Node-Weighted Prize Collecting Steiner Tree and Applications

by

Sina Sadeghian Sadeghabad

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The Steiner Tree problem has appeared in the Karp's list of the first 21 NP-hard problems and is well known as one of the most fundamental problems in Network Design area. We study the Node-Weighted version of the Prize Collecting Steiner Tree problem. In this problem, we are given a simple graph with a cost and penalty value associated with each node. Our goal is to find a subtree T of the graph minimizing the cost of the nodes in T plus penalty of the nodes not in T. By a reduction from set cover problem it can be easily shown that the problem cannot be approximated in polynomial time within factor of $(1 - o(1)) \ln n$ unless NP has quasi-polynomial time algorithms, where n is the number of vertices of the graph.

Moss and Rabani claimed an $O(\log n)$-approximation algorithm for the problem using a Primal-Dual approach in their STOC'01 paper [81]. We show that their algorithm is incorrect by providing a counter example in which there is an $O(n)$ gap between the dual solution constructed by their algorithm and the optimal solution. Further, evidence is given that their algorithm probably does not have a simple fix. We propose a new algorithm which is more involved and introduces novel ideas in primal dual approach for network design problems. Also, our algorithm is a Lagrangian Multiplier Preserving algorithm and we show how this property can be utilized to design an $O(\log n)$-approximation algorithm for the Node-Weighted Quota Steiner Tree problem using the Lagrangian Relaxation method.

We also show an application of the Node Weighted Quota Steiner Tree problem in designing algorithm with better approximation factor for *Technology Diffusion* problem, a problem proposed by Goldberg and Liu in [48] (SODA 2013). In Technology Diffusion, we are given a graph G and a threshold $\theta(v)$ associated with each vertex v and we are seeking a set of initial nodes called the *seed set*. Technology Diffusion is a dynamic process defined over time in which each vertex is either active or inactive. The vertices in the seed set are initially activated and each other vertex v gets activated whenever there are at least $\theta(v)$ active nodes connected to v through other active nodes. The Technology Diffusion problem asks to find the minimum seed set activating all nodes. Goldberg and Liu gave an $O(rl \log n)$-approximation algorithm for the problem where r and l are the diameter of G and the number of distinct threshold values, respectively. We improve the approximation factor to $O(\min\{r, l\} \log n)$ by establishing a close connection between the problem and the Node Weighted Quota Steiner Tree problem.

Acknowledgements

I would like to express my sincere gratitudes to my supervisors Professor Jochen Könemann and Professor Laura Sanitá for their invaluable guidance, continues support and excellent supervision. I would also like to thank Professor Chaitanya Swamy and Professor Ricardo Fukasawa for spending the time to read this thesis and for their insightful comments.

Table of Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

In the Steiner Tree problem, given a graph $G = (V, E)$ and a subset S of vertices, called the *terminals*, and a cost function $c : E(G) \to \mathbb{R}$, we are looking for the minimum cost subtree T spanning all terminals:

$$\min_{\text{subtree } T \text{ of } G: S \subseteq V(T)} c(T),$$

where $c(T) = \sum_{e \in E(T)} c(e)$. The non-terminal vertices $V \setminus S$ of the graph are referred to as the *Steiner vertices*.

The Steiner Tree problem is one of the most fundamental problems in Network Design and Complexity Theory for its real world application [3, 69, 55, 91, 35] and also due to its theoretical importance. The origin of the Steiner Tree problem goes back to the problem raised and solved by J. Steiner in the beginning of the 19th century: given 3 points in the Euclidean plane, find a minimum length network connecting these points. This problem was generalized to n points in the plane by Jarník and Kössler [67] and named Euclidean Steiner Tree problem and later the edge-weighted network version of the problem in graphs was proposed by Hakimi in 1971 [63] which we know as the classic Steiner Tree problem described above.

The Steiner Tree problem is also the natural generalization of the well known minimum spanning tree problem (where $S = V$) but unlike this problem, most variants of Steiner Tree problem are known to be NP-hard. The Steiner Tree problem itself appeared in Karp's original list of 21 NP complete problems [71], motivating much research on the design of approximation algorithms for it. Here we provide a brief review of different variants of Steiner Tree problem relevant to this thesis. We focus on hardness results and approximation algorithms found for each problem.

1.1 Edge-Weighted Steiner Tree problems

The original Steiner Tree problem, as defined above was defined only with edge costs. There is a simple 2-approximation algorithm for the problem: given the set S of terminals, the algorithm constructs a complete graph H with vertex set S such that the weight of each edge is the length of shortest path between its end points in G. Then, a minimum spanning tree is found in H and converted to a Steiner Tree in original graph by replacing each edge with its corresponding shortest path. The proof of approximation guarantee was given in [43] by Gilbert and Polak in 1968.

In 1990, Zelikovsky showed that we can obtain better than 2 approximation factor for the problem by announcing a 11/6-approximation algorithm [96]. Subsequently, many improvements were gradually attained for the problem; a 1.746-approximation algorithm by Berman and Ramaiyer in 1992 [14] was followed by a 1.693-approximation algorithm again by Zelikovsky in 1996 [95]. Then, a 1.667-approximation algorithm by Prömel and Steger [85, 86] and a 1.644-approximation algorithm by Karpinski and Zelikovsky [72] were proposed in 1997. In 1999 a 1.598-approximation algorithm was designed by Hougardy and Prömel [65] and then improved to $1 + \frac{\ln 3}{2} \approx 1.55$ by Robins and Zelikovsky [88] in 2005.

The best known approximation factor for the problem is currently a $\ln 4 \approx 1.39$-approximation algorithm found by Byrka et al. [22]. There is still a big gap between the bound and the best inapproximability result for the problem which is a lower bound of 96/95 due to Chlebík and Chlebíková [31]. Many of the works mentioned above are inspired by a general Primal Dual framework proposed by Goemans and Williamson for network design problems. Their method is capable of solving a broad range of network design problems with approximation factor of 2 (see [46] or [47] for specific characterization of these problems).

Prize Collecting Steiner Tree Problem (PCST). In the Prize Collecting Steiner Tree problem, we are allowed to violate the requirement of spanning a terminal v by paying a penalty $\pi(v)$. In other words, instead of a set of terminals, we are given penalty values for all vertices and our objective function is to find a subtree minimizing the total cost of selected tree plus penalty of the nodes excluded from the tree. It is easy to see that this is a generalization of the classic problem. We may assume the penalty of the Steiner nodes are zero while the penalty of terminals are infinity to obtain an instance of the classic problem.

The prize collecting Steiner Tree problem was first introduced by Balas [9] and the first approximation algorithm for the problem was found by Beinstock et al. with approximation factor of 3 [18]. Geomans and Williamson gave a 2-approximation algorithm for edge

weighted version of PCST [46, 47] using their general primal dual framework. The LP formulation underlying this algorithm is well known to have an integrality gap of ≈ 2, and hence, their result is tight. The result of Geomans and Williamson was the best approximation factor for PCST for 17 years until in 2009, Archer, Bateni, Hajiaghayi and Karloff discovered a $(2-\epsilon)$-approximation algorithm for the problem [3] breaking through the LP integrality gap barrier.

Quota Steiner Tree Problems. The Quota Steiner Tree (QST) problem is a generalization of the k-Minimum Spanning Tree (k-MST) problem. In k-MST, given a graph and edge costs, we are asked to find the minimum cost subtree of size k. In the Quota Steiner Tree problem, we have vertex profits in addition to edge costs and a quota value Q and we are asked to find a minimum cost subtree with total profit at least Q.

The first constant factor approximation algorithm for k-MST problem was given by Blum, Ravi and Vempala in 1996 [19]. Garg gave a 3-approximation as well as a simple 5-approximation algorithm for k-MST [40], improving the previous bound. Later in 1998, Arya and Ramesh improved the factor to 2.5[6]. Currently, the best known result is a 2-approximation found by Garg [41].

In [69], Johnson, Minkoff and Phillips showed that any α-approximation algorithm for k-MST problem can be extended to an α-approximation algorithm for Quota Steiner Tree problem, hence, obtaining 2.5-approximation algorithm for the problem (and therefore using Garg's result in 2005, it leads to a 2-approximation). They also showed how to obtain a $(5+\epsilon)$-approximation algorithm for Budgeted Steiner Tree problem from Garg's 3-approximation algorithm, where we are asked to find a subtree of maximum profit with a given bounded cost.

Chudak, Roughgarden and Williamson [32] present a general technique for approximating *partial* variants of optimization problems. They showed that this task often reduces to obtaining a Lagrangian Multiplier Preserving (LMP) algorithm for prize-collecting version of the problem. Specifically, they showed that Garg's 5-approximation algorithm can be explained as a simple Lagrangian Relaxation (LR) method using Goemans and Williamson's 2-approximation algorithm [46] for Prize Collecting Steiner Tree problem as a subroutine which possesses the LMP property. They also obtain a 5-approximation algorithm for k-Steiner Tree problem from Geomans and Williamson's algorithm using the same technique. k-Steiner Tree problem is a special case of Quota Steiner Tree problem where profits are zero or one.

An LMP α-approximation for a prize collecting problem is an algorithm finding a solution F such that

$$\alpha\pi(F) + c(F) \leq \alpha OPT,$$

where π and c are the penalty and cost functions, respectively, and OPT is the cost plus penalty of the optimal solution.

Although the approximation guarantee of the aforementioned work is not the theoretically best known result, the technique is of particular interest in our work to solve the Node-Weighted Quota Steiner Tree. The LMP property has been also used previously in the works of Arya and Ramesh [6] and Arora and Karakostas [5] for k-MST and also in the recent work of Archer et al. [3] for Prize Collecting Traveling Salesman and Prize Collecting Steiner Tree problems.

1.2 Node-Weighted Steiner Tree Problems

In the Node-Weighted Steiner Tree problem (NW-ST), the cost function is defined on vertices instead of edges. It is easy to see that the node-weighted version of the problem generalizes the edge-weighted version. Unlike the edge-weighted version, the node-weighted version is much harder. By a simple reduction from Set Cover problem, we can show that NW-ST cannot be approximated up to $(1 - o(1)) \ln n$ [37].

NW-ST has a $2 \ln n$ approximation algorithm due to Klein and Ravi [74]. They use a combinatorial approach to obtain their approximation algorithm. However, the general primal dual approach of Goemans and Williamson [47] can be also used to prove the approximation guarantee. Guha et al. give a primal dual analysis of Klein and Ravi's approximation algorithm that shows the solution of the algorithm is not only within $O(\lg n)$ from the optimal integral solution but also within $O(\lg n)$ factor of the optimal fractional solution of a natural LP [55].

The Node-Weighted Prize Collecting Steiner Tree (NW-PCST) and Node-Weighted Quota Steiner Tree (NW-QST) problems are defined similar to those of edge-weighted versions. Both problems were first studied by Moss and Rabani in [81]. They claimed to solve NW-PCST with a primal dual LMP algorithm of $\log(n)$-approximation factor. However, there is a fundamental flaw in their algorithm: the *monotone one-shot dual growing procedure* employed by their algorithm can be shown to produce dual solution of insufficient value. There is no known such primal dual algorithm for set cover problem and therefore it does not seem that their algorithm have a simple fix. We will define monotone one-shot dual growing procedure more precisely in next chapter.

In this thesis we present a new and inherently different LMP algorithm for NW-PCST. Moss and Rabani's solution for NW-QST problem relies on their prize collecting algorithm but uses the prize collecting algorithm in a black-box manner using a *tree packing method*

and its LMP property. Their algorithm for the quota problem is thus valid using our new LMP algorithm for prize collecting problem. We also show in section 3.1 how to use the Lagrangian Relaxation method to solve NW-QST with our LMP algorithm for NW-PCST problem.

1.3 Generalized Steiner Network Problems and Special Graphs

Here we briefly review some results for generalizations of the Steiner Tree problem as well as some results for planar and Euclidean variants of the problem. The focus here is on the problems with recent progresses; the reader is referred to [57] for a thorough survey.

Steiner Forest Problem. In the Steiner Forest problem, one is given a set of pairs of vertices $\mathcal{D}$ and the problem is asking for a subgraph connecting each pair in $\mathcal{D}$. It is clear that the solution to this problem is a forest and the Steiner Tree problem is a special case where the pairs all share a same fixed vertex. The Steiner Forest problem can be solved using Goemans and Williamson's primal dual algorithm [46] to obtain a 2-approximation algorithm, or any of the algorithms in [2] or [45] for more general problems. Bern and Plassmann proved that the problem does not admit a PTAS [16], however we know from the Steiner Tree problem's hardness that there is a 96/95-inapproximability lower bound for the problem.

It is shown that Steiner Forest problem is NP-hard even in planar graphs [39]. Borradaile showed that the Steiner Tree problem admits a polynomial time approximation scheme (PTAS) in planar graphs [20] which has been also extended to a PTAS for Steiner Forest problem by Bateni, Hajiaghayi and Marx [13]. The Node-Weighted Steiner Forest (NW-SF) in planar graphs has been recently studied by Berman and Yaroslavtsev and they gave a 2.4-approximation for the problem, while the APX-hardness of the problem is still an open question.

Prize Collecting Steiner Forest Problem (PCSF). In [60], Hajiaghayi and Jain studied PCSF (also called Prize Collecting Generalized Steiner Tree) and proposed a 2.54-approximation algorithm. Sharma and Swamy generalized their work in [90] and gave a 3-approximation algorithm for the case where connectivity constraints are arbitrary 0-1 functions and penalty function is submodular as well as a 2.54-approximation algorithm based on LP rounding. Hajiaghayi and Nasri later showed how to obtain the same approximation guarantee (3) for Prize Collecting Steiner Forest via iterative rounding in [62] and how to extend it to a 3-approximation algorithm for a Prize Collecting Survivable

Network Design Problem (described later) where there is a penalty equal to the violated connectivity constraint of a pair.

Bateni et al. have studied the prize collecting version of the Steiner Tree and the Steiner Forest on planar graphs. Particularly, they have shown that PCST admits a PTAS in planar graphs while PCSF is APX-hard [10].

Gutner also gives a $(3-4/n)$-approximation for the Prize Collecting generalized version of Steiner Forest problem in which we have sets $\{T_1, \ldots, T_k\}$ and we have to connect the vertices of each set or pay the penalty associated with the set. If the size of the sets are all 2, this problem is the classic Steiner Forest problem [59]. Gupta et al. also studied a game theoretic version of PCSF in [58].

For node-weighted version of the problem, Bateni et al. has recently found a $\log |\mathcal{D}|$-approximation algorithm [12]. Demaine et al. also proved in [33] that Goemans and Williamson's primal dual algorithm for NW-ST is a 6-approximation in planar graphs. In [80], Moldenhauer studies NW-SF in planar graphs and shows that Geomans-Williamson's algorithm has a tight approximation factor of 3. The author designs a new algorithm with a 18/7-approximation guarantee on planar graphs (an originally claimed 9/4-approximation is later fixed in [15]). The state of the art result for the problem is a 2.4 approximation found by Berman and Yaroslavtsev.

Variants of Steiner Forest Problem has been also studied in Euclidean spaces. The Euclidean Steiner Tree and Steiner Forest problems are known to have PTASs [4, 79, 21]. Euclidean PCSF and Euclidean k-Steiner Forest problems have been also studied in [11].

k-Steiner Forest Problem. In k-Steiner Forest problem, we are given a set of commodity pairs $\mathcal{D}$ and we are looking for the minimum forest satisfying at least k pairs of $\mathcal{D}$. The best current approximation factor for the problem is a $O(\min\{\sqrt{k}, \sqrt{n}\})$ algorithm found by Gupta et al. [56]. There is a reduction from k-densest subgraph to k-Steiner Forest problem [60], therefore, finding sub-polynomial approximation factor for the problem seems to be a hard task.

Survivable Network Design Problem. Survivable Network Design Problem (SNDP) is a generalization of Steiner Forest problem in which, each pair of vertices (s,t) in $\mathcal{D}$ is accompanied by a value $r(s,t)$, called the connectivity constrained of (s,t). We are asked to find a minimum cost subgraph such that each pair of vertices (s,t) in D are connected by at least $r(s,t)$ number of edge-disjoint paths. Same as Steiner Forest problem, we have only edge costs in SNDP.

SNDP was studied by Goemans et al. in [45], they gave a $2\lg(k)$-approximation for the problem, where k is the maximum value among the connectivity constraints of the pairs

in D. Same approximation factor was given in [2] by Agrawal, Klein and Ravi for the case where we are seeking a multiset of edges. Finally, Jain gave a 2-approximation algorithm for SNDP in 2001 [66].

As mentioned, Hajiaghayi and Nasri gave a 3-approximation algorithm for prize collecting version of SNDP (PC-SNDP) where there is a penalty equal to the violated connectivity constraint of a pair. Nagarajan, Sharma and Williamson improved this result to a 2.54-approximation for penalty functions linear in the violation of the connectivity requirements. In the most recent work [61], PC-SNDP with a general penalty function was studied. Namely, the authors designed a constant factor approximation algorithm for the case where the penalty function for each pair is a monotone and submodular function.

Williamson et al. also studied a general version of SNDP in which there is a connectivity constraint for each cut and gave a $2k$-approximation where k is again the maximum connectivity constraint [93]. Gabow et al. slightly improved the factor to $2k-1$ later in [38].

Nutov considered the node weighted version of SNDP (NW-SNDP) and gave a $O(k\log(n))$ approximation for the problem in general graphs [83]. Chekuri et al. improved this result for planar graphs and minor-closed families of graphs to an $O(k)$-approximation algorithm [27]. They also give a $O(k^2\log n)$ and $O(k^2)$ approximation algorithms in general graphs and planar graphs, respectively, for prize collecting version of NW-SNDP[28].

Group Steiner Tree problem. Another interesting variation of Steiner Tree problem is the Group Steiner Tree problem. In this problem, we are given some groups of nodes and we are asked to find a minimum cost subtree covering at least one node from each group. If the size of each group is one, this will be the original Steiner Tree problem and so it is a generalization of the original problem. The best known approximation factor for the Group Steiner Tree problem is $O(\log^3 n)$ in general graphs and $O(\log^2 n)$ in trees [42]. Demaine et al. improved this result to $O(\log \text{poly} \log \log n)$ in special case of planar graphs where each group is the set of nodes on a face [33].

