RAIRO Operations Research
RAIRO Oper. Res. **38** (2004) 319–344
DOI: 10.1051/ro:2004028

FAST APPROXIMATION OF MINIMUM MULTICAST CONGESTION – IMPLEMENTATION VERSUS THEORY *

Andreas Baltz[1] and Anand Srivastav[1]

Communicated by Rainer E. Burkard

Abstract. The problem of minimizing the maximum edge congestion in a multicast communication network generalizes the well-known NP-hard multicommodity flow problem. We give the presently best theoretical approximation results as well as efficient implementations. In particular we show that for a network with m edges and k multicast requests, an $r(1+\varepsilon)(r\mathrm{OPT} + \exp(1)\ln m)$-approximation can be computed in $O(km\varepsilon^{-2}\ln k \ln m)$ time, where β bounds the time for computing an r-approximate minimum Steiner tree. Moreover, we present a new fast heuristic that outperforms the primal-dual approaches with respect to both running time and objective value.

Keywords. Combinatorial optimization, approximation algorithms.

Mathematics Subject Classification. 68W25, 90C27.

1. Introduction

A communication network can be modeled as an undirected graph $G = (V, E)$, where each node represents a computer that is able to receive, copy and send packets of data. In *multicast* traffic, several nodes have a simultaneous demand to receive a copy of a single packet. These nodes together with the packet's source specify a *multicast request* $S \subseteq V$. Meeting the request means to establish a connection represented by a *Steiner tree* with *terminal set* S, *i.e.* a subtree of G

* *Research of the first author supported by DFG, Grant SR 7/9 – 2.*

[1] Mathematisches Seminar, Bereich II, Christian-Albrechts-Universität zu Kiel, Christian-Albrechts-Platz 4, D-24118 Kiel, Germany; e-mail: aba;asr@numerik.uni-kiel.de

that contains S and has no leaf in $V \setminus S$. Given G and a set of multicast requests it is natural to ask about Steiner trees such that the maximum *edge congestion*, *i.e.* the maximum number of trees sharing an edge is as small as possible. Solving this minimization problem is NP-hard, since it contains as a special case the well-known NP-hard standard routing problem of finding (unsplittable) paths with minimum congestion. The MAX-SNP-hard minimum Steiner tree problem is also closely related.

1.1. Previous work

Vempala and Vöcking [17] gave a randomized algorithm for approximating the minimum multicast congestion problem within a factor of $O(\log n)$ by applying randomized rounding to an integer linear program relaxation. The considered LP contains an *exponential* number of constraints corresponding to the exponential number of possible Steiner trees. Vempala and Vöcking handled this difficulty by considering a multicommodity flow relaxation for which they could devise a polynomial separation oracle. By rounding the fractional paths in an $O(\log n)$-stage process they proved an $O(\log n)$ approximation. Carr and Vempala [2] gave an algorithm for approximating the minimum multicast congestion within a constant factor *plus* $O(\log n)$. They showed that an r-approximate solution to an LP-relaxation can be written as a convex combination of Steiner trees. Randomized rounding yields a solution not exceeding a congestion of $\max\{2\exp(1)\cdot r\mathrm{OPT}, 2(\varepsilon+2)\log n\}$ with probability at least $1 - n^{-\varepsilon}$, where r is the approximation factor of the network Steiner problem. Both algorithms use the ellipsoid method with separation oracle and thus are of mere theoretical value. A closely related line of research is concerned with combinatorial approximation algorithms for multicommodity flow and – more general – fractional packing and covering problems. Matula and Shahrokhi [11] were the first to develop a combinatorial strongly polynomial approximation algorithm for the uniform concurrent flow problem. Their method was generalized and improved by Goldberg [4], Leighton *et al.* [10], Klein *et al.* [9], Plotkin *et al.* [13], and Radzik [14]. A fast version that is particularly simple to analyze is due to Garg and Könemann [3]. Independently, similar results were given by Grigoriadis and Khachiyan [6]. In two recent papers Jansen and Zhang [7, 8] extended the latter approach. In particular, they presented a randomized algorithm for approximating the minimum multicast congestion within $(1+\varepsilon)(r\mathrm{OPT} + \exp(1)\ln m)$ in time $O(m(\ln m + \varepsilon^{-2}\ln\varepsilon)(k\beta + m\ln\ln(m/\varepsilon))$, where β is the running time of a minimum Steiner tree approximation. The online version of the problem was considered by Aspnes *et al.* [1].

1.2. Our results

In Section 2 we present three algorithms which solve the *fractional* multicast congestion problem up to a relative error of $r(1+\varepsilon)$. The fastest of these takes time $O(k\beta\varepsilon^{-2}\ln k\ln m)$ (where β bounds the time for computing an r-approximate minimum Steiner tree), and thus improves over the previously best running time

bound of Jansen and Zhang. By means of randomized rounding we obtain a $(1+\varepsilon)(r\mathrm{OPT} + \exp(1)\ln m)$-approximation to the *integral* multicast congestion problem. With the tools of Srivastav and Stangier [16] the rounding procedure can be derandomized (Sect. 2.3.2). The three algorithms are based on the approaches of Plotkin, Shmoys and Tardos [13], Radzik [14], and Garg and Könemann [3] to path-packing and general packing problems. In experiments it turned out that straightforward implementations of the above (theoretically fast) combinatorial algorithms are quite slow. However, simple modifications lead to a very fast new algorithm with much better approximation results than hinted at by the worst case bounds (Sect. 3). The central idea is to repeatedly improve badly served requests *via* approximate minimum Steiner trees where the length function punishes highly congested edges more heavily than proposed in the theory. A proof confirming the experimental performance remains a challenging open problem.

1.3. Notations

By $G = (V, E)$ we will denote the given undirected graph on n nodes and m edges. We consider k multicast requests, $S_1, \ldots, S_k \subseteq V$. The set of all Steiner trees with terminal nodes given by S_i is denoted by $\mathcal{T}_i$, while $\hat{\mathcal{T}}_i$ is the set of Steiner trees for S_i computed in the course of the considered algorithm. As to the (possibly fractional) congestion, we distinguish between $c_i(T)$, the congestion caused by tree T for request S_i, $c_i(e) := \sum_{\substack{T\in\mathcal{T}_i \\ e\in E(T)}} c_i(T)$, the congestion of edge $e \in E$ due to request S_i, $c(e) := \sum_{i=1}^{k} c_i(e)$, the total congestion of edge $e \in E$ and $c_{\max} := \max_{e\in E} c(e)$, the maximum edge congestion. We will define a nonnegative length function l on the edges and abbreviate the sum of edge lengths of a subgraph U of G by $l(U) := \sum_{e\in E(U)} l(e)$. The time for computing an r-approximate optimum Steiner tree will be denoted by β. Throughout, $r \geq 1.55$ and $\varepsilon \in (0, 1]$ will be constant parameters describing the guaranteed ratio of our minimum Steiner tree approximation [18] and determining the target approximation of the fractional optimization problem, respectively. $\mathrm{MST}_l(S_i)$ and $\tilde{\mathrm{MST}}_l(S_i)$ are used to denote a minimum Steiner tree and an approximate minimum Steiner tree for S_i with respect to l (l is usually omitted); we assume $l(\tilde{\mathrm{MST}}(S_i)) \leq r \cdot l(\mathrm{MST}(S_i))$. Generally, a variable indexed by a subscript i refers to the ith multicast request S_i, while a superscript (i) refers to the end of the ith iteration of a `while`- or `for`-loop in an algorithm. The optimal fractional congestion is denoted by OPT.

2. Approximation algorithms

The subsequent analyses of algorithms are similar to analyses presented originally for the multicommodity problem with non-uniform commodity demands and non-uniform edge weights. For ease of presentation we shall restrict us here to the uniform multicast congestion problem. However a generalization to non-uniform edge-capacities and requests having different weights should be straightforward.

2.1. LP formulation

The minimum multicast congestion problem can be formulated as an integer linear program. We study a natural LP relaxation and its dual:

$$
\begin{array}{lll}
\text{(LP)} & \min z & \\
& \text{s. t.} & \\
& \sum_{T\in\mathcal{T}_i} c_i(T) \geq 1 & \text{for all } i \in \{1,\dots,k\} \\
& c(e) = \sum_{i=1}^{k} \sum_{\substack{T\in\mathcal{T}_i \\ e\in E(T)}} c_i(T) \leq z & \text{for all } e \in E \\
& c_i(T) \geq 0 & \text{for all } i \text{ and all } T \in \mathcal{T}_i
\end{array}
$$

$$
\begin{array}{lll}
\text{(LP}^*\text{)} & \max \sum_{i=1}^{k} Y_i & \\
& \text{s. t.} & \\
& \sum_{e\in E} l(e) \leq 1 & \\
& \sum_{e\in E(T)} l(e) \geq Y_i & \text{for all } i \text{ and all } T \in \mathcal{T}_i \\
& l(e) \geq 0 & \text{for all } e \in E \\
& Y_i \geq 0 & \text{for all } i \in \{1,\dots,k\}.
\end{array}
$$

Lemma 2.1. *Let $(Y_1^*,\dots,Y_k^*,l^* : E \to \mathbb{R}_{\geq 0})$ be a feasible solution to (LP*). $(Y_1^*,\dots,Y_k^*,l^* : E \to \mathbb{R}_{\geq 0})$ is an optimal dual solution if and only if $\sum_{i=1}^{k} Y_i^* = \sum_{i=1}^{k} l^*(MST(S_i)) = \max\left\{ \frac{\sum_{i=1}^{k} l(MST(S_i))}{l(G)} \mid l : E \to \mathbb{R}_{\geq 0},\ l \not\equiv 0 \right\}$.*

