Alcom-FT Technical Report Series
ALCOMFT-TR-03-103

Sharing the cost more efficiently: Improved Approximation for Multicommodity Rent-or-Buy

L. Becchetti* J. Könemann† S. Leonardi*

November 26, 2003

Abstract

In the *multicommodity rent-or-buy* (MROB) network design problem we are given a network together with a set of k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$. The goal is to provision the network so that a given amount of flow can be shipped between s_i and t_i for all $1 \leq i \leq k$ simultaneously. In order to provision the network one can either *rent* capacity on edges at some cost per unit of flow, or *buy* them at some larger fixed cost. Bought edges have no incremental, flow-dependent cost. The overall objective is to minimize the total provisioning cost.

Recently, Gupta et al. [7] presented a 12-approximation for the MROB problem. Their algorithm chooses a subset of the terminal pairs in the graph at random and then buys the edges of an approximate Steiner forest for these pairs. This technique has previously been successfully used by the same authors for the single sink rent-or-buy network design problem.

In this paper we give a 5.5-approximation for the MROB problem by refining the algorithm of Gupta et al. and greatly simplifying their analysis. The improvement in our paper is based on a more careful adaptation and simplified analysis of the primal-dual algorithm for the Steiner forest problem due to Agrawal, Klein and Ravi [1]. Our result significantly reduces the gap between the single-sink [8] and multi-sink cases.

*Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy. Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT), IST-2000-14084 (APPOL), IST-2001-33555 (COSIN) and by the MIUR Project "Resource Allocation in Real Networks". Email: {becchett, leon}@dis.uniroma1.it.

†Department of Combinatorics and Optimization, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada. Email: jochen@uwaterloo.ca. This work was done while being on leave at the Dipartimento di Informatica e Sistemistica at Università di Roma "La Sapienza", Italy.

1 Introduction

In the *multi-commodity rent-or-buy problem* (MROB) we are given an undirected graph $G = (V, E)$, terminal pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\}$, non-negative costs c_e for all edges $e \in E$, and a parameter $M \geq 0$. In the *multicommodity rent-or-buy* problem the goal is to select a set of *bought* edges F_b and a set of of *rented* edges F_r, respectively, such that $F_b \cup F_r$ contains a path from s to t for all $(s, t) \in R$. The cost of a bought edge $e \in F_b$ is $M \cdot c_e$. A rented edge $e \in F_r$ costs $c_e \cdot \lambda(F, e)$ where $\lambda(F, e)$ denotes the total flow traversing edge e. The aim is to find a feasible solution of smallest total cost.

The MROB problem generalizes the *single-commodity rent-or-buy* problem (SROB). Here we are again given an undirected network together with rental and buying costs on all edges $e \in E$ as before. We are also given a set of terminal nodes and a root node r. The goal is now to provision the network such that all terminals can send a specified amount of flow to the root node r simultaneously. A recent result of Gupta et al. [8] gives a 3.55 approximation algorithm for the problem.

Awerbuch and Azar [2] and Bartal [3] were the first to give an $O(\log |V| \log \log |V|)$-approximation algorithm for the MROB problem. Later, Kumar, Gupta and Roughgarden [12] give the first constant approximation algorithm for the problem based on a primal-dual approach. A more recent result by Kumar et al. [7] builds on the techniques used by Gupta et al. [8] for the single-commodity rent-or-buy problem and obtains a 12-approximation for the MROB problem.[1] Their work also uses the *cost-sharing* concept from game-theory (see, e.g., [4, 9, 14]) in the analysis of the algorithm.

The minimum-cost Steiner tree and forest problems are closely related to both the MROB and SROB problems. In the more general Steiner forest problem, we are given an undirected graph $G = (V, E)$, non-negative costs c_e for all edges $e \in E$, and a set of terminal pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. The goal is to find a forest F of minimum total cost such for all $1 \leq i \leq k$, there is a tree $T \in F$ that contains both, s_i and t_i. It is well-known that the minimum-cost Steiner forest problem is NP-hard[5] and Max-SNP hard. On the positive side, Agrawal, Klein and Ravi [1] and later Goemans and Williamson [6] give a primal-dual 2-approximation for the problem.

The MROB algorithm from [7] crucially relies on the primal-dual algorithm for Steiner forest of Agrawal et al. [1]. The algorithm in [7] first picks a random subset of all terminal pairs $R^0 \subseteq R$ and then uses a modified primal-dual Steiner forest algorithm to compute a feasible Steiner forest F^0 for R^0. The algorithm buys all edges from F^0. Terminal pairs in $R \setminus R^0$ that are not connected in F^0 rent extra capacity in the cheapest possible way to establish connections.

The central feature of the modified primal-dual Steiner forest algorithm used in [7] is *β-strictness*: The algorithm defines *cost-shares* χ_{st} for all terminal pairs (s, t) in R. Let the Steiner forest computed by the algorithm on input $R \setminus \{(s, t)\}$ be denoted by F' and let $G|F'$ be the graph obtained from G by contracting F'. The algorithm then guarantees that the cheapest way of connecting s to t in $G|F'$ costs at most $\beta \cdot \chi_{st}$. Moreover, the sum over all cost-shares of terminal pairs is at most the cost of a minimum-cost Steiner forest for R.

[1] In a personal communication with the authors [13], M. Pál claims that a refined analysis yields an improved approximation ratio of 8 for the MROB-algorithm in [7].

The *prize* for a β-strict Steiner forest algorithm is a worse performance guarantee. Gupta et. al show that their Steiner forest modification returns a 6-approximate and 6-strict Steiner forest and this leads to a 12-approximate MROB-algorithm. In general, they show that any α-approximate and β-strict algorithm leads to an $(\alpha + \beta)$-approximation for the MROB problem.

