Algorithms for Virtual Private Network Design

Diploma Thesis
of

Thomas Rothvoß

Advised by
Prof. Dr. Friedrich Eisenbrand

Second reviewer
PD Dr. Piotr Krysta

Fachbereich Informatik
Universität Dortmund

Remark

This document is an unofficial translation of my diploma thesis „Algorithmen für den Entwurf virtueller privater Netzwerke“.

Most of the presented results have been developed during interesting discussions with Fritz Eisenbrand.

Essen, Germany, October 2006

Thomas Rothvoß

Introduction

Network design deals with installing sufficient capacities in a weighted graph, to ensure safe communication. Many interesting network design problems are $\mathcal{NP}$-hard. In this diploma thesis we will consider the $\mathcal{NP}$-hard problem of *Virtual Private Network Design*, which has attracted much attention in recent literature.

VPND is motivated by the fact that upcoming flow or traffic scenarios, that have to be handled, cannot be anticipated. This is the case particularly with regard to the internet, where stand-alone terminals communicate with each other. Because of this, models are developed, in which a flow is feasible if it respects certain bounds at the terminals. The VPND model assigns two values to each terminal, an upper bound for the value of flow that the terminal is able to send and an upper bound for the amount of flow that the terminal is able to receive. It is easy to see, that we can simplify this problem by just allowing two kinds of terminals, called senders and receivers. Senders can send one unit of flow but cannot receive any flow and receivers only can receive one unit of flow. The design of such virtual private networks will be part of this study.

If it is our goal to compute capacities for a VPND problem at minimal costs, we get a generalization of the $\mathcal{NP}$-hard Steiner tree problem. The previously known reduction from Steiner tree to VPND enforces, that the resulting VPND instance has only one sender. Interestingly we can show, that if the number of senders and receivers is nearly balanced, VPND is still $\mathcal{NP}$-hard (section 4.2), while the tree variant of VPND (denoted by $\text{VPND}_{\text{Tree}}$) can be optimized in polynomial time (section 5.2).

In section 5, we present the structure of $\text{VPND}_{\text{Tree}}$ in detail, which will give us a new view of the problem. This will allow us to prove a couple of new results. We can improve the ratio of $1 + \frac{|R|}{|S|}$ for the shortest path tree [5] to $\frac{|R|+|S|}{2|S|}$, compared to the costs of an optimal tree solution, whereupon S and R denote the sets of senders and receivers, respectively, and we assume $|S| \leq |R|$. A simple instance will show that this bound is tight. Afterwards we prove that $\text{VPND}_{\text{Tree}}$ is efficiently solvable in the case $|R| = |S| \pm O(1)$, extending the previously known case of $|R| = |S|$ [10]. If $|R| = O(1) \cdot |S|$, we will still be able to give a PTAS.

The *Rent-or-Buy problem* consists of sending one unit of flow from a set of nodes to a distinguished root, choosing for each edge, whether to pay all-in or per unit of flow. If we can choose the root arbitrarily, the problem is called *Connected Facility Location problem* (CFL). Meanwhile CFL is a classic problem in recent network design literature. Up to now the best approximation algorithm for Connected Facility Location is due to Gupta, Kumar and Roughgarden [8] and consists of choosing each demand with a particular probability into the set of facilities. Together with an innovative analysis, this simple procedure yields an approximation ratio of 3.55. During this work, we will refine this analysis, showing that adjusting the probability leads to an improved ratio

of 3.05. We also present simple worst-case instances that give a lower bound of 2.36 to the approximation ratio of this sampling algorithm. Despite of the different definitions of CFL and VPND we will see that basically CFL accords to the tree variant of VPND. Because of this we can convey the gained approximation ratio of 3.05 from CFL to $\text{VPND}_{\text{Tree}}$.

The task of the *Maybecast problem,* that is a more general form of the Steiner tree problem, is to connect the nodes of a graph to a root within minimal costs. But each node has a probability, that its connection has to be paid. The first constant approximation ratio for the Maybecast problem has been 40.7 and has been proven in [12]. We will give an approximation preserving reduction from Maybecast to CFL, which implies a ratio of 4.83, applying results for CFL that we achieve in section 2.1.

For the variant of VPND that aims to minimize congestion on the edges (denoted by $\text{VPND}_{\text{MinCon}}$) neither good approximation algorithms nor a lower bound on its approximability are known. But we will be able to prove an integrality gap of at least $\Omega(\sqrt{n}/\log n)$ w.r.t. the natural relaxation of this problem, with n being the number of nodes in the regarded graph. This is the first evidence indicating the hardness of approximation for this problem. Additionally we will show that randomized rounding of an optimal, fractional solution can not even give the modest approximation ratio of $o(n)$.

At first we will define all problems that we will talk about in section 1. Chapter 2 deals with the Connected Facility Location problem that will turn out to be related to VPND. We will apply gained results to give a better approximation algorithm to the Maybecast problem in chapter 3. In the following chapters, we will consider three variants of VPND, which are

- The general case → VPND
- The case restricted to tree solutions → $\text{VPND}_{\text{Tree}}$
- The variant that aims for minimizing congestion instead of costs → $\text{VPND}_{\text{MinCon}}$

In the last chapter we go into unanswered questions, that have evolved during this thesis and are worth to be discussed in the future.

Contents

1 Definitions

To clarify, what we are talking about, we start by declaring some problem definitions.

Def. Steiner tree. We are given an undirected graph $G = (V, E)$ with a cost function $c : E \to \mathbb{R}^+$ and a subset of distinguished nodes $R \subseteq V$, that we will call *terminals*. An optimal solution is a tree $T \subseteq E$, spanning all the terminals and minimizing the costs $c(T) = \sum_{e \in T} c(e)$.

One can imagine this problem as connecting a given set of locations R on the map at minimal costs, by building streets between them. There are many applications of this problem. The aspect of particular relevance to us is, that Steiner trees occur as subproblems in other optimization problems. Because of this, the following facts are of interest.

- The Steiner tree problem is $\mathcal{NP}$-hard. Up to now the best known approximation algorithm has a ratio of $1 + \frac{\ln(3)}{2} \leq 1.55$ [16].
- Steiner tree is even APX-complete [1] what implies that there is no PTAS for it unless $\mathcal{P} = \mathcal{NP}$.
- The problem is solvable in polynomial time via dynamic programming, if $|R| = O(\log |V|)$ [3].

Now we present a problem that contains Steiner tree as a subproblem.

Def. Connected Facility Location (CFL). The input consists of an undirected, weighted graph $G = (V, E)$ with cost function $c : E \to \mathbb{R}^+$, a parameter $M \geq 1$ and distinguished nodes $D \subseteq V$, that are called *demands*. A solution is a tuple (F, T), with so called *facilities* $F \subseteq V$ and a tree T, that is spanning F and has costs of

$$M \cdot c(T) + \sum_{d \in D} \ell(d, F),$$

which are to be minimized.

For two nodes $v, u \in V$ we define $\ell(v, u)$ to be the length of the shortest path from v to u (w.r.t. cost function c) in G. Given a node set $U \subseteq V$, we denote $\ell(v, U) := \min_{u \in U}\{\ell(v, u)\}$ to be the shortest distance of v to a node in U.

Since T is spanning F, T is a Steiner tree w.r.t. terminals F. If we are given a facility r, CFL is also called *Rent-or-Buy problem*, because we can interpret the problem, such

that we have to send one unit of flow from each demand in D to the root r via cables. For each cable we have the choice of renting the cable and pay for each unit of flow or buying the cable for M times the renting costs, such that we can send arbitrary amounts across the cable.

Choosing any terminal as root r, we can interpret the Steiner tree problem as finding the cheapest set of edges, connecting all terminals to the root. The *Maybecast problem* [12] generalizes this view, such that the expected connection costs are to be minimized, since there is a certain probability p_i for each terminal i to be *active* and we only have to pay for the connection of active terminals. More precisely the problem is defined as follows.

Def. Maybecast problem. We are given an undirected weighted graph $G = (V, E)$ with costs $c : E \to \mathbb{R}^+$, root $r \in V$ and terminals $R \subseteq V$ with values $p_i \in \mathbb{Q} \cap [0, 1]$ for each $i \in R$, that define the probabilities, that terminals become active. The problem consists of finding paths P_i from i to r for each terminal $i \in R$, such that the expected costs of all active edges are minimized, while all edges on paths of active terminals become active.

Now let us define the *Virtual Private Network Design problem*.

Def. VPND. The input of the VPND problem is an undirected, weighted Graph $G = (V, E)$ with costs $c : E \to \mathbb{R}^+$, two distinguished sets of nodes, the senders $S \subseteq V$ and the receivers $R \subseteq V$, which have to be distinct. A solution is given by paths $\mathcal{P} = \{P_{sr} | s \in S, r \in R, P_{sr} \text{ is } s\text{-}r \text{ path}\}$ from each sender to each receiver and by edge capacities $u : E \to \mathbb{N}_0$, such that u is sufficient, to support all feasible traffic scenarios and the costs

$$\sum_{e \in E} c(e)u(e)$$

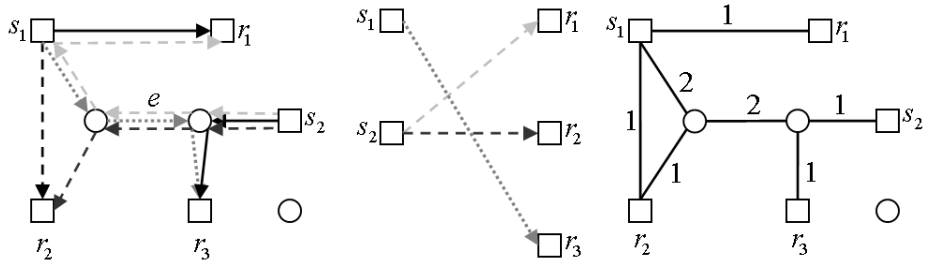
are minimized. A feasible traffic scenario consists of paths $\mathcal{P}' \subseteq \mathcal{P}$, that contain each sender and receiver at most once as starting and ending point, respectively. The capacities $u(e)$ on edge e are sufficient for traffic scenario $\mathcal{P}'$, if $u(e) \geq |\{P \in \mathcal{P}' | e \in P\}|$. This means that the number of paths in $\mathcal{P}'$, containing edge e, may not exceed the capacity on e.

If we switch the set of senders and the set of receivers, the value of the optimal solution remains the same. Because of this, we may assume $|S| \leq |R|$ during this work.

We will call the union of senders and receivers *terminals*.

To give motivation we can imagine that we have to establish a network, connecting computers, divided into senders and receivers, and we have to install enough capacities, such that the computers may communicate with each other. But the pairs of senders and receivers, that communicate, are not known previously. This model is certainly close to reality.

In literature this problem is often denoted as *asymmetric* variant. The *symmetric* variant [7] does not distinguish senders from receivers, such that each terminal can send



(a) chosen paths (b) graph G_e (c) resulting capacities

Figure 1.1: Example for needed capacities, in which s_1, s_2 are senders and r_1, r_2, r_3 are receivers. Terminals (senders or receivers) are depicted as rectangles, circles depict ordinary nodes. (a) shows the chosen paths, (b) depicts the bipartite graph G_e and (c) shows the resulting capacities.

and receive. In this setting an optimal tree solution is computable in polynomial time. This solution is at most twice as expensive as an optimal general solution [7]. The so called *Tree Conjecture* supposes, that there is always an optimal solution for this problem, that forms a tree. Up to now this conjecture has only been proven for ring graphs [9].

The significant part of the definition of VPND is, that we claim that each feasible traffic scenario has to be supported by the installed capacities. We must be able to verify this efficiently, otherwise we would not call this problem to be a well-defined optimization problem[1]. Let us consider an example in figure 1.1 (a). Assume that we decide for the (non-optimal) depicted paths. Let us try to determine the needed amount of capacity on edge e. For this purpose we construct a bipartite graph $G_e = (S \cup R, E_e)$, with senders and receivers forming both partitions. G_e contains an edge (s, r) with $s \in S$ and $r \in R$ if and only if the path from s to r crosses edge e (figure 1.1 (b)). Let $\mathcal{M} \subseteq E_e$ be a matching on G_e, then each sender and receiver is incident to at most one edge in $\mathcal{M}$. This means that $\mathcal{M}$ is in accordance to a valid traffic scenario, in which exactly $|\mathcal{M}|$ paths are crossing e and we know that the capacity on e has to fulfill $u(e) \geq |\mathcal{M}|$. We conclude that we have to install the value of the maximal matching in G_e as capacity on e. Since we can compute maximal matchings efficiently, the needed capacities can be verified in polynomial time.

A (not unique) maximal matching in our example in figure 1.1 consists of edges (s_1, r_3) and (s_2, r_2). This means we have to install 2 units of capacity on e. If we apply this method to the other edges, we obtain the capacities, depicted in figure 1.1 (c). As we can see, the edges, that have strictly positive capacity, contain a circle and hence do not form a tree. In network design, solutions that define a tree are very important. Because of this we define the following tree variant of VPND.

[1] see definition of complexity class $\mathcal{NPO}$ in [19]

Def. $\text{VPND}_{\text{Tree}}$. Input and output are equal to VPND, but the union of all paths must form a tree.

We know that a tree implies unique paths between all pairs of nodes on the tree. This means, that we can use a tree $T \subseteq E$ as a solution for $\text{VPND}_{\text{Tree}}$ instead of paths. In section 5 we will see, that the computation of the needed capacities is very simple, if the solution is a tree.

An easy and intuitive tree solution consists of choosing an arbitrary node $r \in V$ as root and adding edges on shortest paths from root to the terminals to the solution. The resulting tree is called the *shortest path tree* w.r.t. root r. The cheapest of all shortest path trees is of particular importance.

In other articles mostly the following generalization of VPND is defined [4, 5, 7, 8, 10].

Def. General VPND. The input consists of an undirected graph $G = (V, E)$ with costs $c : E \to \mathbb{R}^+$ and values $b^+(v)$ and $b^-(v)$ for each node $v \in V$ that define, how many units v may send and receive, respectively. A solution is defined by paths between each pair of nodes and edge capacities $u : E \to \mathbb{N}_0$ that support each valid traffic scenario, whereupon such a valid traffic scenario is given by a matrix $A : V \times V \to \mathbb{N}_0$ with $\sum_{u \in V, u \neq v} A(u, v) \leq b^-(v)$ and $\sum_{u \in V, u \neq v} A(v, u) \leq b^+(v)$.

We can easily reduce this more general problem to VPND, by replacing each node $v \in V$ by $b^+(v)$ many senders and $b^-(v)$ receivers. Even if $b^+(v)$ or $b^-(v)$ are not bounded polynomially, all presented algorithms can be implemented, such that running time remains polynomial.

Many flow problems possess variants, that deal with minimizing congestion, instead of costs [15]. So we want to examine such a variant for VPND, too.

Def. $\text{VPND}_{\text{MinCon}}$. We are given an undirected graph $G = (V, E)$ with senders $S \subseteq V$ and receivers $R \subseteq V$ (fulfilling $S \cap R = \emptyset$). A solution consists of paths $\mathcal{P} = \{P_{sr} | s \in S, r \in R, P_{sr} \text{ is } s\text{-}r \text{ path}\}$ from each sender to each receiver and edge capacities $u : E \to \mathbb{N}_0$ that support each feasible traffic scenario. The goal is to minimize $\max\{u(e) | e \in E\}$.

2 Connected Facility Location

Recent results and our contributions

The goal of the Connected Facility Location problem is to compute a set of facilities $F \subseteq V$ and a tree T spanning F in a weighted graph $G = (V, E)$, minimizing $c(T) + \sum_{d \in D} \ell(v, F)$. The best known approximation ratio was improved over time from 9.002 [7] via 4.55 [17] to 3.55 [8]. In this section we will refine the algorithm and the analysis in [8] to obtain a ratio of 3.05.