1.4 Overview of The Thesis

In the next chapter, we focus on the Node-Weighted Prize Collecting Steiner Tree problem. First we exhibit an instance demonstrating a flaw in Moss-Rabani's algorithm, then we present our algorithm for the problem which is the main contribution of the thesis, stated in the following theorem.

Theorem 2.5.1. *There is an LMP $O(\log n)$-approximation algorithm for the NW-PCST problem.*

In Chapter 3, we study the applications of the above result. As a direct theoretical application, in section 3.1 we use the Lagrangian Relaxation method, similar to the work of Chudak et.al. [32] to obtain the following result using our algorithm for Prize Collecting problem.

Theorem 3.1.2. *There is an $O(\log n)$-approximation algorithm for Quota Node Weighted Steiner Tree Problem.*

In section 3.2 we focus on a problem proposed by Goldberg and Liu in [48] in social network analysis. The formal definition of the problem as well as a brief review of the related works are given in section 3.2. We present two algorithms for the problem, an $O(r \log n)$-approximation and an $O(l \log n)$-approximation where r and l are the diameter of the graph and number of different threshold values.

Theorems 3.2.5 and 3.2.8. *There are $O(r \log n)$- and $O(l \log n)$-approximation algorithms for Technology Diffusion problem.*

These results improves the $O(rl \log n)$-approximation factor of Goldberg and Liu. Our second algorithm uses the algorithm for Quota Steiner Tree problem as a subroutine. We further show an equivalence of the Technology Diffusion problem to a special case of the Node-Weighted Quota Steiner Tree problem. Finally, in the last part of the thesis the open problems and possible future research are discussed.

Chapter 2

Node-Weighted Prize Collecting Steiner Tree

In the Node-Weighted Prize Collecting Steiner Tree problem (NW-PCST), we are given a graph $G(V, E)$, a cost function $c : V(G) \to \mathbb{R}$ and a penalty function $\pi : V(G) \to \mathbb{R}^+$ and the goal is to find a subtree of the graph minimizing the total cost of the vertices spanned by the tree plus the total penalty of the vertices not spanned in the tree. We study the rooted version of the problem i.e. we are given a vertex r and we are asked to find the optimal tree containing r, therefore our objective is to find

$$\min_{\text{subtree } T \text{ of } G: r \in T} c(T) + \pi(V \setminus T),$$

Where $\pi(S) = \sum_{v \in S} \pi(v)$ for all $S \subseteq V$. Moss and Rabani proposed a polynomial-time algorithm for this problem and claimed that their algorithm is an $O(\log n)$-approximation [81], where $n = |V|$. However, there is a counter example for their algorithm which shows the claimed bound is not correct. The counter example is an instance of the Set Cover problem (the original problem from which the hardness for NW-PCST is derived). There is a $\Theta(n)$ gap between the optimal solution and their the dual solution constructed by their algorithm in this example. We will discuss this in section 2.2.

Moss and Rabani's algorithm as well as the algorithm we propose for the problem use a combinatorial Primal Dual approach. The previous algorithm uses a monotone single-round dual growing procedure and they compare the algorithm's output with a single dual solution constructed during the algorithm. However, there is no known such *single-shot* growing algorithm for the set cover problem. This is the main reason that we believe their

algorithm does not have a simple fix. In the other hand, our algorithm is much more complex and involves multiple rounds of dual growing procedures and produces a feasible dual solution for each.

2.1 Integer Programming Formulation

Let $V' = V \setminus r$. The NW-PCSTprobem can be formulated with the following IP formulation.

$$
\begin{aligned}
\min \quad & \sum_{v \in V'} c(v) x_v + \sum_{S \subseteq V'} \pi(S) z_S && && \text{(P)} \\
\text{s.t.} \quad & \sum_{v \in \Gamma(S)} x_v + \sum_{U | S \subseteq U} z_U \geq 1 && \forall S \subseteq V', && (2.1) \\
& x_v + \sum_{U | v \in U} z_U \geq 1 && \forall v \in V', \\
& x_v \in \{0, 1\} && \forall v \in V', \\
& z_S \in \{0, 1\} && \forall S \subseteq V',
\end{aligned}
$$

Here, x is the characteristic vector of the solution vertices, i.e. $x_v = 1$ if vertex v is in the solution and $x_v = 0$, otherwise. By definition, x_r is always 1. Note that in the node-weighted version of problem, we are not concerned about the edges of the solution tree; as long as the vertices of the solution tree T induce a connected subgraph in G, we can choose any spanning tree of $G[V(T)]$ as the solution tree, where $V(T)$ is the set of vertices of T and $G[V(T)]$ denotes the graph induced by them. Thus, we only need to ensure that the set of vertices characterized by vector x induce a connected subgraph.

The variable z_U in P means the set U is excluded from the solution. We need this variable to enforce the edge connectivity. For a set $S \subseteq V'$, the function $\Gamma(S)$ is defined as the set of vertices adjacent to S but not in S. Therefore the connectivity constraint (2.1) must ensure for every set S not containing r, either S is excluded from solution ($\sum_{U | S \subseteq U} z_U \geq 1$) or there is an adjacent vertex of S included in the solution. In fact, we can convert every optimal integral solution so that there is only one set U for which $z_U = 1$. This is a standard *cut connectivity* constraint, frequently used in IP formulation of network design problems.

We can obtain the linear programming relaxation of (P) by replacing the integrality constraints $x_v, z_S \in \{0,1\}$ with $0 \leq x_v, z_S \leq 1$, then, the dual formulation of the relaxed primal will be as follow:

$$\begin{aligned}
\max \quad & \sum_{S \in V'} y_S + \sum_{v \in V'} p_v && && (\mathrm{D}^0)\\
\text{s.t.} \quad & \sum_{S | v \in \Gamma(S)} y_S + p_v \leq c(v) && \forall v \in V' \\
& \sum_{U \subseteq S} y_U + \sum_{v \in S} p_v \leq \sum_{v \in S} \pi(v) && \forall S \subseteq V' \\
& y \geq \mathbb{0} \\
& p \geq \mathbb{0}
\end{aligned}$$

Note that in the original problem, we may assume that each vertex has either zero cost or zero penalty. The optimal solution's cost is not less than the constant value $\sum_{v \in V} \min\{\pi(v), c(v)\}$. So by subtracting cost and penalty of each vertex v by the value $\min\{\pi(v), c(v)\}$, we only have shifted the value of optimal solution by a constant value and therefore an approximation algorithm's factor is preserved for the problem after this modification. The analogy of this is to set the variable p_v in D^0 to the fixed value $\min\{\pi(v), c(v)\}$ for each vertex v.

To this aim, we call a vertex v an *expensive vertex*, if its cost is more than its penalty, i.e. $c(v) > \pi(v)$, and we call v a *cheap vertex*, otherwise. Now we can define the *reduced cost* and *reduced penalty* $c_r(v)$ and $\bar{\pi}(v)$ of each vertex as $\bar{c}(v) = c(v) - \min\{\pi(v), c(v)\}$ and $\bar{\pi}(v) = \pi(v) - \min\{\pi(v), c(v)\}$ and the value $p(v) = \min\{\pi(v), c(v)\}$. After these modifications, the new dual formulation becomes as follows, with the variable p eliminated:

$$\begin{aligned}
\max \quad & \sum_{S \subseteq V'} y_S + p(V) && && (\mathrm{D})\\
\text{s.t.} \quad & \sum_{S | v \in \Gamma(S)} y_S \leq \bar{c}(v) && v \in V && (2.2)\\
& \sum_{U \subseteq S} y_U \leq \sum_{v \in S} \bar{\pi}(v) && \forall S \subseteq V' && (2.3)\\
& y \geq \mathbb{0}
\end{aligned}$$

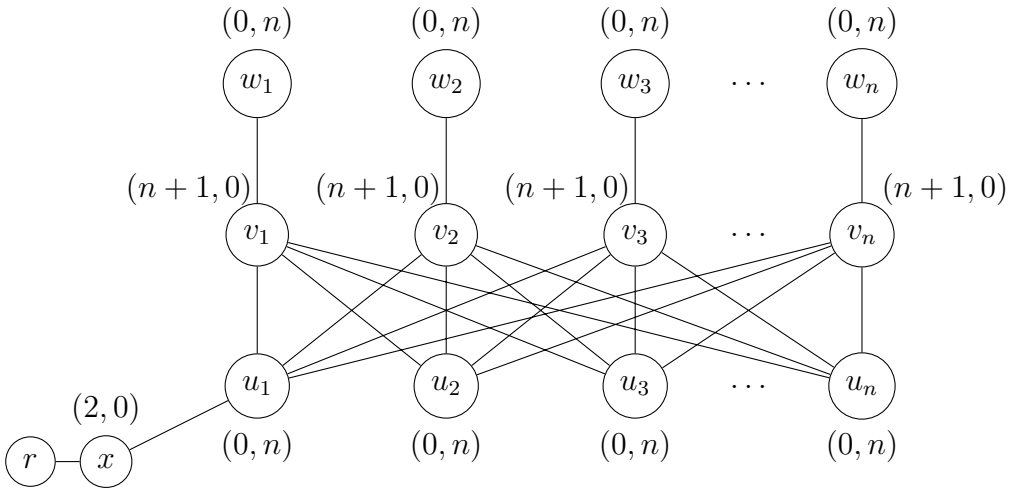

Figure 2.1: Counterexample to Moss-Rabani's algorithm.The value pairs on each node demonstrates the cost and penalty of the node.

2.2 Counterexample to Moss-Rabani's Algorithm

The algorithm of Moss and Rabani uses the same integer program and the relaxed dual formulation (D) given in the previous section. They claimed that their algorithm finds a feasible dual solution along with a tree such that the cost (plus penalty) of the tree is within $O(\log n)$ factor of the total value of the dual solution constructed. Regardless of how they construct the tree, in a counter example, we show that their algorithm returns a dual solution with $\Omega(n)$ gap between the value of dual solution and optimal solution. This clearly shows that their algorithm cannot provide an $O(\log n)$ approximate solution and there is a mistake in their approximation guarantee proof.

Their algorithm constructs a dual solution as follows. Consider the connected components in the graph induced by cheap vertices. Let these be in the family $\mathcal{C}$. Starting with an all-zero solution y, the algorithm continuously increases the values of y_S for each $S \in \mathcal{C}$ until either the inequality (2.3) or the inequality (2.2) gets tight, keeping the dual solution always feasible. The dual value of a set containing the root vertex r is never increased.

In the first case, if S is the set for which the constraint is tight, the algorithm *deactivates* C, meaning that the algorithm stops increasing dual value y_S and moves S from $\mathcal{C}$ to a collection $\mathcal{C}'$ of *deactivated* sets. In the second case, if $\tilde{v}$ is the vertex for which the constraint (2.2) has become tight, the algorithm merges every set in $\mathcal{C}$ and $\mathcal{C}'$ adjacent to $\tilde{v}$ together

with $\tilde{v}$ and removes them from $\mathcal{C}$ and adds this newly merged component to $\mathcal{C}$ as a new active component. The growing process continues until each set is either deactivated or merged with the component containing r.

Figure 2.1 shows our counter example for Moss-Rabani algorithm. It consists of a complete bipartite graph with vertex sets $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. Let $W = \{w_1, \ldots, w_n\}$. For each $1 \leq i \leq n$, node v_i is also attached to another vertex w_i of degree 1. We also have a root vertex r attached to u_1 through a path of length two. The vertices in U and W are cheap vertices and have cost zero and penalty n while the vertices in V are expensive vertices with cost $n+1$ and zero penalty. The vertex x connecting u_1 to the root is also an expensive vertex with cost two.

Running the Moss-Rabani algorithm on this instance of the problem, the dual values $y_{\{u_i\}}$ and $y_{\{w_i\}}$ increases uniformly for each $1 \leq i \leq n$ to value 1. Then the constraint (2.2) gets tight for all vertices in V and therefore they are being merged with their adjacent active sets, which are all sets containing single vertices from V and W. Then there will be a single active set $U \cup V \cup W$ and the dual value of this new set increases for another one unit till finally constraint (2.2) gets tight for x, the set merges the root vertex and the algorithm terminates.

We observe that the dual solution found by the algorithm has total dual value $O(n)$, however, there is a feasible dual solution y^* having $y^*_{\{w_i\}} = n$ for all vertices in W and $y^*_R = 0$ for all other sets R. This dual solution has total dual value $\Theta(n^2)$, so by weak duality the optimal solution for NW-PCST problem also has value $\Omega(n^2)$. This shows that the Moss-Rabani's dual solution cannot provide a $O(\log n)$ approximation algorithm with the constructed dual solution.

The given counter example is actually an instance of Set Cover problem. It is also well known that one-shot monotone uniform growing of the dual values does not produce an $O(\log(n))$ approximate dual solution for set cover problem. Therefore it is expected that uniform growing of dual values does not provide an $O(\log(n))$ for Steiner Tree problem as well. This is the main reason that we believe the one-shot uniform growing is not the right primal-dual approach for the problem.

Note that by one-shot monotone growing procedure, we mean an algorithm which obtains a feasible dual solution only by increasing the dual values. Of course if we seek solutions by allowing other modifications such as scaling or by using other approaches such as finding multiple dual solutions, we may find feasible dual solutions with desired $O(\log n)$ factor of optimal solution.

In [55] it is shown that there actually exists a monotone dual growing approach which gives an $O(\log n)$ approximation algorithm for the classical Node-Weighted Steiner Tree

problem. The question is that why such approach fails for the Prize Collecting version? We can explain this in our counter example. If we grow the dual values monotony, there might be as $O(n)$ many expensive vertices adjacent to each cheap vertex u_i, so that the dual value $y_{\{u_i\}}$ of each cheap vertex contributes to the left hand side of $O(n)$ of inequalities of type (2.2).

This means that we may have expensive vertices in our final component connected to the root with total cost of $O(n)$ times the total dual value. This does not make a problem in classic Node-Weighted Steiner Tree problem since the penalties are infinity: we have to connect all terminal vertices. But in the Prize Collecting version, if we simply decide to connect all of these expensive vertices as the Moss-Rabani's algorithm does, we may pay $O(n)$ times the constructed dual solution.

This problem cannot be solved by simply branching the expensive vertices (which does not connect any cheap vertex to other cheap vertices), as done by Moss-Rabani's algorithm. As in our counter example, there might be still some cheap vertices w_i being actually connected by these expensive vertices and the problem still remains. So the critical problem here is that when to stop the growing and which expensive vertices covered in components to include in our solution.

Our algorithm explained in the next section shows how we overcome this problem by stopping the growing procedure early and how we use more involved procedures to construct the tree after obtaining a dual solution. Our algorithm does not select all of the expensive vertices in the components connecting a cheap vertex but only selects some carefully.

2.3 Algorithm

In this section we propose a different algorithm for NW-PCST problem. Our algorithm works in phases and in each phase i, the algorithm constructs a feasible dual solution y^i for (D). The algorithm also maintains a tree T_r containing the root and incrementally adds vertices to this tree in each phase. Let us denote $V \setminus \{r\}$ by V'. Starting with zero values for y^i_S for all $S \subseteq V'$, the phase i starts by increasing the dual values of sets in a collection $\mathcal{C}^i$, called the *initial components*. As we will prove later the sets in $\mathcal{C}^i$ are always a collection of disjoint sets and no two of them are connected with an edge. The initial components $\mathcal{C}^1$ in the first phase are the connected component of the graph induced by cheap vertices and r, except the component containing r which forms T_r.

We grow all sets in $\mathcal{C}^i$ with same rate, i.e. the growing process is monotone. The algorithm is careful that the dual values always form a feasible dual solution for D. The

component T_r is always inactive in the growing process and we never increase its dual value. At the end of each phase i, except the last phase, the algorithm constructs a tree T^i connecting some (at least two) of the components in $\mathcal{C}^i$. For the next phase, we first set $\mathcal{C}^{i+1} = \mathcal{C}^i$, then, all the initial components connected by T^i are removed from $\mathcal{C}^{i+1}$ and replaced by a component formed by T^i and all components that T^i is connecting. Therefore the set of initial components of phase $i+1$ is determined by $\mathcal{C}^i$ and the tree T^i found in phase i.

We can describe the above process in each phase as process over time in which we grow components uniformly with the same rate. In the rest of this section we are talking about a single phase i. For ease of notation, we eliminate the superscript i whenever there is no ambiguity. Let y^τ be the dual solution at time τ in a phase i, and let $\mathcal{S}^\tau$ be the collection of all components with positive dual values, called the *support* of y^τ. We call every inclusion-wise maximal set $S \in \mathcal{S}^\tau$ a *moat*. A moat S is called *active* if

$$\sum_{U \subseteq S} y_U^\tau < \bar{\pi}(S),$$

and *inactive* otherwise. Let $\mathcal{A}^\tau$ denote the collection of active moats in phase i at time τ and $\mathcal{I}^\tau$ be the collection of inactive moats. Therefore, by definition, $\mathcal{A}^0 = \mathcal{C}^i$ and $\mathcal{I}^0 = \emptyset$. Now we can define the notion of *age* of an initial component. For an initial component $C \in \mathcal{C}^i$ we define the age of C at time τ as follows:

$$\text{age}^\tau(C) = \min\{\tau, \sum_{v \in C} \bar{\pi}(c)\}.$$

The age of a component at time τ denotes the first time the component is inside an inactive moat, if this has ever happened till time τ. Let us call a vertex v a *tight* vertex at a time τ if the inequality (2.2) is tight for v. Note that cheap vertices are always tight. The algorithm ensures that no two active moats are adjacent to a tight vertex at any time during a phase. We will show this later in this section. This means at a time τ, for any moat S, there is at most one initial component in $\mathcal{C}^i$ inside S which has age equal to τ and all other such initial components have age less than τ. We call this initial component the *core* of S for an active S. In other words, an initial component $C \in \mathcal{C}^i$ is the core of a moat S at time τ, if $C \subseteq S$ and C has the maximum age among all other initial components inside S.

We define the age of a moat S equal to age of its core. The notion of core is especially important in the next step in explaining how we construct the tree T^i. Now we are ready to give more details about the growing process. In each phase i, we increase the dual value

y_S of all sets $S \in \mathcal{A}^\tau$ at time τ uniformly until one of the constrains of D becomes tight. We ensure that y^τ forms a feasible dual solution at any time τ during a phase. So either of the following may happen during the growing process.