Our Contribution. Our algorithm uses the cost-sharing framework proposed by Gupta et al. In particular, we also modify the primal-dual Steiner forest algorithm by Agrawal et al. [1]. We prove the following main result:

Theorem 1 *For any* $1 < \beta \leq 3/2$ *there is a polynomial-time* $(2 + 1/(\beta - 1))$*-approximate and* β*-strict algorithm for the minimum-cost Steiner forest problem.*

Choosing $\beta = 3/2$ in Theorem 1 together with [7] yields:

Corollary 1 *There is a* 5.5*-approximate algorithm for the multicommodity rent-or-buy problem.*

Our Steiner forest algorithm has two main phases: The first phase runs the standard primal-dual Steiner forest algorithm from [1] and computes an approximate Steiner forest $\overline{F}$ for a given set of terminal pairs R.

The second phase treats each tree T in $\overline{F}$ as a terminal of a new Steiner tree instance, with a suitably chosen budget $\texttt{budget}_T$. We then run a *budgeted* version of the primal-dual algorithm for Steiner trees to obtain a final Steiner forest. Here, we borrow ideas from earlier work on *price-collecting* variants of the Steiner tree problem (see, e.g., [6]).

The benefit of our method is two-fold: First, we combine existing primal-dual algorithms in a black-box fashion as opposed to modifying technical details of an existing method. This leads to a much simplified algorithm and analysis. Second, since we use standard primal-dual algorithms for the Steiner forest and Steiner tree problems we inherit some their nice properties. Most notably, our dual solutions are laminar.

Organization of this paper. The next section recaps the primal-dual Steiner forest algorithm from [1] since our methods and its analysis strongly relies on it. The simplicity of our algorithm and its analysis depend crucially on a view of the primal-dual Steiner forest algorithm that differs significantly from the one taken in [1]. We introduce this view in Section 2. Subsequently we present our strict Steiner forest algorithm together with its analysis and the proof of Theorem 1.

2 The Minimum-cost Steiner forest problem

In this section we present the primal-dual algorithm for the Steiner forest problem due to Agrawal, Klein, and Ravi (in what follows we refer to it as `AKR`). The presentation given in this Section differs slightly from the one given in [1] and simplifies subsequent parts of this paper.

`AKR` is a *primal-dual* algorithm. This means that the algorithm constructs both a feasible primal and a feasible dual solution for a linear programming formulation of the Steiner forest problem and its dual, respectively. A standard integer programming formulation for the Steiner forest problem has a binary variable x_e for all edges $e \in E$. Variable x_e has value 1 if edge e is part of the resulting forest. We let $\mathcal{U}$ contain exactly those subsets U of V that *separate* at least one terminal pair in R. In other words, $U \in \mathcal{U}$ iff there is $(s,t) \in R$ with $|\{s,t\} \cap R| = 1$. For a subset U of the nodes we also let $\delta(U)$ denote the set of those edges that have exactly one endpoint in U. We then obtain the following integer linear programming formulation for the Steiner forest problem:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \cdot x_e && \text{(IP)} \\ \text{s.t} \quad & \sum_{e \in \delta(U)} x_e \geq 1 \quad \forall U \in \mathcal{U} \\ & x \text{ integer} \end{aligned}$$

The dual of the linear programming relaxation (LP) of (IP) has a variable for y_U for all $U \in \mathcal{U}$.

$$\begin{aligned} \max \quad & \sum_{U \in \mathcal{U}} y_U && \text{(D}^*\text{)} \\ \text{s.t} \quad & \sum_{U : e \in \delta(U)} y_U \leq c_e \quad \forall e \in E \\ & y \geq 0 \end{aligned}$$

For a pair of nodes $u, v \in R$, let c_{uv} the minimum cost of any u,v-path in G. It can then be shown (see [10, 11]) that the following linear program is equivalent to (D*):

$$\begin{aligned} \max \quad & \sum_{U \subseteq R} y_U && \text{(D)} \\ \text{s.t} \quad & \sum_{U \subseteq R : |\{u,v\} \cap U| = 1} y_U \leq c_{uv} \quad \forall u, v \in R && (1) \\ & y \geq 0 \end{aligned}$$

In our presentation, we let `AKR` construct a primal solution for (LP) and a dual solution for (D).

The algorithm now starts with $x_e = 0$ for all $e \in E$ and $y_U = 0$ for all $U \subseteq R$ that separate at least one terminal pair. The algorithm has two goals:

1. Compute a feasible solution for the given Steiner forest instance. The algorithm reduces the degree of infeasibility as it progresses.
2. Create a dual feasible packing of sets of largest possible total value. The algorithm raises dual variables of certain subsets of terminals at all times. The final dual solution is going to be maximal in the sense that no single set can be raised without violating a constraint of type (1).

We think of an execution of Algorithm `AKR` as a process over time and let x^t and y^t be the primal incidence vector and dual feasible solution at time t. We also use F^t to denote the forest corresponding to x^t.

Assume that the forest F^t at time t is infeasible. We then determine a set of *minimal* subsets of R whose duals we raise simultaneously until a constraint of type (1) becomes tight. For a connected component C of F^t, we use $C|R$ to denote the set of terminal nodes in C. We then raise the dual variable corresponding to $C|R$ iff C is a connected component of F^t that separates at least one terminal pair of R. We call such connected components *active*.

At time t, `AKR` raises the dual variables of all active sets simultaneously until a constraint of type (1) is satisfied with equality. Suppose that this happens for terminals $u, v \in R$ and also assume that $u \in C_u$ and $v \in C_v$ for connected components C_u and C_v of F^t. We then add a u, v-path of smallest total cost to F^t and continue.

Merging two components by adding a connecting path to F^t yields a new component that may be *inactive*, i.e. the newly created component $C_u \cup C_v$ may not separate any terminal pairs. The dual variables corresponding to inactive sets are not raised by `AKR`.

The following theorem is proved in the Appendix:

Theorem 2 (See [1]) *Suppose that algorithm* `AKR` *stops at time* t^**. We then must have that*

$$c(F^{t^*}) \leq 2 \cdot \sum_{U \subseteq R} y_U^{t^*}.$$

3 A strict algorithm for minimum-cost Steiner forests

This section is split into three major parts. First, we recall from [7] how to compute the cost shares for each terminal pair $(s, t) \in R$. Subsequently we give our $(2 + 1/(\beta - 1))$-approximate and β-strict algorithm for Steiner forests. The section ends with the strictness-analysis of the algorithm.

