Universidade Federal da Bahia
Escola Politécnica / Instituto de Matemática

Programa de Pós-Graduação em Mecatrônica

ALGORITHMS FOR THE DIRECTED K-SPANNER WITH MINIMUM DEGREE STEINER TREE PROBLEM

Hugo Vinicius Vaz Braga

DISSERTAÇÃO DE MESTRADO

Salvador
2012

HUGO VINICIUS VAZ BRAGA

ALGORITHMS FOR THE DIRECTED K-SPANNER WITH MINIMUM DEGREE STEINER TREE PROBLEM

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Mecatrônica.

Orientador: Prof. Dr. Flávio Morais de Assis Silva

Salvador
2012

Sistema de Bibliotecas - UFBA

Braga, Hugo Vinicius Vaz Braga.

Algorithms for the Directed k-Spanner with Minimum Degree Steiner Tree Problem / Hugo Vinicius Vaz Braga – Salvador, 2012.

45p.: il.

Orientador: Prof. Dr. Prof. Dr. Flávio Morais de Assis Silva.

Dissertação (Mestrado) – Universidade Federal da Bahia, Escola Politécnica / Instituto de Matemática, 2012.

1. Árvores (Teoria dos grafos). 2. Algoritmos computacionais. 3. Otimização combinatória. 4. Heurística.. I. Silva, Flávio Morais de Assis. II. Universidade Federal da Bahia. Escola Politécnica / Instituto de Matemática. III Título.

CDD – 511.5

CDU – 519.17

TERMO DE APROVAÇÃO

HUGO VINICIUS VAZ BRAGA

ALGORITHMS FOR THE DIRECTED K-SPANNER WITH MINIMUM DEGREE STEINER TREE PROBLEM

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Mecatrônica e aprovada em sua forma final pelo Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia.

Salvador, 22 de Outubro de 2012

Prof. Dr. Flávio Morais de Assis Silva (Orientador)
Departamento de Ciência da Computação/UFBA

Prof. Dr. George Marconi de Araújo Lima
(Examinador PPGM)
Departamento de Ciência da Computação/UFBA

Prof. Dr. Karcius Day Rosario Assis (Examinador Externo)
Departamento de Engenharia Elétrica /UFBA

A meu av Clovis - o Major (In Memoriam)

ACKNOWLEDGEMENTS

São muitos a quem eu tenho que agradecer, não só pela elaboração deste trabalho mas também pelo suporte fornecido ao longo da execução do mesmo. Talvez não consiga mencionar todas as pessoas que foram importantes mas me sinto na obrigação de registrar meus agradecimentos para alguns em especial. A ordem não necessariamente representa a prioridade nos agradecimentos.

Gostaria de agradecer à minha família, em especial aos meus pais Geovani e Iris e aos meus irmãos Júnior, Clovis e Marcelle, pelo amor, apoio e principalmente incentivo na busca dos meus objetivos.

Agradeço também ao meu orientador, o Prof. Dr. Flávio Morais de Assis Silva, pelas muitas horas de dedicação e orientação não só na elaboração deste trabalho como também na formação, espero eu, de um futuro pesquisador. Agradeço Flávio pela confiança no meu trabalho e também pela liberdade concedida para realizar minha pesquisa. Agradeço também pelos momentos de descontração, principalmente nas conversas cujo tema era música.

Agradeço à Capes pelo financiamento deste trabalho. Graças ao suporte da Capes, pude dedicar meus esforços de forma única e exclusiva à elaboração deste trabalho.

Agradeço à minha avó Eufélia (pé de ...) pelo carinho e principalmente pelos momentos hilários que passei com a mesma, mesmo que muitas vezes sendo através do telefone. Agradeço à família Braga em geral, incluindo a Velharia.

Agradeço ao Lasid, em especial ao subgrupo Redesens pelas discussões proporcionadas. Um agradecimento especial aos colegas e amigos Fred, Carol e Bruno, com os quais pude compartilhar não apenas momentos de cunho técnico mas também momentos de lazer, dentre eles a ida ao *Rock in Rio IV*.

Agradeço aos professores da Mecatrônica pelos ensinamentos. Agradeço também aos colegas (novos e antigos) de computação, tanto do mestrado em Mecatrônica como do doutorado em Computação, pelas discussões. Agradecimento especial aos colegas e amigos Zé, Marcos, Waltemir, Isaque e Ruth.

Agradeço à Liara e Aida pelo suporte. As mesmas sabem das suas parcelas de contribuição para o andamento deste trabalho.

Agradeço aos amigos (alguns já mencionados anteriormente), dentre eles Daniele e Paula, pelos momentos felizes e pela companhia agradável.

Aos que eu não citei explicitamente mas que, de alguma forma, foram importantes ao longo da elaboração deste trabalho, também deixo meus sinceros agradecimentos.

The formulation of a problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill.

—ALBERT EINSTEIN (NOTA)

RESUMO

Árvores *Steiner* são comumente utilizadas para modelar restrições na execução da operação de *multicast*. Nesta dissertação nós tratamos um novo problema denominado *Árvore Steiner com Grau Mínimo e fator de dilatação k em Grafos Direcionados* (cujo acrônimo em inglês é DSMDStP). Este problema consiste em: dado um grafo direcionado $G(V, E)$, um *nó origem* $s \in V$, um *fator de dilatação* k ($k \in \mathbb{R}^+$, $k \geq 1$) e um conjunto de *terminais* $T \subseteq V \setminus \{s\}$, encontrar uma arborescência onde o custo entre o nó de origem s em G e cada $t \in T$ é menor ou igual a k vezes o custo da menor distância entre este par de nós, ao passo que o grau máximo de saída é minimizado. DSMDStP não admite aproximação sublogarítmica (a menos que $NP \subset DTIME(n^{\log \log n})$). Nós descrevemos um algoritmo de aproximação que gera uma arborescência com grau máximo de saída limitado por $2\sqrt{|T|} + 2 + O(\log |T|) \cdot d^*$, onde d^* consiste no grau máximo da solução ótima e a arborescência é uma *spanner* com fator de dilatação (a partir da raiz) de $k \cdot \left(1 + \frac{max_{t \in T}\{dist(s,t,G)\}}{min_{t \in T}\{dist(s,t,G)\}}\right)$, onde $dist(s, t, G)$ representa o caminho de menor custo entre s e t em G. Embora nosso fator de dilatação viole k, nos experimentos, a restrição de *spanner* foi satisfeita ou, em média, quase satisfeita. Além disso, o grau de saída medido nos experimentos foi baixo. Nós também descrevemos uma heurística que garante um fator de dilatação de $k \cdot (\lfloor\sqrt{|T|}\rfloor + 2)$, mas não limita o grau máximo de saída. Nos experimentos, a heurística mostrou-se extensível com relação ao grau máximo, além de sempre superar os outros algoritmos nesta métrica. A heurística gerou, adicionalmente, uma *spanner* com fator de violação baixo.

Palavras-chave: Árvore *Steiner*, Grafo Direcionado, Grau Mínimo, *Spanner*, Solução de Aproximação, Heurística

ABSTRACT

Steiner trees are commonly used to model constraints in message multicasting. In this dissertation we address a problem called *Directed k-Spanner with Minimum Degree Steiner Tree Problem* (DSMDStP). This problem consists in, given a directed weighted graph $G(V, E)$, a *source node* $s \in V$, a *stretch factor* k ($k \in \mathbb{R}^+$, $k \geq 1$) and a set of *terminals* $T \subseteq V \setminus \{s\}$, finding an arborescence in which the cost (distance) between s and each $t \in T$ is less than or equal to k times the shortest distance between s and t in the original graph, while minimizing the maximum node out-degree. DSMDStP is not approximable sublogarithmically (unless $NP \subset DTIME(n^{\log \log n})$). We describe an approximation algorithm that generates an arborescence with out-degree limited to $2\sqrt{|T|} + 2 + O(\log |T|) \cdot d^*$, where d^* is the maximum degree in an optimum solution and the arborescence is a spanner with a root-stretch factor of $k \cdot \left(1 + \frac{max_{t \in T}\{dist(s,t,G)\}}{min_{t \in T}\{dist(s,t,G)\}}\right)$, where $dist(s, t, G)$ represents the shortest distance between s and t in G. Although our root-stretch factor violates k, in our experiments the spanner constraint was satisfied or almost satisfied in average. Additionally, the resulted out-degree was low. We also describe a heuristic which provides a root-stretch factor of $k \cdot (\lfloor\sqrt{|T|}\rfloor + 2)$ but does not provide a bound to the out-degree. In the experiments, the heuristic was shown to scale well in terms of the maximum degree achieved, and has always outperformed the other algorithms. The heuristic generated additionally a low spanner violation factor.

Keywords: Steiner tree, Directed Graph, Minimum Degree, Spanner, Approximation, Heuristic

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Given a graph $G = (V, E)$, with edge weight function $C : E \rightarrow \mathbb{R}^+$, a source node $s \in V$ and a set of *terminals* $T \subseteq V \setminus \{s\}$, the classical minimum cost Steiner tree problem consists in finding a tree $T_r = (V_{T_r}, E_{T_r})$ rooted at s that spans T and that has minimum cost (i.e. minimizes $\sum_{e \in E_{T_r}} C(e)$). Many variations of this problem have been defined, where additional constraints must be satisfied (e.g., (OLIVEIRA; PARDALOS, 2005; PARSA; ZHU; GARCIA-LUNA-ACEVES, 1998; WANG; LIU; SHI, 2009)).

In the context of communication networks, the Steiner Tree problem is frequently related to multicasting, where a message has to be sent to a subset of the nodes. Constraints are added to the problem, for example, to address quality-of-service requirements. In real-time applications, maximum transmission delays for a message sent from s to each node in T must have to be satisfied (OLIVEIRA; PARDALOS, 2005; NGUYEN; NGUYEN, 2008). Maximum delays can be modeled by defining maximum costs for paths from s to each node in T. Alternatively, restrictions in communication delays can be defined as a limit on the increase of the distance of a pair of nodes on the tree when compared to the distance of the same pair of nodes in the original graph, i.e., by imposing a *stretch factor*. This is commonly modeled as a spanner property.

An additional common constraint is a bounded node degree. Limiting node degree is of practical importance, for example, to address the number of connections that can be managed by communication devices such as routers and switches or by communication protocols (as in Bluetooth), to minimize the effort of replicating data in multicast operations (OLIVEIRA; PARDALOS, 2005; NGUYEN; NGUYEN, 2008), or to decrease topology maintenance in mobile networks. In the context of wireless sensor networks, minimizing node degree is particularly important, due to the need for keeping resource use at a minimum (as sensor nodes are typically scarce-resource devices).

In order to address degree minimization and a spanner property between the source node and terminals, we define in this dissertation a new problem called *Directed k-Spanner with Minimum Degree Steiner Tree Problem* (DSMDStP). Instead of minimizing the total cost of the tree, we are interested in minimizing the maximum out-degree of nodes. Additionally, we aim at generating a tree such that, given a stretch factor k, the distance

between the source node and a terminal on this tree is at most k times the minimum distance between them in the original graph. We call Steiner trees with this kind of parameterized spanner property *single-sink k-spanner Steiner trees* (although we are not minimizing the tree cost, the nomenclature *Steiner problem* is consistent, for example, with (FRAIGNIAUD, 2001)).

DSMDStP is not approximable within $(1-\varepsilon)\log_e n$, for any $0 < \varepsilon < 1$ (unless $NP \subset DTIME(n^{\log\log n})$). We describe first an approximation algorithm for DSMDStP based on the algorithm presented in (ELKIN; KORTSARZ, 2006). Our algorithm provides a guarantee on the maximum out-degree of nodes and gives a stretch factor of $k \cdot \left(1 + \frac{max_{t\in T}\{dist(s,t,G)\}}{min_{t\in T}\{dist(s,t,G)\}}\right)$, where $dist(u, v, G)$ represents the cost of a minimum cost directed path from u to v in G. Although our algorithm does not guarantee the desired stretch factor of k, in our experiments the cost of the generated paths is much lower than the provided upper bound. In fact, on average, the spanner constraint was satisfied on half of the scenarios and almost satisfied on the other half.

We also describe a heuristic for the problem, called *Sliced and Iterative Multiple Set-Cover* (SIM). The heuristic does not provide guarantee on node degree and provides a root-stretch factor of $k \cdot (\lfloor\sqrt{|T|}\rfloor + 2)$. In the experiments, SIM exhibited quite good results. Its curve is uniform for maximum out-degree, which is an important requirement for scalability. Moreover, it always outperformed the approximation algorithm in relation to maximum node degree. Regarding the violation of spanner constraint, SIM was outperformed by the approximation algorithm but the violation occurred by a low factor (in average by 1.4 and by 2 in the worst case).

This dissertation is organized as follows. In Chapter 2 we formally define DSMDStP and show that it is not approximable sublogarithmically (unless $NP \subset DTIME(n^{\log\log n})$). In Chapter 3 we describe the proposed algorithms for DSMDStP. In Chapter 4 we describe the results of experiments performed to evaluate the algorithms. In Chapter 5 we discuss related work. Finally, Chapter 6 concludes the dissertation. In Appendix A we comment how to approach DSMDStP through an alternative way, which comprises the concepts of submodular function and matroids.

In the next chapter we define DSMDStP as well as to prove that it is not approximable sublogarithmically.

Chapter 2

PROBLEM DEFINITION

In this chapter our main goal is to state the problem addressed in this dissertation. We introduce some basic definitions necessary to understand the proof concerning the hardness of approximation of the problem, as well as the notion of approximation used throughout the dissertation. We formally state our problem and also emphasize the main differences between it and previous definitions of similar problems. Finally, we prove our problem is not approximable sublogarithmically.

2.1 BASIC DEFINITIONS

As we will prove later, DSMDStP is not approximable sublogarithmically (unless $NP \subset DTIME(n^{\log \log n})$), which means that not only an optimal solution cannot be found polynomially but there is also a lower bound for the approximation factor of the solution that can be polynomially computable (in our case, the lower factor is logarithmic). So, we will address DSMDStP through an approximation algorithm, which can be formally defined as:

Definition 2.1. (WILLIAMSON; SHMOYS, 2011) An α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution.

α is known as *approximation factor* or *approximation ratio*. Since in DSMDStP we aim to minimize the degree, and considering an optimum solution to DSMDStP has degree Δ^*, an α-approximation algorithm for DSMDStP generates an arborescence whose maximum out-degree is $\leq \alpha \cdot \Delta^*$. Throughout the dissertation we use the expression *approximation algorithm* instead of α-approximation algorithm when is not necessary to mention the approximation factor.

Another important definition is the complexity class *DTIME*. This class is mentioned in our result of the hardness of approximation of DSMDStP. For the following concept of the class *DTIME*, the notions of *Turing machine* and *language* are used. A Turing machine decides a language $L \subseteq \{0,1\}^*$ if it computes the function $f_L : \{0,1\}^* \to \{0,1\}$ where $f_L(x) = 1 \Leftrightarrow x \in L$.

Definition 2.2. (ARORA; BARAK, 2009) The class **DTIME**: Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in **DTIME**$(T(n))$ iff there is a deterministic Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L.

2.2 OUR PROBLEM: DSMDSTP

Let $G(V, E)$ be a directed weighted graph with edge weight function $C : E \to \mathbb{R}^+$. Let P=$\{v_1, v_2, ..., v_m\}$ be a directed path in G, where $v_i \in V$ and $(v_i, v_{i+1}) \in E$, $1 \leq i < m$. We denote $sp(v_i, v_j, G)$ a minimum cost directed path from v_i to v_j in G and $dist(v_i, v_j, G)$ the cost of this path, i.e. the sum of the costs of its edges (for simplicity, we assume that $dist(v_i, v_j, G) = \infty$ if there is no path from v_i to v_j in G). Let $odeg(v, G), v \in V$, be the out-degree of node v in G.