If we are given a facility $r \in V$, we can interpret CFL to be the problem of sending one unit of flow from each demand to the root r. Let k be the amount of flow crossing an edge e, then this causes costs of $k \cdot c(e)$, if $k \leq M$. Otherwise if $k > M$, it is cheaper to add e to T, so that we have to pay $M \cdot c(e)$ for this edge. Another point of view is, that we may decide, whether we want to rent an edge, such that we have to pay $c(e)$ per unit of flow on the edge or we can buy it for $M \cdot c(e)$ and send an arbitrary amount of flow across the edge. Because of this view, that variant is called *Rent-or-Buy problem*. The costs, depending on the flow across an edge are piecewise linear with 2 different slopes. So the Rent-or-Buy problem is a special case of the *Single-Sink-Buy-At-Bulk problem* [8].

There is a generalization of CFL that contains additional costs f_i to open a facility $i \in F$ and restricts the possible facilities to a subset of the nodes. Up to now the best known approximation algorithm for this problem has a ratio of 8.55 [17].

2.1 The sampling algorithm

CFL Algorithm 1 was presented in [8] and chooses each demand with a probability of $1/M$ (i.e., $q = 1$) to be a facility. We assume, that we know a node r^*, that is a facility in an optimal solution. We remember that

$$M \cdot c(T) + \sum_{d \in D} \ell(d, F)$$

defines the costs of a solution (F, T). We call $c(T)$ *Steiner tree costs* and $\sum_{d \in D} \ell(d, F)$ *connection costs*. (F^*, T^*) denotes an arbitrary optimal solution, with objective function value of $OPT = M \cdot S^* + C^*$, whereupon $S^* = c(T^*)$ and $C^* = \sum_{d \in D} \ell(d, F^*)$ denote the Steiner tree costs and connection costs in the optimal solution, respectively. ρ denotes the approximation ratio of the Steiner tree problem. The best known bound on ρ is $\rho \leq 1.55$ due to [16].

Algorithm 1 Sampling algorithm [8]

Input: Weighted graph $G = (V, E)$; demands $D \subseteq V$; parameter $M \geq 1$

Output: Facilities F; tree T

1. Guess a facility $r^* \in F^*$
2. Mark each demand $d \in D$ with probability q/M
3. Set $F := \{\text{marked demands}\} \cup \{r^*\}$
4. Compute ρ-approximative Steiner tree T on terminals F
5. Output (F, T)

The analysis of Gupta et al.

In [8] the authors bound the costs of an optimal Steiner tree, spanning F by OPT, if sampling probability $1/M$ is used. By ρ-approximation one obtains $E[c(T)] \leq \rho \cdot OPT$. Using an innovative analysis the authors show that the connection costs are at most M times the costs of the Steiner tree spanning F, if this tree is constructed using the MST-heuristic. Hence the connection costs can be bounded by $2 \cdot OPT$. In total Gupta, Kumar and Roughgarden [8] were able to bound the approximation ratio by $\rho + 2 \leq 3.55$.

Our contribution consists of proving an upper bound of 3.05 to the expected approximation ratio, choosing $q = 0.67$. To be able to compare our bound to results in [8], at first we will repeat the analysis of Gupta et al. for an arbitrary sampling probability of q/M. We will adopt the analysis of the Steiner tree costs for our analysis, but we will give a completely new proof that bounds the connection costs.

Lemma 2.1. *[8] The Steiner tree costs can be bounded by*

$$E[c(T)] \leq \rho(S^* + \frac{q}{M} C^*)$$

Proof. We show that a tree T', spanning F, exists with expected costs of $E[c(T')] \leq S^* + \frac{q}{M} C^*$. By ρ-approximation the claim then follows. Let E_d be the edges on a shortest path[1] from $d \in D$ to F^*. Apply $T' := T^* \cup \bigcup_{d \in F} E_d$, then T' spans facilities F and has expected costs of

$$E[c(T')] \leq c(T^*) + \sum_{d \in D} \frac{q}{M} c(E_d) = S^* + \frac{q}{M} \sum_{d \in D} \ell(d, F^*) = S^* + \frac{q}{M} C^*$$

□

We proceed with the analysis of the connection costs in [8].

[1] If a shortest path is ambiguous we choose an arbitrary shortest path.

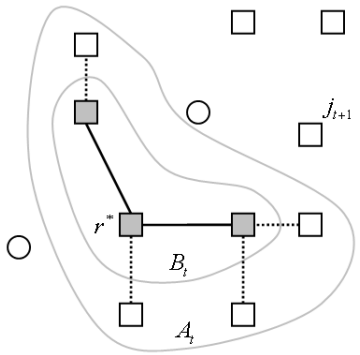


Figure 2.1: Schematic situation of connection cost analysis in [8]. Demands are squares, demands sampled into F are filled gray. Solid edges belong to the MST Steiner tree, dotted lines denote connections from demands to F.

Theorem 2.2. *[8] The connection costs can be bounded by*

$$E[\sum_{d \in D} \ell(d, F)] \leq \frac{2}{q} M S^* + 2C^*$$

Proof. A simple approximation algorithm for Steiner tree consists of computing a minimum spanning tree in the metric closure of graph G. Such a solution gives a 2-approximation [18]. Now let us consider the approximative Steiner tree T on F that has been constructed in this way. We have already shown that a Steiner tree on F exists, with expected costs of at most $S^* + \frac{q}{M} C^*$. So our MST Steiner tree T has costs of at most $2S^* + 2\frac{q}{M} C^*$.

The idea of the analysis in [8] is to compare the connection costs with the costs of the MST-heuristic tree. Let A_t be the set of demands, that has been considered by the algorithm until step t and let $B_t \subseteq A_t$ be the set of considered demands, that have been sampled into F until step t. We start with $A_1 = B_1 = \{r^*\}$. In step t we regard demand $j_{t+1} \in D \backslash A_t$, that is next to B_t. With probability q/M we choose j_{t+1} to be in F and we set $B_{t+1} = B_t \cup \{j_{t+1}\}$ (and $A_{t+1} = A_t \cup \{j_{t+1}\}$). This decision causes *Steiner tree costs* of $\ell(j_{t+1}, B_t)$, because the MST-heuristic would choose now this path from B_t to j_{t+1}. Because we do so with probability q/M, this raises the *expected* Steiner tree costs by $\frac{q}{M} \ell(j_{t+1}, B_t)$.

In the other case, we do not sample j_{t+1} into F and set $B_{t+1} = B_t$ and $A_{t+1} = A_t \cup \{j_{t+1}\}$. This decision causes *connection costs* of $\ell(j_{t+1}, B_t)$. Because this event occurs with probability $1 - \frac{q}{M}$ the expected connection costs for demand j_{t+1} are at most $(1 - \frac{q}{M}) \ell(j_{t+1}, B_t) \leq \ell(j_{t+1}, B_t)$. This situation is depicted in figure 2.1.

By linearity of expectation one obtains

$$\frac{\text{expected connection costs}}{\text{expected costs of MST Steiner tree}} \leq \frac{\sum_t \ell(j_{t+1}, B_t)}{\sum_t \frac{q}{M} \cdot \ell(j_{t+1}, B_t)} = \frac{M}{q}$$

Now one can bound the expected connection costs by

$$\frac{M}{q} \cdot \text{expected costs of MST Steiner tree} \leq \frac{M}{q}(2S^* + 2\frac{q}{M}C^*) = \frac{2}{q}MS^* + 2C^*$$

□

If we sum up Steiner tree and connection costs we obtain total costs of

$$M \underbrace{\rho(S^* + \frac{q}{M}C^*)}_{\text{Steiner tree costs}} + \underbrace{\frac{2}{q}MS^* + 2C^*}_{\text{connection costs}} = \underbrace{(\rho + \frac{2}{q})}_{\searrow \text{with } q} MS^* + \underbrace{(2 + q\rho)}_{\nearrow \text{with } q} C^*$$

for solution (F, T). Because the first coefficient $\rho + \frac{2}{q}$ decreases with q and the second one $2 + q\rho$ increases with q, we determine a solution of $\rho + \frac{2}{q} = 2 + q\rho$, which is $q = 1$. This implies costs of

$$(\rho + \frac{2}{1})MS^* + (2 + 1\rho)C^* = (2 + \rho)OPT$$

and yields, according to [8], an approximation ratio of $2 + \rho \leq 3.55$. Another choice of q would not improve the bound of this analysis. But now we show a tighter bound on the connection costs, which results in an improved bound on the approximation ratio.

Our contribution

Now we present our new method to upper bound the connection costs. The point is the following

Theorem 2.3. *The connection costs can be bounded by*

$$E[\sum_{d \in D} \ell(d, F)] \leq \frac{1}{q}MS^* + 2C^*$$

Proof. To prove this theorem, we connect demands in a particular way to sampled facilities in F. The expected value of the costs of these connections will satisfy the bound of the theorem. Since the algorithm connects each demand to its closest facility, the theorem then follows.

The *connection node* of a demand $d \in D$ is a facility from the optimal set of facilities F^*, to which d is assigned to in the optimal solution. For demands $d \in F^*$ d is the connection node of itself. The shortest path from a demand $d \in D$ to its connection node is called the *connection path* of d.

There exists a cycle $\mathcal{C}$ in the metric closure of G, whose nodes are the connection nodes of the demands and whose cost is bounded by $2 \cdot c(T^*)$ (see figure 2.2). This cycle can in fact be obtained by doubling the edges of T^* and by considering the Euler tour of the thereby obtained graph, in which non-connection nodes and already visited connection nodes are shortcut.

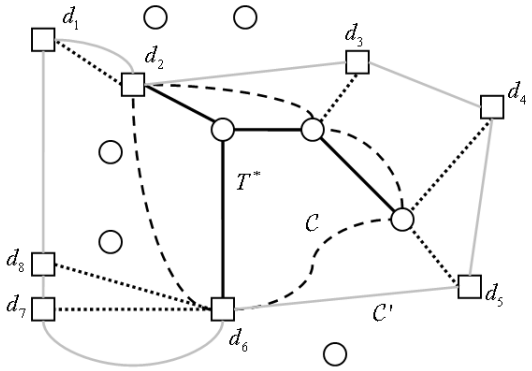


Figure 2.2: Example instance for CFL with depicted optimal solution. Squares are demands, circles represent ordinary nodes. Solid edges are elements of T^* and dotted lines are connection paths. Dashed lines are edges of $\mathcal{C}$, grey ones belong to $\mathcal{C}'$.

Now let $d_1, ..., d_{|D|}$ be an ordering of the demands, which is in accordance to the order in which the connection nodes of the demands are visited on the cycle $\mathcal{C}$. More precisely the ordering has to satisfy the following condition: If $i < j$ then the connection node of d_i precedes the connection node of d_j on a clockwise walk on the cycle $\mathcal{C}$, starting at the connection node of d_1. We have to pay the connection of each demand. We do this by charging the accounts of connection paths and edges in $\mathcal{C}$ by one each time, we have to cross such edges for a connection.
The connection scheme which we use is now the following.

> Connect d_i to the first demand from the following list, that has been sampled into F
>
> $$d_i, d_{i+1}, d_{i-1}, d_{i+2}, d_{i-2}, d_{i+3}, ...$$

We now analyze the expected cost that is caused by this connection scheme. To simplify notation, let us imagine that the sequence $d_1, ..., d_{|D|}$ is continued periodically, i.e., $d_{|D|+i} = d_i$.

First we inspect, by how much each connection path is charged. Each demand $d \in D$ charges at most 2 of such paths, namely the connection path of d and the connection path of the facility in F, to which d is assigned. By symmetry each connection path is charged with the same value in expectation. Since the number of demands equals the number of connection paths, in expectation each connection path is charged at most twice.

Now we want to estimate, by how much each edge of the cycle $\mathcal{C}$ is charged by this connection scheme. Recall that edges of $\mathcal{C}$ correspond to shortest paths between the connection nodes.

For this we consider the cycle $\mathcal{C}'$, which is defined by the sequence $d_1, d_2, \ldots d_{|D|}, d_1$, and distribute charge on edges of $\mathcal{C}'$, if the demands are connected via the suggested connection scheme. For each edge in $\mathcal{C}$ there is one edge in $\mathcal{C}'$, that is charged by the same amount. Thus by upper bounding charge on $\mathcal{C}'$ we get an upper bound on the charge on $\mathcal{C}$.

How many edges of $\mathcal{C}'$ do we have to cross, to connect a demand $d_i \in D$ with the above scheme? Let X_i be the random variable, which corresponds to this number. This number is larger than $k \in \mathbb{N}$, if none of the $2k+1$ demands $d_{i-k}, ..., d_{i+k}$ has been chosen into F. Clearly

$$\Pr(X_i > k) = \Pr(d_{i-k} \notin F, ..., d_{i+k} \notin F) = \prod_{j=i-k}^{i+k} \Pr(d_j \notin F) \leq (1 - \frac{q}{M})^{2k+1}$$

since we choose each demand independently with probability q/M. This probability maybe strict less than $(1 - \frac{q}{M})^{2k+1}$, if r^* is among $d_{i-k}, ..., d_{i+k}$.

It follows that

$$\begin{aligned}
E[X_i] &\overset{(1)}{=} \sum_{k \geq 1} \Pr(X \geq k) \\
&\leq \sum_{k \geq 1} (1 - \frac{q}{M})^{2k-1} \\
&= (1 - \frac{q}{M})^{-1} \sum_{k \geq 1} ((1 - \frac{q}{M})^2)^k \\
&\overset{(2)}{=} (1 - \frac{q}{M})^{-1} \frac{(1 - \frac{q}{M})^2}{1 - (1 - \frac{q}{M})^2} \\
&= \frac{1 - \frac{q}{M}}{(1 - (1 - \frac{q}{M})) \cdot (1 + (1 - \frac{q}{M}))} \\
&= \frac{1 - \frac{q}{M}}{\frac{q}{M} \cdot (2 - \frac{q}{M}))} \\
&\leq \frac{M}{q} \cdot \frac{1 - \frac{q}{2M}}{2(1 - \frac{q}{2M}))} \\
&= \frac{M}{2q}
\end{aligned}$$

Because of $X_i \in \mathbb{N}_0$ equality (1) holds (see [14], p. 31), (2) resolves the geometric series. Thus in expectation, the connection scheme uses at most $\frac{M}{2q}$ edges of $\mathcal{C}'$ to reach the first demand, that has been sampled into F. The number of edges in $\mathcal{C}'$ is $|D|$. Because of symmetry each edge of $\mathcal{C}'$ is charged by at most $\frac{M}{2q}$ in expectation.

By summing the costs distributed on $\mathcal{C}$ and on the connection paths, we can upper bound connection costs by

$$\sum_{d \in D} 2\ell(d, F^*) + \frac{M}{2q} c(\mathcal{C}) \leq 2C^* + \frac{M}{q} S^*$$

because of $c(\mathcal{C}) \leq 2 \cdot S^*$. □

By adding Steiner tree costs from lemma 2.1 and connection costs from theorem 2.3, we obtain the following

Theorem 2.4. *For $q = 0.67$ algorithm 1 is a randomized 3.05-approximation algorithm for Connected Facility Location.*

Proof. Lemma 2.1 and theorem 2.3 reveal, that we can bound expected costs of the solution, given by algorithm 1, by

$$M \underbrace{\rho(S^* + \frac{q}{M} C^*)}_{\text{Steiner tree costs}} + \underbrace{2C^* + \frac{M}{q} S^*}_{\text{connection costs}} = \underbrace{(\rho + \frac{1}{2q})}_{\searrow \text{with } q} MS^* + \underbrace{(2 + q\rho)}_{\nearrow \text{with } q} C^* \tag{2.1}$$

Plugging in $q = 0.67$ yields a value of

$$\leq 3.05 \cdot MS^* + 3.05 \cdot C^* = 3.05 \cdot OPT$$

for equality 2.1 and thus the claim follows. □

2.2 Worst-case instances

In this section we want to engage the question, whether the upper bound, gained in the last section, is tight. For this purpose we present two worst-case instances for the sampling algorithm. To deal with varying sampling probabilities, the first instance is designed to lead to an expensive solution, if q/M is large and the sampling algorithm behaves unfavorable on the second one, if q/M is small. In both cases we consider the limit of the obtained approximation ratio if the number of demands n goes to ∞ and if we define $M := n^{1/3}$. This choice implies that the expected number of facilities, chosen by the sampling algorithm is unbounded, too.