3.1 Computing cost-shares

We start by giving a precise definition of the strictness notion. For a forest F in G, let $G|F$ denote the graph resulting from contracting all trees of F. For vertices $u, v \in V$, we also let $c_G(u, v)$ denote the minimum-cost of any u, v-path in G. In [7], Gupta et al. define the notion of *β-strict* algorithms for the minimum-cost Steiner forest problem.

Definition 1 *An algorithm $\mathcal{A}$ for the Steiner forest problem is β-strict if it returns values χ_i for all $(s_i, t_i) \in R$ such that*

1. $\sum_{(s_i,t_i) \in R} \chi_i \leq c(F^*)$ *where F^* is a feasible Steiner forest for R of minimum total cost, and*
2. $c_{G|F_i}(s_i, t_i) \leq \beta \cdot \chi_i$ *for all $(s_i, t_i) \in R$ where F_i is a Steiner forest for terminal pairs $R \setminus \{(s_i, t_i)\}$ returned by $\mathcal{A}$.*

The algorithm to compute the cost shares χ_i for all terminal pairs $(s_i, t_i) \in R$ is the same as in [7]. We run `AKR` on input graph G with terminal pairs R. Let $\mathtt{age}_i$ be the time at which s_i and

t_i meet during this execution. We say that that s_i (t_i) is *alone* at time $t \leq \texttt{age}_i$ if s_i (t_i) is the only active terminal in its component at that time during AKR's execution. We then define an indicator variable δ_t^i for all terminal pairs (s_i, t_i) and for all times $t \geq 0$:

$$\delta_t^i = \begin{cases} 2 & : \text{ Both, } s_i \text{ and } t_i \text{ are alone at time } t < \texttt{age}_i \\ 1 & : \text{ Exactly one of } s_i \text{ and } t_i \text{ is alone at time } t < \texttt{age}_i \\ 0 & : \ t \geq \texttt{age}_i. \end{cases}$$

The cost-share of terminal pair (s_i, t_i) is defined as

$$\chi_i = \int_0^{t^*} \delta_t^i \, dt. \tag{2}$$

3.2 Adding strictness: A modified Steiner forest algorithm

Fix a terminal pair $(s, t) \in R$ and let $R^0 = R \setminus \{(s, t)\}$. The new algorithm $\texttt{AKR}_2$ first uses AKR to compute a feasible Steiner forest $\overline{F}$ for terminal set R^0. In a second phase the algorithm adds more paths to connect components of $\overline{F}$ that are close together. Selecting paths carefully in this second phase yields a Steiner forest F^0 whose cost is only a constant factor worse than that of $\overline{F}$ and that satisfies the necessary strictness properties.

We now describe the algorithm $\texttt{AKR}_2$ in greater detail. The algorithm works on input R^0 and has two phases:

[**Aerobic Phase**] In this phase we execute AKR on terminal set R^0. This produces a forest $\overline{F}$ that is feasible for R^0 and a corresponding dual solution $\{\overline{y}_U\}_{U \subseteq R^0}$.

[**Anaerobic Phase**] We define a *budget* $\texttt{budget}_C$ for each connected component C of $\overline{F}$ as follows:

$$\texttt{budget}_C = \frac{1}{2(\beta - 1)} \cdot \sum_{U \subseteq C | R^0} \overline{y}_U. \tag{3}$$

We now raise the dual variables of all connected components in the current forest that have positive remaining budget. We decrease the budget of these components at the rate at which their duals grow. Two possible events can occur:

Merge A path connecting two connected components C_1 and C_2 in the current forest becomes tight. In this case, add the edges of the path to the current forest and by this create a new connected component C. The budget of this new component C is the sum of the budgets of C_1 and C_2.

Death A connected component runs out of budget in the growth phase. In this case the component simply dies and we continue growing those components that have positive remaining budget.

We let the final forest computed by $\texttt{AKR}_2$ be F^0 and we denote the corresponding dual solution by $\{y_U^0\}_{U \subseteq R^0}$. Notice that y^0 may not be feasible since, in the anaerobic phase of $\texttt{AKR}_2$, the algorithm raises duals of components that do not separate terminal pairs. In other words, there are no variables corresponding to the terminal sets raised in the dual LP (D). Nevertheless we can show the following lemma.

Lemma 1 *Let F^0 be the final forest produced by* $\texttt{AKR}_2$ *on input R^0 and let $\{y_U^0\}_{U \subseteq R^0}$ be the corresponding (infeasible) dual vector. We then must have that*

$$c(F^0) \leq 2 \cdot \sum_{U \subseteq R^0} y_U^0.$$

Proof: The proof is a replica of the one of Theorem 2. ■

This lemma helps us to bound the cost of the final forest F^{t^*}.

Corollary 2 *The cost of the forest F^0 computed by* $\texttt{AKR}_2$ *on terminal set R^0 is at most*

$$\left(2 + \frac{1}{\beta - 1}\right) \cdot \texttt{opt}_{R^0}$$

where $\texttt{opt}_{R^0}$ *is the cost of a minimum-cost feasible Steiner forest for terminal set R^0.*

Proof: It follows from the description of Algorithm $\texttt{AKR}_2$ and Lemma 1 that

$$c(F^0) \leq 2 \cdot \sum_{U \subseteq R^0} y_U^0 = 2\left(1 + \frac{1}{2(\beta - 1)}\right) \cdot \sum_{U \subseteq R^0} \overline{y}_U = \left(2 + \frac{1}{\beta - 1}\right) \cdot \sum_{U \subseteq R^0} \overline{y}_U.$$

Since $\overline{y}$ is feasible for (D) we can apply weak duality and the corollary follows. ■

3.3 Analyzing the strictness of Algorithm $\texttt{AKR}_2$

We focus on terminal pair $(s,t) \in R$. Recall that $R^0 = R \setminus \{(s,t)\}$ and let F denote the forest computed by $\texttt{AKR}$ on input R. As before we let F^0 be the forest computed by $\texttt{AKR}_2$ on input R^0. As in Section 1 we let graph $G|F^0$ be the graph obtained from G by contracting the connected components of forest F^0. In order to prove that $\texttt{AKR}_2$ is β-strict we need to show that

$$c_{G|F^0}(s,t) \leq \beta \cdot \chi_{st}. \tag{4}$$

The rough outline is as follows: We let P be the unique s,t-path in forest F and we use P^0 to denote the minimum-cost s,t-path in $G|F^0$. Clearly, we must have that the cost of P in $G|F^0$ is at least as high as the cost of P^0 in $G|F^0$. For us it will be easier to argue about P since P is closely related to the cost-share χ_{st} of terminal pair (s,t). In fact, we will show that

$$c_{G|F^0}(P) \leq \beta \cdot \chi_{st} \tag{5}$$

where $c_{G|F^0}(P)$ is the cost of path P in the graph $G|F^0$. (5) implies (4) since $c_{G|F^0}(P^0) \leq c_{G|F^0}(P)$.