The Directed k-Spanner with Minimum Degree Steiner Tree Problem (DSMDStP) is defined as follows: given a directed weighted graph $G = (V, E)$, a *source node* $s \in V$, a *stretch factor* k ($k \in \mathbb{R}^+$, $k \geq 1$) and a set of *terminals* $T \subseteq V \setminus \{s\}$, find an arborescence $A = (V_A, E_A)$, $V_A \subseteq V$ and $E_A \subseteq E$, rooted at s that spans T such that:

- $dist(s, t, A) \leq k \cdot dist(s, t, G)$, $\forall t \in T$; and
- the maximum out-degree of the nodes in A, denoted $D_{max}(A) = max_{v \in V_A}\{odeg(v, A)\}$, is minimized.

The k value is known as a *root-stretch factor*, as we are interested in guaranteeing the spanner property only from the source node to the terminals (instead of between any pair of nodes in the tree, as in (FOMIN; GOLOVACH; LEEUWEN, 2011)).

The proof of the hardness of approximation of DSMDStP is based on a reduction from the Steiner version of the MDST problem in directed graphs (abbreviated for SvMDST), presented in (FRAIGNIAUD, 2001), where this problem is similar to DSMDStP without the spanner constraint. Formally, the instance of SvMDST problem is formed by a directed graph $G = (V, E)$, a set $T \subseteq V$ and a source node $s \in T$ and the objective is finding an arborescence rooted at s that spans T (actually, $T \setminus \{s\}$) and has maximum out-degree minimized. Observe that in SvMDST, the source node $s \in T$, where in DSMDStP $s \notin T$. We take this into consideration in the reduction. The following theorem, which appeared in (FRAIGNIAUD, 2001), is used in our proof:

Theorem 2.1. *(FRAIGNIAUD, 2001). Unless $NP \subset DTIME(n^{\log \log n})$, the optimal solution of the Steiner version of the MDST problem in directed graphs is not approximable in polynomial time within $(1 - \varepsilon) \log_e |T|$ for any $0 < \varepsilon < 1$.*

Claim 2.1. *Directed k-Spanner with Minimum Degree Steiner Tree Problem (DSMDStP) is not approximable in polynomial time within $(1 - \varepsilon) \log_e |T|$, for any $0 < \varepsilon < 1$, unless $NP \subset DTIME(n^{\log \log n})$.*

Proof: Let $\mathcal{S}$ be an instance of SvMDST, defined by a directed graph $G = (V, E)$, a set $T \subseteq V$ and a source node $s \in T$. Let $\Delta^*_{\mathcal{S}}$ be an optimal solution for $\mathcal{S}$. We now create an instance $\mathcal{D}$ of DSMDStP, by using the same graph G, the same source node s, a set

of terminals $T_{\mathcal{D}} = T \setminus \{s\}$ and an integer parameter $k = \infty$ (for example, k might be defined as $\frac{\sum_{e \in E} C(e)}{min_{t \in T}\{dist(s,t,G)\}}$). Let $\Delta^*_{\mathcal{D}}$ be an optimal solution for $\mathcal{D}$.

For this value of k, whatever the solution to $\mathcal{S}$, it satisfies the spanner constraint. So, in this situation, $\Delta^*_{\mathcal{S}} = \Delta^*_{\mathcal{D}}$. Let $\mathcal{A}$ be an α-approximation algorithm for DSMDStP. Let Δ^* be the resulted degree by applying $\mathcal{A}$ to $\mathcal{D}$. We have $\Delta^* \leq \alpha \cdot \Delta^*_{\mathcal{D}}$, which implies that:

$$\frac{\Delta^*}{\Delta^*_{\mathcal{S}}} \leq \alpha. \tag{2.1}$$

This means that we can approximate an optimal solution for SvMDST ($\Delta^*_{\mathcal{S}}$) within α. Based on Theorem 2.1, we know the approximation ratio of $\Delta^*_{\mathcal{S}}$ is $> (1 - \varepsilon) \log_e |T|$. So, $(1 - \varepsilon) \log_e |T| < \frac{\Delta^*}{\Delta^*_{\mathcal{S}}} \leq \alpha$, which implies that:

$$\alpha > (1 - \varepsilon) \log_e |T|, \tag{2.2}$$

concluding the proof. ∎

In the next chapter we present two algorithms to DSMDStP: an approximation algorithm and a heuristic. For each of them, we describe the algorithm as well as giving and proving its properties. For the approximation algorithm, we give an upper bound on the maximum out-degree and an upper bound on the final costs of the arborescence's paths. For the heuristic, we give an upper bound on the final costs of the arborescence's paths. We are also going to present the complexity of each solution.

Chapter 3

PROPOSED ALGORITHMS FOR DSMDSTP

In this chapter we describe an approximation algorithm and a heuristic for DSMDStP. For the approximation algorithm, we give an upper bound for the maximum out-degree. As mentioned before, we were not able to guarantee the given spanning factor. For both solutions, we give an upper bound to the final costs of the arborescence's paths. For each solution, we give detailed information about the complexity. We also describe how the Multiple Set-Cover problem is solved, since it is an essential part of our algorithms. But the reader can skip this last section without impairing the reading.

3.1 AN APPROXIMATION ALGORITHM FOR DSMDSTP

Our algorithm is based on the algorithm presented in (ELKIN; KORTSARZ, 2006) (*Directed Telephone Multicast Problem*). We modified it to address the spanner property.

The algorithm has two phases. In the first phase the algorithm computes a so-called $\sqrt{l}$-partition. In this phase, some of the terminals are covered. In the second phase, an instance of the Multiple Set-Cover problem (defined in (ELKIN; KORTSARZ, 2006)) is used to determine paths to cover the remaining terminals.

Preliminaries

The algorithm has a directed graph $G = (V, E)$ as input. The set V is divided into two disjoint sets: C, the set of *covered* nodes, and U, the set of *uncovered* nodes. Let $UT = U \cap T$ (the set of uncovered terminals) and $CT = C \cap T$ (the set of covered terminals).

Let $l = |T|$ and d^* be the maximum out-degree of an optimum solution for an instance of DSMDStP ($d^* \leq l$). The *out-neighbourhood* of u in G is denoted $N(u, G)$, i.e. $N(u, G) = \{v : (u, v) \in E\}$. For a set of nodes S, $N(S, G)$ is the union of the sets $N(u, G), \forall u \in S$.

Let us denote $G(S)$ the graph induced by a set of nodes S. The graph induced by U is thus denoted $G(U)$. The Δ-*neighbourhood* of a node $u \in U$, denoted Δ-neigh(u), is the node set

$$\Delta\text{-neigh}(u) = \{t : t \in UT \wedge dist(s, u, G) + dist(u, t, G(U)) \leq k \cdot dist(s, t, G)\}.$$

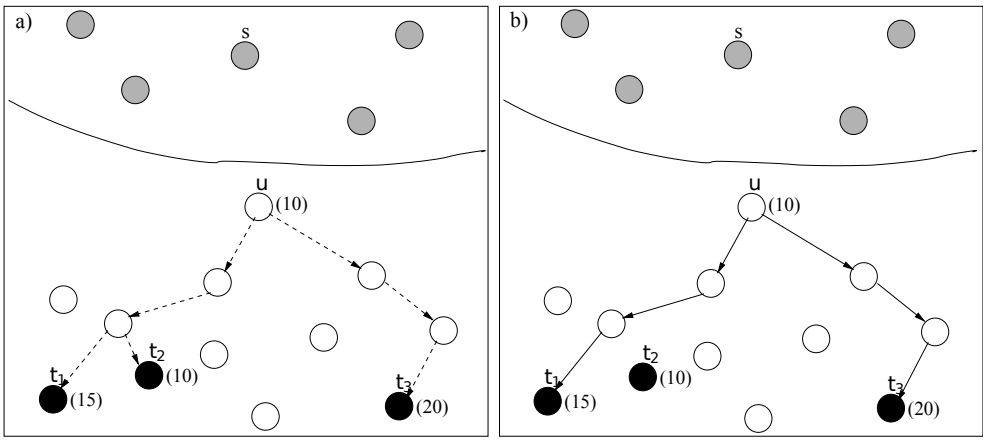

Figure 3.1 Δ-neighbourhood of node u.

Figure 3.1 illustrates the concept of Δ-neighbourhood. The gray nodes represent covered nodes (which are above the division) and the others are uncovered nodes (including the black ones which represent terminals in UT). Consider $k = 2$ and the quantity plotted in the right side of a node v as equal to $dist(s, v, G)$ and consider $dist(u, t_1, G(U)) = 15$, $dist(u, t_2, G(U)) = 12$, $dist(u, t_3, G(U)) = 25$, The dashed paths in Figure 3.1a represent the shortest paths from u to reachable terminals (t_1, t_2, t_3). In Figure 3.1b, the solid paths represent the paths to terminals only that are in Δ-neigh(u), which are t_1 and t_3.

We denote $SPT(s, S_{leaves}, G)$ a tree rooted at s composed of shortest paths in graph G from s to each of the nodes in S_{leaves}. Finally, for a graph G or a path P we denote $V(G)$ and $E(G)$ as well as $V(P)$ and $E(P)$ the sets of nodes and edges, respectively, of the graph and of the path.

3.1.1 First Phase: Computing a $\sqrt{l}$-Partition

In the first phase, the algorithm computes a $\sqrt{l}$-*partition*. A $\sqrt{l}$-*partition* divides V into the disjoint and non-empty sets C and U, $V = C \cup U$, $C \cap U = \emptyset$, $s \in C$, such that the Δ-neighbourhood of any node in U contains at most $\sqrt{l}$ terminals. A $\sqrt{l}$-partition is computed by eliminating $\sqrt{l}$-*bad* nodes. A node u is called a $\sqrt{l}$-*bad* node if u contains more than $\sqrt{l}$ terminals in its Δ-neighbourhood. Procedure `CompPar` (from (ELKIN; KORTSARZ, 2006)) computes a $\sqrt{l}$-partition.

Figure 3.2 illustrates the application of procedure `CompPar`. Gray nodes represent covered nodes and the others are uncovered nodes. Consider $k = 2$ and the quantity plotted in the right side of a node z as equal to $dist(s, z, G)$ and consider $d(u, t_4, G(u)) = 5$, $d(u, t_3, G(u)) = 10$, $d(u, t_1, G(u)) = 11$, $d(u, t_2, G(u)) = 13$, $d(u, t_6, G(u)) = 20$, $d(u, t_5, G(u)) = 22$, $d(v, t_6, G(u)) = 10$, $d(v, t_2, G(u)) = 20$, $d(v, t_5, G(u)) = 24$. The dashed paths lead a $\sqrt{l}$-bad node x to Δ-neigh(x) and the circled areas represent the trees removed from $G(U)$ and rooted at a previous $\sqrt{l}$-bad node. So, $Roots = \{u, v\}$.

The following lemmas state that `CompPar` computes a correct $\sqrt{l}$-partition and that the set *Roots* has cardinality at most $\sqrt{l} + 2$ (these lemmas are similar to Claims 2.2 and 2.3 in (ELKIN; KORTSARZ, 2006) and have analogous proofs - proofs are thus omitted).

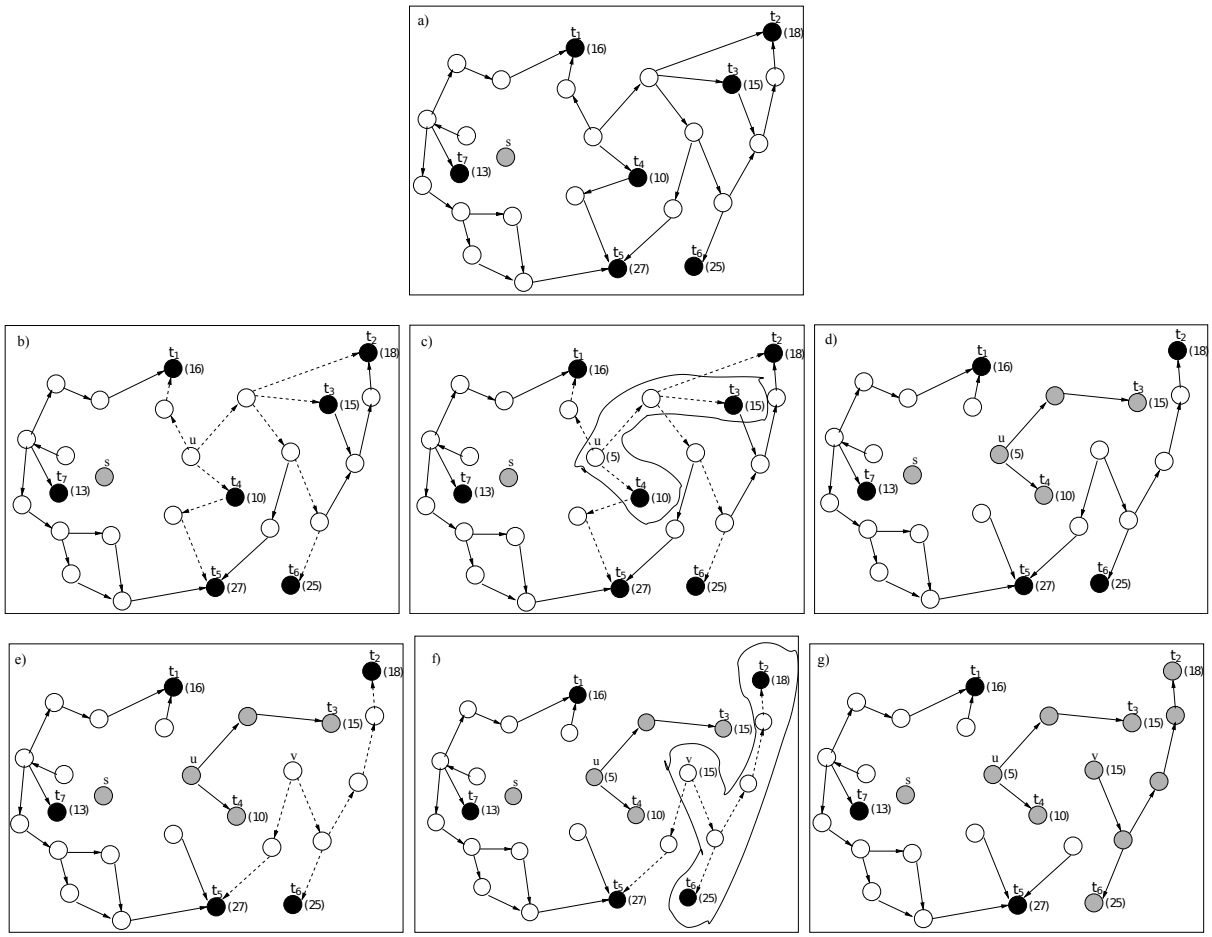

Figure 3.2 Applying procedure *CompPar*.

Lemma 3.1 follows from the condition of stopping of the loop of `CompPar` (line 2). Lemma 3.2 follows from the number of terminals removed in each iteration (line 3 of `CompPar`) along with the number of terminals left to be removed in the last iteration.

Lemma 3.1. *(ELKIN; KORTSARZ, 2006) The pair* (C, U) *output by procedure* `CompPar` *is a* $\sqrt{l}$*-partition.*

Lemma 3.2. *(ELKIN; KORTSARZ, 2006)* $|Roots| \leq \sqrt{l} + 2$*.*

After the computation of a $\sqrt{l}$-partition, we build a directed graph that contains a path from s to each node t in CT that satisfies $k \cdot dist(s, t, G)$ and which has bounded degree. Let us call this graph $G_{\sqrt{l}-Par}$. It is built as the union of a shortest path from s to each node in $Roots$ and the T_v trees calculated in line 4 of `CompPar` (represented by forest $H = (V_H, E_H)$, output by `CompPar`). This is represented by procedure `CompGraphFirstPh`.

