On The Integrality Gap of Directed Steiner Tree Problem

by

Mohammad Shadravan

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, 2014

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

In the Directed Steiner Tree problem, we are given a directed graph G = (V,E) with edge costs, a root vertex $r \in V$, and a terminal set $X \subseteq V$. The goal is to find the cheapest subset of edges that contains an r-t path for every terminal $t \in X$. The only known polylogarithmic approximations for Directed Steiner Tree run in quasi-polynomial time and the best polynomial time approximations only achieve a guarantee of $O(|X|^{\epsilon})$ for any constant $\epsilon > 0$. Furthermore, the integrality gap of a natural LP relaxation can be as bad as $\Omega(\sqrt{|X|})$.

We demonstrate that ℓ rounds of the Sherali-Adams hierarchy suffice to reduce the integrality gap of a natural LP relaxation for Directed Steiner Tree in ℓ-layered graphs from $\Omega(\sqrt{k})$ to $O(\ell \cdot \log k)$ where k is the number of terminals. This is an improvement over Rothvoss' result that 2ℓ rounds of the considerably stronger Lasserre SDP hierarchy reduce the integrality gap of a similar formulation to $O(\ell \cdot \log k)$.

We also observe that Directed Steiner Tree instances with 3 layers of edges have only an $O(\log k)$ integrality gap bound in the standard LP relaxation, complementing the fact that the gap can be as large as $\Omega(\sqrt{k})$ in graphs with 4 layers.

Finally, we consider quasi-bipartite instances of Directed Steiner Tree meaning no edge in E connects two Steiner nodes $V - (X \cup \{r\})$. By a simple reduction from Set Cover, it is still NP-hard to approximate quasi-bipartite instances within a ratio better than $O(\log |X|)$. We present a polynomial-time $O(\log |X|)$-approximation for quasi-bipartite instances of Directed Steiner Tree. Our approach also bounds the integrality gap of the natural LP relaxation by the same quantity. A novel feature of our algorithm is that it is based on the primal-dual framework, which typically does not result in good approximations for network design problems in directed graphs.

Acknowledgements

First and foremost I would like thank my advisors Professor Jochen Könemann and Professor Zachary Friggstad, their support have been invaluable during my studies at Waterloo. Also I would like to thank Professor Chaitanya Swamy and Professor Konstantinos Georgiou for spending the time to read this thesis and for their insightful comments.

To my parents.

Table of Contents

List of Figures

Chapter 1

Introduction

In *combinatorial optimization* problems, we are given a discrete structure such as a graph, and an objective function that we want to optimize with respect to certain conditions on the underlying structure. For example, finding a *shortest path* between two nodes in a graph is an example of a combinatorial optimization problem. Some of the combinatorial optimization problems can be solved using an efficient algorithm whose running time is polynomial in terms of input size. We call this class of problems polynomially solvable or "P". Nonetheless, a large class of practically relevant problems are called "NP"-hard", and do not admit efficient exact algorithms.

A *decision problem* is a problem whose output is either "Yes" or "No". Class "P" contains all decision problems that have polynomial-time algorithms, and roughly speaking, the class "NP" is the set of all decision problems such that for any "Yes" instance of the problem, there is a short, easily verifiable "proof" that the answer is "Yes". A short proof is one whose encoding is bounded by some polynomial in the size of the instance. Clearly $P \subseteq NP$. Arguably the biggest open question is that whether $P = NP$.

It has been shown that there are problems in "NP" that are representative of the entire class, in the sense that if they have polynomial-time algorithms, then P = NP, and if they do not, then P$\neq$ NP. These are the *NP-complete* problems. A problem B is NP-complete if B is in "NP", and for every problem A in "NP", there is a polynomial-time reduction from A to B. Also a problem H is *NP-hard* if and only if there is an NP-complete problem L that is polynomial time reducible to H.

Although most of the combinatorial optimization problems are NP-hard and it is unlikely to find an efficient method to obtain the optimal solution, we might expect to achieve a near optimal solution instead. The efficient algorithms, designed for NP-hard optimization problems, which obtain a solution close to the optimal solution are called *approximation algorithms*. The ratio between the value of solution found by the algorithm and the value of optimal solution is called the *approximation factor*.

1.0.1 Definition. *An α-approximation algorithm for an optimization problem is a polynomial time algorithm that for all instances of the problem produces a solution whose value is within a factor α of the value an optimal solution.*

However, some of the problems are even very hard to be approximated, for instance there is no α-approximation algorithm for them, for every constant α. On the other hand, some problems admit approximation algorithms that can compute solutions whose value is within a $(1+\epsilon)$-factor of the optimal solution OPT for any $\epsilon > 0$. These algorithms are called *polynomial time approximation schemes* (PTAS).

An important family of combinatorial optimization problems are *network design* problems. In network design problems the goal is to design a network (i.e., find a subgraph of a given graph) that satisfies certain connectivity requirements. Edges of the given graph describe the cost of the possible links the network may have, and each requirement is in the form of connecting (or, more generally, providing large connectivity between) a pair of vertices of the graph. The goal is to find the network of minimum cost (i.e., the subgraph of minimum length), where connectivity requirements can be compromised. Most of the known network design problems are NP-hard. In the following, we will explore the common network design problems that we face in this thesis, including *Undirected Steiner Tree*, *Group Steiner Tree* and *Directed Steiner Tree*.

1.1 Steiner Tree Problems

The following three problems are the main variations of the Steiner tree problem that we face with them in this thesis.

1.1.1 Definition. *(Undirected Steiner Tree Problem). We are given an undirected graph $G = (V, E, c)$ with nonnegative edge costs $c : E \to \mathbb{R}^+$, and a subset $S \subseteq V$, called terminal vertices. A minimum Steiner tree is a tree T in G spanning all vertices of S, of minimum cost, where the cost of a tree T is defined as $c(T) := \sum_{e \in T} c_e$. The non-terminal vertices $V \setminus S$ of the graph are called the Steiner vertices.*

For arbitrary weighted graphs, the best approximation ratio achievable within polynomial time was steadily improved from 2 down to 1.39 in a series of papers [22, 35, 54, 5, 44, 32, 29, 49]. On the negative side, it is known that, it is NP-hard to find solutions of cost less than 96/95 times the optimal cost [6]. Hence, the best one can hope for is an approximation algorithm with a small but constant approximation guarantee.

1.1.2 Definition. *(Group Steiner Tree Problem) In this problem, we are given an undirected graph $G = (V, E, c)$ with non-negative edge costs $c_e \geq 0, e \in E$, a root node r, and a collection of subsets $X_1, X_2, \ldots, X_k$ of nodes. The goal is to find the cheapest subset of edges F such that for every group X_i, there is a path from r to some node in X_i using only edges in F.*

It is easy to see that the Group Steiner Tree problem generalizes the Undirected Steiner Tree problem (consider each group as a single terminal). The Group Steiner Tree problem generalizes the *set cover* problem and therefore is at least as hard as this problem. In set cover problem, we are given a collection of weighted subsets of a given ground set and seek a minimum-weight sub collection whose union is the entire ground set. It is known that for the set cover problem there is no $(1-\epsilon).\log k$ approximation for every $\epsilon > 0$, where k is the number of elements in the ground set [15]. Furthermore, Halperin and Krauthgamer [26] showed that for every fixed $\epsilon > 0$, the Group Steiner Tree problem admits no efficient $\log^{2-\epsilon} k$ approximation, where k denotes the number of groups, unless NP has quasi-polynomial algorithms, i.e. NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$). Garg, Konjevood and Ravi [16] design a randomized $O(\log n.\log k)$ approximation algorithm for this problem on tree instances. They use the result of [34] on probabilistic approximation of finite metric spaces by tree metrics problem, and give a randomized $O(\log^2 n.\log k)$-approximation algorithms on general graphs. In Chapter 2, we will show that [16] implicitly shows an $O(h.\log k)$ approximation for Group Steiner Tree problem on trees with height of h.

1.1.3 Definition. *(Directed Steiner Tree Problem) In the Directed Steiner Tree (DST) problem, we are given a directed graph $G = (V, E, c)$ with edge costs $c_e \geq 0, e \in E$. Furthermore, we are given a root node $r \in V$ and a collection of terminals $X \subseteq V$ and the goal is to find the cheapest collection of edges $F \subseteq E$ such that there is a directed $r-t$ path using only edges in F for every terminal $t \in X$. Throughout, we will let $n = |V|$, $m = |E|$, and $k = |X|$.*

If $X \cup \{r\} = V$, then the problem is simply the minimum-cost arborescence problem which can be solved efficiently[21]. However, the general case is well-known to be NP-hard. In fact, the problem can be seen to generalize the Group Steiner Tree problem, and therefore cannot be approximated within an $O(\log^{2-\epsilon}(n))$ factor for any constant $\epsilon > 0$ unless NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$) [26].

1.1.1 Motivation

Undirected Steiner Tree Problem. The Steiner tree problem is one of the fundamental problems in the field of network design. This problem was first introduced by J. Steiner in the Euclidian plane for 3 points [7]. The decision version of this problem is NP-Complete even when the graph is induced by points in the plane.

The Steiner tree problem has many applications, for example in circuit layout or network design. In optimizing the area of Very Large Scale Integrated (VLSI) layouts, a given set of pins (terminal endpoints of a circuit) is to be connected using minimum total wire-length. When all wires are point-to-point, with no intermediate junctions other than points of P, the optimum solution is a minimum spanning tree (MST) over P. However, we can usually introduce intermediate junctions, called Steiner points, in connecting the points of P. Thus the problem we are seeking to solve under this assumption would be

a minimum Steiner tree problem. Steiner trees are important in global routing and wire-length estimation [9], as well as in various non-VLSI applications such as phylogenetic tree reconstruction in biology [30], network routing, and civil engineering, among many other areas [37].

Group Steiner Tree Problem.

The problem was introduced by Reich and Widmayer [48] and finds application in VLSI design. From a practical point of view the group Steiner problem models several scenarios in VLSI layout design [27]. For example, in one model in VLSI design, each terminal is a collection of ports and we seek a minimum length net containing at least one port from each terminal group. In addition, Group Steiner Tree problem models the network design problems with location-theoretic constraints studied by Marathe, Ravi and Sundaram [41]. The location-theoretic objective requires that we choose a subset S of nodes and locate services at the vertices in S, such that every node is within a bounded distance from at least one vertex in S.

Directed Steiner Tree Problem. The directed version of the Steiner tree problem is a natural generalization of the Undirected Steiner Tree problem. It has many applications in network design and network routing [11]. For example, multicasting involves the distribution of the same data from a central server to several nodes in the network and the problem is to choose a set of edges (or communication links) of minimum cost for the server to route the data. The connection between this problem and the Steiner problem is clear. However, there are many networks in practice where the communication links are asymmetric and cannot be modeled by undirected edges. The problem of multicast routing in asymmetric networks has received considerable attention [47].

A version of Directed Steiner Tree problem in the rectilinear plane has a great interest in VLSI designs. Given a set P of n points in the first quadrant of the rectilinear plane, a rectilinear Steiner arborescence tree is a directed tree rooted at the origin, consisting of all paths from the root to points in P with horizontal edges oriented in left to right direction and vertical edges oriented in bottom-up direction. Shi and Su [53] showed that computing the minimum rectilinear arborescence is NP-hard. Lu and Ruan [39] showed, by employing Arora's techniques (the PTAS for connectivity problems in planar graphs), that there is a polynomial-time approximation scheme for this special case of the problem.

Also from theoretical point of view, the importance of Directed Steiner Tree problem is that a wide variety of well-known network design problems in both directed and undirected graphs, such as (undirected) Steiner tree problem, Group Steiner Tree problem, node weighted Steiner tree problem and (non-metric, multilevel) facility location problem can be reduced to this problem in approximation preserving ways.

Zelikovsky [10] showed for any DST instance G, and integer $\ell \geq 1$, there is an ℓ-layered DST instance (defined in 2.5.1) such that $OPT_G \leq OPT_H \leq \ell \cdot k^{1/\ell} \cdot OPT_G$ and that a DST solution in H naturally corresponds to a DST solution in G with the same cost. Charikar et al. [11] exploited this fact and present an $O(\ell^2 k^{1/\ell} \log k)$-approximation with running time $\text{poly}(n, k^\ell)$ for any integer

$\ell \geq 1$. In particular, this can be used to obtain an $O(\log^3 k)$-approximation in quasi-polynomial time and for any constant $\epsilon > 0$ a polynomial-time $O(k^\epsilon)$-approximation. Finding a polynomial-time polylogarithmic approximation remains an important open problem.

1.2 LP/SDP Hierarchies

Consider an integer linear program for a combinatorial optimization problem in the form of

$$OPT = \min\{c^T x | Ax \geq b; x \in \{0,1\}^n\}, \tag{1.2.1}$$

where the system $Ax \geq b$ represents the problem structure. The constraint $x \in \{0,1\}^n$ is needed to obtain an integral solution, since in combinatorial optimization problems usually we need to choose some discrete objects and this binary decision is naturally formulated by 0-1 variables. Therefore the set $K = \{Ax \geq b; x \in \{0,1\}^n\}$ represents the set of all feasible solutions (or *integral solutions*) for the problem and the goal is to find those feasible solutions that optimize the objective function $c^T x$. Note that since c is linear, optimizing over set K is equivalent to optimizing over convex hull of all integral solutions, we denote by $P_I = conv(K)$.

As it turns out, if the combinatorial problem that we are studying is NP-hard, we can not efficiently find the optimum solution over all feasible solutions K. Thus we can instead relax the condition $x \in \{0,1\}^n$ that requires all solutions to be integral, to $x \in [0,1]^n$ which allows us to have *fractional solutions*, and obtain a linear program

$$OPT_f = \min\{c^T x | Ax \geq b; x \in [0,1]^n\}, \tag{1.2.2}$$

i.e., the polytope $P = \{Ax \geq b; x \in [0,1]^n\}$ is the relaxed polytope of P_I that we instead optimize over. From definition we can see $P_I \subseteq P$. Also we call the linear program 1.2.2 the *linear programming relaxation* of the integer program 1.2.1

Although losing the 0-1 constraints will allow us to find the *optimal fractional solution*, with value OPT_f, it is not directly a feasible solution for our problem (we ignored integrality constraints). Hence we need to obtain an integral feasible solution from the optimal fractional solution that we found. That is the concept of rounding a fractional solution to an integral solution in approximation algorithms. And the challenge is to obtain an integral feasible solution as close as possible to the integral optimal solution, with value OPT. In most of the approximation algorithms the approximation guarantee derived from comparing its value to that of the linear programming relaxation. Therefore if α is the approximation factor of an algorithm, often it is compared with OPT_f instead of OPT. Therefore we can not have $\alpha \leq OPT/OPT_f$. This ratio known as integrality gap of the integer program is defined as follows:

1.2.1 Definition. *(Integrality Gap) Given a LP-relaxation P for a minimization problem, let $OPT_f(I)$ and $OPT(I)$ denote the cost of an optimal fractional and*

optimal integral solution to instance I*, and The integrality gap of the relaxation* P *for the problem* Π *is defined as*

$$Sup_I \frac{OPT(I)}{OPT_f(I)},$$

where the supremum is over all instances of Π.

If the integrality gap is large then the optimal fractional and integral solutions may differ greatly. Hence we should in some way strengthen the relaxation P, and obtain $P' \subset P$, such a way that P' still contains P_I, i.e. we do not loose any integral solution. Such a tightening would be aimed at decreasing the integrality gap.

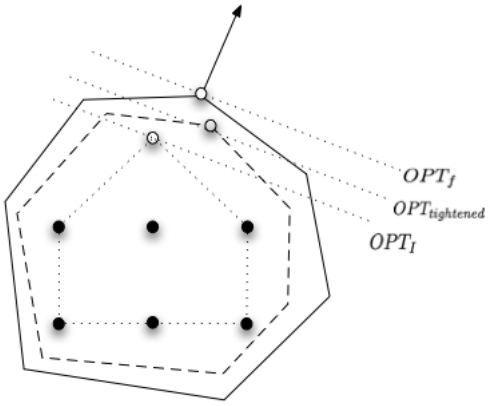

Figure 1.1: The outer polytope is the relaxation P, the yellow one is P_I the integral hull of P and the dashed polytope is the strengthened polytope.

There are several general techniques that can be used to strengthen the relaxation P. The well-known ones are the techniques producing LP/SDP hierarchies such as Lovasz Schrijver (LS) [38], Sherali-Adams (SA) [52] and Lasserre (Las) [36]. The idea behind them is the *lift and project* method which will be discussed in the next chapter. But the common fact about these hierarchies of relaxations is that they provide a sequence of convex bodies $P = P_0 \subseteq P_1 \subseteq \cdots \subseteq P_n = P_I$ (n is the number of variables in P), ranging from P to P_I. Optimizing over P_r requires $O(n^{O(r)})$ time, assuming there is a polynomial time separation oracle for P. Although optimizing over P_n is equivalent to finding the optimal integral solution, it is not efficient as optimizing ove P_n requires exponential time. In general, it is known that the Lasserre SDP hierarchy is strictly stronger than all others, i.e., the r-th level Lasserre tightening is contained in the r-th level Sherali-Adams or Lovasz-Schrijver tightening, ($Las_r(P) \subseteq SA_r(P)$).

For example a natural linear programming (LP) relaxation for Directed

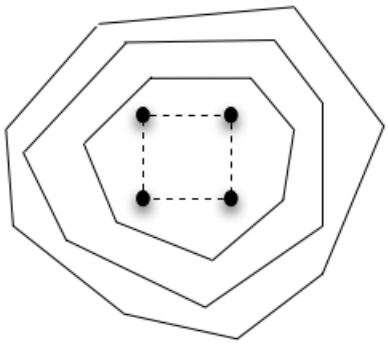

Figure 1.2: A hierarchy starting from the original polytope and ending with the integral hull.

Steiner Tree is given by LP (P0).

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & \text{(P0)} \\ \text{s.t.} \quad & x(\delta^{\text{in}}(S)) \geq 1 \qquad \forall\ S \subseteq V - r, S \cap X \neq \emptyset \\ & x_e \in [0,1] \quad \forall e \in E \end{aligned}$$

The variables x_e indicate whether we pick an edge in the solution or not. Thus the objective function of the above LP is the total cost of the edges we pick in the solution. An integral solution returned by the above LP is a feasible solution for the Directed Steiner Tree problem. Because the cut constraints $x(\delta^{in}(S)) \geq 1$, is saying that for every cut separating the root from some other terminals there should be a crossing edge. It guarantees that for every terminal vertex in $t \in X$, there exists at least a directed path from root to t (otherwise consider all vertices can reach to t as a set S and consider the cut $\delta^{in}(S)$ then $x(\delta^{in}(S)) = 0$). Although, every integral solution is not necessarily a Directed Steiner Tree (there might be many different paths to a terminal vertex), the optimal integral solution is a Directed Steiner Tree (because if a vertex has more than one incoming edge then you can keep one of them which is the last edge in a path from root to t and remove the rest and still have a feasible solution).

Zosin and Khuller [25] demonstrated that the integrality gap of the above LP-relaxation can, unfortunately, be as bad as $\Omega(\sqrt{k})$, where k is the number of terminals, even in instances where G is a 4-layered graph. We will illustrate their example in Chapter 2. Recently, Rothvoss [50] showed that 2ℓ rounds of the *Lasserre* hierarchy suffice to reduce the integrality gap of a similar LP relaxation to only $O(\ell \cdot \log k)$ in ℓ-layered graphs.

SDP Hierarchies Results. Hierarchies of convex programming relaxations, a.k.a. lift-and-project methods, have recently been used successfully in

the design of approximation algorithms. One successful application of Chlamtac [13] uses the 3rd level of the Lasserre relaxation to find $O(n^{0.2072})$-colorings for 3-colorable graphs. Though the $O(\sqrt{\log n})$ approximation of Arora, Rao and Vazirani [2] for *Sparsest Cut* does not explicitly use hierarchies, their triangle inequality is implied by $O(1)$ rounds of Lasserre. Certain hierarchies give a sequence of improvements in the integrality gap in their first O(1) levels, this has been shown for Vertex-Cover in planar graphs [40], Knapsack [31], and Maximum Matching [42]. The importance of $O(1)$ level of these hierarchies is that their running time would be polynomial time and they can be used to design a polynomial time approximation algorithms. There are results where the improved approximation is the state-of-the-art for the respective problems such results include recent work on Chromatic Number [47], Hypergraph Independent Set [14], MaxMin Allocation [4], and Sparsest-Cut [23].

On the negative side, unfortunately, starting with the work of Arora, Bollobas, Lovasz, and Tourlakis [1] on Vertex-Cover, there has been a long line of work showing that for various problems, even after a large (super-constant) number of rounds, various lift and project methods do not yield smaller integrality gaps than a basic LP/SDP relaxation. In particular, Raghavendra and Steurer [45] have recently shown that a super constant number of rounds of certain SDP hierarchies does not improve the integrality gap for any constraint satisfaction problem (MAX-CSP).