3.3.1 Path P and a notion of time

We now introduce a notion of time: For each point p on path P, we let $t(p)$ be the first point in time during the execution of `AKR` on terminal set R at which either the s,p-segment or the t,p-segment of P is tight. Then, there is a point p_m on P where two active components, one containing s and another one containing t, meet at some point during the execution of `AKR` on terminal set R. The time $t(p_m)$ at which point p_m is reached equals the time $\mathtt{age}_{st}$ at which terminals s and t become inactive.

W.l.o.g., we can assume that p_m is a vertex of G on P (if necessary we can create an artificial vertex). Point p_m then splits P into two parts P^s and P^t where the points on P^s are reached first by s and the points on P^t are first reached by t during the execution of `AKR`.

We assume that P^s is an s,p_m-path that passes through components $C_1^s,\dots,C_p^s$ of forest F^0 and let $s \in C_1^s$ and $p_m \in C_p^s$. Observe that components C_s^1 and C_p^s are potentially singleton sets! Similarly, we let P^t be a t,p_m-path that meets components $C_1^t,\dots,C_q^t$ where $t \in C_1^t$ and $p_m \in C_q^t$. We denote the segment of P^v between C_i^v and C_{i+1}^v by P_i^v for $v \in \{s,t\}$.

The idea of the following analysis is to show that we can express the cost-share for (s,t) as

$$\chi_{st} = \sum_{i=0}^{p} \chi_{st}^s(i) + \sum_{i=0}^{q} \chi_{st}^t(i)$$

and that $\chi_{st}^v(i)$ *recovers* a constant fraction of the cost of path P_i^v for $v \in \{s,t\}$ and for all i. In other words, we want to show that

$$\chi_{st}^v(i) \geq \frac{1}{\beta} \cdot c_{G|F^0}(P_i^v) \tag{6}$$

for all $v \in \{s,t\}$ and for all i.

3.3.2 Lower-bounding χ_{st}

The goal in this section is to prove (6). We concentrate on the case $v = s$ and the proof for $v = t$ follows via a symmetric argument. In the following let y be the dual computed by `AKR`(R). For each $0 \leq i \leq p$, we then let y^i be the dual load on P_i^s induced by y, i.e.

$$y^i = \sum_{U \subseteq R : \delta(U) \cap P_i^s \neq \emptyset} y_U.$$

For each i, the dual load on P_i^s can be split into three parts:

[$\mathcal{U}_1^i$] load by active sets that do not contain s or t,

[$\mathcal{U}_2^i$] load by active sets that contain s or t that also separate other terminal pairs, and

[$\mathcal{U}_3^i$] load induced by sets that contain s or t as the only active terminal.

The following lemma is a useful property whose proof is given in the appendix.

Lemma 2 *Let $U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i \cup \mathcal{U}_3^i$. We then must have that $|\delta(U) \cap P_i^s| = 1$.*

Lemma 2 yields another very useful observation whose proof is again deferred to the appendix.

Lemma 3 *Let U be a set in $\mathcal{U}_1^i$. Suppose that $\mathtt{AKR}(R^0)$ grows U during time interval $[t_1, t_2]$. Then there must exist a set U_m that is grown by $\mathtt{AKR}(R^0)$ during the same time interval and that is merged with U at time t_2. Moreover, we must have $U_m \notin \bigcup_{j=1}^3 \mathcal{U}_j^i$.*

Notice that path P_i^s must be tight in $\mathtt{AKR}(R)$ at the time it is added. Together with Lemma 2 we therefore obtain

$$c(P_i^s) = \sum_{U \subseteq R : \delta(U) \cap P_i^s \neq \emptyset} y_U \tag{7}$$

Using this and the definition of χ_{st} in (2) it follows that it is enough to show that

$$y_3^i := \sum_{U \in \mathcal{U}_3^i} y_U \geq \frac{1}{\beta} \cdot c_{G|F^0}(P_i^s) = \frac{1}{\beta} \cdot \sum_{U \in \bigcup_{j=1}^3 \mathcal{U}_j^i} y_U$$

for all $0 \leq i \leq p$ in order to show β-strictness. Simplifying yields that it is enough to show

$$y_3^i \geq \frac{1}{\beta - 1} \cdot \sum_{U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i} y_U. \tag{8}$$

for all $0 \leq i \leq p$.

The basic idea of the proof is to show that C_i^s and C_{i+1}^s are linked by a path in F^0 if (8) is not satisfied. More precisely we can show that, in this case, $\mathtt{AKR}_2$ possesses enough extra budget to *close the gap* between C_i^s and C_{i+1}^s. In order to make this idea more precise we need to compare the dual load that P_i^s sees in runs $\mathtt{AKR}(R)$ and $\mathtt{AKR}_2(R^0)$ from sets in $\mathcal{U}_1^i \cup \mathcal{U}_2^i$. In particular, we prove the following lemma.