Lemma 3.3. *The maximum out-degree of nodes in* $G_{\sqrt{l}-Par}$ *is* $\leq 2\sqrt{l} + 2$

Proof: The T_v trees computed in line 4 of `CompPar` are disjoint. Each node in these trees has degree at most $\sqrt{l}$, as each T_v is a shortest path tree to at most $\sqrt{l}$ terminals. A_{Roots}

Procedure CompPar(G=(V, E), s, k)

1 $C \leftarrow \{s\}$, $U \leftarrow V \setminus \{s\}$, $Roots \leftarrow \emptyset$
2 **while** $G(U)$ *contains a* $\sqrt{l}$*-bad node* v **do**
3 Let $Cl(v)$ be the $\lfloor\sqrt{l}\rfloor$ uncovered terminals closest to v in $G(U)$
4 Let T_v be a shortest path arborescence leading from v to $Cl(v)$ in $G(U)$
5 $C \leftarrow C \cup V(T_v)$, $U \leftarrow U \setminus V(T_v)$, $Roots \leftarrow Roots \cup \{v\}$
6 Let $H(V_H, E_H)$ be the forest formed by the union of T_v, $\forall v \in Roots$
7 Output $(C, U, Roots, H)$

Procedure CompGraphFirstPh(G, s, Roots, H)

1 $A_{Roots} \leftarrow SPT(s, Roots, G)$
2 $G_{\sqrt{l}-Par} \leftarrow A_{Roots} \cup H(V_H, E_H)$
3 $C \leftarrow V(G_{\sqrt{l}-Par})$, $U \leftarrow V \setminus C$
4 Output $(C, U, G_{\sqrt{l}-Par})$

is a shortest path tree from s to the nodes in $Roots$. As the cardinality of $Roots$ is at most $\sqrt{l}+2$ (Lemma 3.2), each node in A_{Roots} has degree at most $\sqrt{l}+2$. As $G_{\sqrt{l}-Par}$ is the union of all these trees, the degree of any of its nodes is limited to $\sqrt{l}+\sqrt{l}+2 = 2\sqrt{l}+2$. ∎

3.1.2 Multiple Set-Cover Problem

As mentioned before, the MSC problem was stated in (ELKIN; KORTSARZ, 2003, 2006). A solution was proposed in (CHEKURI; KUMAR, 2004) based on an approximation algorithm for another problem, called *Maximum Coverage with Group Budgets* (MCG).

Let $\beta(V_1, V_2, E)$ be a bipartite graph. A set $S \subseteq V_1$ is called a *set-cover* of V_2 if $N(S, \beta) = V_2$. The Multiple Set-Cover problem is stated as follows (ELKIN; KORTSARZ, 2006):

Input: A bipartite graph $\beta(V_1, V_2, E)$ with $|V_1| + |V_2| = n$. The set V_1 is partitioned into a disjoint union of sets $V_1 = \bigcup_{j=1}^{d} A_j$.

Output: A set-cover $S \subset V_1$ of V_2 which minimizes $val(S)$, where $val(S) = max\{|S \cap A_i|\}_{i=1}^{d}$. Figure 3.3 illustrates an example.

Theorem 3.1. *(CHEKURI; KUMAR, 2004). The multiple set-cover problem admits a polynomial time greedy* $(\log|V_2| + 1)$*-approximation algorithm.*

In section 3.3 we explain how MSC is solved.

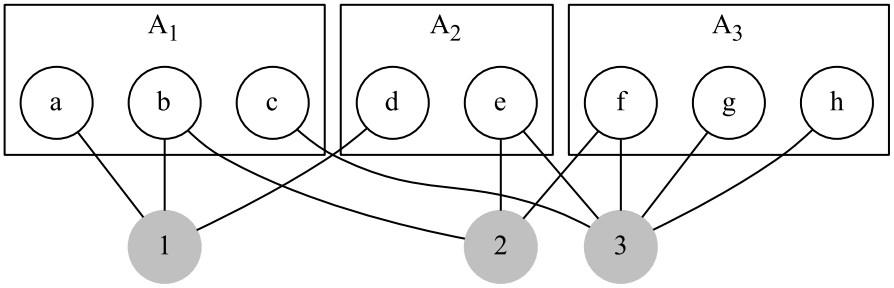

Figure 3.3 Example of MSC. Consider $V_1 = \{a, b, c, d, e, f, g, h\}$, $V_2 = \{1, 2, 3\}$ and A_1, A_2, A_3 represent the partitions of V_1 ($V_1 = A_1 \cup A_2 \cup A_3$). The set-covers $S_1 = \{a, e\}$, $S_2 = \{c, d, f\}$ and $S_3 = \{b, f\}$ are examples of optimum solutions, since $val(S_1) = val(S_2) = val(S_3) = 1$.

3.1.3 Second Phase: Covering Nodes in *UT* using an Instance of the *Multiple Set-Cover* Problem

In the second phase, the remaining nodes in UT become covered. As in (ELKIN; KORTSARZ, 2006), we use an instance of the MSC problem to find paths to cover these nodes while limiting the maximum node degree.

The second phase is divided into two parts. First, an instance of MSC is solved to find a specific subset of the covered nodes. In the second part, we define paths from these covered nodes to the uncovered terminals. The union of these paths with $G_{\sqrt{l}-Par}$ generates a graph which contains an arborescence that is a solution to DSMDStP.

The instance of MSC, $\beta = (V_1, V_2, \varepsilon)$, is defined as follows. V_1 is composed of *pseudo nodes*. A pseudo node $x_{u,v}$ is created for each edge (u, v) such that u is a covered node and v is an uncovered node. Formally:

$$V_1 = \{x_{u,v} : u \in C, v \in U, (u, v) \in E\}.$$

The set V_2 is the set of uncovered terminals, i.e. $V_2 = UT$. The edge set ε is defined as follows. There is an edge from pseudo node $x_{u,v}$ to $t \in V_2$ iff the cost of a shortest path from s to u in G, plus the cost of (u, v), plus the cost of a shortest path from v to t in $G(U)$ satisfies the spanner property for t. I.e.:

$$dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \leq k \cdot dist(s, t, G).$$

Let $A_u = \{x_{u,v} : v \in N(u, G(U))\}$. The disjoint partition of V_1 that is input for the instance of the MSC problem is $V_1 = \bigcup_{u \in C} A_u$. Each partition A_u represents the set of uncovered neighbours of node u in $G(U)$. As there is an algorithm to MSC that limits $A_u \cap S$, MSC is used to provide a bound to node degree in the final arborescence.

Figure 3.4 illustrates how an MSC instance is set. Gray nodes represent covered nodes and the others are uncovered nodes. A dashed path lead a covered node x, followed by an uncovered node y, to a terminal $t \in UT$ s.t. $dist(s, x, G) + C(x, y) + dist(y, t, G(U)) \leq k \cot dist(s, t, G)$. We can set $\beta = (V_1, V_2, \varepsilon)$ as $V_1 = \{X_{u,a}, X_{t_4,b}, X_{s,b}, X_{s,c}\}$, where the disjoint partition of V_1 can be represented by the set $\{\{X_{u,a}\}, \{X_{t_4,b}\}, \{X_{s,b}, X_{s,c}\}\}$, $V_2 = \{t_1, t_5, t_7\}$, and $\varepsilon = \{(X_{u,a}, t_1), (X_{t_4,b}, t_5), (X_{s,b}, t_5), (X_{s,c}, t_7)\}$.

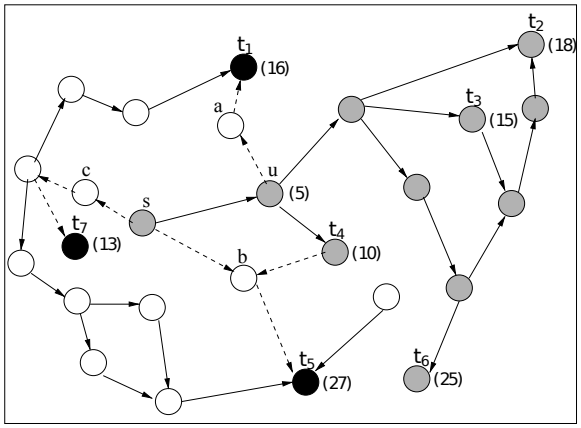

Figure 3.4 Setting an MSC instance.

The following lemmas present properties of solutions for such an MSC instance. Their proofs are similar to the proofs of Lemmas 3.3 and 3.4 in (ELKIN; KORTSARZ, 2006). So we only provide the main idea of the proofs.

Lemma 3.4. *A multiple set-cover instance $\beta = (V_1, V_2, \varepsilon)$, as defined above, admits a solution $S^* \subseteq V_1$ such that $val(S^*) \leq d^*$.*

Proof: Let T^* be an optimal solution to an instance of DSMDStP. For any node $t \in UT$, there is a path from s to t in T^*. Let u be the last node of this path that is in C and v the node after u on this path ($v \in U$). So there will be a pseudo node $x_{u,v}$ in V_1 and an edge $(x_{u,v}, t) \in \varepsilon$. As this happens to all nodes in UT and T^* has maximum degree d^*, there is a solution S^* for β such that for each node u there will be at most d^* pseudo nodes $x_{u,v}$ in the solution S^*, as $x_{u,v} \in S^*$ iff v is a child of u in T^*. Thus, $max_{c \in C}\{|S^* \cap A_c|\} \leq d^*$. ∎

Lemma 3.5. *Let D be a solution to $\beta = (V_1, V_2, \varepsilon)$ with partition $V_1 = \bigcup_{v \in C} A_v$ using Algorithm 3 (presented in (CHEKURI; KUMAR, 2004)). Then $max_{v \in C}|D \cap A_v| = (\log l + 1) \cdot d^*$.*

Proof: As d^* is greater than or equal to $val(S^*)$ for an optimal solution S^* of β and $|V_2| \leq l$, this lemma is a direct application of Theorem 3.1. ∎

In order to build the final solution to DSMDStP, we first add to $G_{\sqrt{l}-Par}$ a set of shortest paths from covered nodes to terminals in UT and compute a shortest path arborescence over the resulting graph. We use D to choose the covered nodes. The complete algorithm to DSMDStP is given in the section 3.1.4.

3.1.4 The Complete Algorithm

Algorithm 1 represents the complete algorithm. Lines 1-2 represent the first phase. After computing a $\sqrt{l}$-partition (line 1), graph $G_{\sqrt{l}-Par}$ is computed (line 2). Lines 3-4 correspond to the first part of the second phase of the algorithm, i.e. the application of the

algorithm in (CHEKURI; KUMAR, 2004) to an instance of the MSC problem (as described in Section 3.1.3). The output of this algorithm is the set D, whose elements cover V_2. Lines 5-12 correspond to the second part of the second phase, when the arborescence that is output of the algorithm is built.

In lines 5-10 the digraph $\Gamma(V_\Gamma, E_\Gamma)$ is built. For each uncovered node t in UT, we choose a node $x_{u,v}$ in D such that there is an edge $(x_{u,v}, t) \in \varepsilon$ and $C(u,v)+dist(v,t,G(U))$ is minimum. The set V_Γ is composed of the nodes u and the set of nodes on a shortest path from v to t. The set E_Γ contains the edges in these paths and edges (u,v).

In line 11, we build the graph G_f, which is the union of graphs $G_{\sqrt{l}-Par}$ and Γ. We finally compute the arborescence $\mathcal{A}_f$ that is the output of the algorithm. $\mathcal{A}_f$ is a shortest path tree in G_f rooted at s, containing shortest paths from s to the terminals.

Algorithm 1: Approximation Algorithm to DSMDStP

Input : $G = (V, E)$, $s \in V$, $T \subset V$, k
Output: $\mathcal{A}_f = (V_{\mathcal{A}_f}, E_{\mathcal{A}_f})$

1 $(C, U, Roots, H) \leftarrow$ `CompPar`(G, s, k)
2 $(C, U, G_{\sqrt{l}-Par}) \leftarrow$ `CompGraphFirstPh`$(G, s, Roots, H)$
3 Build the MSC instance $\beta = (V_1, V_2, \varepsilon)$ as in Section 3.1.3
4 $D \leftarrow$ Apply approximation algorithm (CHEKURI; KUMAR, 2004) on β
5 $\Gamma(V_\Gamma, E_\Gamma), V_\Gamma \leftarrow \emptyset, E_\Gamma \leftarrow \emptyset$
6 **foreach** $t \in V_2$ **do**
7 Choose $x_{u,v} \in D : (x_{u,v}, t) \in \varepsilon$ and $C(u,v) + dist(v,t,G(U))$ is minimum
8 $s \leftarrow sp(v,t,G(U))$
9 $V_\Gamma \leftarrow V_\Gamma \cup \{u\} \cup V(s)$
10 $E_\Gamma \leftarrow E_\Gamma \cup \{(u,v)\} \cup E(s)$
11 $G_f \leftarrow G_{\sqrt{l}-Par} \cup \Gamma(V_\Gamma, E_\Gamma)$
12 $\mathcal{A}_f \leftarrow SPT(s, T, G_f)$

Lemma 3.6. *The maximum out-degree of nodes in G_f is $\leq 2\sqrt{l} + 2 + O(\log l) \cdot d^*$.*

Proof: The algorithm adds nodes to G_f in the first and second phases of the algorithm. After the first phase, each node in G_f has maximum out-degree $2\sqrt{l} + 2$ (Lemma 3.3). In the second phase, new nodes are added to G_f and the out-degree of nodes already in G_f might increase. The new added nodes have maximum degree $\sqrt{l}$, as, in this phase, there are no more $\sqrt{l}$-bad nodes. The nodes already in G_f that might have their degrees increased are those nodes u for which there is a pseudo node $x_{u,v}$ in D. However, from Lemma 3.5, the out-degree of these nodes might be increased by at most $O(\log l) \cdot d^*$. Thus, the maximum out-degree of any node is $2\sqrt{l} + 2 + O(\log l) \cdot d^*$. ∎

Lemma 3.7. *$\forall t \in T$, $dist(s,t,G_f) \leq k \cdot (dist(s,t,G) + dist(s,t_{max},G))$, where $t_{max} \in \{t' | (t' \in T) \wedge (\forall t'' \in T : dist(s,t'',G) \leq dist(s,t',G))\}$.*

Proof: If t is covered in the first phase, $dist(s,t,G_f) \leq k \cdot dist(s,t,G)$ by construction: a tree rooted at a (previously) $\sqrt{l}$-bad node u contains paths from u to terminals t such

that $dist(s,u,G) + dist(u,t,G) \leq k \cdot dist(s,t,G)$. In particular, all nodes v (terminals or not) that are covered in the first phase, are in a path from s to a terminal t with cost less than or equal to $k \cdot dist(s,t,G)$, which is $\leq k \cdot dist(s,t_{max},G)$.

For the terminals t that are only covered in the second phase (as in the proof of Lemma 3.4, for each t, there is a path between s and t), observe that a pseudo node $x_{u,v}$ and an edge $(x_{u,v},t)$ are inserted in the input graph to the MSC instance if $dist(s,u,G) + C(u,v) + dist(v,t,G(U)) \leq k \cdot dist(s,t,G)$. Thus, in particular, $C(u,v) + dist(v,t,G(U)) \leq k \cdot dist(s,t,G)$. As $dist(s,u,G) \leq k \cdot dist(s,t_{max},G)$, we have $dist(s,t,G_f) \leq k \cdot (dist(s,t,G) + \cdot dist(s,t_{max},G))$. ∎

By Lemma 3.7, the algorithm can violate the spanner property for some terminals. Observe, however, that this might only happen to terminals covered in the second phase. Terminals covered in the first phase have their spanner constraint satisfied.

Theorem 3.2. *The approximation algorithm generates an arborescence $\mathcal{A}_f$ with bounded out-degree $2\sqrt{k}+2+O(\log l)\cdot d^*$ and that has paths from s to each terminal $t \in T$ with cost less than or equal to $k \cdot (dist(s,t,G) + dist(s,t_{max},G))$, where $t_{max} \in \{t' | (t' \in T) \wedge (\forall t'' \in T : dist(s,t'',G) \leq dist(s,t',G))\}$.*

Proof: The $\mathcal{A}_f$ tree is generated by computing minimum cost paths in G_f from s to each $t \in T$. The theorem follows from the fact that all nodes in G_f have out-degree $\leq 2\sqrt{l} + 2 + O(\log l) \cdot d^*$ (Lemma 3.6) and that G_f contains paths from s to each $t \in T$ with costs $\leq k \cdot (dist(s,t,G) + dist(s,t_{max},G))$ (Lemma 3.7). ∎

3.1.5 Complexity

The `CompPar` procedure has its complexity bounded by $(\sqrt{|T|} + 2)O(|V|^3)$ due to the number of loop iterations (Lemma 3.2) and the complexity of building a shortest path tree (SPT) for the $\sqrt{l}$-bad node candidates, using Dijkstra's algorithm (CORMEN et al., 2009). The complexity of the procedure `CompGraphFirstPh` is bounded by the running time of building SPT, so it is bounded by $O(|V|^2)$.