LP Hierachies Results. LP Hierarchies also have been used successfully in the tightening of integrality gap of LP-relaxations or in the design of approximation algorithms. For example Mathieu and Sinclair show that after k levels of the Sherali-Adams hierarchy, the integrality gap for the matching polytope reduces to an asymptotically tight expression $1 + 1/k$. [42] Bateni et al [4] consider the problem of MaxMin allocation of indivisible goods. They develop an n^{ϵ}-approximation for this problem that runs in $n^{O(1/\epsilon)}$ time and obtain a polylogarithmic approximation that runs in quasipolynomial time, by applying Sherali-Adams hierarchy to a natural LP relaxation of the problem. The technical core of their result for this case comes from an algorithm for an interesting new optimization problem on directed graphs, MaxMinDegree Arborescence, where the goal is to produce an arborescence of large outdegree. Recently, Gupta et al [23] gave a 2-approximation algorithm based on Sherali-Adams hierarchy for Non-Uniform Sparsest Cut that runs in time $n^{O(k)}$, where k is the treewidth of the graph. This improves on the previous 2^{2k}-approximation in time $poly(n)2^{O(k)}$ due to Chlamtac et al. [13], which was also taking advantage of the Sherali-Adams hierarchy.

LP/SDP Hierarchies in Comparison. There also several example showing the strength of SDP hierarchies compared to LP hierarchies. For instance, the LP-based hierarchies of Lovasz-Schrijver and Sherali-Adams cannot even reduce the *Max-Cut* gap below $2 - \epsilon$ after $\Omega(n)$ [51] and n^{δ} [12] many rounds, respectively. Though, a single round of the SDP based hierarchies reduces the gap to 1.13. Karlin, Mathieu and Nguyen [31] studied the integrality gap of the Knapsack linear program in the Sherali-Adams and Lasserre hierarchies. First, they show that an integrality gap of $2 - \epsilon$ persists up to a linear number of

rounds of Sherali-Adams, despite the fact that Knapsack admits a fully polynomial time approximation scheme. Second, they show that the Lasserre hierarchy closes the gap quickly. Specifically, after t rounds of Lasserre, the integrality gap decreases to $t/(t-1)$. This was the first positive result that uses more than a small number of rounds in the Lasserre hierarchy.

Unique Game Conjecture Results. In computational complexity theory, the *Unique Games Conjecture* (UGC) is a conjecture made by Khot in 2002 [33]. The conjecture states that the problem of determining the approximate value of a certain type of game, known as a unique game, is NP-hard. It has broad applications in the theory of hardness of approximation. If it is true, then for many important problems it is not only too hard to get an exact solution (as implied by the P versus NP problem), but also too hard to get a good approximation. In many cases, including all constraint satisfaction problems and various graph partitioning problems, the best algorithms are based on fairly simple SDP relaxations. The unique game conjecture (UGC) foretells that for these problems, no tighter relaxation than these simple SDPs will yield a better approximation ratio in the worst-case. Arguably, Lasserre SDPs is the strongest known method that is possible to show that the Unique Games conjecture is wrong, as even the possibility of the 4th level of Lasserre SDP relaxation improving upon the Goemans-Williamson 0.878 approximation factor for Max Cut has not been ruled out. Recently, it has also been shown that O(1) rounds of the Lasserre hierarchy are able to solve all candidate gap instances of Unique Games [3]. (On the other hand, for some of the weaker hierarchies, integrality gaps for super-constant rounds are known for various Unique-Games hard problems [34].)

1.3 Quasi-bipartite Instances

In Chapter 4, we consider the class of *quasi-bipartite* DST instances. An instance of DST is quasi-bipartite if the Steiner vertices $V \setminus X$ form an independent set (i.e., no two Steiner vertices are adjacent).

1.3.1 Definition. *(Quasi-bipartite graphs) We are given a directed graph $G = (V, E, c)$ with cost $c : E \to \mathbb{R}^+$ on edges, a vertex root $r \in V$, and a set of terminal vertices $X \subseteq V$. We call the vertices in $V \setminus X$ Steiner vertices. There is no edges $e = uv$ in E where $u, v \notin X$, i.e, there is no edge between Steiner vertices.*

Throughout assume $k = |X|$, $n = |V|$ and $m = |E|$. Such instances still capture the set cover problem, and thus do not admit an $(1-\epsilon)\log k$-approximation, for any $\epsilon > 0$ if P $\neq$ NP [15]. Quasi-bipartite instances have been studied in the context of *undirected* Steiner trees as well. The class of graphs was first introduced by Rajagopalan and Vazirani [46] who studied the integrality gap of (P0) for the *bidirected* map [46] of given Undirected Steiner Tree instances. They also provide a $(3/2+\epsilon)$ primal-dual approximation algorithm for the Undirected Steiner Tree problem on such instances, for every $\epsilon > 0$. The same concept has

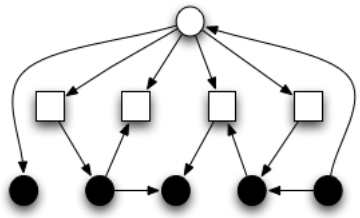

Figure 1.3: An example of a Quasi-bipartite graph. Root is the white circle, terminals are the black circles and Steiner vertices are the rectangles. There is no edge between Steiner vertices.

been used by subsequent authors on the Undirected Steiner Tree problem, e.g. Robins and Zelikovsky use loss-contracting approach [49] to solve the Steiner Tree Problem in quasi-bipartite graphs, achieving an approximation ratio of 1.28 within time $O(kn^2)$.

Note that Gröpl et al. [20] show the upper bound of 73/60 for the more restricted case of uniform quasi-bipartite graphs, where all edges incident to a non-terminal node have the same cost. Then Byrka et al [8] show $73/60 + \epsilon$ approximation for the case of quasi-bipartite instances. Currently, the best approximation for quasi-bipartite instances of Undirected Steiner Tree is $\frac{73}{60}$ by Goemans et al. [19] who also bound the integrality gap of bidirected cut relaxation [19] by the same quantity. This is a slight improvement over a prior $(\frac{73}{60} + \epsilon)$-approximation for any constant $\epsilon > 0$ by Byrka et al. [8]. The best approximation for general instances of Undirected Steiner Tree is $\ln(4) + \epsilon$ for any constant $\epsilon > 0$ [8]. However, the best integrality gap upper bound on the bidirected cut relaxation for non-quasi-bipartite instances is only 2.

These results, motivate us to consider the Directed Steiner Tree problem on this family of graphs. In Chapter 4, we present a logarithmic approximation algorithm for the Directed Steiner Tree problem on these instances. Our algorithm is based on the well-studied primal-dual method. As a result, we can also give a logarithmic bound on the integrality gap of the natural LP-relaxation for the Directed Steiner Tree problem. In contrast to undirected network design problems, primal-dual algorithms have been used very rarely in the design of approximation algorithms for directed network design problems. Our algorithm is one of the few known application of this method in directed network design problems.

1.4 Our Results

We have two main contribution in this thesis. In our first contribution, we consider applying the Sherali-Adams hierarchy to the natural LP-relaxation of the Directed Steiner Tree problem, and in the second one, we consider the

Directed Steiner Tree problem on quasi-bipartite graphs.

As we discussed earlier, Zosin and Khller [25] showed that the integrality gap for this relaxation might be as bad as $\Omega(\sqrt{n})$. However, we show that the l-rounds of Sherali-Adams hierarchy reduces the integrality gap of l-layered instances to $O(l.\log k)$. In particular, we will prove the following theorem:

1.4.1 Theorem. *If the DST instance G is an ℓ-layered graph, then the integrality gap of ℓ-th level Sherali-Adams tightening of LP (P0) is $O(\ell \cdot \log k)$.*

This is an improvement over Rothvoss's result [50] which utilizes $2l$ levels of the stronger Lasserre hieratichy. In addition, we show that for 3-layered instances the integrality gap of LP (P0) is already $O(\log k)$. Note that since solving l-level Sherali-Adams tightening of Directed Steiner Tree LP-relaxation requires $O(n^{\ell})$ time (the starting LP is polynomially solvable), our result does not imply a polylogarithmic approximation in polynomial time. But we can still get polylogarithmic approximation in quasi-polynomial time, or $O(k^{\epsilon})$ in polynomial time as Charikar et al [11] found previously. Therefore, it is still an important open question whether there is a polylogarithmic approximation algorithm in polynomial time for Directed Steiner Tree or not.

Furthermore, we consider the special case of quasi-bipartite instances of the Directed Steiner Tree problem, and affirmatively answer the above question. In particular, we design a logarithmic approximation algorithm for the Directed Steiner Tree problem on quasi-bipartite instances. Our approach is based on the primal-dual method. As a result, we prove a logarithmic integrality gap upper bound for the natural LP relaxation of the Directed Steiner Tree problem on these instances. Our main theorem is the following:

1.4.2 Theorem. *The integrality gap of LP (P0) is at most $3 \cdot H_K = O(\log k)$ in quasi-bipartite graphs with k terminals where H_k is the k'th Harmonic number. Furthermore, a Steiner tree witnessing this integrality gap bound can be constructed in polynomial time.*

This is the first logarithmic integrality gap bound for the quasi-bipartite instances of the Directed Steiner Tree problem.

1.5 Overview of the Thesis

In Chapter 2, we will present basic definitions and theorems related to Sherali-Adams hierarchy, such as *local decomposition* theorem that is at the heart of our algorithm in Chapter 3. Next, we take a fresh look at the Group Steiner Tree problem, and give an $O(l.\log k)$ approximation algorithm, for l-layered instances.

In Chapter 3, we present our first contribution, the application of the Sherali-Adams hierarchy to the natural LP-relaxation of the Directed Steiner Tree problem. In Chapter 4, we present our second contribution, in which we design an LP-based logarithmic approximation algorithm for quasi-bipartite instances

of the Directed Steiner Tree problem. Finally, in Chapter 5, we state some open problems.

Chapter 2

Background

2.1 Lift and Project

Suppose we have a combinatorial optimization problem, and we can formulate it as an integer program (IP), say $\min\{c^T x | Ax \geq 0\, x \in \{0,1\}^n\}$. The objective function in the problem is $c^T x$, which is linear. Also, the set $P_I = \{Ax \geq 0, x \in \{0,1\}^n\}$ describes the rest of feasible integral solutions for the problem. Thus we are seeking to optimize over P_I.

If the problem is NP-hard, then we can not solve the integer program efficiently, unless $P = NP$. Therefore, we usually relax the integer program to a linear program (LP) and find the optimal solution of the LP. For example we relax the variables $x_i \in \{0,1\}$ for every i in the IP to a variable in $[0,1]$.

In fact, we can efficiently solve a linear program in polynomial time in the number of variables and constraints. Also by using ellipsoid method[20] , we can solve linear programs with superpolynomially many constraints in polynomial time given that we have a polynomial time separation oracle. The separation oracle decides whether a given point is feasible or not, and if it is not feasible, it separates the infeasible point from the feasible region by a separating plane.

Hence, we can efficiently optimize over $P = \{Ax \geq 0, x \in [0,1]^n\}$, the relaxed polytope, instead of P_I. However, P is a relaxation of P_I, and there might be a huge difference between their optimal solutions. Thus, the next step is to somehow convert the fractional optimal solution in P to an integral solution in P_I, by rounding a fractional solution into an integral solution. One of the common methods of rounding a fractional solution to an integral solution is the *randomized rounding* method, in which for each variable x_i, we decide whether it should be 0 or 1, with some probability that we choose based on the fractional optimal solution that we found.

There is no efficient way to find the integral hull of integral solutions P_I. But what we can obtain by solving LP is fractional solutions. A great number of known approximation algorithms are based on the idea that cleverly convert those fractional solutions into integral ones. We call this process rounding. Lets

denote by *Round* the value of integral solution we find from rounding. In order to show that approximation factor, the ratio $\frac{Round}{OPT_I}$, is upper bounded, instead we show $\frac{Round}{OPT_f}$ is upper bounded; because we compare the rounded solution with the optimal fractional solution. But this could automatically show that $\frac{OPT_I}{OPT_f}$ is upper bounded and this is the notion of integrality gap.

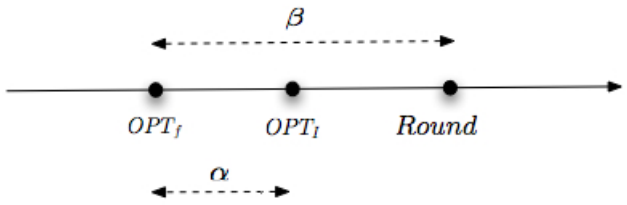

Figure 2.1: α is the integrality gap and β is the approximation factor. OPT and OPT_f are optimal integral and fractional solutions respectively. *Round* is the solution returned by the algorithm.

For many optimization problems the natural LP-relaxation has a big integrality gap which prohibits us from achieving a good approximation factor via that LP. One idea is to add some constraints to the LP and thereby reduce the integrality gap. Finding strong valid inequalities (that preserve every feasible integral solution) to tighten a given relaxation is a non-trivial task, and sometimes lead to produce large LPs (e.g. when the problem is NP-hard).

Another more systematic way to approach the integral hull started by the work of Gomory-Chvátal [15] in which after a linear number of applying a procedure to the LP we reach the integral hull. However, each of these steps may be computationally difficult. For example optimizing over first level of Gomory-Chvátal is NP-hard. The developed procedures based on lift and project methods are providing operators (converting a convex body to another one) that gradually tighten the LP-relaxation and yield the integral hull in finitely many steps. So that the efficiency of applying each step does not cost very much in terms of size of that level (can be done in polynomial time). However, the total time in all these iterations to obtain the integral hull can of course be exponential assuming $P \neq NP$. What these approaches give us is a hierarchy of convex bodies The first one is the polytope from our LP and the last one is the integral hull. In addition, each convex body is contained in the previous one.

In the *lift and project* procedures we lift the solution space to a higher dimensional space (by adding some auxiliary variables) and by adding some constraints we refine that region and again project it to the original space. The idea is that we simulate nonlinear programming by linear programming. For example consider the constraint $x(1-x) = 0$. This nonlinear constraint guarantees that $x \in \{0, 1\}$, so it is powerful enough to simulate any binary integer programming. However, solving binary IPs is NP-hard and we cannot hope to solve this problem in polynomial time. The idea behind Lift and Project

methods is that we replace each nonlinear term with a new auxiliary variable. For instance, we replace $x_i.x_j$ with a new variable x_{ij}.

Consider the *vertex cover* problem defined as follows. We are given an undirected graph $G = (V, E)$, our goal is to select a subset of vertices S with minimum size such that for every edges $e = uv \in E$, either $u \in S$ or $v \in S$. We can write a quadratic program for the above problem:

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v & (2.1.1) \\ \text{s.t.} \quad & (1 - x_u)(1 - x_v) = 0 \qquad \forall \{u, v\} \in E \\ & x_v \in [0, 1] \quad \forall v \in V \end{aligned}$$

The constraint $(1-x_u).(1-x_v) = 0$ implies that, $\forall\{u, v\} \in E$, either $x_u = 1$ or $x_v = 1$ or both (i.e., for each edge at least one of the endpoints will be picked). In the lift and project methods we simulate nonlinear programs by linear programs, in the above quadratic program we replace nonlinear terms with a new auxiliary variable, so we replace $x_u x_v$, $\forall uv \in E$ with a new variable x_{uv}. Thus, we write the constraint in the above quadratic program as $1 - x_u - x_v + x_{uv} = 0$. Therefore we obtain the following linear program:

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v & (2.1.2) \\ \text{s.t.} \quad & 1 - x_u - x_v + x_{uv} = 0 \qquad \forall \{u, v\} \in E \\ & x_v \in [0, 1] \quad \forall v \in V \end{aligned}$$

It is obvious that every integral solution x for the above nonlinear program yields a valid solution for the linear program in the lifted space. Because you can simply set $x_{uv} = x_u x_v$. But not every solution to the lifted problem is integral since there is no way to force $x_{uv} = x_u x_v$.

2.2 Sherali-Adams Hierarchy

The Sherali-Adams hierarchy was first introduced in [23]. It has both SDP-based and LP-based definition. In this thesis, we will focus on the LP-based definition. In the following, we provide a recursive definition of this hierarchy. The hierarchy consists of several levels, starting from the original polytope $\mathcal{P}$, in level $r = 0$, and ending in $\mathcal{P}_I$, the integral hull of $\mathcal{P}$. Each intermediate level of hierarchy gives us a convex body defined with a set of linear constraints, therefore each level of the Sherali-Adams hierarchy is a polytope. The constraints in each level of the hierarchy are defined based on the constraints in the previous level. Also we present a non-recursive definition and prove their equivalence. Next, we show some properties of the hierarchy, in particular, the *local decomposition* theorem, and present the notion of *conditioning* a solution on a subset of variables.

Let $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a polytope with m linear constraints $\sum_{i=1}^{n} A_{j,i}.x_i \geq b_i$, $1 \leq j \leq m$. Suppose the "box constraints" $0 \leq x_i$ and $x_i \leq 1$ (equivalently $-1 \leq x_i$) appear among these constraints for each $1 \leq i \leq n$. For $r \geq 0$, let $\mathscr{P}_r(n) = \{S \subseteq \{0, 1, \cdots, n\} : |S| \leq r\}$ denote the collection of subsets of $\{1, \cdots, n\}$ of size at most r. We also let $\mathbb{R}^{\mathscr{P}_r(n)}$ denote $\mathbb{R}^\alpha$, where $\alpha = |\mathscr{P}_r(n)| = n^{O(r)}$. We index a vector in $\mathbb{R}^{\mathscr{P}_r(n)}$ by sets in $\mathscr{P}_r(n)$. Let $y^{(r)}$ denote a vector with one coordinate for each element in $\mathscr{P}_{r+1}(n)$. Also we denote by $y|_{\mathscr{P}_{r'}(n)}$, the restriction of the vector y on indexes in $\mathscr{P}_{r'}(n)$, for $r' < r$. Thus $y|_{\mathscr{P}_{r'}(n)} \in \mathbb{R}^{\mathscr{P}_{r'}(n)}$. Before we give a definition of Sherali-Adams hierarchy, we define the notion of *cone*:

2.2.1 Definition. *A cone $K \subseteq \mathbb{R}^{n+1}$ is a set of points that is closed under scaling: if $x \in K$ then $c.x \in K$ for all $c \geq 0$.*

2.2.2 Definition. $\mathrm{SA}_r(\mathcal{P})$*, the r-th level of Sherali-Adams hierarchy is a cone in $\mathbb{R}^{\mathscr{P}_{r+1}(n)}$, defined as follows:*

- $\mathrm{SA}_0(\mathcal{P}) = \{t(1, x_1, \cdots, x_n) \in \mathbb{R}^{n+1} | x \in \mathcal{P}, t \geq 0\}$*, is the cone constructed from $\mathcal{P}$: for each constraint $a^T x - b \geq 0$ in $\mathcal{P}$, we add constraint $a'^T \cdot y^{(0)} \geq 0$, to the $\mathrm{SA}_0(\mathcal{P})$, where $a' \in \mathbb{R}^{\mathscr{P}_1(n)}$, $a'_{\{i\}} = a_i$ and $a'_\emptyset = -b$. i.e., $\sum_{i=1}^{n} a_i x_i \geq b$, a constraint of $\mathcal{P}$, is replaced by $\sum_{i=1}^{n} a_i x_i \geq b x_0$ as a constraint for the cone $\mathrm{SA}_0(\mathcal{P})$ (this is called homogenization $\mathcal{P}$).*

- *For every constraint $a^T y^{(r-1)} \geq 0$ of $\mathrm{SA}_{r-1}(\mathcal{P})$, and for every $i \in [n]$ we have two constraints in $\mathrm{SA}_r(\mathcal{P})$:*

  1. $x_i * a^T y^{(r-1)} \geq 0$
  2. $(1 - x_i) * a^T y^{(r-1)} \geq 0$

  Where $*$ distributes over all terms of $a^T y^{(r-1)} = \sum_{S \in \mathscr{P}_r(n)} a_S y_S$ and $x_i * y_S = y_{S \cup \{i\}}$.

For example, suppose $a^T y^{(r-1)} = a_\emptyset y_\emptyset + a_{\{1\}} y_{\{1\}} + a_{\{2\}} y_{\{2\}} + a_{\{1,2\}} y_{\{1,2\}}$, then $x_1 * a^T y^{(r-1)} = x_1 * a_\emptyset y_\emptyset + x_1 * a_{\{1\}} y_{\{1\}} + x_1 * a_{\{2\}} y_{\{2\}} + x_1 * a_{\{1,2\}} y_{\{1,2\}} = a_\emptyset y_{\{1\}} + a_{\{1\}} y_{\{1\}} + a_{\{2\}} y_{\{1,2\}} + a_{\{1,2\}} y_{\{1,2\}}$.

2.2.1 Remark. Note that if we restrict $\mathrm{SA}_0(\mathcal{P})$ to points y with $y_\emptyset = 1$, and project it to $\mathbb{R}^n$, we obtain $\mathcal{P}$, i.e. $\mathrm{SA}_0(\mathcal{P})|_{y_\emptyset = 1} = \{(1, x_1, \cdots, x_n) | x \in \mathcal{P}\}$. Since $a'^T y \geq 0$ implies $\sum_{i \in [n]} a'_{\{i\}} y_{\{i\}} + a'_\emptyset y_\emptyset = \sum_{i \in [n]} a'_{\{i\}} y_{\{i\}} - b \cdot 1 \geq 0$, so $a^T y|_{[n]} \geq b$, where $y|_{[n]}$ is the restriction of vector y to $\mathbb{R}^n$, i.e., if $y = (x_0, x_1, \cdots, x_n)$, then $y|_{[n]} = (x_1, x_2, \cdots, x_n)$.