Lemma 4 *The dual load on path P_i^s inflicted by $\mathtt{AKR}_2(R^0)$ is at least*

$$\left(\frac{\beta}{\beta - 1}\right) \cdot \sum_{U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i} y_U.$$

Once this is proved we obtain the following immediate corollary:

Corollary 3 *If (8) is not satisfied then path P_i^s must be tight in $\mathtt{AKR}_2(R^0)$.*

Proof: The fact that (8) does not hold together with (7) implies

$$\sum_{U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i} y_U = c(P_i^s) - y_3^i > c(P_i^s) - \frac{1}{\beta - 1} \cdot \sum_{U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i} y_U$$

and hence

$$\sum_{U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i} y_U > \frac{\beta - 1}{\beta} \cdot c(P_i^s).$$

Lemma 4 then implies that the dual load felt by P_i^s in $\mathtt{AKR}_2$ is more than $c(P_i^s)$. ■

Pending the proof of Lemma 4, Corollaries 2 and 3 complete the proof Theorem 1.

3.3.3 Proof of Lemma 4

We proceed with the proof of Lemma 4. The proof is subdivided into two major parts corresponding to sets in $\mathcal{U}_1^i$ and $\mathcal{U}_2^i$. In $\mathtt{AKR}_2$, we then distinguish between dual load on P_i^s coming from sets in $\mathcal{U}_1^i$ and $\mathcal{U}_2^i$.

Sets in $\mathcal{U}_1^i$: An easy first observation is that the dual y computed by $\mathtt{AKR}(R)$ is *laminar*: Let U_1, U_2 be two subsets of R with $y_{U_1} > 0$ and $y_{U_2} > 0$. We then must have that either $U_1 \subseteq U_2$, $U_2 \subseteq U_1$, or $U_1 \cap U_2 = \emptyset$.

We say that a set $U \in \mathcal{U}_1^i \cup \mathcal{U}_2^i$ is *outermost* if there is no set $U' \in \mathcal{U}_1^i \cup \mathcal{U}_2^i$ such that $U \subseteq U'$. Let U be an outermost set in $\mathcal{U}_1^i$. Since $\{s,t\} \cap U = \emptyset$ it follows that U and the forest inside U develops in the same way in $\mathtt{AKR}(R^0)$ and $\mathtt{AKR}(R)$. Lemma 2 implies that the dual load by U and its subsets on P_i^s at the end of the aerobic phase of $\mathtt{AKR}_2$ is exactly

$$\sum_{U' \subseteq U, U \in \mathcal{U}_1^i} y_{U'}.$$

Lemma 3 implies that the total budget that is available to the connected component corresponding to U is at least

$$2 \cdot \frac{1}{2(\beta-1)} \cdot \sum_{U' \subseteq U, U \in \mathcal{U}_1^i} y_{U'} = \frac{1}{\beta-1} \cdot \sum_{U' \subseteq U, U \in \mathcal{U}_1^i} y_{U'}.$$

This is a lower-bound on the additional load that U inflicts on P_i^s in the anaerobic phase of $\mathtt{AKR}_2$. Summing over all outermost sets $U \in \mathcal{U}_1^i$ finishes the proof for sets in $\mathcal{U}_1^i$.

Sets in $\mathcal{U}_2^i$: Assume that $\mathcal{U}_2^i$ contains l sets $U_1, \ldots, U_l$. By definition we know that for all $1 \leq j \leq l$ the set U_j separates at least one terminal pair $(s', t') \in R^0$. Let (s_j, t_j) be such a terminal pair of minimum age, i.e.

$$\mathtt{age}_{s_j,t_j} = \min_{(s',t') \in R^0, |\{s',t'\} \cap U| = 1} \mathtt{age}_{s',t'}.$$

We can now assume, w.l.o.g., that $\mathtt{age}_1 \leq \mathtt{age}_2 \leq \ldots \leq \mathtt{age}_l$ where we use $\mathtt{age}_j$ as a short for $\mathtt{age}_{s_j,t_j}$. We must also have $U_1 \subseteq U_2 \subseteq \ldots \subseteq U_l$, since all sets U_j contain s. Also observe that U_1 is the first set that loads P_i^s in $\mathtt{AKR}(R)$. Sets $U_j \in \mathcal{U}_2^i$ load P_i^s while they are active in $\mathtt{AKR}(R)$.

Let $\mathtt{age}_j^0$ be the time at which terminals s_j and t_j get connected in $\mathtt{AKR}(R^0)$. It is then not hard to see that

$$\mathtt{age}_j \leq \mathtt{age}_j^0 \tag{9}$$

for all $1 \leq j \leq l$.

We assume, w.l.o.g., that $s_j \in U_j$ for all $1 \leq j \leq l$. Also let p_j be the closest and let $\overline{p}_j$ be the farthest point from s on P_i^s, respectively, that feels dual load by U_j in $\mathtt{AKR}(R)$.

Consider terminal pair (s_j, t_j). Observation (9) implies that (s_j, t_j) is active during time interval $I_j := [t(p_j), t(\overline{p}_j)]$ in the execution of $\mathtt{AKR}_2(R^0)$. The dual variable of the connected component containing s_j in $\mathtt{AKR}_2(R^0)$ during interval I_j is raised by

$$y_{U_j} = (t(\overline{p}_j) - t(p_j)) \tag{10}$$

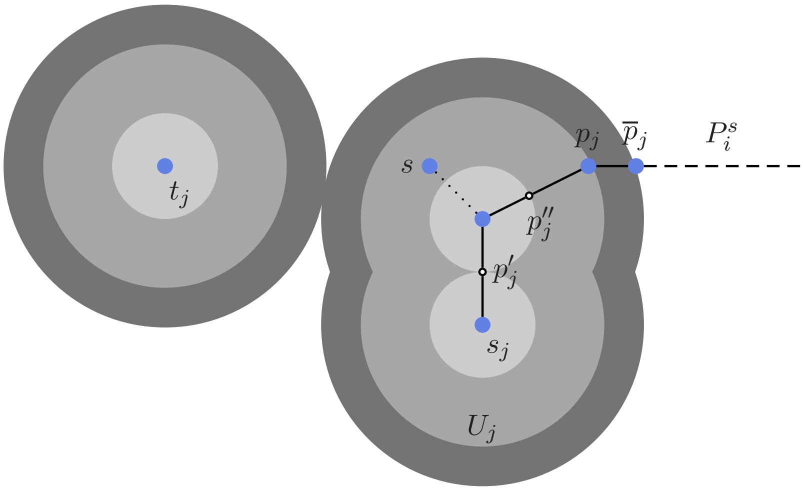

Figure 1: The figure shows a snapshot of $\texttt{AKR}(R)$ at the time when U_j starts loading P_i^s. Darker colors symbolize later points in time during the execution of $\texttt{AKR}$.