Let G_1 be the graph, which is depicted in figure 2.3 and consists of a comb graph and a star with center d_n. Edges (d_i, v_i) have cost 1, the other ones are for free. Now we will examine why the sampling algorithm has difficulties with this instance.

Lemma 2.5. *The expected approximation ratio of the sampling algorithm on graph G_1 is at least $(q + 2) \cdot (1 - o(1))$ for $n \to \infty$ if we define $M := n^{1/3}$.*

Proof. An optimal solution for this instance is given by choosing facilities $F^* = \{v_1, ..., v_n, d_n\}$ and a Steiner tree $T^* = \{(v_i, v_{i+1}) | i \in \{1, ..., n-1\}\} \cup \{(d_n, v_n)\}$ with costs $c(T^*) = 1$. We have to pay connection costs of $n - 1$, thus if we denote OPT_1 to be the value of an optimal solution, we have $OPT_1 \leq n + M \cdot 1$. Because of $M/n \to 0$ this value basically consists of the connection costs.

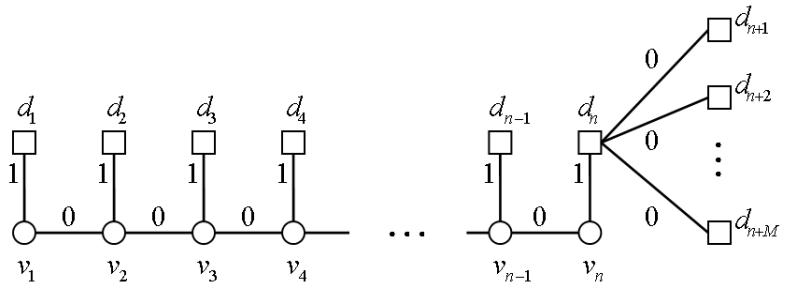


Figure 2.3: Instance G_1 which leads to an expensive solution, if q/M is large. Edges are labeled with their costs.

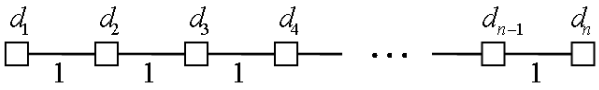


Figure 2.4: Instance G_2 is the metric closure of the depicted graph and leads to an expensive solution, if q/M is small. Edges are labeled with their costs.

Let (F, T) be the solution, computed by the sampling algorithm. The algorithm assumes to know a facility in the optimal solution. Let d_n be this facility. Without the existence of demands $d_{n+1}, ..., d_{n+M}$ we would not have been able to make this choice. Now these demands have served their purpose and we concentrate on the costs that the other demands $D' := \{d_1, ..., d_n\}$ cause.

Because of $n/M \to \infty$, the probability of $|F \cap D'| \geq 2$ converges to 1, so we can assume $|F \cap D'| \geq 2$ w.l.o.g.. Then the computed Steiner tree T contains at least $|F \cap D'|$ edges with cost 1. Each demand $d \in D' \backslash F$ causes connection costs of 2. Thus the expected cost of the solution (F, T) is at least

$$E[|F \cap D'|] \cdot M + E[|D' \backslash F|] \cdot 2 \geq \frac{q}{M} n \cdot M + (1 - \frac{q}{M}) n \cdot 2 = qn + (1 - \frac{q}{M}) n \cdot 2$$

The reached ratio of solution (F, T) is then at least

$$\frac{qn + (1 - \frac{q}{M}) n \cdot 2}{OPT_1} \geq \frac{qn + (1 - \frac{q}{M}) n \cdot 2}{n + M} = \frac{q + 2 \cdot (1 - \frac{q}{n^{1/3}})}{1 + n^{-2/3}} = (q + 2) \cdot (1 - o(1))$$

whereupon the last equality follows by considering $n \to \infty$ and thus the claim follows. □

Now we define graph G_2 as metric closure of the path graph on nodes $V = D = \{d_1, ..., d_n\}$, which is depicted in figure 2.4. Each edge (d_i, d_{i+1}) has cost of 1. Thus, G_2 is a complete graph on D with edge costs $c(d_i, d_j) = |i - j|$.

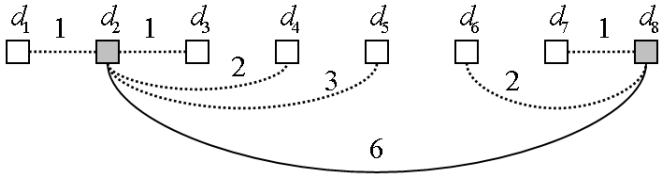


Figure 2.5: Resulting solution of the sampling algorithm on graph G_2 for $n = 8$, if $F = \{d_2, d_8\}$ is chosen. Solid edges belong to the Steiner tree T spanning F, dotted lines are connections from demands to F. Edges are labeled with their costs.

Lemma 2.6. *The expected approximation ratio of the sampling algorithm on graph G_2 is at least $(1 + \frac{1}{2q}) \cdot (1 - o(1))$ for $n \to \infty$ if we define $M := n^{1/3}$.*

Proof. Consider the following solution. Choose $F^* = \{d_1, ..., d_n\}$ as facilities and $T^* = \{(d_i, d_{i+1}) | i = 1, ..., n-1\}$ spanning F^*. This solution costs $M \cdot (n-1)$, which implies $OPT_2 \leq Mn$, if we define OPT_2 to be the value of an optimal solution on graph G_2.

Now let us regard solution (F, T), which is returned by the sampling algorithm. Figure 2.5 will let us better understand the behavior of the algorithm on graph G_2. To give a lower bound we have to assume, that the computed Steiner tree on F is optimal. Then we know $c(T) \geq \max\{i - j | d_i, d_j \in F\}$. The probability that neither one of the first k demands $d_1, ..., d_k$, nor one of the last k demands $d_{n-k+1}, ..., d_n$ is chosen to be a facility, is exponentially small in k. Thus we have $E[c(T)] \geq E[\max\{i-j | d_i, d_j \in F\}] = n \cdot (1 - o(1))$. This means that we have to pay a Steiner tree nearly ranging across the whole path graph.

Now we will show that nearly all of the demands have expected connection costs of approximately $\frac{M}{2q}$. For this purpose we define the following random variable that denotes the costs to connect an arbitrary demand $d_i \in D$.

$$X_i = \min\{|i - j| \, | d_j \in F\} = \text{cost to connect } d_i$$

The sampling algorithm assumes to know a node, which is facility in the optimal solution. We denote this node by $d_{i^*} = r^* \in F^*$. We define a set of demands that are far away from d_{i^*}

$$D' := \left\{ d_i \in D | \, |i - i^*| > n^{2/3} \right\}$$

Because we consider a limit process for $n \to \infty$, we can restrict our claim to values of n such that $n^{2/3}$ is integral. We will assume this from now on. If $\{d_{i-k}, ..., d_{i+k}\} \cap F = \emptyset$ we know that $X_i > k$. For $d_i \in D'$ and $k \leq n^{2/3}$ we know $\Pr(X_i \geq k) \geq (1 - \frac{q}{M})^{2k-1}$. Assuming $d_i \in D'$ we can conclude similar to the analysis in the proof of theorem 2.3

that

$$\begin{aligned}
E[X_i] &= \sum_{k=1}^{\infty} \Pr(X_i \geq k) \\
&\geq \sum_{k=1}^{n^{2/3}} \Pr(X_i \geq k) \\
&\geq \sum_{k=1}^{n^{2/3}} (1 - \frac{q}{M})^{2k-1} \\
&\geq \sum_{k=1}^{n^{2/3}} ((1 - \frac{q}{M})^2)^k \\
&\overset{(1)}{\geq} \frac{\left(1 - \frac{q}{n^{1/3}}\right)^2 - \left(\left(1 - \frac{q}{n^{1/3}}\right)^2\right)^{n^{2/3}+1}}{1 - (1 - \frac{q}{M})^2} \\
&\overset{(2)}{=} \frac{(1 - o(1)) - o(1)}{(1 - (1 - \frac{q}{M})) \cdot (1 + (1 - \frac{q}{M}))} \\
&= \frac{1 - o(1)}{\frac{q}{M} \cdot \left(2 - \frac{q}{n^{1/3}}\right)} \\
&= \frac{M}{2q}(1 - o(1))
\end{aligned}$$

In (1) we resolve the geometric series using

$$\sum_{i=1}^{m} p^i = \frac{p - p^{m+1}}{1 - p}$$

for $-1 < p < 1$ and plug in $M = n^{1/3}$. (2) holds because of $(1 - \frac{q}{n^{1/3}})^2 = 1 - o(1)$ and

$$\left(\left(1 - \frac{q}{n^{1/3}}\right)^2\right)^{n^{2/3}+1} \leq \left(\left(1 - \frac{q}{n^{1/3}}\right)^{n^{1/3}}\right)^{2n^{1/3}} \leq e^{-q \cdot 2n^{1/3}} \to 0$$

for $n \to \infty$. Because of $|D'| = n \cdot (1 - o(1))$ we obtain expected total connection costs of at least

$$n(1 - o(1)) \cdot \frac{M}{2q}(1 - o(1)) = n\frac{M}{2q}(1 - o(1))$$

to connect all demands. If we divide the cost of solution (F, T) by the upper bound on OPT_2, we get a lower bound of

$$\frac{Mn(1 - o(1)) + n\frac{M}{2q}(1 - o(1))}{Mn} = (1 + \frac{1}{2q})(1 - o(1))$$

on the approximation ratio. □

Corollary 2.7. *The sampling algorithm with selection probability q/M has an asymptotical approximation ratio of at least*

$$\max\left\{2+q, 1+\frac{1}{2q}\right\}$$

for $n \to \infty$ choosing $M := n^{1/3}$.

Proof. Add lemmata 2.5 and 2.6. □

Remark. In section 2.1 we have proven, that $\max\{2+\rho q, \rho+\frac{1}{q}\}$ is an upper bound on the approximation ratio of the sampling algorithm. One reason for the gap between lower and upper bound is due to the fact, that we have to use a Steiner tree algorithm as a black-box. The other reason lies in the estimation of the connection costs. Our analysis of these, charges each edge in T^* by $\frac{M}{q}$, while we do not find a worst-case instance in which more than $\frac{M}{2q}$ of charge is needed. It is a conjecture of the author, that this upper bound is not tight and that connection costs can be bounded by

$$E[\sum_{d\in D} \ell(d,F)] \stackrel{?}{\leq} \frac{1}{2q} MS^* + 2C^*$$

Corollary 2.8. *The original algorithm in [8] using $q = 1$ has an approximation ratio of at least* 3*.*

Proof. Plugging $q = 1$ into corollary 2.7 yields a ratio of at least $\max\{2+1, 1+\frac{1}{2}\} = 3$. □

Corollary 2.9. *The sampling algorithm with sampling probability q/M has an approximation ratio of at least* 2.36*, for each constant q.*

Proof. Because $2+q$ increases with q while $1+\frac{1}{2q}$ is decreasing, the best choice for q is the solution of $q+2 = 1+\frac{1}{2q}$ which is $q = \frac{\sqrt{3}-1}{2} \geq 0.36$. Then $\max\{2+q, 1+\frac{1}{2q}\} \geq 2.36$ implies the claim. □

3 The Maybecast problem

Recent results and our contributions

The Maybecast problem demands to find paths from each terminal $i \in R \subseteq V$ in a weighted graph $G = (V, E)$ to the root $r \in V$. Each terminal i becomes active with probability p_i and then activates all the edges on its path to r. The expected costs of active edges are to be minimized.

Clearly this problem is $\mathcal{NP}$-hard, because for $p_i = 1$ the Maybecast problem is equal to the Steiner tree problem. The Maybecast problem was first examined in [12], with the result of a factor 40.7 approximation algorithm. In this thesis we will improve this result by presenting an approximation algorithm with ratio 4.83.

3.1 A new cost function

Let U_e denote the set of terminals that send flow across edge $e \in E$ and let k_e denote the probability that edge e is active. Therefore we know

$$k_e = 1 - \prod_{i \in U_e} (1 - p_i)$$

This means we have to pay $k_e c(e)$ in expectation for edge e. We reason, that the expected costs increase with p_i, but the ratio of expected costs to the sum over all p_i's decreases. Thus the cost function k_e is concave. Karger and Minkoff [12] have proven that a concave cost function implies, that there is always an optimal solution, which forms a tree. From now on we demand the solution of the Maybecast problem to be a tree.

An important simplification relies on replacing k_e by a more simple function $\hat{k}_e$, which is piecewise linear w.r.t. the probabilities p_i and differs from k_e only by a constant factor. This is ensured by the following lemma, which is due to Karger and Minkoff.

Lemma 3.1. *[12] Let* $\hat{k}_e = \min\left\{\sum_{i \in U_e} p_i, 1\right\}$. *Then we have*

$$k_e \leq \hat{k}_e \leq \frac{1}{1 - 1/e} k_e \tag{3.1}$$

for arbitrary probabilities p_i *and sets* U_e.

Proof. We start proving the first inequality. The union bound yields

$$k_e = \Pr(\bigcup_{i \in U_e} i \text{ is active}) \leq \sum_{i \in U_e} \Pr(i \text{ is active}) = \sum_{i \in U_e} p_i$$

Since probabilities are bounded by 1, we have $k_e \leq \min\{\sum_{i \in U_e} p_i, 1\}$. Now we show inequality

$$\min\left\{\sum_{i \in U_e} p_i, 1\right\} \leq \frac{1}{1-1/e}\left(1 - \prod_{i \in U_e}(1-p_i)\right)$$

Let us fix $s := \sum_{i \in U_e} p_i$. The inequality $1 - x \leq e^{-x}$ implies

$$\prod_{i \in U_e}(1-p_i) \leq \prod_{i \in U_e} e^{-p_i} = e^{-s}$$

So we have to verify

$$\min\{s, 1\} \leq \frac{1}{1-1/e}(1-e^{-s})$$

In the case $s \geq 1$ we have $\frac{1}{1-1/e}(1-e^{-s}) \geq \frac{1}{1-1/e}(1-e^{-1}) = 1$. Otherwise if $0 \leq s < 1$ we have to show that

$$s \leq \frac{1}{1-1/e}(1-e^{-s}) \Leftrightarrow \frac{s}{1-e^{-s}} \geq \frac{1}{1-1/e}$$

But $\frac{s}{1-e^{-s}}$ is monotonically increasing for $s \in (0, 1]$, thus we reach a maximum for $s = 1$ and the claim follows. □

This lemma yields that we may assume $\hat{k}_e$ as cost function, if we accept that costs of the resulting solution may increase by a factor of at most $\frac{1}{1-1/e} \leq 1.59$.