The complexity of building the MSC instance (line 3) is related to the complexity of composing the sets V_1 and ε. The former is bounded by $O(|V|^2)$ where the latter is bounded by the running time of all-pair-shortest paths (APSP) algorithm, which is $O(|V|^3)$ (CORMEN et al., 2009).

We assume the algorithm described in (CHEKURI; KUMAR, 2004) (Algorithm 4) for solving instances of the MSC problem (line 4). This algorithm is based on an algorithm (Algorithm 3) for another problem, called MCG. The algorithm to MCG depends on calls to an oracle, which have, in our case, $O(|V|^2|T|)$ running time. The complexity of MCG will thus be $O(|V|^3|T|^2)$. As the MSC algorithm consists in iteratively applying the MCG algorithm, the running time of line 4 is $O((\log |T|)(|V|^3|T|^2)$ (refer to (CHEKURI; KUMAR, 2004) or Section 3.3 for the details).

Using the result of the APSP algorithm run for building the MSC instance, the complexity of a single run of lines 7-10 is bounded to $O(|V|^2)$. As the number of iterations of the loop in lines 6-10 is bounded by the size of D, the complexity of lines 6-10 is $|T|O(|V|^2)$.

So, the complexity of Algorithm 1 is bounded by the complexity of solving MSC, which is $O((\log |T|)(|V|^3|T|^2)$.

3.2 HEURISTIC FOR THE DSMDSTP

In this section we describe a heuristic called *Sliced and Iterative MSC* (SIM) for the DSMDStP. In this heuristic, the cost of the paths from s to each terminal t in the final arborescence is less than or equal to $k \cdot (\lfloor\sqrt{|T|}\rfloor + 2) \cdot dist(s, t, G)$.

3.2.1 Algorithm

As the name says, the heuristic is iterative. The terminals are iteratively covered in ascending order of the costs of the shortest paths from the source node to each of them in G. At each iteration, the heuristic covers the next uncovered $\lfloor\sqrt{l}\rfloor$ terminals in this order, until there are no more uncovered nodes. Instead of using MSC only once to find paths to all the remaining uncovered terminals (as in the approximation algorithm), we use MSC at each iteration to cover the next uncovered $\lfloor\sqrt{l}\rfloor$ terminals in order. The set of these terminals will be referred to as $Next_{ter}^{\sqrt{l}}$.

In order to build an MSC instance, we create pseudo nodes $x_{u,v}$ as in Section 3.1.3. However, in order to try to decrease the out-degree of nodes, we mark nodes u that have already been part of a solution of an MSC instance (in a previous iteration). We try to create paths to uncovered terminals from unmarked nodes. However, when necessary, we use marked nodes. The set of marked nodes is called $Marked$.

SIM is represented as Algorithm 2. In line 1, the sets C, U, $Marked$ and l are initialized (C, U and l are used as in Section 3.1).

The main loop of SIM (lines 2-16) represents the iterations. In lines 3-6, we define the values of V_1, V_2 and ε to build an instance $\beta = (V_1, V_2, \varepsilon)$ of MSC. The sets V_1, V_2 and ε are initialized as follows:

$$V_1 = \{x_{u,v} : u \in (C \setminus Marked), v \in U, (u, v) \in E\},$$
$$V_2 = Next_{ter}^{\sqrt{l}},$$
$$\varepsilon = \{(x_{u,v}, t) : x_{u,v} \in V_1 \wedge t \in Next_{ter}^{\sqrt{l}} \wedge C(u, v) + dist(v, t, G(U)) \leq k \cdot dist(s, t, G)\}.$$

First, we only consider unmarked nodes in C, as explained before. But, in doing so there is the possibility of there being nodes in V_2 that cannot be covered (for example, if the only path from a covered to an uncovered node that has cost $\leq k \cdot dist(s, t, G)$ for some t has, as its last covered node, an already marked one). So, when needed, we include in V_1 pseudo nodes $x_{u,v}$ for marked nodes u. In these cases, we include a node $x_{u,v}$ such that u is a marked node with the smallest out-degree and the sum of the cost of edge (u, v) and of a shortest path from v to t in $G(U)$ is $\leq k \cdot dist(s, t, G)$ (lines 4-6). We thus apply the approximation algorithm 4 to $\beta(V_1, V_2, \varepsilon)$, using $V_1 = \bigcup_{u \in C} A_u$ as the partition, where $A_u = \{x_{u,v} : \exists v \in V$ such that $x_{u,v} \in V_1\}$ (line 7). D is the output of this algorithm.

In lines 8-14 we choose paths from covered nodes to the terminals in $Next_{ter}^{\sqrt{l}}$. For each node $t \in V_2$ we choose a pseudo node $x_{u,v}$ in D such that there is a path from u

Algorithm 2: SIM - *Sliced and Iterative MSC*

Input : $G = (V, E)$, $s \in V$, $T \subset V$, k
Output: $\mathcal{A}_f = (V_{\mathcal{A}_f}, E_{\mathcal{A}_f})$

1 $C \leftarrow \{s\}$, $U \leftarrow V \setminus \{s\}$, $Marked \leftarrow \emptyset$, $l \leftarrow |T|$
2 **while** $UT \neq \emptyset$ **do**
3 Set V_1, V_2 and ε using the current values of C, U, $Marked$ and $Next_{ter}^{\sqrt{l}}$ (See Section 3.2.1)
4 **foreach** $t : t \in V_2$ *and* $\nexists$ *edge* $(x, t) \in \varepsilon$ *for any* x **do**
5 Let $x_{u,v}$ be a pseudo node such that:
 $u \in Marked$,
 u has the smallest out-degree and
 $C(u, v) + dist(v, t, G(U)) \leq k \cdot dist(s, t, G)$
6 $V_1 \leftarrow V_1 \cup \{x_{u,v}\}$, $\varepsilon \leftarrow \varepsilon \cup \{(x_{u,v}, t)\}$
7 $D \leftarrow$ Apply approximation algorithm 4 on $\beta(V_1, V_2, \varepsilon)$
8 $\Gamma(V_\Gamma, E_\Gamma) \leftarrow \emptyset$
9 **foreach** $t \in V_2$ **do**
10 Choose a node $x_{u,v} \in D : (x_{u,v}, t) \in \varepsilon$ and $C(u, v) + dist(v, t, G(U))$ is minimum
11 $Marked \leftarrow Marked \cup \{u\}$ { for which $x_{(u,v)}$ was chosen in the last line}
12 $s \leftarrow sp(v, t, G(U))$
13 $V_\Gamma \leftarrow V_\Gamma \cup \{u\} \cup V(s)$
14 $E_\Gamma \leftarrow E_\Gamma \cup \{(u, v)\} \cup E(s)$
15 $C \leftarrow C \cup V_\Gamma$, $U \leftarrow U \setminus V_\Gamma$
16 $\mathcal{A}_f \leftarrow \mathcal{A}_f \cup \Gamma(V_\Gamma, E_\Gamma)$

to t with cost $\leq k \cdot dist(s, t, G)$. These are the nodes $x_{u,v}$ for which there is an edge $(x_{u,v}, t) \in \varepsilon$. We choose the pseudo node $x_{u,v}$ for which the path composed of the edge (u, v) and a shortest path from v to t has the smallest cost (line 10). We thus mark node u (line 11) and store this path in Γ (lines 12-14).

In line 15, the set of covered (C) and uncovered (U) nodes are updated. The nodes in $V(\Gamma)$ become covered. Finally, the nodes and edges of Γ are added to the arborescence $\mathcal{A}_f$ (line 16).

At the end of the last iteration, $\mathcal{A}_f$ contains the final arborescence.

Lemma 3.8. *Let $\mathcal{A}_f$ be output by SIM. $\forall t \in T$, there is a path from s to t in $\mathcal{A}_f$.*

Proof: Let T^* be an optimal solution to an instance of DSMDStP. For any node $t \in UT$, there is a path from s to t in T^*. Similar to the proof of Lemma 3.4, let u be the last node of this path that is in C and v the node after u on this path ($v \in U$). Irrespective the iteration which t is added to $Next_{ter}^{\sqrt{l}}$, there will be a pseudo node $x_{u,v}$ in V_1 and an edge $(x_{u,v}, t) \in \varepsilon$. So, after applying approximation algorithm 2 on $\beta(V_1, V_2, \varepsilon)$ (line 7), we know there will be at least one path leading from a covered node u to t, and one of

these paths is chosen by SIM to be part of $\mathcal{A}_f$ (line 12), concluding the proof. ∎

Lemma 3.9. *The number of SIM's iterations is bounded by* $\lfloor\sqrt{l}\rfloor + 2$.

Proof:

Consider the number of terminals l. If l is a perfect square, it is easy to see that all terminals will be covered in $\sqrt{l}$ iterations (since $\lfloor\sqrt{l}\rfloor$ terminals = $\sqrt{l}$ terminals). Now let us examine the case where l is not a perfect square.

Let $\sqrt{l} = \lfloor\sqrt{l}\rfloor + \mathcal{F}$, where $\mathcal{F} \in]0,1[$. Then $l = (\lfloor\sqrt{l}\rfloor + \mathcal{F}) \cdot (\lfloor\sqrt{l}\rfloor + \mathcal{F})$. So:

$$
\begin{aligned}
l &= \lfloor\sqrt{l}\rfloor \cdot \lfloor\sqrt{l}\rfloor + (\lfloor\sqrt{l}\rfloor \cdot \mathcal{F}) + (\lfloor\sqrt{l}\rfloor \cdot \mathcal{F}) + (\mathcal{F} \cdot \mathcal{F}) \\
l &= (\lfloor\sqrt{l}\rfloor \cdot \lfloor\sqrt{l}\rfloor) + [\lfloor\sqrt{l}\rfloor \cdot (2 \cdot \mathcal{F})] + (\mathcal{F} \cdot \mathcal{F}) \\
l &= (\lfloor\sqrt{l}\rfloor \cdot \lfloor\sqrt{l}\rfloor) + [\lfloor\sqrt{l}\rfloor \cdot (2 \cdot \mathcal{F})] + \left[\lfloor\sqrt{l}\rfloor \cdot \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor\sqrt{l}\rfloor}\right)\right] \\
l &= (\lfloor\sqrt{l}\rfloor \cdot \lfloor\sqrt{l}\rfloor) + \left(\lfloor\sqrt{l}\rfloor \cdot \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor\sqrt{l}\rfloor}\right)\right]\right) \\
l &= \lfloor\sqrt{l}\rfloor \cdot \left(\lfloor\sqrt{l}\rfloor + \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor\sqrt{l}\rfloor}\right)\right]\right)
\end{aligned}
$$

We can divide l into two numbers: $\lfloor\sqrt{l}\rfloor$ and $(\lfloor\sqrt{l}\rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor\sqrt{l}\rfloor})])$. The first number $(\lfloor\sqrt{l}\rfloor)$ matches the number of terminals covered by SIM in each iteration (see Section 3.2.1), so we need to find an integer upper bound I_{UB} (sufficient condition) for the real number $(\lfloor\sqrt{l}\rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor\sqrt{l}\rfloor})])$, which means I_{UB} iterations is sufficient to cover l terminals. We have:

$$
\begin{aligned}
\lfloor\sqrt{l}\rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor\sqrt{l}\rfloor}) = \\
\lfloor\sqrt{l}\rfloor + (2 \cdot (\sqrt{l} - \lfloor\sqrt{l}\rfloor)) + \frac{(\sqrt{l} - \lfloor\sqrt{l}\rfloor)^2}{\lfloor\sqrt{l}\rfloor} = \\
\lfloor\sqrt{l}\rfloor + \frac{l}{\lfloor\sqrt{l}\rfloor} - \lfloor\sqrt{l}\rfloor = \\
\lfloor\sqrt{l}\rfloor + \frac{l - \lfloor\sqrt{l}\rfloor^2}{\lfloor\sqrt{l}\rfloor}
\end{aligned}
$$

Since $\lfloor\sqrt{l}\rfloor^2$ is the greatest perfect square less than l, then $(l - \lfloor\sqrt{l}\rfloor^2) \leq 2 \cdot \lfloor\sqrt{l}\rfloor$ (the proof is postponed to the last paragraph). This implies that $\lfloor\sqrt{l}\rfloor + \frac{l - \lfloor\sqrt{l}\rfloor^2}{\lfloor\sqrt{l}\rfloor} \leq \lfloor\sqrt{l}\rfloor + \frac{2 \cdot \lfloor\sqrt{l}\rfloor}{\lfloor\sqrt{l}\rfloor} = \lfloor\sqrt{l}\rfloor + 2$, where $\lfloor\sqrt{l}\rfloor + 2$ is our I_{UB}, concluding the proof.

Now, let us prove that $(l - \lfloor\sqrt{l}\rfloor^2) \leq 2 \cdot \lfloor\sqrt{l}\rfloor$. Let SR_x be a perfect square and let R_x be its root. For two consecutive perfect squares SR_i and SR_j, it is known that

$SR_j - SR_i = R_i + R_j$ (see Proposition 2 in (FIBONACCI; SIGLER, 1987)). Alternatively, $SR_j - SR_i = 2 \cdot R_i + 1$. Let X be a non perfect square and let GSR_X be the greatest perfect square less than X. Let R_X be GSR_X's root. Then, $X - GSR_X \leq 2 \cdot R_X$. ∎

Lemma 3.10. *Let $\mathcal{A}_f$ be output by SIM. $\mathcal{A}_f$ is an arborescence.*

Proof: At the beginning, set C contains only node s. At the end of the first iteration, $\mathcal{A}_f$ will be an arborescence, as shortest paths from s to each terminal $t \in V_2$ are chosen to be added to $\mathcal{A}_f$ (line 12). Analogously, starting at the second iteration, for each pair $(x_{u,v}, t)$, where $t \in V_2$ and $x_{u,v}$ is the chosen node for t in line 10, only shortest paths from v to t (line 12) are added to $\mathcal{A}_f$. The proof follows from the inexistence of circuits in $\mathcal{A}_f$, as these paths are formed only by nodes in U (except the first node v). ∎

Theorem 3.3. *Let $\mathcal{A}_f$ be output by SIM. $\forall t \in T, dist(s, t, \mathcal{A}_f) \leq (\lfloor\sqrt{l}\rfloor + 2) \cdot k \cdot dist(s, t, G)$.*

Proof: At each iteration, new terminals are added to $\mathcal{A}_f$. Let t_h be the terminal u in $Next_{ter}^{\sqrt{l}}$ in iteration i for which $dist(s, u, G)$ is highest. Paths are added to $\mathcal{A}_f$ from an already covered node to a terminal t in $Next_{ter}^{\sqrt{l}}$.

First, observe that all paths added to $\mathcal{A}_f$ in iteration i have cost $\leq k \cdot dist(s, t_h, G)$, as there will be an edge $(x_{u,v}, t)$ in the set ε of the MSC instance if $C(u, v) + dist(v, t, G(U)) \leq k \cdot dist(s, t, G) \leq k \cdot dist(s, t_h, G)$. Additionally, as the heuristic covers $\lfloor\sqrt{l}\rfloor$ terminals at each iteration (except possibly the last one), there can be at most $\lfloor\sqrt{l}\rfloor + 2$ iterations (Lemma 3.9).

Now, let t be a terminal added to $\mathcal{A}_f$ in iteration j. Terminal t must be a terminal in $Next_{ter}^{\sqrt{l}}$. t can be reached from the root through a set of paths, added to $\mathcal{A}_f$ in all iterations until iteration j. As at each iteration terminals are in $Next_{ter}^{\sqrt{l}}$ following a non-decreasing order of their minimum cost paths from the source node, all these paths have cost $\leq k \cdot dist(s, t, G)$. As there are at most $\lfloor\sqrt{l}\rfloor + 2$ such paths, $dist(s, t, \mathcal{A}_f) \leq (\lfloor\sqrt{l}\rfloor + 2) \cdot k \cdot dist(s, t, G)$. ∎

3.2.2 Complexity

The complexity of some parts of SIM is similar to algorithm 1, but in SIM the size of V_2 is $\sqrt{|T|}$ rather than $O(|T|)$. The complexity of setting MSC instance in line 3 is relatively similar to the complexity in algorithm 1, which means it is $O(|V|^3)$.