Proof. Let $(Y_1^*,\dots,Y_k^*,l^* : E \to \mathbb{R}_{\geq 0})$ be an optimal dual solution. We may assume that $Y_i^* = l^*(\mathrm{MST}(S_i))$ for all i and $l^*(G) = 1$. The latter equality can be obtained by enlarging some $l^*(e)$; the former must hold, since otherwise we could increase $\sum_{i=1}^{k} Y_i^*$ contradicting optimality. Hence,

$$
\sum_{i=1}^{k} Y_i^* = \frac{\sum_{i=1}^{k} l^*(\mathrm{MST}(S_i))}{l^*(G)} \leq \max\left\{ \frac{\sum_{i=1}^{k} l(\mathrm{MST}(S_i))}{l(G)} \mid l : E \to \mathbb{R}_{\geq 0},\ l \not\equiv 0 \right\}.
$$

On the other hand, let $l : E \to \mathbb{R}_{\geq 0}$ be a length function with maximum ratio $\frac{\sum_{i=1}^{k} l(\mathrm{MST}(S_i))}{l(G)}$. Defining $l' : E \to \mathbb{R}_{\geq 0}$ by $e \mapsto \frac{l(e)}{l(G)}$ we have $l'(G) = 1$ and $l'(T) = \frac{l(T)}{l(G)} \geq \frac{l(\mathrm{MST}(S_i))}{l(G)}$ for all $T \in \mathcal{T}_i$. Now, setting $Y_i' := l'(MST(S_i))$ for all $i \in \{1,\dots,k\}$ yields

$$
\sum_{i=1}^{k} Y_i^* \geq \sum_{i=1}^{k} Y_i' \geq \max\left\{ \frac{\sum_{i=1}^{k} l(\mathrm{MST}(S_i))}{l(G)} \mid l : E \to \mathbb{R}_{\geq 0},\ l \not\equiv 0 \right\}. \qquad \square
$$

Corollary 2.2. *$OPT \geq \frac{\sum_{i=1}^{k} l(MST(S_i))}{l(G)}$ for all $l : E \to \mathbb{R}_{\geq 0}$.*

By complementary slackness, feasible solutions to (LP) and (LP*) are optimal if and only if

$$l(e)(c(e) - z) = 0 \text{ for all } e \in E \tag{1}$$

$$c_i(T)(l(T) - Y_i) = 0 \text{ for all } i \in \{1, \ldots, k\} \text{ and all } T \in \mathcal{T}_i \tag{2}$$

$$Y_i \left(\sum_{T \in \mathcal{T}_i} c_i(T) - 1 \right) = 0 \text{ for all } i \in \{1, \ldots, k\}. \tag{3}$$

To guarantee optimality, it is sufficient to replace (1), (2), and (3) by

$$l(e)(c(e) - c_{\max}) = 0 \text{ for all } e \in E, \tag{1'}$$

$$c_i(T)(l(T) - l(\mathrm{MST}(S_i)) = 0 \text{ for all } i \in \{1, \ldots, k\}, \text{ and all } T \in \mathcal{T}_i, \tag{2'}$$

$$\sum_{T \in \mathcal{T}_i} c_i(T) = 1. \tag{3'}$$

Summing (1′) over all edges and (2′) over all $i \in \{1, \ldots, k\}$ and all $T \in \mathcal{T}_i$ yields

$$\sum_{e \in E} c(e) l(e) = l(G) c_{\max} \tag{1"}$$

$$\sum_{i=1}^{k} l(\mathrm{MST}(S_i)) = \sum_{i=1}^{k} \sum_{T \in \mathcal{T}_i} c_i(T) l(T) = \sum_{e \in E} c(e) l(e). \tag{2"}$$

In the derivation of (2″) we used (3′) and the fact that $\sum_{i=1}^{k} \sum_{T \in \mathcal{T}_i} c_i(T) l(T) = \sum_{e \in E} l(e) \sum_{i=1}^{k} \sum \sum_{\substack{T \in \mathcal{T}_i \\ e \in E(T)}} c_i(T)$ which equals $\sum_{e \in E} l(e) c(e)$ by definition of $c(e)$.

Any approximately optimal solution to (LP) that is found in polynomial time determines for each multicast request S_i a (polynomial) set of trees with fractional congestion. The task of selecting one of these trees for each S_i such that the maximum congestion is minimized constitutes a vector selection problem. Raghavan [15] bounds the quality of a solution by analyzing a randomized rounding procedure. For our problem his analysis gives the following result.

Theorem 2.3. *There exists an $O(k\beta\varepsilon^{-2} \ln k \ln m)$-time randomized algorithm for computing a solution to the minimum multicast congestion problem with congestion bounded by*

$$\begin{cases} (1+\varepsilon) r\,OPT + (1+\varepsilon)(\exp(1)-1)\sqrt{r\,OPT \cdot \ln m} & \text{if } r\,OPT \geq \ln m \\ (1+\varepsilon) r\,OPT + \dfrac{(1+\varepsilon)\exp(1)\ln m}{1 + \ln(\frac{\ln m}{r\,OPT})} & \text{otherwise.} \end{cases}$$

2.2. Approximating the fractional optimum

2.2.1. *The Plotkin-Shmoys-Tardos approach*

Plotkin, Shmoys and Tardos specify conditions that are sufficient to guarantee relaxed optimality. Adapted to the multicast problem, these conditions are the following relaxations of $(1'')$ and $(2'')$.

$$(1-\varepsilon)c_{\max}l(G) \leq \sum_{e\in E} c(e)l(e) \tag{R1}$$

$$(1-\varepsilon)\sum_{e\in E} c(e)l(e) \leq \sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) + \varepsilon c_{\max}l(G). \tag{R2}$$

Their idea is to choose l such that (R1) is automatically satisfied, and to gradually satisfy (R2) by repeated calls to the following routine.

Algorithm 2.4 (ImproveCongestion1)**.**
$\lambda := c_{\max}/2, \alpha := \frac{4\ln(2m\varepsilon^{-1})}{\varepsilon\cdot c_{\max}}, \sigma := 5\varepsilon/(9\alpha k)$
while (R2) is not satisfied and $c_{\max} > \lambda$
  for $e \in E$
    $l(e) := \exp(\alpha \cdot c(e))$
  for $i := 1$ *to* k
    $T_i := \tilde{MST}(S_i)$
    for $e \in E(T_i)$
      $c(e) := c(e) + \sigma/(1-\sigma)$
  for $e \in E$
    $c(e) := c(e) \cdot (1-\sigma)$

Lemma 2.5.

a) *Conditions (R1) and (R2) imply that* $c_{\max} \leq (1+6\varepsilon)r \cdot OPT$ *for* $\varepsilon \leq 1/6$.
b) $l(e) := \exp(\alpha \cdot c(e))$ *for all* $e \in E$ *and* $\alpha \geq \frac{2\ln(2m\varepsilon^{-1})}{c_{\max}\varepsilon}$ *imply (R1).*

Proof. a) From (R1) and (R2) we conclude

$$\begin{aligned} r\sum_{i=1}^{k} l(\mathrm{MST}(S_i)) &\overset{(R2)}{\geq} (1-\varepsilon)\sum_{e\in E} c(e)l(e) - \varepsilon c_{\max}l(G) \\ &\overset{(R1)}{\geq} (1-\varepsilon)^2 c_{\max}l(G) - \varepsilon c_{\max}l(G) \\ &\geq (1-3\varepsilon)c_{\max}l(G). \end{aligned}$$

Hence, by the choice of ε,

$$c_{\max} \leq \frac{1}{1-3\varepsilon} \cdot \frac{r \sum_{i=1}^{k} l(\mathrm{MST}(S_i))}{l(G)} \leq \frac{r\mathrm{OPT}}{1-3\varepsilon} \leq (1+6\varepsilon) r\mathrm{OPT}.$$

b) We show

(i) $\alpha \geq \frac{2\ln(2m\varepsilon^{-1})}{c_{\max}\varepsilon}$ implies $\left(1-\frac{\varepsilon}{2}\right) c_{\max} \leq c(e)$ or $l(e) < \frac{\varepsilon}{2m} l(G)$ for all $e \in E$;
(ii) (i) implies (R1).

To prove (i), consider an arbitrary $e \in E$. If $c(e) < \left(1-\frac{\varepsilon}{2}\right) c_{\max}$ we have $l(e) = \exp(\alpha c(e)) < \exp\left(\alpha\left(1-\frac{\varepsilon}{2}\right) c_{\max}\right)$. Since $l(G) \geq \exp(\alpha \cdot c_{\max})$, we see that $\frac{l(e)}{l(G)} < \exp\left(-\frac{\varepsilon}{2}\alpha c_{\max}\right) \leq \frac{\varepsilon}{2m}$, so $l(e) < \frac{\varepsilon}{2m} l(G)$.

