cost function on the edges:

$$w(e) = \begin{cases} 1 & \text{if } e \in E(T) \\ n \cdot i_e & \text{if } e \in E(G) \setminus E(T) \end{cases}$$

3.2 1-restoration property for directed networks

In this section we extend the 1-restoration property to the scenario in which the communication network is modeled as a directed graph $G = (N, A)$. Our definitions directly extend to the case in which G is directed. In this case, we assume that each traffic demand is directed from a root r to a node v and therefore, instead of spanning trees, we deal with spanning *arborescences*, i.e. directed spanning trees, rooted at r (note that r is now given from the outset and cannot be chosen).

Definition 4. *Given a directed graph $G = (N, A)$ and T a spanning arborescence over G, rooted at $r \in N$. We say that T has the* 1-restoration property*, with respect to G, if there exists a cost function $w : A \to Z_+$ such that:*

- *T is $T(\emptyset)$ (T is the unique shortest path arborescence over G respect to w);*
- *for each $a \in T$ there exists $b \in G \setminus T$ such that $T(a) = T \setminus \{a\} \cup \{b\}$ ($T \setminus \{a\} \cup \{b\}$ is the unique shortest path arborescence over $G \setminus \{a\}$ respect to w).*

Finally, we provide the definition of the 1-restoration property for directed graphs.

Definition 5. *Given a directed graph $G = (N, A)$ and a root node $r \in N$, we say that G has the* 1-restoration property *if there exists a spanning arborescence T over G, rooted at r, such that T has the* 1-restoration property *respect to G.*

As in the previous section, we start with an equivalent definition to the 1-restoration property that does not involve the definition of the arc costs.

Definition 6. *Let $G = (N, A)$ be a directed network and T a spanning arborescence over G. We say that T has the* 1-exchange property *if, for each edge $a \in A(T)$, there exists $b \in A(G) \setminus A(T)$ such that $T(a) = (T \setminus \{a\}) \cup \{b\}$ is a spanning arborescence.*

Lemma 3. *Let $G = (N, A)$ be a directed network and $r \in N$. A spanning arborescence T, rooted at r, has the 1-restoration property if and only if T has the 1-exchange property.*

Proof. Necessity is trivial.

Sufficiency. Number the arcs in $A(G) \setminus A(T)$ from 1 to $|A(G) \setminus A(T)|$ so that the following property holds: for each sub-arborescence T_v of T with root $v \neq r$, the number, or index, i_a associated to every arc $a \in \delta_G^-(v)$ is smaller than the index i_f associated to every arc $f \in \delta_G^-(u)$, if u is a descendant of v in T_v. Notice that this can be easily done in polynomial time, e.g. considering first all nodes v having distance 1 from r in the tree T and numbering the arcs $a \in \delta_G^-(v)$, then all the nodes v having distance 2 from r, and so on. With this numbering, we apply the same cost function w of Lemma 1.

Let us prove that w is such that (i) T is the unique shortest path arborescence with root r for network G and (ii) for each $a \in A(T)$, there exists $f \in A(G) \setminus A(T)$ such that $T(a) = (T \setminus \{a\}) \cup \{f\}$ is the unique shortest path arborescence rooted at r for the network $G \setminus \{a\}$.

Proof of (i). It is a simple extension of the proof of Lemma 1.

Proof of (ii). Let a be an arc of T and let S_a and $\bar{S}_a$, with $r \in S_a$, be the subsets of nodes respectively in the first and second component of $T \setminus \{a\}$. Let $v \in \bar{S}_a$ be the root of the sub-arborescence T_v that is the second component of $T \setminus \{a\}$.

We know that there exists an arc f such that $T(a) = (T \setminus \{a\}) \cup \{f\}$ is an arborescence: necessarily, $f \in \delta_{G\setminus\{a\}}(v)$, since an arborescence must have one arc incoming in v. Let f be the minimum cost arc in $\delta_{G\setminus\{a\}}(v)$. Notice that f is also the minimum cost arc in $\delta^-_{G\setminus\{a\}}(\bar{S}_a)$: in fact, an arc in $\delta^-_{G\setminus\{a\}}(\bar{S}_a)$ either ends in v or it ends in a descendant of v, but in the latter case, the cost function w implies that this arc has cost greater than $w(f)$. With this observation, proving that $T(a)$ is the unique shortest path arborescence in $G \setminus \{a\}$ rooted at r is then an extension of the proof of Lemma 1. □

An example of a directed network (and a spanning arborescence) with the 1-restoration property is illustrated in Figure 3.2.

Building upon the previous lemma, we define another reformulation of the 1-restoration property for some spanning arborescence T of a directed network G.

Lemma 4. *Let $G = (N, A)$ be a directed network and T a spanning arborescence rooted at a node r of G. T has the 1-restoration property if and only if, for each $v \in N(T)$, there exists $(u, v) \in A(G) \setminus A(T)$ such that u is not a descendant of v in T.*

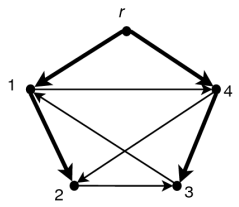

Figure 3.2: A directed network with the 1-restoration property.

Proof. We prove that the above statement is equivalent to requiring that, for each $a \in A(T)$, there exists $f \in A(G) \setminus A(T)$ such that $T(a) = (T \setminus \{a\}) \cup \{f\}$ is a spanning arborescence rooted at r. Then one can use Lemma 3 to conclude.

First we prove necessity. Let $a = (u, v) \in A(T)$ and S_a, $\bar{S}_a$ be the sets of nodes in the two components associated to the removal of a from T, with $r \in S_a$. If in $G \setminus \{a\}$ all the arcs incoming in v come from a descendant of v in T, then there exists no arc $(u, v) \in A(G) \setminus A(T)$, with $u \in S_a$. Therefore, in any spanning arborescence of $G \setminus \{a\}$, rooted in r, there are at least two arcs that do not belong to T, namely, one connecting S_a and $\bar{S}_a$, and the other entering v. So, in this case, there is no arc $f : (T \setminus \{a\}) \cup \{f\}$ is an arborescence rooted at r.

Now we prove sufficiency. Let $f = (u, v)$ be an arc such that u is not a descendant of v in T. Note that $u \in S_e$, therefore (u, v) reconnects the components S_e and $\bar{S}_e$. It follows that $T(a) = T \setminus \{a\} \cup \{f\}$ is a spanning arborescence. The result follows. □

We point out that both the previous lemmas provide an easy way to check in polynomial time if a given arborescence T of a directed network G has the 1-restoration property. Also, when T has the 1-restoration property, the proof of Lemma 3 also provides a simple procedure for assigning weights to the arcs of G as to define T through the SPT protocol.

Existence of arborescences with 1-restoration property

In this section we provide our complexity results, regarding the identification of 1-restoration property in directed graphs. In order to prove the hardness results, we provide a lemma regarding the existence of peculiar arborescences with the 1-restoration property.

Lemma 5. *Let $G = (N, A)$ be a directed graph, and $r \in N$ the root node. G has the 1-restoration property if and only if there exists an arborescence T with the 1-restoration property such that, for each node u at distance one from r, we have that $(r, u) \in A(T)$.*

Proof. Sufficiency is trivial.

Necessity. We show how to construct such an arborescence T, given any arborescence $\bar{T}$ that has the 1-restoration property. Suppose that there is a node $v : d_G(r, v) = 1 \neq d_{\bar{T}}(r, u)$ (otherwise we are done). Let $f = (r, v) \notin A(\bar{T})$, and let $a \in A(\bar{T})$ be the arc incoming in v in the tree T. Finally, let $T := (\bar{T} \setminus \{a\}) \cup \{f\}$. It is easy to see that T is an arborescence rooted at r. We claim that T has the local tree-restoration property. Then, we one can use induction to conclude.

Using Lemma 4, it is enough to check if every node $v \neq r$ has an incoming arc (u, v) for some u that is not a descendant of v in T. Notice that, for each node $u \in N$, the set of nodes $D_T(u)$ that are descendant of u in T is *contained* in the set of nodes $D_{\bar{T}}(u)$ that are descendant of u in $\bar{T}$ (i.e. $D_T(u) \subseteq D_{\bar{T}}(u)$). Therefore, since $\bar{T}$ satisfies the hypothesis of Lemma 4, it follows that this holds also for T. The result follows. □

Now, we prove that deciding if a directed graph G, with a given root r, has the 1-restoration property, is, in general, an intractable problem.

Theorem 2. *Let $G = (N, A)$ be a directed network and $r \in N$. Deciding if (G, r) has the 1-restoration property is NP-hard.*

Proof. In Lemma 4 we have shown that a spanning arborescence T has the 1-restoration property if and only if for each $v \in V \setminus \{r\}$ there is an arc $(u, v) \in E(G) \setminus E(T)$ such that u is not a descendant of v. We will prove here that, given a network $G = (V, E)$ and a node $r \in V$, deciding if there exist a spanning arborescence T rooted at r with the latter property is NP-complete, by using a reduction from k-coloring problem. Let us restate both problems as decision problems.

> **Problem LTR:** Given a directed graph $G = (V, E)$ and a root node $r \in V$, is there an arborescence T such that the following holds: for each $v \in V \setminus \{r\}$ there is an arc $(u, v) \in E(G) \setminus E(T)$ such that u is not a descendant of v in T?

> **Problem k-COL:** Given an undirected graph $B = (V, E)$ and an integer $k > 0$, does there exist a function $cl : V \mapsto \{1, 2, \dots k\}$ such that $cl(u) \neq cl(v)$ for each $\{u, v\} \in E$?

Given an instance (B, k) of k-COL problem, let $|V(B)| = n$ be the number of vertices and $|E(B)| = m$ the number of edges. We build the corresponding instance (G, r) of LTR problem as follows. The set of nodes $V(G)$ includes, besides all the nodes $v_h \in V(B)$, a root node r, a node c_j^0, for each color $j = 1, \dots, k$, and a node c_j^i for each edge e_i, $i = 1, \dots, m$, and each color j, $j = 1, \dots, k$. Formally,

$$\begin{aligned} V(G) &= \{r\} \cup \{v_1, \dots, v_n\} \cup \\ &\quad \{c_j^i : j = 1, 2, \dots k \text{ and } i = 0, 1, \dots m\}. \end{aligned}$$

The set of arcs $E(G)$ is defined as follows:

$$\begin{aligned} E(G) &= \{(r, c_j^0) : j = 1, 2, \dots, k\} \cup \\ &\quad \{(c_j^0, c_{j+1}^0) : j = 1, \dots, k-1\} \cup \{(c_k^0, c_1^0)\} \cup \\ &\quad \{(c_j^i, c_j^{i+1}) : i = 0, 1, \dots, m-1; j = 1, \dots k\} \cup \\ &\quad \{(c_j^m, v_h) : j = 1, \dots, k; h = 1, \dots, n\} \cup \\ &\quad \{(v_h, c_j^i) : v_h \text{ is an endpoint of } e_i \in E(G), \\ &\qquad h = 1, \dots, n;\ j = 1, 2, \dots, k\}. \end{aligned}$$

Suppose $k \geq 2$ (otherwise k-COL problem is trivial). We show that a solution for the instance (B, k) of k-COL implies the existence of a solution for the instance (G, r) of LTR and vice versa.

Given a solution of k-COL, we build a spanning arborescence T rooted at r with the 1-restoration property in the following way. T will include (i) arcs (r, c_j^0) for $j = 1, \dots, k$; (ii) arcs (c_j^i, c_j^{i+1}) for $i = 0, \dots, m-1$ and $j = 1, \dots, k$; (iii) an arc (c_j^m, v_h) for each node v_h, $h = 1, \dots, n$, if and only if v_h is colored with color j in the solution of k-COL on B, i.e. if and only if $c(v_h) = j$. Trivially, T is a spanning arborescence on G rooted at r.

Now we show that for each node $v \in V(G) \setminus \{r\}$ there exist an arc $f = (u, v)$ such that u is not a descendant of v in T. For a node c_j^0, such an arc is $f = (c_{j-1}^0, c_j^0)$ if $j \neq 1$, else, $f = (c_k^0, c_1^0)$. For a node v_h, such an arc can be (c_j^m, v_h) for some $j \neq c(v_h)$. Finally, for a node c_j^i, with $i \geq 1$, we can

choose f as one between the two arcs $(v_{\bar{l}}, c_j^i)$ and $(v_{\bar{h}}, c_j^i)$ incoming c_j^i, since by construction at least one among $v_{\bar{l}}$ and $v_{\bar{h}}$ is not a descendant of c_j^i in T: in fact, recall that $e_i = \{v_{\bar{h}}, v_{\bar{l}}\} \in E(B)$ and at most one can have the color j in the solution of k-COL.

Now we show that a solution for the instance (G, r) of LTR implies the existence of a solution for the instance (B, k) of k-COL. Suppose we have an arborescence $T \subseteq G$ rooted at r that is a solution of LTR. It follows from Lemma 5 that, without loss of generality, we can assume that $(r, c_j^0) \in E(T)$ for $j = 1, \ldots, k$. Let P_{rv_h} be the path from node r to a node v_h defined by T. Observe that this path must contain all the arcs $(c_j^0, c_j^1), \ldots (c_j^{m-1}, c_j^m)$ for exactly one j in $1, .., k$ (in general, j depends on v_h). We claim that assigning the color j to the node v_h, for each $h = 1, \ldots, n$, leads to a feasible solution for the instance (B, k) of the k-COL problem. Suppose the contrary. Then there must be at least a pair of nodes (say v_h and v_l) that have been assigned to a same color j and that are connected by an edge $e_i = \{v_h, v_l\} \in E(B)$. This means that both nodes v_h and v_l are descendant of c_j^i in T. Since the only arc incoming node c_j^i, different from (v_h, c_j^i) and (v_l, c_j^i), is $(c_j^{i-1}, c_j^i) \in E(T)$, it follows that there is not any arc $(u, c_j^i) \in E(G) \setminus E(T)$ such that u is not a descendant of c_j^i in T, i.e. T is not a solution for the instance (G', r), a contradiction. □

After this hardness result, we have investigated the problem searching for some special cases that can be polynomially solvable. Therefore, we have strengthened the previous complexity result, considering the "distance" from the root node. For a directed network $G = (N, A)$ the *distance* of a node $v \in V$ from r is the minimum number of arcs (*hops*) in a directed path from r to v. We denote by $k(G, r)$ the *maximum distance* of a node $v \in V$ from r.

We show in the following that, deciding whether a pair (G, r) has the 1-restoration property can be done in polynomial time if $k(G, r) = 1$, while it is NP-hard otherwise, even for $k(G, r) = 2$. Firstly, we restrict our study to instances of the LTR problem with $k(G, r) = 1$.

Theorem 3. *Let $G = (N, A)$ be a directed network and $r \in N$ such that $k(G, r) = 1$. Deciding if (G, r) has the 1-restoration property can be done in polynomial time.*

Proof. Let T be the arborescence rooted at r, such that $A(T) := \{(r, u), \text{ for all } u \in N \setminus \{r\}\}$. From Lemma 5, we know that if (G, r) has

the 1-restoration property, then T has the 1-restoration property. We check in polynomial time if T has the 1-restoration property using Lemma 4. The result follows. □

We close the analysis with the following hardness result.

Theorem 4. *Let $G = (N, A)$ be a directed network and $r \in N$ such that $k(G, r) \geq 2$. Deciding if (G, r) has the 1-restoration property is NP-hard.*

Proof. We prove the NP-hardness of our problem by reducing it from the hard combinatorial problem SAT-3 (see [21]) defined as follows.

> **Instance:** a boolean expression in conjunctive normal form with m clauses, $\mathcal{C} = \{C_1, C_2, \ldots, C_h, \ldots, C_m\}$, and n variables, $\mathcal{X} = \{x_1, x_2, \ldots, x_j, \ldots, x_n\}$, such that each variable occurs in the clauses at most three times.
>
> **Question:** is there a truth variables assignment that satisfies the boolean expression?

Let us recall that a clause is a set of *literals*, where a literal is either one of the variables x_j or the negation of one of the variables, $\bar{x}_j$.

Let I_{SAT-3} be an instance of SAT-3. Consider the following simplification procedure: for each variable x_j that occurs in the boolean expression only in the form x_j (and never in the form $\bar{x}_j$), we delete all the clauses containing it. Similarly, for each variable x_j that occurs only in the form $\bar{x}_j$ (and never in the form x_j), we delete all the clauses containing it. After this procedure, we have a new instance of SAT-3, that we denote I'_{SAT-3} such that I_{SAT-3} is a YES-instance if and only if I'_{SAT-3} is a YES-instance. In fact, a solution for I_{SAT-3} is a simple extension of a solution of I'_{SAT-3}. We extend the truth assignment that satisfies I'_{SAT-3} by fixing to $TRUE$ the deleted variables that occur only in x_j form, and fixing to $FALSE$ the deleted variables that occur only in $\bar{x}_j$ form.

Therefore, without loss of generality, we consider only I_{SAT-3} instances where each variable appears at least one time as x_j and one as $\bar{x}_j$. So we say that a variable x_j is of TYPE A if it occurs two times as x_j and one time as $\bar{x}_j$, it is of TYPE B if it occurs two times as $\bar{x}_j$ and one time as x_j, it is of TYPE C if it occurs one time as x_j and one time as $\bar{x}_j$.

Let us show how to build an instance I_{LTR} of our problem from I_{SAT-3}. Let us recall that a instance I_{LTR} is a directed graph $G = (N, A)$ with a specified root node $r \in N$. In the following we show how to construct G step by step, by

assigning nodes and arcs to the clauses and variables of I_{SAT-3}. Let us denote by $N[C_h]$ and $N[x_j]$ the sets of nodes associated to clause C_h and variable x_j, respectively, for all $h = 1, 2, \ldots, m$ and $j = 1, 2 \ldots, n$. The node set of G is

$$N = \{r\} \cup \bigcup_{h \in \{1,\ldots,m\}} N[C_h] \cup \bigcup_{j \in \{1,\ldots,n\}} N[x_j].$$

We define the sets of nodes associated to clauses as $N[C_h] = \{c_h, c'_h\}$, for $h = 1, 2, \ldots, m$. Likewise, we define the sets of nodes associated to the variables in the following way: for $j = 1, 2, \ldots, n$ if x_j is a variable that occurs three times in the boolean formula we set

$$N[x_j] = \{v_1^j, v_2^j, v_3^j, f_1^j, f_2^j, f_3^j, a_1^j, a_2^j, a_3^j, a_4^j, r_1^j, r_2^j, b_1^j, b_2^j, b_3^j, g^j\}$$

otherwise if x_j occurs two times we set

$$N[x_j] = \{v_1^j, v_2^j, f_1^j, f_2^j, a_1^j, a_2^j, r_1^j, b_1^j\}.$$

Let us denote by $A[C_h]$, for $h = 1, 2, \ldots, m$, the set of the arcs that are incident to couples of nodes in $N[C_h] \cup \{r\}$, and let us denote by $A[x_j]$, for $j = 1, 2, \ldots, n$, the set of arcs that are incident to couples of nodes in $N[x_j]$. Let us denote by $A_C[x_j]$, for $j = 1, 2, \ldots, n$, the set of arcs that go from a node in $\bigcup_{h \in [1,m]} N[C_h]$ to a node in $A[x_j]$ or viceversa. The arc set for G is

$$A = \bigcup_{h \in \{1,\ldots,m\}} A[C_h] \cup \bigcup_{j \in \{1,\ldots,n\}} A[x_j] \cup \bigcup_{j \in \{1,\ldots,n\}} A_C[x_j].$$

For $h = 1, 2, \ldots, m$, we define:

$$A[C_h] = \{(c_h, c'_h), (c'_h, c_h), (r, c'_h)\}.$$

For $j = 1, 2, \ldots, n$, we define:

$$A[x_j] = A_1[x_j] \cup A_2[x_j] \cup A_3[x_j] \cup B[x_j] \cup R[x_j].$$

From now on, we refer to $c_{\alpha(j)}$, $c_{\beta(j)}$ and $c_{\gamma(j)}$, respectively as the node associated to the clause where the first, the second and the third occurrence of a variable x_j. More precisely, if variable x_j is of TYPE A, we refer to $C_{\alpha(j)}$ and $C_{\beta(j)}$ as the clauses that contain the literal x_j, with $\alpha(j) < \beta(j)$, and to $C_{\gamma(j)}$ as the clause that contains the literal $\bar{x}_j$. If variable x_j is of TYPE B, we refer to $C_{\alpha(j)}$ and $C_{\beta(j)}$ as the clauses that contain the literal $\bar{x}_j$, with $\alpha(j) < \beta(j)$,

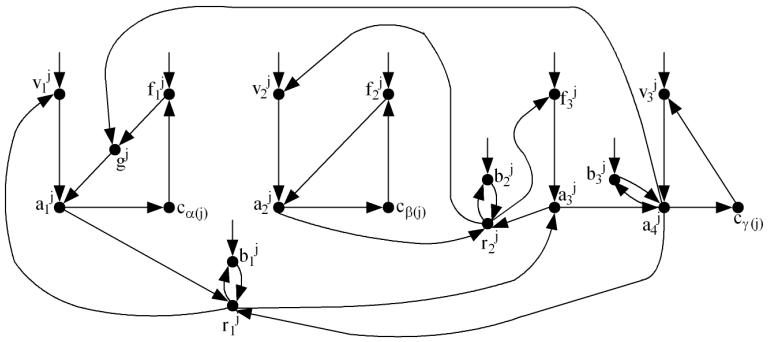

Figure 3.3: The structure built for each variable x_j of TYPE A. In the figure, all the arcs coming from above are outgoing r.

and to $C_{\gamma(j)}$ as the clause that contains the literal x_j. If variable x_j is of TYPE C, we refer to $C_{\alpha(j)}$ as the clause that contains the literal x_j and to $C_{\beta(j)}$ as the clause containing the literal $\bar{x}_j$.

If a x_j variable is of TYPE A we define (see Figure 3.3):

$$
\begin{aligned}
A_1[x_j] &= \{(v_1^j, a_1^j), (f_1^j, g^j), (g^j, a_1^j)\} \\
A_2[x_j] &= \{(v_2^j, a_2^j), (f_2^j, a_2^j)\} \\
A_3[x_j] &= \{(f_3^j, a_3^j), (a_3^j, a_4^j), (v_3^j, a_4^j), (b_3^j, a_4^j), (a_4^j, b_3^j)\} \\
\\
B[x_j] &= \{(r_1^j, v_1^j), (a_1^j, r_1^j), (r_1^j, a_3^j), (a_4^j, r_1^j), (b_1^j, r_1^j), (r_1^j, b_1^j), \\
&\quad (r_2^j, v_2^j), (a_2^j, r_2^j), (r_2^j, f_3^j), (a_3^j, r_2^j), (b_2^j, r_2^j), (r_2^j, b_2^j), (a_4^j, g^j)\} \\
\\
R[x_j] &= \{(r, v_1^j), (r, f_1^j), (r, v_2^j), (r, f_2^j), (r, f_3^j), (r, v_3^j), (r, b_1^j), (r, b_2^j), (r, b_3^j)\} \\
\\
A_C[x_j] &= \{(a_1^j, c_{\alpha(j)}), (c_{\alpha(j)}, f_1^j), (a_2^j, c_{\beta(j)}), (c_{\beta(j)}, f_2^j), \\
&\quad (a_4^j, c_{\gamma(j)}), (c_{\gamma(j)}, v_3^j)\}.
\end{aligned}
$$

If a x_j variable is of TYPE B we define the arc sets in same way of the previous

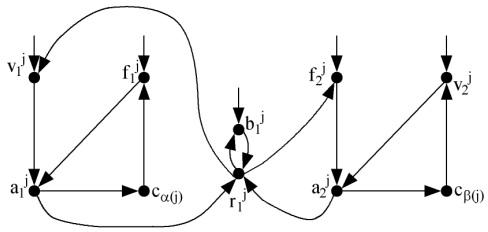

Figure 3.4: The structure built for each variable x_j of TYPE C. In the figure, all the arcs coming from above are outgoing r.

case, except that we exchange nodes v_i^j and f_i^j:

$$\begin{aligned}
A_1[x_j] &= \{(f_1^j, a_1^j), (v_1^j, g^j), (g^j, a_1^j)\} \\
A_2[x_j] &= \{(f_2^j, a_2^j), (v_2^j, a_2^j)\} \\
A_3[x_j] &= \{(v_3^j, a_3^j), (a_3^j, a_4^j), (f_3^j, a_4^j), (b_3^j, a_4^j), (a_4^j, b_3^j)\}
\end{aligned}$$

$$\begin{aligned}
B[x_j] &= \{(r_1^j, f_1^j), (a_1^j, r_1^j), (r_1^j, a_3^j), (a_4^j, r_1^j), (b_1^j, r_1^j), (r_1^j, b_1^j), \\
&\quad (r_2^j, f_2^j), (a_2^j, r_2^j), (r_2^j, v_3^j), (a_3^j, r_2^j), (b_2^j, r_2^j), (r_2^j, b_2^j), (a_4^j, g^j)\}
\end{aligned}$$

$$R[x_j] = \{(r, f_1^j), (r, v_1^j), (r, f_2^j), (r, v_2^j), (r, v_3^j), (r, f_3^j), (r, b_1^j), (r, b_2^j), (r, b_3^j)\}$$

$$\begin{aligned}
A_C[x_j] &= \{(a_1^j, c_{\alpha(j)}), (c_{\alpha(j)}, v_1^j), (a_2^j, c_{\beta(j)}), (c_{\beta(j)}, v_2^j), \\
&\quad (a_4^j, c_{\gamma(j)}), (c_{\gamma(j)}, f_3^j)\}.
\end{aligned}$$

Finally if variable x_j is of TYPE C, we define (see Figure 3.4):

$$\begin{aligned}
A_1[x_j] &= \{(v_1^j, a_1^j), (f_1^j, a_1^j)\} \\
A_2[x_j] &= \{(f_2^j, a_2^j), (v_2^j, a_2^j)\} \\
A_3[x_j] &= \emptyset
\end{aligned}$$

$$B[x_j] = \{(r_1^j, v_1^j), (a_1^j, r_1^j), (r_1^j, f_2^j), (a_2^j, r_1^j), (b_1^j, r_1^j), (r_1^j, b_1^j)\}$$

$$R[x_j] = \{(r, v_1^j), (r, f_1^j), (r, v_2^j), (r, f_2^j), (r, b_1^j)\}$$

$$A_C[x_j] = \{(a_1^j, c_{\alpha(j)}^j), (c_{\alpha(j)}, f_1^j), (a_2^j, c_{\beta(j)}), (c_{\beta(j)}, v_2^j)\}.$$

This ends the construction of I_{LTR}. We must now show that G has the 1-restoration property if and only if I_{SAT-3} is satisfiable.