The loop in lines 4 to 6 has its complexity bounded by the number of iterations and the size of *Marked* set, since the loop takes advantage of the results for building the APSP and SPT in the MSC instance. At each iteration of the loop in line 2, the *Marked* set has, at most, $\lfloor\sqrt{|T|}\rfloor$ elements added to it, so in the last iteration the maximum size of *Marked* is $O(|T|\sqrt{|T|})$. Since the number of iterations in line 4 is, at most, $\lfloor\sqrt{T}\rfloor$, the complexity of this loop is $O(|T|^2)$.

Considering the new size of V_2 set (compared to this set in algorithm 1), the complexity of running the algorithm for the MSC problem (line 7) is $O((\log\sqrt{|T|})(|V|^3|T|))$.

The loop in lines 9 to 14 has its complexity bounded by the number of iterations and the size of D (line 11), since this loop also takes advantage of the results for building the APSP in the MSC instance, so retrieving the nodes in line 12 is $O(1)$. Consequently, the running time of this loop is $(\sqrt{|T|}+2)O(V)$.

Considering the number of iterations in line 2 and the fact that the complexity of SIM is bounded by the complexity of running the algorithm for the MSC problem, we conclude the complexity of SIM is $O((\log\sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.

3.3 HOW MSC IS SOLVED

Observe that the MSC is a minimization version of an optimization problem. The solution proposed in (CHEKURI; KUMAR, 2004) for MSC consists in an iterative application of the algorithm for the MCG problem, where the latter is a maximization version of the optimization problem. Basically, MCG consists in, given the same input for MSC and a budget k_i for each A_i, finding a set-cover $S \subseteq V_1$ such that $|S \cap A_i| \leq k_i$, and the number of elements covered is maximized. Formally:

Input: A ground set X and:

- $S_1, ..., S_m$, where $S_i \subseteq X$, $1 \leq i \leq m$.
- $G_1, ..., G_l$, where $G_j \subseteq \{S_1, ..., S_m\}$, $1 \leq j \leq l$.
- An integer k and a specific integer k_j for each G_j, $1 \leq j \leq l$.

Output: A set $H \subseteq \{S_1, ..., S_m\}$ such that the number of elements of X covered by the sets of H is maximized. H is a solution iff $|H| \leq k$ and $|H \cap G_j| \leq k_j$, $1 \leq j \leq l$.

Algorithm 3 represents the algorithm to the MCG problem. It greedily chooses among the available sets the one (S_r) that covers the maximum number of uncovered elements of X in each iteration. Observe that the selection of a set S_i has to respect the constraint k_j of each set G_j. The function $\mathcal{A}$ in line 6 represents an α-approximate oracle. An α-approximate oracle $\mathcal{A}(G_i, X')$ outputs a set $S_j \in G_i$ such that $S_j \cap X' \geq \frac{1}{\alpha} max_{D \in G_i}|D \cap X'|$. For $\alpha = 1$, the following theorem (from (CHEKURI; KUMAR, 2004)) holds:

Theorem 3.4. *(CHEKURI; KUMAR, 2004). Algorithm 3 is a 2-approximation algorithm for MCG.*

As previously explained, the algorithm to the MSC problem basically consists in an iterative application of Algorithm 3. Algorithm 4 represents the algorithm to the MSC problem.

In Algorithm 4, some parameters are passed to algorithm 3. Each G_j is related to a covered node u that originated a pseudo node $x_{u,v}$ and each S_i is related to a pseudo node. The constraint values (represented by $constraintValues$) and v^* represent, respectively, the values k_j and k that are input to MCG. V_2 represents the ground set X of MCG. Using Theorem 3.4, it is simple to see that Algorithm 4 solves MSC, covering V_2 in at most $\log|v_2| + 1$ iterations (see (CHEKURI; KUMAR, 2004) for more details).

Algorithm 3: Greedy algorithm for MCG (based on (CHEKURI; KUMAR, 2004))

```
1  H ← ∅, X' ← X;
2  for i=1 to l do selec[i] ← F, selecQt[i] ← 0;
3  ;
4  for j=1 to k do
5      for i=1 to l do
6          if selec[i] = False then
7              S_i ← A(G_i, X');
8          else
9              S_i ← ∅;
10     r ← argmax_i |S_i|;
11     selecQt[r] = selecQt[r] + 1;
12     if selecQt[r] = budgetsK[r] then selec[r] ← True;
13     ;
14     H ← H ∪ {S_r}, X' ← X' \ S_r;
15 return (H, X \ X');
```

Regarding the relation between MSC and MCG problems (introduced in the last paragraph), let $\beta = (V_1, V_2, \varepsilon)$ be an instance of MSC as defined in Section 3.1.3. For each $x_{u,v} \in V_1$, suppose there is an order between the nodes in $N(u, G(U))$. Let $N_i(u)$ be the i-th neighbor of u in U. Remember that the disjoint partition of V_1 for the instance of the MSC is $V_1 = \bigcup_{u \in C} A_u$, where $A_u = \{x_{u,v} : v \in N(u, G(U))\}$. The sets S_i, G_j and X, that are passed as parameters to the instance of MCG in Algorithm 4 (line 5), are formed in the following way:

- $X = V_2$.
- $S_{(u,v)} = \{y : (y \in X) \wedge (\exists\ (x_{u,v}, y) \in \varepsilon\}$ for each pseudo-node $x_{u,v} \in V_1$.
- $G_u = \{S_{(u,N_1(u))}, S_{(u,N_2(u))}, ..., S_{(u,N_{|N(u,G(U))|}(u))}\}$ for each index u s.t. A_u is member of the disjoint partition of V_1.

Observe that it is mentioned in the description of Algorithm 4 that v^* is guessed (line 1). It is easy to see that $1 \leq v^* \leq |V_2|$. To find out a value v', s.t. $v^* \leq (\log |V_2| + 1) \cdot v' \leq (\log |V_2| + 1) \cdot v^*$ and all nodes of V_2 are covered by applying Algorithm 4 with v' as the guessed v^*, we can use the following process: we initially execute Algorithm 4 with v' set to 1; if some nodes of V_2 are left uncovered, the value assigned to v' was less than the optimum, otherwise, according to Theorem 3.1, a solution would be found. So we continue incrementing v'. We increment v' and repeat the process until all nodes of V_2 are covered.

In the next chapter we present an evaluation of the proposed algorithms. We compare them with another algorithm, regarding the degree and the spanner property. We will

Algorithm 4: Algorithm for MSC (CHEKURI; KUMAR, 2004)

```
1 Guess (search for) the optimal value v* of the MSC instance;
2 constraintValues ← l copies of v*;
3 S ← ∅;
4 while V2 ≠ ∅ do                          /* iterates at most log|V2| + 1 times */
5 |   (S', V2') ← Algorithm 3(⋃_{i=1}^{m} Si, ⋃_{j=1}^{l} Gj, V2, constraintValues, v*);
6 |   S ← S ∪ S';
7 |   V2 ← V2 \ V2';      /* some sets Si are deleted as well as (possibly)
  |    sets Gj */
8 return S;
```

show that although our algorithms do not guarantee to respect the spanner property, in practice, the results were quite good for both algorithms.

Chapter 4

EVALUATION

In this chapter we present the results of a series of experiments to evaluate the approximation algorithm and SIM in relation to the maximum out-degree and the costs of the paths from the source node to each terminal in the generated arborescence. We first describe the parameters used in the experiments and implementation details (Section 4.1). Then, we describe the algorithms and metrics used in the experiments (Section 4.2). At last, we present the results of experiments for the chosen metrics and algorithms (Section 4.3).

As mentioned before, we were not able to satisfy the spanner constraint. However, our experiments show quite good results. More specifically, for the approximation algorithm, in more than half of the situations there was no violation of the spanner constraint (this will be better explained in Section 4.3, through graphics) and in the other half the violation was by a low factor (on average, less than 10%). For the heuristic, although the results were not as good as for the other algorithm, the ratio of violation (which is the main metric) was quite low too, and the heuristic always outperformed the other algorithms concerning the degree, besides showing a uniform behaviour for both the degree and spanner violation, what contributes to scalability.

4.1 PARAMETERS AND IMPLEMENTATION DETAILS

We created a specific Java program to perform the experiments. The source code of the program can be found in (BRAGA, 2012).

In all experiment scenarios, nodes were spread in a 500×500 Euclidian space. For each node u we assumed a transmission range of approximately 125 distance units. Each node can communicate with all nodes inside the circumference with radius equal to its range and centered at that node. The cost associated with each edge (u, v) is equal to the Euclidean distance between u and v.

The number of nodes (*network size*), n, ranged from 60 to 300. For each network size, the number of terminals, t, ranged from 10 to 50. For each pair of network size and terminals set size, we performed experiments with different spanning factors, sf, ranging

from 1 to 2. For each combination of network size, terminals set size and spanning factor, we generated 30 scenarios.

The algorithm for solving the MSC problem is based on an algorithm described in (CHEKURI; KUMAR, 2004). This algorithm is based on an α-*approximate oracle*, which results a solution set whose quality deviates by a bounded factor from an optimum set (see Section 3.3 for more details). We implemented an optimum oracle, i.e. an 1-approximate oracle.

4.2 ALGORITHMS AND METRICS

As we are not aware of any other algorithm to DSMDStP, we compared the algorithms we presented in this dissertation against a shortest path tree algorithm (SPT), i.e. an algorithm that calculates a tree formed by shortest paths from the source node to each of the terminals. The reason for choosing SPT is that it is a simple and efficient algorithm that generates a tree which satisfies the spanner restrictions as it generates a single-sink 1-spanner Steiner tree ($k = 1$). Another reason for choosing SPT were the similarities between the proposed algorithms (mainly the approximation algorithm) and SPT, since the criterion for choosing the spanner paths (paths that respect the spanner property) is based on shortest paths. These similarities were reflected in the experiment results.

We used four metrics to compare and analyze the algorithms. For each metric, we considered the average of the 30 scenarios. Besides evaluating the maximum out-degree, we measured the final path costs. As we cannot guarantee a root-stretch factor of k, but one of $k \cdot \left(1 + \frac{max_{t \in T}\{dist(s,t,G)\}}{min_{t \in T}\{dist(s,t,G)\}}\right)$ and $k \cdot (\lfloor \sqrt{|T|} \rfloor + 2)$ for Algorithm 1 and SIM respectively, we calculated the ratio between the final path's cost and the desired path's cost (i.e. k times the shortest cost path from the source node to the terminal). We would like to verify the extent of the violation in the experiments. Let us call this ratio *Cost Violation Ratio* (CVR). CVR was calculated using the following formula:

$$CVR = \frac{\sum_{\forall t \in T_{vio}} \frac{dist(s,t,\mathcal{A}_f)}{k \cdot dist(s,t,G)}}{|T_{vio}|} \tag{4.1}$$

where:

$$T_{vio} \subseteq T = \{t | t \in T \wedge dist(s,t,\mathcal{A}_f) > k \cdot dist(s,t,G)\}.$$

Observe that we take into consideration only the terminals whose final costs violate the spanner constraint (represented by the set T_{vio}). We defined the metric in this way because we would like to answer the following question: when the violation occurs, by how much does it occur? Even though this metric worse our results (by increasing the CVR value), it is a more accurate metric for the former question than we had considered all the terminals. CVR's value is always ≥ 1. The lower the CVR, the lower the violation.

As CVR is an average, we also calculated the worst violation ratio (the greatest ratio in the numerator of CVR). Let us call it *MAX_CVR*. Although we ran 30 scenarios for each pair of network size and terminals set size, in order to keep the coherency of the metrics, we averaged CVR and MAX_CVR only by the number of scenarios where violation occurred.

The paths to the nodes covered in the first phase of our algorithm and in the first iteration of SIM have their spanner constraints satisfied. It is supposed, however, that more terminals have their spanner constraints satisfied, as we use the criteria of choosing shortest cost paths in the algorithms. As in CVR's denominator we considered $|T_{vio}|$ instead of $|T|$, we calculated additionally the percentage of spanner constraints violated. This percentage is represented by the PVT (Percentage of Violated Terminals) metric. Similar to CVR, we averaged PVT by the number of scenarios where violation occurred. Formally:

$$PVT = \frac{|T_{vio}|}{|T|} \cdot 100\%. \tag{4.2}$$

For both CVR and PVT, we considered only the scenarios where violation occurred. We calculated additionally the percentage of scenarios where violation occurred. This is represented by the PVR (Percentage of Violated Runs) metric.

4.3 EXPERIMENT RESULTS

Figure 4.1 shows the maximum out-degree for three different parameters: network size (Figure 4.1a), size of the terminals set (Figure 4.1b) and spanning factor (Figure 4.1c). For each of these parameters, fixed values for the others were used. The fixed values are indicated by the labels above the plots (*NS* - Network Size; *Ter* - Terminals Set Size; *SF* - Spanning Factor). Algorithm 1 is referred throughout this section by the abbreviation *APPROX*.

In Figure 4.1a, generally, the degree value is stable for both APPROX and SIM, irrespective of the network size. So, regarding the degree, these algorithms support network scalability well. On the other hand, the degree of SPT increases quickly and the final arborescence has a degree three times greater than the other algorithms in denser networks. Regarding the number of terminals (Figure 4.1b), the greater the number of terminals, the greater the degree of the three algorithms. This behaviour is expected by the two proposed algorithms, since in each step (each *phase* for APPROX and each *iteration* for SIM) of the algorithms, the possible maximum degree is affected by the number of terminals. But, the difference between the three algorithms is the slope ratio. For the proposed algorithms the increasing ratio is low whereas it is high for SPT. Finally, in Figure 4.1c, we ranged the spanning factor. As expected, the degree resulted from applying SPT is the same, besides being high. For the lowest value of the spanning factor, APPROX yielded a much higher degree when comparing to the cases with higher spanning factors, but the maximum degree decreases drastically with the increase in the spanning factor. SIM exhibits a uniform behaviour, so the achieved maximum degree mean was roughly unaffected by the spanning factor. For all the three parameters, SIM outperformed APPROX.