For (ii), let $B := \{e \in E \mid (1-\varepsilon/2)c_{\max} \leq c(e)\}$. Now,

$$\begin{aligned} c_{\max} l(G) &= c_{\max} \sum_{e \in B} l(e) + c_{\max} \sum_{e \in E \setminus B} l(e) \\ &\leq \frac{1}{1-\varepsilon/2} \sum_{e \in B} c(e) l(e) + c_{\max} \sum_{e \in E \setminus B} \frac{\varepsilon}{2m} l(G) \\ &\leq \frac{1}{1-\varepsilon/2} \sum_{e \in E} c(e) l(e) + c_{\max} \frac{\varepsilon}{2} l(G). \end{aligned}$$

This is equivalent to

$$c_{\max} l(G)(1-\varepsilon/2)(1-\varepsilon/2) \leq \sum_{e \in E} c(e) l(e),$$

so $(1-\varepsilon) c_{\max} l(G) \leq \sum_{e \in E} c(e) l(e)$ as claimed. □

The following lemma serves to bound the number of iterations until (R2) is satisfied.

Lemma 2.6. *Let $\varepsilon \leq \frac{1}{6}$ and consider a feasible solution $(c_i(T), c_{\max})$ to (LP), such that for $\alpha \geq \frac{2\ln(2m\varepsilon^{-1})}{c_{\max}\varepsilon}$ and $l(e) := \exp(\alpha \cdot c(e))$ (for all $e \in E$) (R2) is not satisfied. Define $\sigma := 5\varepsilon/(9\alpha k)$ and let $T_i := \tilde{M}ST(S_i)$ be an approximate minimum Steiner tree for $i \in \{1, \ldots, k\}$. Then $\tilde{c}_i(T) := \begin{cases} (1-\sigma) c_i(T), & \text{if } T \neq T_i \\ (1-\sigma) c_i(T) + \sigma, & \text{if } T = T_i, \end{cases}$ and $\tilde{l}(e) := \exp(\alpha \cdot \tilde{c}(e)) := \exp\left(\alpha \sum_{i=1}^{k} \sum_{\substack{T \in \mathcal{T}_i \\ e \in E(T)}} \tilde{c}_i(T)\right)$ satisfy*

$$l(G) - \tilde{l}(G) \geq \frac{5\varepsilon^2 c_{\max}}{9k} l(G).$$

Proof. For $|\delta| \leq \frac{5}{9}\varepsilon \leq \frac{5}{54}$ we can use the Taylor expansion to estimate

$$\exp(x+\delta) < \exp(x) + \delta\exp(x) + |\delta|\frac{\varepsilon}{3}\exp(x).$$

By the choice of α, σ and ε, $\alpha\sigma\left|\left(\sum_{\substack{i=1\\E(T_i)\ni e}}^{k} 1\right) - c(e)\right| \leq \alpha\sigma k = \frac{5}{9}\varepsilon \leq \frac{5}{54}$, for all $e \in E$, so

$$\begin{aligned}
\tilde{l}(e) &= \exp(\alpha\tilde{c}(e)) = \exp\left(\alpha(1-\sigma)c(e) + \alpha\sigma \sum_{\substack{i=1\\E(T_i)\ni e}}^{k} 1\right) \\
&= \exp\left(\alpha c(e) + \alpha\sigma\left[\left(\sum_{\substack{i=1\\E(T_i)\ni e}}^{k} 1\right) - c(e)\right]\right) \\
&< l(e)\left(1 + \alpha\sigma\left[\left(\sum_{\substack{i=1\\E(T_i)\ni e}}^{k} 1\right) - c(e)\right] + \frac{\alpha\sigma\varepsilon}{3}\left[\left(\sum_{\substack{i=1\\E(T_i)\ni e}}^{k} 1\right) + c(e)\right]\right). \quad (4)
\end{aligned}$$

Note that

$$\begin{aligned}
\sum_{e\in E}\sum_{\substack{i=1\\E(T_i)\ni e}}^{k} l(e) &= \sum_{i=1}^{k} l(T_i) = \sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) \quad (5)\\
&\leq r\sum_{i=1}^{k} l(\mathrm{MST}(S_i)) < 2\sum_{i=1}^{k} l(\mathrm{MST}(S_i)), \quad (6)
\end{aligned}$$

and due to feasibility,

$$\begin{aligned}
\sum_{i=1}^{k} l(\mathrm{MST}(S_i)) &\leq \sum_{i=1}^{k}\sum_{T\in\mathcal{T}_i} l(\mathrm{MST}(S_i))c_i(T) \leq \sum_{i=1}^{k}\sum_{T\in\mathcal{T}_i} l(T)c_i(T) \\
&= \sum_{e\in E} l(e)\sum_{i=1}^{k}\sum_{\substack{T\in\mathcal{T}_i\\e\in T}} c_i(T) = \sum_{e\in E} l(e)c(e). \quad (7)
\end{aligned}$$

Hence,

$$\begin{aligned} l(G) - \tilde{l}(G) &\overset{(4)}{>} \sum_{e\in E} \alpha\sigma \left[\left(- \sum_{\substack{i=1 \\ E(T_i)\ni e}}^{k} l(e) \right) + c(e)l(e) \right] \\ &\quad - \sum_{e\in E} \frac{\alpha\sigma\varepsilon}{3} \left[\left(\sum_{\substack{i=1 \\ E(T_i)\ni e}}^{k} l(e) \right) + c(e)l(e) \right] \\ &\overset{(5)}{=} \alpha\sigma \left(-\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) + \sum_{e\in E} c(e)l(e) \right) \\ &\quad - \frac{\alpha\sigma\varepsilon}{3} \left[\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) + \sum_{e\in E} l(e)c(e) \right] \\ &\overset{(6),(7)}{>} \alpha\sigma \left(-\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) + \sum_{e\in E} c(e)l(e) - \varepsilon \sum_{e\in E} l(e)c(e) \right). \end{aligned}$$

Since (R2) is not satisfied,

$$\begin{aligned} l(G) - \tilde{l}(G) &> \alpha\sigma \left(-\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) + \sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i)) + \varepsilon c_{\max} l(G) \right) \\ &= \frac{5\varepsilon^2 c_{\max}}{9k} l(G). \end{aligned}$$

□

We thus see that in one iteration of the while loop of ImproveCongestion1 $l(G)$ decreases by a factor of at least $(1 - 5\varepsilon^2 c_{\max}/(9k)) \le \exp(-5\varepsilon^2 c_{\max}/(9k))$. The initial value of $l(G)$ is at most $m \cdot \exp(\alpha c_{\max})$. Surely, $r(1+6\varepsilon)$-optimality is achieved, if $l(G) \le \exp(r(1+6\varepsilon)\alpha\mathrm{OPT}))$. We will use this criterion to bound the required time complexity. Consider the following two phase process: in phase one, the overall length is reduced to at most $\exp(2r\mathrm{OPT})$; phase two successively reaches $r(1+6\varepsilon)$-optimality. Each phase consists of a number of subphases, corresponding to calls to ImproveCongestion1. During the whole of phase 1 – *i.e.* for each subphase i – we fix $\varepsilon^{(i)}$ at 1/6. The congestion is halved in each subphase; hence the number of subphases of phase 1 is $O(\ln c_{\max}) = O(\ln k)$, and by Lemma 2.6, we can bound the number z of iterations the while-loop of ImproveCongestion1 needs to terminate for the ith subphase of phase 1 by $O(k \ln m / c_{\max}^{(i)})$. As $c_{\max}$ is halved in each subphase, the number of iterations in the last subphase dominates the overall number. Consequently the running time of phase 1 is $O(k^2 \beta \ln m) = \tilde{O}(k^2 m)$ for the Steiner tree approximation by Mehlhorn[12]. (Remember, we assume $c_{\max} \ge \mathrm{OPT} \ge 1$.) The ith subphase of phase 2 starts with $c_{\max}^{(i-1)} \le r(1+6\varepsilon^{(i)})\mathrm{OPT}$ and terminates when the congestion is at most $r(1+3\varepsilon^{(i)})\mathrm{OPT}$. In order to reach $r(1+6\varepsilon)$-optimality

we thus need $O(\ln \varepsilon^{-1})$ subphases. By Lemma 2.6, the ith subphase terminates within z iterations, where z satisfies $m \exp(\alpha^{(i)} c_{\max}^{(i-1)}) \cdot \exp(-z \cdot 5 c_{\max}^{(i-1)} {\varepsilon^{(i)}}^2/(9k)) = \exp\left(\alpha^{(i)} c_{\max}^{(i-1)} \cdot \frac{1+3\varepsilon^{(i)}}{1+6\varepsilon^{(i)}}\right)$, *i.e.* $z = O(k {\varepsilon^{(i)}}^{-2} \ln(m/\varepsilon^{(i)}))$. Hence the running time of the ith subphase of phase 2 is $O(k^2 {\varepsilon^{(i)}}^{-2} \ln(m/\varepsilon^{(i)})\beta)$. Since $\varepsilon^{(i+1)} = \varepsilon^{(i)}/2$, the running time of the last subphase again is dominating, resulting in an overall estimate of $O(k^2\varepsilon^{-2} \cdot \ln(m/\varepsilon)\beta) = \tilde{O}(k^2\varepsilon^{-2}m)$ for the Steiner tree approximation by Mehlhorn. This proves the following theorem.

Theorem 2.7. *For $\varepsilon \leq \frac{1}{6}$ and assuming $OPT \geq 1$, the fractional minimum multicast congestion problem can be approximated within $r(1+6\varepsilon)OPT$ in time $O(k^2\varepsilon^{-2} \cdot \ln(m/\varepsilon)\beta)$. Using the Steiner tree approximation by Mehlhorn, we obtain a $2(1+6\varepsilon)$-approximation in time $\tilde{O}(k^2\varepsilon^{-2}m)$.*

Here is a suggestion how to implement the described algorithm.