A. Inequality (2.3) becomes tight for a set S: we stop increasing the dual value of S and S becomes an inactive component.

B. Inequality (2.2) becomes tight for a vertex v: we alter active components and check the termination condition.

Phase i ends whenever all components are inactive or the termination condition (which we will define later) is satisfied. In the former case, the algorithm terminates and phase i is the last phase. In the later case, the algorithm constructs a subtree T^i based on the dual values y^i found in the phase.

In case (A), the set S is moved from the set of active components $\mathcal{A}^\tau$ to set of inactive components $\mathcal{I}^\tau$. At this time, if there are no active components left, the algorithm terminates, otherwise we simply continue growing the rest of the active components.

Now consider case (B). We say an initial component $C \in \mathcal{C}^i$ *loads* a vertex v, if at a time τ, there exists an active component $S \in \mathcal{S}^\tau$ with $\text{core}(S) = C$ such that $v \in \Gamma(S)$, where $\Gamma(S) = \{u \in V \setminus S | \exists x \in S : ux \in E\}$ is the set of adjacent vertices of S. Let $\mathcal{L}^\tau(v)$ be the set of all initial components loading v at time τ or earlier. The termination condition is as follows: the phase terminates at the earliest time τ that there is a tight vertex $\tilde{v}$ such that $\tilde{v} \in \Gamma(T_r)$ or

$$\sum_{C \in \mathcal{L}^\tau(\tilde{v})} \text{age}^\tau(C) \geq \frac{3}{2}\tau. \qquad (\star)$$

At this time, we construct the tree T^i, connecting the vertex $\tilde{v}$ to the core of all its adjacent moats (either active or inactive) to form a new component. If $\tilde{v} \in \Gamma(T^r)$, T^i is added to T^r, otherwise it is added to $\mathcal{C}^{i+1}$ as a new initial component for phase $i+1$. Later, we will prove that the cost of tree T^i is proportional to y^τ, to guarantee our approximation factor.

After construction of T^i, we construct the initial components for the next phase. First set $\mathcal{C}^{i+1}$ to the set of all components in $\mathcal{C}^i$ not connected to T^i (by connected we mean either sharing a vertex or having a vertex adjacent to T^i). Then, merge T^i with all components connected by T^i together to form a new component and add this component to $\mathcal{C}^{i+1}$.

The rest of this section will focus on explaining how the tree T^i is being constructed. We call the tree T^i the *phase* tree. We first set $T^i = (\{\tilde{v}\}, \emptyset)$ and gradually add vertices to the phase tree using the procedures *FindSubtree* (abbreviated `FST`) and *ConnectVertex* (abbreviated `CVtx`). The *key property* of T^i is as follows: for every expensive vertex $w \in T^i$, we want all cores loading w to be connected by T^i. This is done by starting with the single vertex $\tilde{v}$ and using function `FST`. Given a set S and a set $L \subseteq \Gamma(S)$, the job of `FST`(S, L) is to find a subtree of S connecting vertices in L to core(S).

As expected, in the procedure of connecting the vertices in L to core(S), `FST` needs to add some new expensive vertices to the phase tree, enforcing it to connect more cores to the phase tree. Therefore, this function calls itself recursively to satisfy the mentioned key property of the phase tree. Note that in recursion, we may encounter cases that a core is loading more than one vertex in T^i, however we do not call `FST` for each of those vertices and a moat more than once, rather, we pass all those vertices (as the set L) in one call to `FST` to make sure `FST` is called on each moat at most once during the algorithm.

Let $S_1, \ldots, S_k$ be the inclusion-wise maximal moats adjacent to $\tilde{v}$. We start construction of T^i by calling `FST`$(S, \{\tilde{v}\})$ for all $1 \leq i \leq k$. Note that during the recursive calls of `FST`, there may be two moats S and S' with different cores both loading a vertex in L such that S' is embedded in S (see figure 2.2), in this case, rather than directly calling `FST`(S, L) and `FST`(S', L), we call `FST`(S, L) and `FST`(S', L') will be called recursively from `FST`(S, L) for a set L' containing some vertices of L. The reason is that there may be other expensive vertices loaded by core of S', which will be later added to T^i and we should call `FST` on S' only once passing all these vertices to the procedure all in once. The comprehensive explanation of these two procedures are given bellow. Algorithm 1 shows the pseudo code for the whole algorithm.

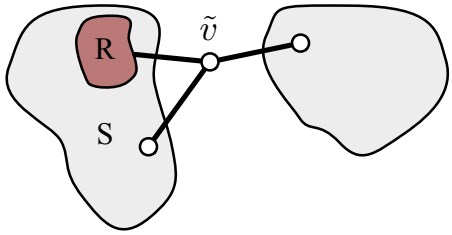

Figure 2.2: An adjacent moat to $\tilde{v}$ (R) may be embedded in another moat (S).

2.3.1 FindSubTree

Consider the procedure `FST`(S, L). Whenever `FST`(S, L) is called during the algorithm, $L \subseteq \Gamma(S)$ by the specification of the algorithm. Let τ be the age the of set S at the end of the phase. The graph H_S is defined as follows. Start by the graph $G[S \cup L]$ and identify

Algorithm 1 PrizeCollectingSteiner($G(V,E), c, \pi$,r)

$\mathcal{C}^1 \leftarrow \{S : S$ is a connected component not containing r in the graph induced by cheap vertices and $r\}$
$T_r \leftarrow$ the connected component containing r in the graph induced by cheap vertices and r.
Let $i = 0$ denote the phase number.
while $\mathcal{C}^{i+1} \neq \emptyset$ **do**
  $i \leftarrow i + 1$
  Initialize $y^i_S \leftarrow 0$ for all $S \subseteq V'$, $\mathcal{A}^i \leftarrow \mathcal{C}^i$, $\mathcal{I}^i \leftarrow \emptyset$, $T^i \leftarrow \emptyset$, $\tau^i \leftarrow 0$
  while $\mathcal{A}^i \neq \emptyset$ **do**
    $\epsilon_1 \leftarrow \min_{v \in V \setminus \cup_{S \in \mathcal{A}^i \cup \mathcal{I}^i} S} \{ \frac{\bar{c}(v) - \sum_{S \subseteq V | v \in \Gamma(S)} y^i_S}{|\{S \in \mathcal{A}^i : v \in \Gamma(S)\}|} \}$
    $\tilde{v} \leftarrow$ the vertex minimizing the statement above
    $\epsilon_2 \leftarrow \min_{S \in \mathcal{A}^i} \{ \sum_{v \in S} \bar{\pi}(v) - \sum_{R \subseteq S} y^i_R \}$
    $\epsilon \leftarrow \min\{\epsilon_1, \epsilon_2\}$
    $\tau^i \leftarrow \tau^i + \epsilon$
    $y^i_S \leftarrow y^i_S + \epsilon$ for all $S \in \mathcal{A}^i$
    $\text{age}^\tau(S) \leftarrow \text{age}^\tau(\text{core}(S)) \leftarrow \tau^i$ for all $\forall S \in \mathcal{A}^i$ and $\tau \geq \tau^i$
    if $\epsilon = \epsilon_2$ **then**
      for $S \in \mathcal{A}^i : \bar{\pi}(S) = \sum_{R \subseteq S} y^i_R$ **do**
        Remove S from $\mathcal{A}^i$ and add it to $\mathcal{I}^i$
      end for
    else
      $\mathcal{N} \leftarrow \{$inclusion-wise maximal $S \in \mathcal{A}^i \cup \mathcal{I}^i : \tilde{v} \in \Gamma(S)\}$
      if $\sum_{C \in \mathcal{L}^{\tau^i}(\tilde{v})} \text{age}^{\tau^i}(C) < \frac{3}{2}\tau^i$ **and** $\tilde{v} \notin \Gamma(T_r)$ **then**
        Remove all sets in $\mathcal{N}$ from $\mathcal{A}^i$ and $\mathcal{I}^i$
        add $\{\tilde{v}\} \cup \big(\cup_{R \in \mathcal{N}} R \big)$ to $\mathcal{A}^i$
      else
        $T^i \leftarrow (\{\tilde{v}\}, \emptyset)$
        for $S \in \mathcal{N}$ **do**
          FST$(S, \{\tilde{v}\})$
        end for
        break while
      end if
    end if
  end while

Algorithm 1 PrizeCollectingSteiner (continued)

```
    if T^i ≠ ∅ then
        C^{i+1} ← C^i \ {C ∈ C^i : C ⊆ T^i}
        if T^i ∩ Γ(T_r) ≠ ∅ then
            Add vertices of T^i to T_r
        else
            Add T^i to C^{i+1} as a new initial component
        end if
    else
        return T_r
    end if
end while
return T_r
```

Algorithm 2 FST(S, L)

```
Construct the auxiliary graph H_S and compute the auxiliary costs c_S
if |L| = 1 then
    P ← L
else
    Assume that L = {a, b} for some a, b ∈ Γ(S)
    P ← a minimum c_S cost a,b-shortest path in H_S
end if
for each super vertex R ∈ H_S : P ∩ Γ(R) ≠ ∅ do
    FST(R, P ∩ Γ(R))
end for
T^i ← T^i ∪ { original vertices in P}
z_0 ← arbitrary vertex in core(S)
CVtx(S, z_0, 0)
T_i ← T_i ∪ core(S)
```

Algorithm 3 $\texttt{CVtx}(S, z, d)$

Construct the auxiliary graph H_S and auxiliary costs c_S
$\hat{T} \leftarrow T_i$.
For every super vertex $R \in V(H_S)$ if $R \cap T_i \neq \emptyset$, identify the vertices of $R \cap \hat{T}$ to super vertex R in $\hat{T}$
$Q = q_1 \dots q_l \leftarrow$ a minimum c_S cost shortest path from z to $\hat{T}$ in H_S
$T_i \leftarrow T_i \cup \{$ original vertices in $Q\}$
for super vertex $R \in V(H_S) \setminus V(\hat{T}) : Q \cap \Gamma(R) \neq \emptyset$ **do**
    $\texttt{FST}(R, \Gamma(R) \cap Q)$
end for
if $q_l = R$ is a super vertex **then**
    $\texttt{CVtx}(R, q_{l-1}, d+1)$
end if

each inclusion wise maximal inactive component R inside S to a vertex. We denote the identified vertex corresponding to R by R as well and we call each identified vertex a *super vertex*.

The graph H_S may contain both vertices and super vertices, however the original vertices except the ones in the core of S are all expensive vertices, since every cheap vertex is included in an initial component as explained before. Moreover, since there are no two adjacent moats anytime during the growing procedure, therefore no two super vertices are adjacent. Also, all vertices included in any moat are tight, therefore all original vertices in H_S are also tight vertices.

We define the auxiliary cost $c_S(v)$ for each original vertex v in H_S as follows:

$$c_S(v) = \sum_{\substack{U \subseteq S | v \in \Gamma(U), \\ \text{core}(U) = \text{core}(S)}} y^{\tau}_U,$$

and let $c_S(R) = 0$ for each super vertex R and $c_S(v) = 0$ for all $v \in \text{core}(S)$. The procedure $\texttt{FST}(S, L)$ has two parts:

Part I. First, $\texttt{FST}(S, L)$ connects the vertices in L by finding a subtree and adding it to the phase tree. Recall that we always have $|L| \leq 2$ (We will prove this in next section). We find a path P in H_S connecting vertices in L with minimum c_S-cost i.e. minimizing $\sum_{v \in P} c_S(v)$. If $L = \{a\}$, then the path P is simply the single vertex a, otherwise, P may have super vertices. In the latter case, all expensive vertices are going to be added to the phase tree and for each super vertex $R \in P$, we find a subtree in R connecting the two

adjacent vertices of R in P, say a' and b'. Indeed, this is done by recursively calling the procedure $\texttt{FST}(R, \{a', b'\})$ for all such R and $\{a', b'\}$.

To maintain the key property of the phase tree described before, we need to connect every initial component in S loading a expensive vertex in P. So for every super vertex R' in H_S adjacent to P, but not in P, we call $\texttt{FST}(R', \Gamma(R') \cap P)$ recursively from $\texttt{FST}(S, L)$. All the expensive vertices of the subtree found in $\texttt{FST}(S, L)$ are added to the phase tree, and therefore at the end, every core loading an expensive vertex in the phase tree will be connected to the phase tree as desired.

Part II. In the second part, the goal is to connect the core of S to phase tree. Here we have already added some vertices from S to the phase tree thus we use another procedure, the `CVtx`, to connect an arbitrary vertex z in core(S) to the phase tree. The procedure `CVtx` takes a moat S, a vertex $z \in S \cup \Gamma(S)$ and a level index i (for analysis purposes) and connects z to the phase tree by adding a subtree in S to the phase tree and of course maintaining the key property of the phase tree.

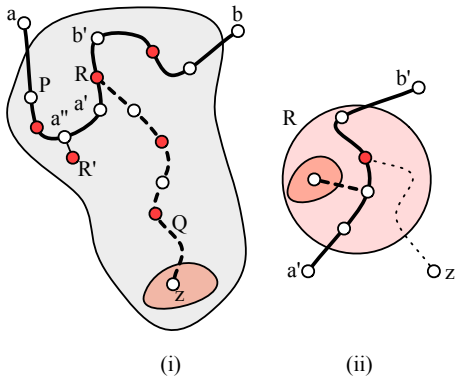

Figure 2.3: (i) Inside procedure `FST`. The core is appearing in small red subset, the path P is shown in solid and the path Q in stroke lines. The white vertices are original expensive vertices and the vertices shown in red are the super vertices. (ii) $\texttt{FST}(R, \{a', b'\})$ is called recursively.

2.3.2 `CVtx`

Consider a procedure call $\texttt{CVtx}(S_d, z_d, d)$ where S_d is a moat, z_d is a vertex in $S_d \cup \Gamma(S_d)$ and d is the recursion level of the procedure call. Similar to `FST` function, `CVtx` first constructs a graph H_{S_d} constructed as follows. Start with $G[S_d \cup \{z_d\} \cup T^i]$ and identify every inclusion wise maximal inactive component R at time τ inside S_d to super vertex R, where τ is the age of S_d at the end of the phase.

Similarly, define $c_{S_d}(v)$ to be the total load of moats with core core(S_d) for every original expensive vertex in H_{S_d} and let $c_{S_d}(v) = 0$ for all super vertices or vertices in core(S_d). First, `CVtx` finds a minimum c_{S_d}-cost path Q_d in H_{S_d} from z_d to T^i. By T^i we mean the vertices and edges added to the phase tree at the point $\texttt{CVtx}(S_d, z_d, d)$ is called. The expensive vertices in Q_d will be added to the phase tree same as expensive vertices of path P in `FST`. Let S_{d+1} be the last (super or non-super) vertex of Q_d (z_d being the first vertex).

Note that here there are some super vertices containing some vertices of T, therefore the last vertex of the path Q_d may be a super vertex in H_{S_d}, or an original vertex of T^i.

If S_{d+1} is a super vertex, then $S_{d+1} \cap T^i$ is not empty, which means we have called the function `FST` on set S_{d+1} before. This is important since as mentioned before, we do not want to call `FST` on a moat more than once. In this case we call $\texttt{CVtx}(S_{d+1}, z_{d+1}, d+1)$, where z_{d+1} is the second last vertex in Q_d. If S_{d+1} is an original vertex, then `CVtx` is not called from $\texttt{CVtx}(S_d, z_d, d)$. Similar to the case for path P in `FST`, we have to call `FST` on every super vertex S' in Q_d to make a connected subgraph connecting z_d to the phase tree. Therefore $\texttt{FST}(S', Q_d \cap \Gamma(S'))$ is called recursively for all super vertex in Q_d if $S' \neq S_{d+1}$.

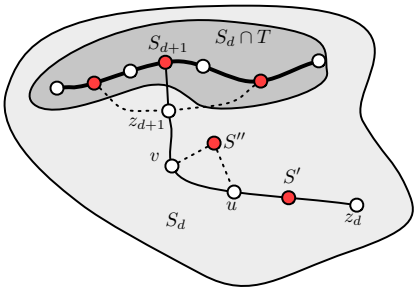

Figure 2.4: Inside procedure $\texttt{CVtx}(S_d, z_d, d)$.

Also, there may be some super vertices S'' adjacent to some expensive vertices in Q_d loading these expensive vertices. Therefore, similarly we call $\texttt{FST}(S'', \Gamma(S'') \cap Q_d)$ for these super vertices as well, if $S'' \cap T^i = \emptyset$ (since we do not want to call `FST` on a most on which `FST` is called before). However, the the only case that a moat intersecting with T^i is loading an expensive vertex in Q_d is that the moat S'' is loading z_d and not any other vertex in Q_d. This follows from the c_{S_d}-minimality of Q_d: if not, the path is not minimal and we can shortcut z_d to obtain a shorter path to T^i.

2.4 Analysis

2.4.1 Correctness

Here we first prove the correctness of properties we claimed in the previous section. It is clear from the specification of `FST` and `CVtx` that the phase tree is a connected subtree in each phase and since T_r is a tree incremented by phase trees adjacent to it in some phases, T_r is also a connected subtree containing the root. However, we have to give formal proofs for the two essential properties of the dual solution procedure calls which we stated previously.

Proposition 2.4.1. *In each phase, each component $S \in \mathcal{S}^\tau$ has a unique core with respect to the dual solution y found in the phase.*

Proposition 2.4.2. *The function* `FST` *is never called with a set L with cardinality more than two.*

The proof of first proposition is straightforward. It is clear that at the beginning of the phase the proposition holds. Now consider the first time τ in which the proposition is violated for a component S at a time τ. Note that S is formed at time τ, therefore there is a vertex v in S which became tight at τ and S formed by merging v with its adjacent moats. However, if there are two initial components (cores) in S with age τ at time τ, then these two components were the cores of two moats adjacent to v before merging and so the termination condition ($\star$) was satisfied since v is being loaded at least 2τ, a contradiction.

Using the first proposition, now we can prove the second proposition. First note that since every moat has a unique core, the specification of `FST` and `CVtx` given in the previous section are valid. Consider each case the procedure `FST` is called:

Case 1. `FST`(S, L) is called from the main procedure. Then, $L = \{\tilde{v}\}$ and $|L| = 1$.