It can be shown that the following non-recursive definition is equivalent to the above definition:

2.2.3 Definition. *(non-recursive definition of Sherali-Adams hierarchy) For every constraint $a^T x \geq b$ of $\mathcal{P}$, and for every $U \in \mathscr{P}_r(n)$, and $W \subseteq U$, consider the constraint*

$$C = (a^T x - b) \prod_{s \in W} x_s \prod_{s \in U \setminus W} (1 - x_s) \geq 0 \tag{2.2.1}$$

and replace every term $\prod_{s \in I} x_s$, where $I \in \mathscr{P}_{r+1}(n)$, with variable y_I (replace constants c with $cy_\emptyset$), we call this new expression the linearization of C and denote it by $\Phi(C)$. Add this constraint to the set of constraints of $\mathrm{SA}_r(\mathcal{P})$.

2.2.2 Theorem. *The recursive definition of Sherali-Adams hierarchy in 2.2.2 and the non-recursive definition in 2.2.3 are equivalent, i.e., every point in cone defined in 2.2.2 is inside the cone defined in 2.2.3 and vice versa.*

Proof. Proof is by induction on r, the level of the Sherali-Adams hierarchy. The base case $r = 0$ is trivial. Since for $r = 0$, the above product is simply $a^T x - b \geq 0$. Thus its linearization is $\sum_{i=1}^n a_i x_{\{i\}} - b x_\emptyset \geq 0$, which is the constraint $(a')^T y^{(1)} \geq 0$ defined in the first case of the recursive definition. Now for the induction step, consider a constraint $\gamma \geq 0$ of $\mathrm{SA}_r(\mathcal{P})$. From the recursive definition, $\gamma \geq 0$ is the linearization of $C = x_i * (a^T y^{(r-1)}) \geq 0$ (or $C'' = (1 - x_i) * (a^T y^{(r-1)}) \geq 0$), where $\beta = a^T y^{(r-1)} \geq 0$ is a constraint of $\mathrm{SA}_{r-1}(\mathcal{P})$. Thus by induction hypothesis, $\beta = \Phi(B)$, where $B = (a^T x - b) \prod_{s \in W} x_s \prod_{s \in U \setminus W} (1 - x_s) \geq 0$, $U \in \mathscr{P}_{r-1}(n)$. So, $\Phi(C) = \Phi(x_i * \beta) = \Phi(x_i * B)$, where $x_i * B = (a^T x - b) \prod_{s \in W \cup \{x\}} x_s \prod_{s \in U \setminus W} (1 - x_s) \geq 0$, considering $W' = W \cup \{x_i\}$, and $U = U \cup \{x_i\}$, we have $U' \subseteq \mathscr{P}_r(n)$. So the linearization of C, $\Phi(C)$, is in the promised form. So every constraint of $\mathrm{SA}_r(\mathcal{P})$ is in the form of linearization of a product like 2.2.1. Similarly we can prove the converse, i.e., the linearalization of every expression like 2.2.1 is also a constraint of $\mathrm{SA}_r(\mathcal{P})$. □

2.2.4 Definition. *Define the projection of $\mathrm{SA}_r(\mathcal{P})$ to the singleton sets as the r-th round Sherali-Adams relaxation of P. We denote it by $S_r(\mathcal{P}) = \{x \in \mathbb{R}^n | \exists y \in \mathrm{SA}_r(\mathcal{P})$ s.t. $y_\emptyset = 1$ and $x_i = y_{\{i\}}, \forall i \in [n]\}$.*

2.2.3 Remark. Note that in contrast to other Lift and Projects methods based on semidefinite programming such as Lasserre hierarchy, the r-th round of Sherali-Adams relaxation is indeed a polytope itself (in other methods like Lasserre, we only know that each level of the hierarchy is a convex body).

In theorem 2.2.4 we show Sherali-Adams relaxation $S_r(\mathcal{P})$ of the polytope $\mathcal{P}$ preserves all integral solutions. In other words, the Sherali-Adams relaxation tightens the polytope $\mathcal{P}$ such that we don't loose any feasible integral solution to $\mathcal{P}$. Furthermore, each level of Sherali-Adams relaxation is contained in the previous level; i.e., as we increase the level of Sherali-Adams relaxation, we indeed have a stronger tightening of the initial polytope $\mathcal{P}$, and finally after at most n-th level, we will end up with the integral hull of $\mathcal{P}$, which allows us to solve an integer programming defined on $\mathcal{P}$. However as it is expected, the running time to solve the n-th level of Sherali-Adams relaxation is exponential in the number of variables.

2.2.4 Theorem. *if $\bar{x} \in P \cap \{0,1\}^n$ then $\bar{x} \in S_r(\mathcal{P})$ for every $r \geq 0$.*

Proof. Let $\bar{x}$ be an integral solution of $\mathcal{P}$. For every $S \in \mathscr{P}_{r+1}(n)$ define $y_S = \prod_{i \in S} \bar{x}_i$. We want to show that $y \in \mathrm{SA}_r(\mathcal{P})$ and hence $\bar{x} \in S_r(\mathcal{P})$. We prove this by induction on r. The case $r = 0$ is trivial since $\mathcal{P} = S_0(\mathcal{P})$.Suppose we know $y \in \mathrm{SA}_{r-1}(\mathcal{P})$ now we want to show that $y \in \mathrm{SA}_r(\mathcal{P})$. For variable x_i and constraint $a^T y^{(r-1)} \geq 0$ in the set of constraints defining $\mathrm{SA}_{r-1}(\mathcal{P})$, we have two constraints in $\mathrm{SA}_r(\mathcal{P})$. For the first constraint, $x_i * (a^T y^{(r-1)}) \geq 0$, If $\bar{x}_i = 0$ we have $x_i * y_S = y_{S \cup \{i\}} = \bar{x}_i . \prod_{i \in S} \bar{x}_i = 0$. So $x_i * (a^T y^{(r-1)}) = 0 \geq 0$. If $\bar{x}_i = 1$ then $x_i * y_S = y_{S \cup \{i\}} = \bar{x}_i . \prod_{i \in S} \bar{x}_i = y_S$. Therefore $x_i * (a^T y^{(r-1)}) = a^T y^{(r-1)}$. Note that from induction hypothesis $a^T y^{(r-1)} \geq 0$ since $y \in \mathrm{SA}_{r-1}(\mathcal{P})$. Therefore, $(1 - x_i) * (a^T y^{(r-1)}) \geq 0$. Hence, in both cases $\bar{x}_i = 0$ and $\bar{x}_i = 1$, $y^{(r)}$ satisfies the first constraint. The argument for the second constraint is similar. Thus, $y \in \mathrm{SA}_r(\mathcal{P})$. □

The following lemma shows that every level of the hierarchy is contained in its previous level.

2.2.5 Theorem. *$S_r(\mathcal{P}) \subseteq S_{r-1}(\mathcal{P})$ for every $r \geq 1$.*

Proof. Based on the recursive definition 2.2.2, the set of constraints defining $\mathrm{SA}_{r-1}(\mathcal{P})$ is subset of the set of constraints defining $\mathrm{SA}_r(\mathcal{P})$. Therefore every point satisfies constraints of $\mathrm{SA}_r(\mathcal{P})$ will satisfy constraints of $\mathrm{SA}_{r-1}(\mathcal{P})$, thus $\mathrm{SA}_r(\mathcal{P}) \subseteq \mathrm{SA}_{r-1}(\mathcal{P})$. Also since $S_r(\mathcal{P})$ and $S_{r-1}(\mathcal{P})$ are both the projection of $\mathrm{SA}_r(\mathcal{P})$ and $\mathrm{SA}_{r-1}(\mathcal{P})$ respectively to $y_\emptyset = 1$, we have $S_r(\mathcal{P}) \subseteq S_{r-1}(\mathcal{P})$. □

In the following we describe the main property of the Sherali-Adams relaxation, which is called *local decompsition*, or *local consistency*. In stronger hierarchies such as Lasserre hierarchy, we have a stronger version of this property which is called *decomposition* theorem [31]. Let's fix a subset S of variables. The local decomposition theorem states that every point in the r-th level of hierarchy, can be written as a convex combination of some points in the lower level of hierarchy ($r - |S|$-level), that are integral on S.

To begin with, we first define *conditioning* a solution on a variable x_i, and then generalize it to a set of variables. The intuition behind conditioning comes from an interesting probabilistic interpretation of the Sherali-Adams hierarchy. We can consider the normalization of every $y \in \mathrm{SA}_r(\mathcal{P})$ with $y_\emptyset > 0$, i.e., $\frac{y}{y_\emptyset}$ (its projection to $y_\emptyset = 1$), as a probability distribution over every integral solutions on the subset S, with $|S| \leq r$, such that for every $A \subseteq S$, the y_A represents the probability of the event $x_i = 1 \ \forall i \in A$, assuming $y_\emptyset = 1$:

$$Pr[x_i = 1 \ \forall i \in A] = y_A$$

From inclusion-exclusion formula one can see the probability of the event $x_i = 1 \ \forall i \in A \ \& \ x_i = 0 \ \forall i \in S \setminus A$ is

$$Pr[x_i = 1 \ \forall i \in A \ \& \ x_i = 0 \ \forall i \in S \setminus A] = \sum_{B : A \subseteq B, \ B \subseteq S} (-1)^{|S \setminus B|} y_B$$

In other words, we can associate every subset $I \subseteq [n]$ such that $|I| \leq r$ with a distribution $D(I)$ of integral assignments on I. In addition these probability distributions $D(I)$ are consistent, i.e., for every two subsets I and J, with $|I|, |J| \leq r$, for $W \subseteq I \cap J$ and $U \subseteq W$ we have $Pr_{D(I)}[x_i = 1 \ \forall i \in U \ \& \ x_i = 0 \ \forall i \in W \setminus U] = Pr_{D(J)}[x_i = 1 \ \forall i \in U \ \& \ x_i = 0 \ \forall i \in W \setminus U]$.

As an extreme case for n-th round of SA relaxation, we have distribution $D(I)$ associated with $I = [n]$, thus we have probability distribution α_T over all integral assignments $T \in \{0,1\}^n$ of all variables. Because those probabilities sum up to 1, we can write $\sum_{T \in \{0,1\}^n} \alpha_T = 1$. So this gives a convex combination over integral solutions on all variables. Hence we can write every $y \in \mathrm{SA}_n(\mathcal{P})$ as a convex combination of integral solutions, i.e., y is in integral hull of $\mathcal{P}$. You can see [17] for the proof of the above properties for every solution in the r-th round of SA relaxation.

With this interpretation, the notion of conditioning a solution $y \in S_r(\mathcal{P})$ on a set S is derived from conditional probability distributions. As mentioned above consider the probability distribution $D(I)$ over integral solutions, associated with set I. Conditioning on a set S intuitively means we consider the conditional distribution D(I) on the events $(x_i = 1 \ \forall i \in S)$ or $(\exists i \in S \ s.t. \ x_i = 0)$. Thus if we denote them by $u = y'_S$ and $v = y'_{-S}$ respectively, the probability associated for a set I with $|I| \leq r$ would be $u_I = \frac{y_{I \cup S}}{y_S}$ from conditional probability. However, we independently define the notion of conditioned solutions and our proofs will be based on the following definitions.

2.2.5 Definition. *Let $y \in \mathrm{SA}_r(\mathcal{P})$ with $y_\emptyset > 0$, we denote by y'_i and y'_{-i}, conditioning on a variable x_i, which is defined as $(y'_i)_S := y_{S \cup \{i\}}$, and $(y'_{-i})_S := y_S - (y'_i)_S, \forall S \in \mathscr{P}_r(n)$. Hence y'_i and y'_{-i} are in $\mathbb{R}^{\mathscr{P}_r(n)}$.*

Clearly from the above definition $y|_{\mathscr{P}_r(n)} = y'_i + y'_{-i}$. Also, in the next lemma, we prove that $y'_i, y'_{-i} \in \mathrm{SA}_{r-1}(\mathcal{P})$. In addition, if $u = y'_{-i}$ and $v = y'_i$, call $\mu = \frac{u}{u_\emptyset} \in S_{r-1}(\mathcal{P})$ and $\nu = \frac{v}{v_\emptyset} \in S_{r-1}(\mathcal{P})$, the *normalization* of u and v. We can see $\nu_{\{i\}} = \frac{v_{\{i\}}}{v_\emptyset} = \frac{y_{\{i\}}}{y_{\{i\}}} = 1$, and $\mu_{\{i\}} = \frac{u_{\{i\}}}{u_\emptyset} = \frac{y_{\{i\}} - y_{\{i\}}}{u_\emptyset} = 0$. i.e., normalization of conditioned solutions are integral (zero-one) at the i-th coordinate. So $y = \alpha.\mu + \beta.\nu$, where $\alpha = u_\emptyset$ and $\beta = v_\emptyset$. Therefore we have written y as a convex combination of two points (μ and ν) in one lower level of the Sherali-Adams relaxation of $\mathcal{P}$, S_{r-1}. By conditioning a point $y \in \mathrm{SA}_r(\mathcal{P})$ on a variable x_i, we obtain points in one lower layer of hierarchy such that their normalizations are zero-one at the i-th coordinate, and we can write y as sum of them.

2.2.6 Lemma. *$w \in \mathrm{SA}_r(\mathcal{P})$ iff $w'_{-i}, w'_i \in \mathrm{SA}_{r-1}(\mathcal{P})$ for every $i \in [n]$.*

Proof. First suppose $w \in \mathrm{SA}_r(\mathcal{P})$ and $i \in [n]$. From definition we have $w|_{\mathscr{P}_r(n)} = w'_{-i} + w'_i$. Also, $w \in \mathrm{SA}_r(\mathcal{P})$, so for every constraint $a^T y^{(r-1)} \geq 0$ in $\mathrm{SA}_{r-1}(\mathcal{P})$, we have constraints $x_i * a^T y^{(r-1)} \geq 0$ and $(1 - x_i) * a^T y^{(r-1)} \geq 0$. Hence,

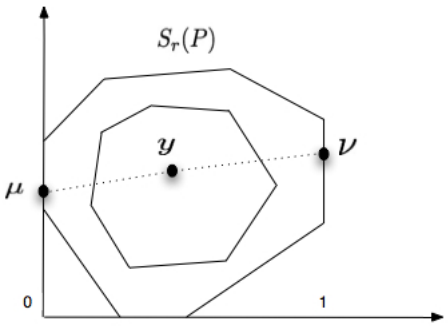

Figure 2.2: Conditioning on a variable x_i, a point y can be written as a convex combination of two conditioned solutions which are in lower level of hierarchy

$$x_i * a^T w = \sum_{S \in \mathscr{P}_r(n)} a_S^T (x_i * w_S) \qquad = \sum_{S \in \mathscr{P}_r(n)} a_S^T (w_{S \cup \{i\}}) \tag{2.2.2}$$

$$= \sum_{S \in \mathscr{P}_r(n)} a_S^T (w'_i)_S = a^T w'_i \qquad \geq 0 \tag{2.2.3}$$

So $a^T w'_i \geq 0$ for every $i \in [n]$ iff the constraint $x_i * a^T w \geq 0$ holds. Thus $w'_i \in \mathrm{SA}_{r-1}(\mathcal{P})$ iff $w \in \mathrm{SA}_r(\mathcal{P})$.
Similarly, from the second constraint we will have $(1 - x_i) * a^T w = a^T w - x_i * a^T w = a^T w - a^T w'_i \geq 0$ so $a^T (w - w'_i) \geq 0$ or $a^T w'_{-i} \geq 0$ iff the constraint $(1 - x_i) * aw$ holds for every $i \in [n]$. So $w \in \mathrm{SA}_r(\mathcal{P})$ iff $w'_{-i}, w'_i \in \mathrm{SA}_{r-1}(\mathcal{P})$ for every $i \in [n]$. □

One way of generalizing the notion of *conditioning* on a set I of variables is that the normalization of conditioned solution is a solution in a lower level of the hierarchy, with 1 on all variables in I.

2.2.6 Definition. *Suppose $y \in \mathrm{SA}_r(\mathcal{P})$, and $I \subseteq [n]$ with $|I| \leq r$ then conditioned solution y'_I is defined as $(y'_I)_S := y_{S \cup I}$ for each $S \in \mathscr{P}_{r-|I|+1}(n)$.*

We can prove the following theorem:

2.2.7 Theorem. *Suppose $y \in \mathrm{SA}_r(\mathcal{P})$, then $y'_I \in \mathrm{SA}_{r-|I|}(\mathcal{P})$. Also if $u = \frac{y'_I}{y_I}$ is the normalization of y'_I, then $u_I = 1$.*

Proof. We can prove the claim by induction on the size of I. If $|I| = 1$, and $I = \{i\}$ then $y'_I = y'_i$ and $y'_i \in \mathrm{SA}_{r-1}(\mathcal{P})$ by lemma 2.2.6. If $|I| \geq 1$, then $(y'_I)_S = y_{S \cup \{I\}}$ from definition. If $i \in I$ and $I' = I \setminus \{i\}$ and $u = y'_i$ then $(u'_{I'})_S = u_{I' \cup S} = y_{I' \cup S \cup \{i\}} = y_{I \cup S}$, therefore $y'_I = u'_{I'}$ but $u' \in \mathrm{SA}_{r-1}(\mathcal{P})$ and from induction hypothesis $u'_{I'} \in \mathrm{SA}_{r-1-|I'|}(\mathcal{P}) = \mathrm{SA}_{r-|I|}(\mathcal{P})$ □

2.2.8 Lemma. *Suppose that $y \in \mathrm{SA}_r(\mathcal{P})$. Then for any $A \subseteq B \subseteq [n]$ with $|B| \leq r+1$, we have $y_B \leq y_A$.*

Proof. It can be proved by induction on $|B \setminus A|$ with the base case being $|B| = |A| + 1$. If $B = A \cup \{i\}$, where $i \notin A$ then using constraint $x_i \leq 1$ and from the non-recursive definition of SA, if we multiply both sides by $\Pi_{j \in A} x_j$ and then linearize it, we have $y_{A \cup \{i\}} \leq y_A$. Suppose we have proved for $|B \setminus A| < k$. Now for the case $|B \setminus A| = k$, if $B = B' \cup \{i\}$, where $i \notin B$, we can say $y_{B'} \leq y_A$ from induction hypothesis. Also $y_B \leq y'_B$. Therefore, we have $y_B \leq y_A$ □

We can also define *conditioning* on a subset of variables $S = I \cup J$ with $I \cap J = \emptyset$, such that the normalization of conditioned solution is all 1 on I, and all 0 on J. We can generalize the notion of conditioning on a set $S = \{i_1, i_2, \cdots, i_k\} \subseteq [n]$ with $|S| \leq r$ in a way that we recursively condition on each element i_j of S. It can be shown that the resulting expression is independent of order of elements (after we proved equivalence of non-recursive and recursive definitions). Then a non-recursive equivalent definition is presented in definition 2.2.12.

2.2.7 Definition. *Conditioning a solution $y \in \mathrm{SA}_r(\mathcal{P})$ on a set S, where $|S| \leq r$ with an integer vector $T \in \{0,1\}^S$, denoted by $y_S(T) \in \mathbb{R}^{\mathscr{P}_{r-|S|+1}(n)}$ is defined recursively as follows:*

Let $i \in S$ is an arbitrary element in S, set $S' = S \setminus \{i\}$, and say $w = y_{S'}(T')$, where T' is the restriction of T on S'.

$$y_S(T) := \begin{cases} w'_i, & \text{if } T_i = 1 \\ w'_{-i}, & \text{if } T_i = 0 \end{cases}$$

and the base case is defined as $y_\emptyset(\emptyset) := y$. Also define the normalization of $y_S(T)$ with $(y_S(T))_\emptyset > 0$, as $Y_S(T) = \frac{y_S(T)}{(y_S(T))_\emptyset}$.