Algorithm 2.8 (ApproximateCongestion1)**.**
Compute a start solution, initialize $c(e), c_{\max}, maxdual$ accordingly.
$\varepsilon := 1/3, factor_0 := 1, iteration := -1$
while $\varepsilon > \varepsilon_0/6$
  $\varepsilon := \varepsilon/2$
  while $c_{\max} > \max\{1, (1+6\varepsilon)maxdual\}$
    $\alpha := \frac{2\ln(2m/\varepsilon)}{\varepsilon \cdot c_{\max}}, \sigma := \frac{5\varepsilon}{9\alpha k}$
    for $e \in E$
      $l(e) := \exp(\alpha(c(e) - c_{\max}))$
    $iteration := iteration + 1$
    for $i := 1$ *to* k
      $T_i := \tilde{MST}(S_i), c(T_i) := c(T_i) + \frac{\sigma}{1-\sigma}, index(T_i) := iteration$
      for $e \in E(T_i)$
        $c(e) := c(e) + \frac{\sigma}{1-\sigma}$
    $maxdual := \max\{maxdual, \frac{\sum_{i=1}^{k} \sum_{e \in E(T_i)} l(e)}{\sum_{e \in E} l(e)}\}$
    $c_{\max} := 0$
    for $e \in E$
      $c(e) := c(e) \cdot (1-\sigma)$, *if* $c(e) > c_{\max}$ *then* $c_{\max} := c(e)$
      $factor_{iteration} := factor_{iteration-1} \cdot (1-\sigma)$
for $T \in \bigcup_{i=1}^{k} \hat{\mathcal{T}}_i$
  $c(T) := c(T) \cdot factor_{iteration - index(T_i)}$

Instead of testing for (R2), it seems favorable to use $maxdual := \sum_{i=1}^{k} \frac{l(\tilde{\mathrm{M}}\mathrm{ST}(S_i))}{l(G)}$ as a lower bound on rOPT and terminate the while-loop as soon as $c_{\max}$ falls below $(1+6\varepsilon)\cdot$`maxdual`. Deviating slightly from our previous notations we use ε_0 to denote the target approximation quality. For better efficiency, $c_i(T)$ is updated to hold the fractional congestion caused by tree T for request S_i only in the end

of the algorithm (in previous iterations it deviates from this by a factor called `factor`). We scale the length function by $\exp(-c_{\max})$ in order to stay within the computers floating point precision.

Goldberg, Oldham, Plotkin, and Stein [5] point out that the practical performance of the described algorithm significantly depends on the choice of α and σ. Instead of using the theoretical value for α, they suggest to choose α such that the ratio $\frac{\left(l(G)c_{\max}/\sum_{e\in E} c(e)l(e)\right)-1}{\left(\sum_{e\in E} c(e)l(e)/\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i))\right)-1}$ remains balanced. For this, they increase α by a factor of $(\sqrt{5}+1)/2$ as long as the ratio is larger than 0.5 and otherwise decrease it by $2/(\sqrt{5}+1)$. Since our aim at choosing α is to satisfy (R1), one could alternatively think of setting $\alpha := \min\left\{\alpha \in \mathbb{R}_{>0} \mid \frac{l(G)c_{\max}}{\sum_{e\in E} c(e)l(e)} \leq \frac{1}{1-\varepsilon}\right\}$. Moreover, they emphasize the necessity of choosing σ dynamically such as to maximize $l(G) - \tilde{l}(G)$ in each iteration. Since $\tilde{l}$ is the sum of exponential functions, we can easily compute $\frac{\mathrm{d}\tilde{l}(\sigma)}{\mathrm{d}\sigma} = \sum_{e\in E} \alpha(c_0(e) - c(e)) \exp(\alpha c(e) + \alpha\sigma(c_0(e) - c(e)))$, where $c_0(e) := |\{T \mid \exists i \in \{1,\ldots,k\} \text{ such that } T = \tilde{\mathrm{MST}}(S_i) \text{ and } e \in E(T)\}|$ abbreviates the congestion of edge e caused by choosing for each i an approximate minimum Steiner tree. The second derivative is positive, so $\tilde{l}$ has a unique minimum in $[\sigma_{\text{theor.}}, 1]$. Goldberg *et al.* suggest to determine this minimum with the Newton-Raphson method. One could also try binary search.

2.2.2. *The Radzik approach*

Adapting the algorithm of Radzik [14] to our problem, yields a theoretical speedup by a factor of k. The key idea is to update the length function after each single approximate Steiner tree computation, if the length of the new tree is sufficiently small. Radzik uses relaxed optimality conditions different from (R1) and (R2).

Theorem 2.9. *Let $\lambda \geq OPT$ and $\varepsilon \leq 1/3$. If*

$$c_{\max} \leq (1+\varepsilon/3)\lambda \quad \textit{and} \tag{8}$$

$$\sum_{i=1}^{k} l(\tilde{MST}(S_i)) \geq (1-\varepsilon/2)\lambda l(G) \tag{9}$$

then $c_{\max} \leq r(1+\varepsilon)OPT$.

Proof.

$$\begin{aligned} r\mathrm{OPT} &\geq r\frac{\sum_{i=1}^{k} l(\mathrm{MST}(S_i))}{l(G)} \geq \frac{\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i))}{l(G)} \\ &\overset{(9)}{\geq} \left(1-\frac{\varepsilon}{2}\right)\lambda \overset{(8)}{\geq} \frac{1-\varepsilon/2}{1+\varepsilon/3} c_{\max}. \end{aligned}$$

Hence, $c_{\max} \leq \frac{1+\varepsilon/3}{1-\varepsilon/2} r\mathrm{OPT} \leq (1+\varepsilon) r\mathrm{OPT}$, by the choice of ε. □

The length function of Radzik is very similar to the one of Plotkin *et al.*

Definition 2.10. Given some fixed $\lambda \geq \mathrm{OPT}$ we define

$$\alpha := \frac{3(1+\varepsilon)\ln(m\varepsilon^{-1})}{\lambda\varepsilon} \text{ and } l(e) := \exp(\alpha \cdot c(e)) \text{ for all } e \in E.$$

Lemma 2.11. *Let $m \geq 3$ and $\varepsilon \leq 1$. If*

$$\left(1-\frac{\varepsilon}{3}\right)\lambda \leq c_{\max} \leq \left(1+\frac{\varepsilon}{3}\right)\lambda \tag{10}$$

then

$$(1-\varepsilon)\lambda l(G) \leq \sum_{e\in E} c(e)l(e) \leq \left(1+\frac{\varepsilon}{3}\right)\lambda l(G). \tag{11}$$

Proof. Since $c(e) \leq c_{\max}$ the second inequality of (11) is immediate from the second inequality of (10). We will prove

$$l(e) \leq l(e)\frac{(1+\varepsilon)c(e)}{\lambda} + \frac{\varepsilon^2}{m}l(G) \text{ for all } e \in E. \tag{12}$$

From (12), $l(G) = \sum_{e\in E} l(e) \leq \sum_{e\in E} l(e)\frac{(1+\varepsilon)c(e)}{\lambda} + \varepsilon^2 l(G)$, thus $l(G)\lambda(1-\varepsilon^2) \leq (1+\varepsilon)\sum_{e\in E} l(e)c(e)$, which is equivalent to $(1-\varepsilon)\lambda l(G) \leq \sum_{e\in E} c(e)l(e)$.
To prove (12), note that the claim is trivial if $c(e) \geq \frac{\lambda}{1+\varepsilon}$. On the other hand, if $c(e) < \frac{\lambda}{1+\varepsilon}$, we have

$$\begin{aligned} l(e) = \exp(\alpha c(e)) &< \exp\left(\frac{\alpha\lambda}{1+\varepsilon}\right) = \left(\frac{m}{\varepsilon}\right)^{3/\varepsilon} = \left(\frac{m}{\varepsilon}\right)^{\varepsilon-2+\frac{3(1+\varepsilon)(1-\varepsilon/3)}{\varepsilon}} \\ &= \left(\left(\frac{m}{\varepsilon}\right)^{\varepsilon}\cdot\frac{1}{m}\right)\frac{\varepsilon^2}{m}\exp\left(\alpha\lambda(1-\varepsilon/3)\right). \end{aligned}$$

Since $\left(\frac{a}{x}\right)^x \leq a$ for $0 < x \leq 1$ and $a \geq \exp(1)$, we conclude

$$l(e) \leq \frac{\varepsilon^2}{m}\exp\left(\alpha\lambda\left(1-\frac{\varepsilon}{3}\right)\right) \overset{(10)}{\leq} \frac{\varepsilon^2}{m}\exp(\alpha c_{\max}) \leq \frac{\varepsilon^2}{m}l(G). \qquad \square$$

Let us consider the development of l and $c_{\max}$ due to gradually transferring congestion onto approximate minimum Steiner trees. We use $l^{(i)}$ and $c^{(i)}_{\max}$ to denote the length function and the maximum edge congestion at stage i of this process.