Case 2. `FST`(S, L) is called when S is a super-vertex in path P or Q_d computed in either another `FST` or `CVtx` respectively. In these case it is clear that $|L| \leq 2$ since there are at most two vertices adjacent to S in these paths and they are both minimum cost paths (otherwise, we can shortcut at least one adjacent vertex of S in the path out of three vertices, to obtain a shorter path)

Case 3. `FST`(S, L) is called when S is a moat loading a vertex in the paths P or Q_d computed in the procedures `FST` or `CVtx` respectively. In this case, $L = P \cap \Gamma(S)$ (the case for Q_d is similar). As stated before, no two super vertices are adjacent, so all vertices in L are expensive vertices. If $|L| = 3$ (or more), then we can shortcut the vertex residing between the other two in the path and find a path with lower cost containing S. Note that auxiliary cost of S is zero. Hence $|L| \leq 2$ in this case as well.

2.4.2 Bounding The Cost of The Phase Tree

Now we are ready to prove how we obtain the claimed approximation guarantee for our algorithm. The sketch of the proof is as follows. We first prove that we can compute a charging scheme $\Phi(C)$ for every initial component C to charge the cost of each phase tree T^i to some initial components connected by T^i. Then, we prove that the total charge $\Phi(C)$ of each initial component is within constant factor of its age at the end of the phase. Using

the fact that whenever an initial component is charged it is merged with some other initial components, and using the special coefficient $\frac{3}{2}\tau$ in our termination condition, we prove among all dual solutions computed during all phases, there is one dual solution such that the total cost of T_r is within factor of $\log n$ of this dual solution.

As mentioned, we first need to prove that the cost of T^i can be charged properly to the initial components of phase i connected by T^i. Assume that phase i finished at time τ and y^τ is the feasible dual solution for (D) computed till time τ in that phase. First note that as discussed in previous section, all active and inactive moats are vertex-disjoint at any time and no two of them are adjacent and so the collection $\mathcal{S}^\tau$ forms a laminar family i.e. every two moats are either vertex disjoint or one is completely embedded in the other. Also it is clear by the definition of the algorithm, a procedure call `FST` on set S never calls `FST` on another set $S' \subseteq S$ more than once and the sub procedure calls are called on disjoint inclusion-wise maximal inactive sets inside S, therefore `FST` is called at most once during the phase on each set.

Consider a procedure call $\texttt{FST}(S_1, L)$. First, a path P is computed and then in a chain of calls to procedures $\texttt{CVtx}(S_1, z_1, 1), \ldots, \texttt{CVtx}(S_m, z_m, m)$, a series of paths $Q_1, \ldots, Q_m$ are created to connect $\text{core}(S_1)$ to the phase tree, where m is the number of consecutive recursive calls starting from $\texttt{FST}(S_1, L)$ (See figure 2.5). First we prove that for every moat $S \in \mathcal{S}^\tau$, the C_S-cost of any minimum path between any two vertices in H_S is not greater than $\text{age}^\tau(\text{core}(S))$.

Lemma 2.4.3. *For every moat S, the C_S-cost of a minimum path P between any vertex $a \in core(S)$ and $b \in V(H_S)$ is at most $age^\tau(core(S))$.*

Proof. Let $C = \text{core}(S)$. We use induction on c_S-distance of b from C. It is clear that this distance is the same for all vertices in C, since $c_S(v) = 0$ for all vertices $v \in C$. Let $\text{dis}_S(b)$ be the c_S-distance of b from C in H_S. We also prove that if b is an expensive vertex, b gets tight at time $\text{dis}_S(b)$ and if b is a super-vertex b merges with a moat with core $\text{core}(S)$ at time $\text{dis}_S(b)$.

For every vertex in $\text{core}(S)$, the statement is true, so consider a vertex $b \in H_S$ with $\text{dis}_S(b) > 0$. Let b' be the adjacent vertex of b with minimum $\text{dis}_S(b')$. First note that if b is a super-vertex, then b' is an original vertex and $\text{dis}_S(b) = \text{dis}_S(b') \leq \text{age}^\tau(S)$ and b is merged with a moat with core C at the time b' gets tight and we are done; so assume that b is an expensive vertex. By induction hypothesis, b' becomes tight (or similarly gets merged with a moat with core C if b' is a super-vertex) at time $\text{dis}_S(b')$ and b' is the first vertex among neighbors of b which gets tight. Before time $\text{dis}_S(b')$, none of the neighbors

of b are tight, therefore b is not adjacent to any moat with core core(S), however, at time $\text{dis}_S(b')$, after b' becomes tight, b becomes adjacent with a moat S' with core C.

Since the growing process is monotone and uniform, and at any time, b is being loaded by exactly one moat with core core(S) (after time $\text{dis}_S(b')$), the load of core(S) on b is equal to the time span from the time that first neighbor of b becomes tight until b becomes tight itself, which is by definition of auxiliary cost, equal to $c_S(b)$. So, b gets tight at time $\text{dis}_S(b') + c_S(b)$ and this is equal to $\text{dis}_S(b)$, since by assumption b' is the closest neighbor of b to core(S) in terms of c_S-cost. It follows that every vertex b gets tight at time $\text{dis}_S(b)$, and therefore $\text{dis}_S(b) \leq \text{age}^\tau(C)$ since b joins a moat with core C at a time before C becomes core of an inactive set. □

Note that for every vertices $a, b \in H_S$, the minimum c_S-cost path is not longer than total of two paths from a and b to core(S):

Remark 2.4.4. *For every moat S, the C_S-cost of a minimum path P between any two vertices $a, b \in V(H_S)$ is at most $2\text{age}^\tau(\text{core}(S))$.*

The previous lemma and the previous remark shows that the reduced costs of paths P and Q_1 in procedure calls `FST`(S_1, L) and `CVtx`$(S_1, z_1, 1)$, are each bounded by $2\text{age}^\tau(S_1)$. These leads us to the idea that we can charge the reduced costs of these paths to C. Note that we can charge the auxiliary cost, not the original cost, which means the cost of each vertex will be charged to many cores, but we know that for every expensive vertex w in T^i, all cores loading w are connected to the tree T^i, and cost of w is exactly the total loads of different initial components on it. This is true because we only include tight vertices in the phase tree, i.e. the vertices which their cost has been equal to the total load on them.

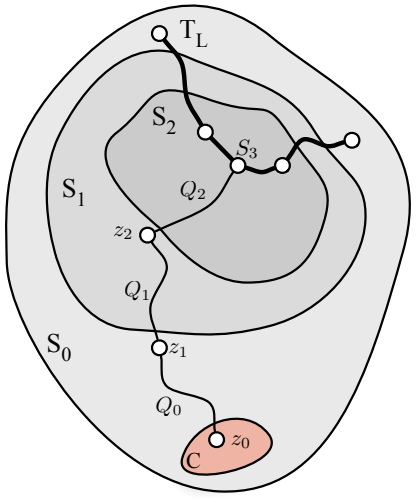

Figure 2.5: The chain of `CVtx` calls invoked from a call to `FST`. The thick black lines indicate edges of the tree T_L constucted by `FST`.

The idea above is a key point in our charging scheme success. However, these are not the only costs we charge to the core of S_1. More precisely, $\Phi(\text{core}(S_1))$ is total of the following costs:

C1. The auxiliary cost $c_{S_1}(P)$ of P in H_{S_1}.

C2. The auxiliary costs $c_{S_1}(Q_1) + \ldots + c_{S_m}(Q_m)$.

C3. The non-auxiliary costs of $\bar{c}(z_2) - c_{S_1}(z_2), \ldots, \bar{c}(z_m) - c_{S_{m-1}}(z_m)$, where z_i is the second last vertex of path Q_{i-1}, $2 \leq i \leq m$.

The exception here are the vertices $z_2, \ldots, z_m$. But why do we charge their non-auxiliary cost as well as auxiliary costs to core(S_1)? Recall that we said z_i is the possible vertex adjacent to a set intersecting the phase tree in the path Q_{i-1} in procedure `CVtx`$(S_{i-1}, z_{i-1}, i-1)$ and we did not call `FST` on sets intersecting the phase tree because we did not want to call `FST` on a set more than once.

So the reason that non-auxiliary cost of z_i is being charged to core(S_1) is that $z_2, \ldots, z_m$ are the only vertices for which we do not call `FST` on every set loading them in the process of connecting core(S_1) to the phase tree. Fortunately, as we will see in the following lemma, the total non-auxiliary cost of these vertices does not exceed $O(1)$.age$^\tau(S_1)$ as desired.

Lemma 2.4.5. $\Phi(\mathit{core}(S_1)) \leq 7\mathit{age}^\tau(S_1)$.

Proof. Let $C = \text{core}(S_1)$. By remark 2.4.4, the auxiliary cost of path P is at most $2\text{age}^\tau(C)$. For bounding the value of C2, we first need to prove the following claim:

Claim. *For any inclusion-wise maximal inactive sets S' inside another set S (either active or inactive), $\mathit{age}^\tau(S') < \mathit{age}^\tau(S)/2$.*

First note that since S' is inside S, it means that at a time τ', S' is joined with a moat $\overline{S}$ with core C, or more precisely, at a time τ', there was a vertex $v \in S$ that became tight which was adjacent to S' and $\overline{S}$. However, at time τ', the total load of S' and $\overline{S}$ on v was less than $\frac{3}{2}\tau'$, otherwise the termination condition ($\star$) was satisfied and the phase would have finished. Moreover, $\overline{S}$ was active at time τ', so $\text{age}^{\tau'}(\overline{S}) = \tau'$ and $\text{age}^{\tau'}(\overline{S}) + \text{age}^{\tau'}(S') < \frac{3}{2}\tau'$ it follows that $\text{age}^{\tau'}(S') < \tau'/2 \leq \text{age}^\tau(S)/2$.

By lemma 2.4.3, $c_{S_i}(Q_i)$ is at most $2\text{age}^\tau(S_i)$ for all $1 \leq i \leq m$ since Q_i it is a minimum path in S_i, so, $c_{S_1}(Q_1) + \ldots + c_{S_m}(Q_m)$ is bounded by $2(\text{age}^\tau(S_i) + \ldots + \text{age}^\tau(S_m))$. Using the above claim, we conclude that

$$c_{S_1}(Q_1) + \ldots + c_{S_m}(Q_m) \leq 2\text{age}^\tau(S_1) + \text{age}^\tau(S_2) + \ldots + \frac{\text{age}^\tau(S_m)}{2^{m-1}} \leq 4\text{age}^\tau(S_1),$$

and so C2 is bounded by $4\text{age}^\tau(S_1)$.

Claim. *For every expensive vertex v inside H_S for a moat S (either active or inactive), $\bar{c}(v) - c_S(v) \leq age^{\tau}(S')/2$.*

Note that v is a tight vertex so there was a time τ' that v become tight at time τ' and joined a moat R with core core(C). So R is also adjacent to v. Since the phase did not finish at time τ', the total age of cores in $\mathcal{L}^{\tau'}(v)$ is less than $\frac{3}{2}\tau'$ and since R does not become inactive at time τ', age$^{\tau'}(R) = \tau'$. It follows that total age of cores in $\mathcal{L}^{\tau'}(v) - \{\text{core}(R)\}$ is at most $\tau'/2$, so $\bar{c}(v) - c_S(v) \leq \tau'/2 \leq \text{age}^{\tau}(S')/2$ and the claim is proved.

Using the two claims above, we conclude that the non-auxiliary costs $\bar{c}(z_2) - c_{S_1}(z_2), \ldots, \bar{c}(z_m) - c_{S_{m-1}}(z_m)$ decrease geometrically by factor of two and $\bar{c}(z_2) - c_{S_1}(z_2) \leq \text{age}^{\tau}(S_1)/2$, so C3 is bounded by age$^{\tau}(S_1)$. □

The final remark of this section is that for every initial component C included in the phase tree, the total charge $\Phi(C)$ is bounded by constant factor of age$^{\tau}(C)$.

Also as we discussed, the cost of each vectex is charged to some cores loading it. This proves that the total cost of the phase tree is bounded by the total charges of the cores connected by the dual solution.

Corollary 2.4.6. *In every phase i,*

$$\bar{c}(T^i) \leq O(1). \sum_{C \in \mathcal{C}^i : C \cap T^i \neq \emptyset} \Phi(C)$$

2.4.3 Approximation Factor Guarantee

In this section, we provide the proof of $O(\log n)$-approximation factor for our algorithm. We show that the tree T_r returned by our algorithm is bounded by $O(\log n)$ times one of the dual values constructed during the algorithm in on of the phases. We actually prove a stronger statement in order to obtain the LMP algorithm in next section. We will give formal definition of LMP algorithm and the reason we prove this stronger statement.

For each feasible dual solution y^i found in phase i, we construct a new decreased dual solution $\overline{y}^i$ from y^i by setting the value of $\overline{y}^i_S$ to zero for each set S whose core is not intersecting with final solution T_r (i.e. if $S \cap T = \emptyset$) and we keep the rest of the values unchanged. It is clear that since we have only decreased dual values and y^i is a feasible solution, $\overline{y}^i$ is a feasible solution too. We show that the cost of the tree T_r is bounded by $O(\log n)$ factor of one of the decreased dual solutions rather than the original dual solution. This stronger result helps us in designing the LMP algorithm in the next section.

We show the claim by showing that there are l phases $t_1, \ldots, t_l$ during the algorithm, (not necessarily distinct) such that the cost of the three T_r is bounded by the total of decreased dual solution of these phases and $l = O(\log n)$:

$$\bar{c}(T_r) \leq O(\sum_{i=1}^{l} \sum_{S \subseteq V'} \overline{y}_S^i).$$

This clearly shows that by picking the largest dual solution among $\overline{y}^{t_1}, \ldots, \overline{y}^{t_l}$, we have found the dual solution proving the existence of the desired dual solution.

From the previous section, we know that the cost of each phase tree T^i is bounded by the charges incurred to the cores connected by T^i (corollary 2.4.6). Also it is clear that if a core is connected by a phase tree it is merged with at least one another core or it is connected to tree T_r, so it will never be charged again. So the total cost of T_r is less than the charges of the initial components of different phases joined to T_r. Let us denote the set of initial components in phase i joined the final tree T_r by $\overline{\mathcal{C}}^i$, i.e. $\overline{\mathcal{C}}^i = \{C \in \mathcal{C}^i : C \subseteq T_r\}$. Also, let $\Phi'(S) = \Phi(S)/7$ for avoiding constants during the proof. Therefore, it suffices to show

$$\sum_{C \subseteq T_r} \Phi'(C) = O(\sum_{i=1}^{l} \sum_{S \subseteq V'} \overline{y}_S^{t_i}).$$

The proof steps are as follows. First we partition the initial components included in T_r to l buckets based on their charged value. Then we show that there is a dual solution y^{t_i} for each of the buckets $1 \leq i \leq l$ such that the total charge of the components in bucket i is bounded by total dual values of y^{t_i}. To this aim, we define a potential function $\beta_j(i)$ for each bucket j over phases, which is equal to y^{t_j} at the beginning of phase 1. Then, we show that in each phase i, β_j decreases by at least amount of the total charges of initial components in phase i which are in bucket j.

Let $\Phi'_{\max} = \max_C \Phi'(C)$. A component C is assigned to a bucket j for $1 \leq j < l$ if

$$\frac{\Phi'_{\max}}{2^j} < \Phi'(C) \leq \frac{\Phi'_{\max}}{2^{j-1}},$$

and assigned to bucket l otherwise (when $\Phi'(C) \leq \frac{\Phi'_{\max}}{2^{l-1}}$). Observe that every component is assigned to exactly one bucket and the bucker 1 is not empty. Our choice for dual solutions $\overline{y}^{t_1}, \ldots, \overline{y}^{t_l}$ is:

For each $1 \leq j < l$, let t_j be the the first phase in which a component in bucket j is charged. If no component is in bucket j, then let t_j be any arbitrary phase index. Also, set $t_l = t_1$.

Note that $t_1, \ldots, t_j$ are not necessarily distinct and we do not require as well. Recall that $l = O(\log n)$. So it suffices to show that for each $1 \leq j \leq l$ the total charges of components assigned to each bucket j is bounded by total dual values of solution $\overline{y}^{t_j}$:

$$\sum_{C\text{: assigned to bucket } j} \Phi'(C) \leq \sum_{S \subseteq V'} \overline{y}_S^{t_j}. \tag{2.4}$$

Note that in our partitioning of components to buckets, the Φ' charge of components in each bucket differ at most by factor of two. More precisely, if we define $b_j = \frac{\Phi'_{\max}}{2^j}$, Φ' charge of components are in range b_j and $2b_j$ in each bucket j. If a component C is charged by value $\Phi'(C)$ in some phase, then the growing process has lasted at least for $\Phi'(C)$ time units that phase. Also, in each phase the components are grown uniformly; this means each other component C' is grown at least $\min\{\Phi'(C), \bar{\pi}(C')\} \geq \min\{b_j, \bar{\pi}(C')\}$. Using the fact that the total dual values of moats with core C' is at least $\Phi'(C)$, we conclude that the value $\sum_{C \in \overline{C}^i} \min\{\bar{\pi}(C), b_j\}$ is an upper-bound (within constant factor) for the Φ' charge of components charged between value b_j and $2b_j$ in a phase i. So this gives us the idea of defining the potential function $\beta_j(i)$ for each bucket j and phase i:

$$\beta_j(i) = \sum_{C \in \overline{C}^i} \min\{b_j, \bar{\pi}(C)\}.$$

As noted before, the total charges of components in bucket j in phase i is $O(\beta_j(i))$, however, as we will prove in the rest of this section, the total charges of components in bucket j in phase i does not exceed $O(\beta_j(i) - \beta_j(i+1))$ too. This along with the fact that β_j is a always non-negative, proves that the potential value $\beta_j(t_j)$ can bound the total charges of component in bucket j for all phases.

Before proving the above claim, let us state a useful remark for our future proofs.