Since APPROX imitates, in some aspects, the SPT algorithm, we evaluated the algorithms for the specific case of spanning factor equal to 1. This graphic is illustrated by Figure 4.2. Observe that the curve of APPROX is very close to the curve of SPT. Assuming a spanning factor of 1 corresponds to the situation when the restrictions on

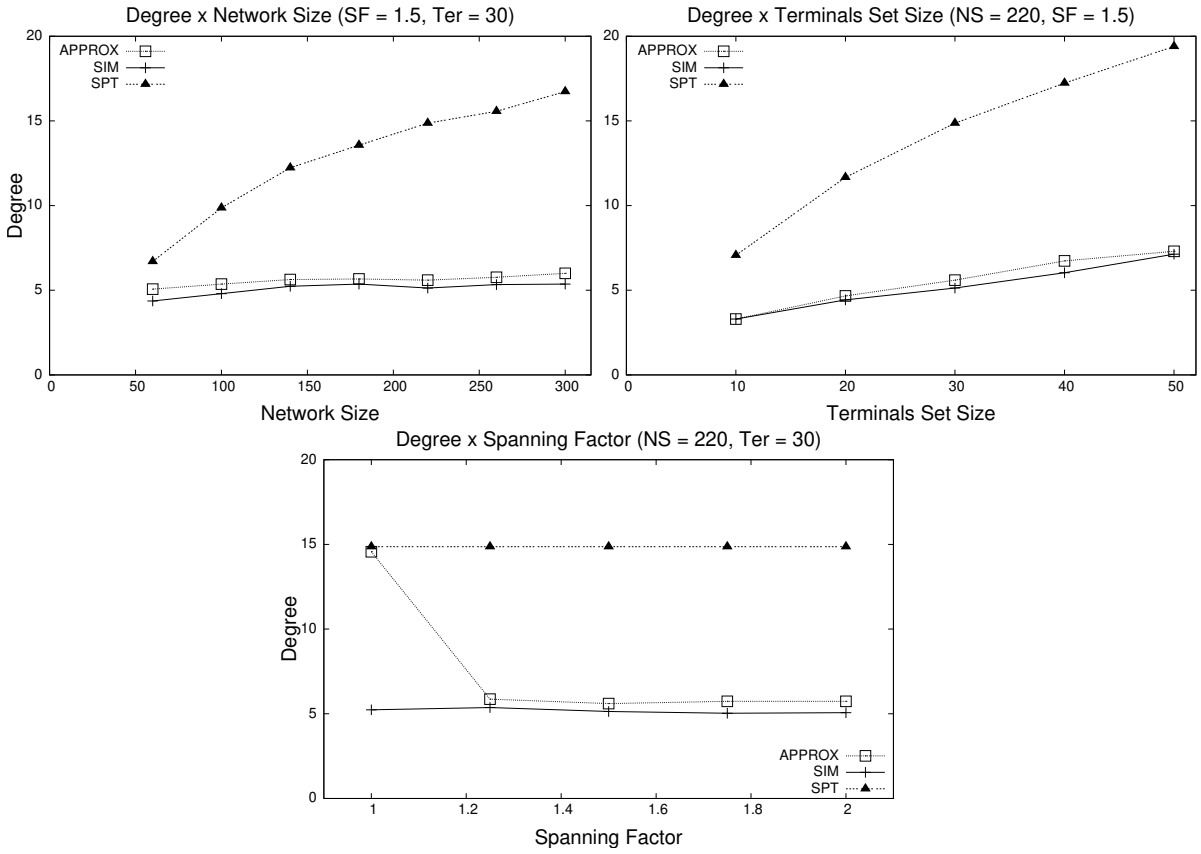

Figure 4.1 Maximum out-degree in function of (a) network size, (b) number of terminals, and (c) spanning factor

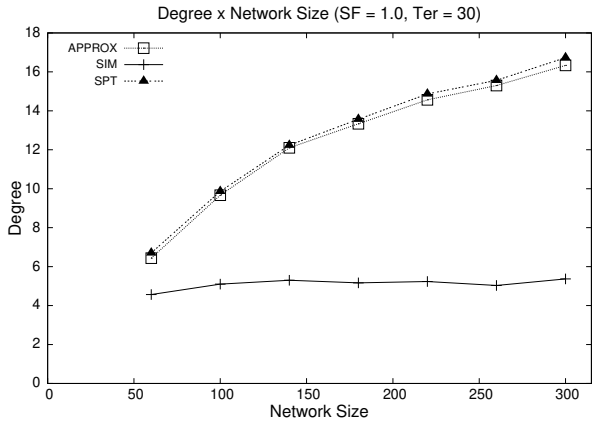

Figure 4.2 Degree x Network Size when spanning factor is 1

the paths from the source node to the terminals correspond to the minimum cost paths. As the restrictions are tighter, there will be less paths to choose leading to the uncovered terminals and respecting the constraints in the second phase of APPROX. This might contribute to an increase in the degree of nodes. On the other hand, SIM is quite good for

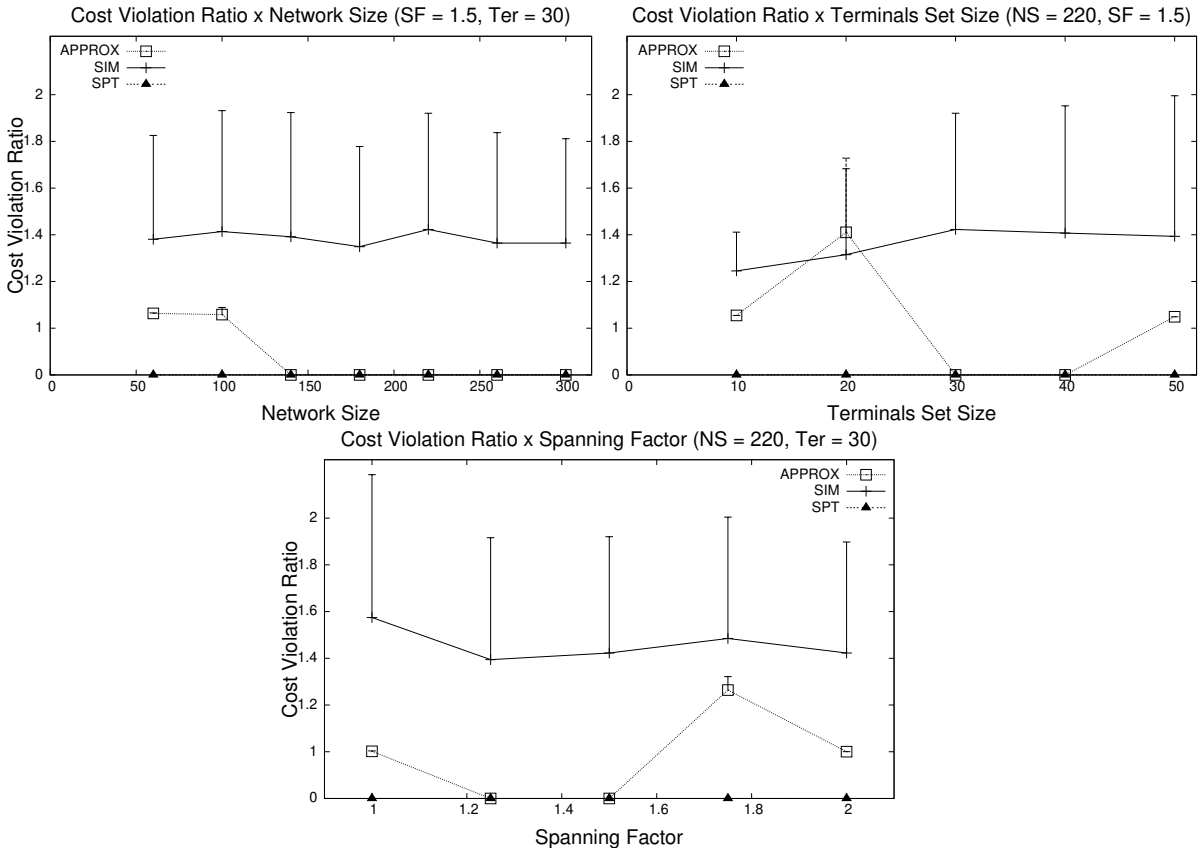

Figure 4.3 Cost Violation Ratio in function of (a) network size, (b) number of terminals, and (c) spanning factor

scalability in this situation too. Although both APPROX and SIM apply the algorithm for the MSC problem, only APPROX exhibited behaviour similar to SPT. This might be explained by the restrictions on the edges that are allowed to be part of a MSC instance, where for APPROX the restriction considers paths from the source node to the terminal and for SIM the restriction only considers the path from a covered node (which is unique) to the terminal.

Figure 4.3 shows the cost violation ratio (CVR) for the three different parameters (network size, terminals set size and spanning factor). The vertical bars on each point were used to represent MAX_CVR (the top of the bars represents its value). On the y axis, the value 0 indicates that at that point no violation for any terminal occurred. For this metric, it is obvious that SPT will always exhibit the best behaviour (all of its points are on the line $y = 0$), as it finds shortest paths. The closer the value to 1, the lesser the violation. In Figure 4.3a, the CVR's value for APPROX is quite close to 1 for the first 2 points, and no violation occurs for the others. The CVR's value for SIM is greater than APPROX, but it is still low (1.4 in average) and it is uniform. In Figures 4.3b and 4.3c, the CVR's behaviour is similar to the one presented in Figure 4.3a for both proposed algorithms. Considering APPROX's values, in almost half of its abscissa's values no violation occurs and in the other points the CVR's value is close to 1 (except for

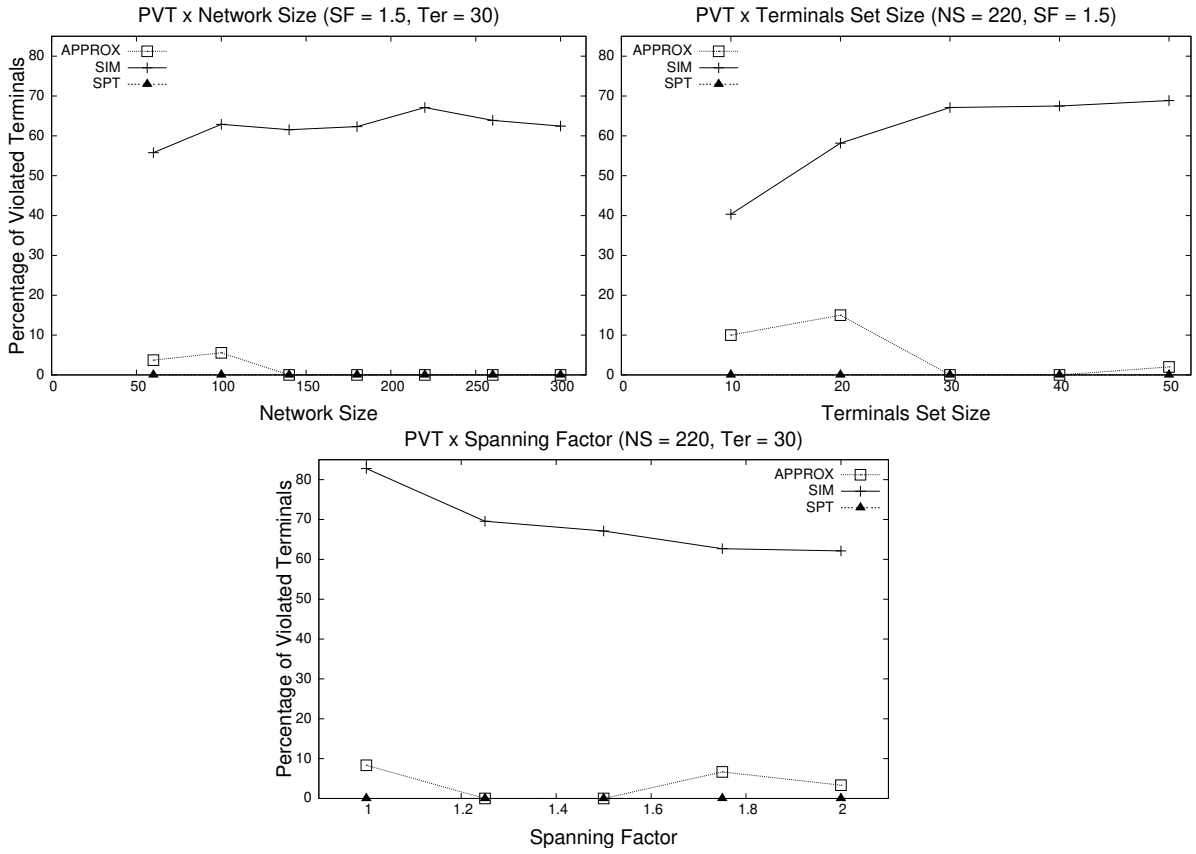

Figure 4.4 Percentage of Violated Terminals in function of (a) network size, (b) number of terminals, and (c) spanning factor

one point in each graphic). Regarding SIM's values, the CVR is again low (between 1.4 and 1.5 in average) and uniform. In the graphics, the behaviour of MAX_CVR for the algorithms is repeated. Regarding the APPROX algorithm, MAX_CVR is really close to CVR (except for one point in Figure 4.3b) and, in some cases, they are almost the same, which it is good because APPROX's worst case is close to the average case. Concerning SIM, the MAX_CVR is slightly higher than CVR, but it is in almost all situations ≤ 2, which is quite acceptable.

CVR measures the quality of violation, but it is important to calculate the amount of violation. PVT captures this concept. More specifically, PVT answers the question: when violation occurs, how many terminals violate their constraint? The values of PVT are shown in Figure 4.4. For all three graphics, the PVT's value is 0 for the same points where no violation occurs in Figure 4.3. In Figure 4.4a, the PVT's value for APPROX is quite low (less than 10%). In fact, the behaviour of PVT for APPROX is similar for all the three graphics, where the value exhibited is generally less than 10%. Concerning SIM, the PVT's value is high but it is uniform for the network size (between 60% and 70%). For SIM, there is a tendency to increase the PVT value with the increse in the terminals set size. On the other hand, when we vary the spanning factor, the higher the spanning factor, the lower the PVT's value for SIM.

Finally, we evaluated the percentage of scenarios where (at least one) violation occurs (PVR). The graphic of PVR is represented by Figure 4.5. For APPROX, the denser the network, the lower PVR is. Additionally, the percentage is no greater than 30%. Even though this value could be considered, in some senses, to be not so low, recall that PVR disregards the percentage of terminals which violate the spanner constraint (the run is considered violated even if the constraint is violated for a single terminal) and the amount of violation. The latter is captured by the CVR metric. A variation of the former is captured by PVT metric. For SIM, PVR has the highest possible value.

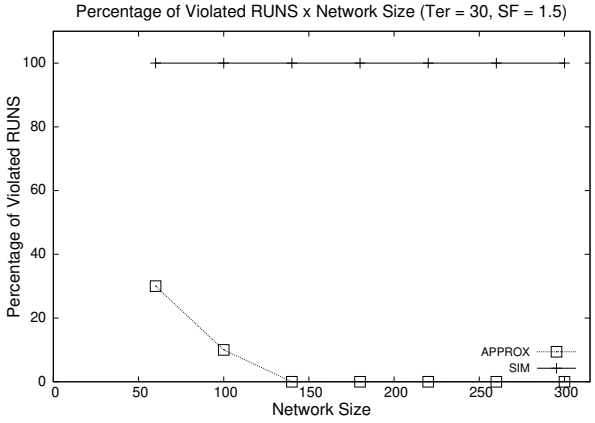

Figure 4.5 Percentage of Violated RUNS x Network Size

According to our experiments, the maximum out-degree for the proposed algorithms was quite low, except for APPROX in the worst scenario when the spanning factor was set to 1 (Figure 4.1c). In this scenario, APPROX mimicked SPT's behaviour. SIM had uniform behaviour at large, supporting better scalability, and always outperformed APPROX in the degree metric. On the other hand, APPROX always outperformed SIM for the CVR, PVT and PVR metrics. More than that, APPROX presented very good results for both the quality (CVR) and the quantity (PVT and PVR) of violation. In almost half of abscissa's values (in some situations, more than half) no violation occurred. Regarding the quality of violation, when it occurred, the violation was quite low, even in worst cases. The same happened for the quantity of violation in the case of APPROX. Considering SIM, although the values were not as so good as for APPROX, especially for the quantity metrics, SIM presented very good results for the metric that measures the quality of violation, which is the one that best summarizes the analysis of the spanner property. For the CVR metric, the value was quite low (in average 1.4) and even in the worst case it was low (in most cases limited by 2). Furthermore, the values achieved by CVR exhibited an uniform behaviour in general.

In the next chapter, we discuss extensively the related work, presenting the main results. Moreover, we argue why DSMDStP is a new problem, giving the differences between our problem and the related ones.

Chapter 5

RELATED WORK

In this chapter we present an extended up-to-date range of related work, from network design problems that address single properties to, similar to our case, problems that address multiple properties. We start by motivating for Steiner trees and spanners. Then, we discuss the related work addressing the properties of interest (spanner and degree) in Euclidean Space (Section 5.1) as well as in (undirected and directed) graphs (Section 5.2). Finally, we explain why DSMDStP is a new problem, giving its differences from related work (Section 5.3).

Steiner trees are typically related to optimizing multicasting (for example, (SUN; LANGENDÖRFER, 1995; RAGHAVAN; MANIMARAN; MURTHY, 1998; FENG; YUM, 1999; PARSA; ZHU; GARCIA-LUNA-ACEVES, 1998; ZHENGYING; BINGXIN; ERDUN, 2001; WANG; LIU; SHI, 2009; WANG; LU; SUN, 2005; CHEN; YANG; XU, 2004)). In classical Steiner tree problems, the goal is to minimize the cost of the tree while satisfying certain constraints.