3.2 Reduction to Connected Facility Location

After introduction of the simplified cost function $\hat{k}_e$ in [12] the Maybecast problem is reduced to the so called *Gathering problem*. This problem consists of choosing some nodes to be hubs, such that the costs to connect the demands to the next hub are minimized while enough terminals have to be assigned to each hub. More precisely the weight w.r.t. p_i of the terminals, assigned to a hub, must be at least $1/2$. Afterwards the Gathering problem is reduced to the *Facility Location problem*, for which a solution consists of a set of facilities like CFL, but we have to pay a fix value for each facility instead of connecting the set of facilities. Because of these reductions the approximation ratio accumulates to a value of 40.7. After introduction of cost function $\hat{k}_e$, we will use another strategy, such that we do not need more than one reduction, which leads to an approximation ratio of 4.83.

Our goal is to show that the Maybecast problem equals the Rent-or-Buy problem, if we use cost function $\hat{k}_e$. Let us recall, that we assumed $p_i \in \mathbb{Q}$. If $M \in \mathbb{N}$ is the common denominator of all p_i's, then we can denote $p_i = n_i/M$ for some $n_i \in \{1, ..., M\}$. For simplicity we replace each terminal $i \in R$ by n_i terminals with activation probability of $1/M$ for each one. This does not change the value of $\hat{k}_e$[1]. Even if $\sum_{i \in R} n_i$ is not

[1]But the value k_e would have changed by this replacement.

polynomially bounded, this can be implemented efficiently. The value of a tree solution $T \subseteq E$ is therefor

$$\sum_{e \in T} c(e) \cdot \min \left\{ \frac{1}{M} \cdot \text{number of terminals in subtree below } e, 1 \right\} \tag{3.2}$$

which is to be minimized. Scaling this objective function by factor M does not change the optimal solution and we obtain

$$\sum_{e \in T} c(e) \cdot \min \left\{ \text{number of terminals in subtree below } e, M \right\}$$

matching the cost function of the Rent-or-Buy problem, using parameter M. Now we have to show, that our approximation algorithm for CFL can be applied to Rent-or-Buy.

Lemma 3.2. *There is a randomized* 3.05*-approximation algorithm for the Rent-or-Buy problem.*

Proof. We apply the sampling algorithm (algorithm 1) to the Rent-or-Buy problem, using root r to be the facility, that is chosen with probability 1. The analysis in section 2.1 can be adopted without any modification. We only have to show, that the Rent-or-Buy problem matches CFL, provided $r \in F^*$. Let (F^*, T^*) be an optimal CFL solution, and let T' denote the union of edges in T^* and edges on connection paths. It is possible to choose connection paths, such that T' forms a tree. Consider an arbitrary edge $e \in T'$. By removing e out of T' the tree resolves into two connected components. Let k be the number of terminals in the component *not* containing r. Edge e causes cost of $c(e) \cdot \min\{k, M\}$ in the Rent-or-Buy setting. Now let us regard the caused costs in CFL. In the case $k < M$, in an optimal solution we have $e \notin T^*$, then we have to pay $c(e) \cdot k$ for k connections crossing e. Otherwise we have $k \geq M$ and in an optimal solution Steiner tree T^* contains e, then we have to pay $c(e) \cdot M$ for e. This argument holds, because all edges charged by at least M are connected. We have proven, that the cost function of both problems are equal and the claim follows. □

We have shown that we can firstly reduce Maybecast to Rent-or-Buy and then we can solve the Rent-or-Buy problem using our 3.05-approximation algorithm. Now we want to assemble both aspects into an explicit algorithm. Therefor we have to deliberate at which probability a terminal $i \in R$ that is active with probability $p_i = n_i/M$ will be part of the Steiner tree, spanning the chosen facilities in the sampling algorithm. Let us call the n_i demands, that replace terminal i during reduction, $d_1, ..., d_{n_i}$. Each of these demands $d_1, ..., d_{n_i}$ is chosen with probability $0.67/M$ by the CFL algorithm. Thus the established Steiner tree contains terminal i with probability

$$\Pr\left(\bigcup_{j=1}^{n_i} d_j \text{ is sampled} \right) = 1 - \left(1 - \frac{0.67}{M}\right)^{n_i} = 1 - \left(\left(1 - \frac{0.67}{M}\right)^{M} \right)^{p_i}$$

Algorithm 2 Maybecast algorithm

Input: Graph $G = (V, E)$; terminals $R \subseteq V$ with probabilities p_i $\forall i \in R$; root $r \in V$

Output: Maybecast tree solution

1. Mark each terminal $i \in R$ with probability $1 - e^{-0.67p_i}$
2. Set $R' := \{\text{marked terminals}\} \cup \{r\}$
3. Compute a ρ-approximative Steiner tree T spanning R'
4. Connect terminals to R' using shortest paths
5. Output union of edges in T and edges on shortest paths

We have chosen M to be common denominator of all p_i's, but instead we can set M to an arbitrary multiple of the common denominator, the analysis remains feasible. If we consider $M \to \infty$ we obtain

$$\lim_{M\to\infty} 1 - \left(\left(1 - \frac{0.67}{M}\right)^M\right)^{p_i} = 1 - e^{-0.67p_i}$$

We conclude, that we can replace the reduction to the Rent-or-Buy problem by algorithm 2.

Corollary 3.3. *Algorithm 2 computes a solution for the Maybecast problem, that has expected costs of at most* 4.83 *times the optimal value.*

Proof. Cost function 3.2 differs from the original cost function by a factor of at most $\frac{1}{1-1/e}$. We are able to approximate the problem defined by cost function $\hat{k}_e$ within a factor of 3.05. Thus we obtain an approximation ratio of

$$\frac{1}{1-1/e} \cdot 3.05 \leq 4.83$$

□

4 The complexity of VPND

Recent results and our contributions

During the last years, the best known approximation ratio for VPND was improved from the first constant ratio of 5.55 [8] via 4.74 [4] to 3.55 [5]. Both of the first two algorithms give tree solutions and imply approximation ratios of 4.74 for $\text{VPND}_{\text{Tree}}$.

Before this work, $\mathcal{NP}$-hardness of VPND was known only in the special case that either cardinality of the senders or of the receivers is 1, using a reduction from Steiner tree problem to VPND [7]. There is also a reduction from *Max Leaf Spanning Tree problem* to $\text{VPND}_{\text{Tree}}$ which proves $\mathcal{NP}$-hardness for $|S| = 2$ and $|R| = 2$, respectively. We will repeat the reduction from Steiner tree to VPND in section 4.1. This will allow us to state a new reduction in section 4.2, proving $\mathcal{NP}$-hardness for the general case $|R| = |S| + b$ for each $b \in \mathbb{Z}\backslash\{0\}$. Interestingly in section 5.2 we are able to show that $\text{VPND}_{\text{Tree}}$ is still solvable in polynomial time in the case $b = \pm O(1)$.

4.1 The previously known reduction from Steiner tree to VPND

Let us consider the well-known reduction from Steiner tree problem to VPND. Regard a Steiner tree instance consisting of graph $G = (V, E)$, cost function $c : E \to \mathbb{R}^+$ and terminals $\{s, r_1, ..., r_k\} \subseteq V$. Now we construct a VPND instance by taking over graph G and costs c and by choosing $S = \{s\}$ to be the set of senders and $R = \{r_1, ..., r_k\}$ to be the receivers. Each VPND solution has to connect the senders to all receivers. Because of this $T = \{e \in E | u(e) > 0\}$ is spanning all terminals. In each optimal solution, we have $u(e) \in \{0, 1\}$, because $|S| = 1$. Furthermore, circles in T can be eliminated making the solution cheaper. We conclude that in an optimal VPND solution of this instance, set T is the cheapest tree spanning all the terminals and therefore T is an optimal solution of the Steiner tree instance (see figure 4.1).

This reduction even implies strong $\mathcal{NP}$-hardness and the absence of a PTAS for VPND, unless $\mathcal{NP} = \mathcal{P}$.

4.2 The new reduction

Now we present a modified reduction, that proves $\mathcal{NP}$-hardness even if $|S| > 1$. Let $\mathcal{I}_{\text{St}}$ denote a Steiner tree instance, consisting of graph $G = (V, E)$ and terminals $r_1, s_1, ..., s_k$. Let $b \geq 1$ be an arbitrary integral number. We choose $m = |S| + b$ and construct a VPND instance $\mathcal{I}_{\text{VPND}}$ with senders $S = \{s_1, ..., s_k\}$ and receivers $R = \{r_1, ..., r_m\}$ as follows

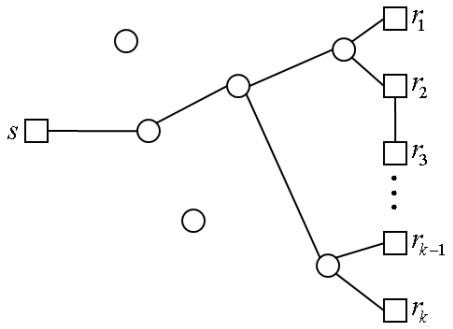


Figure 4.1: Graph with depicted optimal Steiner tree which represents an optimal VPND solution too, if one unit of capacity is reserved on all drawn edges.

- Take graph G.
- Add receivers $r_2, ..., r_m$ and dummy node d.
- Add edges (s_i, d) at costs K for $i = 1, ..., k$ choosing K sufficiently large (for example choose $K := |V| \sum_{e \in E} c(e)$).
- Add edges (d, r_i) at costs 0 for $i = 2, ..., m$.

Let T^* be an optimal solution of $\mathcal{I}_{\text{St}}$.

Theorem 4.1. *An optimal VPND solution of instance $\mathcal{I}_{\text{VPND}}$ consists of reserving exactly one unit of capacity on each edge in $T^* \cup \{(s_i, d) | i = 1, ..., k\} \cup \{(d, r_i) | i = 2, ..., m\}$. Paths starting at r_1 are given by the unique path using edges in T^*. The s_i-r_j path with $i \in \{1, ..., k\}$ and $j \in \{2, ..., m\}$ is defined as path $s_i \to d \to r_j$.*

Proof. It is easy to see that the given solution is feasible, because the paths imply that never more than one unit of capacity is needed on an edge. Let $OPT_{\text{St}} = c(T^*)$ and OPT_{VPND} denote costs of the optimal Steiner tree and VPND solution, respectively. The presented VPND solution has costs of

$$OPT_{\text{St}} + k \cdot K$$

Thus, we have to prove, that each VPND solution for this instance costs at least $OPT_{\text{St}} + k \cdot K$.

Let us consider the traffic scenario in which k senders communicate to k receivers $\{r_2, ..., r_{k+1}\}$[1]. To support this scenario, at least k units of capacity have to be reserved on expensive edges (s_i, d).

Assume that there is not enough capacity on edges in sub graph G to connect r_1 to the senders using only edges in G. Then there must be a pair of senders $s_i \neq s_j$ such

[1] here the assumption $m > k$ is needed

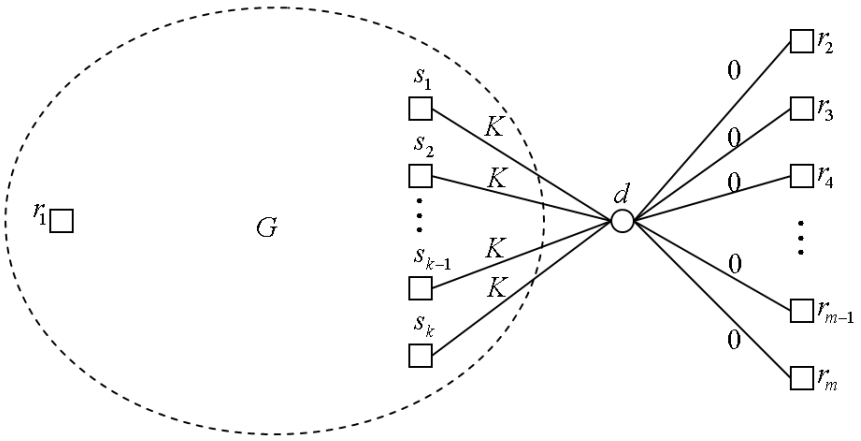


Figure 4.2: VPND instance $\mathcal{I}_{\text{VPND}}$. It is important for the proof of theorem 4.1 that the number of receivers exceeds the number of senders.

that a path $s_i \to d \to s_j \to ... \to r_1$ exists, which is supported by installed capacities. However, this path needs 2 expensive edges. Choosing the traffic scenario, in which s_i sends to r_1 and the remaining $k-1$ senders communicate with $r_2, ..., r_k$, leads to $k+1$ units of capacity needed on expensive edges. Because of $(k+1)K > OPT_{\text{VPND}}$, such a solution can not be optimal.

We conclude, that an optimal solution contains exactly k expensive edges (s_i, d) and edges in G, that are spanning $\{r_1, s_1, ..., s_k\}$. But the optimal Steiner tree T^* is the cheapest set of edges in G spanning $\{r_1, s_1, ..., s_k\}$, so we have

$$OPT_{\text{VPND}} \geq c(T^*) + k \cdot K$$

and the claim follows, that the presented VPND solution is optimal. □

The possibility of optimizing VPND in the case $|R| = |S| + b$ would also imply that the $\mathcal{NP}$-hard Steiner tree problem could be solved. Since we can switch the sets of senders and receivers in the VPND problem, we obtain $\mathcal{NP}$-hardness for VPND in the general case of $|R| = |S| + b$ for all $b \in \mathbb{Z} \backslash \{0\}$.

Unfortunately this proof does not work in the interesting case $|S| = |R|$. We observe that the optimal VPND solution contains the circle $r_1 \to ... \to s_i \to d \to s_j \to ... \to r_1$ for each $i, j \in \{1, ..., k\}$ and therefore does not form a tree. This means that this reduction does *not* work for $\text{VPND}_{\text{Tree}}$.

5 The $\text{VPND}_{\text{Tree}}$ problem

Recent results and our contributions

$\text{VPND}_{\text{Tree}}$ was studied in [7], resulting to a factor 9.002 approximation. This was done using a reduction to Connected Facility Location. Up to now the best CFL algorithm would lead to a ratio of 3.55 for $\text{VPND}_{\text{Tree}}$. Using our refined analysis of the sampling algorithm, an approximation ratio of 3.05 follows even for $\text{VPND}_{\text{Tree}}$.

We can adopt the reduction in 4.1, to proof $\mathcal{NP}$-hardness for $\text{VPND}_{\text{Tree}}$ in the case $|S| = 1$. In this chapter, we will show, that it is worthwhile to distinguish VPND and $\text{VPND}_{\text{Tree}}$, because we are able to proof results for the tree variant, that do not hold in the general case.

After working on the structure of $\text{VPND}_{\text{Tree}}$ in section 5.1, we proceed by extending the cases that are solvable in polynomial time from $|R| = |S|$ [7] to $|R| = |S| \pm O(1)$. Later we give a PTAS in the case $|R| = O(1) \cdot |S|$. In section 5.4, we will improve the approximation ratio of the cheapest shortest path tree solution from $1 + |R|/|S|$ [5] to $\frac{|R|+|S|}{2|S|}$ and we will show that this bound is tight.