Remark 2.4.7. *Let $\bar{\tau}^i$ be the time when phase i terminates. If there is a component $C \in \overline{\mathcal{C}}^i$ which is assigned to bucket j, then $\bar{\tau}^i \geq age^{\bar{\tau}^i}(C) \geq \Phi'(C) \geq b_j$. Moreover, $\bar{\pi}(C) \geq \Phi'(C)$.*

The proof directly follows from lemma 2.4.5 and definitions of b_j and ϕ'. The following lemma proves formally why $\beta_j(t_j)$ does not exceed the dual solution $\overline{y}^{t_j}$.

Lemma 2.4.8. $\beta_j(t_j) \leq \sum_S \overline{y}_S^{t_j}$.

Proof. By Remark 2.4.7 above, the time $\tau := \bar{\tau}^{t_j}$ when phase t_j terminates is at least b_j. So for every component $C \in \overline{C}^{t_j}$ we have that the age of C at time τ is equal to $\min\{\bar{\pi}(C), \tau\} \geq \min\{\bar{\pi}(C), b_j\}$ and therefore,

$$\sum_S \overline{y}_S^{t_j} \geq \sum_{C \in \overline{\mathcal{C}}^{t_j}} \text{age}^{\tau}(C) \geq \sum_{C \in \overline{\mathcal{C}}^{t_j}} \min\{b_j, \bar{\pi}(C)\} = \beta_j(t_j).$$

□

The following lemma proves correctness of the key idea we discussed above: the function $\beta_j(i)$ decreases proportional to the charges assigned to components in bucket j in each phase i. Note that no component is charged at last phase and we prove the statement for budget l separately. The following lemma states the claim formally.

Lemma 2.4.9. *Assume that for a bucket $j < l$, there are k components in $\overline{\mathcal{C}}^i$ assigned to bucket j in a phase $i < m$. Then,*

$$\beta_j(i) - \beta_j(i+1) \geq \frac{k}{2} b_j$$

Proof. Let τ be the time phase i finishes and T^i be the subtree constructed in phase i. Let $\overline{\mathcal{C}}(T^i)$ be the set of initial components in $\overline{\mathcal{C}}^i$ connected by T^i, i.e. $\overline{\mathcal{C}}(T^i) = \{C \in \overline{\mathcal{C}}^i : C \subseteq T^i \neq \emptyset\}$. First note that $\beta_j(i)$ is a non-increasing function over i for a fixed j, since in each phase, some components are replaced by their union (and possibly other vertices), therefore it is easy to see that sum of min() functions does not increase. Therefore, if $k = 0$, the statement holds.

Now assume that $k \geq 1$. The only difference between $\overline{\mathcal{C}}^i$ and $\overline{\mathcal{C}}^{i+1}$ is that the components in $\overline{\mathcal{C}}(T^i)$ are removed from $\overline{\mathcal{C}}^{i+1}$ and instead, either T^i is added to $\overline{\mathcal{C}}^{i+1}$ as a new component or T^i is added to T_r. In either case, we have,

$$\begin{aligned}
\beta_j(i) - \beta_j(i+1) &= \sum_{C \in \overline{\mathcal{C}}^i} \min\{b_j, \bar{\pi}(C)\} - \sum_{C \in \overline{\mathcal{C}}^{i+1}} \min\{b_j, \bar{\pi}(C)\} \\
&\geq \sum_{C \in \overline{\mathcal{C}}(T^i)} \min\{b_j, \bar{\pi}(C)\} - \min\{b_j, \bar{\pi}(T^i)\}
\end{aligned}$$

Now we consider two cases depending on value of k.
If $k > 1$, let $\overline{C} \in \overline{\mathcal{C}}^i$ be any component assigned to bucket j. Then, $\bar{\pi}(T^i) \geq \bar{\pi}(\overline{C}) \geq \Phi'(\overline{C}) \geq b_j$, by Remark 2.4.7, and therefore:

$$\beta_j(i) - \beta_j(i+1) \geq \Big(\sum_{C \in \overline{\mathcal{C}}(T^i) : \Phi'(C) \geq b_j} b_j \Big) - b_j$$

$$\geq kb_j - b_j \geq (k-1)b_j \geq \frac{k}{2}\, b_j.$$

If $k = 1$, again $\bar{\pi}(T^i) \geq b_j$. Since $k = 1$, there is exactly one component $\overline{C} \in \overline{\mathcal{C}}^i$ merging into T^i that is assigned to bucket j, and we have $\bar{\pi}(\overline{C}) \geq \Phi'(C) \geq b_j$. Moreover, for every $C \in \overline{C}(T^i)$ we have $\min\{\bar{\pi}(C), b_j\} \geq \min\{\text{age}^\tau(C), b_j\}$ and since the phase termination condition ($\star$) is satisfied, we have:

$$\begin{aligned}
\beta_j(i) - \beta_j(i+1) &\geq \big(\min\{\bar{\pi}(\overline{C}), b_j\} + \sum_{C \in \overline{\mathcal{C}}(T^i) - \overline{C}} \min\{\bar{\pi}(C), b_j\}\big) - b_j \\
&\geq (b_j + \sum_{C \in \overline{\mathcal{C}}(T^i) - \overline{C}} \min\{\text{age}^\tau(C), b_j\}) - b_j \\
&\geq \min\{\sum_{C \in \overline{\mathcal{C}}(T^i) - \overline{C}} \text{age}^\tau(C), b_j\} \\
&= \min\{\sum_{C \in \overline{\mathcal{C}}(T^i)} \text{age}^\tau(C) - \text{age}^\tau(\overline{C}), b_j\} \\
&\geq \min\{\frac{3}{2}\,\tau - \tau, b_j\} \geq \frac{1}{2}\, b_j \geq \frac{k}{2}\, b_j
\end{aligned}$$

□

Note that for each component $C \in \overline{\mathcal{C}}^i$ assigned to a bucket j, we have $\Phi'(C) \leq 2b_j$. Therefore, using the previous lemma, for each phase i and bucket j we also have,

$$\sum_{C \in \overline{\mathcal{C}}^i\text{: assigned to bucket } j} \Phi'(C) \leq 4\big(\beta_j(i) - \beta_j(i+1)\big) \tag{2.5}$$

Now using the above inequality and lemma 2.4.8, we can prove inequality 2.4. For a bucket $j < l$, we have:

$$\begin{aligned}
\sum_{C\text{: assigned to bucket } j} \Phi'(C) &= \sum_{i=1}^{m-1} \sum_{C \in \overline{\mathcal{C}}^i\text{: assigned to bucket } j} \Phi'(C) \\
&\leq \sum_{i=1}^{m-1} 4\big(\beta_j(i) - \beta_j(i+1)\big) \\
&= 4\big(\beta_j(t_j) - \beta_j(m)\big) \\
&\leq 4 \sum_{S \subseteq V'} \overline{y}_S^{t_j}.
\end{aligned}$$

It remains to prove the bound for bucket l. At the beginning of the algorithm we have at most n components and in each phase (except the last phase) at least two of these components are merged together, so we have at most $2n$ distinct component through out the algorithm. For last bucket, recall that we chose $t_l = t_1$ and there is a component C' with $\Phi'(C') = \Phi'_{\max}$ in first bucket, so $\sum_{S \subseteq V'} \overline{y}^{t_l} \geq \Phi'_{\max}$. Moreover, for each component C in bucket l, we have $\Phi'(C) \leq \frac{\Phi'_{\max}}{2^l} \leq \frac{\Phi'_{\max}}{n/2}$ thus $\sum_{C\text{: assigned to bucket } j} \Phi'(C) = \sum_{i=1}^{m-1} \leq 2n\frac{\Phi'_{\max}}{n/2} \leq 4\sum_{S \subseteq V'} \overline{y}^{t_l}$ and we are done.

Using the discussion above, we proved that the total charges of components in each bucket j is bounded by a dual solution $\overline{y}^{t_j}$ and we have $\log n$ buckets, so the maximum dual solution among these is within $O(\log n)$ factor of total charges. Also, the cost of the tree returned by algorithm is within constant factor of the total charges, so we have the following upper bound for cost of tree.

Theorem 2.4.10. *Let T be the tree returned by the algorithm and let y^* be the dual solution to (D) among $\overline{y}^1, \ldots, \overline{y}^m$ with maximum value. Then*

$$\bar{c}(T) \leq O(\ln n) \sum_{S \subseteq V'} y^*_S.$$

Finally, we need to give an upper bound for penalty of the vertices excluded from the final solution tree T_r returned by algorithm. This can be simply obtained by the dual solution y^m of the last phase of the algorithm. In the last phase, every moat gets deactivated, and no two active components get merged (otherwise the phase termination condition ($\star$) would have been satisfied) so for each initial component $C \in \mathcal{C}^m$, the total dual solution of moats with core C is equal to $\bar{\pi}(C)$. Also, all cheap vertices in $V' \setminus T_r$ are included in one initial component in last phase, therefore we have the result shown in following lemma.

Lemma 2.4.11. *Let y^m be the dual solution found in the last phase of the algorithm, then every component S in support of this solution is disjoint from T_r and*

$$\bar{\pi}(V' \setminus T_r) \leq \sum_S y^m_S$$

The approximation bound now follows.

Theorem 2.4.12. *Let OPT be the value (cost plus penalty) of the optimal solution for Prize Collecting Steiner Tree problem on an instance of the problem with n vertices, then, Algorithm 1 finds a solution T_r such that $c(T_r) + \pi(V' \setminus T_r) = O(\ln n)OPT$.*

Proof. Recall that given a feasible solution y to the dual (D), setting $p_v = c(v)$ if v is cheap, and $p_v = \pi(v)$ otherwise, yields a feasible solution (y, p) to the dual (D^0). Using weak duality together with Theorem 2.4.10 and Lemma 2.4.11, we have

$$c(T) + \pi(V' \setminus T) \leq \sum_{v \in V'} p_v + \bar{c}(T) + \bar{\pi}_r(V' \setminus T)$$

$$= O(\ln n)OPT + OPT = O(\ln n)OPT$$

□

2.5 LMP Algorithm

Suppose that we have an algorithm for a prize collecting problem such as NW-PCST problem in which a solution to problem is a subset F of a set V (here set of vertices) and there is a cost $c(v)$ and $\pi(v)$ associated to each element and. Suppose that the objective is to minimize $c(F)+\pi(V' \setminus F)$. A *Lagrangian Multiplier Preserving* (LMP) α-approximation algorithm for such a prize collecting problem is an algorithm that finds a solution F for the problem such that

$$c(F) + \alpha\pi(V \setminus F) \leq \alpha OPT.$$

Obtaining an LMP algorithm is both essential for solving quota variant of the problem and hard to obtain in comparison to a non-LMP α-approximation. We will define the quota problem formally in the next section as an application of the NW-PCST problem and will show how the LMP property of the NW-PCST algorithm help us in solving the quota problem. In this section, we show how a simple approach can lead us to an LMP $O(\lg(n))$-approximation algorithm for NW-PCST problem from algorithm 1.

Theorem 2.5.1. *There is an LMP $O(\log n)$-approximation algorithm for the NW-PCST problem.*

Proof. Consider the following algorithm: *Let $\pi'(v) = 2\pi(v) - c(v)$ for every cheap vertex v and $\pi'(v) = \pi(v)$ for every expensice vertex v. Run the algorithm 1 for these values of penalties and the original costs. Let T be the output of this run; return T as the result*

Note that if we show the reduced penalties of the new instance of problem by $\bar{\pi}'$, then $\bar{\pi}'(v) = 2\bar{\pi}(v)$ for every vertex. Suppose that 1 is an α-approximation algorithm for NW-PCST problem. By theorem 2.4.10, we obtain a dual solution y^*, constructed in some phase i of the algorithm which has a value within α factor of optimal solution. Let $\overline{y}$ be a

dual solution obtained from y^* by setting the value $\overline{y}_S$ for each component $S \subseteq V'$ equal to y_S if $\text{core}(S) \subseteq T$ and equal to 0 otherwise. We also showed that $\overline{y}$ is within α factor of optimal solution, i.e.

$$\bar{c}(T) \leq \alpha \sum_{S \subseteq V' : \text{core}(S) \subseteq T} \overline{y}_S. \tag{2.6}$$

Let z be the dual solution found in the last phase of the algorithm, then by Lemma 2.4.11,

$$\bar{\pi}'(V' \setminus T) = 2\bar{\pi}(V' \setminus T) \leq \sum_{S \subseteq V'} z_S. \tag{2.7}$$

Claim. $y'_S = \frac{\overline{y}_S}{2} + \frac{z_S}{2} : S \subseteq V'$ is a feasible solution for the instance of the problem with original penalties π.

Note that for constraint 2.2, for every vertex $v \in V'$, we have

$$\sum_{S | v \in \Gamma(S)} \overline{y}_S \leq \bar{c}(v), \sum_{S | v \in \Gamma(S)} z_S \leq \bar{c}(v) \Rightarrow \sum_{S | v \in \Gamma(S)} y'_S = \sum_{S | v \in \Gamma(S)} \frac{\overline{y}_S + z_S}{2} \leq \bar{c}(v),$$

therefore constraint 2.2 is satisfied by solution y'. So it suffices to prove that constraint 2.3 is satisfied too. Let $\mathcal{C}$ be the set of the initial components at the beginning of phase i (in which solution y^* was constructed) and $\mathcal{D}$ be the set of the initial components at the beginning of the last phase (in which z was constructed). Each component in support of $\overline{y}$ and z has a single core and the age of a component at the end of the phase is no greater than reduced penalty of its core. Moreover, we know the core of each component in support of $\overline{y}$ is contained in T, so for each $S \subseteq V'$ we have,

$$\begin{aligned}
\sum_{R \subseteq S} y'_R &= \sum_{R \subseteq S} (\overline{y}_R/2 + z_R/2) = \sum_{R \subseteq S} \overline{y}_R/2 + \sum_{R \subseteq S} z_R/2 \\
&\leq \sum_{C \in \mathcal{C} : C \subseteq S \cap V(T)} \bar{\pi}'(C)/2 + \sum_{D \in \mathcal{D} : D \subseteq S} \bar{\pi}'(D)/2 \\
&= \sum_{C \in \mathcal{C} : C \subseteq S \cap V(T)} \bar{\pi}(C) + \sum_{D \in \mathcal{D} : D \subseteq S} \bar{\pi}(D)
\end{aligned}$$

Note that z is the last dual solution constructed by algorithm, so the set of cheap vertices of every core in $\mathcal{D}$ is the union of the set of cheap vertices of some other cores in $\mathcal{C}$. For each component Y in support of $\overline{y}$, $\text{core}_{\overline{y}}(Y)$ is merged into T at some phase. Moreover, all components of the last phase of the algorithm get deactivated and none of them contain any vertex of T, thus, none of them contains a core of a component in support of $\overline{y}$. Hence, no component in $\mathcal{C}$ which resides in S is appearing more than once in the summation above, and so,

$$\sum_{R \subseteq S} y'_R \leq \sum_{C \in \mathcal{C}: C \subseteq S} \bar{\pi}(C) = \bar{\pi}(S).$$

Therefore, the penalty constraint holds for S and the claim is proved.

Adding $\alpha \lg(n)$ times the inequality (2.7) to the inequality (2.6), we get

$$\begin{aligned}
\bar{c}(T) + 2\alpha\bar{\pi}(V' \setminus T) &\leq \alpha \sum_{S \subseteq V'} \overline{y}_S + \alpha \sum_{S \subseteq V'} z_S \\
&\leq 2\alpha\Big(\sum_{S \subseteq V'} \frac{\overline{y}_S}{2} + \sum_{S \subseteq V'} \frac{z_S}{2} \Big) = 2\alpha \sum_{S \subseteq V'} y'_S \\
\Rightarrow c(T) + 2\alpha\pi(V' \setminus T) &\leq 2\alpha\Big(\sum_{S \subseteq V'} y'_S + \sum_{v \in V'} p_v \Big) \\
&\leq 2\alpha OPT.
\end{aligned}$$

The last inequality follows from weak duality and the fact that (y', p) is feasible solution if $p_v = c_v$ for cheap vertices and $p_v = \pi(v)$ expensive ones. So the algorithm given is an LMP $O(\log n)$-approximation algorithm since $\alpha = O(\log n)$. □

Chapter 3

Applications

3.1 Node-Weighted Quota Steiner Tree Problem

As a theoretical application of our algorithm for Node-Weighted Prize Collecting Steiner Tree problem given in previous chapter, we discuss the *quota* version of the problem here.

Definition 3.1.1. *Given a graph G, costs $c(v)$ and profit $\pi(v)$ for every vertex $v \in V(G)$ and a quota value $Q \geq 0$, the Quota Node Weighted Steiner Tree problem (shortly the quota problem) asks to find a minimum cost subtree of G with total profit not less than Q.*

For a vertex $r \in V(G)$, the rooted version of the problem is defined similarly, asking to find a tree containing r. By iterating over all possible values for Q, we can solve the Prize Collecting problem using an algorithm for the quota problem, preserving the approximation factor. However, the other direction is not trivial. We observe that in order to solve the quota problem using an algorithm for prize collecting problem, LMP property plays a key role in the reduction.

In [32], Chudak et al., show that how a primal dual LMP approximation algorithm for prize collecting network design problems, namely for the Edge Weighted version of our Prize Collecting problem, can be used by a Langrangean Relaxation (LR) method to obtain an approximation algorithm for the quota version of the problem (preserving the approximation factor within a constant factor). Using a similar framework and our LMP algorithm for NW-PCST, we give an $O(\lg n)$ approximation algorithm for the quota problem.

Theorem 3.1.2. *There is an $O(\log n)$-approximation algorithm for Quota Node Weighted Steiner Tree Problem.*

Let us first formulate the quota problem with the following linear program.

$$
\begin{aligned}
\min \quad & \sum_{v \in V'} c(v) x_v && && \text{(Quota)} \\
\text{s.t.} \quad & \sum_{v \in \Gamma(S)} x_v + \sum_{U | S \subseteq U} z_U \geq 1 && \forall S \subseteq V', \\
& x_v + \sum_{U | v \in U} z_U \geq 1 && \forall v \in V', \\
& \sum_{S \subseteq V'} \pi(S) z_S \leq \sum_{v \in V'} \pi(v) - Q && && (3.1) \\
& x_v \in \{0,1\} && \forall v \in V', \\
& z_S \in \{0,1\} && \forall S \subseteq V',
\end{aligned}
$$

The differences between the formulation above and the linear program P for prize collecting version are the added quota constraint 3.1 and that in prize collecting problem, we have penalty values in our objective function whereas here we do not.