Let us assume that I_{SAT-3} instance is a YES-instance, we must show that also I_{LTR} is a YES-instance. Using Lemma 4 we will provide a solution for I_{LTR}, e.g. we will build a rooted arborescence such that for each node there is a *backup* arc, (u, v), with u not descendant of v. We name *primary* arc for node v an arc incoming in v that belongs to the arborescence and *backup* arc for node v an arc incoming in v that is outgoing a node u, that is not descendant of v respect to the arborescence (Lemma 4). The idea is that a node c_h is descendant of a node in $N[x_j]$ in our I_{LTR} solution if and only if variable x_j satisfies clause C_h in solution of I_{SAT-3}.

Let us consider a truth assignment which satisfies the I_{SAT-3} instance. We associate to each clause C_h the variable that satisfies it, say x_j. If there is more than one variable satisfying it, then we pick one arbitrarily.

If x_j is of TYPE A, we identify five subcases:

(a) literal x_j satisfies the clause $C_{\alpha(j)}$,

(b) literal x_j satisfies the clause $C_{\beta(j)}$,

(c) literal $\bar{x}_j$ satisfies the clause $C_{\gamma(j)}$,

(d) literal x_j satisfies both the clauses $C_{\alpha(j)}$ and $C_{\beta(j)}$,

(e) variable x_j does not satisfy any clause.

When building the solution of I_{LTR}, for subcase (a) we choose the primary and backup arcs associated to x_j, i.e. those incoming each node in $N[x_j] \cup N[C_{\alpha(j)}]$, in the following way (see Figure 3.5):

v_1^j : arc (r, v_1) is a primary arc and (r_1^j, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (v_1^j, a_1^j) is a primary arc and (g^j, a_1^j) is a backup arc;

$c_{\alpha(j)}$: $(a_1^j, c_{\alpha(j)})$ is a primary arc and $(c'_{\alpha(j)}, c_{\alpha(j)})$ is a backup arc;

$c'_{\alpha(j)}$: $(r, c'_{\alpha(j)})$ is a primary arc and $(c_{\alpha(j)}, c'_{\alpha(j)})$ is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and (r_2^j, v_2^j) is a backup arc;

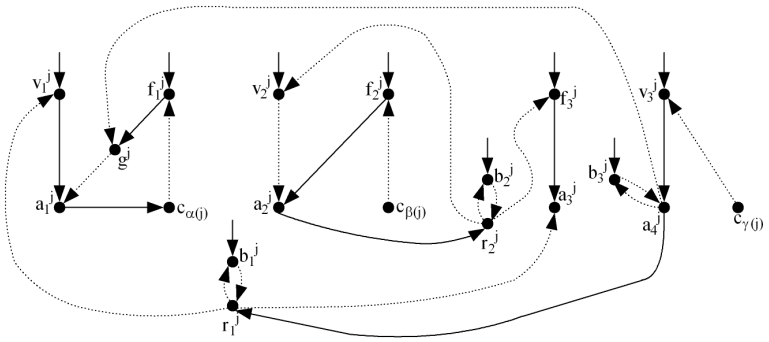

Figure 3.5: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, literal x_j satisfies the clause $C_{\alpha(j)}$. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

f_2^j : (r, f_2^j) is a primary arc and $(c_{\beta(j)}, f_2^j)$ is a backup arc;

a_2^j : (f_2^j, a_2^j) is a primary arc and (v_2^j, a_2^j) is a backup arc;

f_3^j : (r, f_3^j) is a primary arc and (r_2^j, f_3^j) is a backup arc;

v_3^j : (r, v_3^j) is a primary arc and $(c_{\gamma(j)}, v_3^j)$ is a backup arc;

a_3^j : (f_3^j, a_3^j) is a primary arc and (r_2^j, a_3^j) is a backup arc;

a_4^j : (v_3^j, a_4^j) is a primary arc and (b_3^j, a_4^j) is a backup arc;

r_1^j : (a_4^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

r_2^j : (a_2^j, r_2^j) is a primary arc and (b_2^j, r_2^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc;

b_2^j : (r, b_2^j) is a primary arc and (r_2^j, b_2^j) is a backup arc;

b_3^j : (r, b_3^j) is a primary arc and (a_4^j, b_3^j) is a backup arc;

g^j : (f_1^j, g^j) is a primary arc and (a_4^j, g^j) is a backup arc;

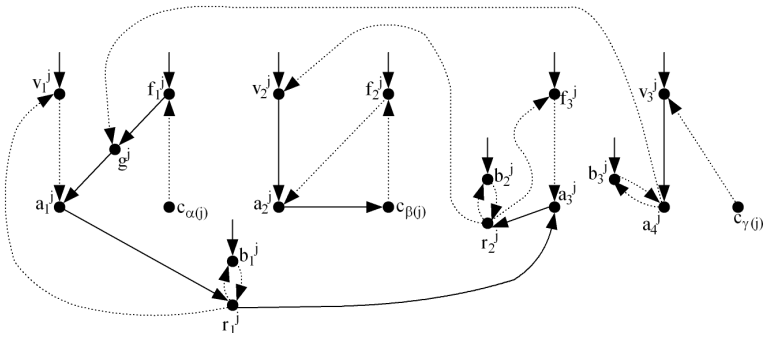

Figure 3.6: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, literal x_j satisfies the clause $C_{\beta(j)}$. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

Analogously, for subcase (b), the primary and backup arcs associated to x_j that are incoming each node in $N[x_j] \cup N[C_{\beta(j)}]$ are (see Figure 3.6):

v_1^j : (r, v_1^j) is a primary arc and (r_1^j, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (g^j, a_1^j) is a primary arc and (v_1^j, a_1^j) is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and (r_2^j, v_2^j) is a backup arc;

f_2^j : (r, f_2^j) is a primary arc and $(c_{\beta(j)}, f_2^j)$ is a backup arc;

a_2^j : (v_2^j, a_2^j) is a primary arc and (f_2^j, a_2^j) is a backup arc;

$c_{\beta(j)}$: $(a_2^j, c_{\beta(j)})$ is a primary arc and $(c'_{\beta(j)}, c_{\beta(j)})$ is a backup arc;

$c'_{\beta(j)}$: $(r, c'_{\beta(j)})$ is a primary arc and $(c_{\beta(j)}, c'_{\beta(j)})$ is a backup arc;

f_3^j : (r, f_3^j) is a primary arc and (r_2^j, f_3^j) is a backup arc;

v_3^j : (r, v_3^j) is a primary arc and (c_3^j, v_3^j) is a backup arc;

a_3^j : (r_1^j, a_3^j) is a primary arc and (f_3^j, a_3^j) is a backup arc;

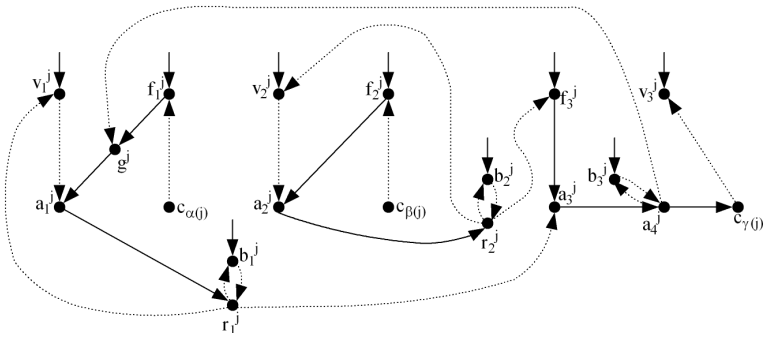

Figure 3.7: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, literal $\bar{x}_j$ satisfies the clause $C_{\gamma(j)}$. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

a_4^j : (v_3^j, a_4^j) is a primary arc and (b_3^j, a_4^j) is a backup arc;

r_1^j : (a_1^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

r_2^j : (a_3^j, r_2^j) is a primary arc and (b_2^j, r_2^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc;

b_2^j : (r, b_2^j) is a primary arc and (r_2^j, b_2^j) is a backup arc;

b_3^j : (r, b_3^j) is a primary arc and (a_4^j, b_3^j) is a backup arc;

g^j : (f_1^j, g^j) is a primary arc and (a_4^j, g^j) is a backup arc;

We also define, for subcase (c), the primary and backup arcs associated to x_j that are incoming each node in $N[x_j] \cup N[C_{\gamma(j)}]$ as follows (see Figure 3.7):

v_1^j : (r, v_1^j) is a primary arc and (r_1^j, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (g^j, a_1^j) is a primary arc and (v_1^j, a_1^j) is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and (r_2^j, v_2^j) is a backup arc;

f_2^j : (r, f_2^j) is a primary arc and $(c_{\beta(j)}, f_2^j)$ is a backup arc;

a_2^j : (f_2^j, a_2^j) is a primary arc and (v_2^j, a_2^j) is a backup arc;

f_3^j : (r, f_3^j) is a primary arc and (r_2^j, f_3^j) is a backup arc;

v_3^j : (r, v_3^j) is a primary arc and (c_3^j, v_3^j) is a backup arc;

a_3^j : (f_3^j, a_3^j) is a primary arc and (r_1^j, a_3^j) is a backup arc;

a_4^j : (a_3^j, a_4^j) is a primary arc and (b_3^j, a_4^j) is a backup arc;

$c_{\gamma(j)}$: $(a_4^j, c_{\gamma(j)})$ is a primary arc and $(c'_{\gamma(j)}, c_{\gamma(j)})$ is a backup arc;

$c'_{\gamma(j)}$: $(r, c'_{\gamma(j)})$ is a primary arc and $(c_{\gamma(j)}, c'_{\gamma(j)})$ is a backup arc;

r_1^j : (a_1^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

r_2^j : (a_2^j, r_2^j) is a primary arc and (b_2^j, r_2^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc;

b_2^j : (r, b_2^j) is a primary arc and (r_2^j, b_2^j) is a backup arc;

b_3^j : (r, b_3^j) is a primary arc and (a_4^j, b_3^j) is a backup arc;

g^j : (f_1^j, g^j) is a primary arc and (a_4^j, g^j) is a backup arc;

We also define, for subcase (d), the primary and backup arcs associated to x_j that are incoming each node in $N[x_j] \cup N[C_{\alpha(j)}] \cup N[C_{\beta(j)}]$ as follows (see Figure 3.8):

v_1^j : (r, v_1^j) is a primary arc and (r_1^j, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (v_1^j, a_1^j) is a primary arc and (g^j, a_1^j) is a backup arc;

$c_{\alpha(j)}$: $(a_1^j, c_{\alpha(j)})$ is a primary arc and $(c'_{\alpha(j)}, c_{\alpha(j)})$ is a backup arc;

$c'_{\alpha(j)}$: $(r, c'_{\alpha(j)})$ is a primary arc and $(c_{\alpha(j)}, c'_{\alpha(j)})$ is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and (r_2^j, v_2^j) is a backup arc;

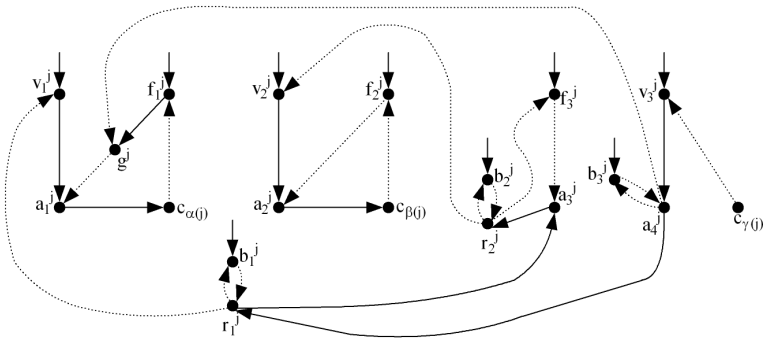

Figure 3.8: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, literal x_j satisfies both the clauses $C_{\alpha(j)}$ and $C_{\beta(j)}$. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

f_2^j : (r, f_2^j) is a primary arc and $(c_{\beta(j)}, f_2^j)$ is a backup arc;

a_2^j : (v_2^j, a_2^j) is a primary arc and (f_2^j, a_2^j) is a backup arc;

$c_{\beta(j)}$: $(a_2^j, c_{\beta(j)})$ is a primary arc and $(c'_{\beta(j)}, c_{\beta(j)})$ is a backup arc;

$c'_{\beta(j)}$: $(r, c'_{\beta(j)})$ is a primary arc and $(c_{\beta(j)}, c'_{\beta(j)})$ is a backup arc;

f_3^j : (r, f_3^j) is a primary arc and (r_2^j, f_3^j) is a backup arc;

v_3^j : (r, v_3^j) is a primary arc and $(c_{\gamma(j)}, v_3^j)$ is a backup arc;

a_3^j : (r_1^j, a_3^j) is a primary arc and (f_3^j, a_3^j) is a backup arc;

a_4^j : (r_1^j, a_4^j) is a primary arc and (b_3^j, a_4^j) is a backup arc;

r_1^j : (a_4^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

r_2^j : (a_3^j, r_2^j) is a primary arc and (b_2^j, r_2^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc;

b_2^j : (r, b_2^j) is a primary arc and (r_2^j, b_2^j) is a backup arc;

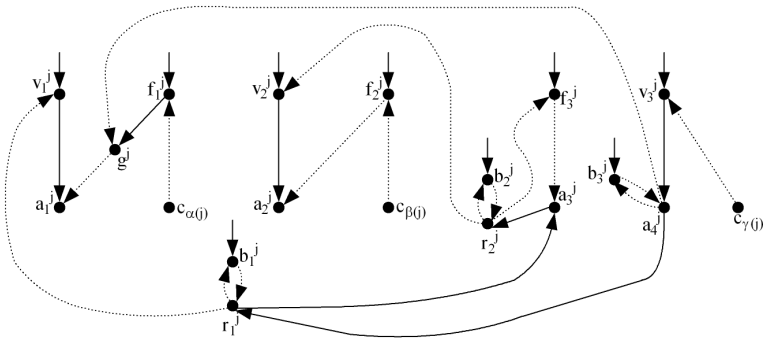

Figure 3.9: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, variable x_j does not satisfy any clause. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

b_3^j : (r, b_3^j) is a primary arc and (a_4^j, b_3^j) is a backup arc;

g^j : (f_1^j, g^j) is a primary arc and (a_4^j, g^j) is a backup arc;

We also define, for subcase (e), the primary and backup arcs associated to x_j that are incoming each node in $N[x_j]$ as follows (see Figure 3.9):

v_1^j : (r, v_1^j) is a primary arc and (r_1^j, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (v_1^j, a_1^j) is a primary arc and (g^j, a_1^j) is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and (r_2^j, v_2^j) is a backup arc;

f_2^j : (r, f_2^j) is a primary arc and $(c_{\beta(j)}, f_2^j)$ is a backup arc;

a_2^j : (v_2^j, a_2^j) is a primary arc and (f_2^j, a_2^j) is a backup arc;

f_3^j : (r, f_3^j) is a primary arc and (r_2^j, f_3^j) is a backup arc;

v_3^j : (r, v_3^j) is a primary arc and $(c_{\gamma(j)}, v_3^j)$ is a backup arc;

a_3^j : (r_1^j, a_3^j) is a primary arc and (f_3^j, a_3^j) is a backup arc;

a_4^j : (v_3^j, a_4^j) is a primary arc and (b_3^j, a_4^j) is a backup arc;

r_1^j : (a_4^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

r_2^j : (a_3^j, r_2^j) is a primary arc and (b_2^j, r_2^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc;

b_2^j : (r, b_2^j) is a primary arc and (r_2^j, b_2^j) is a backup arc;

b_3^j : (r, b_3^j) is a primary arc and (a_4^j, b_3^j) is a backup arc;

g^j : (f_1^j, g^j) is a primary arc and (a_4^j, g^j) is a backup arc;

Let us consider case in which x_j is of TYPE B. As we have pointed out during the graph construction, this case is symmetrical to case in which x_j is of TYPE A. So we build our solution by fixing primary and backup arcs in the same way as the previous case. The only difference is that we must exchange the roles of the nodes v_i^j and f_i^j.

Let us consider case in which x_j is of TYPE C. In this scenario we identify three subcases:

(a) literal x_j satisfies the clause $c_{\alpha(j)}$,

(b) literal $\bar{x}_j$ satisfies the clause $c_{\beta(j)}$,

(c) variable x_j does not satisfy any clause.

For subcase (a), when building the solution of I_{LTR}, we choose the primary and backup arcs associated to x_j, i.e. those incoming each node in $N[x_j] \cup N[c_{\alpha(j)}]$, in the following way (see Figure 3.10):

v_1^j : (r, v_1^j) is a primary arc and (r_1^j, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (v_1^j, a_1^j) is a primary arc and (f_1^j, a_1^j) is a backup arc;

$c_{\alpha(j)}$: $(a_1^j, c_{\alpha(j)})$ is a primary arc and $(c'_{\alpha(j)}, c_{\alpha(j)})$ is a backup arc;

$c'_{\alpha(j)}$: $(r, c'_{\alpha(j)})$ is a primary arc and $(c_{\alpha(j)}, c'_{\alpha(j)})$ is a backup arc;

f_2^j : (r, f_2^j) is a primary arc and (r_1^j, f_2^j) is a backup arc;

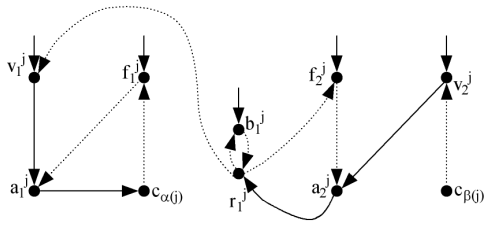

Figure 3.10: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, literal x_j satisfies the clause $c_{\alpha(j)}$. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

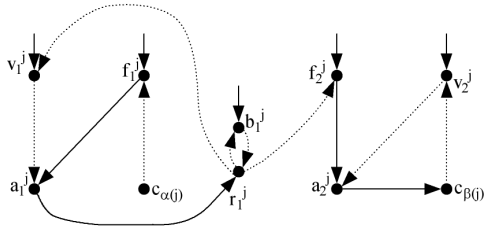

Figure 3.11: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, literal $\bar{x}_j$ satisfies the clause $c_{\beta(j)}$. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

v_2^j : (r, v_2^j) is a primary arc and $(c_{\beta(j)}, v_2^j)$ is a backup arc;

a_2^j : (v_2^j, a_2^j) is a primary arc and (f_2^j, a_2^j) is a backup arc;

r_1^j : (a_2^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc.

For subcase (b), for each node in $N[x_j] \cup N[c_{\beta(j)}]$, we choose the primary and backup arcs as follows (see Figure 3.11):

v_1^j : (r, v_1^j) is a primary arc and (r_1^j, v_1^j) is a backup arc;

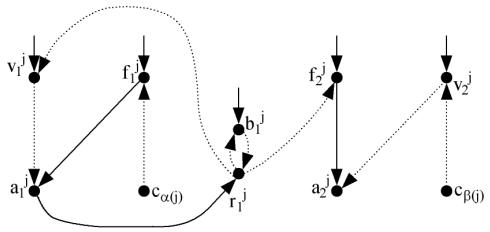

Figure 3.12: The fragment of the solution of I_{LTR}, for the structure of x_j, if x_j is of TYPE A and, in I_{SAT-3}, variable x_j does not satisfy any clause. In the figure, all the arcs coming from above are outgoing r. The solid arcs are primary arcs, and the dotted arcs are backup arcs.

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (f_1^j, a_1^j) is a primary arc and (v_1^j, a_1^j) is a backup arc;

f_2^j : (r, f_2^j) is a primary arc and (r_1^j, f_2^j) is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and $(c_{\beta(j)}, v_2^j)$ is a backup arc;

a_2^j : (f_2^j, a_2^j) is a primary arc and (v_2^j, a_2^j) is a backup arc;

$c_{\beta(j)}$: $(a_2^j, c_{\beta(j)})$ is a primary arc and $(c'_{\beta(j)}, c_{\beta(j)})$ is a backup arc;

$c'_{\beta(j)}$: $(r, c'_{\beta(j)})$ is a primary arc and $(c_{\beta(j)}, c'_{\beta(j)})$ is a backup arc;

r_1^j : (a_1^j, r_1^j) is a primary arc and (b_1^j, r_1^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1^j, b_1^j) is a backup arc.

For subcase (c), for each node listed below (nodes in $N[x_j]$), we select the primary and backup arcs as follows (see Figure 3.12):

v_1 : (r, v_1^j) is a primary arc and (r_1, v_1^j) is a backup arc;

f_1^j : (r, f_1^j) is a primary arc and $(c_{\alpha(j)}, f_1^j)$ is a backup arc;

a_1^j : (f_1^j, a_1^j) is a primary arc and (v_1, a_1^j) is a backup arc;

f_2^j : (r, f_2^j) is a primary arc and (r_1, f_2^j) is a backup arc;

v_2^j : (r, v_2^j) is a primary arc and $(c_{\beta(j)}, v_2^j)$ is a backup arc;

a_2^j : (f_2^j, a_2^j) is a primary arc and (v_2, a_2^j) is a backup arc;

r_1^j : (a_1^j, r_1^j) is a primary arc and (b_1, r_1^j) is a backup arc;

b_1^j : (r, b_1^j) is a primary arc and (r_1, b_1^j) is a backup arc.

The above illustrated procedure completely defines the solution of I_{LTR}. In fact, all the primary arcs constitute a spanning arborescence rooted at r, and all the backup arcs satisfy Lemma 4.

Let us assume that I_{LTR} is a YES-instance, we must show that also I_{SAT-3} is a YES-instance. To prove this we build a truth variables assignment, $t : \mathcal{X} \rightarrow \{TRUE, FALSE\}$, which satisfy all the clauses in I_{SAT-3}.

Without loss of generality, by Lemma 5, we may consider a solution of I_{LTR} in which all arcs outgoing the root node r are primary arcs. We build a corresponding truth assignment as follows. For each clause C_h, with $h = 1, 2, \ldots, m$, if node c_h is a descendant of a node v_i^j, for some $i \in \{1, 2, 3\}$ and $j \in \{1, 2, \ldots, n\}$, we define $t(x_j) = TRUE$. Else, if node c_h is a descendant of a node f_i^j, we define $t(x_j) = FALSE$. For each variable, x_j, that has no clause node descendant from its structure (i.e. from a node in $N[x_j]$), we fix $t(x_j)$ to an arbitrary value. To prove that t is a solution for I_{SAT-3}, we have to show that the following two properties hold:

Property 1. Every clause is satisfied by t.

Property 2. The definition of t is consistent, i.e. for each variable x_j, either $t(x_j) = TRUE$ or $t(x_j) = FALSE$ (not both).

Proof of Property 1. In order to prove this property we first consider the structure of a solution of a I_{LTR} YES-instance and then perform a case analysis to show that for every clause there is at least one literal value set to $TRUE$, by proving that each clause node c_h is a descendant of either a node v_i^j for some literal x_j that occurs in C_h or a node f_i^j for some literal $\bar{x}_j$ that occurs in C_h.

First we show that in the solution of I_{LTR}, the nodes $b_i^j \in N$ and $c'_h \in N$ (with $i \in \{1, 2, 3\}$, $j \in \{1, 2, \ldots, n\}$ and $h \in \{1, 2, \ldots, m\}$) are leafs in the primary arborescence. We prove this statement for nodes b_1^j, for $j = 1, 2, \ldots, n$. The same reasoning can be easily applied to nodes b_2^j, b_3^j (for x_j of TYPE A and B) and c'_h with $j = 1, 2, \ldots, n$ and $h = 1, 2, \ldots, m$. Each node b_1^j, with

$j = 1, 2, \ldots, n$, has two incoming arcs, (r, b_1^j) and (r_1^j, b_1^j), and one outgoing arc, (b_1^j, r_1^j). In particular, (r, b_1^j) is a primary arc for $j = 1, 2, \ldots, n$, thus (r_1^j, b_1^j) is a backup arc for b_1^j. Therefore, the only arc outgoing b_1^j, (b_1^j, r_1^j) cannot be an arc of the primary arborescence (due to Lemma 4) and hence, b_1^j cannot have any child. The same argument applies to nodes b_2^j, b_3^j and c'_h by considering, in place of r_1^j, nodes r_2^j, a_4^j and c_h, respectively.

Now, for $h = \{1, 2, \ldots, m\}$, let C_h be a clause in I_{SAT-3} and let c_h be the corresponding node in $N[C_h]$. In the solution of I_{SAT-3}, since c'_h is a leaf, c_h must be a child of an a_i^j node, for some $i \in \{1, 2, 4\}$ and some $j \in \{1, 2, \ldots, n\}$. Let us now consider three different cases depending on whether variable x_j is of TYPE A, B or C.

Case A: variable x_j is of TYPE A. In this case we identify three subcases: the first in which c_h is a child of a_1^j, the second in which c_h is a child of a_2^j and the third in which c_h is a child of a_4^j.

Subcase A.1. When c_h is a child of a node a_1^j we have that, by construction, the literal x_j occurs in C_h and $h = \alpha(j)$. Moreover, a_1^j must be a child of v_1^j or g^j. If a_1^j is a child of v_1^j, this means that c_h is a descendant of v_1^j, so C_h is satisfied since we have defined $t(x_j) = TRUE$. Otherwise, if a_1^j is a child of g^j, observe that g^j can not be a child of f_1^j (in this case c_h would be a descendant of f_1^j and, since (c_h, f_1^j) is the backup arc for f_1^j, we would have a contradiction due to Lemma 4). So, g^j must be a child of a_4^j. Now, a_4^j can be a child of v_3^j or a_3^j. If it is a child of v_3^j, we are done (c_h is a descendant of a v_i^j node). Differently, assume a_4^j a child of a_3^j. This latter node can be a child of f_3^j or r_1^j. In the first situation we have that arc (r_1^j, a_3^j) must be the backup arc for a_3^j. Since r_1^j must be a child of a_1^j or a_4^j, in both cases, by Lemma 4, we get a contradiction. So, a_3^j can not be a child of f_3^j. Therefore, assume a_3^j is a child of r_1^j. Since r_1^j must be a child of a_1^j or a_4^j, in such solution there would be a cycle (contradiction). Thus a_3^j can not be a child of r_1^j. This concludes the proof when c_h is a child of a a_1^j node.

Subcase A.2. When c_h is a child of a node a_2^j we have that, by construction, the literal x_j occurs in C_h and $h = \beta(j)$. Then, necessarily a_2^j must be a child of f_2^j or v_2^j. If a_2^j is a child of f_2^j, then c_h would be a descendant of f_2^j. But, since (c_h, f_2^j) is the backup arc for node f_2^j, we contradict Lemma 4. Thus a_2^j can not be a child of f_2^j. Differently, if a_2^j is a child of v_2^j, then c_h is a descendant of v_2^j and therefore, $t(x_j) = TRUE$. This truth assignment satisfies C_h.