2.2.9 Lemma. *$y_S(T) \in \mathrm{SA}_{r-|S|}(\mathcal{P})$. Also $Y_S(T)$ is integral on S, i.e., $(Y_S(T))_{\{i\}} \in \{0,1\}$ for $i \in S$.*

Proof. It can be proved by induction on the size of S. The base case $|S| = 1$ is proved in lemma 2.2.7. Let $i \in S$ and $S' = S \setminus \{i\}$ and say $w = y_{S'}(T')$. From the above recursive definition of $y_S(T)$, we have $y_S(T)$ is either w'_i or w'_{-i}. From induction hypothesis $w \in \mathrm{SA}_{r-|S'|}(\mathcal{P})$, thus from lemma 2.2.7, $w'_i, w'_{-i} \in \mathrm{SA}_{r-|S'|-1}(\mathcal{P}) = \mathrm{SA}_{r-|S|}(\mathcal{P})$.
For the second part, note that $w'_i = 1$ and $w'_{-i} = 0$. In addition, if $u = w'_{-i}$ and $v = w'_i$, then $\mu = \frac{u}{u_\emptyset} \in S_{r-1}(\mathcal{P})$ and $\nu = \frac{v}{v_\emptyset} \in S_{r-1}(\mathcal{P})$, are the normalization of u and v. So $Y_S(T)$ is either ν or μ. Also as we saw $\nu_{\{i\}} = 1$, and $\mu_{\{i\}} = 0$. Thus $(Y_S(T))_{\{i\}}$ is either $\mu_{\{i\}} = 0$ or $\nu_{\{i\}} = 1$. □

2.2.10 Theorem. *Let $y \in \mathrm{SA}_r(\mathcal{P})$ and $S = \{i_1, \cdots, i_k\} \subseteq [n]$ with $|S| \leq r$. Then $y|_{\mathscr{P}_{r-k+1}(n)} = \sum_{T \in \{0,1\}^S} y_S(T)$.*

Proof. We prove by induction on $|S|$. Let $S' = \{i_1, \cdots, i_{k-1}\}$. Let $u = y_{i_k}$ and $v = y_{-i_k}$ From definition 2.2.5, we know $y = u + v$, and from induction hypothesis we know

$$u = \sum_{T \in \{0,1\}^{S'}} u_{S'}(T) = \sum_{T \in \{0,1\}^S, T_{i_k}=1} y_S(T) \tag{2.2.4}$$

and

$$v = \sum_{T \in \{0,1\}^{S'}} v_{S'}(T) = \sum_{T \in \{0,1\}^S, T_{i_k}=0} y_S(T) \tag{2.2.5}$$

So $y|_{\mathscr{P}_{r-k+1}(n)} = u + v = \sum_{T \in \{0,1\}^S} y_S(T)$ □

From the above lemma we can conclude that for a set S with $|S| \leq r+1$, we can write every point in level r of the Sherali-Adams relaxation as a convex combination of some points in $r - |S|$ level, such that their normalizations are integral on the set S, as follows:

2.2.11 Theorem. *(local decomposition) Let $x \in S_r(\mathcal{P})$ and $S = \{i_1, \cdots, i_k\} \subseteq [n]$ with $|S| \leq r$ then we can write x as a convex combination of some points in $S_{r-|S|}$. In particular, $x|_{\mathscr{P}_{r-k+1}(n)} = \sum_{T \in \{0,1\}^S} \alpha_T Y_S(T)$, where $\sum_{T \in \{0,1\}^S} \alpha_T = 1$ and $\alpha_T \geq 0$ for every $T \in \{0,1\}^S$.*

Proof. Let $y \in \mathrm{SA}_r(\mathcal{P})$ and its projection is x. From theorem 2.2.10

$$y|_{\mathscr{P}_{r-k+1}(n)} = \sum_{T \in \{0,1\}^S} y_S(T) = \sum_{T \in \{0,1\}^S} (y_S(T))_\emptyset \frac{y_S(T)}{(y_S(T))_\emptyset} \tag{2.2.6}$$

Note that $y_S(T) \in \mathrm{SA}_{r-|S|}(\mathcal{P})$, so $\frac{y_S(T)}{(y_S(T))_\emptyset} \in S_{r-|S|}(P)$. Define $\alpha_T = (y_S(T))_\emptyset$ then

$$1 = y_\emptyset = \sum_{T \in \{0,1\}^S} (y_S(T))_\emptyset = \sum_{T \in \{0,1\}^S} \alpha_T. \tag{2.2.7}$$

□

We can give an equivalent non-recursive definition for definition 2.2.7, and this also shows that the above recursive definition is independent of the order of elements we pick in the set S.

2.2.12 Theorem. *(non-recursive definition of conditioning on a set) Let $S \subseteq [n]$, with $|S| \leq r$ and $S = I \cup J$, $I \cap J = \emptyset$. Also,*

$$T(i) := \begin{cases} 1, & \text{if } i \in I \\ 0, & \text{if } i \in J \end{cases}$$

Then, $(y_S(T))_K = \sum_{H \subseteq J} (-1)^{|H|} y_{I \cup H \cup K}$, for every $K \in \mathscr{P}_{r-|S|+1}(n)$.

Proof. We prove the theorem by induction on the size of S. Let $e \in S$ and $S' = S \setminus \{e\}$. If $T(e) = 1$ then $e \in I$. Let $w = y_{S'}(T)$, then by induction hypothesis

$$w_K = \sum_{H \subseteq J} (-1)^{|H|} y_{I' \cup H \cup K},$$

for every $K \in \mathscr{P}_{r-|S'|+1}(n)$, where $I' = I \setminus \{e\}$. Therefore,

$$(w'_e)_K = \sum_{H \subseteq J} (-1)^{|H|} y_{I \cup H \cup K}$$

For every $K \in \mathscr{P}_{r-|S'|}(n)$. Also since $y_S(T) = w'_e$, the induction step in the case $T(e) = 1$ is true. Now if $T(e) = 0$ it means $e \in J$. Let $w = y_{S'}(T)$, then by induction hypothesis

$$w_K = \sum_{H \subseteq J'} (-1)^{|H|} y_{I \cup H \cup K}$$

For every $K \in \mathscr{P}_{r-|S'|+1}(n)$. Then

$$(w'_e)_K = \sum_{H \subseteq J'} (-1)^{|H|} y_{I \cup \{e\} \cup H \cup K}.$$

For every $K \in \mathscr{P}_{r-|S'|}(n)$. Let $H' = H \cup \{e\}$, then $(w'_e)_K = \sum_{I \cup \{e\} \subseteq J} (-1)^{|H'|-1} y_{I \cup H' \cup K}$. Also from definition 2.2.7, $y_S(T) = w - w'_e$, thus

$$(y_S(T))_K = w_K - (w'_e)_K = \sum_{H \subseteq J'} (-1)^{|H|} y_{I \cup H \cup K} + \sum_{I \cup \{e\} \subseteq J} (-1)^{|H'|} y_{I \cup H' \cup K},$$

For every $K \in \mathscr{P}_{r-|S'|}([n]) = \mathscr{P}_{r-|S|+1}([n])$, where the first sum is over all subsets of J not containing e and the second sum is over all subsets of J containing e, therefore $(y_S(T))_K = \sum_{H \subseteq J} (-1)^{|H|} y_{I \cup H \cup K}$, for every $K \in \mathscr{P}_{r-|S|+1}([n])$. This completes the induction step. □

Finally we show that the nth level of the hierarchy is exactly the convex hull of integral solutions.

2.2.13 Theorem. $S_n(\mathcal{P}) = \mathcal{P}_I$.

Proof. Suppose $y \in \mathrm{SA}_n(\mathcal{P})$ and $x \in S_n(\mathcal{P})$ is its projection and $S = \{1, \cdots, n\}$ As a result of previous theorem $x|_{\mathscr{P}_{r-|S|+1}(n)} = \sum_{T \in \{0,1\}^S} \alpha_T y_S(T)$, where $\sum_{T \in \{0,1\}^S} \alpha_T Y_S(T) = 1$. But note that $Y_S(T)$ is integral on S, i.e $(Y_S(T))_{\{i\}} \in \{0, 1\}$. So x is a convex combination of some integral solutions in $S_n(\mathcal{P})$. So it is in the integral hull. □

Finally, assuming that optimizing over $\mathcal{P}$ can be done in polynomial time, the following theorem shows the running time required for solving an optimization problem over $S_r(\mathcal{P})$.

2.2.14 Theorem. *If there is a polynomial time separation oracle for $\mathcal{P}$, then there is a separation oracle for $S_r(\mathcal{P})$ with running time of $n^{O(r)}$, i.e., there is an $n^{O(r)}$ algorithm for optimizing a linear function over $S_r(\mathcal{P})$*

Proof. Using lemma 2.2.5 a separation oracle for $S_r(\mathcal{P})$ can be obtained by a separation oracle for $S_{r-1}(\mathcal{P})$ and calling it for each $i \in [n]$ for each of the conditioned solution on y'_i and y'_{-i}. Therefore we are calling the separation oracle $n^{O(r)}$ times. □

2.3 Example: Applying Sherali-Adams to the Gap Example of Cut Based LP-relaxation

In this section we want to show an example utilizing the decomposition theorem to show that first layer of Sherali-Adams hierarchy eliminates the well-known bad example for the natural cut based LP-relaxation of Undirected Steiner Tree problem with integrality gap of $2 - \epsilon$, for every $\epsilon > 0$. That example is the graph G an instance of the Undirected Steiner Tree problem that consists of a single ring on n terminal verices $X = \{1, \cdots, n\}$, and all edges have unit cost. The following is the cut based LP-relaxation of Undirected Steiner Tree, where X is the set of terminals and V the set of vertices and the set of edges, and r is the root.

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(\delta(S)) \geq 1 \qquad \forall\ S \subseteq V - r, S \cap X \neq \emptyset \\ & x_e \in [0,1] \quad \forall e \in E \end{aligned}$$

Suppose the set of edges of G are $E = \{e_1, \cdots, e_n\}$ and we denote by x_i and c_i the variable corresponding to e_i in the LP and the cost of edge e_i respectively. The following is the LP for the special graph G.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_{e_i} x_{e_i} \\ \text{s.t.} \quad & x_i + x_j \geq 1 \qquad \forall\ i \neq j \in [n] \\ & x_i \in [0,1] \quad \forall i \in [n] \end{aligned} \tag{Q}$$

The optimal solution for the Steiner tree problem is a minimum spanning tree, with $n-1$ edges. On the other hand, the standard LP has a solution where $x_i = 1/2$ for each edge of G. Therefore, the ratio of the optimal fractional solution to optimal integral solution in this example is $2(n-1)/n$ which approaches to 2 when n goes to infinity. In addition, since there is a 2 approximation algorithm based on this cut based LP-relaxation, the ratio can not be worse than 2.

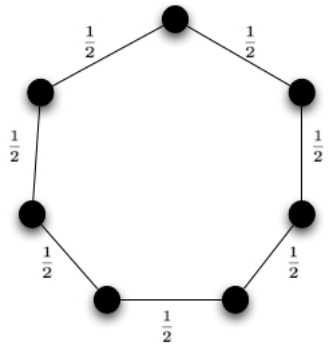

Figure 2.3: A ring of n vertices and a fractional solution that assigns 1/2 to each edge. All vertices are terminals.

First we apply one round of Sherali-Adams and see what is the worst integrality gap ratio on the special graph G. We will denote by OPT the optimal integral solution of the Undirected Steiner Tree LP-relaxation for the ring graph instance G. Let Q be the polytope for the above LP corresponding to the ring graph G, and suppose

$$OPT_{SA} = \min \left\{ \sum_{i=1}^{n} c_i \cdot y_{\{i\}} : y \in \mathrm{SA}_1(Q) \right\} \tag{SA1}$$

2.3.1 Lemma. *The integrality gap of the first level of Sherali-Adams tightening of* Q *is at most* 3/2. *i.e.,* $OPT/OPT_{SA} \leq 3/2$.

Proof. Suppose vector $y \in \mathrm{SA}_1(\mathrm{Q})$ is an optimal fractional solution for SA1, and $x \in \mathrm{S}_1(\mathrm{Q})$ is the projection of y over singleton variables. Then since the structure of G is symmetric, we may assume that $x_i = x_j = \alpha, \forall i, j$ (by taking average over all rotations of vertices of G). Now apply the local decomposition theorem by conditioning on an arbitrary edge i, we can write

$$y = \alpha w + (1 - \alpha) z, \tag{2.3.1}$$

where $w = y'_i$ and $z = y'_{-i}$ are the conditioning solutions which are integral on the i-th edge. Both $w, z \in \mathrm{SA}_0(\mathrm{Q})$ and therefore feasible solution for Q. But $z_{\{i\}} = 0$, so if we consider constraints $z_{\{i\}} + z_{\{j\}} \geq 1$, this implies that $z_{\{j\}} = 1, \forall j \neq i$. By looking at the j-th coordinate of 2.3.1 we have $y_{\{j\}} = \alpha w_{\{j\}} + (1 - \alpha) z_{\{j\}}$. Also we have $x_i = x_j, \forall j, i$ (symmetric), thus $y_{\{j\}} = \alpha, \forall j \neq i$. By equation 2.3.1, $\alpha = y_{\{j\}} = \alpha w_{\{j\}} + (1 - \alpha).1$. So, $w_{\{j\}} = \frac{\alpha}{2\alpha - 1}$. On the other hand since $y_{\{j\}} = \alpha$ and $z_{\{j\}} = 1$ for all $j \neq i$, then $w_{\{j\}} = \beta, \forall j \neq i$, for some $\beta > 0$.
Hence, from the constraint in Q we have $w_{\{j\}} + w_{\{k\}} \geq 1$. So $\beta \geq 1/2$. Consequently, $\frac{\alpha}{2\alpha - 1} \geq 1/2$ which yields $\alpha \geq 2/3$. As a result, $OPT_{SA} = \sum_{i=1}^{n} c_i \cdot y_{\{i\}} = \alpha.n$ and the integrality ratio OPT/OPT_{SA} is at most $\frac{n-1}{2/3n}$, which approaches 3/2 when n goes to infinity. □

Now we generalize this lemma for the k-th round of Sherali-Adams. Suppose

$$OPT_k = \min\left\{\sum_{i=1}^{n} c_i \cdot y_{\{i\}} : y \in \mathrm{SA}_k(Q)\right\} \tag{SAK}$$

2.3.2 Theorem. *Suppose $n > k+2$, then the integrality gap of the k-th level of Sherali-Adams tightening of* Q *is at most $\frac{k+2}{k+1}$. i.e., $OPT/OPT_k \leq \frac{k+2}{k+1}$.*

Proof. We use induction on k. The base case $k = 1$ is proved in the previous lemma. Now suppose $y \in \mathrm{SA}_k(\mathrm{Q})$ is an optimal fractional solution. Again since the structure of G is symmetric we may assume $y_{\{i\}} = \alpha, \forall i$. Similar to previous lemma apply the local decomposition theorem. For an arbitrary edge i, we can write

$$y = \alpha w + (1-\alpha)z, \tag{2.3.2}$$

where $w = y'_i$ and $z = y'_{-i}$ are the conditioning solutions which are integral on the i-th edge. Both $w, z \in \mathrm{SA}_{k-1}(\mathrm{Q})$ and therefore feasible solution for Q. But $z_{\{i\}} = 0$, so if we consider constraints $z_{\{i\}} + z_{\{j\}} \geq 1$, this implies that $z_{\{j\}} = 1, \forall j \neq i$. By equation 2.3.2, $\alpha = y_{\{j\}} = \alpha w_{\{j\}} + (1-\alpha).1$. So, $w_{\{j\}} = \frac{\alpha}{2\alpha-1}$. On the other hand, since $y_{\{j\}} = \alpha$ and $w_{\{j\}} = 1$ for all $j \neq i$. Then $\beta = w_{\{j\}}, \forall j \neq i$.
So we know $w_{\{i\}} = 1$ and $w_{\{j\}} = \beta, \forall j \neq i$. Before we complete the proof, we prove the following.

2.3.3 Remark. The variable $\gamma \in \mathrm{SA}_{k-1}(\mathrm{Q})$ defined as $\gamma_c = z_c, \forall c, |c| \leq k$, such that $j \notin c$ and $\gamma_c = \beta$ such that $j \in c$, where $\beta = \gamma_{c'}$ for $|c'| = |c|$ and $j \notin c'$ is also a feasible solution in $\mathrm{SA}_{k-1}(\mathrm{Q})$.

Proof. First of all note that γ is a symmetric variable, i.e., $\gamma_S = \gamma_{S'}$ for $|S| = |S'|$. Consider a constraint of $\mathrm{SA}_{k-1}(\mathrm{Q})$ corresponds to

$$C = \Phi(\Pi_{i\in I} x_i \Pi_{j\in J} x_j (x_l + x_m - 1)) \geq 0, \tag{2.3.3}$$

where $|I| + |J| \leq k$. If C does not contain variable y_S such that $j \in S$, then if z satisfies the constraint, γ also will satisfy it (they are the same except at coordinates S that $j \in S$). But if C contains y_S with $j \in S$, consider a variable i that there is no y_S in C that $i \in S$. There exists such a variable because $|I| + |J| \leq k < n$ and for the variables y_S in a constraint in the form of 2.3.3 $S \subseteq I \cup J \cup \{l, m\}$. Also $|I \cup J \cup \{l, m\}| \leq k+2 < n$, thus there exists a variable i that is not involved in any variable in C. Now consider a permutation π of variables that maps all variables y_S where $i \in S$ to $y_{S'}$ where $S' = S \setminus \{i\} \cup \{j\}$ because of the symmetry in the constraint of Q if we replace y_S with $y_{S'}$ in C it would be also another constraint C' of $\mathrm{SA}_{k-1}(\mathrm{Q})$. But C' does not contain a variable y_S with $j \in S$ so if z satisfies C', the γ also would satisfy it. But γ is symmetric, so γ satisfies C before applying π on variables as well. Hence γ satisfies all constraints of $\mathrm{SA}_{k-1}(\mathrm{Q})$. □

Now that we know $\gamma \in \mathrm{SA}_{k-1}(\mathrm{Q})$ we can use induction hypothesis and we have $\gamma_{\{i\}} = \beta \geq k/k+1$. Similar to previous lemma:

$$\frac{2\alpha - 1}{\alpha} \geq \frac{k}{k+1}$$

so,

$$\implies (2k+2)\alpha - (k+1) \geq k\alpha$$

$$\implies k\alpha + 2\alpha - k - 1 \geq 0$$

$$\implies \alpha(k+2) \geq k+1$$

$$\implies \alpha \geq \frac{k+1}{k+2}$$

Hence $\sum_{i=1}^{n} c_i.y_{\{i\}} = \alpha.n$ and the integrality ratio is $\frac{n-1}{(k+1)/(k+2)n}$, approaches $\frac{k+2}{k+1}$. □

2.4 Strengthened Integrality Gap Bound for Group Steiner Tree

Garg, Konjevod and Ravi [16] design a $O(\log n. \log k)$ approximation algorithm for Tree instances of Group Steiner Tree problem, where n is the number of vertices and k is the number of groups. Since their algorithm is based on the natural LP for Group Steiner Tree problem, they also bound the integrality gap of this problem on tree instances with the same amount. In this section, we provide a more granular bound and show that the gap is $O(h. \log k)$, where h is the height of tree.

2.4.1 Theorem. *The Group Steiner Tree problem admits an $O(h. \log k)$ approximation algorithm based on its natural LP-relaxation on tree instances of height h. In particular, the integrality gap of the LP-relaxation of Group Steiner Tree problem on these instances is $O(h.logk)$.*

To prove the above theorem, we follow the Garg, Konjevod and Ravi [16] randomized algorithm with the following two properties:

2.4.2 Lemma. *There exists a randomized algorithm that takes a tree instance I of the Group Steiner Tree problem and outputs a set of edges $\hat{E}$ such that:*

1. *E[cost of edges in $\hat{E}$] $\leq OPT(I)$*
2. *For any group g, $Pr[\hat{E}$ connects the root to $g] \geq \frac{1}{h}$*

First assume we have such a randomized algorithm. We run the algorithm $L = O(h. \log k)$ times, and construct the tree as described in their proof of Group Steiner Tree. Similarly we would have Pr[group g does not get connected in L round] $\leq (1 - 1/h)^L \leq 1/k$. Then, we take union of all of these edges. If some

group is still not connected to the root, we find the cheapest way to connect this group to the root by a path. The expected cost of final solution would be still $(L+1).OPT(I)$, as shown in their proof.

First consider the natural LP for the Group Steiner Tree problem.

$$\min \quad \sum_e c_e x_e \tag{2.4.1}$$

$$\text{s.t.} \quad \sum_{e \in \delta_{in}(S)} x_e \geq 1 \quad \forall S \subseteq V, \text{s.t } S \text{ separates } r \text{ from group } g_i \text{ for some } i$$

They round a fractional feasible solution to the above LP formulation as follows:

1. "Mark" each edge with probability $x_e/x_{p(e)}$. If there is no parent edge, then mark the edge with probability x_e.

2. Pick an edge e if all its ancestor edges are marked, and e itself is marked.

Then they show the following lemma that every edge will be picked with probability x_e.

2.4.3 Lemma. *$Pr[$edge e is picked$] = x_e$*

This lemma implies that $E[c(\hat{E})] \leq \sum_{e \in E} c_e x_e$ by linearity of expectation. Therefore we have the first property. Also as stated in the Garg, Konjevod and Ravi proof, we can show the following lemma.

For terminal vertex v in group g, suppose e is the incident edge to v in the tree. For convenience, we denote x_e by x_v.

2.4.4 Lemma. *$1/4 \leq \sum_{v \in g} x_v \leq 1$, for every group g.*

The above lemma gives us a lower bound on the expected number of root-paths which end in vertices of a group. Lets define the random variable Z to be the number of paths from the root to any vertex v in a group g. Therefore, we have $1/4 \leq E[Z] \leq 1$.

They claim if we have a new solution x', with $x'_e \leq x_e, \forall e \in E$. Then $Pr[$failure to connect g using $x'] \geq Pr[$failure to connect g using $x]$. They construct a minimal feasible x' in several steps by rounding all values of x_e to the nearest power of 2, and also removing edges with "small"" values. The minimality of x' implies that the flow going down to the leaves is exactly 1. The key insight of GKR is to prove an upper bound on Z in order to lower bound $Pr[Z \geq 1]$.