5.1 The structure

As we have seen in section 1, we need just a tree $T \subseteq E$ as solution for $\text{VPND}_{\text{Tree}}$; all paths are induced unique by this tree. Of course, we could compute needed capacities in the same manner as for VPND, using maximal matchings. But we will work out, that this aspect becomes easier, if we restrict to tree solutions. Consider an edge $e \in T$. Let T_1 and T_2 denote the subtrees that remain, if we remove e from T. s_1 and s_2 denote the number of senders in T_1 and T_2, respectively. We define r_1 to be the number of receivers in T_1 and r_2 to be the number of receivers in T_2. This situation is depicted in figure 5.1. Let us assume $s_1 + r_1 \leq s_2 + r_2$ (this means T_1 is the subtree with less terminals). Obviously the worst-case scenario for edge e consists of sending $\min\{s_1, r_2\}$ units of flow from T_1 across e to T_2 and $\min\{s_2, r_1\}$ units of flow from T_2 across e to T_1. Therefore the amount of capacity, that has to be installed on e, is

$$u(e) = \min\{s_1, r_2\} + \min\{s_2, r_1\}$$

Let us consider the case $s_1 > r_2$, then we know about the number of terminals in T_1, that $s_1 + r_1 > r_1 + r_2$ holds. Since we assume that the number of senders does not exceed the number of receivers (i.e., $r_1 + r_2 \geq s_1 + s_2$), tree T_1 would contain more than half of the terminals. Because we assumed T_1 to be the smaller of both subtrees, a contradiction follows. Thus we have $s_1 \leq r_2$ and

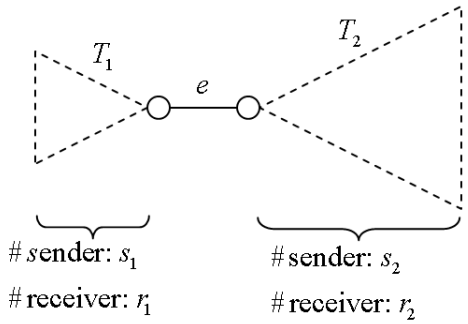


Figure 5.1: Represents notation used in computation of capacities for tree solutions.

$$u(e) = s_1 + \min\{s_2, r_1\}$$

holds. Now two cases are possible. If we have $s_1 + r_1 < |S|$, we conclude $r_1 < s_2$, because of $s_1 + s_2 = |S|$. It follows

$$u(e) = s_1 + \min\{s_2, r_1\} = s_1 + r_1 = \#\text{terminals in } T_1$$

Otherwise we have $s_1 + r_1 \geq |S|$, then $r_1 \geq s_2$ holds and we have

$$u(e) = s_1 + \min\{s_2, r_1\} = s_1 + s_2 = |S|$$

All in all we have proven the following lemma.

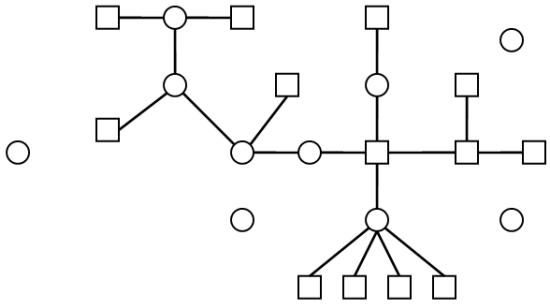
Lemma 5.1. *Consider a VPND$_{Tree}$ instance with graph $G = (V, E)$, senders S and receivers R with $|S| \leq |R|$. For a solution T and an edge $e \in T$ the needed capacity $u(e)$ on edge e is given by*

$$u(e) = \min\{|S|, \#\text{terminals in smaller subtree below } e\}$$

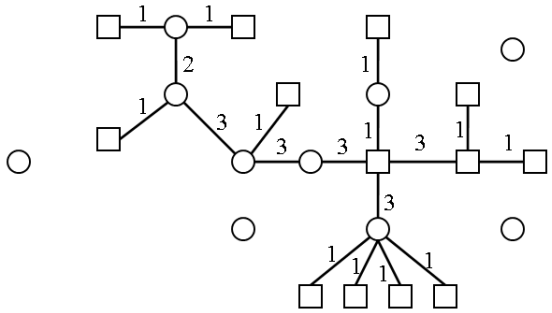
(The smaller subtree is here defined to be the subtree adjacent to e that contains less terminals.)

Using this simplified rule, it is easy to compute capacities for the example in figure 5.2 (a), as we see in figure 5.2 (b). To clarify that capacities do not depend on which terminal is sender and which is receiver, terminals in figure 5.2 are not labeled. Only the number of senders is of relevance. We can divide edges in a tree solution T into two categories

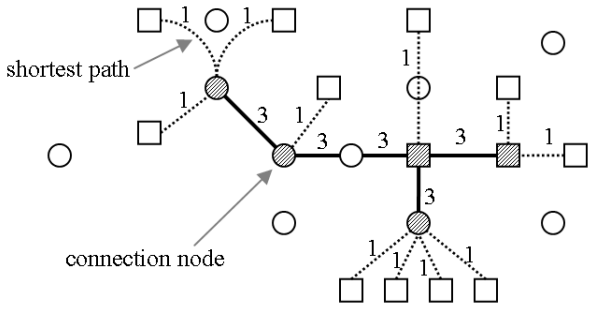
- edges with capacity $|S|$
- edges with capacity $< |S|$



(a) tree solution T



(b) needed capacities



(c) separation into core and shortest paths

Figure 5.2: Needed capacities for tree solution with $|S| = 3$. Edges in (b) are labeled with capacities. In (c) solid edges belong to core C, dotted lines denote shortest paths. Hatched nodes are connection nodes.

We deliberate that edges with maximal capacity of $|S|$ are connected and hence, form a tree. We call this subtree of T, the *core* $C = \{e \in T | u(e) = |S|\}$. After removing the core, distinct subtrees $T_1, ..., T_k \subseteq T$ remain, that share exactly one node with the core. Let $T_i \cap C = \{v_i\}$ for all $i \in \{1, ..., k\}$. Regard a subtree T_i with senders S_i and receivers R_i. Capacities on edges $e \in T_i$ are smaller than $|S|$ and are equal to the number of terminals in the subtree below e. We can interpret capacities inside of T_i, such that there is a separate path from each terminal in $S_i \cup R_i$ to v_i. This means, each terminal in T_i has its own "private" path to the first node in the core, with one unit of capacity installed on it. We call v_i to be the *connection node* of the terminals in T_i. Such paths never share capacity. Because of this, we can change paths independently from each other. But this means, that in an optimal solution paths from terminals to connection nodes must be shortest paths between both.

Let us assume we know all edges in C. We know that we have to install capacity $|S|$ on all of these edges and then we have to connect each terminal in $S \cup R$ via its shortest path to the next node in C, installing one unit of capacity cumulative. Cumulative means, that if k shortest paths cross an edge, we have to install capacity k on this edge. The cost of such a $\text{VPND}_{\text{Tree}}$ solution with core C and connection nodes V' is

$$|S| \cdot c(C) + \sum_{v \in S \cup R} \ell(v, V')$$

Core C is a Steiner tree, spanning all connection nodes, thus in an optimal solution C must be the optimal Steiner tree on the connection nodes.

In other words: If we knew the set of connection nodes in the optimal solution and if we were able to compute an optimal Steiner tree spanning these connection nodes, then we would be able to compute the whole optimal solution. This leads to an exact algorithm for a special case of $\text{VPND}_{\text{Tree}}$.

5.2 A polynomial time algorithm for the case $|R| = |S| \pm O(1)$

Let T^* denote an optimal tree solution with corresponding core C^*. Regard algorithm 3. We have already clarified, that the algorithm computes an optimal solution. But in general, it does not do this in polynomial time. We have to implement step 1 by choosing the cheapest solution after iterating all possible subsets. Thus running time for this step will only be polynomial, if the number of connection nodes can be bounded by a constant. Step 2 is still computable in polynomial time, if $|V'| = O(\log |V|)$, using the Wagner-Dreyfus algorithm [3]. All in all the algorithm runs in polynomial time, if the number of connection nodes is bounded by a constant. This leads to the following question. How many connection nodes are included in an optimal $\text{VPND}_{\text{Tree}}$ solution?

Lemma 5.2. *We are given a $\text{VPND}_{\text{Tree}}$ instance with senders S and receivers R. If $|R| = |S| + O(1)$, then there are $O(1)$ many connection nodes in the optimal solution.*

Proof. Let C^* be the core in the optimal tree solution T^*. So C^* forms a tree. Let $T_1^*, ..., T_k^* \subseteq T^*$ denote the connected components in $T^* \backslash C^*$. There is one of the subtrees

Algorithm 3 Exact algorithm for $\text{VPND}_{\text{Tree}}$

Input: Graph $G = (V, E)$; costs c; senders $S \subseteq V$; receivers $R \subseteq V$

Output: Tree solution $T \subseteq E$

1. Guess connection nodes $\rightarrow V'$
2. Compute an optimal Steiner tree T^* spanning V'
3. Install capacity $|S|$ on edges in T^*
4. Connect all terminals in $S \cup R$ to T^* using shortest paths and install one unit of capacity cumulative
5. Output resulting tree

T_i^*, hanging below each connection node. If C^* is empty, there is only one connection node and the claim follows. Otherwise C^* has at least 2 leafs, because it's a non-empty tree. Trees T_i^*, hanging below leafs of C^*, contain at least $|S|$ terminals, since otherwise there would be no need to install $|S|$ units of capacity on the edge that T_i^* is attached to[1]. So we have at least $|S|$ terminals at each of at least 2 leafs, that have the same connection node. We conclude, that we can upper bound the number of connection nodes by

$$|S| + |R| - 2(|S| - 1) = |R| - |S| + 2 = O(1)$$

using assumption $|R| = |S| + O(1)$. □

So if we regard the case $|R| = |S| + O(1)$, algorithm 3 runs in polynomial time. The case $|R| = |S| - O(1)$ follows similarly. Of course this case $|R| = |S| \pm O(1)$ is strongly restricted. Now we consider the case, in which we can not guess all connection nodes, but at least some of them.

5.3 A PTAS for the case $|R| = O(1) \cdot |S|$

Again T^* denotes an optimal tree solution for the given $\text{VPND}_{\text{Tree}}$ instance, as depicted in figure 5.3 and C^* denotes the core of this solution. We try to find out, how useful it is, to guess $2k$ appropriate nodes belonging to C^* for an integral constant k. Let us regard approximation algorithm 4.

The number of possibilities to choose $2k$ nodes is at most n^{2k} and thus polynomially bounded. Since an optimal Steiner tree on V' can be computed in polynomial time, because of $|V'| = 2k = O(1)$, the running time of the algorithm is still polynomial.

[1] We should mention, that we cannot give a proposition about the number of terminals, belonging to a specific connection node that is an *inner* node of C^*.

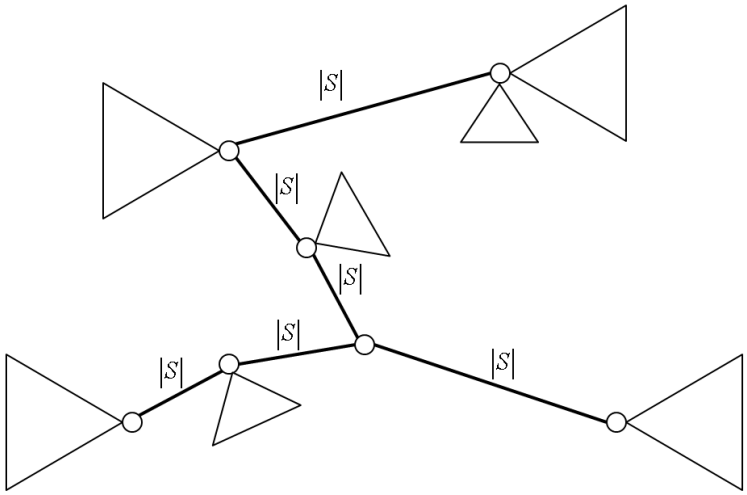


Figure 5.3: An example for the optimal tree solution T^* is depicted. The drawn edges belong to C^* and are labeled with their capacities. Triangles represent connected components of $T^* \backslash C^*$.

Algorithm 4 PTAS for $\text{VPND}_{\text{Tree}}$

Input: Graph $G = (V, E)$; costs c; $S, R \subseteq V$; parameter $k \in \mathbb{N}$

Output: Tree solution $T \subseteq E$

1. For each $V' \subseteq V$ with $|V'| \leq 2k$:
   a) Compute an optimal Steiner tree C spanning V'
   b) Install capacity $|S|$ on edges in C
   c) Connect all terminals in $S \cup R$, using shortest paths to next node in V' and install one unit of capacity cumulative
2. Return the cheapest regarded solution

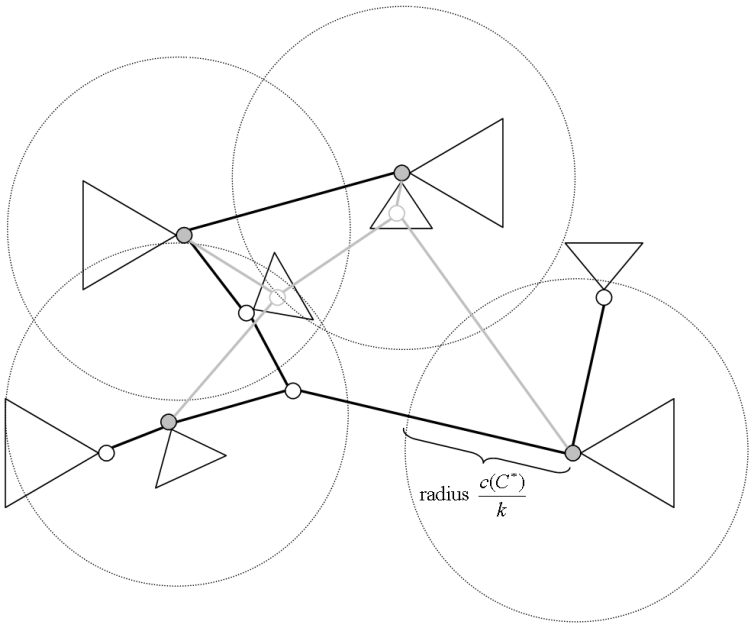


Figure 5.4: Illustration of the analysis of algorithm 4. Circles have radius $c(C^*)/k$. Nodes in V' are filled grey. Grey edges belong to Steiner tree C.

Since the algorithm considers all subsets of size at most $2k$, we obtain a correct upper bound on the gained approximation ratio, if we choose V' in a particular manner. The following lemma will guarantee the existence of an appropriate choice of V'.

Lemma 5.3. *Let C^* be an arbitrary tree, then a subset $V' \subseteq V(C^*)$[2] exists, such that $|V'| \leq 2k$ and $\ell(v, V') \leq \frac{c(C^*)}{k}$ for all $v \in V(C^*)$.*

Proof. Use the following method to construct set V'. We start by setting $V' := \emptyset$. While there is a node $v \in V$ with $\ell(v, V') > \frac{c(C^*)}{k}$, we add this node to V' (set $V' := V' \cup \{v\}$). It is easy to see, that this procedure terminates after at most $|V(C^*)|$ steps. The question to answer is: How many vertices does V' contain in the end?

Consider the Euler tour, containing each edge in C^* exactly twice. Of course the length of this tour is $2c(C^*)$ and each node in V' occurs at least once in the tour. We assign all edges in this tour to the unique node $v \in V'$ that is visited next on a clockwise walk. Since nodes in V' have a distance of at least $\frac{c(C^*)}{k}$ to each other, the total length of all edges assigned to each node is at least $\frac{c(C^*)}{k}$. Because the length of the edges sums up to $2c(C^*)$ we conclude

$$|V'|\frac{c(C^*)}{k} \leq 2c(C^*)$$

By rearranging terms, we obtain $|V'| \leq 2k$. □

[2] $V(C^*)$ denotes the nodes of C^*

In figure 5.4 an appropriate choice of V' and the resulting Steiner tree is depicted. We use this lemma to proof the ratio of our algorithm during the next theorem.

Theorem 5.4. *The approximation ratio of algorithm 4 is at most* $1 + \frac{|S|+|R|}{k|S|}$.