Subcase A.3. When c_h is a child of a node a_4^j we have that, by construction, the literal $\bar{x}_j$ occurs in C_h and $h = \gamma(j)$. Then a_4^j must be a child of v_3^j or a_3^j. If it is a child of v_3^j, then c_h would be a descendant of v_3^j. But this contradicts Lemma 4 because (c_h, v_3^j) is the backup arc for v_3^j. Thus a_4^j must be a child of a_3^j node. Moreover, a_3^j must be a child of f_3^j or r_1^j. If a_3^j is a child of f_3^j then $t(x_j) = FALSE$, and we are done. If a_3^j is a child of r_1^j, then r_1^j can not be a child of a_4, because in such solution there would be a cycle (contradiction). So, r_1^j must be a child of a_1^j. Furthermore, a_1^j can be a child of v_1^j or g^j. If a_1^j is a child of v_1^j, r_1^j would be a descendant of v_1^j, that is a contradiction of Lemma 4 (because (r_1^j, v_1^j) is the backup arc for v_1^j). Thus a_1^j must be child of g^j. Now, g^j can be a child of f_1^j or a_4^j. If g^j is a child of f_1^j, then the arc (a_4^j, g^j) would be the backup arc for g^j, and this contradicts the Lemma 4 as well. If g^j is a child of a_4^j, then in such solution there would be a cycle (contradiction). Therefore, if c_h is a child of a_4^j (subcase A.3), then c_h must be a descendant of f_3^j. In conclusion, in Case A, if c_h is a child of a_1^j, it must be a descendant of v_1^j or v_3^j, else, if c_h is a child of a_2^j, it must be a descendant of v_2^j, otherwise, if c_h is a child of a_4^j, it must be a descendant of f_3^j. Hence, if in C_h there is the literal x_j, it will be set $t(x_j) = TRUE$, else, if in C_h there is the literal $\bar{x}_j$, it will be set $t(x_j) = FALSE$, so that C_h is satisfied in Case A.

Case B: variable x_j is of TYPE B. In this case we proceed in the same way of the previous, except that the nodes v_i^j and f_i^j are exchanged. In this case we conclude that if c_h is a child of a_1^j, it must be a descendant of f_1^j or f_3^j. If c_h is a child of a_2^j, it must be a descendant of f_2^j. If c_h is a child of a_4^j, it must be a descendant of v_3^j. Thus, if in C_h there is the literal x_j, it will be set $t(x_j) = TRUE$, else, if in C_h there is the literal $\bar{x}_j$, it will be set $t(x_j) = FALSE$, so that C_h is satisfied in Case B.

Case C: variable x_j is of TYPE C. In this case we identify two subcases: the first in which c_h is a child of a_1^j and the second in which c_h is a child of a_2^j.

Subcase C.1. When c_h is a child of a node a_1^j we have that, by construction, the literal x_j occurs in C_h and $h = \alpha(j)$. In this case a_1^j can be a child of v_1^j or f_1^j. If a_1^j is a child of f_1^j, c_h would be a descendant of f_1^j. Since (c_h, f_1^j) is the backup arc for f_1^j we get a contradiction with Lemma 4. Thus a_1^j must be a child of v_1^j. Therefore, c_h is a descendant of v_1^j.

Subcase C.2. When c_h is a child of a node a_2^j we have that, by construction, the literal $\bar{x}_j$ occurs in C_h and $h = \beta(j)$. In this case a_2^j can be a child of v_2^j or f_2^j. If a_2^j is a child of v_2^j, c_h would be a descendant of v_2^j. Since (c_h, v_2^j) is

the backup arc for v_2^j we get a contradiction with Lemma 4. Thus a_2^j must be a child of f_2^j. Therefore, c_h is a descendant of v_2^j.

In conclusion, in Case C, if c_h is a child of a_1^j, it must be a descendant of v_1^j, else, if c_h is a child of a_2^j, it must be a descendant of f_2^j. Therefore, if in C_h there is the literal x_j, it will be set $t(x_j) = TRUE$, else, if in C_h there is the literal $\bar{x}_j$, it will be set $t(x_j) = FALSE$, so that C_h is satisfied in Case C.

Proof of Property 2. The Property 2 states that, for each variable x_j for $j = 1, 2, \ldots, n$, our definition of t can not assign both $t(x_j) = TRUE$ and $t(x_j) = FALSE$. We structure the proof in the following way: for each variable x_j, for $j = 1, 2, \ldots, n$, we perform a case analysis on the variable types, to show that two clause nodes c_k and c_l, with $k, l \in \{1, 2, \ldots, m\}$, can not descend from v_i^j and $f_{i'}^j$, respectively, for some $i, i' \in \{1, 2, 3\}$.

In order to prove Property 2, we recall that, if there is a clause node c_h, with $h = 1, 2, \ldots, m$, that is a descendant of v_i^j, for some $i \in \{1, 2, 3\}$ and $j \in \{1, 2, \ldots, n\}$, we define $t(x_j) = TRUE$, else, if c_h is a descendant of f_i^j, for some $i \in \{1, 2, 3\}$ and $j \in \{1, 2, \ldots, n\}$, we define $t(x_j) = FALSE$. Now, consider a variable x_j with $j \in \{1, 2, \ldots, n\}$.

Case A: variable x_j is of TYPE A. In this case, as shown in the proof of Property 1, a node c_h, with $h \in \{1, 2, \ldots, m\}$, can be a descendant of nodes v_i^j or f_i^j belonging to $N[x_j]$ in the solution of I_{LTR} only in the following ways (see Figure 3.13):

- if $h = \alpha(j)$, then, the directed path from r to c_h in the primary arborescence is either $p_1(h) = (r, v_1^j, a_1^j, c_h)$ or $p_1'(h) = (r, v_3^j, a_4^j, g^j, a_1^j, c_h)$
- if $h = \beta(j)$, then, the directed path from r to c_h in the primary arborescence is $p_2(h) = (r, v_2^j, a_2^j, c_h)$
- if $h = \gamma(j)$, then, the directed path from r to c_h in the primary arborescence is $p_3(h) = (r, f_3^j, a_3^j, a_4^j, c_h)$

Let us consider two clauses C_k and C_l, with $k, l \in \{1, 2, \ldots, m\}$, such that the literal x_j occurs in C_k and the literal $\bar{x}_j$ occurs in C_l. To prevent multiple truth assignments, we have to prove that, in the solution of I_{LTR}, at the same time paths $p_1(k)$ and $p_3(l)$ or $p_1'(k)$ and $p_3(l)$ or $p_2(k)$ and $p_3(l)$ can not coexist. Now, consider a I_{LTR} solution in which there are both $p_1(k)$ and $p_3(l)$, and consider the node r_1^j. The arcs (r_1^j, v_1^j) and (r_1^j, a_3^j) are backup arcs, respectively, for v_1^j and a_3^j. So, r_1^j must be a child of a_1^j or a_4^j. If r_1^j is a child of a_1^j, this contradicts the Lemma 4. If r_1^j is a child of a_4^j, this contradicts the Lemma 4

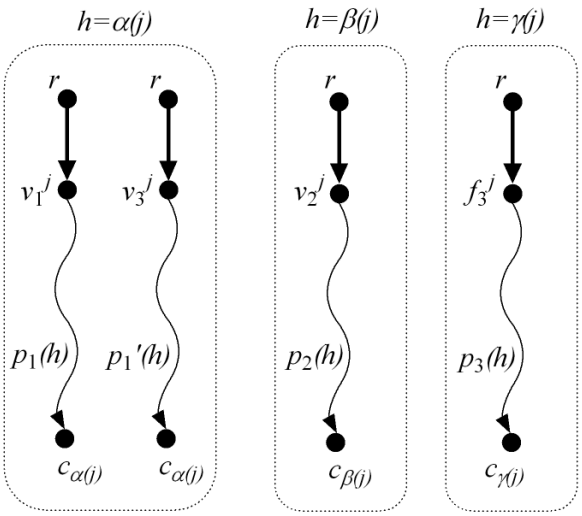

Figure 3.13: All the possible paths from r to c_h trough $N[x_j]$ nodes, where x_j is of TYPE A.

again. So we conclude that, in solution, node r_1^j can not be a child of any node (contradiction). Hence, in a I_{LTR} solution $p_1(k)$ and $p_3(l)$ can not coexist. Let us consider a solution in which there are both $p_1'(k)$ and $p_3(l)$. This case is impossible, because there will be a node, a_4^j, that would be a child of two nodes and the solution could not be a rooted arborescence (contradiction). Hence, in a I_{LTR} solution $p_1'(k)$ and $p_3(l)$ can not coexist. Finally, consider a solution in which there are $p_2(k)$ and $p_3(l)$ both, and consider node r_2^j. The arcs (r_2^j, v_2^j) and (r_2^j, f_3^j) are backup arcs, respectively, for v_2^j and f_3^j. Now, r_2^j must be a child of a_2^j or a_3^j. If r_1^j is a child of a_2^j, this contradicts the Lemma 4. If r_1^j is a child of a_3^j, this contradicts the Lemma 4 again. So we conclude that, in solution, node r_2^j can not be a child of any node (contradiction). Hence, in a I_{LTR} solution $p_2(k)$ and $p_3(l)$ can not coexist.

Case B: variable x_j is of TYPE B. The proof that, for variables of this type, there are no multiple truth assignments, is the same of Case A, except that the nodes v_i^j and f_i^j are exchanged.

Case C: variable x_j is of TYPE C. In the proof of the Property 1 we

have shown that, in this situation, a node c_h, with $h \in \{1, 2, \ldots, m\}$ can be a descendant of nodes v_i^j or f_i^j belonging to $N[x_j]$ only in the following ways (see Figure 3.14):

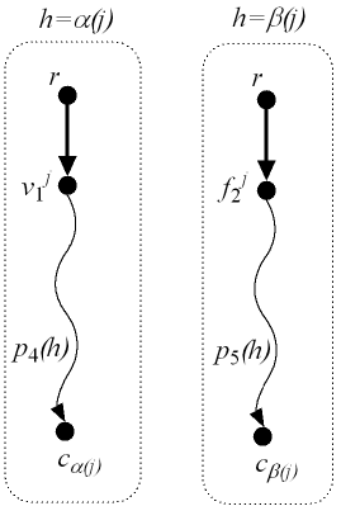

Figure 3.14: All the possible paths from r to c_h trough $N[x_j]$ nodes, where x_j is of TYPE C.

- if $h = \alpha(j)$, then, the directed path from r to c_h in the primary arborescence is $p_4(h) = (r, v_1^j, a_1^j, c_h)$
- if $h = \beta(j)$, then, the directed path from r to c_h in the primary arborescence is $p_5(h) = (r, f_2^j, a_2^j, c_h)$

Let us consider two clauses C_k and C_l, with $k, l \in \{1, 2, \ldots, m\}$, such that the literal x_j occurs in C_k and the literal $\bar{x}_j$ occurs in C_l. To prevent multiple assignments in this situation we have to prove that, in solution of I_{LTR}, at same time paths $p_4(k)$ and $p_5(l)$ can not coexist. To prove this consider node r_1^j. The arcs (r_1^j, v_1^j) and (r_1^j, f_2^j) are backup arcs, respectively, for v_1^j and f_2^j. Now, r_1^j must be a child of a_1^j or a_2^j. If r_1^j is a child of a_1^j, this contradicts the Lemma 4. If r_1^j is a child of a_2^j, this contradicts the Lemma 4 again. Thus we conclude

that, in solution, node r_1^j can not be a child of any node (contradiction). Hence, in a I_{LTR} solution $p_4(k)$ and $p_5(l)$ can not coexist. This completes the proof that there are no multiple truth assignments in our definition of t, so Property 2 also holds.

Thus, both Property 1 and Property 2 guarantee that if I_{LTR} is a YES-instance, also I_{SAT-3} is a YES-instance, so that the theorem hods.

□

3.3 Directed Two-level graphs

In this section we focus our attention to a special case in which the problem of determining whether a graph has the 1-exchange property becomes solvable in polynomial time. Namely, we consider the case in which the instance graphs are such that $k(G, r) = 2$, i.e. all nodes are at distance at most 2 from r. We have proved that in general, for $k(G, r) = 2$, the problem is NP-hard, however, as we will see hereafter, it becomes polynomially solvable when there are no arcs between nodes at the same distance from r.

Definition 7. *A directed graph $G = (N, A)$ is a* two-level-sparse *graph if all nodes in G are at distance at most 2 from r and there are no arcs between nodes at the same distance from r.*

Let us denote by x_i, with $i = 1, 2, \ldots m$, and y_j, with $j = 1, 2, \ldots, n$, respectively, the nodes in N at distance 1 and 2 from r. Let $X = \{x_1, x_2, \ldots x_m\}$ and $Y = \{y_1, y_2, \ldots y_n\}$. Thus, we refer to a two-level-sparse graph with the notation $G = (X, Y, r, A)$. Furthermore, in such a graph we require that each node in X has at least one incoming arc from a node in Y and each node in Y has at least two incoming arcs from X (note that this is a necessary condition for admitting the 1-exchange property). In order to prove that it is simple to identify if a two-level-sparse graph has the 1-exchange property (1-exchange problem), we introduce an auxiliary problem, the *bipartite backup problem*, and prove, firstly, that the bipartite backup problem is a polynomial time solvable problem, and, secondly, that the bipartite backup problem and the 1-exchange problem are polynomially equivalent.

First, we need the following definition:

Definition 8. *A* 2-cycle *is a directed cycle of length 2.*

Hence, we state the bipartite backup problem:

Definition 9. Bipartite backup problem (BB)*: Given a directed bipartite graph* $G = (X, Y, F)$ *such that* $d^-(x_i) \geq 1$ *and* $d^-(y_j) \geq 2$ *for each* $x_i \in X$ *and* $y_j \in Y$*, find* $Q \subseteq F$ *such that* $\forall v \in (X \cup Y), d_Q^-(v) = 1$ *and* Q *contains no 2-cycles.*

In order to prove that the BB problem is solvable in polynomial time, we provide a procedure (see Algorithm 1) that returns the required set Q. The idea of the algorithm is to first mark all nodes belonging to a cycle (which is not a 2-cycle) and then mark all nodes reachable from a marked node through a directed path. If all nodes are marked, then a set of arcs Q is built.

Algorithm 1 Solve Bipartite Backup Problem

Input: $G = (X, Y, F)$
Output: If it exists, $Q \subseteq F$
1: initialize $Q = \emptyset$;
2: initialize all the nodes to unmarked;
3: **while** there exists a directed cycle of *length* > 2 with all unmarked nodes **do**
4: mark all the nodes of the cycle;
5: put all the arcs of the cycle in Q;
6: **for** each arc from a marked node to an unmarked node **do**
7: mark the unmarked endpoint of the arc;
8: put the arc in Q;
9: **if** all the nodes are marked **then**
10: **return** Q

Lemma 6. *Algorithm 1 solves the bipartite backup problem.*

Proof. First, we show that if the algorithm produces an output, then the bipartite backup problem admits a solution. Let Q be the output of Algorithm 1. By construction, every node has been marked once (and only once), therefore in Q there is only one incoming arc for each node. We must show that in Q there are no 2-cycles. Assume there is a 2-cycle involving nodes u and v, then in the subgraph G_Q of G induced by Q, $\{u, v\}$ is a connected component. In this case, the only way Algorithm 1 could add both (u, v) and (v, u) in Q is by considering the 2-cycle as a cycle in Step 3 of the algorithm. Clearly, this is a contradiction.

We now have to show that if the bipartite backup problem has a solution Q', then Algorithm 1 outputs a feasible solution. To prove this we only need to

show that Algorithm 1 will mark every node. In fact, in this case the algorithm outputs a set of arcs Q and we have already seen that Q represents a feasible solution. By hypothesis, there exists exactly one arc (u, v) in Q' for every node v. Let us consider an arbitrary node u_1 and let u_2 be its only predecessor in Q'. Analogously, let u_3 be the predecessor of u_2 in Q'. By iterating this process we may obtain a sequence of nodes $u_1, u_2, u_3, \ldots u_h$, such that $(u_{i+1}, u_i) \in Q'$ for all $i = 1, 2, \ldots h-1$ and $u_h = u_i$ for some $i = 1, 2, \ldots h-2$ (since the number of nodes in the graph is finite). Then, $C = (u_h, u_{h-1}, \ldots u_i)$ is a simple directed cycle (which is not a 2-cycle). Hence, C is either (i) found by Algorithm 1 in Step 3 or (ii) it has an arc in common with another directed cycle C' found by the algorithm. In case (i) the algorithm marks all the nodes in C and then during Step 6 it marks all the nodes in the directed path connecting cycle C to u_1, including u_1. In case (ii) the algorithm first marks the nodes in C' (and

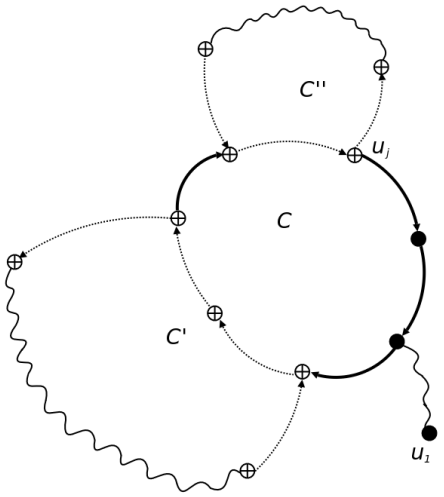

Figure 3.15: The cycle C not found by Algorithm 1. The cycles C' and C'' are found.

possibly in other cycles) from G. This implies that, as it can be seen with an example in Figure 3.15, there always exists a directed path from a marked node of C, e.g. u_j, to u_1. Therefore, the algorithm during Step 6 marks all the nodes in the directed path from u_j to u_1, including u_1. Hence, in both cases

u_1 is marked by the algorithm and, by the arbitrariness of u_1, all the nodes will be marked. So Algorithm 1 outputs a feasible solution for the bipartite backup problem.

□

Lemma 7. *A directed two-level-sparse graph $G = (X, Y, r, A)$ admits the 1-exchange property if and only if the instance $G' = (X, Y, A)$ of the bipartite backup problem admits a solution.*

Proof. Firstly, we prove that if the instance $G' = (X, Y, A)$ of the BB problem admits a solution, then graph $G = (X, Y, r, A)$ has the 1-exchange property. If the instance $G' = (X, Y, A)$ admits a solution then there exists $Q \subseteq A$ such that $\forall v \in (X \cup Y)$, $d^-_Q(v) = 1$ and Q contains no 2-cycles. So, consider the following rooted arborescence T over G:

$$T = \{(r, x_i) \text{ for all } 1 \leq i \leq m\} \cup \{(x_i, y_j) \in Q \text{ for all } 1 \leq j \leq n\}.$$

T is an arborescence because each node in X and Y has only one incoming arc and there are no directed or undirected cycles. T is rooted in r because each node in X and Y has exactly one incoming path, outgoing r. Let us prove that T has the 1-exchange property. By Lemma 4, we only need to prove that for each node v in X and Y there must exist an incoming arc coming from a node u which is not a descendant of v in T. Hence, consider a node $y_h \in Y$. Since y_h does not have descendant nodes in T and has at least two incoming arcs in G, there exists an arc, entering y_h, that does not belong to T. Consider a node $x_k \in X$. Let $a = (y_j, x_k)$ be the only arc, incoming x_k that belongs to Q. By hypothesis, there are no arcs among the nodes of X in G and, by construction, there are no arcs from nodes in Y to nodes in X in T, so that the only descendants of node x_k, in T, are its children. Now, if we assume that y_j is a descendant (child) of x_k, then arc $a' = (x_k, y_j)$ of T, by construction, belongs to Q. This implies that a belongs to Q as well as a'. This means that in Q there is a 2-cycle (contradiction). Therefore, y_j can not be a descendant of x_k, so that the arc a is an arc incoming x_k, outgoing from a node not descendant of x_k.

Now, we prove that if graph $G = (X, Y, r, A)$ has the 1-exchange property then the instance $G' = (X, Y, A)$ of the BB problem admits a solution. If G admits a spanning arborescence that has the 1-exchange property, the by Lemma 3 and Lemma 5, G must admit a spanning arborescence, T', with the 1-exchange property such that for each node u at distance one from r, we have that $(r, u) \in A(T')$. By Lemma 4, for each node u there exists an arc incoming

u, outgoing a node not descendant of u in T'. Consider, for each x_i, with $1 \leq i \leq m$, an arc $a_i = (y_j, x_i)$ such that y_j is not a descendant of x_i. We define the following set as a solution for the BB problem:

$$Q = \{(x_i, y_j) \in T' \text{ for all } 1 \leq j \leq n\} \cup \{a_i \text{ for all } 1 \leq i \leq m\}.$$

Obviously, for each node in X and in Y there exists exactly one incoming arc in Q. Further, by construction, if in Q there is an arc (u, v), then (v, u) can not belong to Q, so that Q does not contain 2-cycles.

□

Let us analyze the complexity of the Algorithm 1. The *while* loop, in Step 3, searches for a cycle with length greater than 2 over the unmarked nodes. This can be done by performing a depth-first search on the graph induced by the unmarked nodes, possibly starting from every node of such graph. This procedure is obviously polynomial. For every iteration of Step 3, in Step 4 the algorithm marks at least 3 nodes, so that the total number of iterations of the *while* loop can not be greater than $n/3$. The statements in Step 4 and Step 5 have a computational complexity of at most $O(n)$, depending on the data structures used to represent the graph and the set Q. The *for* loop at Step 6 has a number of iterations strictly less than the number of the arcs in the graph. It is simple to observe that also the statements in Step 7 and Step 8 have a computational complexity of at most $O(n)$. Clearly, the whole algorithm has a polynomial complexity. So, we may conclude that the problem to identifying if a two-level-sparse graph has the 1-exchange property or the 1-restoration property is a problem solvable in polynomial time.

3.4 Conclusions

In this chapter we have addressed the problem of verifying whether a directed or undirected network has the 1-restoration property. We have proved that for undirected networks this can be done in polynomial time, while for directed networks the problem is NP-hard, even on networks with a very special structure. For directed graphs, we have provided a particular case, the two-level-sparse graphs, in which the problem becomes polynomially solvable.

A natural extension for further research is to check whether our results for the survivable networks apply to the case where more links fail at the same time. We investigate this topic in the next section.

Chapter 4

Multiple links failure restoration

In survivability network design literature, it is satisfactory to develop restoration procedures that guarantee the functionality of the communication network, after a single link failure. This is mainly due to the fact that it seems negligible the possibility that more than one link breaks at the same time. Further, it is considered that, between a link failure and the next one, there is enough time to permit to a network operator to repair the first break, so that, at the successive break, the network is completely working. In this analysis, we also consider, as the worst case, that multiple links failure can happen.

In this chapter, we extend the properties proved for the case of single link failure. Firstly, we consider the case of undirected graph. We are given a network $G = (V, E)$ where at most $q \in \mathcal{Z}_+$ edges may fail. We assume that there are traffic demands between each pair of vertices. In order to route the traffic demands, we select a spanning tree T within the STP protocol.

Since we research the maximum efficiency of the network, in case of multiple links failure, we require that if at most q edges fail, there must exist a spanning tree, over the residual network, that share, with the initial communication topology, each path that is not affected by the failure. This amounts to require that the initial routing topology has the following property:

Definition 10. *Given an undirected network $G = (V, E)$, and a spanning tree T over G, we say that T has the* q-exchange property*, if, for each $F \subseteq E$ with $|F| \leq q$, there exists $H(F) \subseteq E \setminus F$ such that $(T \setminus F) \cup H(F)$ is a spanning tree over $G \setminus F$.*

We observe that the q-exchange property is the natural extension of the 1-

exchange property. Note also that G has a spanning tree with the q-exchange property only if it is $(q+1)$-connected.

4.1 The undirected graphs scenario

The following step, after understanding the utility of q-exchange property in survivability network area, is the research of which networks guarantee a routing topology T, that has the property.

In the undirected graphs scenario, the following lemma gives us a criterion for identifying, in polynomial time, such kind of networks.

Lemma 8. *Let $G = (V, E)$ be an undirected graph. G has a spanning tree T with the q-exchange property if and only if G is $(q+1)$-connected. Moreover, in this case, each spanning tree of G has the q-exchange property.*

Proof. The only-if part is trivial. Suppose the contrary. Then there exists a set $F \subseteq E$, with at most q edges, such that $G \setminus F$ is not connected: trivially, there is no spanning tree for $G \setminus F$.

For the if part, let T be any spanning tree of G and F a subset of at most q edges of E. Let $S_1, \ldots, S_p$ the connected components of $T \setminus F$; note that $p \leq q+1$. Since G is $(q+1)$-connected, it follows that $G \setminus F$ is connected: in particular, for every $i = 1..p-1$, $\delta_{G\setminus F}(\bigcup_{j=1,\ldots,i} S_j)$ is non-empty. We can also assume without loss of generality that, for every $i = 1, \ldots, p-1$, there exists an egde $\{u_i, v_i\} \in G \setminus F$, such that $u_i \in (\bigcup_{j=1,\ldots,i} S_j)$ and $v_i \in S_{i+1}$. Therefore, if we add to $T \setminus F$ the edges $\{u_i, v_i\}$, $i = 1, \ldots, p-1$, we obtain a spanning tree of $G \setminus F$. Since this holds for every set F and spanning tree T, the result follows. □

We recall that the interest in this survivability analysis is sparkled by the study of the STP protocol family. Thus, we extend the 1-restoration property (i.e. the routing topology built by edge weights assignment is *resilient*, see previous chapter) to the case of q simultaneous failures.

Definition 11. *Given an undirected graph $G = (V, E)$ and T a spanning tree over G. We say that T has the* q-restoration property*, respect to G, if there exists a cost function $w : E \mapsto \mathcal{Z}_+$ such that:*

- *T is $T(\emptyset)$ (T is the unique shortest path tree over G respect to r and the weight function w);*

- *for each $F \subseteq E$, with $|F| \leq q$ and $F \neq \emptyset$, there exists $H(F) \subseteq E \setminus F$ such that $T(F) = (T \setminus F) \cup H(F)$; ($T(F) = (T \setminus F) \cup H(F)$ is the unique shortest path tree over $G \setminus F$ respect to r and the weight function w).*

In this case, we say that w defines T.

Now, let T be a spanning tree of a $(q+1)$-connected network G. The next question we want to investigate is whether is possible to define a suitable cost vector $w : E \to Z_+$ such that T has the q-restoration property.

Theorem 5. *Let $G = (V, E)$ be a $(q+1)$-connected graph and T be a spanning tree. The following cost function defines T:*

$$w(e_i) = \begin{cases} 2^{i-1} \cdot n & \text{if } 1 \leq i \leq q \\ n + \sum_{j=i-q}^{i-1} \quad w(e_j) & \text{if } q+1 \leq i \leq m-n+1 \\ 1 & \text{if } i > m-n+1, \text{ i.e. if } e_i \in E(T) \end{cases}$$

where $\{e_1, e_2, \ldots, e_m\}$ is any ordering of $E(G)$ with $e_{m-n+2}, \ldots, e_m \in E(T)$.