2.4.5 Lemma. *$E[Z|Z \geq 1] \leq O(h)$*

Proof. Let ϵ_i be the event that $P_i \subseteq S'$, i.e., the entire set P_i is chosen by the random process. For some vertex $u \in g$, let us first show that $\Delta_u = \sum_w Pr[\epsilon_u \cap \epsilon_w] \leq O(h).x'_u$. Summing this over all $u \in g$ gives us $O(h)$, since

the x'_u's sums up to at most 1 by minimality of x', and hence complete the proof.

For any $w \in g$, let e be the lowest edge shared by the path from u to the root, and the path from w to the root. Let the child edge of e on the path to w be denoted by c_e.

$$Pr[\epsilon_w|\epsilon_u] = \frac{x'_w}{x'_{p(w)}}.\frac{x'_{p(w)}}{x'_{p(p(w))}} \cdots .\frac{x'_{c_e}}{x'_e} = \frac{x'_w}{x'_e}. \tag{2.4.2}$$

Hence $Pr[\epsilon_u \cap \epsilon_w] = (x'_u.x'_w/x'_e)$. Now we sum over all w such that the paths from u to root and w to root have the edge e as their least common ancestor edge as above. Now use the fact that $\sum_{suchw} x'_w \leq O(x'_e)$, since all the flow going to such ws must have been supported on the edge e before rounding down. Hence, the sum of $Pr[\epsilon_u \cap \epsilon_w]$ over all the relevant w for some edge e is $O(x'_u)$. Now there are at most h edges on the path from u to the root. So we get $\Delta_u = O(h.x'_u)$. □

GKR make use of a sophisticated probabilistic result, the Janson inequality. However, the desired bound can be achieved much easier:

2.4.6 Lemma. $Pr[Z \geq 1] \geq \Omega(\frac{1}{h})$

Proof. By the law of total probability

$$1/4 \leq E[Z] = Pr[Z = 0].E[Z|Z = 0] + Pr[Z \geq 1].E[Z|Z \geq 1] \leq 0 + \Omega(h). \tag{2.4.3}$$

thus $Pr[Z \geq 1] \geq \Omega(\frac{1}{h})$. □

So the second property is also can be satisfied. Now as we mentioned in the beginning we repeat the algorithm $O(h.\log k)$ times, and connect every not connected group via shortest path. The probability that a group would not be connected is $(1 - 1/h)^k \leq 1/k$. So the expected cost of the final solution would be $L.OPT_f + \sum_g 1/k.OPT \leq (L + 1).OPT = O(h.\log k).OPT$.

2.4.7 Theorem. *The integrality gap of LP 2.4.1 (the input graph is tree) is $O(h.\log k)$, where h is the height of tree and k is the number of groups.*

Proof. As we stated at the beginning, We run the algorithm $L = O(h.\log k)$ times, and select the edges as described in the lemma 2.4.2. Let E_i is the set of edges we obtain after i-th iteration. Hence, Pr[group g does not get connected in L round] $\leq (1 - 1/h)^L \leq 1/k$. Then, we take union of all of these edges. If some group is still not connected to the root, we find the cheapest way to connect this group to the root by a path. The cost of the cheapest path from some vertex in g to the root is at most OPT. Since OPT connects some vertex v in g to the root r, the path in the optimal solution that connects v and r is at least the cheapest path that was taken as part of the solution. So the expected

cost of the final solution is

$$\begin{aligned}\sum_i E[\text{cost of } \hat{E}_i] + \sum_{\text{group g}} Pr[\text{g is not connected}].OPT \\ = L.OPT + \sum_{\text{group g}} Pr[\text{g is not connected}].OPT \\ \leq (L+1).OPT = O(h.\log k).OPT\end{aligned}$$

2.5 Layered Instances

Zelikovsky [54] introduces l-layered instances of the Directed Steiner Tree as follows:

2.5.1 Definition. *Say that an instance $G = (V, E)$ of DST with terminals X is ℓ-layered if V can be partitioned as $V_0, V_1, \ldots, V_\ell$ where $V_0 = \{r\}, V_\ell = X$ and every edge $uv \in E$ has $u \in V_i$ and $v \in V_{i+1}$ for some $0 \leq i < \ell$.*

He provided the following useful insight:

2.5.1 Theorem. *For every $l \geq 1$, there is a tree T (potentially using edges in the metric closure) of cost $c(T) = \sum_{e \in T} c_e \leq l.|X|^{1/l}.OPT$, such that every $r - s$ path(with $s \in X$) in T contains at most l edges.*

In other words, for any DST instance G and any integer $\ell \geq 1$, Zelikovsky show that there is an ℓ-layered DST instance H such that $OPT_G \leq OPT_H \leq \ell \cdot k^{1/\ell} \cdot OPT_G$ and that a DST solution in H naturally corresponds to a DST solution in G with the same cost [10]. Charikar et al. [11] exploit this fact and present an $O(\ell^2 k^{1/\ell} \log k)$-approximation[1] with running time $\text{poly}(n, k^\ell)$ for any integer $\ell \geq 1$. In particular, this can be used to obtain an $O(\log^3 k)$-approximation in quasi-polynomial time and for any constant $\epsilon > 0$ a polynomial-time $O(k^\epsilon)$-approximation.

Algorithm 1 in the next chapter is designed for these layered instances. We use a reduction from Directed Steiner Tree problem to Group Steiner Tree problem. In this reduction we convert an instance of Directed Steiner Tree to an instance of Group Steiner Tree problem.

2.5.1 Reduction from Directed Steiner Tree to Group Steiner Tree

Suppose G is an ℓ-layered instance of Directed Steiner Tree with root r, terminals X, and layers $\{r\} = V_0, V_1, \ldots, V_\ell = X$. We will assume every $v \in V$ can be reached by r. In particular, for every $v \in V_1$ we have $rv \in E$.

[1]The algorithm in [11] is presented as an $O(\ell k^{1/\ell} \log k)$-approximation and relied on an incorrect claim in [**?**]. A correction to this claim was made in [10] which gives in the stated DST approximation bound.

Say a path in G is *rooted* if it begins at r. The notation $\langle v_j, v_{j+1}, v_{j+2}, \ldots, v_i\rangle$ refers to a path in G that follows edges $v_jv_{j+1}, v_{j+1}v_{j+2}, \ldots, v_{i-1}v_i \in E$ in succession. The subscript of a vertex in this notation will always indicate which layer the node lies in. For any node $v \in V(G)$ we let

$$Q(v) = \{\langle r, v_1, v_2, \ldots, v_i\rangle : v_i = v\}$$

and for any $e \in E(G)$ we let

$$Q(e) = \{\langle r, v_1, v_2, \ldots, v_i\rangle : v_{i-1}v_i = e\}$$

denote all rooted paths ending at node v or ending with edge e, respectively. More generally, for a vertex v and another vertex u or an edge e, we let $Q(v, u)$ and $Q(v, e)$ denote all paths starting at v and ending at u or ending with edge e, respectively. It will also sometimes be convenient to think of a path as a set of edges $\{v_jv_{j+1}, \ldots, v_{i-1}v_i\}$.

2.5.2 Definition. *Suppose $G = (V, E)$ is an ℓ-layered instance of DST with root r and k terminals X. Then we consider the Group Steiner Tree instance on a tree $\mathcal{T}(G)$ with terminals $X_t, t \in X$ defined as follows.*

- *The vertex set of $\mathcal{T}(G)$ consists of all rooted paths $\cup_{v\in G}Q(v)$ in G.*
- *For any rooted path $P \neq \langle r\rangle$, we connect P to its maximal proper subpath and give this edge cost c_e, where $P \in Q(e)$. Denote this edge by $m(P)$.*
- *For each terminal $t \in X$, we let $X_t = Q(t)$: the set of all $r - t$ paths in G.*

This construction is illustrated in Figure 2.4.
The following holds simply by construction of $\mathcal{T}(G)$.

2.5.2 Lemma. *Let $|V| = n$. The graph $\mathcal{T}(G)$ constructed from an ℓ-layered Directed Steiner Tree instance G is a tree with height ℓ when rooted at $\langle r\rangle$. For every GST solution in $\mathcal{T}(T)$ there is a DST solution in G of no greater cost, and vice-versa.*

2.6 Lower Bound for the Integrality Gap of 4-Layer DST

In the following we describe the example by Zosin and Khuller, which has the integrality gap of $\Omega(\sqrt{k})$ Zossin and Khuller define a graph for which the integrality gap of LP P0 is as big as $\theta(\sqrt{k})$, where k is the number of terminals.

The graph G consists of 5 levels of nodes and 4 levels of edges. The 5 levels of nodes are:

- Level 1. Root r;
- Level 2. a node a_S for each set S of $\sqrt{k}$ elements from k elements, $\binom{k}{\sqrt{k}}$ nodes;

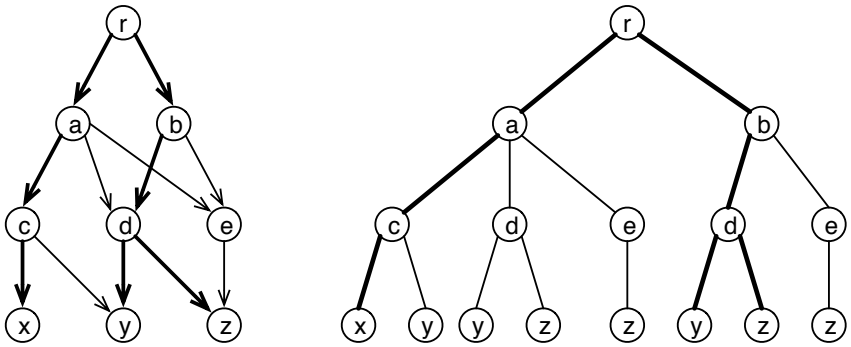

Figure 2.4: A 3-layered DST instance with terminals $X = \{x, y, z\}$ (left) and the corresponding GST instance $\mathcal{T}(G)$ (right). Each node in $\mathcal{T}(G)$ corresponds to a path P in G and is labelled in the figure with the endpoint of P in G. A terminal group in $\mathcal{T}(G)$ in the figure consists of all leaf nodes with a common label. A DST solution and its corresponding GST solution are drawn with bold edges.

- Level 3. a node $b_{S'}$ for each set S' of $\sqrt{k}+1$ elements from k elements, $\binom{k}{\sqrt{k}+1}$ nodes;
- Level 4. node $c_{S'}$ for each set S' of $\sqrt{k}+1$ elements from k elements, $\binom{k}{\sqrt{k}+1}$ nodes;
- Level 5. k terminals.

The four levels of edges are defined as follows:

- Level 1. root r is connected to every node a_S on level 2, and the weight of this edge is k;
- Level 2. a_S in level 2 connected to $b_{S'}$ in level S, off $S \subseteq S'$.
- Level 3. every node $b_{S'}$ on level 3 connected to a node $c_{S'}$ on level 4, that corresponds to the same set S' of $\sqrt{k}+1$ elements. The weight of this edge is $\sqrt{k}$;
- Level 4. a node $c_{S'}$ on level 4 is connected to terminal on level 5 if the terminal belongs to the set S' of the node, the weight of this edge is 0.

2.6.1 Lemma. *There exists a fractional feasible solution x for LP P0 with total cost of $\sum_{e \in E} c_e x_e = O(k)$.*

Proof. The argument is based on constructing a flow f_i correspond to each terminal i as follows. For each node a_S in level 2, push $\frac{k}{(k-\sqrt{k}).\binom{k}{\sqrt{k}}}$ units of flow

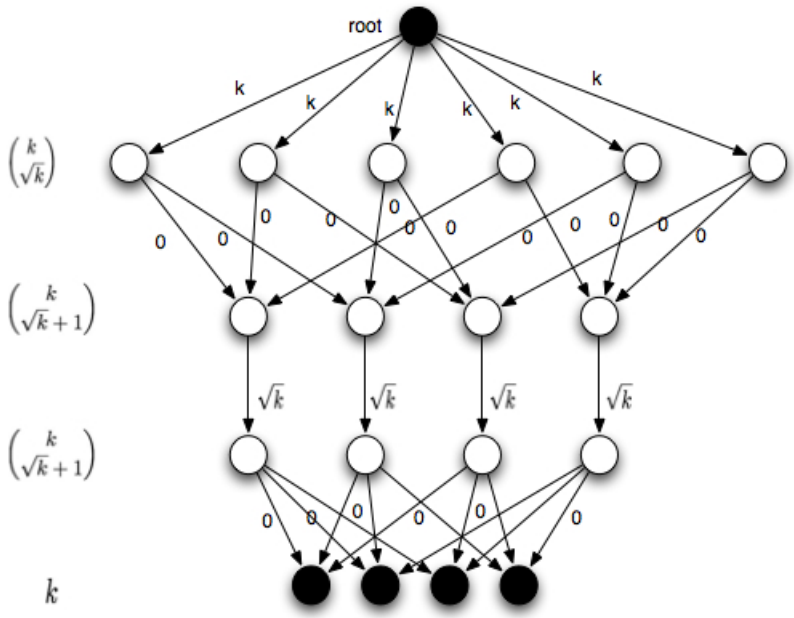

Figure 2.5: Zosin Khuller example for $k = 4$

f_i from root r to a_S of $k - \sqrt{k}$ commodities correspond to terminals $i \notin S$, and assign the value x_e of the edge e between r and a_S to be $\frac{k}{(k-\sqrt{k}).\binom{k}{\sqrt{k}}}$.

The total flow from r will be k and total fractional weight of edges on level 1 will be $\frac{k}{1-1/\sqrt{k}}$. From node a_S we push $\frac{k}{(k-\sqrt{k}).\binom{k}{\sqrt{k}}}$ units of flow of commodity i to the node $b_{S'}$ such that $i \in S'$. From node $b_{S'}$ we push all the flow of $\sqrt{k}+1$ commodities that enter then node to the node $c_{S'}$ and from it to the corresponding terminals. We assign x_e to be $\frac{k}{(k-\sqrt{k}).\binom{k}{\sqrt{k}}}$ for $e = b_{S'}c_{S'}$, and the total fractional weight of edges on level 3 will be $\frac{k}{1+1/\sqrt{k}}$. The flow is a feasible flow (for every vertex $v \notin X \cup \{r\}$ the incoming flow and outcoming flow are equal) with total fractional weight $O(k)$, and because of the symmetric structure of the graph, the total flow for all terminals are equal. As a result, the total flow for each terminal is equal to the total flow from r divided by k which is 1. Hence the assigned x variables satisfy cut constraints in the LP, so x is feasible fractional solution for the LP with $\sum_{e \in E} c_e x_e = O(k)$. □

2.6.2 Lemma. *The minimum Directed Steiner Tree, has weight at least* $\Theta(k\sqrt{k})$

Proof. Define the density of a tree (not necessarily covering all terminals but all leafs are terminal) to be its total weight divided by the number of terminals. In order to prove the lemma, we instead prove that the density of a tree T (all leafs

of T are terminals) is $\Theta(\sqrt{k})$. Let T be a directed tree with minimum density that spans t terminals. We can assume without loss of generality, that T has only one edge on level 1. Since otherwise, we could consider its minimal density subtree (the subtree with minimum density among all subtrees consisting of one edge $e = rv$ incident to the root and the subtree rooted at v). If $t \leq \sqrt{k}$ then its weight is at least k (the weight of edge from level 1) and its density is at least $\sqrt{k}$. If $t \geq \sqrt{k}$ then it has at least $t - \sqrt{k} + 1$ edges from level 3 (if $e = uv$ is the edge from level 1 and v corresponds to set S then every terminal vertex $t \notin S$ corresponds to different child of v, say $b_{S'}$, where $S' = S \bigcup t$) and its weight is at least $(n-\sqrt{k}+1).\sqrt{k}+k$ and its density is $\sqrt{k}$. Thus, the density of minimum Directed Steiner Tree, which is greater than the density of minimum density tree is at least $\sqrt{k}$. Hence, its total weight is at least $k\sqrt{k}$. □

2.6.3 Theorem. *The integrality gap of Directed Steiner Tree problem on the above instance is* $\Omega(\sqrt{k})$.

Chapter 3

Sherali-Adams Hierarchy and Directed Steiner Tree

We are given a directed graph $G = (V, E)$ with edge costs $c_e \geq 0, e \in E$. Furthermore, we are given a root node $r \in V$ and a collection of terminals $X \subseteq V$. Throughout, we will let $n = |V|$, $m = |E|$, and $k = |X|$.

As we stated in Chapter 1, a natural LP relaxation for Directed Steiner Tree is given by LP (P0).

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & \text{(P0)} \\ \text{s.t.} \quad & x(\delta^{\text{in}}(S)) \geq 1 \qquad \forall\, S \subseteq V - r, S \cap X \neq \emptyset & & (3.0.1) \\ & x_e \in [0,1] \quad \forall e \in E \end{aligned}$$

In Chapter 2, we saw that the integrality gap of this relaxation can, unfortunately, be as bad as $\Omega(\sqrt{k})$, even in instances where G is a 4-layered graph.

A related problem that will frequently occur throughout this chapter is the Group Steiner Tree (GST) problem mentioned in Chapter 1. As we proved in Chapter 2, unlike DST, the integrality gap of the natural LP relaxation (P4) (introduced in Section 3.1.1) is polylogarithmically bounded.

3.0.4 Theorem (Garg, Konjevod, and Ravi [16])**.** *The integrality gap of LP (P4) is $O(\min\{\ell, \log n\} \cdot \log k)$ in GST instances that have n nodes, k terminal groups, and are trees with height ℓ when rooted at r.*

Only the bound of $O(\log n \log k)$ is explicitly show in [16] but as we proved in Theorem 2.4.7, the bound $O(\ell \cdot \log k)$ easily follows from their techniques

3.0.1 Our Results and Techniques

For any feasible, minimal DST solution F, every node has indegree 1. We are then justified in adding the simple constraint $x(\delta^{in}(v)) \leq 1$ for each $v \in V - r$

to the standard LP formulation and obtain the following LP relaxation.

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e x_e && && \text{(P1)}\\ \text{s.t.} \quad & x(\delta^{\text{in}}(S)) \geq 1 && \forall\ S \subseteq V - r, S \cap X \neq \emptyset && (3.0.2)\\ & x(\delta^{in}(v)) \leq 1 && \forall\ v \in V - r && (3.0.3)\\ & x_e \in [0,1] && \forall e \in E \end{aligned}$$

Both LPs (P0) and (P1) have the same optimum integer solution cost.

Using the ellipsoid method, it is possible to design a separation oracle for the ℓ-th level lift of (P1) in the Sheral-Adams hierarchy with running time being polynomial in n and k^ℓ. However, we will start with a much simpler LP relaxation with only polynomially many constraints.

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e x_e && && \text{(P2)}\\ \text{s.t.} \quad & x(\delta^{\text{in}}(t)) \geq 1 && t \in X && (3.0.4)\\ & x(\delta^{in}(v)) \leq 1 && \forall\ v \in V - r && (3.0.5)\\ & x(\delta^{in}(v)) \geq x_e && \forall\ v \in V - (X \cup \{r\}), e \in \delta^{out}(v) && (3.0.6)\\ & x_e \in [0,1] && \forall e \in E \end{aligned}$$

While Constraints (3.0.6) are not present in LP (P1), they are satisfied by any optimal, integer or fractional, solution x to LP (P1): if $x_e > x(\delta^{in}(v))$ for some solution x to LP (3.0.6) then we could decrease x_e to $x(\delta^{in}(v))$ and maintain feasibility.

Our main result is the following. The notation $\text{SA}_\ell(\mathcal{P})$ (defined properly in Chapter 2) refers to the ℓ-th level lift of polytope $\mathcal{P} \subseteq [0,1]^n$ in the Sherali-Adams hierarchy, which can be optimized over in time that is polynomial in the size of LP (P2) and m^ℓ, where m is the number of constraints. Thus, we consider the following LP.

$$\min \left\{ \sum_{e\in E} c_e \cdot y^*_{\{e\}} : y \in \text{SA}_\ell(\mathcal{P}) \right\} \qquad \text{(P3)}$$

where $\mathcal{P}$ is the polytope given by the constraints of the LP relaxation (P2).

3.0.5 Theorem. *If the DST instance G is an ℓ-layered graph, then the integrality gap of LP (P3) is $O(\ell \cdot \log k)$.*

Furthermore, given oracle access to some fixed $y^* \in \text{SA}_\ell(\mathcal{P})$ (i.e., if we can somehow find a feasible solution $y^* \in \text{SA}_\ell(\mathcal{P})$) there is a randomized algorithm that, with high probability, finds a Directed Steiner Tree solution in expected time $O(\text{poly}(n))$ and with expected cost at most $O(\ell \cdot \log k)$ times the cost of y^*.

Rothvoss proved an analogous result for the Lasserre hiearchy, but his arguments relied on a particular decomposition theorem proven by Karlin, Mathieu, and Nguyen [31]. This decomposition theorem does not hold in weaker LP hierarchies such as Sherali-Adams, so we must proceed in a different manner.