Proof. Lemma 5.3 guarantees, by considering all possible subsets of size at most $2k$, that algorithm 4 will find a set $V' \subseteq V$ with $|V'| \leq 2k$ such that $\ell(v, V') \leq \frac{c(C^*)}{k}$ holds for all $v \in V(C^*)$. We bound costs for such a choice of V'. Let again T^* be an optimal tree solution with core C^*. We denote $w : S \cup R \to V$ to be the function that returns the connection node of a terminal in an optimal solution. So we can upper bound costs of the computed solution by

$$\begin{aligned}
& |S|c(C) + \sum_{v \in S \cup R} \ell(v, V') \\
\overset{(1)}{\leq} \; & |S|c(C^*) + \sum_{v \in S \cup R} \ell(v, w(v)) + \sum_{v \in S \cup R} \ell(w(v), V') \\
\overset{(2)}{=} \; & OPT + \sum_{v \in S \cup R} \ell(w(v), V') \\
\overset{(3)}{\leq} \; & OPT + |S \cup R| \cdot \frac{c(C^*)}{k} \\
\overset{(4)}{\leq} \; & OPT + |S \cup R| \cdot \frac{OPT}{k|S|} \\
= \; & OPT \cdot \left(1 + \frac{|S| + |R|}{k|S|}\right)
\end{aligned}$$

(1) follows by triangle inequality and by $c(C) \leq c(C^*)$, since an optimal Steiner tree spanning $V' \subseteq V(C^*)$, can not be more expensive than a Steiner tree C^* spanning $V(C^*)$. (2) holds, since in an optimal solution, each terminal v is connected to its connection node $w(v)$ via a private path. This implies

$$|S|c(C^*) + \sum_{v \in S \cup R} \ell(v, w(v)) = OPT.$$

Step (3) follows from our assumption to V', that we gained in lemma 5.3. The last inequality finally holds because of $|S|c(C^*) \leq OPT$. □

Now we state the following

Corollary 5.5. *If* $|R| \leq l|S|$ *for any constant* $l \geq 1$*, algorithm 4 gives a PTAS for* $VPND_{Tree}$.

Proof. We are given $\varepsilon > 0$. The costs of the solution, computed by algorithm 4 is at most

$$OPT \cdot \left(1 + \frac{|S| + |R|}{k|S|}\right) \leq OPT \cdot \left(1 + \frac{|S| + l|S|}{k|S|}\right) = OPT \cdot \left(1 + \frac{1+l}{k}\right)$$

Choosing $k := \lceil \frac{1+l}{\varepsilon} \rceil$, we obtain costs of at most $OPT \cdot (1+\varepsilon)$, implying an approximation ratio of $1 + \varepsilon$. □

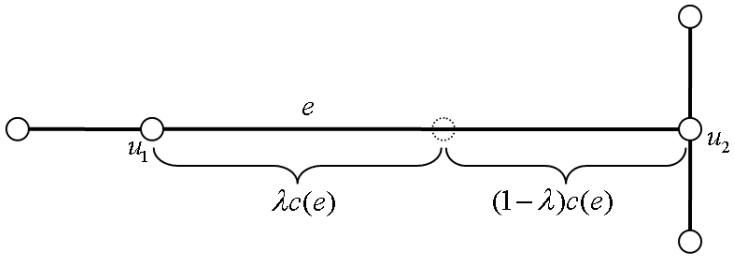


Figure 5.5: Illustration of a tree T, that does not contain a node, which is closer than $\frac{c(T)}{2}$ to all other nodes in T. The point on edge $e = (u_1, u_2)$, fulfilling this property is drawn dotted.

5.4 The shortest path tree is a $\frac{|R|+|S|}{2|S|}$-approximation

The most simple approximative solution for VPND is the cheapest shortest path tree. This means, we choose in turn each node $v \in V$ to be the root and add all edges on shortest paths from the root to terminals into the solution, installing one unit of capacity per path cumulative. In [5] it was shown, that the cost of such a solution is within $1 + |R|/|S|$ times the cost of an optimal VPND solution. This is revealed by summing up the costs of all shortest path trees and arguing that the cost of the cheapest one can not exceed average costs of all shortest path trees.

But if we know, that the optimal solution forms a tree, we are able to argue better. Consider an optimal tree solution $T^* \subseteq E$ for our $\text{VPND}_{\text{Tree}}$ instance with capacities $u : E \to \mathbb{N}_0$ and core $C^* = \{e \in T^* | u(e) = |S|\}$. We will show that a node $v^* \in V$ exists, that leads to an acceptable shortest path tree, if we use it as root.

Lemma 5.6. *In each weighted tree $T = (V, E)$, a node $v^* \in V$ exists, such that $\sum_{v \in V} \ell(v, v^*) \leq |V| \frac{c(T)}{2}$.*

Proof. If a node v^* exists with $\ell(v, v^*) \leq \frac{c(T)}{2}$ for each $v \in V$, there is nothing to show. But regarding figure 5.5 reveals that this is not always the case. But then there must be an edge $e = (u_1, u_2)$ in T, such that removing e divides T into two connected components with costs of $< \frac{c(T)}{2}$ for each one. We denote V_1 as the nodes in the first component and denote $V_2 = V \backslash V_1$ to be the nodes in the other component. If we think of edge e as a line of length $c(e)$, there must be a point on e that has a distance of at most $\frac{c(T)}{2}$ to all nodes in V. We choose $\lambda \in [0, 1]$, such that the distance between u_1 and this point is $\lambda c(e)$ and distance between u_2 and this points is $(1 - \lambda)c(e)$. We insert a new node v^* at this point, by substituting e through edges (v^*, u_1) and (v^*, u_2) with costs $c(v^*, u_1) := \lambda c(e)$ and $c(v^*, u_2) := (1 - \lambda)c(e)$, respectively. Our artificial node v^* fulfills

$\ell(v, v^*) \leq \frac{c(T)}{2}$ for all $v \in V$. We conclude the following fact for V_1

$$|V_1|\frac{c(T)}{2} \geq \sum_{v \in V_1} \ell(v, v^*) = \sum_{v \in V_1} \ell(v, u_1) + \lambda |V_1| c(e)$$

analogous we know for V_2

$$|V_2|\frac{c(T)}{2} \geq \sum_{v \in V_2} \ell(v, v^*) = \sum_{v \in V_2} \ell(v, u_2) + (1-\lambda) |V_2| c(e).$$

Adding both inequalities we obtain

$$\sum_{v \in V_1} \ell(v, u_1) + \lambda |V_1| c(e) + \sum_{v \in V_2} \ell(v, u_2) + (1-\lambda)|V_2| c(e) \leq (|V_1| + |V_2|)\frac{c(T)}{2}$$

By rearranging terms we get

$$\underbrace{\sum_{v \in V_1} \ell(v, u_1) + \sum_{v \in V_2} \ell(v, u_2) + |V_2| c(e)}_{\text{independent from } \lambda} + \lambda c(e)\left(|V_1| - |V_2|\right) \leq |V|\frac{c(T)}{2} \tag{5.1}$$

using $V_1 \cup V_2 = V$. If we now minimize the left hand side over $\lambda \in [0, 1]$, we either reach a minimum for $\lambda = 0$ or for $\lambda = 1$. The first case occurs if $|V_1| \geq |V_2|$, then we choose $v^* := u_1$, otherwise we set $v^* := u_2$. Since the left hand side of 5.1 is equal to $\sum_{v \in V} \ell(v, v^*)$ we conclude

$$\sum_{v \in V} \ell(v, v^*) \leq |V|\frac{c(T)}{2}$$

□

The main work has been done, now we proof the following theorem.

Theorem 5.7. *The cheapest shortest path tree has an approximation ratio of at most* $\frac{|R|+|S|}{2|S|}$ *for* $VPND_{Tree}$.

Proof. Let C^* be the core of an optimal $\text{VPND}_{\text{Tree}}$ solution T^*, then we denote $w(v)$ as the connection node of terminal $v \in S \cup R$ in the optimal solution. We apply lemma 5.6 to tree C^*, regarding V as multiset containing each node in C^* with the same multiple, that the node occurs as connection node of terminals. We obtain a node v^* with the following property

$$\sum_{v \in R \cup S} \ell(w(v), v^*) \leq |R \cup S|\frac{c(C^*)}{2} \tag{5.2}$$

Now we bound the cost of a shortest path tree using v^* as root

$$\begin{aligned}
& \sum_{v \in R \cup S} \ell(v, v^*) \\
\overset{(1)}{\leq} & \sum_{v \in R \cup S} \ell(v, w(v)) + \sum_{v \in R \cup S} \ell(w(v), v^*) \\
\overset{(2)}{\leq} & \sum_{v \in R \cup S} \ell(v, w(v)) + (|R| + |S|)\frac{c(C^*)}{2} \\
= & \sum_{v \in R \cup S} \ell(v, w(v)) + 2|S|\frac{c(C^*)}{2} + (|R| - |S|)\frac{c(C^*)}{2} \\
\overset{(3)}{\leq} & OPT + (|R| - |S|)\frac{OPT}{|S|2} \\
= & OPT\frac{|R| + |S|}{2|S|}
\end{aligned}$$

In (1) we apply the triangle inequality, step (2) follows from inequality 5.2. (3) holds because of

$$\sum_{v \in R \cup S} \ell(v, w(v)) + |S|c(C^*) = OPT$$

and $|S|c(C^*) \leq OPT$. □

We wonder whether this bound is tight. But the next section will show, that this is the fact.

5.5 A worst-case example for the shortest path tree solution

Let us regard instance $\mathcal{I}$ depicted in figure 5.6. We assume $|S|$ and $|R|$ to be even. Roughly spoken, the depicted graph contains one half of the terminals on each side. Now we deliberate, how expensive the cheapest shortest path tree solution is for this instance.

Lemma 5.8. *The approximation ratio of the shortest path tree solution on instance $\mathcal{I}$ is $\frac{|R|+|S|}{2|S|}(1-\varepsilon)$ w.r.t. an optimal $VPND_{Tree}$ solution.*

Proof. The distance from v^* to each demand is $1-\varepsilon$, thus a shortest path tree with root v^* has costs of $(|R|+|S|)\cdot(1-\varepsilon)$. If we use any other node as root the distance to one half of the terminals will be 0 and the distance to the other half will be $2 \cdot (1-\varepsilon)$, implying that a shortest path tree having such a node as root has costs of $\frac{|R|+|S|}{2} \cdot 2 \cdot (1-\varepsilon) = (|R|+|S|) \cdot (1-\varepsilon)$. Hence, the cheapest shortest path tree costs $(|R|+|S|) \cdot (1-\varepsilon)$, too.

The larger R compared to S is, the better is a solution that is created by installing capacity $|S|$ on edges (v_1, v^*) and (v_2, v^*) and adding edges from terminals to v_1 and v_2, respectively. Since the cost of this solution is $2|S|$, the ratio of the shortest path tree is $\frac{|R|+|S|}{2|S|}(1-\varepsilon)$. □

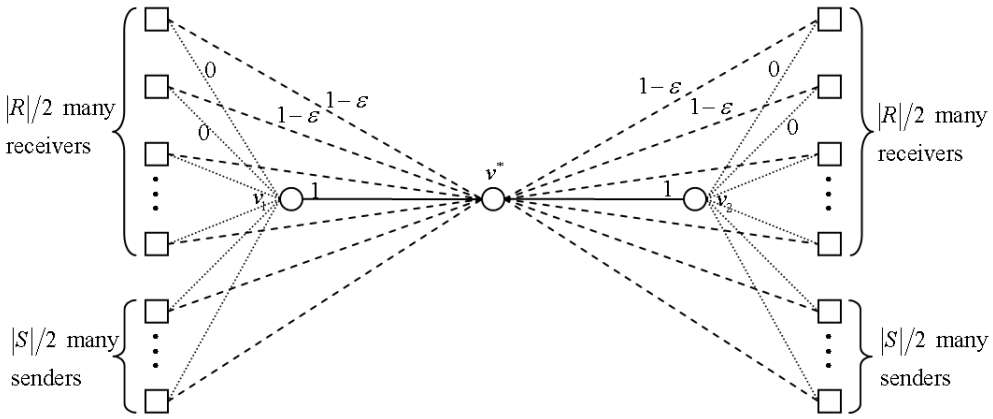


Figure 5.6: Worst-Case instance $\mathcal{I}$ for shortest path tree solution. Dotted edges have cost 0, dashed edges cost $1 - \varepsilon$ and solid edges have cost 1. Rectangles represent terminals, other nodes are represented by circles.

Because $\varepsilon > 0$ can be chosen arbitrarily small, the ratio of the shortest path tree is in general not better than $\frac{|R|+|S|}{2|S|}$, according to the proven upper bound.

5.6 A factor 3.05 approximation algorithm

Let us recall results from section 5.1. An alternative view to $\text{VPND}_{\text{Tree}}$ is to choose connection nodes, a Steiner tree spanning all these connection nodes and connections from each demand to the next connection node. If we denote V' as the set of connection nodes, we can state the value of an optimal $\text{VPND}_{\text{Tree}}$ solution as

$$\min_{V' \subseteq V} \left\{ |S| \cdot \text{costs of a Steiner tree spanning } V' + \sum_{v \in S \cup R} \ell(v, V') \right\}$$

Consider a CFL instance with given demands $D \subseteq V$ and parameter M. The optimal objective function value of this problem is

$$\min_{F \subseteq V} \left\{ M \cdot \text{costs of a Steiner tree spanning } F + \sum_{v \in D} \ell(v, F) \right\}$$

It is easy to see that we can state an approximation preserving reduction from $\text{VPND}_{\text{Tree}}$ to Connected Facility Location by choosing $S \cup R$ as demands and setting $M := |S|$. Thus we can solve $\text{VPND}_{\text{Tree}}$ with algorithm 2.1, obtaining an approximation ratio of 3.05 for $\text{VPND}_{\text{Tree}}$, too.

6 The $\text{VPND}_{\text{MinCon}}$ problem

Recent results and our contributions

Up to now there are no relevant publications about $\text{VPND}_{\text{MinCon}}$. Neither approximation algorithms with non-trivial bounds, nor lower bounds on its approximability are known. The only known fact is the $\mathcal{NP}$-hardness of this problem. In this chapter we will show, that the relaxed problem is solvable in polynomial time. We state a general lower bound for $\text{VPND}_{\text{MinCon}}$ in section 6.3 that will put us into the position, to proof an integrality gap of $\Omega(\sqrt{n}/\log n)$, related to the natural relaxation of $\text{VPND}_{\text{MinCon}}$. This is an indication for the non-approximability of this problem, but unfortunately not a proof. Additionally we will show that randomized rounding leads to disastrously bad approximation ratios.

6.1 A polynomial time algorithm for the relaxed problem

We start by giving a definition of the relaxed $\text{VPND}_{\text{MinCon}}$ problem.

Def. Relaxed $\text{VPND}_{\text{MinCon}}$. We are given an undirected graph graph $G = (V, E)$, senders $S \subseteq V$ and receivers $R \subseteq V$ (with $S \cap R = \emptyset$). A solution is given by paths $P^i_{s,r}$ for each pair of sender $s \in S$ and receiver $r \in R$ with weights $\lambda^i_{s,r} \geq 0$ fulfilling $\sum_i \lambda^i_{s,r} = 1$, as well as edge capacities $u : E \to \mathbb{N}_0$, supporting each valid traffic scenario. The goal is to minimize $\max\{u(e) | e \in E\}$.

Similar to VPND, we can compute needed capacities on an edge $e \in E$, by creating a complete, bipartite graph $G_e = (S \cup R, S \times R)$ with weight $\sum_{i:\, e \in P^i_{s,r}} \lambda^i_{s,r}$ on edge (s, r). The value of a maximal matching in G_e denotes the capacity, needed on e.

Now we want to state an integral linear program for $\text{VPND}_{\text{MinCon}}$ using well-known flow conditions from literature [15, 13]. But these only work on directed graphs. Hence we replace each undirected edge $\{u, v\}$ in our graph by two directed edges (u, v) and (v, u). The congestion on $\{u, v\}$ is defined to be the sum of the congestion on (u, v) and (v, u).