Proof. Suppose that T is a spanning tree of G. Choose r arbitrarily. Observe that, by construction, the following holds, for each $i \leq m-n+1$, i.e. for each $e_i \notin E(T)$:

$$w(e_i) = n + \sum_{j=\max(1,i-q)}^{i-1} w(e_j). \tag{4.1}$$

We show that the following statements hold, with respect to the cost function w: (i) T is the unique shortest path tree rooted at r of G, i.e. $T = T(\emptyset)$; (ii) for each $F \subseteq E(G)$, with $|F| \leq q$, there is a shortest path tree in $G \setminus F$ and it contains all the edges in $E(T) \setminus F$.

Proof of (i). Assume that T is not the unique shortest path tree rooted at r of G. Then, there must exists a path from r to some vertex v that is no longer than the (r, v)-path in T and uses at least one edge f not belonging to $E(T)$. This yields to a contradiction, since such path has a cost greater than $n-1$, while the (r, v)-path in T has cost less or equal than $n-1$.

Proof of (ii). Let F be a subset of at most q edges of G. We show that every shortest path tree $T(F)$ of $G \setminus F$ must contain all the edges in $E(T) \setminus F$.

Suppose the contrary, and let $\{u, v\}$ be an edge such that: $\{u, v\} \in E(T)$ but $\{u, v\} \notin E(T(F))$. In the following, we denote by P_{xy} the unique path

from a vertex x to a vertex y on $T(F)$ (we let $P_{xx} = \emptyset$), and by $w(P_{xy})$ its cost. Notice that $u \notin P_{rv}$, since otherwise we might replace the sub-path P_{uv}, with cost $w(P_{uv}) > 1$, with the edge $\{u, v\}$, and get a shorter path of $G \setminus F$ from r to v. Similarly, $v \notin P_{ru}$.

Let $s \in V$ be such that $P_{ru} = P_{rs} \cup P_{su}$, $P_{rv} = P_{rs} \cup P_{sv}$ and $E(P_{su}) \cap E(P_{sv}) = \emptyset$. The following holds:

$$w(P_{sv}) \leq w(P_{su}) + w(uv) \text{ and } w(P_{su}) \leq w(P_{sv}) + w(uv). \tag{4.2}$$

Let $\overline{e} := argmax\{w(e), \text{ with } e \in E(P_{su}) \cup E(P_{sv})\}$, and without loss of generality let $\overline{e} \in P_{sv}$. Note that $\overline{e} \notin E(T)$, since otherwise the edges in $E(P_{su}) \cup E(P_{sv}) \cup \{u, v\}$ would define a cycle of unit cost edges, while by construction the set of unit cost edges defines a tree.

Now observe that P_{su} contains at most $n - 2$ edges in $E(T)$. Then, let $Z := \{E(P_{su}) \cap (E(G) \setminus E(T))\}$. We also claim that $|Z| \leq q$. Otherwise, since $T \setminus F$ is a forest with at most $q+1$ connected components, it follows that there are at least two vertices x and y such that $E(P_{xy}) \cap Z \neq \emptyset$, with x and y in the same component of $T \setminus F$: in this case there would be a shorter path of $G \setminus F$ from r to either x or y.

Now recall that $w(\overline{e}) > w(e)$ for each $e \in Z$. Since $|Z| \leq q$ and, by construction, $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_{m-n+1})$, it follows from (4.1) that $w(\overline{e}) \geq n + \sum_{e \in F} w(e)$. Then $w(P_{su}) + w(uv) \leq \sum_{e \in Z} w(e) + n - 1 < w(\overline{e}) \leq w(P_{sv})$, a contradiction to (4.2).

□

4.2 The directed graphs scenario

When we move to the case in which G is directed, we consider that there are traffic demands only from some common source $r \in V$ to every other node. Thus, a routing topology must be a spanning arborescence rooted at r (in the following, for sake of shortness, we often simply refer to spanning arborescences, assuming that they are rooted at r). In this case, we need to define a slightly weaker exchange property. Let $T \subset G$ be a spanning arborescence rooted at r and $F \subset E(G)$. We denote by $cl(F, T)$ the union of the set of failed arcs and the set of the arcs that do not belong to the arborescence rooted at r in the graph $T \setminus F$, i.e.:

$$cl(F, T) = F \cup \bigcup_{(u,v) \in E(T) \cap F} E(T_v)$$

where T_v is the sub-arborescence of T rooted at v.

Then, we want that if at most q edges fail, there must exist a spanning arborescence, over the residual network, that share, with the initial communication topology, each path that is not affected by the failure. This amounts to requiring that the initial routing topology has the following property:

Definition 12. *A spanning arborescence T rooted at r of a directed graph G has the* weak q-exchange property*, if, for each $F \subseteq E$ with $\mid F \mid \leq q$, there exists $H(F) \subseteq E \setminus F$ such that $(T \setminus cl(F,T)) \cup H(F)$ is a spanning arborescence rooted at r.*

In fact, when some set of arcs F fail, we may route the traffic on the spanning arborescence $T(F) := (T \setminus cl(F,T)) \cup H(F)$, since, by definition, T and $T(F)$ share every directed (r,v)-path on T that is not affected by the failure of F. Note also that G has a spanning arborescence rooted at r with the weak q-exchange property only if, for each node $v \neq r$, there are at least $(q+1)$ arc-disjoint directed (r,v)-paths: G is $(q+1)$-arc-connected (in the sequel we refer to a $(q+1)$-arc-connected graph as a $(q+1)$-connected graph for shortness).

The first question we investigate is in which case, given a directed graph G and a root node, $r \in V$, there is a spanning arborescence T rooted at r with the weak q-exchange property.

Lemma 9. *Let $G = (N, A)$ be a directed graph and $r \in V$ be the root node. Let T be a spanning arborescence, rooted at r. We say that T has the weak q-exchange property if and only if G is $(q+1)$-connected. Moreover, in this case, each spanning arborescence of G rooted at r has the weak q-exchange property.*

Proof. Let us prove that if G is $(q+1)$-connected then T has the weak q-exchange property. Suppose the contrary. Then there exists a set $F \subseteq E$, with at most q edges, and a node $v \neq r$ such that $G \setminus F$ does not contain any (r,v)-path: trivially, there is no spanning arborescence rooted at r for $G \setminus F$.

Let us prove that if T has the weak q-exchange property then G is $(q+1)$-connected. Let T be any spanning arborescence rooted at r and F a subset of at most q edges of E. Let S_1 be the connected component in $T \setminus cl(F,T)$ that contains r, and $S_2, \ldots, S_p$ the connected components correspondent to singletons (i.e. isolated nodes) of $T \setminus cl(F,T)$; note that p here could be $\geq q+1$. Since G is $(q+1)$-connected, it follows that $G \setminus F$ is connected: in particular, for every $i = 1, \ldots, p-1$, $\delta^+_{G \setminus F}(\bigcup_{j=1,\ldots,i} S_j)$ is non-empty. We can also assume without loss of generality that that, for every $i = 1, \ldots, p-1$, there exists an arc $(u_i, v_i) \in G \setminus F$, such that $u_i \in (\bigcup_{j=1,\ldots,i} S_j)$ and $v_i \in S_{i+1}$.

Therefore, if we add to $T \setminus cl(F,T)$ the arcs (u_i, v_i), $i = 1, \ldots, p-1$, we obtain a spanning arborescence of $G \setminus F$. Since this holds for every set F and spanning arborescence T, the result follows. □

For directed graphs, q-restoration property definition is analogous to the case of undirected graphs. Here, we investigate for a criterion to identify whether a network routing topology (spanning arborescence) has the q-restoration property or not.

Lemma 10. *Let $G = (V, E)$ be a $(q+1)$-connected graph and T be a spanning arborescence rooted at r. The following weight function defines T:*

$$w(e) = \begin{cases} 1 & \text{if } e \in E(T) \\ n & \text{otherwise.} \end{cases}$$

Proof. Suppose that T is a spanning arborescence of G rooted at a node r. We show that the following statements hold, with respect to the weight function w: (i) T is the unique shortest path arborescence rooted at r of G ; (ii) for each $F \subseteq E(G)$, with $|F| \leq q$, every shortest path arborescence in $G \setminus F$ contains all the directed (r, v)-paths that are not affected by the failure of F.

Proof of (i). Assume that T is not the unique shortest path arborescence rooted at r of G. Then, there must exists a path from r to some node v that is no longer than the (r, v)-path in T and uses at least one arc f not belonging to $E(T)$. This yields to a contradiction, since such path has a weight greater than $n-1$, while the (r, v)-path in T has weight less or equal than $n-1$.

Proof of (ii). Let F be a subset of at most q arcs of G, and suppose there exists a shortest path arborescence in $G \setminus F$ that does not contain all the directed (r, v)-paths in $T \setminus F$. Then, there must exists in $G \setminus F$ a path from r to some node v that is no longer than the (r, v)-path in T, a contradiction to (i).

□

The q-exchange property for directed graphs

To improve the survivability faculties of a communication network, we require that the communication topology, after multiple links failure, preserves unchanged the traffic flows that are routed over paths that are not affected by breaks. This survivability requirement is assured if the communication topology has the weak q-exchange property. But, also in directed graphs scenario, we can require the q-exchange property for the network. We point out that,

while if a spanning arborescence T has the weak q-exchange property, to restore a set of fault arcs, we may replace a set of working arcs, if T has the q-exchange property, in the event of k link failures, with $k \leq q$, we can restore the routing topology simply by taking other k new arcs from the communication network.

We provide a lemma to identify directed graphs that admit the q-exchange property.

Lemma 11. *Let $G = (V, E)$ be a directed graph and $r \in V$ a node such that, for each $w \in V \setminus \{r\}$, there is $(r, w) \in E$. G admits a spanning arborescence, rooted at r, with the q-exchange property, if and only if the in-degree of each node $v \neq r$ is at least $q + 1$.*

Proof. Necessity is trivial. Sufficiency: consider the spanning arborescence T with arc set $E(T) := \{(r, v),$ for each $v \in V \setminus \{r\}\}$. Since there are at least $q + 1$ arcs incoming in each node, after the failure of q arcs, each node still has the backup arc. □

The problem to identify if a directed graph G has the q-exchange property is a generalization of the problem to identify if G has the 1-exchange property. Since we have shown, in the previous chapter, that checking 1-exchange property on directed graphs is a NP-hard problem, the same hardness result also holds for the q-exchange property.

Chapter 5

Introduction to broadcast problems

A fundamental issue, in telecommunication and computer networks, is the information transmission. Since, from the birth of the first telegraph network, there was the primary need to communicate important news from a source to all the connected devices, the researchers start to analyze the information dissemination problems. Broadcast is a process used in communication networks to deliver an information from a source to all the peers. There are broadcasting processes in which the source of the information, in the network, is a single peer or, otherwise, we have a set of originators, knowing all together the information at the beginning of the procedure. In a broadcast process we require that the time of the procedure is minimized and that, at the end of the process, all the peers are informed.

Different problems on broadcasting in communication networks have been well studied in the past. Depending on the communication model, an informed peer can communicate with only one peer at time (telephone model) or with all its neighbors simultaneously (wireless model).

In a wireless scenario, since the communication is provided by an antenna, through the air medium, a device communicates always with all its neighbors. Moreover, in a wireless network, we consider that each peer is connected to all the other devices that are in its transmission range. But, since the wireless devices, during the network activity, are free to move, we consider that the wireless network topology changes in a fast way. In this setting there is a large possibility that more communications can happen at the same time, so that

some of them complete in an erroneous way. In this case, we say that the transmissions make a *collision*. This type of interferences are very unusual in wired network. In a wired system, each communication link is shared only between two system components, that are often synchronized in their communications. The analysis of broadcast problems in a wired network is usually modeled with the telephone model. Since we are interested in study this latter model, also known as telegraph or whispering model, we explain broadcast issues in this setting, distinctly, in the following section.

5.1 Broadcasting in telephone model

In broadcast research area, the most primitive and known communication model is the *telephone model*. In this communication setting we have that the broadcast network is represented by an undirected graph $G = (V, E)$. The set of information originators is denoted by V_0 and, in the case in which the process starts from a single vertex, we have that $V_0 = \{r\}$. There is only one kind of message sent in the process. In a broadcast procedure we denote the time in which a vertex v receives the message with $t(v)$. If we consider, at the same time, more broadcast procedures, we will use $t(v)$, $t'(v)$ and $t''(v)$. In the telephone model we are restricted to the following constraints:

1. only the informed vertices can communicate;
2. the time duration of a message communication along any edge in the graph is one round;
3. each vertex can partecipate to only one communication per round.

Now, we formally define a broadcast process P:

Definition 13. *A broadcast process P is a sequence:*

$$V_0, E_1, V_1, E_2, V_2, \ldots, E_h, V_h$$

where:

- $\forall i = 1, \ldots, h, V_i \subseteq V$ *(V_i is the set of vertices who are informed at time i with transmissions along the edges in E_i),*
- $\forall i = 1, \ldots, h, E_i \subseteq E$,

- $\forall i = 1, \ldots, h, E_i$ *composed by only edges with exactly one endpoint in* V_{i-1},
- $V_i = V_{i-1} \cup \{v : (u, v) \in E_i, u \in V_{i-1}\}$,
- $V_h = V$,

In this case we say that P *has a time duration of* h *rounds.*

Therefore, we state the problem of Minimum Broadcast Time: given an undirected network $G = (V, E)$, a set of information originators V_0, find the minimum k such that there exists the broadcast process P over G, starting from V_0, with length k.

In literature, among the broadcast problems, the Minimum Broadcast Time is the more basic and studied question. It has received a lot of interest, and is treated in the past in a great number of works, [15, 24, 27, 26, 28, 29, 41, 44, 46, 59, 74, 85, 89].

The first analysis on the minimum broadcast problem were done on tree networks. In 1981, Slater, Cockayne and Hedetniemi [79] consider broadcasting on trees and provide an optimal polynomial algorithm for the minimum broadcast time problem. They also showed that the minimum broadcast time problem with a unique originator is a NP-hard problem. In the same year, also Farley and Proskurowski (see [29]) have considered the problem in trees. Given a tree network T and an integer q, they provide an algorithm to identify the smallest vertex set V_0', such that there are $|V_0'|$ subtrees with the following properties: (i) the subtrees are a covering for the T; (ii) each vertex of V_0' belongs to a different subtree; (iii) for each subtree there exists a broadcasting process of length lesser or equal to q.

Also other important network structures were analyzed by researcher to find fast broadcast procedure. While Farley and Hedetniemi, in [28], have analyzed broadcasting in grid-graphs, Feige, Peleg, Raghavan, and Upfal in [30] deal with hypercubes and random graphs.

In the area of network broadcasting, a substantial portion of the work regards constructing good broadcast graphs. A broadcast graph $G = (V, E)$ is a graph that allows a broadcast process that terminates in the quickest way, that is $\lceil \log |V| \rceil$. A minimum broadcast graph on n vertices is a broadcast graph with the minimum number of edges over all broadcast graphs on n vertices. There are many papers on designing broadcast graphs, we limit to cite only some of them. While, in [34], the authors propose methods for reducing the maximum degree of the vertices in graphs that maintain optimal broadcast

time, the problem to design fault-tolerant broadcast graphs were considered in [15, 58].

For the minimum broadcast problem there is a large number of papers regarding approximation theory. An approximation algorithm for the minimum broadcast time problem was developed, in 1992, by Kortsarz and Peleg [54]. They provide an $O(\sqrt{n})$ additive approximation algorithm in an n-node graph. They give better approximation algorithms for chordal graphs, outerplanar graphs, seriesparallel graphs, and trees of cliques. The best approximation algorithm, for minimum broadcast time, was provided by Ravi [72], that guarantees an approximation ratio of $O(\log^2 n/\log\log n)$ for general graph. Ravi provided, also, an approximation algorithm for the case in which the degree of the network is bounded by a constant. We also mention the heuristic algorithm, without guarantee, of Scheuermann and Wu, proposed in [77].

In the field of network broadcasting, an important variant of the broadcast problem is the gossip problem, wherein every node has its own message that must be disseminated to every other node. If we have a network with n peers, we may consider that a gossip process is a composition of n different broadcast processes, each one originated by a different node. We recall [46] as a comprehensive survey of the literature related to the broadcast and gossip problem.

5.2 Minimum Service Time

Our interest on broadcast problems is focused on a variant of the Minimum Broadcast Problem in the telephone model. We call this broadcast problem version, the *Minimum Service Time* problem. Formally, we define the service time in a broadcast process P that has the "information time" function $t : V \mapsto Z_+$ (the function that indicate the time in which a vertex receive the information with respect to P).

Definition 14. *Given an undirected graph $G = (V, E)$. Given a broadcast process P over G, defined by its information time function t. The* Service Time *of P is:*

$$ST(P) = \frac{\sum_{v \in V} t(v)}{n}.$$

We investigate, in the following chapters, the problem of minimizing the service time over an undirected graph G.

Chapter 6

Broadcast in centralized scenario

In this chapter, we investigate the problem of minimizing the service time in a communication network. We have, in this setting, that the network is represented by an undirected graph $G = (V, E)$ and, the communication rules follow the formerly cited telephone model. In order to find the minimum service time, we deal with a centralized scenario, i.e. there is a complete knowledge of the network topology for each algorithm that provide a broadcast solution. In the sequel we refer to minimum service time problem as ST-r. In the follows, we will provide an hardness result for the problem ST-r: we prove that the ST-r problem is a NP-hard problem, by reducing the well known NP-hard problem 3-Dimensional Matching (see [39]) to it. After, we show that problem ST-r, given a tree network, is polynomial time solvable.

6.1 Minimizing the Service Time is NP-hard

In this section we discuss the complexity of problem ST-r. We show that the problem of minimizing the service time is NP-hard.

In our way to proceed, we introduce the sum of the information times of a broadcast P:

$$SUM_t(P) = \sum_{v_V} t(v).$$

Note that, if we have a communication network, the problem of minimizing the service time is the same of minimizing the sum of the information times, over all the peers, except a normalization factor related to the instance size (the number of the vertices in the graph). Indeed, given a network G, the

broadcast scheme P that minimize the $SUM_t(P)$ is the same that minimize $ST(P)$. In our analysis, we deal with the sum of the information times as objective function, since we prefer to handle an integer function instead of a rational valued one.

In order to prove the hardness result, we introduce the generalized version of the minimum service time problem, in which the source of the information is more than one vertex. We formally state the decision version of the problem of minimizing the service time in the general case, ST, i.e. the case in which a set of vertices, V_0, know the information at the beginning of the broadcast process.

> **Instance:** Let $G = (V, E)$ be an undirected graph and let $V_0 \subseteq V$ be the set of information source vertices. Let k be a positive integer.
>
> **Question:** Does there exist a sequence:
>
> $$V_0, E_1, V_1, E_2, V_2, \ldots, E_h, V_h$$
>
> where:
>
> - $\forall i = 1, \ldots, h, V_i \subseteq V$ (V_i is the set of vertices who are informed at time i with transmissions along the edges in E_i),
> - $\forall i = 1, \ldots, h, E_i \subseteq E$,
> - $\forall i = 1, \ldots, h, E_i$ composed by only edges with exactly one endpoint in V_{i-1},
> - $V_i = V_{i-1} \cup \{v : (u, v) \in E_i, u \in V_{i-1}\}$,
> - $V_h = V$,
> - $k \geq \sum_{v \in V} t(v)$, where $t(v) = \min\{i : v \in V_i\}$ ($t(v)$ represents the time in which v is informed)?

Note that in every graph all the edges that partecipate to an arbitrary broadcast procedure, create a spanning tree over the graph. This observation is underlined by the example in Figure 6.1, where the labels represent the instants in which the nodes receive the message. In this case, a broadcast process defines a broadcast spanning tree.

In order to prove that problem ST-r is NP-hard, we provide a lemma regarding the sum of the information times of the vertices that compose a path.

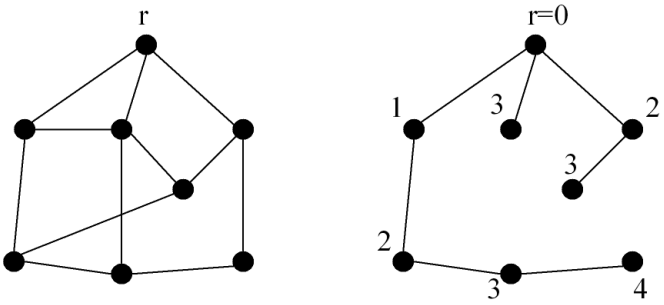

Figure 6.1: A broadcast process that defines a spanning tree on the network.

Lemma 12. *Let $G = (V, E)$ be a graph. Let $P = \{v_1, v_2, \ldots, v_p\}$ such that $P \subseteq V$ and $d(v_i) = 2$ for all $i \in \{1, 2, \ldots, p-1\}$ and $d(v_p) = 1$. Let B a broadcast process that starts from vertices in $V_0 \subseteq V$. Let $V_0 \cap P = \emptyset$. We have:*

$$\sum_{i \in \{1,2,\ldots,p\}} t(v_i) = p(t(v_1) - 1) + \frac{p(p+1)}{2}.$$

Proof. Vertices $v_1, v_2, \ldots, v_p$ create a path in G that is connected to the other vertices of the graph by an unique edge, adjacent to v_1. Since in our model is not allowed that an informed vertex, that has an adjacent not informed vertex, does not transmit, the information time of a vertex v_i is $t(v_i) = t_(v_{i-1} + 1)$. Hence, we have that:

$$\sum_{i \in \{1,2,\ldots,p\}} t(v_i) = \sum_{i \in \{1,2,\ldots,p\}} (t(v_1)+j-1) = \sum_{i \in \{1,2,\ldots,p\}} (t(v_1)-1) + \sum_{i \in \{1,2,\ldots,p\}} j =$$

$$= p(t(v_1) - 1) + \frac{p(p+1)}{2}.$$

□

To prove that problem ST-r is NP-hard, firstly, we reduce 3-Dimensional Matching to ST, with a technique similar to the one presented in [79], and secondly, we reduce problem ST to problem ST-r.

Theorem 6. *Problem ST is NP-hard.*

Proof. We will reduce the well known NP-complete problem 3DM [39] to the decisional version of ST. We state, formally, problem 3DM:

> **Instance:** let $X = \{x_1, x_2, \ldots, x_m\}$, $Y = \{y_1, y_2, \ldots, y_m\}$, $Z = \{z_1, z_2, \ldots, z_m\}$ and let $C \subseteq X \times Y \times Z$
>
> **Question:** Does there exist a subset of C of size m such that each pair of elements of the subset disagree in all three coordinates?

We will prove that given any instance I_{3DM} of 3DM we can build, in polynomial time, an instance I_{ST} of ST such that I_{3DM} is a YES-Instance if and only if I_{ST} is a YES-Instance. Now we explain how to build I_{ST-r} from I_{3DM}. Let $m = |X| = |Y| = |Z|$ and $M = |C|$. Let us take, arbitrarily, five positive integers, namely p_1, p_2, p_3, p_4, p_5, such that:

$$\begin{aligned} p_1 &> M(p_2 + 2p_3 + 4) + m(3p_4 + 4p_5 + 6) \\ p_2 &> M(2p_3 + 4) + m(3p_4 + 4p_5 + 6) \\ p_3 &> 4M + m(3p_4 + 4p_5 + 6) \\ p_4 &> 4M + m(4p_5 + 6) \\ p_5 &> 4M + 6m. \end{aligned}$$

Now, we build the graph $G = (V, E)$ of I_{ST} (see Figure 6.2). The set of the vertices is:

$$V = V_0 \cup V_1 \cup X \cup Y \cup Z \cup U_1 \cup U_2 \cup U_3 \cup U_4 \cup U_5$$

where

- $V_0 = \{v_i^0 : i = 1, 2, \ldots, M\}$
- $V_1 = \{v_i^1 : i = 1, 2, \ldots, M\}$
- $X = \{v_i^X : i = 1, 2, \ldots, m\}$
- $Y = \{v_i^Y : i = 1, 2, \ldots, m\}$
- $Z = \{v_i^Z : i = 1, 2, \ldots, m\}$
- $U_1 = \{u_{i,j}^1 : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, p_1\}$
- $U_2 = \{u_{i,j}^2 : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, p_2\}$
- $U_3 = \{u_{i,j}^3 : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, p_3\}$

- $U_4 = \{u^4_{i,j} : i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, p_4\}$
- $U_5 = \{u^5_{i,j} : i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, p_5\}$

Let us define the edge set of the graph of the I_{ST}.

$$E = K_{V_0,V_1} \cup E_X \cup E_Y \cup E_Z \cup E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$$

where each edge set shown in the formula is defined in the following list:

- $K_{V_0,V_1} = \{(v^0_i, v^1_j) : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, M\}$
- $E_X = \{(v^X_i, v^1_j) :$ if there is a clause $c_j \in C$ such that $x_i \in c_j\}$
- $E_Y = \{(v^Y_i, v^1_j) :$ if there is a clause $c_j \in C$ such that $Y_i \in c_j\}$
- $E_Z = \{(v^Z_i, v^1_j) :$ if there is a clause $c_j \in C$ such that $Z_i \in c_j\}$
- $E_1 = \{(u^1_{i,j}, u^1_{i,j+1}) : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, p_1 - 1\}$
  $\cup \{(v^X_i, u^1_{i,1}) : i = 1, 2, \ldots, m\}$
  $\cup \{(v^0_i, u^1_{i,1}) : i = m + 1, m + 2, \ldots, M\}$
- $E_2 = \{(u^2_{i,j}, u^2_{i,j+1}) : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, p_2 - 1\}$
  $\cup \{(v^0_i, u^2_{i,1}) : i = 1, 2, \ldots, M\}$
- $E_3 = \{(u^3_{i,j}, u^3_{i,j+1}) : i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, p_3 - 1\}$
  $\cup \{(v^0_i, u^2_{i,1}) : i = 1, 2, \ldots, M\}$
- $E_4 = \{(u^4_{i,j}, u^4_{i,j+1}) : i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, p_4 - 1\}$
  $\cup \{(v^Y_i, u^2_{i,1}) : i = 1, 2, \ldots, m\}$
- $E_5 = \{(u^5_{i,j}, u^5_{i,j+1}) : i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, p_5 - 1\}$
  $\cup \{(v^Z_i, u^2_{i,1}) : i = 1, 2, \ldots, m\}$

The edge sets E_1, E_2, E_3, E_4, E_5 represent sets of paths of length, respectively, p_1, p_2, p_3, p_4, p_5. We define the value of k as:

$$k = \frac{M}{2}\Big(p_1(p_1 + 1) + p_2(p_2 + 1) + p_3(p_3 + 1)\Big) + \frac{m}{2}\Big(p_4(p_4 + 1) + p_5(p_5 + 1)\Big) +$$

$$+M(p_2 + 2p_3 + 4) + m(2p_1 + 3p_4 + 4p_5 + 6).$$

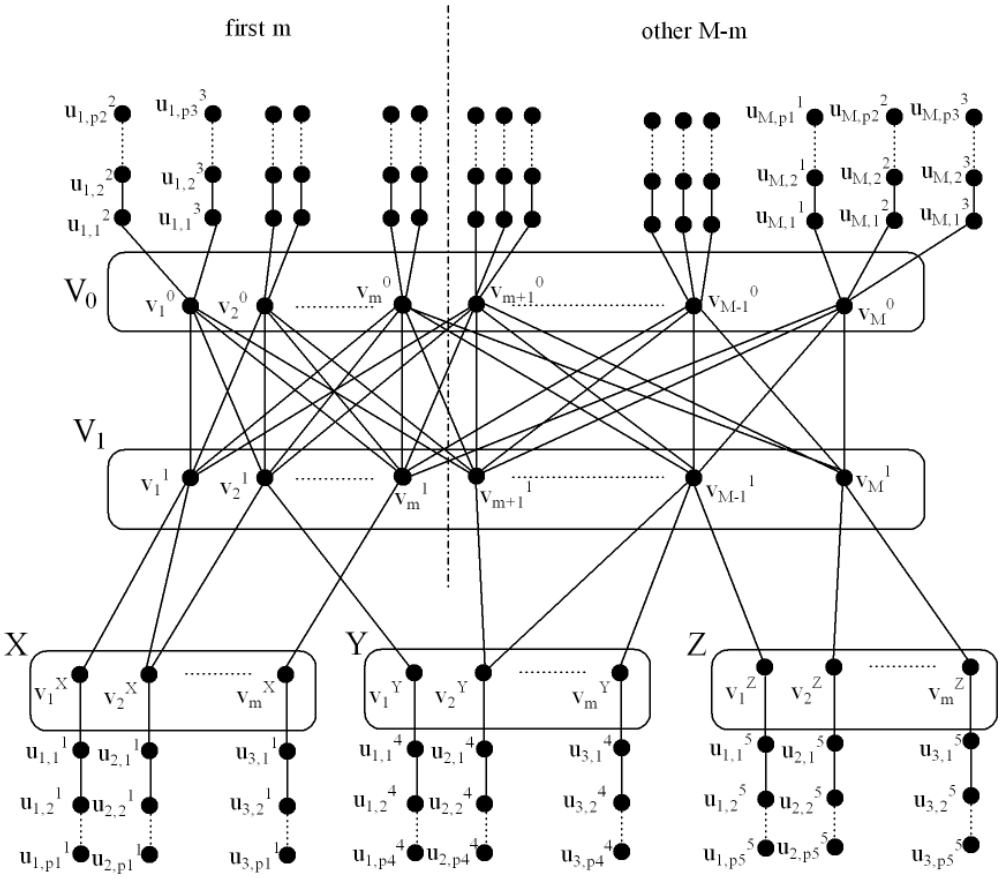

Figure 6.2: The construction of I_{ST}.