At a high level, we prove Theorem 3.0.5 by mapping a point y^* in the Sherali-Adams lifted polytope into a LP solution with the same cost as y^* for a related Group Steiner Tree instance. Using Theorem 3.0.4, we find a GST solution with cost $O(\ell \cdot \log k)$ times the cost of y^* and this will naturally correspond to a DST solution in G.

As a special case, we obtain the following interesting bound that shows that even without lift-and-project techniques the integrality gap of the natural LP for 3-layer graphs is logarithmic.

3.0.6 Theorem. *The integrality gap of LP (P0) is $O(\log k)$ in 3-layered graphs.*

As with Theorem 3.0.5, this is obtained by mapping a point in LP (P0) to a LP solution for the corresponding GST instance. However, the restriction to only 3 layers allows us to accomplish this without the use of hierarchies. In contrast as we saw in Chapter 2, the integrality gap of LP (P0) is $\Omega(\sqrt{k})$ in some graphs with 4 layers.

3.1 Preliminaries

First we restate the properties of the Sherali-Adams hierarchy required for our proof. Their proofs are provided in Chapter 2. Consider a polytope $\mathcal{P} \subseteq \mathbb{R}^n$ specified by m linear constraints $\sum_{i=1}^{n} A_{j,i} \cdot x_i \geq b_j, 1 \leq j \leq m$. Suppose the "box constraints" $0 \leq x_i$ and $x_i \leq 1$ (equivalently, $-x_i \geq -1$) appear among these constraints for each $1 \leq i \leq n$.

We only use some of the many well-known properties of the Sherali-Adams hierarchy, the Theorem 2.2.7 and 2.2.8

3.1.1 Lemma. *Suppose $y \in \mathrm{SA}_r(\mathcal{P})$ for some $r \geq 0$. Then the following hold.*

- *Suppose $y \in \mathrm{SA}_r(\mathcal{P})$, and $I \in \mathscr{P}_r(n)$ then $y'_I \in \mathrm{SA}_{r-|I|}$. Therefore, if $y_I > 0$, $\frac{y_{I \cup S}}{y_I} \in S_{r-|I|}(\mathcal{P})$*
- *For any $A \subseteq B \subseteq [n]$ with $|B| \leq r+1$, we have $y_B \leq y_A$.*

As we stated in Chapter 2 (definition 3.1.2), the following holds simply by construction of $\mathcal{T}(G)$.

3.1.2 Lemma. *Let $|V| = n$. The graph $\mathcal{T}(G)$ constructed from an ℓ-layered Directed Steiner Tree instance G is a tree with height ℓ when rooted at $\langle r \rangle$. For every GST solution in $\mathcal{T}(T)$ there is a DST solution in G of no greater cost, and vice-versa.*

3.1.1 Rounding for 3-Layered Graphs

We first demonstrate that the natural LP relaxation (P2) for Directed Steiner Tree has an integrality gap of $O(\log k)$ in 3-layered graphs without using any lift-and-project machinery. As mentioned earlier, this complements the observation of Zosin and Khuller [25] that the integrality gap is $\Omega(\sqrt{k})$ in some 4-layered instances.

We show this by directly embedding a solution to the Directed Steiner Tree LP relaxation (P0) for some 3-layered instance G into a feasible LP solution to the Group Steiner Tree LP (P4) on instance $\mathcal{T}(G)$. The reason we can do this with 3-layered instances boils down to the fact that for any edge $e = uv$ that either $v \in X$ or $|Q(e)| = 1$ (Figure 2.4 also helps illustrate this). This property does not hold in general for instances with at least 4 layers.

Consider a Group Steiner Tree instance $H = (V, E)$ with root r, groups $X_1, X_2, \ldots, X_k \subseteq V$, and edge costs $c_e, e \in E$. The LP relaxation we consider for Group Steiner Tree is the following.

$$\min \quad \sum_{e \in E} c_e z_e \tag{P4}$$

$$\text{s.t.} \quad z(\delta(S)) \geq 1 \qquad \forall S \subseteq V - r, X_i \subseteq S \text{ for some group } X_i \tag{3.1.1}$$

$$z \geq 0$$

Now we can prove our warmup result which we state again for reference.

3.1.3 Theorem. *The integrality gap of LP (P0) is $O(\log k)$ when G is a 3-layered graph.*

Proof. Let $G = (V, E)$ be a 3-layered instance of Directed Steiner Tree with layers $\{r\} = V_0, V_1, V_2, V_3 = X$ and $\mathcal{T}(G)$ the corresponding Group Steiner Tree instance. Let x^* be an optimal solution to LP (P0). Note that for edge $uv \in E$ with $v \notin X$ there is a unique rooted path in G ending with e (i.e. $|Q(e)| = 1$).

We construct a feasible solution z^* to LP relaxation (P4) for the Group Steiner Tree instance $\mathcal{T}(G)$. For every edge $e = uv$ of G where $v \notin X$, we set $z^*_{m(P)} := x^*_e$ where $Q(e) = \{P\}$, and remember that $m(P)$ indicates the last edge in the path P. All that is left to set is the the z^*-value for the leaf edges of $\mathcal{T}(G)$.

To do this, fix a terminal $t \in X$. By the max-flow/min-cut theorem and Constraints (3.0.1), there is a flow f^t sending 1 unit of flow from r to t satisfying $f^t_e \leq x^*_e$ for every edge e. Furthermore, for each $e \in \delta^{in}(t)$ we may assume that $x^*_e = f^t_e$, otherwise we could reduce x^*_e while maintaining feasibility. Consider any path decomposition of f^t and say that this decomposition places weight w^t_P on a path $P \in Q(t)$. That is, $f^t_e = \sum_{P \in Q(t): e \in P} w^t_P$ for every edge $e \in G$. Then we set $z^*_{m(P)} := w^t_P$ for each $P \in Q(t)$.

We claim that z^* is a feasible solution for LP (P4) with cost equal to $\sum_{e \in E} c_e x^*_e$. To see why z^* is feasible, we prove for every group t that there is a flow g^t of value 1 from $\langle r \rangle$ to the nodes in X_t with $g^t_{m(P)} \leq z^*_{m(P)}$ for every edge $m(P), P$ of H. By the max-flow/min-cut theorem, this means every

constraint of (P4) is satisfied by z^*. That such a flow exists essentially follows from the path decomposition of the flow f^t. Recall that a path decomposition of f^t placed weight w^t_P on $P \in Q(t)$. So, for each group X_t we define a flow g^t in $\mathcal{T}(G)$ by $g^t_{m(P)} = \sum_{P^* \in Q(t): P \subseteq P^*} w^t_{P^*}$.

First we verify that $g^t \leq z^*$ on an edge-by-edge basis. Consider an edge $m(P)$ in $\mathcal{T}(G)$ with P not ending in X. Say $P \in Q(e)$, then we have

$$g^t_{m(P)} = \sum_{\substack{P^* \in Q(t) \\ P \subseteq P^*}} w^t_{P^*} = f^t_e \leq x^*_e = z^*_{m(P)}.$$

The last equality holds because $Q(e) = \{P\}$. If $m(P)$ is an edge in $\mathcal{T}(G)$ with $P \in Q(t)$ then $g^t_{m(P)} = w^t_P = z^*_{m(P)}$ by construction. Finally, if $P \in Q(t')$ for $t' \neq t$ then $g^t_{m(P)} = w^t_P = 0$. Therefore, g^t satisfies the capacities z^*.

Next we verify flow conservation of g^t at intermediate nodes. For every internal node $P \neq \langle r \rangle$ of $\mathcal{T}(G)$ with parent edge $m(P), P$, we have

$$\sum_{P' : P' = P \cup \{e\}} g^t_{m(P')} = \sum_{P' : P' = P \cup \{e\}} \sum_{\substack{P^* \in Q(t) \\ P' \subseteq P^*}} w^t_{P^*} = \sum_{\substack{P^* \in Q(t) \\ P \subseteq P^*}} w^t_{P^*} = g^t_{m(P)}$$

so g^t satisfies flow conservation at internal nodes of $\mathcal{T}(G)$. Furthermore,

$$g^t(\delta^{\text{out}}(\langle r \rangle)) = \sum_{\substack{P^* \in Q(t) \\ \langle r \rangle \subseteq P^*}} w^t_{P^*} = \sum_{P^* \in Q(t)} w^t_{P^*} = 1$$

so g^t consists of one unit of flow from $\langle r \rangle$ to the leaf nodes. Finally, we verify that all of the flow g^t terminates at a leaf node in X_t by simply noting that for $P \in Q(t')$ where $t' \neq t$ we have that no path $P^* \in Q(t)$ contains P as a subpath, so $g^t_{m(P)} = 0$.

Next we show that both x^* and z^* have the same cost in their respective LPs. We do this by showing $\sum_{P \in Q(e)} z^*_{m(P)} \leq x^*_e$ for every edge e of G and recalling that the cost of the edge $m(P), P$ in $\mathcal{T}(G)$ is equal to c_e for any $P \in Q(e)$. For every $e \in E$ not ending in X, we have $Q(e) = \{P\}$ for some rooted path P. We had set $z^*_{m(P)} = x^*_e$ so $\sum_{P \in Q(e)} z^*_{m(P)} = x^*_e$ is trivially true for such edges. Finally, consider some $e \in E$ that ends in X. Then

$$\sum_{P \in Q(e)} z^*_{m(P)} = \sum_{P \in Q(e)} w^t_P = f^t_e = x^*_e.$$

The last equality holds by our earlier observation for edges in $\delta^{in}(t)$. Now we know z^* is feasible for LP (P4) and z^* has the same cost as x^*. By Theorem 3.0.4, there is a Group Steiner Tree solution of cost at most $O(\log k)$ times the cost of x^*. We conclude by using Lemma 3.1.2 to note that there is then a Directed Steiner Tree solution of cost at most $O(\log k)$. □

3.2 Rounding for ℓ-Layered Graphs

Our basic approach for proving Theorem 3.0.5 is similar to our approach for Theorem 3.0.6. We show how to embed a point y in the Sherali-Adams lift of LP (P2) for an instance G to a feasible solution to LP (P4) for the corresponding Group Steiner Tree instance $\mathcal{T}(G)$. Let $\mathcal{P}$ denote the polytope defined by the constraints of LP (P2).

Describing the embedding is straightforward. For every edge $m(P)$ in $\mathcal{T}(G)$, simply set $z^*_{m(P)} := y^*_P$. The rest of our analysis shows that z^* is feasible for LP (P4) for instance $\mathcal{T}(G)$ and the cost of z^* in $(P4)$ is equal to $\sum_{e\in E} c_e \cdot y^*_{\{e\}}$.

Before delving into the proofs of these statements, we establish a technical result about the structure of Sherali-Adams solutions which will be very helpful.

3.2.1 Lemma. *Suppose $0 \le i < j \le \ell$. For any node $v \in V_i$, any edge $e = uw$ with $w \in V_j$, and any $y \in \mathrm{SA}_{j-i}(\mathcal{P})$ we have $\sum_{P\in Q(v,e)} y_P \le y_{\{e\}}$. Furthermore, if $v = r$ then this bound holds with equality.*
Note that $|P| = j - i$ for any $P \in Q(v, e)$ so it is valid to index $y \in \mathrm{SA}_{j-i}(\mathcal{P})$ with P in the sum.

Proof. We prove the lemma by induction on $j - i$. The base case $j = i + 1$ is trivial since either $Q(v, e) = \emptyset$ (so the sum in question is 0) or $Q(v, e)$ consists of the singleton path that only uses edge e (so the sum in question is just $y_{\{e\}}$ already). Furthermore, if $v = r$ then $e \in \delta^{out}(r)$ so the bound holds with equality.

Inductively, suppose $j > i+1$. If $y_{\{e\}} = 0$ then by Lemma 3.1.1 we have $y_P \le y_{\{e\}} = 0$ for every $P \in Q(v, e)$ so the bound holds with equality. Otherwise, define the conditioned solution $y'_{\{e\}} \in \mathrm{SA}_{j-i-1}(\mathcal{P})$ and let $\gamma \in \mathrm{S}_{j-i-1}(\mathcal{P})$ be its normalization. So $\gamma_I = \frac{y_{I\cup\{e\}}}{y_{\{e\}}}$ for every $I \subseteq E, |I| \le i - j$ (cf. Lemma 3.1.1). Then

$$\begin{aligned}\sum_{P\in Q(v,e)} y_P &= \sum_{e'\in\delta^{in}(u)} \sum_{P\in Q(v,e')} y_{P\cup\{e\}} = y_{\{e\}} \sum_{e'\in\delta^{in}(u)} \sum_{P\in Q(v,e')} \gamma_P \\ &\le y_{\{e\}} \sum_{e'\in\delta^{in}(u)} \gamma_{\{e'\}} = y_{\{e\}}\end{aligned}$$

where the inequality follows by induction (note that the endpoint of e' is in V_{j-1}). The last equality follows by Constraints (3.0.5) and (3.0.6) of LP (P2) plus the fact that $\gamma_{\{e\}} = 1$. Finally, if $v = r$ then the inequality above holds with equality by induction, so $\sum_{P\in Q(v,e)} y_P = y_{\{e\}}$. □

3.2.1 Cost Analysis

The cost bound is an easy consequence of Lemma 3.2.1.

3.2.2 Lemma. *The cost of z^* in LP (P4) is $\sum_{e\in E(G)} c_e \cdot y^*_{\{e\}}$.*

Proof.

$$\begin{aligned}\sum_{m(P)\in E(\mathcal{T}(G))} c_{m(P)} \cdot z^*_{m(P)} &= \sum_{e\in E(G)} \sum_{P\in Q(e)} c_e \cdot z^*_{m(P)} \\ &= \sum_{e\in E(G)} \sum_{P\in Q(e)} c_e \cdot y^*_P = \sum_{e\in E(G)} c_e \cdot y^*_{\{e\}}\end{aligned}$$

where the last equality is by Lemma 3.2.1 applied with $v = r$. □

3.2.2 Feasibility

Similar to the proof of Theorem 3.0.6, for every group X_t we construct a one unit of $\langle r\rangle - X_t$ flow g^t in $\mathcal{T}(G)$ which satisfies the capacities given by z^*. Thus, by the max-flow/min-cut theorem we have that $z^*(\delta(S)) \geq 1$ for every subset $S \subseteq V(\mathcal{T}(G)) - \langle r\rangle$ such that $X_t \subseteq S$ for some group X_t.

We now fix a terminal $t \in X$ and describe the flow g^t by giving a path decomposition of the flow. For each $P \in Q(t)$, we assign a weight of y^*_P to the $\langle r\rangle - P$ path in $\mathcal{T}(X)$. So, the flow $g^t_{m(P)}$ crossing edge $m(P)$ in $\mathcal{T}(G)$ is just $\sum_{P^*\in Q(t):P\subseteq P^*} y^*_{P^*}$.

3.2.3 Lemma. *g^t is one unit of $\langle r\rangle - X_t$ flow in $\mathcal{T}(G)$.*

Proof. It is an $\langle r\rangle - X_t$ flow because we constructed it from a path decomposition using only paths in $Q(t)$. Furthermore,

$$\begin{aligned}g^t(\delta^{out}_{\mathcal{T}(G)}(\langle r\rangle)) &= \sum_{P^*\in Q(t):\langle r\rangle\subseteq P^*} y^*_{P^*} = \sum_{P\in Q(t)} y^*_P \\ &= \sum_{e\in\delta^{in}_G(t)} \sum_{P\in Q(e)} y^*_P = \sum_{e\in\delta^{in}_G(t)} y^*_{\{e\}} = 1.\end{aligned}$$

Here, the second last equality follows from Lemma 3.2.1. The last equality follows from combining Constraint (3.0.4) with Constraint (3.0.5) for $v = t$. □

All that is left is to prove that each flow g^t for a terminal group X_t satisfies the capacities given by z^*. The following lemma is the heart of this argument. A similar result was proven in [50] which relied on the strong decomposition property for the Lasserre hierarchy of Karlin, Mathieu, and Nguyen [31]. We emphasize that our proof only uses properties of the Sherali-Adams LP hierarchy.

3.2.4 Lemma. *For every rooted path P and every terminal group X_t, we have $\sum_{P^*\in Q(t):P\subseteq P^*} y^*_{P^*} \leq y^*_P$.*

Proof. If $y^*_P = 0$ this is trivial since $y^*_{P^*} \leq y^*_P$ for any $P^* \supseteq P$ by Lemma 3.1.1. Otherwise, form the conditioned solution $y'_P \in \mathrm{SA}_{\ell-|P|}(\mathcal{P})$ and its normalization say γ, we have $\gamma_I = \frac{y^*_{P\cup I}}{y^*_P}$ for any $|I| \leq \ell + 1 - |P|$.

Say v is the endpoint of P. Note that $\gamma \in \mathrm{S}_{\ell-|P|}(\mathcal{P})$ by Lemma 3.1.1. So, for any $e \in \delta^{in}(t)$ we have $\sum_{P^* \in Q(v,e)} \gamma_{P^*} \leq \gamma_{\{e\}}$ from Lemma 3.2.1 (with $i = |P|$ and $j = \ell - |P|$). Summing over all $e \in \delta^{in}(t)$ and using Constraints (3.0.5), we have $\sum_{P^* \in Q(v,t)} \gamma_{P^*} \leq 1$.

Multiplying both sides of this bound by y^*_P, we see that $\sum_{P^* \in Q(v,t)} y^*_{P \cup P^*} \leq y^*_P$. But the left hand side is precisely $\sum_{P^* \in Q(t): P \subseteq P^*} y^*_{P^*}$, which we were required to show. □

We can now easily verify that the capacity constraints are satisfied.

3.2.5 Corollary. *For every terminal group X_t and every edge $m(P)$ of $\mathcal{T}(G)$, $g^t_{m(P)} \leq z^*_{m(P)}$.*

Proof. By Lemma 3.2.4

$$g^t_{m(P)} = \sum_{\substack{P^* \in Q(t) \\ P \subseteq P^*}} y^*_{P^*} \leq y^*_P = z^*_{m(P)}$$

□

3.2.3 A Rounding Algorithm

We have proved the integrality gap bound in Theorem 3.0.5 by demonstrating that there is a LP solution to LP (P4) for instance $\mathcal{T}(G)$ of no greater cost, by using the integrality gap bound in Theorem 3.0.4, and by using the fact that a feasible solution to the Group Steiner Tree instance $\mathcal{T}(G)$ can be mapped to a feasible solution to the Directed Steiner Tree instance G with no greater cost.

In fact, we can emulate the rounding algorithm of [16] in GST instance $\mathcal{T}(G)$ without explicitly constructing z^* or $\mathcal{T}(G)$. Algorithm 1 contains the main subroutine in the rounding algorithm; this is just the main subroutine in the Group Steiner Tree algorithm in [16]. Note that if P' is a subpath of P, then $y^*_P \leq y^*_{P'}$ by Lemma 3.1.1. So, this algorithm behaves exactly the same as the corresponding Group Steiner Tree algorithm in the instance $\mathcal{T}(G)$ with LP solution z^*.

Let F be the set of edges returned from one run of Algorithm 1. The analysis of [16] shows $\mathrm{E}[\mathrm{cost}(F)] = OPT_f$ where OPT_f is the optimum LP solution value to LP (P3). Furthermore, the probability that any particular terminal t is reached by F is at least $\Omega\left(\frac{1}{\ell}\right)$ (c.f. 2.4.6).

3.2.6 Theorem. *Repeating Algorithm 1, $2 \cdot \ell \cdot \log k$ times and taking the union of the returned edge sets is, with probability at least $\frac{1}{2}$, a feasible Directed Steiner Tree solution with cost $O(\ell \cdot \log k) \cdot OPT_f$.*

Proof. Let H be the union of the returned edges in all iterations. The probability that a fixed terminal $t \in X$ is not connected is bounded by $(1 - \frac{1}{l+1})^{2l \log k} \leq \frac{1}{2k}$. Thus by union bound the probability that at least one terminal is not connected is at most $\sum_{t \in X} \frac{1}{2k} = 1/2$.

Such a DST solution can be found with high probability by repeating the algorithm $\log_2 n$ times. After repeating $\log_2 n$ times the above procedure the probability that we have still unconnected terminal is at most $(1-1/2)^{\log_2 n} \leq 1/n$. The solution can be pruned to a Directed Steiner Tree so that it is an integer solution to LP (P2) if desired.

Algorithm 1 **Directed Steiner Tree Rounding Subroutine**

$S_0 \leftarrow \langle r \rangle$
for $i = 1, \ldots, \ell$ **do**
  $S_i \leftarrow \emptyset$
  for each $P \in S_{i-1}$ **do**
    for each $e \in \delta^{out}(v)$ where v is the endpoint of P **do**
      Add $P \cup \{e\}$ to S_i with probability $\frac{y^*_{P\cup\{e\}}}{y^*_P}$
    end for
  end for
end for
$F \leftarrow$ edges used by some path in S_ℓ
return F

In fact, it is easy to see that in one run of Algorithm 1 we have for any rooted path P that $\Pr[P \in S_{|P|+1}] = y^*_P$. This leads to an interesting observation.