Assume that we ignore the quota constraint and we allow all the solutions to deviate from the requested profit value Q and in return, incur a penalty equal to λ times the *profit deviation* of the solution from the desired value Q for an arbitrary real value λ. Let $\Pi = \sum_{v \in V'} \pi(v)$. The profit deviation of a primal solution (x, z) is defined as $\sum_{S \subseteq V'} \pi(S) z_S - (\Pi - Q)$, which is the difference between profit of the solution in comparison to Q (a negative value if the solution has profit more than Q). This modification to our linear program gives us the Lagrangian Relaxation of the quota problem:

$$
\begin{aligned}
\min \quad & \sum_{v \in V'} c(v) x_v + \lambda \Big(\sum_{S \subseteq V'} \pi(S) z_S - (\Pi - Q) \Big) && && \text{(LRQuota)} \\
\text{s.t.} \quad & \sum_{v \in \Gamma(S)} x_v + \sum_{U | S \subseteq U} z_U \geq 1 && \forall S \subseteq V', \\
& x_v + \sum_{U | v \in U} z_U \geq 1 && \forall v \in V', \\
& x_v \in \{0, 1\} && \forall v \in V', \\
& z_S \in \{0, 1\} && \forall S \subseteq V',
\end{aligned}
$$

Writing the dual of the LRQuota, we get the following linear program denoted by LRQuota-D:

$$
\begin{aligned}
\max \quad & \sum_{S \in V'} y_S + \sum_{v \in V'} p_v - \lambda(\Pi - Q) && && \text{(LRQuota-D)} \\
\text{s.t.} \quad & \sum_{S | v \in \Gamma(S)} y_S + p_v \leq c(v) && \forall v \in V' \\
& \sum_{U \subseteq S} y_S + \sum_{v \in S} p_v \leq \sum_{v \in S} \lambda \pi(v) && \forall S \subseteq V' \\
& y \geq \mathbb{0} \\
& p \geq \mathbb{0}
\end{aligned}
$$

Note that for any arbitrary λ, every solution (x, z) to the quota problem is a feasible solution for the relaxed version, so the optimal solution for the LRQuota problem is a lower bound for the quota problem. Therefore, by weak duality, the value of any solution to the dual problem LRQuota-D is also not larger than the optimal solution of the quota problem.

The dual formulation LRQuota-D differs from the dual of the price collecting problem only in the constant value $-\lambda(\Pi - Q)$ added to objective function, so if we give an instance of the quota problem with penalty values $\pi'(v) = \lambda \pi(v)$ for each $v \in V'$, by the LMP property of the prize collecting algorithm (theorem 2.5.1), it gives us a feasible solution T with a dual solution (y, p) to LRQuota-D such that:

$$
\sum_{v \in T} c_v + \alpha(n) \pi'(V' \setminus T) \leq \alpha(n) \Big(\sum_{S \subseteq V'} y_S + \sum_{v \in V'} p_v \Big)
$$

Where $\alpha(n)$ is a function of order $\lg n$. For ease of notation let us denote the value $\sum_{S\subseteq V'} y_S + \sum_{v\in V'} p_v$ of the dual solution (y,p) by Y and denote $V' \setminus T$ by $\bar{T}$. So we have:

$$\sum_{v\in T} c_v + \alpha(n)\lambda\pi(\bar{T}) \leq \alpha(n)Y. \tag{3.2}$$

Subtracting $\alpha(n)\lambda(\Pi - Q)$ from both sides, we get:

$$\begin{aligned}\sum_{v\in V'} c_v + \alpha(n)\lambda(\pi(\bar{T}) - \Pi + Q) &\leq \alpha(n)(Y - \lambda(\Pi - Q)) \\ &\leq \alpha(n)OPT_Q, \end{aligned}\tag{3.3}$$

where the last inequality follows from the fact that the right hand side is the objective function of the dual LP LRQuota-D multiplied by $\alpha(n)$. For every λ, the prize collecting algorithm finds a feasible solution for the relaxed quota problem. If we can find a suitable λ for which this solution has profit exactly equal to Q, then we have a feasible solution for the original quota problem and we are done. However the solution may have profit less than or greater than Q. If we run the prize collecting algorithm for the λ values equal to 0 and a value large enough we find two solutions with profit 0 and Π respectively. We may assume that $0 \leq Q \leq \Pi$, so there are always two values of λ, say λ_1 and λ_2 and by binary searching we can make these two values close enough such that:

- The prize collecting algorithm gives us two solutions T_1 and T_2 and their corresponding dual solutions $(y^{(1)}, p^{(1)})$ and $(y^{(2)}, p^{(2)})$ for the relaxed quota problem with λ value equal to λ_1 and λ_2 respectively, such that $\pi(T_1) \leq Q$ and $\pi(T_2) \geq Q$.
- $\lambda_2 - \lambda_1 \leq \frac{OPT_Q}{\Pi}$.

Note that we do not need to know the exact value of the OPT_Q, instead, we can continue the binary search until $\lambda_2 - \lambda_1$ is less than a value which is certainly less than OPT_Q such as the minimum non-zero cost of all vertices. We enumerate over all possible values of OPT_Q and continue as follows. At first, remove every vertex with node-cost distance more than OPT_Q from r. Clearly no such vertex exists in the optimal solution. Obtain two solutions T_1 and T_2 for the relaxed quota prblem with the above properties and *merge* them as we will explain later. Before getting to the merging procedure, we prove that a good convex combination of the cost of the two solutions T_1 and T_2 is no greater than $O(\lg n)OPT_Q$.

Let a_1 and a_2 be two real values such that $a_1 + a_2 = 1$ and they are proportional to the profit deviation of the solution T_2 to the solution T_1. More precisely, let

$$a_1 = \frac{\pi(T_2) - Q}{\pi(T_2) - \pi(T_1)}, \quad a_2 = \frac{Q - \pi(T_1)}{\pi(T_2) - \pi(T_1)}.$$

Lemma 3.1.3. $a_1 c(T_1) + a_2 c(T_2) \leq O(\lg n) OPT_Q$.

Proof. Since $a_1 + a_2 = 1$, the convex combination $(a_1 y^{(1)} + a_2 y^{(2)}, a_1 p^{(1)} + a_2 p^{(2)})$ is a feasible solution for LRQuota-D with λ equal to λ_2 (note $\lambda_2 \geq \lambda_1$) and therefore $a_1 Y^{(1)} + a_2 Y^{(2)} - \lambda_2(\Pi - Q) \leq OPT_Q$. By equation 3.2 we have:

$$\begin{aligned} & a_1 c(T_1) + a_2 c(T_2) \\ \leq\ & \alpha(n)\big(a_1 Y^{(1)} + a_2 Y^{(2)} - (a_1 \lambda_1 \pi(\bar{T}_1) + a_2 \lambda_2 \pi(\bar{T}_2))\big) \\ \leq\ & \alpha(n)\big(a_1 Y^{(1)} + a_2 Y^{(2)} - \lambda_2(a_1 \pi(\bar{T}_1) + a_2 \pi(\bar{T}_2)) + (\lambda_2 - \lambda_1)\pi(\bar{T}_2)\big) \end{aligned}$$

By the definition of a_1 and a_2, we have $a_1 \pi(\bar{T}_1) + a_2 \pi(\bar{T}_2) = a_1(\Pi - \pi(T_1)) + a_2(\Pi - \pi(T_2)) = \Pi - Q$, so,

$$\begin{aligned} a_1 c(T_1) + a_2 c(T_2) &\leq \alpha(n)\big(OPT_Q + (\lambda_2 - \lambda_1)\pi(\bar{T}_2)\big) \\ &\leq \alpha(n)\big(OPT_Q + \frac{OPT_Q}{\Pi}\Pi\big) \\ &= 2\alpha(n) OPT_Q. \end{aligned}$$

□

It only remains to show that given the trees T_1 and T_2, how to merge them to find a tree with profit at least Q and cost $O(\lg n) OPT_Q$.

Our final solution consists of the vertices of T_1 supplemented by some vertices of T_2 with profit at least $q = Q - \pi(T_1)$. First, identify all of the vertices of T_1 to a vertex r' to obtain a graph G'. Let T_2' be the corresponding vertices of T_2 in G'. Note that both T_1 and T_2 are connected subgraphs in G, so the vertices of T_2', which are $V(T_1) \setminus V(T_2) \cup \{r'\}$, form a connected subgraph of G' and also we have $\pi(T_2') \geq \pi(T_2) - \pi(T_1) \geq q$. We call a subgraph S *cost-effective*, if $\frac{c(S)}{\pi(S)} \leq \frac{c(T_2')}{\pi(T_2')}$. Starting with the cost-effective tree $T' = V(T_2')$ in G', continue the following procedure (which keeps T' a cost-effective connected subgraph) while it changes the graph T':
Select an edge e from T' with non-leaf vertices at both ends, let S and R be the two components of T' after removing the edge e. At least one of S or R are cost-effective. For each cost-effective subgraph S or R, say S, do the following:

1. If $\pi(S) \geq q$, remove R from T'.

2. If $\pi(S) < q$, identify all vertices of S to a super node S and set the cost and the profit of the super vertex S equal to the total cost and total profits of the vertices of S, respectively.

It is easy to see that at the end, T' is a cost-effective star graph with profit at least q.

Lemma 3.1.4. *There is a connected subgraph T'' of T' with profit at least q and cost at most $O(\lg n)OPT_Q$.*

Proof. If a set S is cost-effective, and has size no more than $O(1)q$, then by the previous lemma and the definition of a cost-effective set we have:

$$c(S) \leq \pi(S)\frac{c(T'_2)}{\pi(T'_2)} \leq O(1)q\frac{c(T'_2)}{\pi(T'_2)} \leq O(1)(Q-\pi(T_1))\frac{c(T'_2)}{\pi(T'_2)}$$

$$\leq O(1)(Q-\pi(T_1))\frac{c(T'_2)}{\pi(T_2)-\pi(T_1)} \leq O(1)a_2c(T'_2)$$

$$\leq O(1)a_2c(T_2) = O(\lg n)OPT_Q.$$

During the procedure above, we never identify a subgraph to a super vertex if the subgraph has profit more than q or if it is not cost-effective. Therefore, every super vertex S is cost-effective and has cost $O(\lg n)OPT_Q$. Moreover, at the beginning, we removed every vertex with cost more than OPT_Q. So regardless of a vertex being an original vertex or a super vertex, it does not have cost more than $O(\lg n)OPT_Q$.

If there is a vertex v in T' with profit at least q, the single vertex v is the desired subgraph. Also, if the total profit of T' is not more than $2q$, then since T' is cost effective T' has cost $O(\lg n)OPT_Q$ and we are done. Now suppose that every vertex has profit less than q and T' has profit more than $2q$, then T' has at leas 3 vertices. Suppose that T' consists of a vertex u of degree more than 1 and all other vertices in T' are leaves adjacent to u. For an arbitrary vertex $v \in V(T') \setminus \{u\}$, by removing the edge uv from T', we have a single vertex v and a star graph R with profit more than q (a star graph is a tree with only one vertex of degree more than one). Since none of the cases 1 and 2 were applicable, R is not cost-effective, hence $\{v\}$ is cost-effective. By our arbitrary choice of v, we conclude that every vertex in T' except u is cost-effective.

Consider an empty set U and add adjacent vertices of u in T' to U until $U \cup \{u\}$ induces a subgraph of T' with profit at least q. Note that U does not have profit more than $2q$ and it is cost effective. So by the arguments above, $c(U \cup \{u\}) = O(\lg n)OPT_Q$ and it is connected and we are done. □

Finally, if the supplemental subgraph T'' provided by the procedure above and previous lemma is not connected to the root, add a minimum cost path P from root to T''. The graph $T_1 \cup T''$ has profit at least $q + \pi(T_1) = Q$ and cost $O(\lg n)OPT_Q$ and satisfies our quota problem constraints. Note that P has cost at most OPT_Q since we have removed every vertex from the graph with node-cost distance more than OPT_Q at the beginning.

3.2 Technology Diffusion Problem

This section is dedicated to study an application of the quota problem, particularly in solving an optimization problem in the field of social network analysis. The problem itself is important from a theoretical point of view since as we will show, it can not only be solved using the quota problem, but also it is equivalent to a special case of the quota problem. Here we first give a brief overview of the problems in the field and then define the problem formally.

Consider a newly evolved technology being gradually propagated in a network based on the word-of-the-mouth phenomenon in which each node's utility in adoption of the technology increases as the number of adopters in his *community* increases. This process can be formulated as a dynamic process over time where each node is being influenced by its acquaintances to migrate to the new technology. This approach has been the basis for modeling propagation of a new product in viral marketing [34, 78], spread of information and ideas in social networks [23, 30, 8, 89, 50] and in modeling the diffusion of a network technology upgrade [70, 24, 68] and many other dynamic processes in networks.

In such a propagation process, the final number of adopters is mainly affected by two factors: the underlying propagation mechanism, or how the individuals are being influenced by others and the initial set of individuals who triggered the process. There have been a vast study to model the propagation mechanisms in different contexts, for example in [84, 24, 68, 70] dynamics of adoption of a new network technology has been studied or [49, 73] studied the propagation of influence in social networks as an application of viral marketing. Both in viral marketing and planning for deployment of a new technology, *influence maximization* is one of the most natural optimization problems in the field: given the underlying propagation mechanism, what is the smallest set of nodes to choose as the early adopters of the technology so that at the end of the propagation process, they cause the technology to become widely adopted?

Unfortunately, influence maximization problem has been proved to be a hard task in majority of the proposed models [29, 26, 1]. Therefore, many have tried to design approximation algorithms for different variations of the problems, e.g. [73] studied the Linear

Threshold and Independent Cascade models and [1, 51] studied combinatorial models of the problem.

We focus on the problem of finding the minimum initial set of technology adopters to make a technology de facto based on a model proposed by Goldberg and Liu ([48]) which captures the propagation model of communication technologies. Their model follows the threshold models suggested by many studies such as the seminal work of Kempe, Kleinberg and Tardos in 2003 ([73]) and others such as [24, 52, 53, 92]. However, while most studies consider only the influence from one's direct connections, Goldberg-Liu's model is motivated by the fact that in many real world scenarios, one's decisions is impacted by the total number of people who have adopted technology and are reachable through other influenced users' connections rather than number of direct acquaintances.

The Goldberg-Liu model targets a natural case lacking in the previous studies which is evident in cases where the technology itself enhances the communication e.g. adoption of ipv6 over ipv4 [54, 68], deployment of new security protocols over internet [84, 7, 44, 87] and introducing a new mobile phone service or social networking application. Specifically, in deployment of a new security protocol or enhancement in service quality over internet, a node benefits from the enhanced connection with another node if there is a path between these two such that all other internal nodes are implementing the new protocol or upgraded to the new technology; the same measurement the Goldberg-Liu's model uses as the utility of the node in adopting the technology.

We are interested in finding an initial set of nodes leading to a complete adoption. This can be justified in an upgrade from old technologies to new technologies such as an upgrade from ipv4 to ipv6, since the goal is to reach a state in which every one has adopted the new technology. The problem itself is also interesting from theoretical point of the view as we will establish a close connection between the problem and the Node-Weighted Quota Steiner Tree problem studied in previous chapter and also the *submodular Set Cover Problem*.

3.2.1 Problem Definition

Let us give a more precise definition of the technology diffusion process and its affiliated optimization problem. Given a network of individuals and connections between them as a graph $G(V, E)$ and a threshold function $\theta : V \to \mathbb{Z}$, we want to select a subset $\mathcal{A}_0 \subseteq V(G)$ of initial nodes called the *seed set*, to be the set of early adopters of the technology. We alternatively call a node who has adopted the technology as an activated node. For each integer value $t \geq 0$ representing time, let $\mathcal{A}_t$ be the set of activated nodes at time t. The

technology diffusion process is defined as follows: At time zero, the vertices in $\mathcal{A}_0$ are active and all other vertices are deactivated. At each time $t \geq 1$, a vertex v is active if either it was active at time $t-1$ or the size of the connected component containing v in the induced subgraph by vertices in $\mathcal{A}_{t-1} \cup \{v\}$ is at least $\theta(v)$. The smallest t for which a vertex v is activated is called the activation time of the vertex v. Note that we consider the progressive propagation i.e. if one becomes activated, it will never change its state to deactivated later.

The *technology diffusion optimization problem* is the problem of finding minimum cardinality seed set $\mathcal{A}_0$ that activates all of the vertices at the end of the process i.e. for some $t \geq 0$, $\mathcal{A}_t = V$. Through this section, we assume $\text{OPT} \subseteq V$ is an optimal solution for the technology diffusion optimization problem in graph G and r is the diameter of the graph G (the maximum distance between any two vertices in G) and $\theta : V \to \{\theta_1, \ldots, \theta_l\}$ such that $\theta_1 < \theta_2 < \ldots < \theta_l \leq |V|$ i.e. there are l distinct threshold values. Also, we assume that for each θ_i $(1 \leq i \leq l)$, there exists a vertex with threshold θ_i.

3.2.2 Previous Result and Our Contributions

Goldberg and Liu gave a reduction from set cover to the technology diffusion optimization problem showing that asymptotically, a $\lg(n)$-approximation algorithm is the best achievable algorithm for the problem. In the same paper they propose an $O(r.l.\log n)$-approximation algorithm. We improve the approximation factor to $O(\min\{r, l\}.\lg(n))$ by giving two algorithms, an $O(l.\log n)$-approximation algorithm using the node-weighted version of Quota Steiner Tree problem and an $O(r.\log n)$-approximation algorithm based on submodular set cover maximization problem.

While the algorithm proposed in [48] uses randomized LP rounding and needs to solve an LP, both of our algorithms are based on combinatorial ideas and are simpler than the previous one. We also give an inapproximability for the problem by providing a reduction from Node-Weighted Quota Steiner Tree problem which is essentially a harder problem than the set cover problem even for the case where we have only two threshold values. This suggests that finding an $O(\log n)$-approximation algorithm for the problem probably requires substantially new ideas as to the best of our knowledge, there is no known $O(\log n)$-approximation algorithm for the Node-Weighted Quota Steiner Tree problem with more than $O(1)$ quota constraints.