Spanners are applied in many scenarios. For instance, spanners can be used to support efficient routing. Spanners allow routing protocols to store smaller routing tables while packets' headers are still small, since spanners allow a tradeoff between packets' headers and routing table (when packets are routed through shortest paths, each node has to store a complete routing table or the packet's header has to contain a complete description of the shortest path; spanners allow the tradeoff mentioned before as they use paths with small stretch factors rather than routing through shortest paths) (THORUP; ZWICK, 2001). (Geometric) spanners are also applied in wireless networks to decrease energy expenditure (SCHINDELHAUER; VOLBERT; ZIEGLER, 2007) (for other applications of geometric spanners, see (NARASIMHAN; SMID, 2007)). Spanners are also commonly used to approximate shortest path distances (FEIGENBAUM et al., 2008). Since calculating shortest paths is common for classical shortest path algorithms, approximating distances is of great utility (FEIGENBAUM et al., 2008). Motivation for the adoption of spanners in shortest path algorithms can also be seen in (ELKIN, 2001).

In a similar way to this last application of spanners, they are also applied in distance oracles (BASWANA; SEN, 2006; THORUP; ZWICK, 2005). In (BASWANA;

SEN, 2006), the authors state that in many applications the objective is not to compute all distances (the same as shortest paths) but to be able to retrieve distance values in an efficient way. This can be done through some kind of preprocessing of the input graph. Due to the complexity of the traditional shortest path algorithms, researchers have been trying to find out structures which report approximate distance values instead of the exact values. This motivates the works on *t-approximate distance oracles*, where for a pair of vertices (u, v) in the input graph G, the value returned by the oracle is $\geq dist(u, v, G)$ and $\leq t \cdot dist(u, v, G)$.

Spanner tree, a restricted kind of spanner, is of great importance too. According to (LIEBCHEN; WüNSCH, 2008), spanner trees are used in telecommunications since they allow routing protocols to be simpler. They are also used as a model for broadcast (PELEG, 2000) and, in similar way to Steiner trees, spanners trees can be used in message multicasting. In the literature, there are also references to their application in solving some kinds of linear systems of equations problems (ELKIN et al., 2005) and in finding approximation solutions to the bandwidth minimization problem (VENKATESAN et al., 1997).

The Steiner tree problems addressed in (RAGHAVAN; MANIMARAN; MURTHY, 1998; FENG; YUM, 1999; SUN; LANGENDÖRFER, 1995; PARSA; ZHU; GARCIA-LUNA-ACEVES, 1998) involve a single constraint, a cost bound on the paths between a source node and the terminals (to model maximum transmission delay). In (CHEN; YANG; XU, 2004) the authors take into consideration both delay and degree constraints. In (CHEN; YANG; XU, 2004), the authors assume undirected graph and do not guarantee that the delay constraints will be respected. Additionally, the problem involves a single delay constraint, which applies to a subset of nodes. Degree constraints arise in scenarios in communication networks to address resource limitations in routers and switches and to balance load during multicast operations.

5.1 WORKS THAT CONSIDER SPANNER AND DEGREE IN EUCLIDEAN SPACE

Much previous work has addressed network design problems with more than one property (including being a tree, the spanner property, bounded maximum degree, and others) as we do (ARYA et al., 1995; DINITZ; ELKIN; SOLOMON, 2008; KORTSARZ; PELEG, 1999). All these mentioned works assume an Euclidean space, which restricts the applicability of their work to scenarios where geometric relations hold as well as associating a different cost based on the direction of the edge is not allowed, as directed graphs are not allowed to be modelled. Other works address degree too (CHAN, 2003; FEKETE et al., 1997; MONMA; SURI, 1991; LUKOVSZKI, 1999; FARSHI; GUDMUNDSSON, 2007; LUKOVSZKI; SCHINDELHAUER; VOLBERT, 2006; GRüNEWALD et al., 2002). But they also assume an Euclidean Space.

5.2 WORKS THAT CONSIDER SPANNER AND DEGREE IN (DIRECTED AND UNDIRECTED) GRAPHS

More spanner and degree problems on graphs are addressed in (FOMIN; GOLOVACH; LEEUWEN, 2011; DINITZ; KRAUTHGAMER, 2011; BERMAN et al., 2011; ELKIN; SOLOMON, 2011b; HAJIAGHAYI; KORTSARZ; SALAVATIPOUR, 2009; NAOR; SCHIEBER, 1997) and (KHANDEKAR; KORTSARZ; NUTOV, 2011; GOEMANS, 2006; SINGH; LAU, 2007; ELKIN; KORTSARZ, 2006; BANSAL; KHANDEKAR; NAGARAJAN, 2009; NUTOV, 2011; FEDER; MOTWANI; ZHU, 2006; RAVI; RAGHAVACHARI; KLEIN, 1992) respectively. These work assume undirected graphs, which is a restricted case of our more general assumption of directed graphs.

On the other hand, the authors in (DINITZ; KRAUTHGAMER, 2011; NAOR; SCHIEBER, 1997; BERMAN et al., 2011) deal with related problems in directed graphs. Their works differ from ours as the resulted graph is not a tree or the authors aim at minimizing a different function (e.g., the number of edges, the total cost of the tree). The works in (BANSAL; KHANDEKAR; NAGARAJAN, 2009; NUTOV, 2011) address tree problems with degree restrictions in directed graphs too. They are interested in guarateeing k-connectivity. In these works, the authors aim to satisfy the degree restriction imposed to each node rather than minimize the maximum degree. Additionally, they do not consider the spanner property. The problems addressed in (FEDER; MOTWANI; ZHU, 2006) are simitar to the ones addressed in the former papers. The authors consider the problem of minimizing the maximum degree but they do not consider the spanner property neither.

5.3 OUR PROBLEM: THE NOVELTY IN DSMDSTP

Our work differs from related work as in our case we consider directed graphs instead of working in metric spaces or assuming undirected graphs. Additionally, we aim to find Steiner trees, a more general case of spanning trees. Moreover, besides building a Steiner tree with bounded maximum degree in directed graphs, we also address the k-spanner property, more specifically, single-sink k-spanner trees, where k is a parameter of the problem (instead of a bound achieved by a solution to the problem).

The closest works to ours are (ELKIN; SOLOMON, 2009, 2011a; ELKIN; KORTSARZ, 2006; KHANDEKAR; KORTSARZ; NUTOV, 2011). In (ELKIN; SOLOMON, 2009, 2011a), the authors address the problem of building narrow-shallow-low-light trees. The authors thus address the problem of generating trees with parameterized root-stretch factor and bounded maximum degree, as we do, but they consider two additional properties (generation of *low* and *light* trees). However, their work builds a spanning tree rather than a Steiner tree and they consider metric spaces. Our assumptions are more general, as we do not restrict our problem to metric spaces. Additionally, the authors' work provides a constant root-stretch factor, similarly to other works where shallow trees are generated, rather than supporting a parameterized stretch factor. The authors in (ELKIN; KORTSARZ, 2006; KHANDEKAR; KORTSARZ; NUTOV, 2011) address the problem of minimizing the out-degree of directed graphs in Steiner tree problems. In (ELKIN; KORTSARZ, 2006), similar to our work, the problem consists in covering all

the set T of terminals, while in (KHANDEKAR; KORTSARZ; NUTOV, 2011) the problem is generalized as the objective is to cover p terminals, where $p \leq |T|$. In both works, the authors do not consider the spanner property. They also address problems somewhat similar to ours, where besides minimizing the maximum degree it is required to limit the height of the tree. So, they are interested in limiting the number of hops, which differs from our spanner property. To the best of our knowledge, our work is the first attempt to address the problem of building a Steiner tree in directed graphs with limited maximum degree and with a parameterized root-stretch factor.

5.3.1 Limitation in the Results

Our algorithms are not able to guarantee the root-stretch factor k for all paths from the source node to the terminals. Our approximation algorithm and heuristic give a root-stretch factor of $k \cdot \left(1 + \frac{max_{t \in T}\{dist(s,t,G)\}}{min_{t \in T}\{dist(s,t,G)\}}\right)$ and $k \cdot (\lfloor \sqrt{|T|} \rfloor + 2)$ respectively. However, as argued in the introduction, in the experiments the cost of the generated paths is much lower than the provided upper bounds.

5.3.2 The Base Algorithmic Tool for the Proposed Solutions: The MSC Problem

Our algorithms are based on the algorithm presented in (ELKIN; KORTSARZ, 2006). In (ELKIN; KORTSARZ, 2006), the goal is to compute a schedule with a minimal number of rounds that delivers a message from a given node to all the terminals. The problem is called *Directed Telephone Multicast Problem*. Our approximation algorithm follows the same steps as in (ELKIN; KORTSARZ, 2006), but we use different criteria for defining some of the used concepts, such as $\sqrt{k}$-bad nodes, and for the construction of the instance of the Multiple Set-Cover (MSC) problem. The bound on the degree is obtained in exactly the same way as in (ELKIN; KORTSARZ, 2006). The heuristic SIM deviates from the algorithm in (ELKIN; KORTSARZ, 2006), as it is based on an iterative application of the MSC problem (instead of applying this problem only once, as in our algorithm and in (ELKIN; KORTSARZ, 2006)).

The Multiple Set-Cover (MSC) problem was stated for the first time in (ELKIN; KORTSARZ, 2003). A solution for this problem was presented in (CHEKURI; KUMAR, 2004). In (CHEKURI; KUMAR, 2004), MSC is called *Set Cover with Group Budgets* (SCG). The MSC problem can be generalized by the problem called *Maximization of Monotone Submodular Function subject to Matroids Constraint*, which was addressed in (CĂLINESCU et al., 2011). However, our goal in this dissertation is to present a deterministic algorithm. We will discuss this in greater detail in Appendix A.

In the next chapter, we conclude the dissertation and present future work.

Chapter 6

CONCLUSION

In this dissertation we presented a version of a Steiner tree problem called *Directed k-Spanner with Minimum Degree Steiner Tree Problem* (DSMDStP). Unlike commonly defined Steiner tree problems, in DSMDStP we are interested in minimizing the maximum out-degree of the arborescence while respecting the terminals spanner constraints. We assume as input a directed graph. To the best of our knowledge, these properties characterize DSMDStP as a new problem. We showed that DSMDStP is not approximable sublogarithmically (unless $NP \subset DTIME(n^{\log\log n})$) and described an approximation algorithm and a heuristic to it. For both proposed algorithms, we analyzed their complexity.

The approximation algorithm is based on the algorithm described in (ELKIN; KORTSARZ, 2006). The algorithm generates an arborescence $\mathcal{A}_f$ with maximum degree $2\sqrt{|T|} + 2 + O(\log |T|) \cdot d^*$, where d^* is the maximum degree of an optimum arborescence. This seems a reasonable limit, as DSMDStP does not admit a sublogarithmic approximation. Additionally, in $\mathcal{A}_f$ the paths from s to each terminal t have cost $\leq k \cdot (dist(s,t,G) + dist(s,t_{max},G))$, where $t_{max} \in \{t' | t' \in T \wedge (\forall t'' \in T : dist(s,t'',G) \leq dist(s,t',G))\}$. Although the algorithm can violate the spanner constraint, this only happens to terminals covered in the second phase of the algorithm.

The heuristic, called SIM, is based on an iterative application of an algorithm for the MSC problem. SIM does not provide guarantee on maximum node degree. However, in our experiments SIM provided lower maximum node out-degree than the approximation algorithm. In the final arborescence, the path from s to $t \in T$ has cost $\leq (|T| + 2) \cdot k \cdot dist(s,t,G)$.

The experiments presented quite good results for both proposed algorithms. Regarding the node out-degrees, both the approximation algorithm and SIM yielded low maximum out-degree and they outperformed the results of a shortest path tree algorithm (SPT). Additionally, SIM has always outperformed the approximation algorithm and it has a uniform behaviour, what contributes to scalability. On the other hand, the approximation algorithm always outperformed SIM in metrics concerning the spanner

constraint. Moreover, in half of the situations no violation occurred. The metrics addressing the spanner constraint measured the quality and the quantity of violation. Even when violation occurred, these metrics showed that the approximation algorithm violated by low factors. Although SIM did not present as good results as the approximation algorithm, concerning the metric that addresses the quality of violation (how much the violation occurred), the results were also quite good, since on average the violation was by a factor of 1.4 and in worst case by a factor of 2, which are quite acceptable.

We also described how we can improve (with high probability) the degree guarantee for the approximation algorithm. This can be done through the concepts of *submodular functions* and *matroid*. More specifically, we can solve DSMDStP using an instance of a problem called MCG. The improved result can be achieved by modelling MCG as a problem of *maximizing a monotone submodular function subject to partition matroid constraint* and using the algorithm presented in (CĂLINESCU et al., 2011) to solve it.

6.1 FUTURE WORK

A possible future work could be to investigate new algorithms to try to improve the resutls, mainly concerning the spanner property. It is important to mention that we do not know if DSMDStP admits a solution.

Turning the solutions into distributed ones would be challenging and another possible future work. Distributed algorithms are necessary in some scenarios, such as in Wireless Sensor Networks. Besides this, a distributed solution is more scalable than a centralized one, which is an important property.

We argued in Appendix A how we could improve the degree guarantee of the proposed approximation algorithm by modelling the problem (actually, the subproblem MCG) through the concepts of submodular functions and matroids. But doing this, our algorithm would be a probabilistic one. Although being probabilistic, it would be interesting to see by how much the degree is improved in practice. So, another possible future work would be to solve the MCG problem through the solution proposed in (CĂLINESCU et al., 2011) and compare the results with the ones presented in this dissertation.

BIBLIOGRAPHY

ARORA, S.; BARAK, B. *Computational Complexity: A Modern Approach*. 1st. ed. New York, NY, USA: Cambridge University Press, 2009.

ARYA, S. et al. Euclidean spanners: short, thin, and lanky. In: *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1995. (STOC '95), p. 489–498. ISBN 0-89791-718-9.

BANSAL, N.; KHANDEKAR, R.; NAGARAJAN, V. Additive guarantees for degree-bounded directed network design. *SIAM J. Comput.*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 39, n. 4, p. 1413–1431, out. 2009. ISSN 0097-5397.

BASWANA, S.; SEN, S. Approximate distance oracles for unweighted graphs in expected o(n2) time. *ACM Trans. Algorithms*, ACM, New York, NY, USA, v. 2, n. 4, p. 557–577, out. 2006. ISSN 1549-6325.

BERMAN, P. et al. Improved approximation for the directed spanner problem. In: ACETO, L.; HENZINGER, M.; SGALL, J. (Ed.). *Automata, Languages and Programming*. [S.l.]: Springer Berlin-Heidelberg, 2011, (Lecture Notes in Computer Science, v. 6755). p. 1–12. ISBN 978-3-642-22005-0.

BRAGA, H. *Implementation of the algorithms for DSMDStP*. 2012. Website. Url = http://gdurl.com/1Qcp/download.

CĂLINESCU, G. et al. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, v. 40, n. 6, p. 1740–1766, 2011.

CHAN, T. M. Euclidean bounded-degree spanning tree ratios. In: *Proceedings of the nineteenth annual symposium on Computational geometry*. New York, NY, USA: ACM, 2003. (SCG '03), p. 11–19. ISBN 1-58113-663-3.

CHEKURI, C.; KUMAR, A. Maximum coverage problem with group budget constraints and applications. In: *APPROX-RANDOM*. [S.l.]: Springer-Verlag New York, Inc., 2004. p. 72–83.

CHEN, L.; YANG, Z.; XU, Z. A degree-delay-constrained genetic algorithm for multicast routing tree. In: *Computer and Information Technology, 2004. CIT '04. The Fourth International Conference on*. [S.l.: s.n.], 2004. p. 1033–1038.

CORMEN, T. H. et al. *Introduction to Algorithms*. 3. ed. [S.l.]: The MIT Press, 2009.

DINITZ, M.; KRAUTHGAMER, R. Directed spanners via flow-based linear programs. In: *Proceedings of the 43rd annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2011. (STOC '11), p. 323–332. ISBN 978-1-4503-0691-1.

DINITZ, Y.; ELKIN, M.; SOLOMON, S. Shallow-low-light trees, and tight lower bounds for euclidean spanners. In: *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2008. (FOCS '08), p. 519–528. ISBN 978-0-7695-3436-7.