3.2.7 Lemma. $\mathrm{E}\left[\sum_{i=0}^{\ell} |S_i|\right] \leq n$

Proof. As noted before, for any edge $e = uv$ with $v \in V_i$ and any $P \in Q(e)$ we have $\Pr[P \in S_i] = y^*_P$. Thus, $\mathrm{E}[|S_i \cap Q(e)|] = \sum_{P\in Q(e)} y^*_P = y^*_{\{e\}}$ where the second equality is by Lemma 3.2.1. Summing over all edges e and using Constraints (3.0.5) shows $\sum_{e\in E} y^*_{\{e\}} \leq n-1$. Finally, since $y^*_{\langle r\rangle} = y^*_\emptyset = 1$ then $\mathrm{E}\left[\sum_{i=0}^{\ell} |S_i|\right] \leq n$. □

This also completes the proof of Theorem 3.0.5 since the total number of iterations of the loop in Step (4) of Algorithm 1 is precisely $\sum_{i=0}^{\ell-1} |S_i|$, which is polynomial in expectation. Therefore, with high probability the running time of the entire rounding algorithm is polynomial in n. Furthermore from Theorem 3.2.6 we can have a feasible directed Steiner tree with high probability.

3.3 Conclusion

We showed that only ℓ rounds of the Sherali-Adams hierarchy suffice to reduce the integrality gap of LP (P2) to $O(\ell \cdot \log k)$ in ℓ-layered graphs. In fact, our analysis shows that we could have omitted the constraints of the form (2.2.3) where $U \setminus W \neq \emptyset$ and still achieved the same integrality gap bound. This is because all arguments in our analysis that require "conditioning" never condition a variable to 0.

Chapter 4

A Logarithmic Integrality Gap Bound for Directed Steiner Tree in Quasi-bipartite Graphs

In this chapter we consider the class of *quasi-bipartite* DST instances. We are given a quasi-bipartite directed graph $G = (V, E, c)$ with cost $c : E \to \mathbb{R}^+$ on edges, a vertex root $r \in V$, and a set of terminal vertices $X \subseteq V$. Thus, there is no edge between Steiner vertices.

Our main theorem is the following.

4.0.1 Theorem. *The integrality gap of LP (P0) is at most $3 \cdot H_K = O(\log k)$ in quasi-bipartite graphs with k terminals where H_k is the k'th Harmonic number. Furthermore, a Steiner tree witnessing this integrality gap bound can be constructed in polynomial time.*

Theorem 4.0.1 is also tight since any of the well-known $\Omega(\log k)$ integrality gap constructions for set cover instances with k items translate directly to an integrality gap lower bound for LP (P0), using the usual reduction from Set Cover to 2-layered quasi-bipartite instances of Directed Steiner Tree.

We prove the above theorem by designing a primal-dual algorithm using LP (P0). The algorithm constructs a Directed Steiner Tree in an iterative manner. An iteration starts with a *partial* Steiner tree, consisting of multiple directed components containing the terminals in X. The algorithm constructs a feasible solution for the dual of (P0), and augments the given partial solution by a carefully chosen collection of arcs. The cost of these arcs can be bounded using the dual solution, and adding these arcs to the partial starting solution reduces the number of connected components.

There are a few examples of primal-dual algorithms for directed network design problems, for instance *the strong connectivity problem* [43]. Often the

difficulty of overlapping *moats* (the growing sets in a primal-dual algorithm that we increase their associated dual variables by time until one constraint goes tight and then we expand the corresponding moat) in instances of such problems is hard to control. We are able to do this here, exploiting the quasi-bipartite nature of our instances crucially. On the other hand, the primal-dual method is quite successful in undirected graphs: Goemans and Williamson give a primal-dual 2-approximation for the more general Steiner forest problem [18].

We note that Hibi and Fujito [28] had previously given a greedy algorithm that achieves a performance ratio of $O(\log k)$ for quasi-bipartite DST instances. Their approach iteratively chooses *low-density full Steiner trees* and adds them to the existing partial solution similar to [10]. This approach seems unlikely to yield integrality gap bounds for DST.

4.1 The Rounding

We now present an algorithmic proof of Theorem 4.0.1. As we will follow a primal-dual strategy, we give the LP dual of (P0) first.

$$\max \quad \sum_S y_S \tag{D}$$

$$\text{s.t.} \quad \sum_{S: e \in \delta^{\text{in}}(S)} y_S \leq c_e \ \forall e \in E \tag{4.1.1}$$

$$y \geq 0$$

In LP (D), the sums range only over sets of nodes S such that $S \subseteq V - r$ and $S \cap X \neq \emptyset$.

Our algorithm builds up partial solutions in the following sense.

4.1.1 Definition. *A* partial Steiner tree *is a tuple* $\mathcal{U} = (\{B_i, h_i, F_i\}_{i=0}^{\ell}, \overline{B})$ *where, for each* $0 \leq i \leq \ell$, B_i *is a subset of nodes,* $h_i \in B_i$, *and* F_i *is a subset of edges with endpoints only in* B_i *such that the following hold.*

- *The sets* $B_0, B_1, \ldots, B_\ell, \overline{B}$ *form a partition* V.
- $\overline{B} \subseteq V - X - r$ *(i.e.* $\overline{B}$ *is a subset of Steiner nodes).*
- $h_0 = r$ *and* $h_i \in X$ *for each* $1 \leq i \leq \ell$.
- *For every* $0 \leq i \leq \ell$ *and every* $v \in B_i$, F_i *contains an* $h_i - v$ *path.*

We say that $\overline{B}$ *is the set of* free Steiner nodes *in* $\mathcal{U}$ *and that* h_i *is the* head *of* B_i *for each* $0 \leq i \leq \ell$. *The edges of* $\mathcal{U}$, *denoted by* $E(\mathcal{U})$, *are simply* $\cup_{i=0}^{\ell} F_i$. *We say that* $B_1, \ldots, B_\ell$ *are the* non-root components *of* $\mathcal{U}$ *and that* $B_0, \ldots, B_\ell$ *are simply the* components *of* $\mathcal{U}$.

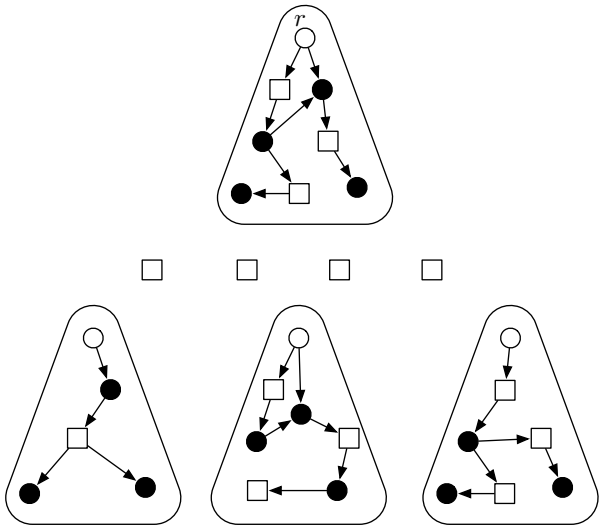

Figure 4.1: A partial Steiner tree with $\ell = 3$ non-root components (the root is pictured at the top). The only edges shown are those in some F_i. The white circles are the heads of the various sets B_i and the black circles are terminals that are not heads of any components. The squares outside of the components are the free Steiner nodes $\overline{B}$. Note, in particular, that each head can reach every node in its respective component. We do not require each F_i be a minimal set of edges with this property.

Figure 4.1 illustrates a partial Steiner tree. Note that if $\mathcal{U}$ is a partial Steiner tree with $\ell = 0$ non-root components, then $E(\mathcal{U})$ is in fact a Steiner tree.

Our algorithm essentially builds up partial Steiner trees in an iterative manner while taking care to ensure the cost does not increase by a significant amount between iterations. Specifically, we prove the following in Section 4.2. Let OPT_f refer to the optimum solution value for LP (P2).

4.1.1 Lemma. *Given a partial Steiner tree $\mathcal{U}$ with $\ell \geq 1$ non-root components, there is a polynomial-time algorithm that finds a partial Steiner tree $\mathcal{U}'$ with ℓ' non-root components where $0 \leq \ell' < \ell$ such that $E(\mathcal{U}) \subseteq E(\mathcal{U}')$ and*

$$\text{cost}(E(\mathcal{U}')) \leq \text{cost}(E(\mathcal{U})) + 3 \cdot OPT_f \cdot \frac{\ell - \ell'}{\ell}.$$

Theorem 4.0.1 follows from Lemma 4.1.1 in a standard way. *Proof.* [of Theorem 4.0.1] Initialize a partial Steiner tree $\mathcal{U}_k$ with k non-root components as follows. Initially, $\overline{B}$ is the set of all Steiner nodes, $B_0 = \{r\}$ and $F_0 = \emptyset$. Furthermore, label the terminals as $t_1, \ldots, t_k \in X$ and for each $1 \leq i \leq k$ let $B_i = \{t_i\}, h_i = t_i$ and $F_i = \emptyset$. Note that $\text{cost}(E(\mathcal{U}_k)) = 0$.

Iterate Lemma 4.1.1 to obtain a sequence of partial Steiner trees $\mathcal{U}_{\ell_0}, \mathcal{U}_{\ell_1}, \mathcal{U}_{\ell_2}, \ldots, \mathcal{U}_{\ell_a}$ where $\mathcal{U}_{\ell_i}$ has ℓ_i non-root components, $k = \ell_0 > \ell_1 > \ldots, > \ell_a = 0$ and such that $E(\mathcal{U}_i) \subseteq E(\mathcal{U}_{i+1})$ with $\text{cost}(E(\mathcal{U}_{i+1})) \leq \text{cost}(E(\mathcal{U}_i)) + 3 \cdot OPT_f \cdot \frac{\ell_i - \ell_{i+1}}{\ell_i}$ for each $0 \leq i < a$. Return $E(\mathcal{U}_{\ell_a})$ as the final Steiner tree.

That $E(\mathcal{U}_a)$ can be found efficiently follows simply because we are iterating the algorithm mentioned in Lemma 4.1.1 at most k times. The cost of this Steiner tree can be bounded as follows.

$$\begin{aligned}
\text{cost}(E(\mathcal{U}_a)) &\leq 3 \cdot OPT_f \cdot \sum_{i=0}^{a-1} \frac{\ell_i - \ell_{i+1}}{\ell_i} = 3 \cdot OPT_f \cdot \sum_{i=0}^{a-1} \sum_{j=\ell_{i+1}+1}^{\ell_i} \frac{1}{\ell_i} \\
&\leq 3 \cdot OPT_f \cdot \sum_{i=0}^{a-1} \sum_{j=\ell_{i+1}+1}^{\ell_i} \frac{1}{j} = 3 \cdot OPT_f \cdot \sum_{j=1}^{k} \frac{1}{k} \\
&= 3 \cdot OPT_f \cdot H_k.
\end{aligned}$$

□

4.2 The Primal-Dual Phase

In this section, we prove Lemma 4.1.1 with a primal-dual algorithm. Similar to a usual primal-dual algorithm, we start with a feasible solution $y_S = 0,\ \forall S \subseteq V - r,\ S \cap X \neq \emptyset$, for the dual program and an infeasible solution $x_e = 0\ \forall e \in E$ for the primal program. Then we increase dual variables for some growing subsets of vertices we call them *moats*. At the beginning they only include terminal vertices $M_1 = \{t_1\}, \cdots, M_k = \{t_k\}$. For the moat sets M_i, we simultaneously increase dual variables y_{M_i} by a parameter *time* that we indicate by Δ until one constraint, say $\sum_{S:e \in \delta^{in}(S)} y_S \leq c_e$, for some edge $e = uv \in \delta^{in}(M_i)$, in the dual program goes tight ($\sum_{S:e \in \delta^{in}(S)} y_S = c_e$). This happens after time $\epsilon = min_{i, e \in \delta^{in}(M_i)}(c_e - \sum_{S:e \in \delta^{in}(S)} y_S)$. Then we grow the moat M_i by adding the vertex v to M_i.

The main difference of our primal-dual algorithm with usual ones such as the primal-dual algorithm for Undirected Steiner Tree problem is that in those algorithms the growing moats are disjoint (if they collide then the algorithm terminates), but in our algorithm they are allowed to have intersection and by taking advantage of a novel structure that we define, the partial Steiner tree, and the fact that the graph is quasi-bipartite, we handle the overlap of the moats. The intuition is that each moat would have a subgraph which we will later call it the *virtual body* of the moat. We will instead show that the virtual body of the moats are disjoint. Also we will establish some invariants which hold during growing the moats, which help us to control the overlaps. Normally, primal-dual methods that simultaneously grow around multiple "moats" fail to produce good solutions in directed graphs. This is primarily due to the fact that it is difficult to keep the moats disjoint. However, the quasi-bipartite structure allows us to control the extent to which these moats overlap.

Fix a partial Steiner tree $\mathcal{U} = (\{B_i, h_i, F_i\}_{i=0}^{\ell}, \overline{B})$ with $\ell \geq 1$ non-root components. Lemma 4.1.1 promises a partial Steiner tree $\mathcal{U}'$ with $\ell' < \ell$ non-root components with $E(\mathcal{U}) \subseteq E(\mathcal{U}')$ and $\text{cost}(E(\mathcal{U}')) \leq \text{cost}(E(\mathcal{U})) + 3 \cdot OPT_f \cdot \frac{\ell - \ell'}{\ell}$ At a high level, this will be accomplished by simultaneously growing dual around each of the ℓ heads $h_1, \ldots, h_\ell$ until some stopping condition is satisfied. If the growing process takes Δ units of time, then the fact that we maintain feasibility in the dual ensures the total dual packed ($\sum_S y_S$) has value at most $\ell \cdot \Delta \leq OPT_f$. Roughly speaking, once the stopping condition is satisfied we will find a collection of edges F' with cost at most $3 \cdot \Delta \cdot b$ that connects the heads of $b \geq 1$ non-root components to the head of some other non-root component $B_{i'}$. The component $B_{i'}$ absorbs these other b components and we set $F_{i'}$ to the union of the F_i for the absorbed components B_i, plus the edges in F'.

We remark that the idea of growing dual until some termination condition happens, and selecting a subset of edges, then repeating the primal-dual algorithm and combining all selected edges until we find a feasible solution was also used in the [21] to show the integrality gap of a natural relaxation for undirected node-weighted Steiner tree is $O(\log k)$.

Our algorithm is described in Algorithm 2. It maintains a collection of "moats" $M_i \subseteq V$ and edges F_i' for each $1 \leq i \leq \ell$, while ensuring that the dual solution y remains feasible. To help describe what should be done when an edge goes tight, Algorithm 2 also maintain a "virtual body" $\beta_i \subseteq V$ for each $0 \leq i \leq \ell$ that contains B_i and some free Steiner nodes from $\overline{B}$ in the moat M_i. This virtual body will maintain the property that every free Steiner node in β_i can be reached from B_i by along some edge $e \in F_{i'}$.

Note that we will not grow a moat around $r = h_0$, as no sets S in the dual contain r, but we will still maintain a virtual body β_0 for notational simplicity even though β_0 will not change throughout the growing phase.

The following invariants will be maintained, where we say Δ is the total time the algorithm has been raising dual variables.

1. For each $1 \leq i \leq \ell$, $h_i \in M_i$ and $M_i \subseteq V - r$ so there is a variable y_{M_i} in the dual. Furthermore, all edges in F_i' have both endpoints in M_i and for every $v \in M_i$ there is a unique $v - h_i$ path in the subgraph $G(M_i, F_i')$ with cost at most Δ.

2. $M_i \cap M_j \subseteq \overline{B}$ and $M_i \cap \beta_j = \emptyset$ for distinct $1 \leq i, j \leq \ell$. Furthermore, $M_i \cap \beta_0 = \emptyset$ for all $1 \leq i \leq \ell$.

3. For each $1 \leq i \leq \ell$ we have $B_i \subseteq \beta_i \subseteq B_i \cup M_i$. Furthermore, for each $v \in \beta_i - B_i$ there is some $uv \in F_i'$ with $u \in B_i$. Finally, $\beta_0 = B_0$.

4. y is feasible for LP (D) with value exactly $\ell \cdot \Delta$.

The usual conventions of primal-dual algorithms will be adopted. We think of the dual-growing procedure as a continuous process that increases the value of some dual variables over time. Say that some edge e *goes tight* if the dual constraint for e becomes tight as the dual variables are being increased. If multiple edges go tight at the same time, then we process them in any order

with the understanding that if, say, both e and e' become tight at the same time and e is processed first, then e' is only processed after e if e still lies on some set $\delta^{in}(M_j)$ for some moat M_j after the moats are updated from processing e.

We remark that when the condition in Step (6) is checked then there is certainly some moat M_i with $uv \in M_i$. However, it might be that $j = i$ in which case we will also require some other i' to have $v \in M_{i'}$.

Algorithm 2 Dual Growing Procedure

[h!]

1: $M_i \leftarrow \{h_i\}, 1 \leq i \leq \ell$
2: $F'_i \leftarrow \emptyset, 1 \leq i \leq \ell$
3: $\beta_i \leftarrow B_i$ for $0 \leq i \leq \ell$
4: $y \leftarrow \mathbf{0}$
5: Raise $y_{M_{i'}}$ uniformly with variable Δ indicating time parameter, for each moat $M_{i'}$ until some edge uv goes tight
6: **if** $u \in \beta_j$ for some $0 \leq j \leq \ell$ **and** $v \in M_{i'}$ for some $i' \neq j$ **then**
7: **return** the partial Steiner tree $\mathcal{U}'$ referenced in Lemma 4.2.4.
8: **else**
9: Let M_i be the unique moat with $uv \in \delta^{in}(M_i)$ ▷ c.f. Proposition 4.2.2
10: $M_i \leftarrow M_i + u$
11: $F'_i \leftarrow F'_i + uv$
12: **if** $u \in \beta_i$ **then**
13: $\beta_i \leftarrow \beta_i + v$
14: **go to** Step (5)

4.2.1 Lemma. *Invariants 1-4 are maintained by Algorithm 2. Furthermore, the condition in Step (6) eventually becomes true (i.e. the algorithm terminates).*

Proof. Clearly the invariants are true after the initialization steps. To see termination, note that each iteration increases the size of a moat by 1 and leaves the other moats unchanged. The algorithm cannot iterate the main loop indefinitely because we would eventually have that some moat M_j would have some $uv \in \delta^{in}(M_j)$ go tight where $u \in B_0 = \beta_0$. In this case the algorithm would terminate[1].

Invariant 1:
To see the first condition, simply note that the sets M_i and F'_i are grown in the same way that a BFS tree is grown in the primal-dual interpretation of Dijkstra's algorithm. If an edge rv from the root goes tight where $v \in M_i$, then $r \in \beta_0$ and $v \in M_i$ would cause the algorithm to terminate, so r is never added to a moat.

[1]Implicitly, we are assuming here that there is some $r - t$ path in G for every $t \in X$. However, this must be the case as, otherwise, there is no feasible DST solution.

Invariant 4:
Feasibility is clear since we never raise a dual variable y_{M_j} if some edge $e \in \delta^{in}(M_j)$ is tight. Algorithm 2 raises the value of exactly ℓ dual variables at a time. So the total value of the dual is exactly $\ell \cdot \Delta$.

Now suppose the conditions hold at the moment some edge $e = uv$ goes tight and suppose the termination conditions in Step (6) are not satisfied. We show that the Invariants 2 and 3 continue to hold after the updates in Steps (10)-(13) are executed.

We establish the following proposition before verifying these invariants.

4.2.2 Proposition. *If the algorithm reaches Step (9) after uv goes tight, then there is exactly one moat M_i such that $uv \in \delta^{in}(M_i)$.*

Proof. There is at least one since uv went tight. If $e \in \delta^{in}(M_j)$ as well for some $j \neq i$ then $v \in M_i \cap M_j$ so $v \in \overline{B}$ by Invariant 2. But since G is quasi-bipartite, then $u \in X + r$ so $u \in \beta_{j'}$ for some $0 \leq j' \leq r$. But then the termination condition in Step (6) would have been satisfied with $u \in \beta_{j'}$ and $v \in M_i \cap M_j$ where one of $i \neq j'$ or $j \neq j'$ because $i \neq j$. □

Now let i be the index of the unique moat M_i with $uv \in \delta^{in}(M_i)$ from Step (9).

Invariant 2:
The only change to a moat this iteration is that u is added to M_i. If $(M_i + u) \cap \beta_j \neq \emptyset$ for some $j \neq i$ then the algorithm would have terminated because $u \in \beta_j$ and $v \in M_i$. So, $(M_i + u) \cap \beta_j = \emptyset$.

Now suppose $(M_i + u) \cap M_j \not\subseteq \overline{B}$ for some $j \neq i$. Then $u \in M_j$ and $u \in B_{j'}$ for some j'. By Invariant 3, $B_{j'} \subseteq \beta_{j'}$ so $u \in \beta_{j'}$ as well. But then Invariant 2 and $u \in M_j \cap \beta_{j'}$ means $j = j'$ so $j' \neq i$. But $u \in \beta_{j'}$ and $j' \neq i$ contradicts what we showed in the previous paragraph: $(M_i + u) \cap \beta_{j'} = \emptyset$.