3.2.3 $O(r.\lg(n))$-Approximation Algorithm

In this section we give an $O(r.\log n)$-approximation for the technology diffusion problem using the submodular set covering problem. Let us first show a lower bound for the optimal solution. Consider an initially activated set $\mathcal{A}_0$ of nodes and the technology diffusion process started from $\mathcal{A}_0$. Suppose that we call the time span from starting point of the process till the time there are $\theta_2 - 1$ active nodes is called the first phase, then the time span the number of active nodes grows from θ_2 nodes to $\theta_3 - 1$ is called the second phase and so on.

Note that $\mathcal{A}_0$ must have at least $\theta_1 - 1$ nodes, otherwise, none of the nodes get activated. Also, in each phase i, only nodes with threshold θ_i or less activate and at the end of phase i, we must have at least $\theta_{i+1} - 1$ active nodes to be able to active nodes with greater thresholds than θ_i. This gives us an idea to show a lower bound for $\mathcal{A}_0$. In each phase i, assuming S_i is the set of all nodes reachable from $\mathcal{A}_0$ by nodes with threshold at most θ_i, then $|S_i|$ must be at least $\theta_{i+1} - 1$ for each $1 \leq i < l$.

For proving the above illustrated idea, we first define some notations. For each $v \in V$ and each threshold value θ_i, let $G_v^{\theta_i}$ be the subgraph induced by v and all vertices of G with threshold at most θ_i, i.e.

$$G_v^{\theta_i} = G[\{v\} \cup \{u | \theta(u) \leq \theta_i\}].$$

Where $G[S]$ denotes the subgraph of G induced by vertices S. Note that $G_v^{\theta_i}$ may be disconnected. Let $\Gamma(\theta_i, v)$ be the set of vertices of the connected component of $G_v^{\theta_i}$ containing v. Note that, by definition, $v \in \Gamma(\theta_i, v)$. Similarly, for every $S \subseteq V$, define $\Gamma(\theta_i, S) = \bigcup_{v \in S} \Gamma(\theta_i, v)$. Also, let $\theta_0 = 0$, so that $\Gamma(\theta_0, S) = S$ for every $S \subseteq V$. Now, the idea above can be stated formally as the following lemma.

Lemma 3.2.1. *For every $\theta_i \in \{\theta_0, \ldots, \theta_{l-1}\}$, $|\Gamma(\theta_i, OPT)| \geq \theta_{i+1} - 1$.*

Proof. For a $\theta_i \in \{\theta_0, \ldots, \theta_{l-1}\}$, assume that v is the first vertex (not in OPT) getting activated with threshold more than θ_i and assume it activates at time t. Let OPT_{t-1} be the set of vertices activated till time $t - 1$. By definition, $OPT_{t-1} \subseteq \Gamma(\theta_i, OPT)$. Also, vertex v gets activated at time t, therefore there are at least $\theta_{i+1} - 1$ active vertices at time $t - 1$ so $|\Gamma(\theta_i, OPT)| \geq |OPT_{t-1}| \geq \theta_{i+1} - 1$. □

The following corollary follows directly from the previous lemma which gives the desired lower bound on the size of the optimal solution.

Corollary 3.2.2. *Let S be the minimum set satisfying*

$$|\Gamma(\theta_i, S)| \geq \theta_{i+1} - 1 \text{ for all } \theta_i \in \{\theta_0, \ldots, \theta_{l-1}\}, \tag{MT}$$

then, $|OPT| \geq |S|$.

Our first algorithm also uses an idea similar the one used for proving the lower bound. Suppose that we have found the optimal set S satisfying (MT). We show if we add some additional vertices to S so that the induced subgraph of vertices in S form a connected subgraph, then we have a set of initial vertices activating all vertices. First, we need to show that finding a set satisfying constraints of (MT) is feasible in polynomial time. Let us call the problem of finding the minimum set satisfying the (MT) constraints the *minimization problem* (MT). A submodular set is defined as follows.

We use the *Submodular Set Covering* problem to solve (MT) problem.

Definition 3.2.3. *Given a set U and a set function $f : 2^U \to \mathbb{R}$, f is submodular, if for every subsets $A \subseteq B \subseteq U$ and every element $e \in U$, we have $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$.*

In other words, a set function is submodular if it has non-increasing marginal growth. In Submodular Set Covering (SSC) problem, we are asked to find a minimum cardinality set S with $f(S) = f(U)$. The best achievable polynomial time approximation algorithm for SSC problem is $O(\log(\max_{e \in U} f(\{e\}))$ and the famous greedy algorithm for set covering problem has been proved to provide this approximation factor by Wolsey in [94]. We use Wolsey's result to show that there is an $O(\log n)$-approximation algorithm for our (MT) problem.

Theorem 3.2.4. *There is an $O(\log n)$-approximation algorithm for solving (MT).*

Proof. For each $0 \leq i < l$, define function $f_i : 2^V \to \mathbb{R}$ as

$$f_i(D) = \min\{\theta_i - 1, |\Gamma(\theta_{i+1}, D)|\}$$

for every $D \subseteq V$. Also define the function $f : 2^V \to \mathbb{R}$ as

$$f(D) = \sum_{i=0}^{l-1} f_i(D).$$

It is easy to see that f_i is a submodular function, so f is also a submodular function. Therefore, we can solve this instance of SSC with universe set V with the greedy algorithm to obtain a $O(\log n)$-approximate solution D^* using Wolsey's algorithm ([94]) such that $f(D^*) = f(V)$, since $\log(f(V)) \leq \log(n^2) = O(\log n)$. Observe that $f(D^*) = \sum_{i=0}^{l-1} \min\{\theta_{i+1} - 1, |\Gamma(\theta_i, V)|\} = \sum_{i=0}^{l-1}(\theta_i - 1)$, since $\Gamma(\theta_i, V) = V$ for all $0 \leq i < l$. Therefore, $|\Gamma(\theta_i, D)| \geq \theta_{i+1} - 1$ for all $0 \leq i < l$ and D^* is a feasible solution for (MT). □

Using the previous theorem now we can prove there is an $O(r.\log n)$-approximation algorithm for technology diffusion problem. Algorithm 4 shows the pseudo code.

Algorithm 4 *SelectSeedSet*1$(G, \{\theta_1, \ldots, \theta_l\})$

$S_0 \leftarrow$ an $O(\log n)$-approximate solution for (MT) problem for the instance $(G, \{\theta_1, \ldots, \theta_l\})$
$\mathcal{A}_0 \leftarrow S_0$
while $G[\mathcal{A}_0]$ is not connected **do**
  find a shortest path P connecting at least two connected components of $G[\mathcal{A}_0]$
  $\mathcal{A}_0 \leftarrow \mathcal{A}_0 \cup V(P)$.
end while
return $\mathcal{A}_0$

Theorem 3.2.5. *Algorithm 4 is an $O(r.\log n)$-approximation algorithm for Technology Diffusion problem, where r is the diameter of the graph.*

Proof. First note that S_0 is a $O(\log n)$ approximate solution for (MT) and by corollary 3.2.2, the size of minimum solution to (MT) is a lower bound for $|OPT|$, therefore, $S_0 \leq O(\log n)|OPT|$. In each iteration of the **while** loop, the path P has at most r. Also, after adding the vertices of the shortest path, the number of connected components of $G[\mathcal{A}_0]$ deacreases at least by one, so at most $|S_0|$ paths will be added to $\mathcal{A}_0$, hence $|\mathcal{A}_0| \leq r|S_0| \leq O(r.\log n)|OPT|$.

By induction on i, we prove starting from seed set $\mathcal{A}_0$, all vertices in $\Gamma(\theta_i, \mathcal{A}_0)$ will be activated for all $1 \leq i \leq l$. By definition of (MT) problem, $\Gamma(\theta_0, \mathcal{A}_0) = |\mathcal{A}_0| \geq \theta_1 - 1$, and $G[\mathcal{A}_0]$ is connected, so every node adjacent to $\mathcal{A}_0$ with threshold θ_1 gets activated, then the nodes adjacent to these newly activated nodes with threshold θ_1 and finally, every node in $\Gamma(\theta_1, \mathcal{A}_0)$ gets activated.

Now suppose every node in $\Gamma(\theta_i, \mathcal{A}_0)$ is activated for an $i \geq 1$. Since inequality (MT) is satisfied, $\Gamma(\theta_i, \mathcal{A}_0) \geq \theta_{i+1} - 1$. Similar to the case for $i = 1$, this means every node in

$\Gamma(\theta_{i+1}, \mathcal{A}_0)$ will be activated. At the end, all nodes in $\Gamma(\theta_l, \mathcal{A}_0) = V$ get activated, hence $\mathcal{A}_0$ returned by algorithm 4 is a feasible solution for the technology diffusion problem.

□

3.2.4 $O(l.\lg(n))$-Approximation Algorithm

As we will see in next section, the technology diffusion problem with only two threshold values is equivalent to the Quota Steiner Tree problem discussed in previous chapter, this intrigues us to look for more sophisticated algorithms to solve the problem. Specifically, it is unlikely to be able to eliminate the factor r from the approximation factor of the algorithm in previous section. In his section we give an $O(l.\log n)$-approximation algorithm for the problem using the Quota Steiner Tree algorithm we gave in section 3.1.

First we use a result from Goldberg and Liu [48] which states we can narrow down our search space to seed sets inducing a *connected activation sequence*.

Definition 3.2.6. *Consider a seed set $\mathcal{A}_0$. We say $\mathcal{A}_0$ induces a connected activation sequence if there is a permutation of vertices $v_1, \ldots, v_n$ such that for all $1 \leq t \leq n$, (i) vertices $v_1, \ldots, v_t$ induce a connected subgraph, and (ii) if $\theta(v_t) > t$, then $v_t \in \mathcal{A}_0$.*

Similarly, we can define a seed set inducing a connected activation sequence as follows. Consider an anchor vertex s (the vertex v_1 in definition above) and a seed set $\mathcal{A}_0$. A *connected activation* process starting at $\mathcal{A}_0$ is a process in which each vertex v not in $\mathcal{A}_0$ gets activated at time t, if v is adjacent to the connected component of the subgraph induced by active vertices $\mathcal{A}_t$ containing s and this component has at least $\theta(v) - 1$ vertices. So, in a connected activation, we only consider the number of vertices in a connected component of active vertices containing a specific vertex s to activate other vertices.

The following result from [48] by Goldberg and Liu is both interesting and insightful for understanding the structure of seed sets inducing connected activation sequences and help us achieve the $O(l.\log n)$-approximation algorithm, therefore, we briefly summarize the proof here.

Lemma 3.2.7. *The size of the optimal seed set inducing a connected activation sequence is at most twice the size of the optimal seed set.*

Proof. Given an optimal activation sequence $\mathcal{A}_0$, we construct an activation sequence $\mathcal{A}'_0$ with size no more than $2|\mathcal{A}_0|$, inducing a connected activation sequence. First let $\mathcal{A}'_0 = \mathcal{A}_0$ and consider a diffusion process starting from $\mathcal{A}_0$. Consider the vertices in the reverse

order they activate, say $v_n, \ldots, v_1$, in which v_n being the last vertex getting activated and all vertices in $\mathcal{A}_0$ appearing in $v_1, \ldots, v_{|\mathcal{A}_0|}$.

For a vertex v_t, suppose $S_1, \ldots, S_k$ are the set of vertices of connected components adjacent to v_t in the graph induced by active vertices in $\{v_1, \ldots, v_{t-1}\}$. If $|k > 1$, then add v_t to $\mathcal{A}'_0$. We call such vertex v_t a *connector* vertex. Now we claim that $\mathcal{A}'_0$ induces a connected activation sequence. We can prove this by induction on number of vertices of the graph G. If G has size 1, then clearly $\mathcal{A}'_0 = \mathcal{A}_0$. Assume that the statement is true for every graph with size less than n and let G be a graph of size n.

Consider the order $v_1, \ldots, v_n$ that vertices activate in the diffusion process started from $\mathcal{A}_0$ and let v_t be the last connector vertex v_t not in $\mathcal{A}_0$ and getting activated. Let $S_1, \ldots, S_k$ be the set of vertices of connected components adjacent to v_t in the $G[\{v_1, \ldots, v_{t-1}\}]$. Also, without loss of generality, assume that S_1 has the largest cardinality among the sets $\{S_1, \ldots, S_k\}$. Note that the connector vertices in each set S_i is defined independent of all vertices in $V \setminus S_i$ for all $1 \leq i \leq k$. Therefore by induction hypothesis, the vertices in $\mathcal{A}'_0 \cap S_1$ induce a connected activation sequence in S_1.

Since S_1 has the largest cardinality, $v \in \mathcal{A}'_0$ and each of $S_2, \ldots, S_k$ induce a connected subgraph and every vertex in $S_i \setminus \mathcal{A}'_0$ also gets activated by $S_1 \cup \{v\}$, inducing a connected activation sequence in $\mathcal{S} = \{v\} \cup \left(\cup_{i=1}^{k} S_1 \right)$ with an anchor vertex in S_1. By assumption there is no connector vertex in $V \setminus \mathcal{S}$ in activation sequence induced by $\mathcal{A}_0$, therefore after activation of vertices in $\mathcal{S}$, all activated vertices form a connected component containing vertices of $\mathcal{S}$, therefore $\mathcal{A}'_0$ induces a activation sequence activating $\mathcal{S}$ and all other vertices afterwards.

It only remains to prove that $|\mathcal{A}'_0| \leq 2|\mathcal{A}_0|$. There are at most $|\mathcal{A}_0|$ connected components of active vertices in $\mathcal{A}_0$ and in the diffusion process starting by $\mathcal{A}_0$, each connector connects at least two components, therefore the total number of connector vertices is at most $|\mathcal{A}_0|$ and we have added at most $|\mathcal{A}_0|$ vertices to $|\mathcal{A}'_0|$. □

Our second algorithm finds a seed set inducing a connected activation sequence with an anchor vertex s. The optimal solution then can be find by iterating over all vertices of the graph as the anchor vertex. The key idea is to relate the connected activation sequence of a seed set inducing a connected activation sequence anchored at s for each $1 \leq i \leq l$ to a sub tree of the graph rooted at s, containing $\theta_i - 1$ vertices. Consider a seed set $\mathcal{A}_0$ inducing a connected activation sequence anchored at s and the first vertex v with threshold θ_i or more getting activated. At the time t that v gets activated, there is connected component S of active nodes containing s with at least $\theta_i - 1$ vertices. In this component, every node with threshold θ_i or more is included in $\mathcal{A}_0$. Now assume that the cost of each vertex with

threshold θ_i or more is one and cost of all other vertices are zero. Then, S contains a subtree of $\theta_i - 1$ vertices and the cost of the tree is the number of vertices in $\mathcal{A}_0$.

The algorithm finds l minimum cost subtrees rooted at s, each for a threshold θ_i satisfying that ith subtree has at least $\theta_i - 1$ vertices. Each subtree can be found by our Node-Weighted Quota Steiner Tree algorithm from section 3.1. Also we prove the union of the cost 1 vertices of all trees form a feasible solution for technology diffusion problem. Algorithm 5 shows the pseudo code of our $O(l.\log n)$-approximation algorithm.

Algorithm 5 *SelectSeedSet2*$(G, \{\theta_1, \ldots, \theta_l\})$

$\mathcal{A}_0 \leftarrow \emptyset$
for $i = 1 \ldots l$ **do**
For each $v \in V$, define $w_i(v) = 0$ if $\theta(v) < \theta_i$ and $w_i(v) = 1$, otherwise
$T_i \leftarrow$ an $O(\log n)$-approximate tree for Steiner Quota Problem on G with costs w_i rooted at s and containing at least $\theta_i - 1$ vertices
Add every vertex v in T_i with $w_i(v) = 1$ to $\mathcal{A}_0$
end for
return $\mathcal{A}_0$

Theorem 3.2.8. *Algorithm 5 finds an $O(l \log n)$-approximate solution containing a vertex s for the technology diffusion problem.*

Proof. Let $\mathcal{A}_0^*$ be an optimal seed set inducing a connected activation sequence. First we prove that the algorithm finds a seed set $\mathcal{A}_0$ such that $|\mathcal{A}_0| \leq O(l \log n)|\mathcal{A}_0^*|$, which together with lemma 3.2.7 guarantees the approximation factor. In order to prove this bound we prove for each $1 \leq i \leq l$, the w_i-cost of subtree T_i is not greater than $O(\log n)|\mathcal{A}_r|$ by showing that $|\mathcal{A}_r|$ contains vertices of cost 1 of a subtree rooted at s with at least $\theta_i - 1$ vertex. Note that the algorithm adds the vertices with cost 1 to the tree in each iteration, so the number of vertices added to $\mathcal{A}_0$ in iteration i is at most $w_i(T_i)$.

Consider the connected activation sequence started by $\mathcal{A}_0^*$ and for an arbitrary $1 \leq i \leq l$, assume that v is the first vertex not in $\mathcal{A}_0^*$ with $\theta(v) \geq \theta_i$ getting activated. Also assume that v activates at time t. Since $\mathcal{A}_0^*$ induces a connected activation sequence, at time t, v is adjacent to a connected component of active vertices containing s and at least $\theta_i - 1$ vertices. Consider the subtree T_i' of this component containing s and at least $\theta_i - 1$ vertices. Every vertex with threshold θ_i or more in T_i' must be in $\mathcal{A}_0^*$, therefore, $w_i(T_i') \leq |\mathcal{A}_0^*|$. Since T_i is an $O(\log n)$-approximate minimum w_i-cost subtree with $\theta_i - 1$ vertices, therefore $w_i(T_i) \leq O(\log n) w_i(T_i') \leq O(\log n)|\mathcal{A}_0^*|$, as desired.