Lemma 2.12. *Let $\varepsilon \leq 1$. If there are stages i, j $(i < j)$, such that $c^{(i)}_{\max} = \lambda$ and $l^{(j)}(G) \leq l^{(i)}(G)$, then $c^{(j)}_{\max} \leq \left(1+\frac{\varepsilon}{3}\right)c^{(i)}_{\max}$.*

Proof. Suppose, $c_{\max}^{(j)} > \lambda\left(1+\frac{\varepsilon}{3}\right)$, then

$$\begin{aligned} l^{(j)}(G) &= \sum_{e\in E} l^{(j)}(e) > \exp(\alpha \cdot c_{\max}^{(j)}) > \exp\left(\alpha\lambda\left(1+\frac{\varepsilon}{3}\right)\right) \\ &= \left(\frac{m}{\varepsilon}\right)^{1+\varepsilon} \exp(\alpha\lambda) > m\exp(\alpha\lambda) \geq l^{(i)}(G). \end{aligned}$$ □

The following routine ImproveCongestion2 is the heart of Radzik's algorithm. As the input it takes sets of trees $\hat{\mathcal{T}}_i^{(0)}$ with congestion $c_i^{(0)}$ and a parameter ε determining the approximation quality.

Algorithm 2.13 (ImproveCongestion2)**.**
$\lambda := c_{\max}, \alpha := \frac{3(1+\varepsilon)\ln(m\varepsilon^{-1})}{\lambda\varepsilon}, \sigma := \frac{\varepsilon}{4\alpha\lambda}$
for $e \in E$
    $l(e) := \exp(\alpha \cdot c(e))$
do $LG := l(G)$
    for $i := 1$ *to* k
        $T_i := \tilde{MST}(S_i)$
        $Li\_neu := l(T_i)$, $Li\_alt := \sum_{e\in E} c_i(e)l(e)$
        if $Li\_alt - Li\_neu \geq \varepsilon \cdot Li\_alt$ *then*
            for $e \in E$
                $c_{\max} := 0$
                $c(e) := c(e) - c_i(e)$, $c_i(e) := c_i(e)\cdot(1-\sigma)$
                $c(e) := c(e) + c_i(e)$, $c_{\max} := \max\{c_{\max}, c(e)\}$
                $l(e) := \exp(\alpha \cdot c(e))$
            for $e \in E(T_i)$
                $c_i(e) := c_i(e) + \sigma$, $c(e) := c(e) + \sigma$, $l(e) := \exp(\alpha \cdot c(e))$
            $\hat{\mathcal{T}}_i := \hat{\mathcal{T}}_i \cup \{T_i\}$, *update* $c_{\max}$
while $c_{\max} > \left(1-\frac{\varepsilon}{3}\right)\lambda$ *and* $LG - l(G) > \frac{\varepsilon_2}{8} LG$

Theorem 2.14. *Let* $\varepsilon \leq \frac{1}{24}$*. Algorithm* ImproveCongestion2 *terminates with maximum congestion* $c_{\max}^{(q)} \leq \left(1-\frac{\varepsilon}{3}\right) c_{\max}^{(0)}$ *or* $c_{\max}^{(q)} \leq r(1+8\varepsilon)OPT$*.*

Before proving Theorem 2.14 let us analyze the running time.

Theorem 2.15. *Let* $\varepsilon \leq 1$*.* ImproveCongestion2 *takes* $O(\varepsilon^{-2}\ln n)$ *iterations of the* `while`*-loop to terminate.*

Proof. Suppose, ImproveCongestion2 needs more than $q := \frac{24}{\varepsilon^2}\ln\left(\frac{m}{\varepsilon}\right)$ iterations to terminate. Let $\hat{\mathcal{T}}_1^{(q)}, \ldots, \hat{\mathcal{T}}_k^{(q)}$ denote the sets of Steiner trees after q

iterations. In every, but the last, iteration, $l(G)$ is decreased by a factor $\geq \left(1-\frac{\varepsilon^2}{8}\right)$, so $l^{(q)}(G) < \left(1-\frac{\varepsilon^2}{8}\right)^{24\varepsilon^{-2}\ln(m/\varepsilon)} \cdot l^{(0)}(G) \leq \exp\left(-3\ln\left(\frac{m}{\varepsilon}\right)\right) \cdot l^{(0)}(G) = \left(\frac{\varepsilon}{m}\right)^3 \cdot l^{(0)}(G)$.

The qth iteration is not the last one. Hence, $c_{\max}^{(q)} > \left(1-\frac{\varepsilon}{3}\right)\lambda$ and

$$l^{(q)}(G) > \exp\left(\alpha\lambda\left(1-\frac{\varepsilon}{3}\right)\right) = \exp(-\alpha\varepsilon\lambda/3)\exp(\alpha\lambda)$$
$$= \left(\frac{\varepsilon}{m}\right)^{1+\varepsilon}\exp(\alpha\lambda) \geq \left(\frac{\varepsilon}{m}\right)^2 \cdot \frac{1}{m}l^{(0)}(G) \geq \left(\frac{\varepsilon}{m}\right)^3 l^{(0)}(G),$$

a contradiction. Consequently, ImproveCongestion2 must terminate within q iterations. Assuming $\varepsilon = \Omega\left(\frac{1}{\mathrm{poly}(n)}\right)$ implies $q = O(\varepsilon^{-2}\ln n)$. □

Using the ε-scaling technique described in the proof of Theorem 2.3 in the last section the following corollary is straightforward. (We first set $\varepsilon := 1/8$ constant and call the algorithm $O(\ln k)$ times to obtain $2r(= r(1+8\varepsilon))$-optimality. When we divide ε by 2, at most 24 calls result in $r(1+4\varepsilon)$-optimality. To see this, let z denote the number of calls. Substituting ε by $\varepsilon/2$, we want $r(1+16\varepsilon)\mathrm{OPT}\exp(-z\varepsilon/3) = r(1+8\varepsilon)\mathrm{OPT}$, which is equivalent to $\exp(z\varepsilon/3) = 1+\frac{8\varepsilon}{1+8\varepsilon} \leq 1+8\varepsilon$. From this we see $z \leq \frac{2}{\varepsilon}\ln(1+8\varepsilon) \leq 24$. Thus at most $O(\ln\varepsilon^{-1})$ additional calls are necessary. Since the running time of the last call dominates, the term $\ln\varepsilon^{-1}$ can be omitted from the estimate on the overall running time.)

Corollary 2.16. *a) $O\left(\ln k + \ln\frac{1}{\varepsilon}\right)$ calls to* ImproveCongestion2 *suffice to produce an $r(1+\varepsilon)$-optimal solution.*
b) The overall running time required is $O\left(\varepsilon^{-2}\cdot k\ln n\ln k(m+\beta)\right) = \tilde{O}(km\varepsilon^{-2})$ for the Steiner tree approximation by Mehlhorn.

It remains to prove Theorem 2.14. We split the proof into three lemmas.

Lemma 2.17. *Let $\varepsilon \leq 1$. Let the superscripts $(i-1)$ and (i) refer to values in subsequent iterations of the* `for`*-loop in* ImproveCongestion2*. Then*

$$l^{(i)}(G) \leq l^{(i-1)}(G), \tag{13}$$

$$c_{\max}^{(i)} \leq \left(1+\frac{\varepsilon}{3}\right)\lambda, \tag{14}$$

and, if the congestion changes for S_i,

$$\lambda\left(l^{(i-1)}(G) - l^{(i)}(G)\right) \geq \frac{\varepsilon}{8}\left(\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - l^{(i-1)}(T_i)\right). \tag{15}$$

(Note that c_i changes only in the ith iteration of the `for`*-loop; hence $c_i^{(j)} = c_i^{(0)}$ for all $j \in \{1, \ldots, i-1\}$. As before, we use T_i to represent the approximate Steiner tree chosen in the ith iteration.)*

Proof. The proof is by induction on the overall number of iterations of the `for`-loop. Consider the jth iteration of the `while`-loop and within this the ith iteration of the `for`-loop. By induction assumption, $c_{\max}^{(i-1)} \leq \left(1 + \frac{\varepsilon}{3}\right)\lambda$. (The induction start is trivial.) There is nothing to prove, if the congestion for S_i is unchanged, so suppose it does change. We have, for all $e \in E(T_i)$,

$$\begin{aligned} l^{(i)}(e) &= \exp(\alpha \cdot c^{(i)}(e)) = \exp(\alpha c^{(i-1)}(e) - \alpha\sigma c_i^{(0)}(e) + \alpha\sigma) \\ &= \exp\left(\alpha c^{(i-1)}(e)\right) \cdot \exp\left(-\alpha\sigma(c_i^{(0)}(e) - 1)\right). \end{aligned}$$