We start by introducing the integral variables

$$\begin{aligned} x_e^{s,r} &= \begin{cases} 1 \text{ if edge } e \text{ lies on path from } s \in S \text{ to } r \in R \\ 0 \text{ otherwise} \end{cases} \\ y &= \text{maximal congestion} \end{aligned}$$

Now the integral linear program (ILP) is the following

$$\begin{aligned}
& \min y \\
\sum_{e\in\delta^+(v)} x_e^{s,r} - \sum_{e\in\delta^-(v)} x_e^{s,r} = & \begin{cases} 1 \text{ if } v = s \\ -1 \text{ if } v = r \\ 0 \text{ otherwise} \end{cases} && \forall s \in S\ \forall r \in R\ \forall v \in V \\
\sum_{(s,r)\in\mathcal{M}} x_e^{s,r} + x_{e^{-1}}^{s,r} \leq & \quad y && \forall e \in E\ \forall \text{ matchings } \mathcal{M} \text{ in } S \times R \\
x_e^{s,r} \in & \quad \{0,1\} && \forall e \in E\ \forall s \in S\ \forall r \in R
\end{aligned}$$

For an edge $e = (u, v)$, we define $e^{-1} = (v, u)$ to be the edge in the opposite direction. Let $\delta^+(v)$ and $\delta^-(v)$ denote outgoing and incoming edges of v, respectively. Integrality of y follows by integrality of $x_e^{s,r}$. The first inequality ensures, that for each pair $(s, r) \in S \times R$ of senders and receivers, a flow of strength 1 is sent from s to r. The second inequality guarantees, that the congestion on edge e, defined by any valid traffic scenario, does not exceed y.

We obtain relaxation (LP) by substituting condition $x_e^{s,r} \in \{0, 1\}$ by $0 \leq x_e^{s,r} \leq 1$. But we cannot solve (LP) as usual, because the second inequality causes trouble. Since there is an exponential number of matchings on $S \times R$, there are exponential many inequalities of type 2. However we can proof the following lemma.

Lemma 6.1. *An optimal fractional solution of (LP) can be computed in polynomial time.*

Proof. We can solve this problem by using the ellipsoid method, because this algorithm does not need an explicit formulation of all inequalities. It can compute an optimal solution in polynomial time, if we give an oracle that can decide whether a solution $(x_e^{s,r}, y)$ is feasible and in the negative case, return a violated inequality. This principle is called *optimization by separation* [6, 13].

Let $(x_e^{s,r}, y)$ denote a solution, for which we have to solve the separation problem. We use the following oracle:

1. *Verify inequalities of type 1 and 3.* If one of this polynomial many inequalities is violated by $(x_e^{s,r}, y)$, return the according inequality.

2. *Verify inequalities of type 2.* For each edge $e \in E$, we construct a complete, bipartite graph $G_e = (S \cup R, S \times R)$, in which edge $(s, r) \in S \times R$ is weighted with $x_e^{s,r} + x_{e^{-1}}^{s,r}$. If the value of maximal matching in G_e exceeds y, the following inequality is returned
$$\sum_{(s,r)\in\mathcal{M}} x_e^{s,r} + x_{e^{-1}}^{s,r} \leq y$$
whereupon $\mathcal{M}$ denotes a maximal matching on G_e.

3. If no violated inequality is found in steps (1) and (2), return "solution feasible".

Since a maximal matching can be computed efficiently and verifying a polynomial number of inequalities takes polynomial time, separation can be done efficiently. We conclude, that we are able to compute an optimal solution of (LP) in polynomial time. $\square$

We aren't ready, yet, because we have just weights $x_e^{s,r}$, that are induced by all s-r paths and not yet the s-r paths itself and their weights. Now it is our goal to compute these paths. We define a weighted graph $G_{s,r} = (V, E_{s,r})$ with

$$e \in E_{s,r} :\Leftrightarrow x_e^{s,r} > 0$$

and weight $x_e^{s,r}$ on edge e. Let $P_{s,r}^1 = (s = v_1, v_2, ..., v_l = r)$ be an arbitrary path in $G_{s,r}$ from s to r obtained via depth first search. We assign a weight of

$$\lambda_{s,r}^1 := \min_{e \in P_{s,r}^1} \{x_e^{s,r}\} > 0$$

to this path $P_{s,r}^1$. Afterwards we lower weights $x_e^{s,r}$ on all edges in $P_{s,r}^1$ by $\lambda_{s,r}^1$. Edges now having weight 0 are removed. We proceed with this method, until graph $G_{s,r}$ is empty. Because in each step at least one edge is being removed, we obtain at most $|E|$ s-r paths and their weights $\lambda_{s,r}^i$. This principle of flow decomposition, called *path stripping*, has already been applied in [15]. We conclude

Corollary 6.2. *An optimal fractional solution of VPND$_{MinCon}$ can be computed in polynomial time.*

6.2 Permutation Routing

An alternative view to VPND$_{\text{MinCon}}$ is that we have to compute paths, on which we send packets, such that the number of packets crossing a single edge in the worst-case is minimized. Now we want to present a well-known routing problem, that is related to VPND$_{\text{MinCon}}$.

Def. Permutation Routing. We are given an undirected graph $G = (V, E)$ with $V = \{1, ..., n\}$. We have to compute paths $\mathcal{P} = \{P_{u,v} | u, v \in V, P_{u,v}$ is u-v path$\}$ for all pairs of nodes, such that for each permutation $\sigma \in S_n$ the maximal number of steps is minimized, that are needed to route n packets from nodes $i \in \{1, ..., n\}$ to $\sigma(i)$ using paths $P_{i,\sigma(i)}$, whereupon in each step at most one packet may cross a single edge; other packets have to wait.

The main difference to VPND$_{\text{MinCon}}$ is that Permutation Routing tries to minimize the number of steps, needed until all packets have reached their destinations, while VPND$_{\text{MinCon}}$ minimizes maximal congestion on an edge. But if k packets cross an edge, the number of needed steps in Permutation Routing must be at least k, since only one packet may pass that edge per step. So both problems have substantial similarity. Since Permutation Routing is a well examined problem in literature [2, 14, 11] we can adopt results and use it for VPND$_{\text{MinCon}}$.

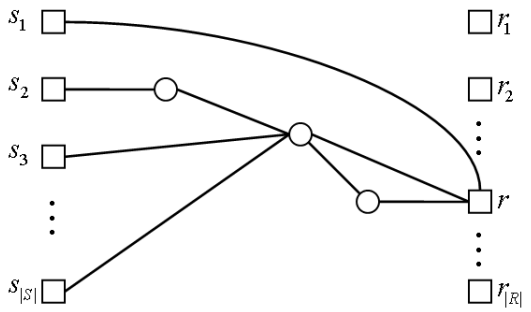


Figure 6.1: Destination graph G_r for receiver r

6.3 A general lower bound

In section 6.4, it is our goal to proof a large integrality gap for $\text{VPND}_{\text{MinCon}}$. Before we can do this, we have to do some preliminary work. We claim that graphs with low degrees never have cheap $\text{VPND}_{\text{MinCon}}$ solutions. Borodin and Hopcroft [2] proved that oblivious deterministic Permutation Routing in a graph with n nodes and degree of at most d, takes time $\Omega(\sqrt{n}/d^{3/2})$. This bound was improved to $\Omega(\sqrt{n}/d)$ in [11]. Now we will adopt a theorem from [11] for our use.

Theorem 6.3. *We are given a graph $G = (V, E)$ with n nodes, degree of at most d, senders $S \subseteq V$ and receivers $R \subseteq V$, fulfilling $|S| = \Omega(n)$ and $|R| = \Omega(n)$. Then the value of each solution for $VPND_{MinCon}$ is at least $\Omega(\sqrt{n}/d)$.*

Proof. Let $P_{s,r}$ denote the path from sender s to receiver r. Consider a fixed receiver $r \in R$ and all paths that have r as destination. We call the union of this paths the *destination graph* G_r (see figure 6.1). We define an edge in G_r, that is contained in at least k paths with destination r, to be *k-congested*. S_k denotes the set of all k-congested edges and $V(S_k)$ denotes the set of vertices, being incident to k-congested edges. We assume $k \leq |S|/d$. Since r has degree of at most d and the number of paths reaching r is $|S|$, we know that $r \in V(S_k)$. Because a k-congested edge is incident to only two nodes, we have $|V(S_k)| \leq 2|S_k|$.

We consider s-r path $P_{s,r} = (s = v_1, v_2, ..., v_{l-1}, v_l = r)$ with sender $s \in S$. Let v_i be the first node on this path, belonging to $V(S_k)$ (nevertheless no edge on path $P_{s,r}$ has to be k-congested). Such a node must exist, because $r \in V(S_k)$. We conclude $v_{i-1} \notin V(S_k)$ and hence (v_{i-1}, v_i) is not k-congested. Let us assign each of the $|S|$ many paths to the first node in such paths, lying in $V(S_k)$. In each path this first node in $V(S_k)$ is reached, using an edge that is not k-congested. Therefore there are at most $(k-1)d = O(kd)$ paths assigned to each node in $V(S_k)$. This implies $|S| = O(kd) \cdot |V(S_k)|$. Because of

$|S| = \Omega(|V|)$ it follows that

$$\begin{aligned} n &= |V| \\ &= O(kd \cdot |V(S_k)|) \\ &= O(kd \cdot |S_k|) \end{aligned}$$

For the number of k-congested edges we know that

$$|S_k| = \Omega\left(\frac{n}{kd}\right)$$

holds. We define the weight of an edge, to be the number of destination graphs, in which this edge is k-congested. Since we have $|R| = \Omega(n)$ many destination graphs, total weight of all edges is at least $\Omega(\frac{n^2}{kd})$. Because the graph is degree-bounded, it contains at most nd edges and thus, the pigeon-hole principle implies, that there is an edge $e \in E$ with weight of at least $\Omega(\frac{n^2}{kd \cdot nd}) = \Omega(\frac{n}{kd^2})$. Now we choose k, fulfilling $k = \Theta(\frac{n}{kd^2}) \Leftrightarrow k = \Theta(\frac{\sqrt{n}}{d})$, then we have chosen k, such that e is k-congested in k destination graphs. This means, a valid traffic scenario can be built, which puts a congestion of at least $k = \Omega(\frac{\sqrt{n}}{d})$ on edge e. We conclude that the objective function value of each $\text{VPND}_{\text{MinCon}}$ solution is at least $\Omega(\frac{\sqrt{n}}{d})$ and the claim then follows. □

6.4 Integrality gap

To state a proof for a high integrality gap for $\text{VPND}_{\text{MinCon}}$ we need a graph

1. having a small degree, such that the lower bound given by theorem 6.3 can be applied
2. allowing a cheap fractional solution

It will turn out, that the well-known k-dimensional hypercube fits these requirements.

Def. Hypercube. We call graph $H_k = (V, E)$ with $V = \{0,1\}^k$, containing an edge $((x_1, ..., x_k), (y_1, ..., y_k)) \in \{0,1\}^k$ if and only if $(x_1, ..., x_k)$ and $(y_1, ..., y_k)$ differ in exactly one position, the k-dimensional hypercube (see figure 6.2).

Now we apply the lower bound, proven in theorem 6.3 to prove the following lemma.

Lemma 6.4. *Let $H_k = (V, E)$ be the k-dimensional hypercube containing senders $S = \{(0, x_2, ..., x_k) | x_i \in \{0,1\}\}$ and receivers $R = \{(1, x_2, ..., x_k) | x_i \in \{0,1\}\}$. Then for $n = |V| = 2^k$ the value of any $VPND_{MinCon}$ solution for this instance is at least $\Omega(\sqrt{n}/\log n)$.*

Proof. The degree in graph H_k is $k = \log n$ in each node. Because of $|S| = |R| = n/2$ theorem 6.3 gives a lower bound of $OPT = \Omega(\sqrt{n}/\log n)$. □

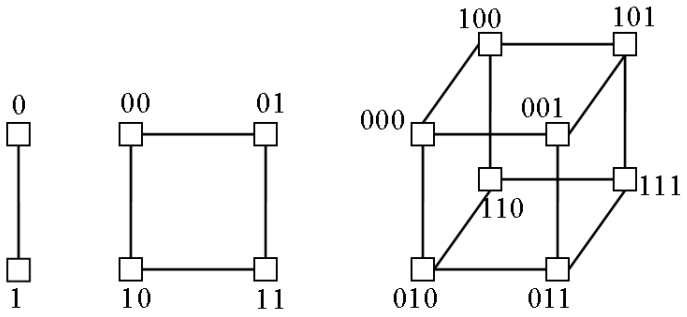


Figure 6.2: Hypercubes H_1, H_2 and H_3

Remark. The authors in [11] present a deterministic scheme for Permutation Routing in the hypercube, that is able to send all packets to their destinations in time $O(\sqrt{n}/\log n)$. If we take the same paths for our $\text{VPND}_{\text{MinCon}}$ setting, we obtain a solution, in which the congestion on edges is within $O(\sqrt{n}/\log n)$. This implies $OPT = \Theta(\sqrt{n}/\log n)$ for the hypercube graph.

Let OPT_f denote the value of an optimal fractional solution of $\text{VPND}_{\text{MinCon}}$ on hypercube H_k with senders $S = \{(0, x_2, ..., x_k) | x_i \in \{0, 1\}\}$ and receivers $R = \{(1, x_2, ..., x_k) | x_i \in \{0, 1\}\}$. Now we want to construct a cheap fractional solution, to give an upper bound on OPT_f.

We have to state fractional paths from each $s \in S$ to each $r \in R$. We use each node $w \in V$ to serve as interstation and hence put a weight of $\frac{1}{2^k}$ on path $s \to w \to r$, whereupon paths from s to w and from w to r, respectively, are given by bitfixing routing. The path, given by bitfixing routing from $(x_1, ..., x_k)$ to $(y_1, ..., y_k)$, is defined as path starting at $(x_1, ..., x_k)$, then going to $(y_1, ..., y_{i^*}, x_{i^*+1}, ..., x_k)$ with $i^* = \min\{i | x_i \neq y_i\}$. Afterwards bitfixing proceeds routing recursively to $(y_1, ..., y_k)$. For example in the case $k = 3$ the path from 100 to 010 visiting interstation 111 would be

$$100 \to 110 \to 111 \to 011 \to 010$$

The idea behind this method is that the weight we put on a path is equal to the probability at which the randomized algorithm for Permutation Routing in [14] chooses this path. Thus, we can adopt the analysis from [14].

Lemma 6.5. *Maximal congestion caused by the given fractional solution on H_k is at most* 1.

Proof. Consider the congestion, that is caused by the first half of the paths from senders to the interstations (this means we do not consider the full paths $s \to w \to r$, but $s \to w$). Since interstations do not depend on the receiver, which is given as destination, the following analysis is valid for each traffic scenario. Because of symmetry we can bound congestion, caused by the second half, in the same manner.