Finally, the only possible change to some virtual body is that v might have been added to β_i. This can only happen if $u \in \beta_i$. In this case, for any $j \neq i$ we claim that $M_j \cap (\beta_i + v) = \emptyset$. Otherwise $v \in M_j$ for some $j \neq i$, but then the algorithm would have terminated with $u \in \beta_i$ and $v \in M_j$.

Invariant 3:
If v is not added to β_i in Step (13), then there is nothing to show. Also, it cannot be that $i = 0$ because we do not grow moats around $r = h_0$. Thus, β_0 continues to be B_0.

If v is added to β_i then $v \in M_i$ so $\beta_i + v \subseteq B_i \cup M_i$ continues to hold. We also note that for every $j \neq i$, $v \notin \beta_j$ by Invariant 2 and the fact that $v \in M_i$. Since $\beta_i \cap \beta_j = \emptyset$ and $v \notin \beta_j$, then $(\beta_i + v) \cap \beta_j = \emptyset$

For the final statement, suppose $v \notin B_i$. Then $v \in \overline{B}$ by the fact that $v \in M_i$ and $M_i \cap B_j \subseteq M_i \cap \beta_j = \emptyset$ for $j \neq i$ (Invariants 2 and 3). Since $v \in \overline{B}$, since G is quasi-bipartite, and since uv is an edge of G, then $u \notin \overline{B}$.

Now, since $u \in \beta_i$ then either $u \in B_i$ or $u \in M_i - B_i$. The latter cannot happen, because $u \notin \overline{B}$ means $u \in B_j$ for some j. But $u \in M_i - B_i$ means $j \neq i$ which contradicts $u \in M_i \cap B_j \subseteq M_i \cap \beta_j$ for $j \neq i$ (Invariants 2 and 3 again).

Thus, $u \in B_i$. Since the edge uv is also added to F'_i, then the last statement of Invariant 3 continues to hold when v is added to β_i.

□

To complete the analysis of the algorithm, we now describe how to construct the partial Steiner tree after Step (6) has been reached. The correctness of this construction is proven in Lemma 4.2.1.

The invariants enforce the following property.

4.2.3 Claim. *Invariants 2 and 3 imply $\beta_i \cap \beta_j = \emptyset$ for any distinct $0 \leq i, j \leq \ell$.*

Proof. If $1 \leq i, j$ then $\beta_i \cap \beta_j \subseteq (B_i \cap M_i) \cap (B_j \cup M_j)$ by Invariant 3. We have $B_i \cap B_j = \emptyset$ and $B_i \cap M_j = B_j \cap M_i = \emptyset$ by Invariants 2 and 3, so $\beta_i \cap \beta_j \subseteq M_i \cap M_j$. If $u \in \beta_i \cap \beta_j$, then $u \in M_j$ which contradicts $\beta_i \cap M_j = \emptyset$ (Invariant 2).

On the other hand, the case $i = 0$ is similar. We have $\beta_i \cap \beta_j \subseteq B_0 \cap (B_j \cup M_j)$ by Invariant 3. By Invariants 2 and 3 and the fact that $B_0 \cap B_j = \emptyset$ we have $B_0 \cap (B_j \cup M_j) = \emptyset$. □

So, let j be the unique index such that $u \in \beta_j$ at the time the condition in Step (6) becomes true. There is exactly one such j as otherwise we have $u \in \beta_j \cap \beta_{j'} \subseteq (B_j \cup M_j) \cap (B_{j'} \cup M_{j'}) = M_j \cap M_{j'}$ by Invariants 2 and 3 and the fact that $B_j \cap B_{j'} = \emptyset$. But then $u \in M_j \cap \beta_{j'}$, which contradicts Invariant 2. Next, let $J = \{i' \neq j : v \in M_{i'}\}$ and note that J consists of all indices i' (except, perhaps, j) such that $uv \in \delta^{in}(M_{i'})$. By the termination condition, $J \neq \emptyset$.

For each $i' \in J$, let $P_{i'}$ be any $v - h_i$ path using edges in $F'_{i'}$ (one exists by Invariant 1). Now, if $u \in \beta_j - B_j$ then let P_j simply denote the path consisting of the single edge $wu \in F'$ with $w \in B_j$ whose existence is ensured by Invariant 3. If $u \in B_j$, then let P_j be the trivial path consisting of node u and no edges. Let $E(P)$ and $V(P)$ denote the vertices and edges on a path P, respectively.

Construct a partial Steiner tree $\mathcal{U}'$ obtained from $\mathcal{U}$ and Algorithm 2 as follows.

- The sets $B_{j'}, F_{j'}$ and head $h_{j'}$ are unchanged for all $j' \notin J + j$.
- Replace the components $\{B_i\}_{i \in J+j}$ with a component $B := \cup_{i' \in J+j} (B_{i'} \cup V(P_{i'}))$ having head $h := h_j$. The edges of this component in $\mathcal{U}'$ are $F := \cup_{i' \in J+j} (F_{i'} \cup E(P_{i'})) + uv$.
- The free Steiner nodes $\overline{B}'$ of $\mathcal{U}'$ are the Steiner nodes not contained in any component.

Namely, $\overline{B}'$ consists of those nodes in $\overline{B}$ that are not contained on any path $P_{i'}, i' \in J + j$.

4.2.4 Lemma. *When Step (7) is reached, the Steiner tree $\mathcal{U}'$ constructed above satisfies the conditions guaranteed by Lemma 4.1.1.*

Proof. We first verify that $\mathcal{U}'$ as constructed above is indeed a valid partial Steiner tree. Clearly the new sets $\overline{B}', \{B_i\}_{i \notin J+j}$ and B partition V and $\overline{B}'$ is a subset of Steiner nodes.

Note that if $0 \in J+j$ in the above construction, then $j = 0$ because no moat contains r. Thus, if B_0 is replaced when B is constructed, then $h = r$.

Next, consider any $w \in B$. If $w \in B_j$ then there is an $h - v$ path in $F_j \subseteq F$. If $w \in B_{i'}, i' \neq j$ then it can be reached in (B, F) by following the sequence of paths and edges $P_j, uv, P_{i'}$, all of which lie in F, followed by the $h_{i'} - w$ path in $F_{i'}$. Finally, if $w \notin B_{i'}, i' \in J + j$ then w lies on some path $P_{i'}$, in which case it can be reached in a similar way.

It is also clear that $E(\mathcal{U}) \subseteq E(\mathcal{U}')$ and that the number of non-root components in $\mathcal{U}'$ is $\ell - |J| < \ell$. Also, $\text{cost}(E(\mathcal{U}')) - \text{cost}(E(\mathcal{U}))$ is at most the cost of the paths $\{P_{i'}\}_{i' \in J+i}$ plus c_{uv}.

Say the algorithm terminates at time Δ. By Invariant 4, the total value of the feasible solution y is $\ell \cdot \Delta \leq OPT_f$. There are two cases to consider and both of them show the cost increase is bounded by $3\Delta|J|$. This shows the cost increase is at most $3 \cdot OPT_f \cdot \frac{|J|}{\ell}$. Figure 4.2 illustrates these cases.

Case: $u \notin B_j$
Then $u \in M_j$ for the moat M_j at the time of termination (i.e. when Step (7) is reached). Since $M_j \cap B_{j'} \subseteq M_j \cap \beta_{j'} = \emptyset$ for any $j' \neq j$ by Invariants 2 and 3, then $u \in \overline{B}$.

Since G is quasi-bipartite, then $v \in B_{i'}$ for some $i' \in J$. By the termination condition and the fact that $M_i \cap B_{i'} = \emptyset$ for $i \neq i'$ by Invariant 2, we have i' is the only index in J. A simple extension of Invariant 1 shows the cost of the path uv followed by the $v - h_{i'}$ path in F'_i is at most Δ.

Since u is a Steiner node then $P_j = \{wu\}$ for some $wu \in F'_j$ meaning $c_{wu} \leq \Delta$ by Invariant 1. Thus, $\text{cost}(E(\mathcal{U}')) - \text{cost}(E(\mathcal{U})) \leq \text{cost}(P_{i'}) + c_{uv} + c_{wu} \leq 2\Delta \leq 3\Delta|J|$.

Case: $u \in B_j$.
Then $P_j = \emptyset$. The cost of each of the paths $P_{i'}, i' \in J$ is also at most Δ by Invariants 1. Finally, at any stage of the algorithm the edge uv lies in at most $|J| + 1$ cuts of the form $\delta^{in}(M_{i'})$ for some moat $M_{i'}$ (namely, only indices in $J + j$ could have had their moats contributing to the load of uv). Thus, $c_{uv} \leq \Delta(|J| + 1) \leq 2\Delta|J|$. Overall, the cost increase is at most $3\Delta|J|$.

□

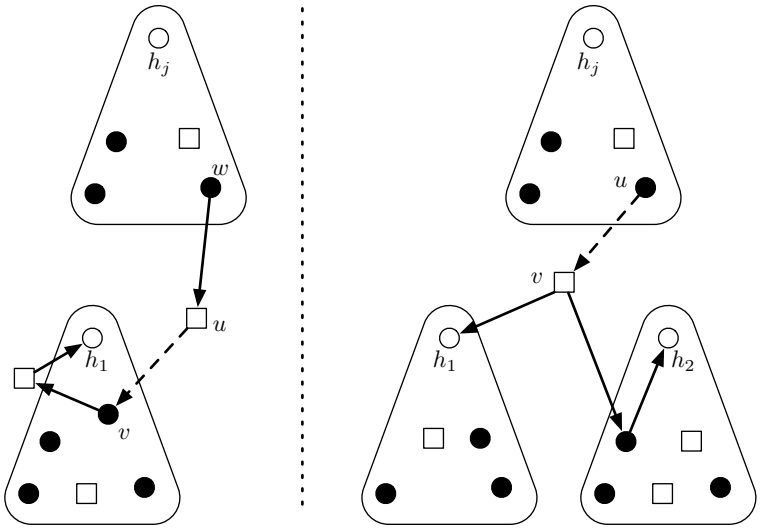

Figure 4.2: The two cases in the proof of Lemma 4.2.4. The left figure illustrates the case $u \notin B_j$ and has $J = \{1\}$. Note that in this case we have $u \in \beta_j \cap M_j$. The right figure illustrates the case $u \in B_j$ and has $J = \{1, 2\}$. In this case, v lies in both moats M_1 and M_2. In both cases, the edge uv is drawn with dashed edges and the paths $P_{i'}, i' \in J + j$ are drawn with solid edges.

Chapter 5

Open Problems

An intriguing possibility for getting a polynomial-time polylogarithmic approximation for Directed Steiner Tree would be to show that a constant-number of rounds of some hierarchy suffice. Alternatively, perhaps the projection of $\mathrm{SA}_\ell(\mathcal{P})$ to the singleton variables $y_{\{e\}}, e \in E$ can be optimized over in time that is polynomial in 2^ℓ and the size of LP (P2). Similar running time improvements have been explored in other work, e.g. [24].

In Chapter 4, we have shown that the integrality gap of LP relaxation (P0) is $O(\log k)$ in quasi-bipartite instances of Directed Steiner Tree. The gap is known to be $\Omega(\sqrt{k})$ in 4-layered instances [25] and $O(\log k)$ in 3-layered instances. Since quasi-bipartite graphs are one way to generalize 2-layered instances, it is natural to ask if there is a generalization of 3-layered instances which has an $O(\log k)$ or even an $o(\sqrt{k})$ integrality gap.

One possible generalization of 3-layered graphs would be when the subgraph of G induced by the Steiner nodes has no node with both positive indegree and outdegree. None of the known results on Directed Steiner Tree suggest such instances have a bad gap.

Bibliography

[1] Arora, Bollobas, Lovasz, and Tourlakis, *Proving integrality gaps without knowing the linear program*, Theory of Computing – An Open Access Journal, http://theoryofcomputing.org, vol. 2, 2006.

[2] Arora, Rao, and Vazirani, *Expander flows, geometric embeddings and graph partitioning*, STOC: ACM Symposium on Theory of Computing (STOC), 2004.

[3] Boaz Barak, Fernando G. S. L. Brandão, Aram Wettroth Harrow, Jonathan A. Kelner, David Steurer, and Yuan Zhou, *Hypercontractivity, sum-of-squares proofs, and their applications*, CoRR **abs/1205.4484** (2012).

[4] MohammadHossein Bateni, Moses Charikar, and Venkatesan Guruswami, *Maxmin allocation via degree lower-bounded arborescences*, STOC (Michael Mitzenmacher, ed.), ACM, 2009, pp. 543–552.

[5] Berman and Ramaiyer, *Improved approximations for the steiner tree problem*, ALGORITHMS: Journal of Algorithms **17** (1994).

[6] Bern and Plassmann, *The steiner problem with edge lengths 1 and 2*, IPL: Information Processing Letters **32** (1989).

[7] Marcus Brazil, Ronald L. Graham, Doreen A. Thomas, and Martin Zachariasen, *On the history of the Euclidean Steiner tree problem*, Arch. Hist. Exact Sci. **68** (2014), no. 3, 327–354. MR 3200931

[8] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità, *An improved LP-based approximation for steiner tree*, STOC (Leonard J. Schulman, ed.), ACM, 2010, pp. 583–592.

[9] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov, and A. Zelikovsky, *On wirelength estimations for row-based placement*, Proceedings of the International Symposium on Physical Design (ISPD-98) (New York), ACM Press, April 6–8 1998, pp. 4–11.

[10] Gruia Calinescu and Alexander Zelikovsky, *The polymatroid steiner problems*, J. Comb. Optim **9** (2005), no. 3, 281–294.

[11] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha, *Approximation algorithms for directed steiner tree problems*, Technical Note CS-TN-97-56, Stanford University, Department of Computer Science, March 1997.

[12] Moses Charikar, Konstantin Makarychev, and Yury Makarychev, *Integrality gaps for sherali-adams relaxations*, STOC (Michael Mitzenmacher, ed.), ACM, 2009, pp. 283–292.

[13] Chlamtac, *Approximation algorithms using hierarchies of semidefinite programming relaxations*, FOCS: IEEE Symposium on Foundations of Computer Science (FOCS), 2007.

[14] Eden Chlamtac and Gyanit Singh, *Improved approximation guarantees through higher levels of SDP hierarchies*, APPROX-RANDOM (Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, eds.), Lecture Notes in Computer Science, vol. 5171, Springer, 2008, pp. 49–62.

[15] Irit Dinur and David Steurer, *Analytical approach to parallel repetition*, CoRR **abs/1305.1979** (2013).

[16] Garg, Konjevod, and Ravi, *A polylogarithmic approximation algorithm for the group steiner tree problem*, ALGORITHMS: Journal of Algorithms **37** (2000).

[17] Konstantinos Georgiou, *Integrality gaps for strong linear programming and semidefinite programming relaxations*, November 2010.

[18] Goemans and Williamson, *A general approximation technique for constrained forest problems*, SICOMP: SIAM Journal on Computing **24** (1995).

[19] Michel X. Goemans, Neil Olver, Thomas Rothvoß, and Rico Zenklusen, *Matroids and integrality gaps for hypergraphic steiner tree relaxations*, CoRR **abs/1111.7280** (2011).

[20] M. GRÖTSCHEL, L. Lovász, and A. SCHRIJVER, *Relaxations of vertex packing*, jco (1986), 330–343.

[21] Guha, Moss, Naor, and Schieber, *Efficient recovery from power outage (extended abstract)*, STOC: ACM Symposium on Theory of Computing (STOC), 1999.

[22] Guitart and Basart, *A high performance approximate algorithm for the steiner problem in graphs*, RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science, LNCS, 1998.

[23] Anupam Gupta, Kunal Talwar, and David Witmer, *Sparsest cut on bounded treewidth graphs: Algorithms and hardness results*, CoRR **abs/1305.1347** (2013).

[24] Venkatesan Guruswami and Ali Kemal Sinop, *Faster SDP hierarchy solvers for local rounding algorithms*, Electronic Colloquium on Computational Complexity (ECCC) **19** (2012), 111.

[25] Halperin, Kortsarz, Krauthgamer, Srinivasan, and Wang, *Integrality ratio for group steiner trees and directed steiner trees*, SICOMP: SIAM Journal on Computing **36** (2007).

[26] Halperin and Krauthgamer, *Polylogarithmic inapproximability*, STOC: ACM Symposium on Theory of Computing (STOC), 2003.

[27] C. S. Helvig, Gabriel Robins, and Er Zelikovsky, *New approximation algorithms for routing with multi-port terminals*, September 19 2000.

[28] Tomoya Hibi and Toshihiro Fujito, *Multi-rooted greedy approximation of directed steiner trees with applications*, WG (Martin Charles Golumbic, Michal Stern, Avivit Levy, and Gila Morgenstern, eds.), Lecture Notes in Computer Science, vol. 7551, Springer, 2012, pp. 215–224.

[29] Hougardy and Promel, *A 1.598 approximation algorithm for the steiner problem in graphs*, SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms), 1999.

[30] Hwang, Richards, and Winter, *The steiner tree problem*, ANNALSDM: Annals of Discrete Mathematics **53** (1992).

[31] Anna R. Karlin, Claire Mathieu, and C. Thach Nguyen, *Integrality gaps of linear and semi-definite programming relaxations for knapsack*, CoRR **abs/1007.1283** (2010).

[32] Marek Karpinski and Alexander Zelikovsky, *New approximation algorithms for the steiner tree problems*, Tech. Report TR-95-036, International Computer Science Institute, Berkeley, CA, August 1995.

[33] Khot, *On the unique games conjecture (short)*, FOCS: IEEE Symposium on Foundations of Computer Science (FOCS), 2005.

[34] Subhash Khot and Rishi Saket, *SDP integrality gaps with local ell_1-embeddability*, FOCS, IEEE Computer Society, 2009, pp. 565–574.

[35] Kou, Markowsky, and Berman, *A fast algorithm for steiner trees*, ACTAINF: Acta Informatica **15** (1981).

[36] Lasserre, *An explicit exact SDP relaxation for nonlinear 0-1 programs*, IPCO: 8th Integer Programming and Combinatorial Optimization Conference, 2001.

[37] Zi-Cheng Liu and Ding-Zhu Du, *On Steiner minimal trees with L_p distance*, Algorithmica **7** (1992), 179–191.

[38] L. Lovász and A. SCHRIJVER, *Cones of matrices and set-functions and 0-1 optimization*, siopt **1** (1991), no. 2, 166–190.

[39] Bing Lu and Lu Ruan, *Polynomial time approximation scheme for the rectilinear steiner arborescence problem*, J. Comb. Optim **4** (2000), no. 3, 357–363.

[40] Avner Magen and Mohammad Moharrami, *Robust algorithms for on minor-free graphs based on the sherali-adams hierarchy*, APPROX-RANDOM (Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, eds.), Lecture Notes in Computer Science, vol. 5687, Springer, 2009, pp. 258–271.

[41] Madhav V. Marathe, R. Ravi, and R. Sundaram, *Improved results on service-constrained network design problems*, Network design: connectivity and facilities location (Princeton, NJ, 1997), DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 40, Amer. Math. Soc., Providence, RI, 1998, pp. 269–276. MR 1613008 (99a:68135)

[42] Claire Mathieu and Alistair Sinclair, *Sherali-adams relaxations of the matching polytope*, STOC (Michael Mitzenmacher, ed.), ACM, 2009, pp. 293–302.

[43] Vardges Melkonian and Éva Tardos, *Primal-dual-based algorithms for a directed network design problem*, INFORMS Journal on Computing **17** (2005), no. 2, 159–174.

[44] Promel and Steger, *RNC-approximation algorithms for the steiner problem*, STACS: Annual Symposium on Theoretical Aspects of Computer Science, 1997.

[45] Prasad Raghavendra and David Steurer, *Integrality gaps for strong SDP relaxations of UNIQUE GAMES*, FOCS, IEEE Computer Society, 2009, pp. 575–585.

[46] Sridhar Rajagopalan and Vijay V. Vazirani, *On the bidirected cut relaxation for the metric steiner tree problem*, Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (N.Y.), ACM-SIAM, January 17–19 1999, pp. 742–751.

[47] Ram Ramanathan, *An algorithm for multicast tree generation in networks with asymmetric links*, INFOCOM, 1996, pp. 337–344.

[48] Reich and Widmayer, *Beyond steiner's problem: A VLSI oriented generalization*, WG: Graph-Theoretic Concepts in Computer Science, International Workshop WG, 1989.

[49] Gabriel Robins and Alexander Zelikovsky, *Improved steiner tree approximation in graphs.*, Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (N.Y.), ACM Press, January 9–11 2000, pp. 770–779.

[50] Thomas Rothvoß, *Directed steiner tree and the lasserre hierarchy*, CoRR **abs/1111.5473** (2011).

[51] Schoenebeck, Trevisan, and Tulsiani, *A linear round lower bound for lovasz-schrijver SDP relaxations of vertex cover*, ECCCTR: Electronic Colloquium on Computational Complexity, technical reports, 2006.

[52] Sherali and Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIJDM: SIAM Journal on Discrete Mathematics **3** (1990).

[53] Shi and Su, *The rectilinear steiner arborescence problem is NP-complete*, SICOMP: SIAM Journal on Computing **35** (2006).

[54] Alexander Zelikovsky, *A series of approximation algorithms for the acyclic directed steiner tree problem*, Algorithmica **18** (1997), no. 1, 99–110.