First we prove that if I_{3DM} is a YES-Instance then I_{ST} is a YES-Instance. Consider a solution, $C_{SOL} \subseteq C$, of I_{3DM}, such that $C_{SOL} = \{c_{s_1}, c_{s_2}, \ldots, c_{s_m}\}$. We name the clauses that are not in the solution as $C \setminus C_{SOL} = \{c_{t_1}, c_{t_2}, \ldots, c_{t_{M-m}}\}$. Let us consider the following broadcast, B^*, on I_{ST}:

Round 1**:**

send the message through $(v_i^0, v_{s_i}^1)$ for all $i = 1, 2, \ldots, m$ such that the vertex $v_{s_i}^1$ corresponds to the clause c_{s_i} in the solution C_{SOL};
send the message through $(v_i^0, u_{i,1}^1)$ for all $i = m+1, m+2, \ldots, M$;

Round 2**:**

send the message through $(v_i^0, u_{i,1}^2)$ for all $i = 1, 2, \ldots, M$;
send the message through $(v_{s_i}^1, v_i^X)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,1}^1, u_{i,2}^1)$ for all $i = m+1, m+2, \ldots, M$;

Round 3**:**

send the message through $(v_i^0, u_{i,1}^3)$ for all $i = 1, 2, \ldots, M$;
send the message through $(v_{s_i}^1, v_i^Y)$ for all $i = 1, 2, \ldots, m$;
send the message through $(v_i^X, u_{i,1}^1)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,2}^1, u_{i,3}^1)$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(u_{i,1}^2, u_{i,2}^2)$ for all $i = 1, 2, \ldots, M$;

Round 4**:**

send the message through $(v_i^0, v_{t_{i-m}}^1)$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(v_{s_i}^1, v_i^Z)$ for all $i = 1, 2, \ldots, m$;
send the message through $(v_i^Y, u_{i,1}^4)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,3}^1, u_{i,4}^1)$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(u_{i,1}^1, u_{i,2}^1)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,2}^2, u_{i,3}^2)$ for all $i = 1, 2, \ldots, M$;
send the message through $(u_{i,1}^3, u_{i,2}^3)$ for all $i = 1, 2, \ldots, M$;

Round 5

send the message through $(v_i^Z, u_{i,1}^5)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,5}^1, u_{i,6}^1)$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(u_{i,2}^1, u_{i,3}^1)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,3}^2, u_{i,4}^2)$ for all $i = 1, 2, \ldots, M$;
send the message through $(u_{i,2}^3, u_{i,3}^3)$ for all $i = 1, 2, \ldots, M$;
send the message through $(u_{i,1}^4, u_{i,2}^4)$ for all $i = 1, 2, \ldots, m$;

Round 6 **and the followings:**

send the message through each edge $(v^h_{i,j}, v^h_{i,j+1})$ such that, at round 6, the vertex $v^h_{i,j}$ knows the message and the vertex $v^h_{i,j+1}$ does not know the message.

Let us count the sum of the times $t(w)$ for all $w \in V$. For each $i = 1, 2, \ldots, M$ we have $t(v^0_i) = 0$ and $t(v^1_i) = 1$. For each $i = m+1, m+2, \ldots, M$ we have $t(v^1_i) = 4$. For each $i = 1, 2, \ldots, m$ we have $t(v^X_i) = 2$, $t(v^Y_i) = 3$ and $t(v^Z_i) = 4$. Thus the sum of the times in which the vertices in V_0, V_1, X, Y, Z receive the message is $4M - 6m$. The sum of the times for each vertex in U_1 is:

$$\sum_{w \in U_1} t(w) = \sum_{\substack{i \in \{1,\ldots,m\} \\ j \in \{1,\ldots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i \in \{m+1,\ldots,M\} \\ j \in \{1,\ldots,p_1\}}} t(u^1_{i,j}) =$$

since, for all $i \in \{1, \ldots, M\}$, the vertices $u^1_{i,1}, u^1_{i,2}, \ldots, u^1_{i,p_1}$ create a path in I_{ST} connected to the graph through a unique edge, adjacent to $u^1_{i,1}$, by Lemma 12 we have:

$$= \sum_{i \in \{1,\ldots,m\}} \Big(2p_1 + \frac{p_1(p_1+1)}{2}\Big) + \sum_{i \in \{m+1,\ldots,M\}} \frac{p_1(p_1+1)}{2} =$$

$$= m\Big(\frac{p_1(p_1+1)}{2} + 2p_1\Big) + (M-m)\frac{p_1(p_1+1)}{2} = M\frac{p_1(p_1+1)}{2} + 2mp_1.$$

Similarly, we observe that, for all $i \in \{1, \ldots, M\}$ and $h \in \{2, 3\}$, the vertices $u^h_{i,1}, u^h_{i,2}, \ldots, u^h_{i,p_h}$ create a path in I_{ST} connected to the graph through a unique edge, adjacent to $u^h_{i,1}$, and, for all $i \in \{1, \ldots, m\}$ and $h \in \{4, 5\}$, the vertices $u^h_{i,1}, u^h_{i,2}, \ldots, u^h_{i,p_h}$ create a path in I_{ST} connected to the graph through a unique edge, adjacent to $u^h_{i,1}$. Hence, by Lemma 12, we obtain the sums of the times of the vertices of the sets U_2, U_3, U_4, U_5,

$$\sum_{w \in U_2} t(w) = M\Big(p_2 + \frac{p_2(p_2+1)}{2}\Big),$$

$$\sum_{w \in U_3} t(w) = M\Big(2p_3 + \frac{p_3(p_3+1)}{2}\Big),$$

$$\sum_{w \in U_4} t(w) = m\Big(3p_4 + \frac{p_4(p_4+1)}{2}\Big),$$

$$\sum_{w \in U_5} t(w) = m\Big(4p_5 + \frac{p_5(p_5+1)}{2}\Big),$$

thus, the sum of the information times for each vertex of V in B^* is:

$$\sum_{w \in V} t(w) = \frac{M}{2}\Big(p_1(p_1{+}1){+}p_2(p_2{+}1){+}p_3(p_3{+}1)\Big){+}\frac{m}{2}\Big(p_4(p_4{+}1){+}p_5(p_5{+}1)\Big){+}$$

$$+M(p_2 + 2p_3 + 4) + m(2p_1 + 3p_4 + 4p_5 - 6).$$

We conclude that there is a broadcast for I_{ST} with a total time equal to k, hence I_{ST} is a YES-Instance.

Now we show that if I_{ST} is a YES-Instance then I_{3DM} is a YES-Instance. This proof is organized in the following way: initially, we prove five statements, Claim I,II,III,IV,V, finally, we conclude the proof using the statements.

Claim I: if I_{ST} is a YES-Instance then, in any broadcast that satisfies the instance, $t(u^1_{i,1}) = 3$, for all $i \in \{1, \ldots, m\}$, and $t(u^1_{i,1}) = 1$, for all $i \in \{m+1, \ldots, M\}$.

Proof. In order to prove Claim I, we observe that, for all $i \in \{m+1, \ldots, M\}$, $t(u^1_{i,1})$ can not be lesser than 1, because $u^1_{i,1} \notin V_0$. We also observe that, for all $i \in \{1, \ldots, m\}$, $t(u^1_{i,1})$ can not be lesser than 3, because $\min_{w \in V_0} d(w, u^1_{i,1}) = 3$, where $d(a, b)$ means the distance between vertices a and b in the graph. So, we have:

$$\begin{array}{ll} t(u^1_{i,1}) \geq 3 & \forall i \in \{1, \ldots, m\} \\ t(u^1_{i,1}) \geq 1 & \forall i \in \{m+1, \ldots, M\}. \end{array}$$

Now, consider a vertex $u^1_{a,1}$, with $a \in \{1, \ldots, M\}$, that does not receive the information as soon as possible, but with a delay of at least 1. Consider the following cases: (i) $a \in \{1, \ldots, m\}$ or (ii) $a \in \{m+1, \ldots, M\}$. In case (i) we have $t(u^1_{a,1}) \geq 4$. So that, by Lemma 12, the sum of information times for U_1 is bounded as follows:

$$\sum_{\substack{i \in \{1,\ldots,M\} \\ j \in \{1,\ldots,p_1\}}} t(u^1_{i,j}) = \sum_{j \in \{1,\ldots,p_1\}} t(u^1_{a,j}) + \sum_{\substack{i \in \{1,\ldots,m\}, i \neq a \\ j \in \{1,\ldots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i \in \{m+1,\ldots,M\} \\ j \in \{1,\ldots,p_1\}}} t(u^1_{i,j}) \geq$$

$$3p_1 + \frac{p_1(p_1+1)}{2} + \sum_{i \in \{1,\ldots,m\}, i \neq a} \Big(2p_1 + \frac{p_1(p_1+1)}{2}\Big) + \sum_{i \in \{m+1,\ldots,M\}} \frac{p_1(p_1+1)}{2} =$$

$$= 3p_1 + \frac{p_1(p_1+1)}{2} + (m-1)\Big(2p_1 + \frac{p_1(p_1+1)}{2}\Big) + (M-m)\frac{p_1(p_1+1)}{2} =$$

$$= 3p_1 + \frac{p_1(p_1+1)}{2} + 2mp_1 + m\frac{p_1(p_1+1)}{2} - 2p_1 +$$

$$-\frac{p_1(p_1+1)}{2} + M\frac{p_1(p_1+1)}{2} - m\frac{p_1(p_1+1)}{2} =$$

$$= M\frac{p_1(p_1+1)}{2} + 2mp_1 + p_1.$$

Let us analyze the case (ii). In this case we have $t(u^1_{a,1}) \geq 2$. Again, by Lemma 12, the sum of information times for U_1 is bounded as follows:

$$\sum_{\substack{i\in\{1,\ldots,M\}\\ j\in\{1,\ldots,p_1\}}} t(u^1_{i,j}) = \sum_{\substack{i\in\{1,\ldots,m\}\\ j\in\{1,\ldots,p_1\}}} t(u^1_{i,j}) + \sum_{j\in\{1,\ldots,p_1\}} t(u^1_{a,j}) + \sum_{\substack{i\in\{m+1,\ldots,M\},i\neq a\\ j\in\{1,\ldots,p_1\}}} t(u^1_{i,j}) \geq$$

$$\sum_{i\in\{1,\ldots,m\}} \Big(2p_1 + \frac{p_1(p_1+1)}{2}\Big) + p_1 + \frac{p_1(p_1+1)}{2} + \sum_{i\in\{m+1,\ldots,M\},i\neq a} \frac{p_1(p_1+1)}{2} =$$

$$= m\Big(2p_1 + \frac{p_1(p_1+1)}{2}\Big) + p_1 + \frac{p_1(p_1+1)}{2} + (M-m-1)\frac{p_1(p_1+1)}{2} =$$

$$= 2mp_1 + m\frac{p_1(p_1+1)}{2} + p_1 + \frac{p_1(p_1+1)}{2} +$$

$$+M\frac{p_1(p_1+1)}{2} - m\frac{p_1(p_1+1)}{2} - \frac{p_1(p_1+1)}{2} =$$

$$= M\frac{p_1(p_1+1)}{2} + 2mp_1 + p_1.$$

Hence, in both the cases, (i) and (ii), we have:

$$\sum_{\substack{i\in\{1,\ldots,M\}\\ j\in\{1,\ldots,p_1\}}} t(u^1_{i,j}) \geq M\frac{p_1(p_1+1)}{2} + 2mp_1 + p_1.$$

Now, for all $w \in U_2 \cup U_3 \cup U_4 \cup U_5$, we observe that $t(w) \geq 1$, because $w \notin V_0$. Furthermore, in I_{ST}, for all $i \in \{1,\ldots,M\}$ and $h \in \{2,3\}$, the vertices $u^h_{i,1}, u^h_{i,2}, \ldots, u^h_{i,p_h}$ create a path, connected to the graph through a unique edge, adjacent to $u^h_{i,1}$, and, for all $i \in \{1,\ldots,m\}$ and $h \in \{4,5\}$, the vertices $u^h_{i,1}, u^h_{i,2}, \ldots, u^h_{i,p_h}$ create a path, connected to the graph through a unique edge,

adjacent to $u^h_{i,1}$. Hence, by Lemma 12, we bound the sums of the times of the vertices of the sets U_2, U_3, U_4, U_5 as follows:

$$\sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_2\}}} t(u^2_{i,j}) \geq \sum_{i\in\{1,\dots,M\}} \frac{p_2(p_2+1)}{2} = M\frac{p_2(p_2+1)}{2},$$

$$\sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) \geq \sum_{i\in\{1,\dots,M\}} \frac{p_3(p_3+1)}{2} = M\frac{p_3(p_3+1)}{2},$$

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) \geq \sum_{i\in\{1,\dots,m\}} \frac{p_4(p_4+1)}{2} = m\frac{p_4(p_4+1)}{2},$$

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq \sum_{i\in\{1,\dots,m\}} \frac{p_5(p_5+1)}{2} = m\frac{p_5(p_5+1)}{2}.$$

Therefore, the sum of all the information times is:

$$\sum_{w\in V} t(w) = \sum_{w\in V_0\cup V_1\cup X\cup Y\cup Z} t(w) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_2\}}} t(u^2_{i,j}) +$$

$$+ \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq$$

$$M\frac{p_1(p_1+1)}{2} + 2mp_1 + p_1 + M\frac{p_2(p_2+1)}{2} + M\frac{p_3(p_3+1)}{2} +$$

$$+m\frac{p_4(p_4+1)}{2} + m\frac{p_5(p_5+1)}{2} =$$

$$= \frac{M}{2}\Big(p_1(p_1{+}1){+}p_2(p_2{+}1){+}p_3(p_3{+}1)\Big){+}\frac{m}{2}\Big(p_4(p_4{+}1){+}p_5(p_5{+}1)\Big){+}2mp_1{+}p_1.$$

Thanks to our choice of p_1, we bound the previous term substituting the last occurrence of p_1:

$$\sum_{w\in V} t(w) > \frac{M}{2}\Big(p_1(p_1{+}1){+}p_2(p_2{+}1){+}p_3(p_3{+}1)\Big){+}\frac{m}{2}\Big(p_4(p_4{+}1){+}p_5(p_5{+}1)\Big){+}$$

$$+M(p_2+2p_3+4) + m(2p_1+3p_4+4p_5+6) = k$$

that is a contradiction (I_{ST} is a YES-Instance). Thus, we conclude the proof of Claim I, i.e. if I_{ST} is a YES-Instance then in any broadcast that satisfies the instance $t(u^1_{i,1}) = 3$, for all $i \in \{1, \ldots, m\}$, and $t(u^1_{i,1}) = 1$, for all $i \in \{m+1, \ldots, M\}$.

Therefore, the broadcast process that satisfies I_{ST} must be as the following partial broadcast B_1:

Round 1:
send the message through $(v^0_i, v^1_{s_i})$ for all $i = 1, 2, \ldots, m$ for some sequence $s_1, s_2, \ldots, s_m$ such that in I_{3DM} the clauses $c_{s_1}, c_{s_2}, \ldots, c_{s_m}$ are a cover for the elements $x_1, x_2, \ldots, x_m$;
send the message through $(v^0_i, u^1_{i,1})$ for all $i = m+1, m+2, \ldots, M$;

Round 2:
send the message through $(v^1_{s_i}, v^X_i)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u^1_{i,1}, u^1_{i,2})$ for all $i = m+1, m+2, \ldots, M$;

Round 3:
send the message through $(v^X_i, u^1_{i,1})$ for all $i = 1, 2, \ldots, m$;
send the message through $(u^1_{i,2}, u^1_{i,3})$ for all $i = m+1, m+2, \ldots, M$;

Round 4 and the followings:
send the message through each edge $(v^h_{i,j}, v^h_{i,j+1})$ such that, at round 4, the vertex $v^h_{i,j}$ knows the message and the vertex $v^h_{i,j+1}$ does not know the message.

Claim I implies that, if I_{ST} is a YES-Instance, in a broadcast that satisfies it, the sum of the information times of U_1 is:

$$\sum_{\substack{i \in \{1,\ldots,M\} \\ j \in \{1,\ldots,p_1\}}} t(u^1_{i,j}) \geq M \frac{p_1(p_1+1)}{2} + 2mp_1. \tag{6.1}$$

Claim II: if I_{ST} is a YES-Instance then, in any broadcast that satisfies the instance, $t(u^2_{i,1}) = 2$, for all $i \in \{1, \ldots, M\}$.

Proof. Firstly, we note that $t(u^2_{i,1})$, for all $i \in \{1, \ldots, M\}$, can not be lesser than 2, because at time 1, as we shown in B_1, only vertices in U_1 or V_1 can be informed. Secondly, if there is a vertex $u^2_{a,1}$ such that $t(u^2_{a,1}) \geq 3$, we consider

the path composed by the vertices $u^2_{a,j}$, with $j \in \{1, \ldots, p_2\}$. By Lemma 12, we have that:

$$\sum_{j \in \{1,\ldots,p_2\}} t(u^2_{a,j}) \geq 2p_2 + \frac{p_2(p_2+1)}{2}.$$

For all the other $i \in \{1, \ldots, M\}$ with $i \neq a$, since it must be $t(u^2_{i,1}) \geq 2$, again by Lemma 12, we have:

$$\sum_{j \in \{1,\ldots,p_2\}} t(u^2_{i,j}) \geq p_2 + \frac{p_2(p_2+1)}{2}.$$

So we can bound the sum of the information times of U_2 as follows:

$$\sum_{\substack{i \in \{1,\ldots,M\} \\ j \in \{1,\ldots,p_2\}}} t(u^2_{i,j}) = \sum_{j \in \{1,\ldots,p_2\}} t(u^2_{a,j}) + \sum_{\substack{i \in \{1,\ldots,M\}, i \neq a \\ j \in \{1,\ldots,p_2\}}} t(u^2_{i,j}) \geq$$

$$2p_2 + \frac{p_2(p_2+1)}{2} + \sum_{i \in \{1,\ldots,M\}, i \neq a} \Big(p_2 + \frac{p_2(p_2+1)}{2}\Big) =$$

$$= 2p_2 + \frac{p_2(p_2+1)}{2} + (M-1)\Big(p_2 + \frac{p_2(p_2+1)}{2}\Big) =$$

$$= M\frac{p_2(p_2+1)}{2} + Mp_2 + p_2.$$

Now, for all $w \in U_3 \cup U_4 \cup U_5$, we observe that $t(w) \geq 1$, because $w \notin V_0$. Furthermore, in I_{ST}, for all $i \in \{1, \ldots, M\}$, the vertices $u^3_{i,1}, u^3_{i,2}, \ldots, u^3_{i,p_3}$ create a path, connected to the graph through a unique edge, adjacent to $u^3_{i,1}$, and, for all $i \in \{1, \ldots, m\}$ and $h \in \{4, 5\}$, the vertices $u^h_{i,1}, u^h_{i,2}, \ldots, u^h_{i,p_h}$ create a path, connected to the graph through a unique edge, adjacent to $u^h_{i,1}$. Hence, by Lemma 12, we bound the sums of the times of the vertices of the sets U_3, U_4, U_5 as follows:

$$\sum_{\substack{i \in \{1,\ldots,M\} \\ j \in \{1,\ldots,p_3\}}} t(u^3_{i,j}) \geq \sum_{i \in \{1,\ldots,M\}} \frac{p_3(p_3+1)}{2} = M\frac{p_3(p_3+1)}{2},$$

$$\sum_{\substack{i \in \{1,\ldots,m\} \\ j \in \{1,\ldots,p_4\}}} t(u^4_{i,j}) \geq \sum_{i \in \{1,\ldots,m\}} \frac{p_4(p_4+1)}{2} = m\frac{p_4(p_4+1)}{2},$$

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq \sum_{i\in\{1,\dots,m\}} \frac{p_5(p_5+1)}{2} = m\frac{p_5(p_5+1)}{2}.$$

Therefore, using (6.1), we calculate the sum of all the information times:

$$\sum_{w\in V} t(w) = \sum_{w\in V_0\cup V_1\cup X\cup Y\cup Z} t(w) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_2\}}} t(u^2_{i,j}) +$$

$$+ \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq$$

$$M\frac{p_1(p_1+1)}{2} + 2mp_1 + M\frac{p_2(p_2+1)}{2} + Mp_2 + p_2 + M\frac{p_3(p_3+1)}{2} +$$

$$+m\frac{p_4(p_4+1)}{2} + m\frac{p_5(p_5+1)}{2} =$$

$$= \frac{M}{2}\Big(p_1(p_1+1) + p_2(p_2+1) + p_3(p_3+1)\Big) + \frac{m}{2}\Big(p_4(p_4+1) + p_5(p_5+1)\Big) +$$

$$+2mp_1 + Mp_2 + p_2.$$

Thanks to our choice of p_2, we bound the previous term substituting the last occurrence of p_2:

$$\sum_{w\in V} t(w) > \frac{M}{2}\Big(p_1(p_1+1)+p_2(p_2+1)+p_3(p_3+1)\Big)+\frac{m}{2}\Big(p_4(p_4+1)+p_5(p_5+1)\Big)+$$

$$+M(p_2+2p_3+4)+m(2p_1+3p_4+4p_5+6) = k$$

that is a contradiction (I_{ST} is a YES-Instance). Thus, we conclude the proof of Claim II, i.e. if I_{ST} is a YES-Instance then in any broadcast that satisfies the instance $t(u^2_{i,1}) = 2$, for all $i \in \{1,\dots,M\}$.

Therefore, the broadcast process that satisfies I_{ST} must be as the following partial broadcast B_2:

Round 1**:**

send the message through $(v^0_i, v^1_{s_i})$ for all $i = 1, 2, \dots, m$ for some sequence $s_1, s_2, \dots, s_m$ such that in I_{3DM} the clauses $c_{s_1}, c_{s_2}, \dots, c_{s_m}$ are a cover for the elements $x_1, x_2, \dots, x_m$;
send the message through $(v^0_i, u^1_{i,1})$ for all $i = m+1, m+2, \dots, M$;

Round 2**:**
send the message through $(v_i^0, u_{i,1}^2)$ for all $i = 1, 2, \ldots, M$;
send the message through $(v_{s_i}^1, v_i^X)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,1}^1, u_{i,2}^1)$ for all $i = m+1, m+2, \ldots, M$;

Round 3**:**
send the message through $(v_i^X, u_{i,1}^1)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,2}^1, u_{i,3}^1)$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(u_{i,1}^2, u_{i,2}^2)$ for all $i = 1, 2, \ldots, M$;

Round 4 **and the followings:**
send the message through each edge $(v_{i,j}^h, v_{i,j+1}^h)$ such that, at round 4, the vertex $v_{i,j}^h$ knows the message and the vertex $v_{i,j+1}^h$ does not know the message.

Claim II implies that, if I_{ST} is a YES-Instance, in a broadcast that satisfies it, the sum of the information times of U_2 is:

$$\sum_{\substack{i\in\{1,\ldots,M\}\\ j\in\{1,\ldots,p_2\}}} t(u_{i,j}^2) \geq M\frac{p_2(p_2+1)}{2} + Mp_2. \tag{6.2}$$

Claim III: if I_{ST} is a YES-Instance then, in any broadcast that satisfies the instance, $t(u_{i,1}^3) = 3$, for all $i \in \{1, \ldots, M\}$.