For $|\delta| \leq \varepsilon/3 \leq 1$ the estimate $\exp(-\delta) \leq 1 - \delta + \frac{3}{4}\delta^2 \leq 1 - \delta + \frac{\varepsilon}{4}|\delta|$ holds. Since $|\alpha\sigma(c_i^{(0)}(e) - 1)| \leq \frac{\varepsilon}{4\lambda}\max\{c_{\max}^{(i-1)}, 1\} \leq \frac{\varepsilon}{4\lambda}\left(1 + \frac{\varepsilon}{3}\right)\lambda \leq \frac{\varepsilon}{3}$, we can apply this estimate to obtain $l^{(i)}(e) \leq l^{(i-1)}(e) \cdot (1 - \alpha\sigma(c_i^{(0)}(e) - 1) + \frac{\varepsilon}{4}\alpha\sigma|c_i^{(0)}(e) - 1|)$. Similarly, for all $e \in E \setminus E(T_i)$, $l^{(i)}(e) \leq l^{(i-1)}(e) \cdot (1 - \alpha\sigma c_i^{(0)}(e) + \frac{\varepsilon}{4}\alpha\sigma c_i^{(0)}(e))$. Thus, $l^{(i)}(G) \leq l^{(i-1)}(G) - \alpha\sigma(\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e) - l^{(i-1)}(T_i)) + \frac{\varepsilon}{4}\alpha\sigma(\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e) + l^{(i-1)}(T_i))$. Using the fact that

$$\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e) - l^{(i-1)}(T_i) \geq \varepsilon \cdot \sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e), \tag{16}$$

which implies

$$l^{(i-1)}(T_i) \leq \sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e), \tag{17}$$

we conclude

$$\begin{aligned} l^{(i)}(G) &\overset{(17)}{\leq} l^{(i-1)}(G) - \alpha\sigma\left(\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e) - l^{(i-1)}(T_i) \right. \\ &\qquad \left. - \frac{\varepsilon}{2}\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e)\right) \\ &\overset{(16)}{\leq} l^{(i-1)}(G) - \frac{1}{2}\alpha\sigma\left(\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e) - l^{(i-1)}(T_i)\right) \\ &= l^{(i-1)}(G) - \frac{\varepsilon}{8\lambda}\left(\sum_{e\in E} c_i^{(0)}(e) l^{(i-1)}(e) - l^{(i-1)}(T_i)\right), \end{aligned}$$

which is equivalent to (15). Together with (16), inequality (15) yields (13). Finally, (14) follows from Lemma 2.12. □

Lemma 2.18. *Let $\varepsilon \leq 1$ and consider an arbitrary iteration of the* `while`*-loop. For iterations i, j of the* `for`*-loop, where $0 \leq i \leq j \leq k$ and for all $e \in E$ we have*

$$l^{(j)}(e) \geq \left(1 - \frac{\varepsilon}{3}\right) l^{(i)}(e).$$

Proof. Using $c^{(j)}(e) \geq c^{(i)}(e) - \sigma c^{(i)}(e)$, inequality (14) of Lemma 2.17 gives

$$\begin{aligned} l^{(j)}(e) &= \exp(\alpha c^{(j)}(e)) \geq \exp(\alpha c^{(i)}(e) - \alpha\sigma c^{(i)}(e)) \\ &= l^{(i)}(e) \exp\left(-\frac{\varepsilon}{4\lambda} c^{(i)}(e)\right) \geq l^{(i)}(e) \exp\left(-\frac{\varepsilon}{4\lambda} c^{(i)}_{\max}\right) \\ &\geq l^{(i)}(e) \exp\left(-\frac{\varepsilon}{4}\left(1 + \frac{\varepsilon}{3}\right)\right) \geq \left(1 - \frac{\varepsilon}{4}\left(1 + \frac{\varepsilon}{3}\right)\right) l^{(i)}(e) \\ &\geq \left(1 - \frac{\varepsilon}{3}\right) l^{(i)}(e). \end{aligned}$$

□

Lemma 2.19. *Let $\varepsilon \leq 1$. If in the end of an iteration of the* `while`*-loop we have $c_{\max} > \left(1 - \frac{\varepsilon}{3}\right) \lambda$ and*

$$l^{(0)}(G) - l^{(k)}(G) \leq \frac{\varepsilon^2}{8} l^{(0)}(G), \tag{18}$$

where superscripts refer to iterations of the `for`*-loop, then*

$$\sum_{i=1}^{k} l^{(k)}(\tilde{MST}(S_i)) \geq (1 - 4\varepsilon)\lambda l^{(k)}(G).$$

Proof. We are going to show

$$\sum_{i=1}^{k} l^{(k)}(\tilde{\mathrm{MST}}(S_i)) \geq \left(1 - \frac{\varepsilon}{3}\right) \sum_{i=1}^{k} l^{(i-1)}(T_i), \tag{19}$$

$$\sum_{i=1}^{k} l^{(i-1)}(T_i) \geq (1 - \varepsilon) \sum_{i=1}^{k} \sum_{e \in E} c_i^{(i-1)}(e) l^{(0)}(e) - \frac{4}{3}\varepsilon\lambda l^{(k)}(G), \tag{20}$$

$$\sum_{i=1}^{k} \sum_{e \in E} c_i^{(0)}(e) l^{(i-1)}(e) \geq \left(1 - \frac{4}{3}\varepsilon\right) \lambda l^{(k)}(G). \tag{21}$$

Conditional on these inequalities being true, we can estimate

$$\begin{aligned}
\sum_{i=1}^{k} l^{(k)}(\tilde{\mathrm{MST}}(S_i)) &\overset{(19)}{\geq} \left(1-\frac{\varepsilon}{3}\right)\sum_{i=1}^{k} l^{(i-1)}(T_i) \\
&\overset{(20)}{\geq} \left(1-\frac{\varepsilon}{3}\right)(1-\varepsilon)\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - \frac{4}{3}\varepsilon\lambda l^{(k)}(G) \\
&\overset{(21)}{\geq} \left(1-\frac{\varepsilon}{3}\right)(1-\varepsilon)\left(1-\frac{4}{3}\varepsilon\right)\lambda l^{(k)}(G) - \frac{4}{3}\varepsilon\lambda k^{(k)}(G) \\
&= \left(1-\varepsilon-\frac{\varepsilon}{3}+\frac{\varepsilon^2}{3}\right)\left(1-\frac{4}{3}\varepsilon\right)\lambda l^{(k)}(G) - \frac{4}{3}\varepsilon\lambda l^{(k)}(G) \\
&= \left(1-\frac{4}{3}\varepsilon+\frac{\varepsilon^2}{3}-\frac{4}{3}\varepsilon+\frac{16}{9}\varepsilon^2-\frac{4}{9}\varepsilon^3-\frac{4}{3}\varepsilon\right)\lambda l^{(k)}(G) \\
&\geq (1-4\varepsilon)\lambda l^{(k)}(G).
\end{aligned}$$

To prove (19), let $\tilde{T}_i$ be an approximate minimum Steiner tree for $l^{(k)}$ (remember, we assume T_i to be the minimum Steiner approximation computed for $l^{(i-1)}$). By Lemma 2.18,

$$\sum_{i=1}^{k} l^{(k)}(\tilde{\mathrm{MST}}(S_i)) = \sum_{i=1}^{k} l^{(k)}(\tilde{T}_i) \geq \sum_{i=1}^{k}\left(1-\frac{\varepsilon}{3}\right) l^{(i-1)}(\tilde{T}_i) \geq \sum_{i=1}^{k}\left(1-\frac{\varepsilon}{3}\right) l^{(i-1)}(T_i).$$

For (20), note that if the congestion does not change for S_i, then

$$\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - l^{(i-1)}(T_i) < \varepsilon\cdot\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e),$$

otherwise Lemma 2.17 yields

$$\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - l^{(i-1)}(T_i) \leq \frac{8}{\varepsilon}\lambda(l^{(i-1)}(G) - l^{(i)}(G)).$$

Altogether,

$$\begin{aligned}
\sum_{i=1}^{k} l^{(i-1)}(T_i) &\geq \sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - \varepsilon\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e) \\
&\quad - \sum_{i=1}^{k}\frac{8}{\varepsilon}\lambda\left(l^{(i-1)}(G) - l^{(i)}(G)\right) \\
&= (1-\varepsilon)\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - \frac{8}{\varepsilon}\lambda\left(l^{(0)}(G) - l^{(k)}(G)\right) \\
&\overset{(18)}{\geq} (1-\varepsilon)\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - \varepsilon\lambda l^{(0)}(G) \\
&\geq (1-\varepsilon)\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) - \frac{4}{3}\varepsilon\lambda l^{(k)}(G),
\end{aligned}$$

where the last inequality follows from (18) and the fact that $\varepsilon \leq 1$, *i.e.*

$$l^{(0)}(G) \leq l^{(k)}(G)\cdot\left(1-\frac{\varepsilon^2}{8}\right)^{-1} \text{ and } \left(1-\frac{\varepsilon^2}{8}\right)^{-1} = \frac{8}{8-\varepsilon^2} \leq \frac{8}{7} \leq \frac{4}{3}.$$

To prove (21) we use Lemma 2.18 for estimating

$$\begin{aligned}
\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(i-1)}(e) &\geq \left(1-\frac{\varepsilon}{3}\right)\sum_{i=1}^{k}\sum_{e\in E} c_i^{(0)}(e)l^{(0)}(e) \\
&= \left(1-\frac{\varepsilon}{3}\right)\sum_{e\in E} c^{(0)}(e)l^{(0)}(e).
\end{aligned}$$

By Lemma 2.11,

$$\left(1-\frac{\varepsilon}{3}\right)\sum_{e\in E} c^{(0)}(e)l^{(0)}(e) \geq \left(1-\frac{\varepsilon}{3}\right)(1-\varepsilon)\lambda l^{(0)}(G) \geq \left(1-\frac{4}{3}\varepsilon\right)\lambda l^{(0)}(G).$$

Since inequality (13) of Lemma 2.17 implies $l^{(0)}(G) \geq l^{(k)}(G)$, the claim of (21) is proven. □

Proof of Theorem 2.14. Suppose, that ImproveCongestion2 terminates with $c_{\max}^{(q)} > \left(1-\frac{\varepsilon}{3}\right)c_{\max}^{(0)}$. Then, $\sum_{i=1}^{k} l^{(k)}(\tilde{\mathrm{M}}\mathrm{ST}(S_i)) \geq (1-4\varepsilon)\lambda l^{(k)}(G)$, by Lemma 2.19, and because of Lemma 2.17 we may apply Theorem 2.9 to conclude that $c_{\max}^{(q)} \leq r(1+8\varepsilon)\mathrm{OPT}$. □

2.2.3. *The Garg-Könemann approach*

The algorithm of Garg and Könemann [3] differs from the previous algorithms in the fact that there is no transfer of congestion from $\tilde{\mathrm{M}}\mathrm{ST}(S_i)^{(j)}$ to $\tilde{\mathrm{M}}\mathrm{ST}(S_i)^{(j+1)}$. Instead, the functions describing the congestion are built up from scratch. A feasible solution is obtained by scaling the computed quantities in the very end.