It only remains to prove that $\mathcal{A}_0$ is a feasible solution. By induction on i, we prove that every vertex in T_i activates at some time for each $1 \leq i \leq l$. For $i = 1$, we have $w_1(v) = 1$ for every $v \in V$, therefore $T_1 \subseteq \mathcal{A}_0$. Assume that the assertion is true for T_{i-1} which means at some time during the activation process, there exist at least $\theta_{i-1} - 1$ active vertices forming a connected component including s. Note that T_i also includes s and induces a connected subgraph, so every vertex in T_i with threshold less than θ_i get activated. However, the rest of the vertices in T_i are already added to $\mathcal{A}_0$, so all vertices in T_i get activated at some point and the claim is proved. At the end, we have $|T_l| \geq \theta_l - 1$ active vertices that activate all vertices of graph. □

3.2.5 Complexity

In previous section we saw how our algorithm utilizes the Node-Weighted Quota Steiner Tree algorithm from section 3.1 to obtain an $O(l.\log(n))$-approximation algorithm. Here we show that any approximation algorithm for technology diffusion problem also can solve the Node-Weighted Quota Steiner Tree problem with $\{0, 1\}$ costs and unit penalties, preserving the approximation factor.

There is a simple reduction from Set Cover problem to the Node-Weighted Quota Steiner Tree problem with $\{0, 1\}$-costs and unit penalties, which shows the best approximation factor achievable for Quota Steiner Tree problem is $O(\log n)$ in polynomial time. Goldberg and Liu proved the same for the Technology Diffusion problem by giving a reduction from Set Cover problem to Technology Diffusion problem. Our reduction gives a more general hardness for Technology Diffusion problem and suggests the improvement of the algorithms to $o(\min\{r, l\}\log(n))$-approximation probably needs an algorithm with same approximation factor for Node-Weighted Quota Steiner Tree problem with l quota constraints.

Assume that we have an instance of Quota Steiner Tree problem $(G(V, E), c, \pi, Q)$, such that $\pi(v) = 1$ and $c(v) \in \{0, 1\}$ for all $v \in V$. We show that if we have an α-approximation for technology diffusion problem, we can use this algorithm to give an $O(\alpha)$-approximate solution for this instance of Quota Steiner Tree problem. For each $v \in V$, define $\theta(v) = Q$ if $c(v) = 1$ and $\theta(v) = 1$ otherwise. Now compute a α-approximate solution $\mathcal{A}_0$ for (G, θ) instance of technology diffusion problem. Let v be a vertex of cost Q which activates before any other vertex of cost Q starting with seed set $\mathcal{A}_0$ and assume that v activates at time t. Let $\mathcal{A}_{t-1}$ be the set of active vertices at time $t-1$ and let T be the set of vertices of the connected component containing v in $G[\mathcal{A}_{t-1} \cup \{v\}]$.

Every vertex with threshold Q in T (except v) is in $\mathcal{A}_0$, therefore $c(T) \leq |\mathcal{A}_0| + 1$. Also,

T contains at least Q vertices since v is activated by vertices of $T - v$, so any spanning tree of $G[T]$ is a feasible solution for the given Quota Steiner Tree problem. This means every solution $\mathcal{A}_0$ for technology diffusion problem gives us a feasible solution of the Node Weighted Quota Steiner Tree problem with cost at most one unit more. Also in previous section we proved that we can use a solution for Quota Steiner Tree problem to obtain a solution with same cost for technology diffusion problem (with two thresholds), therefore these two problems are equivalent.

Chapter 4

Future Work

We discussed the Node-Weighted Steiner tree problem, specifically its Prize Collecting and Quota restricted generalizations and gave asymptotically tight solutions for the problems. However there are still various open versions of the problem. Also, we improved the known approximation factor for technology diffusion problem, and showed its close relation to Steiner Network Problem. Here we list some of the possible future research directions in these two areas.

Steiner Network Problems.

- We solved the Node-Weighted Quota Steiner Tree problem with asymptotically best possible approximation factor, i.e. $O(\log n)$, however for its generalization, the k-Steiner Forest problem, the best given approximation factor is $\sqrt{\min\{k, n\}}$ [56] while there is only a reduction from k-densest subgraph to the problem which we know is APX-hard and the best known approximation algorithm is a $O(n^{1/4+\epsilon})$-approximation for any constant ϵ [17]. Is it possible to improve this factor for k-Steiner forest to the same one for densest k-subgraph? Or further, is it possible to give sub polynomial approximation factor for both problems?

- For the edge weighted version of classic Steiner Tree problem the best known approximation algorithm is a 1.39 algorithm [22] while the largest inapproximability result is 96/95. Filling this gap is probably the most important problem in the field of network design problems. The gap is even larger between approximability of Steiner Forest problem (currently 2) and the known inapproximability (96/95 from inapproximability of Steiner Tree problem).

- Improving the 2.4 approximation algorithm of Berman and Yaroslavtsev for NW-ST in planar graphs [15] is an open problem, particularly we do not know the problem admits a PTAS in planar graphs or not.

- For the edge weighted PCST, Archer et al. have recently found a $(2-\epsilon)$-approximation algorithm, improving the approximation factor or inapproximability for the problem is a big open question.

Technology Diffusion Problem

- We showed that the best achievable approximation factor for Technology Diffusion problem is $O(\log n)$ and there are approximation algorithms with $O(r \log n)$ and $O(l \log n)$ factors for the problem, where r and l are the maximum distance in the graph and the number of distinct thresholds, respectively. Is it possible to show a stronger inapproximability or design an $O(\log n)$-approximation for the problem?

- We studied the *complete propagation* problem in our model of Technology Diffusion, however, there are other standard problems studied in the field of influence propagation such as

  - ◇ **MAX-INF**: given a number k, find a set of k initially activated vertices that starting with them activate maximum number of vertices. On the other side we may also have the problem of finding the minimum seed set activating at least k vertices, however this problem can be easily reduced to the complete propagation version by setting all threshold values larger than $k+1$ equal to $k+1$.

  - ◇ **Weighted Variants:** we considered the problem where each node has unit cost. An interesting variation of the problem is to consider the case where each node has a cost and we want to minimize the total costs of the selected nodes in the seed set. Also one can consider influence weights for individuals, where each node is activated if the total weight of activated nodes in its community exceeds its threshold.

  - ◇ **Other Threshold Models:** We considered deterministic arbitrary threshold values, however in may cases the random threshold values are well justified and shown to help finding better approximation factors, specifically the *Linear Threshold* model of Kempe Kleinberg and Tardos (see [73]). One interesting problem is to consider linear thresholds for technology diffusion problem.

- ◇ **Special Graphs:** There are well known properties of social network graphs such as Power-Law degree distribution. Utilizing these properties may help to find better approximation algorithms for the problem for real world graphs (see [36] for example).

References

[1] E. Ackerman, O. Ben-Zwi, and G. Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44):4017–4022, 2010.

[2] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[3] Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for prize-collecting steiner tree and tsp. *SIAM Journal on Computing*, 40(2):309–332, 2011.

[4] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[5] Sanjeev Arora and George Karakostas. A 2+ ε approximation algorithm for the k-mst problem. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 754–759. Society for Industrial and Applied Mathematics, 2000.

[6] Sunil Arya and H. Ramesh. A 2.5-factor approximation algorithm for the k-mst problem. *Information Processing Letters*, 65(3):117 – 118, 1998.

[7] I. Avramopoulos, M. Suchara, and J. Rexford. How small groups can secure interdomain routing. *Princeton University Computer Science Department, Tech. Rep. TR-808-07*, 2007.

[8] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 65–74, New York, NY, USA, 2011. ACM.

[9] Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[10] M Bateni, Chandra Chekuri, Alina Ene, MohammadTaghi Hajiaghayi, Nitish Korula, and Dániel Marx. Prize-collecting steiner problems on planar graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1028–1049. SIAM, 2011.

[11] MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Euclidean prize-collecting steiner forest. *Algorithmica*, 62(3):906–929, 2012.

[12] Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Vahid Liaghat. Improved approximation algorithms for (budgeted) node-weighted steiner problems. In *Automata, Languages, and Programming*. 2013.

[13] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Dániel Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM (JACM)*, 58(5):21, 2011.

[14] Piotr Berman and Viswanathan Ramaiyer. Improved approximations for the steiner tree problem. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 325–334. Society for Industrial and Applied Mathematics, 1992.

[15] Piotr Berman and Grigory Yaroslavtsev. Primal-dual approximation algorithms for node-weighted network design in planar graphs. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 50–60, 2012.

[16] Marshall Bern and Paul Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

[17] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an o(n¼) approximation for densest k-subgraph. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 201–210, New York, NY, USA, 2010. ACM.

[18] Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1):413–420, 1993.

[19] Avrim Blum, R Ravi, and Santosh Vempala. A constant-factor approximation algorithm for the k mst problem. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 442–448. ACM, 1996.

[20] Glencora Borradaile, Philip Klein, and Claire Mathieu. An o (n log n) approximation scheme for steiner tree in planar graphs. *ACM Transactions on Algorithms (TALG)*, 5(3):31, 2009.

[21] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for euclidean steiner forest. In *FOCS*, pages 115–124, 2008.

[22] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. An improved lp-based approximation for steiner tree. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 583–592. ACM, 2010.

[23] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 721–730, New York, NY, USA, 2009. ACM.

[24] Haowen Chan, Debabrata Dash, Adrian Perrig, and Hui Zhang. Modeling adoptability of secure bgp protocol. *SIGCOMM Comput. Commun. Rev.*, 36(4):279–290, August 2006.

[25] C.L. Chang and Y.D. Lyuu. Spreading messages. *Theoretical Computer Science*, 410(27):2714–2724, 2009.

[26] C.L. Chang and Y.D. Lyuu. Bounding the number of tolerable faults in majority-based systems. *Algorithms and Complexity*, pages 109–119, 2010.

[27] Chandra Chekuri, Alina Ene, and Ali Vakilian. Node-weighted network design in planar and minor-closed families of graphs. *Automata, Languages, and Programming*, pages 206–217, 2012.

[28] Chandra Chekuri, Alina Ene, and Ali Vakilian. Prize-collecting survivable network design in node-weighted graphs. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 98–109, 2012.

[29] N. Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[30] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1029–1038, New York, NY, USA, 2010. ACM.

[31] Miroslav Chlebík and Janka Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, 2008.

[32] Fabián A Chudak, Tim Roughgarden, and David P Williamson. Approximate k-msts and k-steiner trees via the primal-dual method and lagrangean relaxation. *Mathematical Programming*, 100(2):411–421, 2004.

[33] Erik Demaine, MohammadTaghi Hajiaghayi, and Philip Klein. Node-weighted steiner tree and group steiner tree in planar graphs. *Automata, Languages and Programming*, pages 328–340, 2009.

[34] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[35] Karoline Faust, Pierre Dupont, Jérôme Callut, and Jacques Van Helden. Pathway discovery in metabolic networks by subgraph extraction. *Bioinformatics*, 26(9):1211–1218, 2010.

[36] MohammadAmin Fazli, Mohammad Ghodsi, Jafar Habibi, Pooya Jalaly Khalilabadi, Vahab Mirrokni, and Sina Sadeghabad. On the non-progressive spread of influence through social networks. *LATIN 2012: Theoretical Informatics*, pages 315–326, 2012.

[37] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[38] Harold N Gabow, Michel X Goemans, and David P Williamson. An e cient approximation algorithm for the survivable network design problem. In *Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pages 57–74, 1993.

[39] Michael R Garey and David S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[40] Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 302–309. IEEE, 1996.

[41] Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 396–402, New York, NY, USA, 2005. ACM.

[42] Naveen Garg, Goran Konjevod, and R Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 253–259. Society for Industrial and Applied Mathematics, 1998.

[43] E. Gilbert and H. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[44] Phillipa Gill, Michael Schapira, and Sharon Goldberg. Let the market drive deployment: a strategy for transitioning to bgp security. *SIGCOMM Comput. Commun. Rev.*, 41(4):14–25, August 2011.

[45] Michel X Goemans, Andrew V Goldberg, Serge Plotkin, David B Shmoys, Eva Tardos, and David P Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 223–232. Society for Industrial and Applied Mathematics, 1994.

[46] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[47] Michel X Goemans and David P Williamson. *Approximation algorithms for NP-hard problems*, chapter The primal-dual method for approximation algorithms and its application to network design problems, pages 144–191. In [64], 1996.

[48] S. Goldberg and Z. Liu. Technology diffusion in communication networks. *arXiv preprint arXiv:1202.2928*, 2012.

[49] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001.

[50] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *ACM Trans. Knowl. Discov. Data*, 5(4):21:1–21:37, February 2012.

[51] A. Goyal, F. Bonchi, L.V.S. Lakshmanan, and S. Venkatasubramanian. Approximation analysis of influence spread in social networks. *arXiv preprint arXiv:1008.2005*, 2010.

[52] Amit Goyal, Francesco Bonchi, and Laks V.S. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 241–250, New York, NY, USA, 2010. ACM.

[53] M. Granovetter. Threshold models of collective behavior. *American journal of sociology*, pages 1420–1443, 1978.

[54] R. Guérin and K. Hosanagar. Fostering ipv6 migration through network quality differentials. *ACM SIGCOMM Computer Communication Review*, 40(3):17–25, 2010.

[55] Sudipto Guha, Anna Moss, Joseph Seffi Naor, and Baruch Schieber. Efficient recovery from power outage. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 574–582. ACM, 1999.

[56] Anupam Gupta, MohammadTaghi Hajiaghayi, Viswanath Nagarajan, and R Ravi. Dial a ride from k-forest. *ACM Transactions on Algorithms (TALG)*, 6(2):41, 2010.

[57] Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.

[58] Anupam Gupta, Jochen Könemann, Stefano Leonardi, R Ravi, and Guido Schäfer. An efficient cost-sharing mechanism for the prize-collecting steiner forest problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1153–1162. Society for Industrial and Applied Mathematics, 2007.

[59] Shai Gutner. Elementary approximation algorithms for prize collecting steiner tree problems. *Information Processing Letters*, 107(1):39–44, 2008.

[60] Mohammad Taghi Hajiaghayi and Kamal Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 631–640. ACM, 2006.

[61] MohammadTaghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. Prize-collecting steiner network problems. *Integer Programming and Combinatorial Optimization*, pages 71–84, 2010.

[62] MohammadTaghi Hajiaghayi and Arefeh Nasri. Prize-collecting steiner networks via iterative rounding. *LATIN 2010: Theoretical Informatics*, pages 515–526, 2010.

[63] SL Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1(2):113–133, 1971.

[64] Dorit S Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1996.

[65] Stefan Hougardy and Hans Jürgen Prömel. A 1.598 approximation algorithm for the steiner problem in graphs. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–453. Society for Industrial and Applied Mathematics, 1999.

[66] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[67] Vojtěch Jarník and Miloš Kössler. O minimálních grafech, obsahujících n daných bodů. *Časopis pro pěstování matematiky a fysiky*, 63(8):223–235, 1934.

[68] Youngmi Jin, Soumya Sen, Roch Guérin, Kartik Hosanagar, and Zhi-Li Zhang. Dynamics of competition between incumbent and emerging network technologies. In *Proceedings of the 3rd international workshop on Economics of networked systems*, NetEcon '08, pages 49–54, New York, NY, USA, 2008. ACM.

[69] David S Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: theory and practice. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 760–769. Society for Industrial and Applied Mathematics, 2000.

[70] Dilip Joseph, Nikhil Shetty, John Chuang, and Ion Stoica. Modeling the adoption of new network architectures. In *Proceedings of the 2007 ACM CoNEXT conference*, CoNEXT '07, pages 5:1–5:12, New York, NY, USA, 2007. ACM.

[71] RM Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

[72] Marek Karpinski and Alexander Zelikovsky. New approximation algorithms for the steiner tree problems. *Journal of Combinatorial Optimization*, 1(1):47–65, 1997.

[73] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[74] Philip Klein and R Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[75] J. Könemann, S. Sadeghian, and L. Sanità. An lmp o(log n)-approximation algorithm for node weighted prize collecting steiner tree. 2012.

[76] J. Könemann, S. Sadeghian, and L. Sanità. An LMP O(log n)-Approximation Algorithm for Node Weighted Prize Collecting Steiner Tree. *ArXiv e-prints*, February 2013.

[77] L Kou, George Markowsky, and Leonard Berman. A fast algorithm for steiner trees. *Acta informatica*, 15(2):141–145, 1981.

[78] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.

[79] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[80] Carsten Moldenhauer. Primal-dual approximation algorithms for node-weighted steiner forest on planar graphs. *Automata, Languages and Programming*, pages 748–759, 2011.

[81] Anna Moss and Yuval Rabani. Approximation algorithms for constrained node weighted steiner tree problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 373–382. ACM, 2001.

[82] Chandrashekhar Nagarajan, Yogeshwer Sharma, and David Williamson. Approximation algorithms for prize-collecting network design problems with general connectivity requirements. *Approximation and Online Algorithms*, pages 174–187, 2009.

[83] Zeev Nutov. Approximating steiner networks with node-weights. *SIAM Journal on Computing*, 39(7):3001–3022, 2010.

[84] A. Ozment and S.E. Schechter. Bootstrapping the adoption of internet security protocols. In *Proc. Fifth Workshop on the Economics of Information Security*, 2006.

[85] Hans Prömel and Angelika Steger. Rnc-approximation algorithms for the steiner problem. In *STACS 97*, pages 559–570. Springer, 1997.

[86] Hans Jürgen Prömel and Angelika Steger. A new approximation algorithm for the steiner tree problem with performance ratio 5/3. *Journal of Algorithms*, 36(1):89–101, 2000.

[87] J. Rexford and J. Feigenbaum. Incrementally-deployable security for interdomain routing. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 130–134. IEEE, 2009.

[88] Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005.

[89] M.G. Rodriguez and B. Schölkopf. Influence maximization in continuous time diffusion networks. In *29th International Conference on Machine Learning (ICML)*, 2012.

[90] Yogeshwer Sharma, Chaitanya Swamy, and David P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1275–1284, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[91] Anand Srinivas, Gil Zussman, and Eytan Modiano. Mobile backbone networks–: construction and maintenance. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 166–177. ACM, 2006.

[92] Thomas W. Valente. Social network thresholds in the diffusion of innovations. *Social Networks*, 18(1):69 – 89, 1996.

[93] David P Williamson, Michel X Goemans, Milena Mihail, and Vijay V Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

[94] L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[95] Alexander Zelikovsky. Better approximation bounds for the network and euclidean steiner tree problems. *University of Virginia, Charlottesville, VA*, 1996.

[96] Alexander Z Zelikovsky. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica*, 9(5):463–470, 1993.