Proof. Initially, we observe that $t(u_{i,1}^3)$, for all $i \in \{1, \ldots, M\}$, can not be lesser than 3, because at time 1 amd 2, as we shown in B_2, vertices in V_0 transmit towards vertices U_1 and U_2. Instead, if there is a vertex $u_{a,1}^3$ such that $t(u_{a,1}^3) \geq 4$, we consider the path composed by the vertices $u_{a,j}^3$, with $j \in \{1, \ldots, p_3\}$. By Lemma 12, we have that:

$$\sum_{j\in\{1,\ldots,p_3\}} t(u_{a,j}^3) \geq 3p_3 + \frac{p_3(p_3+1)}{2}.$$

For all the other $i \in \{1, \ldots, M\}$ with $i \neq a$, since it must be $t(u_{i,1}^3) \geq 3$, again by Lemma 12, we have:

$$\sum_{j\in\{1,\ldots,p_3\}} t(u_{i,j}^3) \geq 2p_3 + \frac{p_3(p_3+1)}{2}.$$

So we can bound the sum of the information times of U_3 as follows:

$$\sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) = \sum_{j\in\{1,\dots,p_3\}} t(u^3_{a,j}) + \sum_{\substack{i\in\{1,\dots,M\},i\neq a\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) \geq$$

$$3p_3 + \frac{p_3(p_3+1)}{2} + \sum_{i\in\{1,\dots,M\},i\neq a} \Big(2p_3 + \frac{p_3(p_3+1)}{2}\Big) =$$

$$= 3p_3 + \frac{p_3(p_3+1)}{2} + (M-1)\Big(2p_3 + \frac{p_3(p_3+1)}{2}\Big) =$$

$$= M\frac{p_3(p_3+1)}{2} + 2Mp_3 + p_3.$$

Now, for all $w \in U_4 \cup U_5$, we observe that $t(w) \geq 1$, because $w \notin V_0$. Furthermore, in I_{ST}, for all $i \in \{1,\dots,m\}$ and $h \in \{4,5\}$, the vertices $u^h_{i,1}, u^h_{i,2}, \dots, u^h_{i,p_h}$ create a path, connected to the graph through a unique edge, adjacent to $u^h_{i,1}$. Hence, by Lemma 12, we bound the sums of the times of the vertices of the sets U_4, U_5 as follows:

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) \geq \sum_{i\in\{1,\dots,m\}} \frac{p_4(p_4+1)}{2} = m\frac{p_4(p_4+1)}{2},$$

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq \sum_{i\in\{1,\dots,m\}} \frac{p_5(p_5+1)}{2} = m\frac{p_5(p_5+1)}{2}.$$

Therefore, using (6.1), (6.2), we calculate the sum of all the information times:

$$\sum_{w\in V} t(w) = \sum_{w\in V_0\cup V_1\cup X\cup Y\cup Z} t(w) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_2\}}} t(u^2_{i,j}) +$$

$$+ \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq$$

$$M\frac{p_1(p_1+1)}{2} + 2mp_1 + M\frac{p_2(p_2+1)}{2} + Mp_2 + M\frac{p_3(p_3+1)}{2} +$$

$$+2Mp_3 + p_3 + m\frac{p_4(p_4+1)}{2} + m\frac{p_5(p_5+1)}{2} =$$

$$= \frac{M}{2}\Big(p_1(p_1+1)+p_2(p_2+1)+p_3(p_3+1)\Big)+\frac{m}{2}\Big(p_4(p_4+1)+p_5(p_5+1)\Big)+$$
$$+2mp_1+Mp_2+2Mp_3+p_3.$$

Thanks to our choice of p_3, we bound the previous term substituting the last occurrence of p_3:

$$\sum_{w\in V} t(w) > \frac{M}{2}\Big(p_1(p_1+1)+p_2(p_2+1)+p_3(p_3+1)\Big)+\frac{m}{2}\Big(p_4(p_4+1)+p_5(p_5+1)\Big)+$$
$$+M(p_2+2p_3+4)+m(2p_1+3p_4+4p_5+6)=k$$

that is a contradiction (I_{ST} is a YES-Instance). Thus, we conclude the proof of Claim III, i.e. if I_{ST} is a YES-Instance then in any broadcast that satisfies the instance $t(u^3_{i,1})=3$, for all $i\in\{1,\dots,M\}$.

Therefore, the broadcast process that satisfies I_{ST} must be as the following partial broadcast B_3:

Round 1:
send the message through $(v^0_i, v^1_{s_i})$ for all $i=1,2,\dots,m$ for some sequence $s_1,s_2,\dots,s_m$ such that in I_{3DM} the clauses $c_{s_1},c_{s_2},\dots,c_{s_m}$ are a cover for the elements $x_1,x_2,\dots,x_m$;
send the message through $(v^0_i, u^1_{i,1})$ for all $i=m+1,m+2,\dots,M$;

Round 2:
send the message through $(v^0_i, u^2_{i,1})$ for all $i=1,2,\dots,M$;
send the message through $(v^1_{s_i}, v^X_i)$ for all $i=1,2,\dots,m$;
send the message through $(u^1_{i,1}, u^1_{i,2})$ for all $i=m+1,m+2,\dots,M$;

Round 3:
send the message through $(v^0_i, u^3_{i,1})$ for all $i=1,2,\dots,M$;
send the message through $(v^X_i, u^1_{i,1})$ for all $i=1,2,\dots,m$;
send the message through $(u^1_{i,2}, u^1_{i,3})$ for all $i=m+1,m+2,\dots,M$;
send the message through $(u^2_{i,1}, u^2_{i,2})$ for all $i=1,2,\dots,M$;

Round 4 and the followings:
send the message through each edge $(v^h_{i,j}, v^h_{i,j+1})$ such that, at round 4, the vertex $v^h_{i,j}$ knows the message and the vertex $v^h_{i,j+1}$ does not know the message.

CLAIM III implies that, if I_{ST} is a YES-Instance, in a broadcast that satisfies it, the sum of the information times of U_3 is:

$$\sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) \geq M\frac{p_3(p_3+1)}{2} + 2Mp_3. \tag{6.3}$$

CLAIM IV: if I_{ST} is a YES-Instance then, in any broadcast that satisfies the instance, $t(u^4_{i,1}) = 4$, for all $i \in \{1,\dots,m\}$.

Proof. At first, we prove that $t(u^4_{i,1})$, for all $i \in \{1,\dots,m\}$, can not be lesser than 4. Consider a path P, from a vertex $w \in V_0$ to $u^4_{i,1}$. We observe that, in I_{ST}, P must be long at least 3. Moreover, if P is long 3, then it must be like $w, v^1_a, v^Y_b, u^4_{i,1}$, for some $a \in \{1,\dots,M\}$ and $b \in \{1,\dots,m\}$. By CLAIM I we know that, at time 1, exactly m vertices of V_1 know the information and, at round 2, the m informed vertices of V_1 must transmit towards X. Thus, v^Y_b can not receive the message at time 2 and, thereby, $u^4_{i,1}$ can not receive the message at time 3. So $t(u^4_{i,1})$, for all $i \in \{1,\dots,m\}$, can not be lesser than 4. Let us prove that $t(u^4_{i,1})$, for all $i \in \{1,\dots,m\}$, can not be greater than 4. If there is a vertex $u^4_{a,1}$ such that $t(u^4_{a,1}) \geq 5$, we consider the path composed by the vertices $u^4_{a,j}$, with $j \in \{1,\dots,p_4\}$. By Lemma 12, we have that:

$$\sum_{j\in\{1,\dots,p_4\}} t(u^4_{a,j}) \geq 4p_4 + \frac{p_4(p_4+1)}{2}.$$

For all the other $i \in \{1,\dots,m\}$ with $i \neq a$, since it must be $t(u^4_{i,1}) \geq 4$, again by Lemma 12, we have:

$$\sum_{j\in\{1,\dots,p_4\}} t(u^4_{i,j}) \geq 3p_4 + \frac{p_4(p_4+1)}{2}.$$

So we can bound the sum of the information times of U_4 as follows:

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) = \sum_{j\in\{1,\dots,p_4\}} t(u^4_{a,j}) + \sum_{\substack{i\in\{1,\dots,m\},i\neq a\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) \geq$$

$$4p_4 + \frac{p_4(p_4+1)}{2} + \sum_{i\in\{1,\dots,m\},i\neq a} \left(3p_4 + \frac{p_4(p_4+1)}{2}\right) =$$

$$= 4p_4 + \frac{p_4(p_4+1)}{2} + (m-1)\Big(3p_4 + \frac{p_4(p_4+1)}{2}\Big) =$$

$$= m\frac{p_4(p_4+1)}{2} + 3mp_4 + p_4.$$

Now, for all $w \in U_5$, we observe that $t(w) \geq 1$, because $w \notin V_0$. Furthermore, in I_{ST}, for all $i \in \{1,\ldots,m\}$, the vertices $u^5_{i,1}, u^5_{i,2}, \ldots, u^5_{i,p_5}$ create a path, connected to the graph through a unique edge, adjacent to $u^5_{i,1}$. Hence, by Lemma 12, we bound the sum of the times of the vertices of the sets U_5 as follows:

$$\sum_{\substack{i\in\{1,\ldots,m\}\\ j\in\{1,\ldots,p_5\}}} t(u^5_{i,j}) \geq \sum_{i\in\{1,\ldots,m\}} \frac{p_5(p_5+1)}{2} = m\frac{p_5(p_5+1)}{2}.$$

Therefore, using (6.1), (6.2), (6.3), we calculate the sum of all the information times:

$$\sum_{w\in V} t(w) = \sum_{w\in V_0\cup V_1\cup X\cup Y\cup Z} t(w) + \sum_{\substack{i\in\{1,\ldots,M\}\\ j\in\{1,\ldots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i\in\{1,\ldots,M\}\\ j\in\{1,\ldots,p_2\}}} t(u^2_{i,j}) +$$

$$+ \sum_{\substack{i\in\{1,\ldots,M\}\\ j\in\{1,\ldots,p_3\}}} t(u^3_{i,j}) + \sum_{\substack{i\in\{1,\ldots,m\}\\ j\in\{1,\ldots,p_4\}}} t(u^4_{i,j}) + \sum_{\substack{i\in\{1,\ldots,m\}\\ j\in\{1,\ldots,p_5\}}} t(u^5_{i,j}) \geq$$

$$M\frac{p_1(p_1+1)}{2} + 2mp_1 + M\frac{p_2(p_2+1)}{2} + Mp_2 + M\frac{p_3(p_3+1)}{2} +$$

$$+2Mp_3 + m\frac{p_4(p_4+1)}{2} + 3mp_4 + p_4 + m\frac{p_5(p_5+1)}{2} =$$

$$= \frac{M}{2}\Big(p_1(p_1+1) + p_2(p_2+1) + p_3(p_3+1)\Big) + \frac{m}{2}\Big(p_4(p_4+1) + p_5(p_5+1)\Big) +$$

$$+2mp_1 + Mp_2 + 2Mp_3 + 3mp_4 + p_4.$$

Thanks to our choice of p_4, we bound the previous term substituting the last occurrence of p_4:

$$\sum_{w\in V} t(w) > \frac{M}{2}\Big(p_1(p_1+1)+p_2(p_2+1)+p_3(p_3+1)\Big)+\frac{m}{2}\Big(p_4(p_4+1)+p_5(p_5+1)\Big)+$$

$$+M(p_2+2p_3+4)+m(2p_1+3p_4+4p_5+6) = k$$

that is a contradiction (I_{ST} is a YES-Instance). Thus, we conclude the proof of Claim IV, i.e. if I_{ST} is a YES-Instance then in any broadcast that satisfies the instance $t(u^4_{i,1}) = 4$, for all $i \in \{1, \ldots, m\}$.

Therefore, the broadcast process that satisfies I_{ST} must be as the following partial broadcast B_4:

Round 1**:**
send the message through $(v^0_i, v^1_{s_i})$ for all $i = 1, 2, \ldots, m$ for some sequence $s_1, s_2, \ldots, s_m$ such that in I_{3DM} the clauses $c_{s_1}, c_{s_2}, \ldots, c_{s_m}$ are a cover for the elements $x_1, x_2, \ldots, x_m$;
send the message through $(v^0_i, u^1_{i,1})$ for all $i = m+1, m+2, \ldots, M$;

Round 2**:**
send the message through $(v^0_i, u^2_{i,1})$ for all $i = 1, 2, \ldots, M$;
send the message through $(v^1_{s_i}, v^X_i)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u^1_{i,1}, u^1_{i,2})$ for all $i = m+1, m+2, \ldots, M$;

Round 3**:**
send the message through $(v^0_i, u^3_{i,1})$ for all $i = 1, 2, \ldots, M$;
send the message through $(v^1_{s_i}, v^Y_i)$ for all $i = 1, 2, \ldots, m$;
send the message through $(v^X_i, u^1_{i,1})$ for all $i = 1, 2, \ldots, m$;
send the message through $(u^1_{i,2}, u^1_{i,3})$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(u^2_{i,1}, u^2_{i,2})$ for all $i = 1, 2, \ldots, M$;

Round 4 **and the followings:**
send the message through $(v^Y_i, u^4_{i,1})$ for all $i = 1, 2, \ldots, m$;
send the message through each edge $(v^h_{i,j}, v^h_{i,j+1})$ such that, at round 4, the vertex $v^h_{i,j}$ knows the message and the vertex $v^h_{i,j+1}$ does not know the message.

Claim IV implies that, if I_{ST} is a YES-Instance, in a broadcast that satisfies it, the sum of the information times of U_4 is:

$$\sum_{\substack{i \in \{1,\ldots,m\} \\ j \in \{1,\ldots,p_4\}}} t(u^4_{i,j}) \geq m\frac{p_4(p_4+1)}{2} + 3mp_4. \tag{6.4}$$

Claim IV: if I_{ST} is a YES-Instance then, in any broadcast that satisfies the instance, $t(u^5_{i,1}) = 5$, for all $i \in \{1, \ldots, m\}$.

Proof. Initially, we prove that $t(u^5_{i,1})$, for all $i \in \{1,\dots,m\}$, can not be lesser than 5. Consider a path P, from a vertex $w \in V_0$ to $u^5_{i,1}$. We observe that, in I_{ST}, P must be long at least 3. Moreover, if P is long 3, then it must be like $w, v^1_a, v^Z_b, u^5_{i,1}$, for some $a \in \{1,\dots,M\}$ and $b \in \{1,\dots,m\}$. By CLAIM I we know that, at time 1, exactly m vertices of V_1 know the information and, at round 2, the m informed vertices of V_1 must transmit towards X. By CLAIM IV we know that, at time 3, the m informed vertices of V_1 must transmit towards Y. Thus, v^Z_b can not receive the message at time 3 and, thereby, $u^5_{i,1}$ can not receive the message at time 4. So $t(u^5_{i,1})$, for all $i \in \{1,\dots,m\}$, can not be lesser than 5. Let us prove that $t(u^5_{i,1})$, for all $i \in \{1,\dots,m\}$, can not be greater than 5. If there is a vertex $u^5_{a,1}$ such that $t(u^5_{a,1}) \geq 6$, we consider the path composed by the vertices $u^5_{a,j}$, with $j \in \{1,\dots,p_5\}$. By Lemma 12, we have that:

$$\sum_{j\in\{1,\dots,p_5\}} t(u^5_{a,j}) \geq 5p_5 + \frac{p_5(p_5+1)}{2}.$$

For all the other $i \in \{1,\dots,m\}$ with $i \neq a$, since it must be $t(u^5_{i,1}) \geq 5$, again by Lemma 12, we have:

$$\sum_{j\in\{1,\dots,p_5\}} t(u^5_{i,j}) \geq 4p_5 + \frac{p_5(p_5+1)}{2}.$$

So we can bound the sum of the information times of U_5 as follows:

$$\sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) = \sum_{j\in\{1,\dots,p_5\}} t(u^5_{a,j}) + \sum_{\substack{i\in\{1,\dots,m\},i\neq a\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq$$

$$5p_5 + \frac{p_5(p_5+1)}{2} + \sum_{i\in\{1,\dots,m\},i\neq a} \Big(4p_5 + \frac{p_5(p_5+1)}{2}\Big) =$$

$$= 5p_5 + \frac{p_5(p_5+1)}{2} + (m-1)\Big(4p_5 + \frac{p_5(p_5+1)}{2}\Big) =$$

$$= m\frac{p_5(p_5+1)}{2} + 4mp_5 + p_5.$$

Therefore, using (6.1), (6.2), (6.3), (6.4), we calculate the sum of all the information times:

$$\sum_{w\in V} t(w) = \sum_{w\in V_0\cup V_1\cup X\cup Y\cup Z} t(w) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_1\}}} t(u^1_{i,j}) + \sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_2\}}} t(u^2_{i,j}) +$$

$$+\sum_{\substack{i\in\{1,\dots,M\}\\ j\in\{1,\dots,p_3\}}} t(u^3_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_4\}}} t(u^4_{i,j}) + \sum_{\substack{i\in\{1,\dots,m\}\\ j\in\{1,\dots,p_5\}}} t(u^5_{i,j}) \geq$$

$$M\frac{p_1(p_1+1)}{2} + 2mp_1 + M\frac{p_2(p_2+1)}{2} + Mp_2 + M\frac{p_3(p_3+1)}{2} +$$

$$+2Mp_3 + m\frac{p_4(p_4+1)}{2} + 3mp_4 + m\frac{p_5(p_5+1)}{2} + 4mp_5 + p_5 =$$

$$= \frac{M}{2}\Big(p_1(p_1+1) + p_2(p_2+1) + p_3(p_3+1)\Big) + \frac{m}{2}\Big(p_4(p_4+1) + p_5(p_5+1)\Big) +$$

$$+2mp_1 + Mp_2 + 2Mp_3 + 3mp_4 + 4mp_5 + p_5.$$

Thanks to our choice of p_5, we bound the previous term substituting the last occurrence of p_5:

$$\sum_{w\in V} t(w) > \frac{M}{2}\Big(p_1(p_1+1)+p_2(p_2+1)+p_3(p_3+1)\Big)+\frac{m}{2}\Big(p_4(p_4+1)+p_5(p_5+1)\Big)+$$

$$+M(p_2+2p_3+4)+m(2p_1+3p_4+4p_5+6) = k$$

that is a contradiction (I_{ST} is a YES-Instance). Thus, we conclude the proof of Claim V, i.e. if I_{ST} is a YES-Instance then in any broadcast that satisfies the instance $t(u^5_{i,1}) = 5$, for all $i \in \{1,\dots,m\}$.

Therefore, the broadcast process that satisfies I_{ST} must be as the following partial broadcast B_5:

Round 1:

send the message through $(v^0_i, v^1_{s_i})$ for all $i = 1, 2, \dots, m$ for some sequence $s_1, s_2, \dots, s_m$ such that in I_{3DM} the clauses $c_{s_1}, c_{s_2}, \dots, c_{s_m}$ are a cover for the elements $x_1, x_2, \dots, x_m$;
send the message through $(v^0_i, u^1_{i,1})$ for all $i = m+1, m+2, \dots, M$;

Round 2:

send the message through $(v^0_i, u^2_{i,1})$ for all $i = 1, 2, \dots, M$;
send the message through $(v^1_{s_i}, v^X_i)$ for all $i = 1, 2, \dots, m$;
send the message through $(u^1_{i,1}, u^1_{i,2})$ for all $i = m+1, m+2, \dots, M$;

Round 3:

send the message through $(v^0_i, u^3_{i,1})$ for all $i = 1, 2, \dots, M$;
send the message through $(v^1_{s_i}, v^Y_i)$ for all $i = 1, 2, \dots, m$;

send the message through $(v_i^X, u_{i,1}^1)$ for all $i = 1, 2, \ldots, m$;
send the message through $(u_{i,2}^1, u_{i,3}^1)$ for all $i = m+1, m+2, \ldots, M$;
send the message through $(u_{i,1}^2, u_{i,2}^2)$ for all $i = 1, 2, \ldots, M$;

Round 4

send the message through $(v_{s_i}^1, v_i^Z)$ for all $i = 1, 2, \ldots, m$;
send the message through $(v_i^Y, u_{i,1}^4)$ for all $i = 1, 2, \ldots, m$;
send the message through each edge $(v_{i,j}^h, v_{i,j+1}^h)$ such that, at round 4, the vertex $v_{i,j}^h$ knows the message and the vertex $v_{i,j+1}^h$ does not know the message.

Round 5 **and the followings:**

send the message through $(v_i^Z, u_{i,1}^5)$ for all $i = 1, 2, \ldots, m$;
send the message through each edge $(v_{i,j}^h, v_{i,j+1}^h)$ such that, at round 4, the vertex $v_{i,j}^h$ knows the message and the vertex $v_{i,j+1}^h$ does not know the message.

At this point, broadcast B_5 implies that, in I_{ST}, there are m vertices of V_1, $v_{s_1}^1, v_{s_2}^1, \ldots, v_{s_m}^1$, such that, in round 2 they transmit, simultaneously, the message to all the m vertices in X, in round 3 they transmit, simultaneously, the message to all the m vertices in Y, and in round 4 they transmit, simultaneously, the message to all the m vertices in Z. Since the vertices of V_1 represent the clauses and X,Y and Z represent the three elements sets in I_{3DM}, we have that $C^* = \{c_{s_1}, c_{s_2}, \ldots, c_{s_m}\}$ must be a solution for I_{3DM}, so that I_{3DM} is a YES-Instance. □

Theorem 7. *Problem ST-r, i.e. the problem ST in which in V_0 there is only one vertex, named r, is NP-complete.*

Proof. We will reduce the problem ST to the problem ST-r. Given an instance I_{ST} of the problem ST, we build an instance of I_{ST-r} of the problem ST-r. Then, we prove that, I_{ST-r} is a YES-Instance if and only if I_{ST} is a YES-Instance. The idea of the proof is to build the instance I_{ST-r} like the instance I_{ST}, but with a further structure such that, any broadcast must disseminate the message so that the vertices in V_0 receive it at the same time. After that vertices in V_0 have received the message, the broadcast continues as a broadcast in I_{ST}.

Consider an instance I_{ST} in which we have a graph $G = (V, E)$, a set $V_0 \subseteq V$ and a positive integer k. We build I_{ST-r}, i.e. $G' = (V', E'), r \in V', k'$,

as follows (see Figure 6.3). Let $q = |V_0|$, $n = |V|$ and $V_0 = \{a_1, a_2, \ldots, a_q\}$. Consider an arbitrary integer p, such that:

$$p > k + n(q+1) + \frac{q(q^2+1)}{3} + \frac{q(q-1)}{2}.$$

We define the following sets of vertices:

- $S = \{s_1, s_2, \ldots, s_q\}$,
- $U = \bigcup_{i \in \{1,\ldots,q-1\}} \{u_i^1, u_i^2, u_i^3, \ldots, u_i^{q-i}\}$,
- $P = \{w_1, w_2, \ldots, w_p\}$,

so that $V' = S \cup U \cup P \cup V$. Consider the following sets of edges:

- $E_S = \{(s_1, s_2), (s_2, s_3), \ldots, (s_{q-1}, s_q)\}$,
- $E_U = \bigcup_{i \in \{1,\ldots,q-1\}} \{(s_i, u_i^1), (u_i^1, u_i^2), (u_i^2, u_i^3), \ldots, (u_i^{q-i-1}, u_i^{q-i}), (u_i^{q-i}, a_i)\}$,
- $E_P = \{(s_q, w_1), (w_1, w_2), (w_2, w_3), \ldots, (w_{p-1}, w_p)\}$,

so that $E' = E_S \cup E_U \cup E_P \cup E'$. We define $r = s_1$ and k' as:

$$k' = k + n(q+1) + \frac{q(q^2+1)}{3} + p(q-1) + \frac{p(p+1)}{2} + \frac{q(q-1)}{2}.$$

Let us prove that, if I_{ST} is a YES-Instance then I_{ST-r} is a YES-Instance. Let B be a broadcast that satisfies I_{ST}. Let us define a broadcast B', over I_{ST-r}, and a function $t' : V' \to \mathbb{N}$ (the function of the information times for B'). In B' each vertex s_i, with $i \in \{1, \ldots, q-1\}$, after receiving the information, firstly sends the message to s_{i+1} and, secondly, sends the message to u_i^1. The vertex s_q, after receiving the information, sends it, before, to w_1, and, after, to u_q^1. Vertices in U and in $P \setminus \{w_p\}$, since have degree 2, after receiving the message, coming from one of the two incident edges, forward it towards the other. Thus, a vertex a_i, with $i \in \{1, \ldots, q\}$, receives the message after it has passed through the path $r = s_1, s_2, \ldots, s_i, u_i^1, u_i^2, \ldots, u_i^{q-i}$. The delay collected by the message, while it is passing that path, is 1: only the delay introduced by the vertex s_i (it sends first to s_{i+1}). Thus, we have in B', for all $i \in \{1, 2, \ldots, q\}$:

$$t'(a_i) = length(s_1, s_2, \ldots, s_i, u_i^1, u_i^2, \ldots, u_i^{q-i}, a_i) + delay =$$

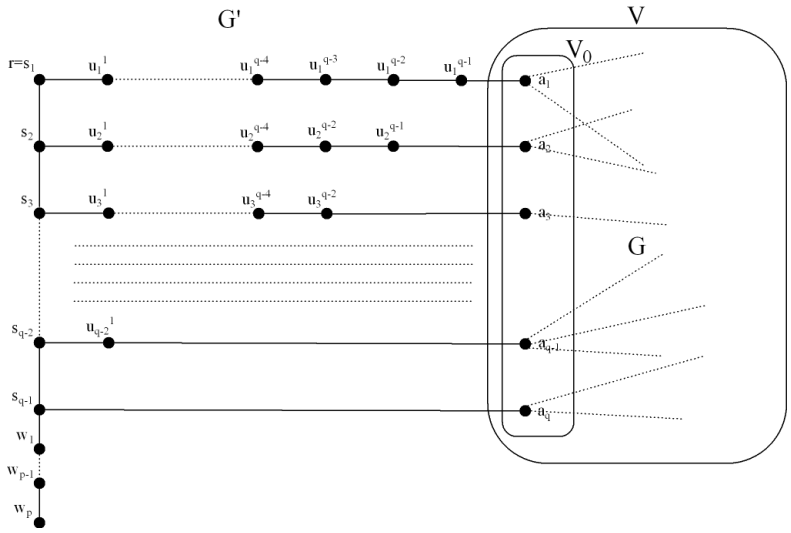

Figure 6.3: The construction of the graph G'.

$$= i + (q - i) + 1 = q + 1.$$

Therefore, all the vertices of V_0 receive the message at the same time in B'. So, we define the rest of the broadcast B', among the vertices in V, as B. Let us count the sum of the information times, in B', of vertices in S:

$$\sum_{i\in\{1,\ldots,q\}} t'(s_i) = \sum_{i\in\{1,\ldots,q\}} (i-1) = \sum_{i\in\{1,\ldots,q\}} i - \sum_{i\in\{1,\ldots,q\}} 1 =$$

$$= \frac{q(q+1)}{2} - q = \frac{q^2+q-2q}{2} = \frac{q(q-1)}{2}.$$

Let us count the sum of the information times, in B', of vertices in U:

$$\sum_{i\in\{1,\ldots,q-1\}} \sum_{j\in\{1,\ldots,q-i\}} t'(u_i^j) = \sum_{i\in\{1,\ldots,q-1\}} \Big((q-i)i + \frac{(q-i)(q-i+1)}{2}\Big) =$$

$$= \sum_{i\in\{1,\ldots,q-1\}} \frac{2qi - 2i^2 + q^2 - qi + q - qi + i^2 - i}{2} = \sum_{i\in\{1,\ldots,q-1\}} \frac{q^2+q-i^2-i}{2} =$$

$$= \frac{1}{2}\Big(\sum_{i\in\{1,\dots,q-1\}} (q^2+q) - \sum_{i\in\{1,\dots,q-1\}} i^2 - \sum_{i\in\{1,\dots,q-1\}} i \Big)$$

$$= \frac{1}{2}\Big((q-1)(q^2+q) - \frac{(q-1)q(2(q-1)+1)}{6} - \frac{q(q-1)}{2} \Big) =$$

$$= \frac{1}{2}\Big(q^3+q^2-q^2-q - \frac{2q^3-q^2-2q^2+q}{6} - \frac{q^2-q}{2} \Big) =$$

$$= \frac{6q^3-6q-2q^3+3q^2-q-3q^2+3q}{12} = \frac{4q^3-4q}{12} = \frac{q(q^2-1)}{3}.$$

Let us count the sum of the information times, in B', of vertices in P, using Lemma 12:

$$\sum_{i\in\{1,\dots,p\}} t'(w_i) = p(q-1) + \frac{p(p+1)}{2}.$$

Since, in B', each vertex of V_0 receives the message at time $q+1$ and the broadcast continues as in B, each vertex in $V \setminus V_0$ will receive the message with the same delay of the vertices of V_0, $q+1$. So, for all $v \in V$, we have that $t'(v) = t(v)+q+1$. Let us count the sum of the information times of broadcast B'.

$$\sum_{v\in V'} t'(v) = \sum_{i\in\{1,\dots,q\}} t'(s_i) + \sum_{i\in\{1,\dots,q-1\}} \sum_{j\in\{1,\dots,q-i\}} t'(u_i^j) + \sum_{i\in\{1,\dots,p\}} t'(w_i) + \sum_{v\in V} t'(v) =$$

$$= \frac{q(q-1)}{2} + \frac{q(q^2-1)}{3} + p(q-1) + \frac{p(p+1)}{2} + \sum_{v\in V}(t(v)+q+1) =$$

$$= \sum_{v\in V} t(v) + \sum_{v\in V}(q+1) + \frac{q(q^2-1)}{3} + p(q-1) + \frac{p(p+1)}{2} + \frac{q(q-1)}{2} =$$

$$= \sum_{v\in V} t(v) + n(q+1) + \frac{q(q^2-1)}{3} + p(q-1) + \frac{p(p+1)}{2} + \frac{q(q-1)}{2} \leq$$

since I_{ST} is a YES-Instance

$$k + n(q+1) + \frac{q(q^2-1)}{3} + p(q-1) + \frac{p(p+1)}{2} + \frac{q(q-1)}{2} = k'.$$

Hence, B' is a broadcast that satisfies I_{ST-r}, so that, if I_{ST} is a YES-Instance then I_{ST-r} is a YES-Instance.