ELKIN, M. Computing almost shortest paths. In: *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2001. (PODC '01), p. 53–62. ISBN 1-58113-383-9.

ELKIN, M.; KORTSARZ, G. Approximation algorithm for directed telephone multicast problem. In: *ICALP'03*. Berlin, Heidelberg: Springer-Verlag, 2003. p. 212–223. ISBN 3-540-40493-7.

ELKIN, M.; KORTSARZ, G. An approximation algorithm for the directed telephone multicast problem. *Algorithmica*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 45, p. 569–583, August 2006. ISSN 0178-4617.

ELKIN, M.; SOLOMON, S. Narrow-shallow-low-light trees with and without steiner points. In: *ESA*. [S.l.: s.n.], 2009. p. 215–226.

ELKIN, M.; SOLOMON, S. Narrow-shallow-low-light trees with and without steiner points. *SIAM J. Discret. Math.*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 25, n. 1, p. 181–210, fev. 2011. ISSN 0895-4801.

ELKIN, M.; SOLOMON, S. Steiner shallow-light trees are exponentially lighter than spanning ones. In: *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2011. (FOCS '11), p. 373–382. ISBN 978-0-7695-4571-4.

ELKIN, M. et al. Lower-stretch spanning trees. In: *STOC'05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. [S.l.]: ACM Press, 2005. p. 494–503.

FARSHI, M.; GUDMUNDSSON, J. Experimental study of geometric t-spanners: a running time comparison. In: *Proceedings of the 6th international conference on Experimental algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007. (WEA'07), p. 270–284. ISBN 978-3-540-72844-3.

FEDER, F.; MOTWANI, R.; ZHU, A. *k-Connected Spanning Subgraphs of Low Degree*. [S.l.], 2006.

FEIGENBAUM, J. et al. Graph distances in the data-stream model. *SIAM J. Comput.*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 38, n. 5, p. 1709–1727, dez. 2008. ISSN 0097-5397.

FEKETE, S. P. et al. A network-flow technique for finding low-weight bounded-degree spanning trees. *Journal of Algorithms*, v. 24, n. 2, p. 310 – 324, 1997. ISSN 0196-6774.

FENG, G.; YUM, T.-S. P. Efficient multicast routing with delay constraints. *International Journal of Communication Systems*, John Wiley & Sons, Ltd., v. 12, n. 3, p. 181–195, 1999. ISSN 1099-1131.

FIBONACCI, L.; SIGLER, L. *The book of squares*. [S.l.]: Academic Press, 1987. ISBN 9780126431308.

FISHER, M. L.; NEMHAUSER, G. L.; WOLSEY, L. A. An analysis of approximations for maximizing submodular set functions-ii. In: *Polyhedral Combinatorics*. [S.l.]: Springer Berlin Heidelberg, 1978, (Mathematical Programming Studies, v. 8). p. 73–87.

FOMIN, F. V.; GOLOVACH, P. A.; LEEUWEN, E. J. van. Spanners of bounded degree graphs. *Inf. Process. Lett.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 111, n. 3, p. 142–144, jan. 2011. ISSN 0020-0190.

FRAIGNIAUD, P. Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode. In: *Proceedings of the 9th Annual European Symposium on Algorithms*. London, UK: Springer-Verlag, 2001. (ESA '01), p. 440–451. ISBN 3-540-42493-8.

GOEMANS, M. X. Minimum bounded degree spanning trees. In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2006. (FOCS '06), p. 273–282. ISBN 0-7695-2720-5.

GRüNEWALD, M. et al. Distributed maintenance of resource efficient wireless network topologies (extended abstract). In: *European Conference On Parallel Computing (EUROPAR*. [S.l.: s.n.], 2002. p. 935–946.

HAJIAGHAYI, M. T.; KORTSARZ, G.; SALAVATIPOUR, M. R. Approximating buy-at-bulk and shallow-light k-steiner trees. *Algorithmica*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 53, n. 1, p. 89–103, jan. 2009. ISSN 0178-4617.

KHANDEKAR, R.; KORTSARZ, G.; NUTOV, Z. Network-design with degree constraints. In: GOLDBERG, L. et al. (Ed.). *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. [S.l.]: Springer Berlin, 2011, (Lecture Notes in Computer Science, v. 6845). p. 289–301. ISBN 978-3-642-22934-3.

KORTSARZ, G.; PELEG, D. Approximating the weight of shallow steiner trees. *Discrete Appl. Math.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 93, n. 2-3, p. 265–285, jul. 1999. ISSN 0166-218X.

LIEBCHEN, C.; WüNSCH, G. The zoo of tree spanner problems. *Discrete Appl. Math.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 156, n. 5, p. 569–587, mar. 2008. ISSN 0166-218X.

LUKOVSZKI, T. New results of fault tolerant geometric spanners. In: *Proceedings of the 6th International Workshop on Algorithms and Data Structures*. London, UK, UK: Springer-Verlag, 1999. (WADS '99), p. 193–204. ISBN 3-540-66279-0.

LUKOVSZKI, T.; SCHINDELHAUER, C.; VOLBERT, K. Resource efficient maintenance of wireless network topologies. *j-jucs*, v. 12, n. 9, p. 1292–1311, sep 2006.

MONMA, C.; SURI, S. Transitions in geometric minimum spanning trees (extended abstract). In: *Proceedings of the seventh annual symposium on Computational geometry*. New York, NY, USA: ACM, 1991. (SCG '91), p. 239–249. ISBN 0-89791-426-0.

NAOR, J.; SCHIEBER, B. Improved approximations for shallow-light spanning trees. In: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1997. (FOCS '97), p. 536–. ISBN 0-8186-8197-7.

NARASIMHAN, G.; SMID, M. *Geometric Spanner Networks*. [S.l.]: Cambridge University Press, 2007.

NEMHAUSER, G. L.; WOLSEY, L. A.; FISHER, M. L. An analysis of approximations for maximizing submodular set functions-i. *Mathematical Programming*, Springer Berlin / Heidelberg, v. 14, p. 265–294, 1978.

NGUYEN, U. T.; NGUYEN, H. L. *Multicast Routing Algorithms*. [S.l.], 2008.

NUTOV, Z. Approximating directed weighted-degree constrained networks. *Theor. Comput. Sci.*, Elsevier Science Publishers Ltd., Essex, UK, v. 412, n. 8-10, p. 901–912, mar. 2011. ISSN 0304-3975.

OLIVEIRA, C. A. S.; PARDALOS, P. M. A survey of combinatorial optimization problems in multicast routing. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 32, p. 1953–1981, August 2005. ISSN 0305-0548.

PARSA, M.; ZHU, Q.; GARCIA-LUNA-ACEVES, J. An iterative algorithm for delay-constrained minimum-cost multicasting. *Networking, IEEE/ACM Transactions on*, v. 6, n. 4, p. 461–474, 1998. ISSN 1063-6692.

PELEG, D. *Distributed computing: a locality-sensitive approach*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. ISBN 0-89871-464-8.

RAGHAVAN, S.; MANIMARAN, G.; MURTHY, C. S. R. Algorithms for delay-constrained low-cost multicast tree construction. *Comp. Communications*, v. 21, p. 1693–1706, 1998.

RAVI, R.; RAGHAVACHARI, B.; KLEIN, P. N. Approximation through local optimality: Designing networks with small degree. In: *Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science*. London, UK, UK: Springer-Verlag, 1992. p. 279–290. ISBN 3-540-56287-7.

SCHINDELHAUER, C.; VOLBERT, K.; ZIEGLER, M. Geometric spanners with applications in wireless networks. *Comput. Geom. Theory Appl.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 36, n. 3, p. 197–214, abr. 2007. ISSN 0925-7721.

SCHRIJVER, A. *Combinatorial Optimization - Polyhedra and Efficiency*. [S.l.]: Springer, 2003.

SINGH, M.; LAU, L. C. Approximating minimum bounded degree spanning trees to within one of optimal. In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2007. (STOC '07), p. 661–670. ISBN 978-1-59593-631-8.

SUN, Q.; LANGENDÖRFER, H. Efficient multicast routing for delay-sensitive applications. In: *Proc. Second Workshop on Protocols for Multimedia Systems (PROMS*. [S.l.: s.n.], 1995. p. 452–458.

THORUP, M.; ZWICK, U. Compact routing schemes. In: *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*. New York, NY, USA: ACM, 2001. (SPAA '01), p. 1–10. ISBN 1-58113-409-6.

THORUP, M.; ZWICK, U. Approximate distance oracles. *J. ACM*, ACM, New York, NY, USA, v. 52, n. 1, p. 1–24, jan. 2005. ISSN 0004-5411.

VENKATESAN, G. et al. Restrictions of minimum spanner problems. *Inf. Comput.*, Academic Press, Inc., Duluth, MN, USA, v. 136, n. 2, p. 143–164, ago. 1997. ISSN 0890-5401.

WANG, L.; LIU, W.; SHI, H. Delay-constrained multicast routing using the noisy chaotic neural networks. *Computers, IEEE Transactions on*, v. 58, n. 1, p. 82–89, 2009. ISSN 0018-9340.

WANG, Y.; LU, Y.; SUN, Y. A degree-constrained multicast routing algorithm for multimedia communications. In: *Proceedings of the SPIE, Network Architectures, Management, and Applications III*. [S.l.: s.n.], 2005. v. 6022, p. 1023–1031.

WILLIAMSON, D. P.; SHMOYS, D. B. *The Design of Approximation Algorithms*. 1st. ed. New York, NY, USA: Cambridge University Press, 2011.

ZHENGYING, W.; BINGXIN, S.; ERDUN, Z. Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. *Computer Communications*, p. 685–692, 2001.

Appendix

A

DISCUSSION ABOUT HOW TO APPROACH DSMDSTP THROUGH AN ALTERNATIVE WAY

In this appendix we comment on how we could approach the MCG problem through an alternative way and the impact of this new approaching for the DSMDStP. This new modeling of MCG would allow us to decrease the maximum degree for the DSMDStP. In order to do this, we would model the MCG problem through the concepts of *submodular functions* and *matroid*. We briefly present these concepts and show how MCG could be modeled through them. Although there is a result that allows the improvement of the degree guarantee of our problem, this would turn our algorithm in a probabilistic one, as the presented result for this new modeling of MCG is a randomized algorithm. The results discussed here are a mere conjecture.

As mentioned in (CĂLINESCU et al., 2011), the MCG problem (CHEKURI; KUMAR, 2004) could be addressed through the concepts of *monotone submodular functions* and *matroids*. More specifically, MCG can be modeled as a problem of maximizing a monotone submodular function under matroid constraint. As we can solve MSC using MCG, improvements on solution to MCG extend to MSC as well (ELKIN; KORTSARZ, 2006) (the SCG problem in (CHEKURI; KUMAR, 2004)). Let X be a ground set of n elements. A function $f : 2^X \rightarrow \mathbb{R}^+$ is *submodular* iff:

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B),$$

for all $A, B \subseteq X$ (SCHRIJVER, 2003). 2^X represents the set of all subsets of X, including the empty set and X itself. f is called *monotone* if $f(A) \leq f(B)$, for all $A \subseteq B$. In order to address the *matroid* concept, besides a ground set X we need the concept of an *independence family* $I \subseteq 2^X$, a family of subsets that is downward closed, which means that $A \in I$ and $B \subseteq A$ implies that $B \in I$. A *matroid* is a pair $M = (X, I)$ where $I \subseteq 2^X$ and

$$\forall B \in I, A \subset B \Rightarrow A \in I. \quad \text{(i)}$$

$$\forall A, B \in I; |A| < |B| \Rightarrow \exists x \in B \setminus A; A \cup x \in I. \quad \text{(ii)}$$

The problem addressed in (CĂLINESCU et al., 2011) consists in maximizing $f(S)$ over the independent sets $S \in I$. Similar to (CĂLINESCU et al., 2011), let us denote this kind of problem by *SUB-M*. The SUB-M problem can be modeled with a matroid constraint called *partition matroid*, where X is partitioned into l subsets $X_1, X_2, ..., X_l$ with associated integers $k_1, k_2, ..., k_l$, and $A \subseteq X$ is considered independent iff $|A \cap X_i| \leq k_i$.

Maximizing a monotone submodular function under matroid constraint (SUB-M) was addressed for the first time in (NEMHAUSER; WOLSEY; FISHER, 1978; FISHER; NEMHAUSER; WOLSEY, 1978). In these papers, the authors analyze a greedy algorithm and give some approximation results. For SUB-M with partition matroid, the greedy algorithm gives a 2-approximation (FISHER; NEMHAUSER; WOLSEY, 1978). For the restricted case of SUB-M with uniform matroid ($l = 1$, so the objective is maximizing $f(S) : |S| \leq k$), the algorithm gives a $(e/(e-1))$-approximation. The authors in (CĂLINESCU et al., 2011) improved the previous results to an approximation factor of $(e/(e-1))$ for any matroid, including the partition matroid, and this is optimal in the oracle model (since this approximation ratio is the best one for the restricted version, the one with uniform matroid (NEMHAUSER; WOLSEY; FISHER, 1978; FISHER; NEMHAUSER; WOLSEY, 1978)). The authors in (CĂLINESCU et al., 2011) improved the previous results through a randomized algorithm.

The MCG problem can be modeled by SUB-M with partition matroid. Regarding MCG, a solution $H \subset \{S_1, S_2, ..., S_m\}$ is independent iff $|H \cap G_i| \leq k_i$ (see problem definition in Section 3.3). Remember that there is a set of partitions $G_1, G_2, ..., G_l$ as input to the MCG problem. For this problem, $f : H \subset \{S_1, S_2, ..., S_m\} \rightarrow \mathbb{R}^+$ represents the number of elements of the ground set X covered by the elements of H. This kind of coverage function is submodular, as mentioned in (CĂLINESCU et al., 2011). Based on Theorem 1.1 in (CĂLINESCU et al., 2011) and the previous discussion, the following result holds:

Theorem A.1. *(CĂLINESCU et al., 2011). There is a randomized algorithm which gives an $(e/(e-1))$-approximation [1] (in expectation) to the MCG problem.*

The expression *in expectation* in theorem A.1 means the approximation factor is guaranteed with high probability. Based on the idea proposed in (CHEKURI; KUMAR, 2004) to solve the MSC problem (the same as the SCG problem in (CHEKURI; KUMAR, 2004)), we can infer that:

Lemma A.1. *There is an algorithm which gives a $(\log_e n + 1)$-approximation (in expectation) to the MSC problem.*

So, based on lemma A.1, theorem A.1 and theorem 3.2, we cojecture that *there is an approximation algorithm that generates an arborescence $\mathcal{A}_f$ with (expected) bounded out-degree $2\sqrt{k} + 2 + O(\log_e l) \cdot d^*$ and that, with high probability, has paths from s to each terminal $t \in T$ with cost less than or equal to $k \cdot (dist(s, t, G) + dist(s, t_{max}, G))$.*

[1] Our notation of the approximation factor is different from the one presented in (CĂLINESCU et al., 2011), since the latter is represented by $\frac{1}{f}$, where f is the approximation factor in our case.

As the algorithm proposed in (CĂLINESCU et al., 2011) guarantees that the approximation factor holds with high probability, the bounded degree of the aforementioned approximation algorithm is guaranteed in expectation, since it depends upon the bound provided by the algorithm to the MCG problem, what is probabilistic in this case, so the expectation follows. Moreover, as the authors in (CHEKURI; KUMAR, 2004) calculate the number of iterative runs of the MCG algorithm (and consequently the upper bound to the MSC) for the solution to the MSC problem based on the approximation factor of the algorithm to MCG, by applying the probabilistic solution presented in this chapter, the $(\log_e l + 1)$ is an expected upper bound of iterations to cover all terminals.

Although the MCG problem could be modeled as SUB-M with partition matroid and it could be solved by a deterministic algorithm with approximation ratio of 2 (the one analyzed in (NEMHAUSER; WOLSEY; FISHER, 1978; FISHER; NEMHAUSER; WOLSEY, 1978)), which was the first result, this deterministic algorithm gives the same approximation factor of the algorithm presented in (CHEKURI; KUMAR, 2004).