Algorithm 2.20 (ImproveCongestion3).
for $e \in E$
  $l(e) := \delta$
while $\sum_{e\in E} l(e) < \xi$
  for $i := 1$ *to* k
    $T := \tilde{MST}(S_i)$, $c_i(T) := c_i(T) + 1$
    for $e \in T$
      $l(e) := l(e) \cdot \left(1 + \frac{\varepsilon}{u}\right)$
scale $\bar{c}_i(T) := c_i(T)/\#iterations$

Lemma 2.21. *Suppose that $l(e)$ is initialized to some constant δ for all $e \in E$. Let q be the number of iterations of the* `while`*-loop* ImproveCongestion3 *needs to terminate. Then* $\bar{c}_{\max}^{(q)} < \frac{u\cdot\ln(\xi/\delta)}{(q-1)\cdot\ln(1+\varepsilon)}$ *and* $q \le \left\lceil \frac{u}{OPT} \cdot \frac{\ln(\xi/\delta)}{\ln(1+\varepsilon)} \right\rceil$.

Proof. Whenever $c(e)$ increases by u, $l(e)$ grows by a factor of at least $(1+\varepsilon)$. Let z count the number of times this happens. Since $l^{(q-1)}(e) < \xi$, we have $\delta(1+\varepsilon)^z < \xi$, so $z < \frac{\ln(\xi/\delta)}{\ln(1+\varepsilon)}$. After $q-1$ iterations, $\sum_{T\in\hat{\mathcal{T}}_i} c_i(T) = q-1$ for all $i \in \{1,\dots,k\}$. Thus $(\bar{c}_{\max}, (\bar{c}_i(T)_{i\in\{1,\dots,k\}})_{T\in\mathcal{T}_i}) := \left(\frac{c_{\max}}{q-1}, \left(\frac{c_i(T)}{q-1}\right)_{i\in\{1,\dots,k\}}\right)_{T\in\mathcal{T}_i}$ is a feasible solution satisfying $\bar{c}_{\max} \le \frac{u\cdot z}{q-1} < \frac{u\cdot\ln(\xi/\delta)}{(q-1)\ln(1+\varepsilon)}$. The second estimate follows since $1 \le \frac{\bar{c}_{\max}}{\mathrm{OPT}} < \frac{u\ln(\xi/\delta)}{(q-1)\ln(1+\varepsilon)\mathrm{OPT}}$. □

To achieve an $r(1+\varepsilon)$OPT-approximation, we must fix the free parameters δ, ξ and u appropriately.

Lemma 2.22. *Let $\varepsilon \le \frac{1}{36}$. For $u < 2OPT$ and $\xi := \delta \cdot \left(\frac{m}{1-2\varepsilon r}\right)^{\frac{1+\varepsilon}{\varepsilon}}$ the algorithm terminates with $\bar{c}_{\max} < r(1+6\varepsilon)OPT$.*

Proof. The way the length function is updated together with Corollary 2.2 lets us estimate $l^{(q)}(G) \le l^{(q-1)}(G) + \frac{\varepsilon}{u} r \sum_{i=1}^{k} l^{(q)}(\mathrm{MST}(S_i)) \le l^{(q-1)}(G) + \frac{\varepsilon}{u} r\mathrm{OPT} \cdot l^{(q)}(G)$. Hence, $l^{(q)}(G) \le \frac{l^{(q-1)}(G)}{1-\frac{\varepsilon}{u}r\mathrm{OPT}} \le \frac{m\delta}{\left(1-\frac{\varepsilon}{u}r\cdot\mathrm{OPT}\right)^q}$, as we initialized $l^{(0)}(e) := \delta$ for all $e \in E$. By the choice of u and since $1 + x \le \exp(x)$ for all x, we have,

$$\begin{aligned} l^{(q)}(G) &\le \frac{m\delta}{1-\frac{\varepsilon}{u}r\cdot\mathrm{OPT}} \left(\frac{u}{u-\varepsilon r\cdot\mathrm{OPT}}\right)^{q-1} \le \frac{m\delta}{1-2\varepsilon r} \left(1 + \frac{\varepsilon r\mathrm{OPT}}{u-\varepsilon r\cdot\mathrm{OPT}}\right)^{q-1} \\ &\le \frac{m\delta}{1-2\varepsilon r} \exp\left(\frac{\varepsilon r\cdot\mathrm{OPT}(q-1)}{u-\varepsilon r\mathrm{OPT}}\right) \le \frac{m\delta}{1-2\varepsilon r} \exp\left(\frac{q-1}{u}\cdot\frac{\varepsilon r\mathrm{OPT}}{1-2\varepsilon r}\right). \end{aligned}$$

Since $l^{(q)}(G) \geq \xi$, $\xi \leq \frac{m\delta}{1-2\varepsilon r} \exp\left(\frac{q-1}{u} \cdot \frac{\varepsilon r \mathrm{OPT}}{1-2\varepsilon r}\right)$, and thus $\frac{q-1}{u} \geq \ln\left(\frac{\xi(1-2\varepsilon r)}{m\delta}\right) \cdot \frac{1-2\varepsilon r}{\varepsilon r \mathrm{OPT}}$. Now, using Lemma 2.21,

$$\bar{c}_{\max} \leq \frac{u \ln(\xi/\delta)}{(q-1)\ln(1+\varepsilon)} \leq r\mathrm{OPT}\left(\frac{\varepsilon}{(1-2\varepsilon r)\ln(1+\varepsilon)} \cdot \frac{\ln(\xi/\delta)}{\ln\left(\frac{\xi(1-2\varepsilon r)}{m\delta}\right)}\right).$$

If we take ξ such that

$$\frac{\ln(\xi/\delta)}{\ln\left(\frac{\xi(1-2\varepsilon r)}{m\delta}\right)} \leq 1+\varepsilon, \tag{22}$$

and bound $\ln(1+\varepsilon)$ by its second order Taylor expansion, we obtain

$$\begin{aligned}
\bar{c}_{\max} &\leq r\mathrm{OPT}\left(\frac{\varepsilon(1+\varepsilon)}{(1-2\varepsilon r)\left(\varepsilon - \frac{\varepsilon^2}{2}\right)}\right) = \frac{1+\varepsilon}{1-\frac{\varepsilon}{2}-2\varepsilon r+\varepsilon^2 r} r\mathrm{OPT} \\
&= 1 + \frac{\frac{3}{2}\varepsilon + 2\varepsilon r - \varepsilon^2 r}{1-\frac{\varepsilon}{2}-2\varepsilon r+\varepsilon^2 r} r\mathrm{OPT} \leq 1 + \frac{\frac{3}{2}\varepsilon + 4\varepsilon}{1-\frac{\varepsilon}{2}-4\varepsilon} r\mathrm{OPT} \text{ for } r \leq 2 \\
&= 1 + \frac{11\varepsilon}{2-9\varepsilon} r\mathrm{OPT} \leq 1 + 6\varepsilon r\mathrm{OPT},
\end{aligned}$$

by the choice of ε. Note that (22) holds for our choice of ξ. □

Theorem 2.23. *An $r(1+6\varepsilon)$OPT-approximate solution to the fractional minimum multicast problem can be obtained in time $O(\beta\varepsilon^{-2}k \cdot \ln k \ln m) = \tilde{O}(\varepsilon^{-2}km)$ using the Steiner tree approximation by Mehlhorn.*

Proof. We repeatedly call ImproveCongestion3 in the following u-scaling process. Throughout the process we maintain the condition $u < 2\mathrm{OPT}$ to make sure that the algorithm terminates with the claimed guarantee. So all we have to worry about is the number of iterations. The first call is performed with $u := \frac{k}{2}$. By Lemma 2.22, if $u \leq \mathrm{OPT}$, the algorithm terminates within $\left\lceil \frac{\ln(\xi/\delta)}{\ln(1+\varepsilon)} \right\rceil$ iterations. Otherwise, we know that $\mathrm{OPT} < \frac{k}{2}$ and restart the algorithm with $u := \frac{k}{4}$. Again, ImproveCongestion3 either terminates in $\left\lceil \frac{\ln(\xi/\delta)}{\ln(1+\varepsilon)} \right\rceil$ iterations or we may conclude that $\mathrm{OPT} < \frac{k}{4}$ and restart the algorithm with $u := \frac{k}{8}$. Repeating this at most $O(\ln k)$ times yields the claim, since $\frac{\ln(\xi/\delta)}{\ln(1+\varepsilon)} = O(\varepsilon^{-2} \ln m)$ and because one iteration takes $O(k(m+\beta))$-time. □

Unfortunately, the above u-scaling process is *not practically efficient*, since the number of iterations in each scaling phase is considerably large (note that we need $\varepsilon \leq \frac{1}{36}$ to guarantee the approximation quality, so even for moderate values of m, say $m := 500$, we may have to await over 8000 iterations before we can decide whether or not $\mathrm{OPT} < u$). However, without u-scaling the running time

increases by a factor of $k/\ln k$. Again, we may (practically) improve the running time by using $\bar{c}_{\max} < (1+6\varepsilon)\sum_{i=1}^{k} l(\tilde{\mathrm{MST}}(S_i))/l(G)$ instead of $l(G) \geq \xi$ as the termination criterion. It is an open question whether or not ε-scaling as described in the previous sections can be advantageously applied. (Our experiments strongly indicate that ε-scaling substantially decreases the running time, but an analysis is still missing.) The following implementation takes care of the fact that due to limited precision it may be necessary to rescale the edge lengths.