In addition, the connected components containing s_j and t_j, respectively, collect a total of

$$2 \cdot \frac{1}{2(\beta-1)} \cdot (t(\overline{p}_j) - t(p_j)) = \frac{1}{\beta-1} \cdot y_{U_j} \tag{11}$$

budget. We have to show that the dual load on P_i^s attributed to terminal pair (s_j, t_j) is at least the sum of the quantities in (10) and (11).

Let U_j^0 be the moat containing s_j at time $t(p_j)$ during the execution of $\texttt{AKR}_2$ on terminal set R^0. The subtle point to notice is that P_i^s may not intersect U_j^0. On the other hand we know that U_j *does* intersect P_i^s. How is this possible?

The only possible reason is that the moat containing s helps loading the path connecting s_j and p_j in $\texttt{AKR}(R)$. Figure 1 shows the point of time in $\texttt{AKR}(R)$ when U_j starts loading P_i^s. Assume that the moats containing terminals s and s_j meet at point p_j' at time t_j'. Point p_j'' is the closest point to p_j that feels dual load from s at time t_j'. For any $1 \leq j \leq l$, the dual load on the path connecting s_j to p_j'' due to subsets of the moat containing s in time interval $[0, t_j')$ is bounded by $2t_j'$.

The crucial observation is that s_j is part of a moat U_j^0 at time $t(p_j)$ in $\texttt{AKR}_2(R^0)$ that is active. Moreover, s_j was part of an active component (a subset of U_j^0) at all times $t \in [0, t(p_j))]$. Similarly, t_j was part of a separate active moat in $\texttt{AKR}_2(R^0)$ during all times $t \in [0, t(p_j)]$. The total budget collected by these moats until time $t(p_j)$ is at least

$$2 \cdot \frac{1}{2(\beta-1)} \cdot t(p_j) \geq 2t(p_j) \tag{12}$$

from the fact that $\beta \leq 3/2$.

The cost of growing s_j's moat U_j^0 until it reaches $t(p_j)$ is at most (12). We can therefore charge the cost for reaching p_j in $\texttt{AKR}_2(R^0)$ to the budget portion characterized in (12).

The two cases imply the correctness of the lemma. ■

Acknowledgment We would like to thank R. Ravi for sharing his insights on primal-dual algorithm for Steiner forests and trees with us.

References

[1] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. *SIAM J. Comput.*, 24:440–456, 1995.

[2] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 542–547, 20–22 October 1997.

[3] Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings, ACM Symposium on Theory of Computing*, pages 161–168, May 23–26 1998.

[4] Feigenbaum, Papadimitriou, and Shenker. Sharing the cost of multicast transmissions. *JCSS: Journal of Computer and System Sciences*, 63, 2001.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[6] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995.

[7] A. Gupta, A. Kumar, M. Pal, and T. Roughgarden. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, 2003.

[8] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings, ACM Symposium on Theory of Computing*, pages 365–372, 2003.

[9] K. Jain and V. V. Vazirani. Applications of approximation algorithms to cooperative games. In *Proceedings, ACM Symposium on Theory of Computing*, pages 364–372, 2001.

[10] J. Könemann. *Approximation Algorithms for Minimum-Cost Low-Degree Subgraphs*. PhD thesis, Carnegie Mellon University, 2003.

[11] J. Könemann and R. Ravi. Quasi-polynomial time approximation algorithm for low-degree minimum-cost steiner trees. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science.*, 2003.

[12] Amit Kumar, Anupam Gupta, and Tim Roughgarden. A constant-factor approximation algorithm for the multicommodity. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 333–344, 2002.

[13] M. Pál. Personal communication. 2003.

[14] M. Pál and É. Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, 2003.

Appendix

A Analysis of AKR

In this section we give a self-contained analysis of the primal-dual algorithm for Steiner forests by Agrawal, Klein, and Ravi. In particular we provide a proof of Theorem 2.

Recall that we view AKR as a process over time and that the algorithm maintains a forest F^t for each time t. Recall also, that we call a component C of F^t active if C separates at least one terminal pair from R. The algorithm then raises dual variables corresponding to the set of terminals contained in active sets simultaneously until a constraint of type (1) becomes tight. We merge the two components that are colliding and add the cheapest connecting path whenever this happens.

Merging two components by adding a connecting path to F^t yields a new component that may be *inactive*, i.e. the newly created component $C_u \cup C_v$ may not separate any terminal pairs. The dual variables corresponding to inactive sets are not raised by AKR. Inactive components play an important role in our analysis and we let $\mathcal{I}^t$ be the set of all inactive components at time t. For $C \in \mathcal{I}^t$ we let $\texttt{age}_C$ be the time at which C was created (i.e. the time at which it became inactive).

Every connected component $C \in \mathcal{I}^t$ is either disjoint from other components, or included in a larger component C', if at some time $t' \geq \texttt{age}_C$ a path P_{uv} connecting u to v touches C to form a larger connected component C'. Motivated by this distinction, we define the following partition of $\mathcal{I}^t$:

$$\mathcal{I}^t = \mathcal{D}^t \cup \mathcal{Z}^t.$$

$\mathcal{D}^t \subset \mathcal{I}^t$ is the set of all inactive isolated components in $\mathcal{I}^t$. We refer to those components as being *dead*. The set $\mathcal{Z}^t \subseteq \mathcal{I}^t$ contains those inactive components that are contained in larger connected components (active or inactive). We refer to a component $C \in \mathcal{Z}^t$ as a *zombie* component.

Figure 2 shows an example situation where AKR adds path P_{uv} at time t in order to merge active components C_u and C_v. Assume first that P_{uv} touches a dead component $C \in \mathcal{D}^t$. Let P_{uv}^C be the part of P_{uv} that runs inside C. The idea is now to *charge* C for (parts) of the cost of P_{uv}. More precisely, we want C to cover exactly the cost of the part of P_{uv}^C that is loaded by duals corresponding to active subsets of $C|R$. The following lemma bounds the charge of component C.