Let $e = ((a_1, ..., a_{j-1}, a_j, a_{j+1}, ..., a_k), (a_1, ..., a_{j-1}, 1 - a_j, a_{j+1}, ..., a_k))$ be an arbitrary edge in the hypercube. An edge in H_k connects two nodes that differ in exactly one position. For edge e this is bit j. All paths that may cross e, using bitfixing, are of the following form

$$\begin{aligned}(*, ..., *, a_j, a_{j+1}, ..., a_k) &\rightarrow (a_1, ..., a_{j-1}, a_j, a_{j+1}, ..., a_k) \\ &\rightarrow (a_1, ..., a_{j-1}, 1 - a_j, a_{j+1}, ..., a_k) \\ &\rightarrow (a_1, ..., a_{j-1}, 1 - a_j, *, ..., *)\end{aligned}$$

whereupon $*$ is a wild-card for arbitrary values in $\{0, 1\}$. Such paths can only have their beginning in nodes, whose bits match the bits of the starting node of e up from bit j. The first j bits of the destination node of such a path have to match the corresponding bits in the end node of e. Thus there are at most 2^{j-1} many senders, that have paths crossing e. From each of these senders at most 2^{k-j} paths, that contain edge e, have their beginning. Thus the weight of all paths starting at a single sender and crossing e is bounded by $(\frac{1}{2})^j$. Summing up weights of all paths crossing e, we obtain a total weight of at most $\frac{1}{2}$. If we now add the left out second half, we conclude $OPT_f \leq 1$. □

All in all we have proven the following

Corollary 6.6. *The integrality gap of $VPND_{MinCon}$ related to its natural relaxation is at least $OPT/OPT_f = \Omega(\sqrt{n}/\log n)$.*

A fractional $\text{VPND}_{\text{MinCon}}$ solution can be interpreted, such that paths are chosen with a probability, according to their weight. This possibility of randomization does not exist in the integral problem. This discrepancy corresponds to the difference between randomized and deterministic routing. It is well-known, that deterministic routing is quite more expensive w.r.t. congestion than randomized routing [2, 11].

If $\text{VPND}_{\text{MinCon}}$ occurs in practice, it will be a good idea to verify, whether the given application allows choosing paths randomized. As we have seen in section 6.1 in this case, optimal paths and their probabilities can be computed efficiently.

6.5 Why randomized rounding will not work

Many approximation algorithms for $\mathcal{NP}$-hard optimization problems consist of rounding an optimal fractional solution. Since we can compute an optimal solution in polynomial time, we should answer the question, whether this proceeding can be successfully applied to $\text{VPND}_{\text{MinCon}}$.

Simplifying our problem, such that only one previously known traffic scenario has to be supported, is known as MinCongestion [15]. The algorithm presented in [15] uses randomized rounding and its ratio can be bounded by $O(\log n/\log\log n)$, with n being the number of sender-receiver pairs.

Although we have proven an integrality gap of $\Omega(\sqrt{n}/\log n)$ in section 6.4 we could hope to get a feasible integral solution that has costs of $O(\sqrt{n}/\log n) \cdot OPT_f \leq O(\sqrt{n}/\log n) \cdot OPT$ via randomized rounding, to gain an approximation ratio of $O(\sqrt{n}/\log n)$. In this

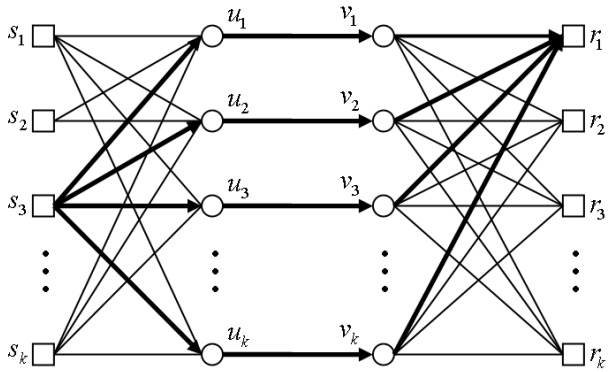


Figure 6.3: Worst-case instance for randomized rounding. All s_3-r_1 paths with weight $1/k$ for each, are drawn bold.

section we want to give a (negative) answer to this question, because we can show, that randomized rounding will not work.

For this purpose let us regard graph $G = (V, E)$ with nodes $V = \{s_i, r_i, u_i, v_i | i = 1, ..., k\}$ (i.e., $n = |V| = 4k$), senders $S = \{s_1, ..., s_k\}$ and receivers $R = \{r_1, ..., r_k\}$ which is depicted in figure 6.3. The edges (s_i, u_j) and (v_i, r_j) exist for all $i, j = 1, ..., k$ and for all $i = 1, ..., k$ we have edges (u_i, v_i).

Let OPT and OPT_f be values of an optimal integral and fractional solution, respectively. Choosing path $s_i \to u_i \to v_i \to r_j$ from sender s_i to receiver r_j, we obtain an integral solution with maximal congestion of 1, thus we have $OPT = 1$.

Now we give a fractional solution, having the same value. We define the set of paths from sender s_i to receiver r_j to be the set $\mathcal{P}_{ij} = \{s_i \to u_l \to v_l \to r_j | l = 1, ..., k\}$, containing each of these k paths with weight $1/k$. No matter which valid traffic scenario is chosen, each edge (u_i, v_i) is crossed by exactly k paths having weight $1/k$ for each one, therefore the value of this fractional solution is 1. Each traffic scenario defined by a maximal matching, sends k units of flow from the senders to the receivers. Because edges $\{(u_i, v_i) | i = 1, ..., k\}$ form a cut containing k edges, the pigeon-hole principle implies, that there is an edge congested with at least one unit of flow. Hence there can not be a cheaper fractional solution and $OPT_f = 1$ follows.

Now we reveal what happens, if we use standard randomized rounding.

Theorem 6.7. *Consider a VPND solution in graph G, in which we choose path $s_i \to u_l \to v_l \to r_j$ for sender-receiver pair $(s_i, r_j) \in S \times R$ independently with probability $1/k$. Then with probability $1 - 2^{-\Omega(k)}$ there is an edge, congested with $\Omega(k)$.*

Proof. Clearly, we obtain a lower bound on maximal congestion on an edge, if we give a lower bound on congestion on edge (u_1, v_1). We give the following view for determining congestion on (u_1, v_1).

We construct a bipartite graph $G' = (S \cup R, E')$, in which the node partitions are given by senders and receivers. Graph G' contains edge (s_i, r_j) if and only if the randomly

chosen path from s_i to r_j crosses edge (u_1, v_1) in G. This means G' contains each edge independently with probability $1/k$. Maximal congestion on (u_1, v_1) (w.r.t. graph G) then equals the cardinality of the maximal matching in G'.

We give a lower bound to the cardinality of a maximal matching, by choosing an arbitrary matching $\mathcal{M}$ that is large enough for our purposes with high probability. We start with the empty matching $\mathcal{M} := \emptyset$. Now we consider all senders s_i for $i = 1, ..., k/2$ and check, whether an edge $(s_i, *)$ exists, that is not adjacent to an edge in $\mathcal{M}$ and in the positive case, we add this edge to $\mathcal{M}$. We call receivers, that are not incident to any edge in $\mathcal{M}$ *free* receivers. In the course of time the number of free receivers decreases, but since we stop our procedure if we reach sender $s_{k/2}$, we have $|\mathcal{M}| \leq k/2$ and the number of free receivers is at least $k/2$ at any time. For simplicity we modify our procedure, such that only edges incident to the first $k/2$ free receivers are accepted. Then in each step, we have *exactly* $k/2$ free receivers. Clearly, edge set $\mathcal{M}$ remains a matching, since we never add an edge, which is violating this property. Let $X_i : \Omega \to \{0, 1\}$ denote a random variable, that is defined as follows

$$X_i := \begin{cases} 1 \text{ if } (s_i, *) \in \mathcal{M} \\ 0 \text{ otherwise} \end{cases}$$

Consider a sender s_i with $i \in \{1, ..., k/2\}$. There are $k/2$ possible edges starting at s_i and each edge exists with probability $1/k$. Thus we can bound the probability, that we are able to add an edge to our matching in step i by

$$\Pr(X_i = 1) \geq 1 - \left(1 - \frac{1}{k}\right)^{k/2} = 1 - \left(\left(1 - \frac{1}{k}\right)^{k}\right)^{1/2} \overset{(1)}{\geq} 1 - e^{-1/2} \overset{(2)}{\geq} \frac{1}{3}$$

Inequality (1) holds, because of $(1-\frac{1}{k})^k \leq \frac{1}{e}$ and (2) follows from $1-e^{-1/2} \approx 0.3935.. > \frac{1}{3}$. If we define $X := X_1 + ... + X_{k/2}$ this variable gives the cardinality of matching $\mathcal{M}$. Thus the expected size of a maximal matching in G' must be at least

$$E[X] = E[X_1 + ... + X_{k/2}] = E[X_1] + ... + E[X_{k/2}] \geq \underbrace{\frac{1}{3} + ... + \frac{1}{3}}_{k/2 \text{ summands}} = \frac{k}{6}$$

Here, second equality holds because of linearity of expectation and the inequality follows from

$$E[X_i] = 0 \cdot \Pr(X_i = 0) + 1 \cdot \Pr(X_i = 1) = \Pr(X_i = 1) \geq \frac{1}{3}$$

Since randomized rounding chooses paths in G independently, distribution of edges in the bipartite graph G' is also independent. Random variable X is a sum of independently 0/1-distributed random variables X_i, thus we can apply Chernoff bounds. Here, we will use the following Chernoff bound

Theorem 6.8. *(Chernoff bound) Let $Y_1, ..., Y_m$ be independently 0/1-distributed random variables. Then, for $Y := Y_1 + ... + Y_m$ and $0 < \delta < 1$*

$$\Pr(Y \leq (1 - \delta)E[Y]) \leq e^{-E[Y]\delta^2/2}$$

holds.

Proof. See [19, 14]. □

Now we can accomplish proof of theorem 6.7 by applying this Chernoff bound to cardinality X of matching $\mathcal{M}$

$$\Pr(X \leq k/12) = \Pr(X \leq (1-1/2)E[X]) \leq e^{-\frac{k}{6}\frac{1}{4}\frac{1}{2}} = e^{-k/48} = \left(\frac{1}{2}\right)^{\Omega(k)}$$

We conclude that with a probability exponential close to 1, congestion on edge (u_1, v_1) in the rounded solution is at least $k/12 = \Omega(k)$, and thus the claim follows. □

Because of $OPT = 1$ and $n = 4k$, randomized rounding leads to an approximation ratio of $\Omega(n)$, which is asymptotically the worst possible value, because the value of each $\text{VPND}_{\text{MinCon}}$ problem can be bounded by $\min\{|S|, |R|\} \leq |V| = n$.

Some approximation algorithms that use randomized rounding exploit properties, that so called base solutions of linear programs possess [18]. Unfortunately the solution given in this section is *not* such a base solution, which means that the solution does not define a vertex of the polyhedron, that is induced by the LP. We can show this, by expressing our solution as a convex combination of other feasible solutions. For this purpose we consider all $n!$ possible permutations on $\{1, ..., k\}$. For each permutation $\sigma \in S_k$, we define a VPND solution consisting of paths $\mathcal{P}_{ij}^{\sigma} = \{s_i \to u_{\sigma(i)} \to v_{\sigma(i)} \to r_j\}$. Thus, for each sender-receiver pair, we choose one path with weight 1. Since all paths of a sender s_i cross the same edge $(u_{\sigma(i)}, v_{\sigma(i)})$, the value of these solutions is always 1. The mean of the solutions, defined by paths $\mathcal{P}_{ij}^{\sigma} = \{s_i \to u_{\sigma(i)} \to v_{\sigma(i)} \to r_j\}$, is a fractional solution, which contains a path $s_i \to u_l \to v_l \to r_j$ with weight $1/k$ for all $i, j, l \in \{1, ..., n\}$. So we obtain the solution consisting of paths $\mathcal{P}_{ij}$ from above.

Nevertheless we should not conclude that $\text{VPND}_{\text{MinCon}}$ is hard to approximate, only because of the fact that randomized rounding results in a catastrophe, since the same fact holds for VPND itself. To show this, we install costs of 1 on edges (u_i, v_i), while other edges are for free and use the same fractional paths $\mathcal{P}_{ij}$ with weight $1/k$ for each one. Then, randomized rounding will lead to solutions that need capacity of $\Omega(k)$ on each edge (u_i, v_i) with high probability. Hence such a rounded solution would cost $\Omega(k^2)$, which is extremely expensive, compared to an optimal solution with costs of n. However there are algorithms for VPND, that have a constant approximation ratio. But in contrast to $\text{VPND}_{\text{MinCon}}$ no instances are known to the author, that would reveal a non-constant integrality gap for VPND.

One remaining possibility to deal with $\text{VPND}_{\text{MinCon}}$ is to use iterated randomized rounding. This means, after determining an optimal fractional solution, we would choose a path according to the computed probabilities. After this, we could reoptimize the LP while setting the variables belonging to the chosen path constant. Then we would repeat this procedure, until all variables are constant. Clearly, this procedure can not give better approximation ratios than $O(\sqrt{n}/\log n)$.

7 Open questions

We have been able to answer some questions during this work, but there are many unsolved problems left. In this section we want to present open questions, that are worth to think about.

There are polynomial time algorithms for Steiner tree and $\text{VPND}_{\text{Tree}}$ in the case of a constant number of terminals. It is not known whether such algorithms exist for VPND. We also have not been able to proof $\mathcal{NP}$-hardness of VPND in the interesting case $|R| = |S|$.

An unsatisfying fact is, that we could only give an indication that $\text{VPND}_{\text{MinCon}}$ is hard to approximate, by showing a high integrality gap, but a formal proof is still absent.

A further point of interest is the following. Let OPT be the optimal value of a VPND instance. We denote OPT_{Tree} to be the optimal objective function value of $\text{VPND}_{\text{Tree}}$ for the same instance. Clearly we have $OPT \leq OPT_{\text{Tree}}$, but the critical question is: By which factor can OPT_{Tree} be larger than OPT? The best known VPND algorithm, that also gives a tree solution, has a ratio of 4.74 [4]. Thus we have $OPT_{\text{Tree}} \leq 4.74 \cdot OPT$. But we do not know an example, in which relation OPT_{Tree}/OPT reaches a value close to 4.74. It is an author's conjecture, that at least $OPT_{\text{Tree}}/OPT \leq 2$ holds. Unfortunately we haven't been able to proof this. But possibly the reader is more successful in proving this assumption...

Bibliography

[1] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2,. *Inf. Process. Lett.*, 32(4):171–176, 1989.

[2] A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Sciences*, 30, 1985.

[3] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[4] F. Eisenbrand and F. Grandoni. An improved approximation algorithm for virtual private network design. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 05)*, volume 16, pages 928–932, Vancouver, Canada, 2005. SIAM.

[5] F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella. New approaches for virtual private network designs. In *Annual International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1151–1162, Lisboa, Portugal, 2005. Springer.

[6] M. Grötschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[7] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 389–398, New York, USA, 2001. ACM Press.

[8] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *ACM Symposium on Theory of Computing (STOC)*, 2003.

[9] C. Hurkens, J. Keijsper, and L. Stougie. Virtual private network design: A proof of the tree routing conjecture on ring networks. In *11th Integer Programming and Combinatorial Optimization Conference*, 2005.

[10] G. Italiano, S. Leonardi, and G. Oriolo. Design of networks in the hose model. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 65–76, 2002.

[11] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. In *Theory of Computing Systems, Issue 1*, volume 24, pages 223 – 232, 1991.

[12] D. R. Karger and M. Minkoff. Building steiner trees with incomplete global knowledge. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 613, Washington, USA, 2000. IEEE Computer Society.

[13] B. Korte and J. Vygen. *Combinatorial Optimization - Theory and Algorithms*. Springer-Verlag, Second Edition, 2002.

[14] M. Mitzenmacher and E. Upfal. *Probability and computing*. Cambridge, 2005.

[15] P. Raghavan and C. D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[16] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. *Proc. of ACM/SIAM Symposium on Discrete Algorithms (SODA)*, 2000.

[17] C. Swamy and A. Kumar. Prima-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[18] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[19] I. Wegener. *Komplexitätstheorie, Grenzen der Effizienz von Algorithmen*. Springer, 2003.