Let us prove that, if I_{ST-r} is a YES-Instance then I_{ST} is a YES-Instance. We, firstly, prove a claim that is necessary to conclude the proof.

CLAIM I: if I_{ST-r} is a YES-Instance then, in any broadcast that satisfies I_{ST-r}, $t(w_1) = q$. *Proof.* Consider a broadcast that satisfies I_{ST-r}. In order to prove CLAIM I, we observe that, since $d(r, w_1) = q$, the message can not arrive in w_1 before q. Instead, consider that the message arrives in w_1 after q, so that $t(w_1) \geq q + 1$. In this broadcast, the sum of all the information times is:

$$\sum_{v \in V'} t'(v) = \sum_{i \in \{1,\ldots,q\}} t'(s_i) + \sum_{i \in \{1,\ldots,q-1\}} \sum_{j \in \{1,\ldots,q-i\}} t'(u_i^j) +$$

$$+ \sum_{i \in \{1,\ldots,p\}} t'(w_i) + \sum_{v \in V} t'(v) \geq \sum_{i \in \{1,\ldots,p\}} t'(w_i) \geq$$

that by Lemma 12 becomes

$$pq + \frac{p(p+1)}{2} = p(q-1) + \frac{p(p+1)}{2} + p >$$

thanks to our choice of p, we can bound the previous as:

$$p(q-1) + \frac{p(p+1)}{2} + k + n(q+1) + \frac{q(q^2+1)}{3} + \frac{q(q-1)}{2} = k'.$$

This is a contradiction (I_{ST-r} is YES-Instance). So, we conclude that if I_{ST-r} is a YES-Instance then, in any broadcast that satisfies I_{ST-r}, $t(w_1) = q$.

Consider a broadcast B' that satisfies I_{ST-r}. By CLAIM I, we know that, if a vertex s_i, with $i \in \{1, \ldots, q-1\}$, receives the information, first it sends the message to s_{i+1}, and, after, it sends the message to u_i^1. In this case, as we have shown formerly, all the vertices in V_0 receives the message at the same time, $q + 1$. Moreover, at time $q + 1$, no one vertex in $V \setminus V_0$ has received the message. Let us define a broadcast B for I_{ST} as a broadcast that transmits the message among vertices of V, in the same way of the transmissions of B' among vertices of V. Let $t : V \to \mathbb{N}$ the function of the information times in B. Since each vertex in V_0 receives the message at time $q+1$, each vertex in V, respect to B', receives the message with a delay of $q + 1$, respect to B. Thus we have that, for all $v \in V$, $t(v) = t'(v) - (q + 1)$. Therefore, the sum of the information times over all the vertices in V, respect to B, is:

$$\sum_{v \in V} t(v) = \sum_{v \in V} (t'(v) - (q+1)) = \sum_{v \in V} t'(v) - n(q+1) =$$

$$= \sum_{v \in V'} t'(v) - \sum_{v \in S} t'(v) - \sum_{v \in U} t'(v) - \sum_{v \in P} t'(v) - n(q+1) =$$

$$= \sum_{v \in V'} t'(v) - \frac{q(q-1)}{2} - \frac{q(q^2-1)}{3} - p(q-1) + \frac{p(p+1)}{2} - n(q+1) \leq$$

since B' is a broadcast that satisfies I_{ST-r}, we can bound the previous as:

$$k' - \frac{q(q-1)}{2} - \frac{q(q^2-1)}{3} - p(q-1) + \frac{p(p+1)}{2} - n(q+1) =$$

$$= k + n(q+1) + \frac{q(q^2+1)}{3} + p(q-1) + \frac{p(p+1)}{2} + \frac{q(q-1)}{2} +$$

$$- \frac{q(q-1)}{2} - \frac{q(q^2-1)}{3} - p(q-1) + \frac{p(p+1)}{2} - n(q+1) = k.$$

Hence, we have that the sum of the information times over all the vertices of V, respect to B, is lesser or equal than k, so that, if I_{ST-r} is a YES-Instance, then I_{ST} is a YES-Instance. □

6.2 Minimum service time in trees

Minimum broadcast time has been studied in case the underlying network is a tree [79, 29]. In this section we extend the analysis on minimum service time problem to trees. We provide an algorithm to solve, in polynomial time, the ST-r problem on trees. Consider a tree $T = (V, E)$, and a root node $r \in V$. Let $v \in V$ be a vertex, we define T_v the subtree of T rooted at v, and $|T_v|$ the number of the vertices belonging to the subtree rooted in v. The algorithm is the following: for each round,

- each vertex v sends the message to the adjacent vertex u, such that u does not know the information and $|T_u|$ is the maximum among the adjacent vertices of v.

In order to prove the correctness of the algorithm, we provide some formal definition.

We say that a broadcast B_T has an *inversion* if there is a node in T having two children u and v such that $|T_u| > |T_v|$ and $t(u) > t(v)$. In this section, we consider the possibility that, in a broadcast process, a vertex can be idle also if it has an adjacent that does not have received the message. Formally, the broadcast B_T has *idle times* if there is a node v having a child u such that

$t(u) > d(v) + t(v)$, where $d(v)$ is the number of children of v respect to T (i.e. there is work to be done, but the node is sitting idle).

Note that the broadcast produced by our algorithm has no inversions and no idle times.

Lemma 13. *All the broadcasts with inversions or idle times are not optimal, over trees.*

Proof. Let B_T be a broadcast with inversions and, the function $t : V \to \mathbb{N}$ the information time function for B_T. We will construct a new broadcast B'_T such that the sum of all the information times of B'_T is lesser or equal than the sum of all the information times in B_T. Consider an inversion in which u is informed before v and $|T_u| < |T_v|$. Consider the new broadcast B'_T in which the father of u and v sends the message first to v and, after, to u. All the other vertices, in B'_T, do the same of B_T. Let $t' : V \to \mathbb{N}$ be the function of the information times in B'_T. Therefore, we have that:

$$\sum_{w \in V} t'(v) = \sum_{w \in V \setminus \{T_v \cup T_u\}} t'(w) + \sum_{w \in T_v} t'(w) + \sum_{w \in T_u} t'(w) =$$

since, for each vertex $w \in V \setminus \{T_v \cup T_u\}$, $t'(w) = t(w)$, we rewrite the previous one as

$$= \sum_{w \in V \setminus \{T_v \cup T_u\}} t(w) + \sum_{w \in T_v} (t'(v) + t'(w) - t'(v)) + \sum_{w \in T_u} (t'(u) + t'(w) - t'(u)) =$$

$$= \sum_{w \in V \setminus \{T_v \cup T_u\}} t(w) + \sum_{w \in T_v} t'(v) + \sum_{w \in T_v} (t'(w) - t'(v)) + \sum_{w \in T_u} t'(u) +$$

$$+ \sum_{w \in T_u} (t'(w) - t'(u)).$$

Each vertex transmits, in B'_T, in the same way as in B_T, except the father of v and u. Consider a vertex w descendant of v. Since the father of v can not be in any path between v and w, the delay between the information time of v and the information time of w, in B'_T, is the same of B_T. The same statement holds for u and its descendants. So we have that:

$$\sum_{w \in T_v} (t'(w) - t'(v)) = \sum_{w \in T_v} (t(w) - t(v))$$

and also

$$\sum_{w\in T_u}(t'(w)-t'(u)) = \sum_{w\in T_u}(t(w)-t(u)).$$

Thus we can write:

$$\sum_{w\in V} t'(v) = \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + |T_v|t'(v) + \sum_{w\in T_v}(t(w)-t(v)) + |T_u|t'(u)+$$

$$+\sum_{w\in T_u}(t(w)-t(u)) =$$

thanks to $t'(v)=t(u)$ and $t'(u)=t(v)$, we have:

$$= \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w)+|T_v|t(u)+\sum_{w\in T_v}(t(w)-t(v))+|T_u|t(v)+\sum_{w\in T_u}(t(w)-t(u)).$$

Since $|T_v|>|T_u|$ and $t(v)>t(u)$, we bound the sum in B'_T as:

$$\sum_{w\in V} t'(v) < \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + \sum_{w\in T_v}(t(w)-t(v)) + \sum_{w\in T_u}(t(w)-t(u))+$$

$$+|T_v|t(u)+|T_u|t(v)+(|T_v|-|T_u|)(t(v)-t(u)) =$$

$$= \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + \sum_{w\in T_v}(t(w)-t(v)) + \sum_{w\in T_u}(t(w)-t(u))+$$

$$+|T_v|t(u)+|T_u|t(v)+|T_v|t(v)-|T_v|t(u)-|T_u|t(v)+|T_u|t(u) =$$

$$= \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + \sum_{w\in T_v}(t(w)-t(v)) + \sum_{w\in T_u}(t(w)-t(u))+$$

$$+|T_v|t(v)+|T_u|t(u) =$$

$$= \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + \sum_{w\in T_v}(t(w)-t(v)) + \sum_{w\in T_u}(t(w)-t(u))+$$

$$+\sum_{w\in T_v} t(v) + \sum_{w\in T_u} t(u) =$$

$$= \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + \sum_{w\in T_v}(t(w)-t(v)+t(v)) + \sum_{w\in T_u}(t(w)-t(u)+t(u)) =$$

$$= \sum_{w\in V\setminus\{T_v\cup T_u\}} t(w) + \sum_{w\in T_v} t(w) + \sum_{w\in T_u} t(w) =$$

$$= \sum_{w \in V} t(v).$$

Hence, we have a broadcast, B'_T, that has the value of the service time lesser than B_T. This contradicts the thesis that B_T has minimum service time. Thus, we conclude that in a broadcast that has minimum service time, there are no inversion.

We also affirm that in a broadcast that has minimum service time, there are no idle times. Consider B_T a broadcast with minimum service time that has an idle time. Let v be the node having a child u such that, in B_T, $t(u) > d(v)+t(v)$. We can construct a broadcast B'_T, and its information time function t', such that $t'(u) \leq d(v) + t'(v)$. We get this solution if v, in B'_T, transmits towards u during the time in which it is sitting idle. Therefore, all the other nodes, in B'_T, can not receive the message later than in B_T. Thus the service time, in B'_T, is lesser than the service time in B_T. This is a contradiction with the thesis that B_T is a broadcast that has minimum service time over T. □

Theorem 8. *Broadcast produced by our algorithm is optimal.*

Proof. Lemma 13 says that the optimal broadcast has no inversions and no idle times. The broadcast produced by our algorithm has no inversions and no idle times. Hence to prove the theorem it is sufficient to show that all the broadcast without inversions and idle times have the same cost. If two different broadcasts have neither inversions nor idle times, then they might not produce exactly the same informing times for all the nodes. They can only differ in the order in which children of some node are informed, if these children are roots of subtrees with the same size. Anyway by an induction on the number of such pseudo-inversions, using the same exchange argument of Lemma 13 we can prove that such broadcast may be converted into an other without modify its cost. □

6.3 Local search algorithms

Until now, we have analyzed special cases in which the Minimum Service Time problem is polynomially solvable. In this section we provide a local search algorithm for the general version of the problem. This heuristic has sparkled out by considering that a broadcast over a network, in the telephone model, is always a spanning tree. In fact, each vertex receives the information from only one predecessor, while it can send, in different rounds, to many other successor vertices. The broadcast tree is always rooted at the message originator. Thus,

we discussed which kind of tree could minimize the service time. We recall the general local search algorithm with the following scheme, where $f(x)$ is the value of the objective function on the solution x.

1. Choose a starting solution x;
2. generate the neighborhood solution set $N(x)$;
3. if there is a solution $y \in N(x)$ such that $f(y) < f(x)$, set $y := x$ and go to Step 2, else STOP.

In our local search algorithm we start from a random spanning tree of the graph and consider the neighborhood solution set as the set of all the spanning trees that differ from the current spanning tree by one edge. Since there are many approaches to examine the solution neighborhood, we have implemented the local search algorithm with two opposite techniques: the *first improvement* and the *steepest descent*. In the first (see Algorithm 2), when we find a better solution in the neighborhood, with respect to the current solution, this becomes the new current solution. In the second version of our local search algorithm (see Algorithm 3), the current solution is updated with the best solution of the neighborhood (if it corresponds to an improvement).

Instances description

In our computer implementation of the algorithms, we have generated several sets of problem instances. Each instance is a random graph, built following the

Algorithm 2 Local Search v.1

Input: G, r;
Output: a broadcast tree;
1: build a random spanning tree T over the instance graph G;
2: let T' be a spanning tree with one swapped edge respect to T;
3: **if** $ST(T') < ST(T)$ **then**
4: set T to T' and go to 2;
5: **else**
6: **if** there are other possible moves **then**
7: go to 2;
8: **else**
9: **return** the current broadcast tree.

Algorithm 3 Local Search v.2

Input: G, r;
Output: a broadcast tree;
1: build a random spanning tree T over the instance graph G;
2: let T' be a spanning tree with one swapped edge respect to T;
3: **if** $\min_{T' \in N(T)} ST(T') < ST(T)$ **then**
4: set T to $\min_{T' \in N(T)} ST(T')$ and go to 2;
5: **else**
6: **if** there are other possible moves **then**
7: go to 2;
8: **else**
9: **return** the current broadcast tree.

Erdős-Rényi model (see [11]), denoted by $G(n, p)$, where n is the number of the vertices of the instance, and p is the probability that every edge may occur independently from the others. We have generated two kinds of instances: *sparse* graphs, in which we fix $p = 0.1$, and *dense* graphs in which $p = 0.4$. Furthermore, we have generated graphs with different sizes. We have built instance sets with 25, 50, 75 and 100 vertices. For each instance set we have built 20 graphs. In Table 6.1 we list all the instance sets, showing, in the column *AVG edges*, the average number of the edges in the instance set.

Table 6.1: Instance sets.

instance set	type	n.vertices	AVG edges
is1	sparse	25	33.80
is2	dense	25	119.95
is3	sparse	50	122.05
is4	dense	50	489.35
is5	sparse	75	277.75
is6	dense	75	1,121.85
is7	sparse	100	497.80
is8	dense	100	1,974.15

Table 6.2: Average objective function (ST over instances).

instance set	Algorithm 2	Algorithm 3
is1	107.35	105.30
is2	94.75	94.00
is3	246.70	245.00
is4	238.55	237.00
is5	401.05	399.95
is6	401.50	398.00
is7	574.30	573.20
is8	577.65	573.00

Computational results

The two local search algorithms have been tested on all the generated instances. Since the algorithms results on a single random generated instance is insignificant for our study, we propose the computational results as aggregate values. For each instance set we show the average value, over the twenty instances, of the results obtained by the algorithms execution. In the same way, we propose the average running time, on each instance set, of the two algorithms. We can see in Table 6.2 the Service Time values of the solutions provided by the two local search algorithms, while in Table 6.3 the running time values are presented.

Table 6.3: Average running time (seconds).

instance set	Algorithm 2	Algorithm 3
is1	0.04	0.05
is2	0.50	0.74
is3	2.40	2.92
is4	21.81	27.86
is5	22.14	24.15
is6	243.89	233.27
is7	115.38	120.79
is8	1,289.13	1,275.89

Performance analysis

In this section we carry out a performance analysis between the two local search algorithms. Evaluating the computational results, we discuss a trade-off between the quality of the obtained solution and the running time. We analyze the cases of sparse graphs and dense graphs separately.

Sparse graphs

The analysis, over sparse graphs, is provided by the comparison of the two objective function values and by the comparison of the two running times. In Figure 6.4 we can see the percentage difference between the results obtained by the two algorithm. We underline that the difference decreases with the size of the instance. This suggests that the steepest descent method of neighborhood analysis does not seem to be a winning choice. But, if we consider Figure 6.5, we note also that the running time of the algorithms, when the number of vertices increases, becomes roughly the same. Therefore, we need to compare the decreasing trends of the to differences to understand which technique is the better. It seems that the percentage difference of the running times is always

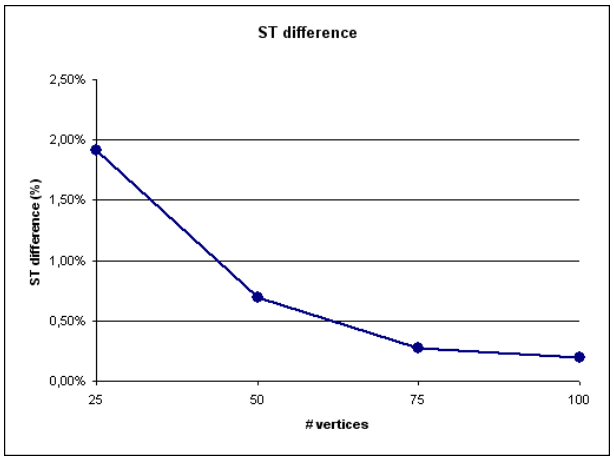

Figure 6.4: The difference between the service times of the two algorithms solutions in sparse graphs.

Figure 6.5: The difference between the running times of the two algorithms in sparse graphs.

greater than the percentage difference of the broadcast service times provided by the algorithms. Hence, we may conclude that it would be better to use the first variant of the local search, in case of sparse instances, because the second technique of neighborhood analysis has larger costs in term of computation time.

Dense graphs

In this section, we analyze the computational results for the case of dense graphs. In Figure 6.6 we can see the percentage difference between the first algorithm with respect to the second one. This chart shows that, in general, the second algorithm outputs better solutions than the first one. On the other hand, in Figure 6.7, the percentage difference between the second algorithm and the first one is depicted. In this case, like in the sparse graphs scenario, with the growth of the instance, the difference approaches to zero. Rather, in the case of graphs with 75 vertices, the steepest descent technique is more efficient than the other one. This is due to the fact that the number of the step needed by the second algorithm to reach the local minimum, is strictly less than that of the first heuristic. Thus, we may conclude that, in case of dense

graphs, the second version of the local search is better than the first version.

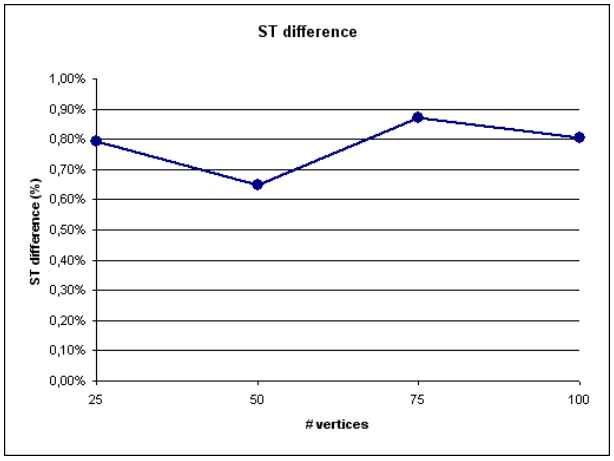

Figure 6.6: The difference between the service times of the two algorithms solutions in dense graphs.

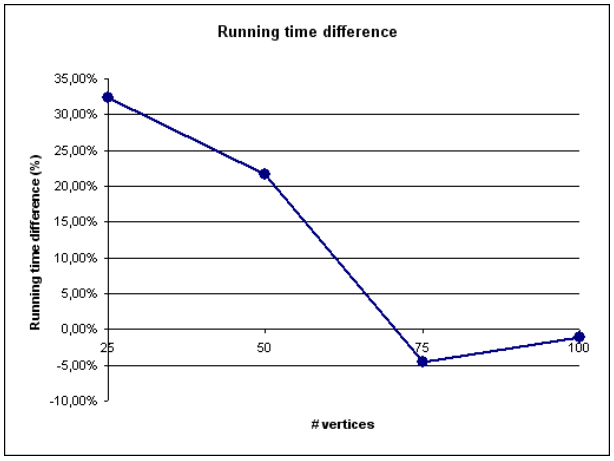

Figure 6.7: The difference between the service times of the two algorithms in dense graphs.

Chapter 7

Broadcast in distributed scenario

In this section we deal with the problem of minimizing the service time, assuming that each peer does not know the complete topology of the communication network. Here, we do not consider a central algorithm that produces a broadcast process over the network, but each peer follows its own communication rules to send the message. We analyze a technique which approximate the solution.

7.1 Jabber Algorithm

Idea. The idea behind the Jabber algorithm is very simple. Every vertex, which knows the information, sends it to a random neighbor which does not knows it.

We consider, in this first version of the algorithm, that, for each time, each vertex knows the information status of its adjacent vertices (i.e. for every adjacent vertex, if it has received the message). The pseudocode of the Jabber algorithm is provided in the table Algorithm 4.