Lemma 5 *Let AKR add path P_{uv} at time t to merge two components C_u and C_v. Also let P_{uv} touch a dead component $C \in \mathcal{D}^t$. The cost of P_{uv} that is added to C's charge is at most $2 \cdot \texttt{age}_C$. Moreover, an inactive components $C \in \mathcal{I}^t$ is charged at most once during the execution of AKR.*

Proof: P_{uv}^C consists of at most two segments (refer to Figure 2). Let those be P_1^C and P_2^C. Each one of these paths consists of edges that feel dual load exclusively from active subsets of $C|R$. It can now be seen that at each time $t' \leq t$ there is at most one active subset of $C|R$ that loads P_i^C for $i \in \{1, 2\}$. Hence, the cost of P_i^C is bounded by $\texttt{age}_C$ for $i \in \{1, 2\}$.

Notice that C becomes a zombie as a direct consequence of merging C_u and C_v along P_{uv}. Since zombie components are never charged the lemma follows. ■

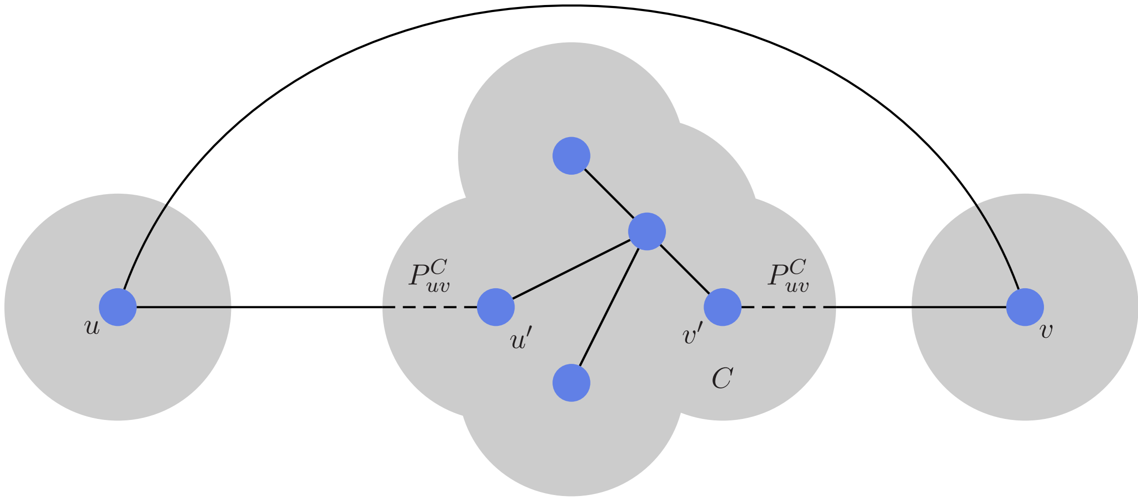

Figure 2: AKR merges two active components C_u and C_v along path P_{uv}. The path touches an inactive component $C \in \mathcal{D}^t$. The part of P_{uv} labeled P_{uv}^C has to be added inside C. This part consists of edges that feel dual load from active subsets of $C|R$.

P_{uv} may touch zombie components. The previous argument takes care of any zombie components that P_{uv} touches within dead components. For these components, we simply charge the cost of the part of P_{uv} within them to the dead parent component surrounding it. We are left with zombie components within C_u and C_v, respectively. A zombie component within C_u (within C_v) is connected to C_u (C_v).

In the following we focus on C_u. The argument for C_v is analogous. Figure 3 demonstrates the notation we use. Let C'_u be the zombie component within C_u that is touched last by P_{uv} on the way from u to v. Assume that u' is the first node of C'_u on P_{uv} and u'' is the last. Let c_u be the cost of the part of the u, u'-path $P_{uu'}$ that is not loaded by duals inside C'_u.

Component C'_u was already dead when it was integrated into C_u. In other words, c_u must be at least the age $\texttt{age}_{C'_u}$ of zombie component C'_u. A symmetric argument leads to a similar bound for the cost c_v of the path between vertex v and the first vertex v' within zombie component C'_v.

Notice that u'' and v'' are connected to u and v, respectively, prior to the merger at time t. Hence, AKR effectively only adds the segment of P_{uv} between u'' and v'' in the process of merging C_u and C_v. The following lemma bounds the cost of the path segment that AKR adds.

Lemma 6 *Let* AKR *merge C_u and C_v at time t using path P_{uv} and assume that P_{uv} passes through zombie components $C'_u \subseteq C_u$ and $C'_v \subseteq C_v$. Using the notation from Figure 3 we then must have*