Analysis

Lemma 14. *Let T be a tree rooted in r. The ratio between the solution provided by Algorithm 4 and the optimal solution of the minimum service problem is lesser than $O(\sqrt{n})$.*

Algorithm 4 (Jabber)

Input: $G = (V, E), r \in V$.

```
1: hasInformation(r) := TRUE
2: for w ∈ V \ {r} do
3:    hasInformation(w) := FALSE
4: for each round t do
5:    for v ∈ V do
6:       if hasInformation(v) then
7:          for u ∈ V : u ∈ neighborhood(v) do
8:             if not hasInformation(u) then
9:                hasInformation(u) := TRUE (send information from u to v)
10:               Break for
```

Proof. Consider a tree T. We will prove the lemma by induction on the height of the tree. Let h be the height of T. It is simple to observe that the lemma holds if $h \leq 1$. So that, we consider $h \geq 2$. Let c be the number of the children of r. Let n be the number of the tree. Consider $f_1, f_2, \ldots, f_c$ the children of the root such that, $|T_{f_1}| \geq |T_{f_2}| \geq \cdots \geq |T_{f_c}|$. Let us define $n_i = |T_{f_i}|$, for each $i \in \{1, 2, \ldots, c\}$. We name S^* the sum of the information times in an optimal broadcast and S the sum of the information times in the broadcast provided by the algorithm. In the same way, we denote with S_i^* and S_i the sums of the information times, respectively, in an optimal broadcast and in the broadcast provided by the algorithm, respect to instance T_{f_i}, for each $i \in \{1, 2, \ldots, c\}$. Therefore we have:

$$S^* = \sum_{i=1}^{c} S_i^* + \sum_{i=1}^{c} in_i$$

and

$$S \leq \sum_{i=1}^{c} S_i + \sum_{i=1}^{c} (c - i + 1)n_i = \sum_{i=1}^{c} S_i + (c+1)(n-1) - \sum_{i=1}^{c} in_i.$$

Consider the term $\sum_{i=1}^{c} in_i$. This term is minimized when all the vertices are in T_{f_1} and all the other subtrees $T_{f_2}, T_{f_3}, \ldots, T_{f_c}$ contain only the root, respectively, $f_2, f_3, \ldots, f_c$. In this case we have $n_1 = n - c$ and $n_i = 1$, for all $i \in \{2, 3, \ldots, c\}$. Therefore, we write:

$$\sum_{i=1}^{c} in_i \geq (n - c) + \sum_{i=2}^{c} i = (n-1) + \frac{c(c-1)}{2} = n + \frac{c^2}{2} - \frac{c}{2} - 1.$$

Now, we can rewrite the values S^* and S as follows:

$$S^* \geq \sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1$$

and

$$S \leq \sum_{i=1}^{c} S_i + (c+1)(n-1) - n - \frac{c^2}{2} + \frac{c}{2} + 1.$$

Let us analyze the approximation ratio of the Algorithm 4:

$$\frac{S}{S^*} \leq \frac{\sum_{i=1}^{c} S_i + (c+1)(n-1) - n - \frac{c^2}{2} + \frac{c}{2} + 1}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1} \leq$$

and using the inductive hypothesis

$$\leq \frac{\sum_{i=1}^{c} \sqrt{n_i} S_i^* + (c+1)(n-1) - n - \frac{t^2}{2} + \frac{c}{2} + 1}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1}.$$

Now, we focus our attention to bound the value of $\sqrt{n_i}$. Since, in T, the root has c children, we have that $n_i \leq n - c$, so that:

$$\sqrt{n_i} \leq \sqrt{n-c} \leq \sqrt{\frac{(2n-c)^2}{4n}} = \sqrt{n} - \frac{c}{2\sqrt{n}}.$$

Hence, we have:

$$\frac{S}{S^*} \leq \frac{(\sqrt{n} - \frac{c}{2\sqrt{n}}) \sum_{i=1}^{c} S_i^* + (c+1)(n-1) - n - \frac{c^2}{2} + \frac{c}{2} + 1}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1} \leq$$

$$\leq \frac{\sqrt{n} \sum_{i=1}^{c} S_i^* - \frac{c}{2\sqrt{n}} \sum_{i=1}^{c} S_i^* + cn - \frac{c^2}{2} - \frac{c}{2}}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1}.$$

Since the sum of the solution in the subproblems is:

$$\sum_{i=1}^{c} S_i^* \geq n - c - 1$$

we conclude

$$\frac{S}{S^*} \leq \frac{\sqrt{n} \sum_{i=1}^{c} S_i^* - \frac{c(n-c-1)}{2\sqrt{n}} + cn - \frac{c^2}{2} - \frac{c}{2}}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1} =$$

$$= \sqrt{n} + \frac{\sqrt{n}(-n - \frac{c^2}{2} + \frac{c}{2} + 1) - \frac{c(n-c-1)}{2\sqrt{n}} + cn - \frac{c^2}{2} - \frac{c}{2}}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1} =$$

$$= \sqrt{n} + \frac{-n\sqrt{n} - \frac{\sqrt{n}c^2}{2} + \frac{\sqrt{n}c}{2} + \sqrt{n} - \frac{c\sqrt{n}}{2} + \frac{c^2}{2\sqrt{n}} + \frac{c}{2\sqrt{n}} + cn - \frac{c^2}{2} - \frac{c}{2}}{\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1} =$$

$$= \sqrt{n} + \frac{-2n^2 - nc^2 + nc + 2n - nc + c^2 + c + 2n\sqrt{n}c - \sqrt{n}c^2 - \sqrt{n}c}{2\sqrt{n}(\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1)} =$$

$$= \sqrt{n} + \frac{-(n + \sqrt{n} - 1)c^2 + (2n\sqrt{n} - \sqrt{n} + 1)c - 2n^2 + 2n}{2\sqrt{n}(\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1)}.$$

Consider, in the last formula, the second term. We will prove that it is always negative. It is simple to observe that the denominator is positive. Let us analyze the numerator. We compute the derivative, in c, to find the maximum:

$$N(t, n) = -(n + \sqrt{n} - 1)c^2 + (2n\sqrt{n} - \sqrt{n} + 1)c - 2n^2 + 2n,$$

thus

$$\frac{\partial N(c, n)}{\partial c} = -2c(n + \sqrt{n} - 1) + 2n\sqrt{n} - \sqrt{n} + 1.$$

We put $\frac{\partial N(c,n)}{\partial c} = 0$:

$$-2c(n + \sqrt{n} - 1) + 2n\sqrt{n} - \sqrt{n} + 1 = 0,$$

so, it results

$$c_{max}(n) = \frac{2n\sqrt{n} - \sqrt{n} + 1}{2(n + \sqrt{n} - 1)}.$$

Now we analyze the function $N(c_{max}(n), n)$, considering the maximum respect to c:

$$N(c_{max}(n), n) = -(n + \sqrt{n} - 1)(\frac{2n\sqrt{n} - \sqrt{n} + 1}{2(n + \sqrt{n} - 1)})^2 +$$

$$+(2n\sqrt{n} - \sqrt{n} + 1)(\frac{2n\sqrt{n} - \sqrt{n} + 1}{2(n + \sqrt{n} - 1)}) - 2n^2 + 2n =$$

$$= -\frac{(2n\sqrt{n} - \sqrt{n} + 1)^2}{4(n + \sqrt{n} - 1)} + \frac{(2n\sqrt{n} - \sqrt{n} + 1)^2}{2(n + \sqrt{n} - 1)} - 2n^2 + 2n =$$

$$= \frac{(2n\sqrt{n} - \sqrt{n} + 1)^2}{4(n + \sqrt{n} - 1)} - 2n^2 + 2n =$$

$$= \frac{(2n\sqrt{n} - \sqrt{n} + 1)^2 + (-8n^2 + 8n)(n + \sqrt{n} - 1)}{4(n + \sqrt{n} - 1)} =$$

$$= \frac{-4n^3 - 8n^2\sqrt{n} + 12n^2 + 12n\sqrt{n} - 7n - 2\sqrt{n} + 1}{4(n + \sqrt{n} - 1)} =$$

$$= -\frac{4n^2(n-3) + 8n\sqrt{n}(n - 3/2) + (6n + 2\sqrt{n}) + (n-1)}{4(n + \sqrt{n} - 1)},$$

that, considering our hypothesis $n \geq 2$ is always negative. Therefore, it results that:

$$\frac{S}{S^*} \leq \sqrt{n} + \frac{-(n + \sqrt{n} - 1)c^2 + (2n\sqrt{n} - \sqrt{n} + 1)c - 2n^2 + 2n}{2\sqrt{n}(\sum_{i=1}^{c} S_i^* + n + \frac{c^2}{2} - \frac{c}{2} - 1)} \leq \sqrt{n}.$$

□

Lemma 15. *Let T be a tree rooted in r. The ratio between the solution provided by Algorithm 4 and the optimal solution of the minimum service problem is lesser than $\Delta(G)$, (maximum degree over G).*

Proof. Let $p(v) = (r, u_1, u_2, \ldots, u_h, v)$ be the path from r to v in T. At the beginning of the procedure, the root sends the message to all its adjacent vertices, without idle times. Therefore, the vertices u_1 receives the message until $d(r)$, the degree of r. Thus we have $t(u_1) \leq d(r)$. For the same reason, $t(u_2) \leq t(u_1) + d(u_1) \leq d(r) + d(u_1)$. Hence, it follows that $t(v) \leq d(r) + d(u_1) + d(u_2) + \cdots + d(u_h)$. Now, since every vertex in the graph has degree lesser or equal than $\Delta(G)$, it holds that $t(v) \leq l_v\Delta(G)$, where l_v is the number of the edges in the path $p(v)$. Furthermore, the minimum number of rounds required to transmit the information from r to a vertex v is at least l_v. Let us define $t : V \to \mathbb{N}$ the information time function for the broadcast provided by the algorithm. Let $t^* : V \to \mathbb{N}$ the information time function of a broadcast that minimizes the service time. Putting all together, it results that:

$$\frac{S}{S^*} = \frac{\sum_{v \in V} t(v)}{\sum_{v \in V} t^*(v)} \leq \frac{\sum_{v \in V} l_v \Delta(G)}{\sum_{v \in V} l_v} = \frac{\Delta(G) \sum_{v \in V} l_v}{\sum_{v \in V} l_v} = \Delta(G).$$

□

Conclusions

In this dissertation we have analyzed several communication network issues.

Regarding the study of the survivability networks field, our analysis started from routing protocols that are used in the Ethernet based networks, the SPT protocols (Shortest Path Tree protocols). Since the routing topologies generated by this kind of protocols are trees, structures that are very vulnerable to failures, even a single link failure causes the disconnection of the routing topology. Our aim is to provide, in case of link failures, an auxiliary link which has not been used until now in order to have a connected routing topology at any time. Hence, we asking for networks that have, for each link of the routing topology that may fail, an other link that works as backup link with respect to the former one.

In detail, the routing topologies derived by SPT protocols family are shortest path spanning trees, with respect to links weights set by network operators. We investigated in which cases, given a communication network and a desired routing topology (spanning tree), it is possible to assign links weights so that the routing topology is the unique shortest path tree over the network and, in the event of a single link failure, there is a link that, if swapped with the broken one, brings to a new routing topology that is again the unique shortest path tree (*1-restoration property*). Due to the complexity of 1-restoration property, we provided a further property, the *1-exchange property*, that does not involve the links weights and the network shortest paths. The 1-exchange property is a combinatorial property regarding only the structure of the communication network and the routing topology.

By modeling the communication network as a graph, we have analyzed undirected graphs as well as directed graphs, discussing in which cases there have the 1-restoration property and the 1-exchange property. We showed that, while for undirected graphs the problem is polynomially solvable, to identify if a directed graph has the two above mentioned properties, is a NP-hard problem.

We proved that the problem remains NP-hard even if the instance is a directed graph in which nodes have distances at most equal to 2 from the root. Finally, we also have extended our analysis to the case in which multiple links failures may occur.

Among the possible future directions of research, on one side it is of interest to develop strategies that allow to build networks with the q-exchange property. Furthermore, heuristic algorithms can be proposed to solve the optimization version of the general problem in which we want to minimize the number of arcs without backup.

Regarding the study of communication networks broadcasting, we analyzed a new version of the well known Minimum Broadcast Time problem. Firstly, we formalized in detail the mathematical model underlying the broadcast procedure. We considered the so-called telephone model (also called whispering model), in which, for each communication round, a node can communicate with at most one neighbor.

Differently from the minimum broadcast time problem, in which the goal is to find a broadcast procedure with the minimum duration, in our version of the problem, named the Minimum Service Time, we minimize the average time in which a node receives the message. From the computational complexity point of view, we proved that the Minimum Service Time problem is an NP-hard problem. We also have analyzed some special cases in which the problem becomes solvable in polynomial time. Finally, by considering the possibility that the nodes in a communication network do not know the whole topology of the network, we provided a distributed approximation algorithm for the Minimum Service Time problem and analyzed its performance.

A possible topic for further research is to develop smarter distributed algorithms that guarantee better solution for the general case, as well as for special graph instances.

Bibliography

[1] "IEEE 802.1Q-2003", *IEEE standards for local and metropolitan area networks. Virtual bridged local area networks*, (2003).

[2] A. Al-Rumaih, "A Spare Capacity Planning Methodology for Wide Area Survivable Networks", University of Pittsburgh, Dissertation.com, (1999).

[3] D. Alevras, M. Grötschel, R. Wessäly, "Cost-efficient network synthesis from leased lines", *Annals of Operations Research*, Vol. 76, pp. 1–20, (1998).

[4] K. Altinkemer, "Topological design of ring networks", *Computers and Operations Research*, Vol. 21, pp. 421–431, (1994).

[5] S. Arora, "Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems", *Proceedings 37th Annual Symposium on Foundations of Computer Science*, pp. 2–11, (1996).

[6] A. Atamtürk, D. Rajan, "A directed cycle based column-and-cut generation method for capacitated survivable network design", *Networks*, Vol. 43, pp. 201–211, (2004).

[7] A. Atamtürk, D. Rajan, "A new model for designing survivable networks", *Proceedings of the 10th International Conference on Telecommunication Systems Modeling and Analysis* (ICTSM10), (2002).

[8] A. Atamtürk, D. Rajan, "On splittable and unsplittable flow capacitated network-design arc-set polyhedra", *Mathematical Programming*, Vol. 92, pp. 315–333, (2002).

[9] B. Awerbuch, Y. Azar, A. Blum, S. Vempala, "Improved approximation guarantees for minimum weight k-tree and prize-collecting salesman", *Pro-*

ceedings, 27^{th} *Annual ACM Symposium on Theory of Computing*, pp. 277–283, (1995).

[10] A. Balakrishnan, T. L. Magnanti, J. S. Sokol, Y. Wang, "Spare-capacity assignment for line-restoration using a single-facility type", *Operations Research*, Vol. 50, pp. 617–635, (2002).

[11] B. Bollobás, "Random Graphs", *Studies in Advanced Mathematics*, Cambridge University Press, (2001).

[12] P. Berman, V. Ramaiyer, "Improved Approximations for the Steiner Tree Problem", *J. of Algorithms*, Vol. 17, pp. 381–408, (1994).

[13] M. Bern, P. Plassmann, "The Steiner Tree Problem with Edge Lengths 1 and 2", *Information Processing letters*, Vol. 32, pp. 171–176, (1989).

[14] G. Brightwell, G. Oriolo, F,B. Sheperd, "Reserving Resilient Capacity in a Network", *Siam Journal on Discrete Mathematics*, Vol. 14(4), pp. 524-539, (2001).

[15] S. C. Chau and A. L. Leistman, "Constructing fault-tolerant minimal broadcast networks", J. Combin. Info. and Syst. Sci., Vol. 11, pp. 1–18, (1986).

[16] D. Chen, S. Garg, K. S. Trivedi, "Network survivability performance evaluation: a quantitative approach with applications in wireless ad-hoc networks", In *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, Atlanta, Georgia, USA, pp. 61–68, (2002).

[17] I. Chlamtac, S. Kutten, "On broadcasting in radio networksproblem analysis and protocol design", *IEEE Trans. Commun.*, Vol. 33, pp. 1240–1246, (1985).

[18] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem", *Tech Report 388*, Graduate School of Industrial Administration, Carnegie Mellon University, (1976).

[19] S. H. Chung, H. G. King, Y. S. Yoon, D. W. Tcha, "Cost-minimizing construction of a undirectional SHR with diverse protection", *IEEE Transaction on Networking*, Vol. 4, pp. 921–928, (1996).

[20] A. E. F. Clementi, L. Trevisan, "Improved Non-Approximability Results for Minimum Vertex Cover with Density Constraints", *Electronic Colloquium on Computational Complexity*, TR96-016, (1996).

[21] V. Dalmau, D. K. Ford, "Generalized satisfiability with limited occurrences per variable: a study through delta-matroid parity", *Mathematical Foundations of Computer Science*, (LNCS), Springer, pp. 358–367, (2004).

[22] A. F. de Sousa, G. Soares, "Improving Load Balance and Minimizing Service Disruption on Ethernet Networks using IEEE 802.1S MSTP", *Proc. EuroFGI Workshop on IP QoS and Traffic Control*, Lisbon, Portugal, Vol. 1, pp. 25–35, (2007).

[23] A. Di Salvo, G. Proietti, "Swapping a failing edge of a shortest paths tree by minimizing the average stretch factor", *Theoretical Computer Science*, Vol. 383(1), pp. 23–33, (2007).

[24] M. J. Dinneen, M. R.. Fellows, V. Faber, "Algebraic construction of efficient broadcast networks", *Applied Algebra, Algebraic Algorithms, and Error Correcting Codes*, (LNCS) Vol. 539, pp. 152–158, (1991).

[25] M. Elkin, G. Kortsarz, "Improved broadcast schedule for radio networks", *Proc. 16th ACMSIAM Symp. on Discrete Algorithms* (SODA), pp. 222231, (2005).

[26] A. M. Farley, "Broadcast time in communication networks", *SIAM J. Appl. Math.*, Vol. 39, pp. 385–390, (1980).

[27] A. M. Farley, "Minimum-time line broadcast networks", *Networks*, Vol. 10, pp. 59–70, (1980).

[28] A. M. Farley, S. T. Hedetniemi, "Broadcasting in grid graphs", *Proc. 9th Southeastern Conj. Combin., Graph Theory, Comput.*, pp. 275–288, (1978).

[29] A. M. Farley, A. Proskurowski, "Broadcasting in trees with multiple originators", SIAM. J. Alg. Disc. Meth., Vol. 2, pp. 381-386, (1981).

[30] U. Feige, D. Peleg, P. Raghavan, E. Upfal, "Randomized broadcast in networks", *Proceedings of the 1990 SIGAL Symposium on Algorithms*, (LNCS), Vol. 450, 1990.

[31] P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, P. Widmayer, "Computing all the best swap edges distributively", *Journal of Parallel and Distributed Computing*, Vol. 68(7), (2008).

[32] B. Fortz, M. Thorup, "Internet traffic engineering by optimizing OSPF weights", *Proc. IEEE INFOCOM*, pp. 519–528, (2000).

[33] B. Fortz, M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world, *IEEE Journal on Selected Areas in Communications*, Vol. 20(4), pp.756–767, (2002).

[34] S. Fujita, A. M. Farley, "Sparse Hypercube,a minimal k-line broadcast graph", *Discrete Applied Mathematics*, Vol 127(3), pp. 431–446, (2003).

[35] M. Fürer, B. Raghavachari, "An NC Approximation Algorithm for the Minimum Degree Spanning Tree Problem", *Proceedings of the 28th Annual Allerton Conference on Communication, Control and Computing* pp. 274–281, (1990).

[36] M. Fürer, B. Raghavachari, "Approximating the minimum-degree Steiner tree to within one of optimal", *Journal of Algorithms*, Vol. 17, pp. 409–423. (1994).

[37] E. G. Fusco, A. Pelc, "Acknowledged broadcasting in ad hoc radio networks", *Information Processing Letters*, Vol. 109, pp. 136–141, (2008).

[38] I. Gaber, Y. Mansour, "Centralized broadcast in multihop radio networks", *J. Algorithms*, Vol. 46, pp. 1-20, (2003).

[39] M. R. Garey, D. S. Johnson, "Computers and intractability: a guide to the theory of NP-Completeness", Freeman, (1979).

[40] N. Garg, "A 3-approximation for the minimum tree spanning k vertices", *Proc. 37th Ann. IEEE Symp. on Foundations of Comput. Sci.*, IEEE Computer Society, pp. 302–309., (1996).

[41] L. Garganoand, U. Vaccaro, "On the construction of minimal broadcast networks", *Networks*, Vol. 19, pp. 673–389, (1989).

[42] L. Gasieniec, D. Peleg, Q. Xin, "Faster communication in known topology radio networks", *Proc. 24th ACM Symp. on Principles of Distributed Computing*, (PODC), pp. 129-137, (2005).

[43] O. Goldschmidt, A. Laugier, E. V. Olinick, "SONET/SDH ring assignment with capacity constraints", *Discrete Applied Mathematics*, Vol. 129, pp. 99–128, (2003).

[44] M. Grigni, D. Peleg, "Tight bounds on minimum broadcast networks", *SIAM J. on Disc. Math.*, pp. 207–222, (1991).

[45] M. Grötschel, C.L. Monma, M. Stoer, "Design of survivable networks", *Handbook in OR and MS*, Vol. 7, pp. 617-672, (1995).

[46] S. M. Hedetniemi, S. T. Hedetniemi, A. L. Leistman, "A survey of gossiping and broadcasting in communication networks", *Networks*, Vol. 18, pp. 319–349, (1988).

[47] M. Herzberg, S. J. Bye, A. Utano, "The hop-limit approach for spare capacity assignment in survivable networks", *IEEE/ACM Transaction on Networking*, Vol. 3, pp. 775–784, (1995).

[48] S. Hougardy, H. J. Prömmel, "A 1.598 Approximation Algorithm for the Steiner Problem in Graphs", *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pp. 448–453, (1999).

[49] M. Jorgic, N. Goel, K. Kalaichevan, A. Nayak, I. Stojmenovic, "Localized detection of k-connectivity in wireless ad hoc, actuator and sensor networks", *16th IEEE International Conference on Computer Communications and Networks* (ICCCN), Hawaii, (2007).

[50] M. Karpinski, A. Zelikovsky, "New Approximation Algorithms for the Steiner Tree Problem", *Journal of Combinatorial Optimization*, Vol.1, pp. 47–65, (1997).

[51] S. Khuller , B. Schieber, "Efficient parallel algorithms for testing k-connectivity and finding disjoint s-t paths in graphs", *SIAM Journal on computing*, Vol. 20(2), pp. 352–375, (1991).

[52] J. C. Knight, K. J. Sullivan, "On the definition of survivability", *Technical Report CS-TR-33-00*, University of Virginia, Department of Computer Science, (2000).

[53] A. Kolarov, B. Sengupta, A. Iwata, "Design of Multiple Reverse Spanning Trees in Next Generation of Ethernet-VPNs", *IEEE GLOBECOM04*, Vol. 3, pp. 1390–1395, (2004).

[54] G. Kortsarz, D. Peleg, "Approximation algorithm for minimum time broadcast", *Proceedings of the 1992 Israel Symp. on Theor. Comp. Sci.*, (LNCS), Vol. 601, (1992).

[55] D. Kowalski, A. Pelc, "Optimal deterministic broadcasting in known topology radio networks", *Distrib. Comput.*, Vol. 19, pp. 185-195, (2007).

[56] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", In *Proceedings of the American Mathematical Society*, Vol. 7, No. 1, pp. 48-50, (1956).

[57] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, "The traveling salesman problem", John Wiley, (1985).

[58] A. L. Leistman, "Fault-tolerant broadcast graphs", *Networks*, Vol. 15, pp. 159–171, (1985).

[59] A. L. Leistman, J. G. Peters, "Broadcast networks of bounded degree", *SIAM J. Disc. Math.*, Vol. 1, pp. 531–540, (1988).

[60] V. Marbukh, M. W. Subbarao, "Framework for maximum survivability routing for a MANET", *MILCOM 2000*, Vol. 1, pp. 282–286, (2000).

[61] K. Murakami, H. S. Kim, "Joint optimization of capacity and flow assignment for self-healing ATM networks", *Proceedings of the IEEE International Conference on Communications*, pp. 216–220, (1995).

[62] E. Nardelli, G. Proietti, P. Widmayer, "Swapping a Failing Edge of a Single Source Shortest Paths Tree Is Good and Fast", *Algorithmica*, Vol. 35, pp. 56-74, (2003).

[63] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, C.Diot, "IGP link weight assignment for transient link failures", *Proc. International Teletraffic Congress*, (2003).

[64] P. Orponen, H. Mannila, "On approximation preserving reductions: Complete problems and robust measures", *Technical Report C-1987-28*, Department of Computer Science, University of Helsinki, (1987).

[65] M. Padmaraj, S. Nair, M. Marchetti, G. Chiruvolu, M. Ali, "Traffic Engineering in Enterprise Ethernet with Multiple Spanning Tree Regions", *Proc. of System Communications*, (ICW05), pp. 261–266, (2005).

[66] K. Paul, R. R. Choudhuri, S. Bandyopadhyay, "Survivability analysis of Ad Hoc wireless network architecture", *Lecture Notes in Computer Science*, Springer, Vol. 1818, (2000).

[67] M. D. Penrose, "On k-connectivity for a geometric random graph.", *Random Structures and Algorithms*, Vol. 15(2), pp. 145–164, (1999).

[68] R. C. Prim, "Shortest connection networks and some generalizations", In *Bell System Technical Journal*, Vol. 36, pp. 1389-1401, (1957).

[69] H. J. Prömmel, A. Steger, "RNC-Approximation Algorithms for the Steiner Problem", *Proceedings 14th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 559–570, (1997).

[70] D. Rajan, A. Atamtürk, "Survivable network design : Routing of flows and slacks", *Chapter in Telecommunications Network Design and Management*, Kluwer Academic Publishers, pp. 65–81, (2002).

[71] S. Ramaswamy, T. Dahlberg, "PCS Network Survivability", *Proceedings of IEEE Wireless Communications and Networking Conference* (WCNC'99), pp. 1028–1032, (1999).

[72] R. Ravi, "Rapid rumor ramification: approximating the minimum broadcast time", *SFCS '94: Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pp. 202–213, (1994).

[73] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, S. S. Ravi, "Spanning trees short or small", *SIAM Journal on Discrete Mathematics*, pp. 546–555, (1994).

[74] D. Richards, A. L. Leistman, "Generalizations of broadcasting and gossiping", *Networks*, Vol. 18, pp. 125-138, (1988).

[75] G. Robins, A. Zelikovsky, "Improved Steiner Tree Approximation in Graphs", *Symposium on Discrete Algorithms*, pp. 770–779, (2000).

[76] H. Sakuachi, YNishimura, S. Hasegawa, "A self-healing network with an economical spare-channel assignment", *Proceedings of IEEE GLOBECOM 1990*, pp. 438–443, (1990).

[77] P. Scheuermann, G. Wu, "Heuristic algorithms for broadcasting in point-to-point computer networks", *IEEE Trans. on Computers*, Vol. 33, pp. 804–811, (1984).

[78] S. Sharma, K. Gopalan, S. Nanda, T. Chiueh, "Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks", *IEEE INFOCOM04*, Vol. 4, pp. 2283–2294, (2004).

[79] P. J. Slater, E. J. Cockayne, S. T. Hedetniemi, "Information dissemination in trees", SIAM *J. on Computing*, Vol. 10, pp. 692–701, (1981).

[80] A. P. Snow, U. Varshney, A. D. Malloy, "Reliability and survivability of wireless and mobile networks", *IEEE Computer*, Vol. 33(7), pp. 49–54, (2000).

[81] P. Soriano, C. Wynants, R. Séguin, M. Labbé, M. Gendrau, B. Fortz, "Design and dimensioning of survivable SDH/SONET networks", *Telecommunications Network Planning*, editors B. Sansò, P. Soriano, pp. 147–168, Kluwer Academic Publisher, Netherlands, (1998).

[82] H. Takahashi, A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs", *Math. Jap.* Vol. 24 , pp. 573–577, (1980).

[83] R. Tarjan, "Depth-first search and linear graph algorithms", *SIAM Journal on Computing*, Vol. 1(2), pp. 146–160, (1972).

[84] D. Tipper, T. Dahlberg, H. Shin, C. Charnsripinyo, "Providing fault tolerance in wireless access networks", *IEEE Communications Magazine*, Vol. 40(1), pp. 58–64, (2002).

[85] J. A. Ventura, X. Weng, "A new method for constructing minimal broadcast networks", *Networks*, Vol. 23, pp. 481–497, (1993).

[86] Y. H. Wang, W. S. Soh, M. Y. Tsai, H. S. Kim. "Survivable wireless ATM network architecture", *Ninth International Conference on Computer Communications and Networks*, pp. 368–373, (2000).

[87] Y. Wang, Z. Wang, L. Zhang, "Internet Traffic Engineering without Full Mesh Overlaying", *Proc. IEEE INFOCOM*, (2001).

[88] L. Weifa, B. D. McKay, "Fast parallel algorithms for testing k-connectivity of directed and undirected graphs", *IEEE First International Conference on Algorithms and Architectures for Parallel Processing*, (ICAPP 95), Vol. 1(19-21), pp. 437–441, (1995).

[89] D. B. West, "A class of solutions to the gossip problem, Part I", *Disc. Math.*, Vol. 39, pp. 307-326, (1982).

[90] Y. Xiong, L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks", *IEEE/ACM Transaction on Networking*, Vol. 7, pp. 98–110, (1999).

[91] A. Zelikovsky, "An 11/6-Approximation Algorithm for the Network Steiner Problem", *Algorithmica*, Vol. 9, pp. 463–470, (1993).

[92] A. Zelikovsky, "Better Approximation Bounds for the Network and Euclidean Steiner Tree Problems", *Technical report CS-96-06*, University of Virginia, (1996).