$$c_{uv} + c'_u + c'_v \le 2t.$$

Proof: It follows from the way that AKR grows duals that

$$c_u + c_v + c_{uv} \le 2t. \tag{13}$$

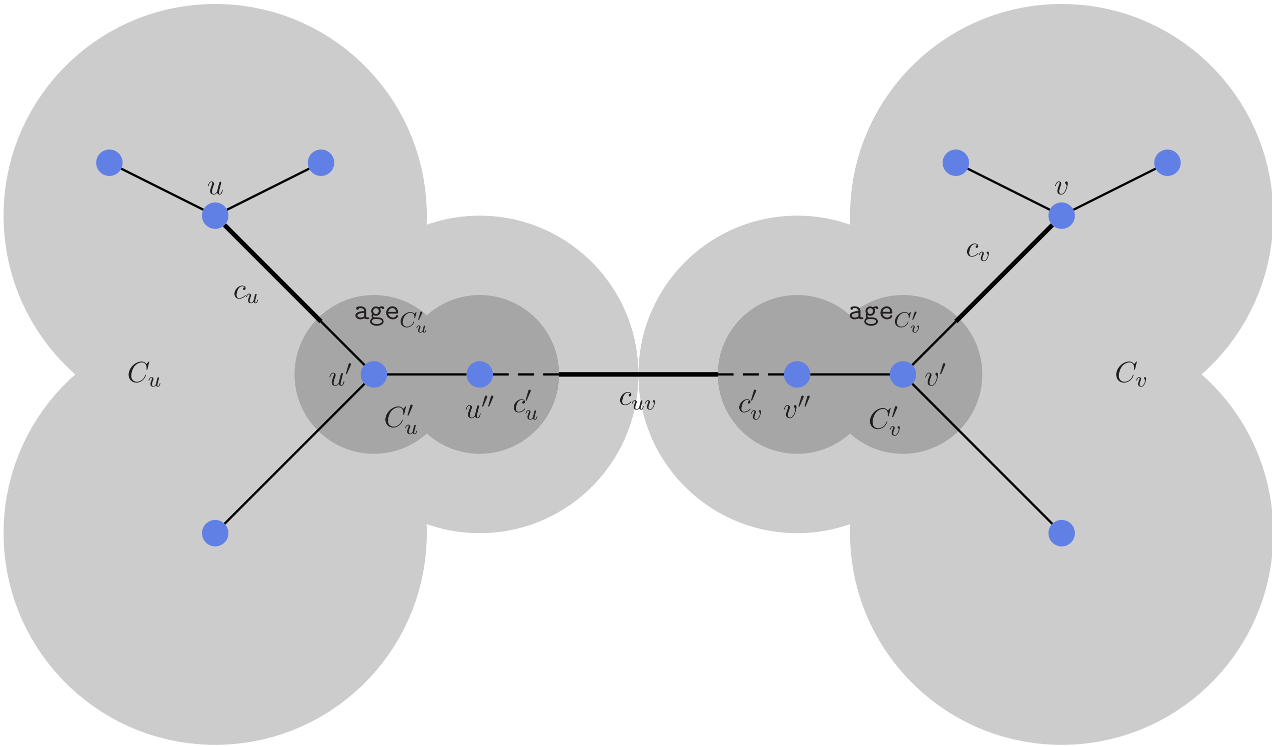

Figure 3: The figure shows components C_u and C_v at time t. C'_u and C'_v are zombie components contained in C_u and C_v, respectively.

Since C'_u was dead when AKR integrated it into C_u, we must have $\mathtt{age}_{C'_u} \leq c_u$ and similarly $\mathtt{age}_{C'_v} \leq c_v$. On the other hand we have that $c'_v \leq \mathtt{age}_{C'_v}$ and $c'_u \leq \mathtt{age}_{C'_v}$.

Together with (13) it then follows that $c_{uv} + c'_u + c'_v \leq 2t$ and this finishes the proof of the lemma. ■

The algorithm continues growing active components until a feasible solution is reached, i.e. the terminals of each pair lie in the same tree of the final forest.

We then conclude with the proof of Theorem 2 which we restate for completeness.

Theorem 2 *Suppose that algorithm* AKR *stops at time* t^*. *We then must have that*

$$c(F^{t^*}) \leq 2 \cdot \sum_{U \subseteq R} y_U^{t^*}.$$

Proof: The proof is by induction over time. Let C be a connected component of F^t. We let $c(C)$ be the total cost of the tree contained in C. We then show

$$c(C) \leq 2 \cdot \sum_{U \subseteq C|R} y_U^t - \begin{cases} 2 \cdot \min\{t, \mathtt{age}_C\} & : \quad C \notin \mathcal{D}^t \\ 0 & : \quad C \in \mathcal{D}^t \end{cases} \tag{14}$$

for all components C. Equation (14) clearly holds for $t = 0$. Using the induction hypothesis we know that

$$c(C) \leq 2 \cdot \sum_{U \subseteq C|R} y_U^t - 2 \cdot \mathtt{age}_C$$

holds for every dead component $C \in \mathcal{D}^t$.

Suppose that AKR adds path P_{uv} add time t to merge components C_u and C_v. Assume that P_{uv} touches dead components $T^1, \ldots, T^k$ and let the newly created component be denoted by C. Using Lemma 5 and 6 we obtain that the cost of the tree in the new component C is bounded by

$$\begin{aligned} c(C) &\leq c(C_u) + c(C_v) + c(P_{uv}) - c_u - c_v + c'_u + c'_v + \sum_{i=1}^{k} \left(c(C^i) + c(P_{uv}^{C^i}) \right) \\ &\leq \sum_{U \subseteq C_u | R} y_U^t + \sum_{U \subseteq C_v | R} y_U^t - 4t + 2t + \sum_{i=1}^{k} (c(C^i) + 2 \cdot \mathtt{age}_{C^i}) \\ &\leq \sum_{U \subseteq C | R} y_U^t - 2t. \end{aligned}$$

■

B Proofs of Lemmas 2 and 3

Proof of Lemma 2: By definition of the sets $\mathcal{U}_j^i$ we must have $|\delta(U) \cap P_i^s| \geq 1$ for all $U \in \bigcup_{j=1}^{3} \mathcal{U}_j^i$. First consider a set $U \in \mathcal{U}_2^i \cup \mathcal{U}_3^i$ that contains either s or t. It follows from the way that AKR grows duals and the way it adds paths that the P_i^s cannot intersect $\delta(U)$ more than once.

Now consider a set $U \in \mathcal{U}_1^i$. Since $\{s, t\} \cap U = \emptyset$ by definition, we have that U develops in the same way in AKR(R) and AKR(R^0). Therefore, U must be fully contained in either C_i^s or C_{i+1}^s. Recall that P is a tight s,t-path in forest F computed by AKR(R). Again from the definition of AKR it follows that P intersects U at most twice. However, only one of these intersections can correspond to segment P_i^s. ■

Proof of Lemma 3: The reason for why AKR(R^0) stops growing U at time t_2 must be a merger with another (active) moat. Let this moat be U_m. Since U_m and U are both active at time t_2 it must be the case that for any point of time during the interval $[t_1, t_2]$ there must be some subset of U_m that is active and grows. Furthermore, U and U_m must be either both in C_i^s or both in C_{i+1}^s. Using the same argument as in Lemma 2 it can then be argued that P_i^s can not intersect both sets and hence $U_m \notin \bigcup_{j=1}^{3} \mathcal{U}_j^i$. ■