Algorithm 2.24 (ApproximateCongestion3)**.**
$maxdual := 0$, $minprimal := k$, $c_{\max} := 0$, $iteration := 0$, $LG := m$
for $e \in E$
    $c(e) := 0$, $l(e) := 1$
    do $LM := 0$
        for $i := 1$ *to* k
            $T_i := \tilde{MST}(S_i)$, $c(T_i) := c(T_i) + 1$
            for $e \in E(T_i)$
                $LG := LG - l(e)$
                $l(e) := l(e) \cdot \left(1 + \frac{\varepsilon}{k}\right)$, $c(e) := c(e) + 1$
                $c_{\max} := \max\{c_{\max}, c(e)\}$
                $LG := LG + l(e)$, $LM := LM + l(e)$
        $iteration := iteration + 1$, $dual := \frac{LM}{LG}$
        $minprimal := \min\{minprimal, \frac{c_{\max}}{iteration}\}$
        $maxdual := \max\{maxdual, dual\}$
        if $LG > 100000$
            $LG := \frac{LG}{100000}$
            for $e \in E$
                $l(e) := \frac{l(e)}{100000}$
    while $minprimal \geq (1+6\varepsilon) maxdual$
$c_{\max} := \frac{c_{\max}}{iteration}$,
for $T \in \hat{\mathcal{T}}_i$
    $c(T) := \frac{c(T)}{|\hat{\mathcal{T}}_i|}$

2.3. Approximating the integral optimum

2.3.1. *Pure combinatorial approach*

Klein *et al.* [9] describe an approximation algorithm for the unit capacity concurrent flow problem that can be modified to approximate an integral solution to our problem without the necessity to round. The algorithm is similar to the one presented in Section 2.2.1. However, instead of updating all trees in one iteration of the while loop, only one "bad" tree per iteration is modified.

Definition 2.25. A tree $T \in \mathcal{T}_i$ is r-bad for S_i if $(1-\varepsilon')l(T) > rl(\mathrm{MST}(S_i)) + \varepsilon' \cdot \frac{c_{\max} \cdot l(G)}{k}$.

This allows us to choose σ by a factor of k larger than in the algorithms described previously, and it can be shown that consequently σ never needs to be reduced below 1, if $\mathrm{OPT} = \Omega(\log n)$.

Theorem 2.26. *An integral solution to the minimum multicast congestion problem with congestion $OPT \cdot O(\sqrt{OPT \cdot \log n})$ can be found in time $O(k^2(m+\beta)\log k)$.*

Aspes *et al.* [1] give an online algorithm for approximating our problem within a factor of $O(\log n)$. Since they explicitly consider the multicast congestion problem, we do not restate their algorithm but instead analyze a simpler online algorithm for which the same approximation bound applies, if the value of an optimal solution is known in advance.

Algorithm 2.27 (OnlineCongestion).
for $e \in E$
    $l(e) := 1$, $c(e) := 0$
for $i = 1$ *to* k
    $T_i := \tilde{MST}(S_i)$
    for $e \in E(T_i)$
        $l(e) := l(e) \cdot A$, $c(e) := c(e) + 1$
return $T_1, \ldots, T_k, c_{\max}$

Theorem 2.28. *For $A := 1 + \frac{1}{2rOPT}$, the above algorithm terminates with maximal congestion $O(rOPT \cdot \log(n))$.*

Proof. Let $l^{(j)}$ denote the length function at the end of the jth iteration of the `for`-loop. From $l^{(j)}(G) = l^{(j-1)}(G) + (A-1)l^{(j-1)}(\tilde{\mathrm{MST}}_{l^{(j-1)}}(S_{i-1}))$ we get $l^{(k)}(G) = m + (A-1)\cdot\sum_{i=1}^{k} l^{(i-1)}(\tilde{\mathrm{MST}}_{l^{(i-1)}}(S_i)) \le m + (A-1)\cdot\sum_{i=1}^{k} l^{(k)}(\tilde{\mathrm{MST}}_{l^{(k)}}(S_i)) \le m + (A-1)\cdot r\mathrm{OPT}l^{(k)}(G)$ by Corollary 1. Since $l^{(k)}(G) \ge A^{c_{\max}-1}$, the claim follows by the choice of A. □

2.3.2. *Derandomization*

A deterministic approximation satisfying the bounds of Theorem 2.3 can be proved with the tools of Srivastav and Stangier [16]. The proof given in [16] has to be slightly modified, as we have to replace the Angluin-Valiant inequality by Raghavan's stronger bound on the deviation of the sum of Bernoulli trials from its expected value [15]. The crucial difference is that in computing the pessimistic estimators we require an approximate solution to the equality $\frac{1}{m} = \left(\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right)^{\mathrm{OPT}}$, *i.e.* we want to determine δ such that $\left(\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right)^{\mathrm{OPT}} \in \left[\frac{1}{2m}, \frac{1}{m}\right]$ or equivalently $0 \le f(\delta) := \ln(2m) + \mathrm{OPT}(\delta - (1+\delta)\ln(1+\delta)) \le \ln 2$. Note that f is a monotone function of δ with $f(0) = \ln 2 + \ln m > \ln 2$ and $f(6 + \ln 2m) \le$

$\ln(2m) + (6 + \ln(2m) - (7 + \ln(2m)) \cdot 2) \leq -8$. Hence, an approximate solution can be found in time $O(\ln \ln m)$ by binary search.

3. A new and practically efficient implementation

In practical experiments the following algorithm was superior to all theoretically efficient approximation algorithms.

Algorithm 3.1 (PracticalCongestion).
$\lambda := 1,\ c(e) := 0$ *for all* $e \in E$
for $i := 1$ *to* k
    for $e \in E$
        $l(e) := n^{\frac{c(e)}{\lambda} - 1}$
    $T_i := \tilde{M}ST(S_i)$
    for $e \in E(T_i)$
        $c(e) := c(e) + 1,\ \lambda = \max\{\lambda, c(e)\}$
$iteration := 1$
while $(iteration \leq 100)$
    for $i := 1$ *to* k
        $A := |E(T_i)|$
        for $e \in E$
            $l(e) := A^{c(e) - c_{\max}}$
        for $e \in E(T_i)$
            $l(e) := l(e)/A$
        $T_i' := \tilde{M}ST(S_i)$
        for $e \in E(T_i)$
            $l(e) := l(e) \cdot A$
        if $l(T_i) > l(T_i')$ *then* $T_i := T_i'$
        update $c(e)$
    $iteration := iteration + 1$

The algorithm uses different length functions for determining a start set of Steiner trees and for updating trees. The first length function is similar to the one in Aspnes *et al.*'s online algorithm, but uses a base of n instead of a base $\in (1, 1.5]$. In fact, we found that the quality of the solutions increases substantially as the base grows. To improve near optimal solutions it turned out to be of advantage to replace the exponent $c(e)/c_{\max}$ in the length function by $c(e) - c_{\max}$, thereby increasing the impact of highly congested edges. On the other hand, the exponential base must be neither too small nor too large to "correctly" take into account edges with lower congestion. Here, the size of the current best tree (experimentally) proved to be a good choice.

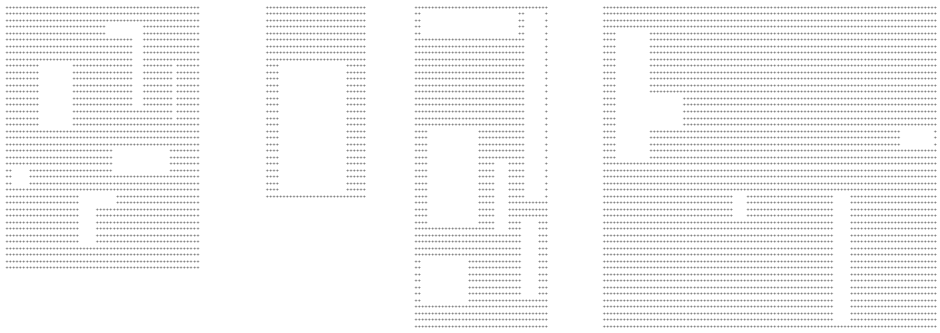

FIGURE 1. The four test instances.

4. Experimental results

We tested our new algorithm against Aspnes *et al.*'s online algorithm and a version of ApproximateCongestion3, which is (together with ApproximateCongestion2) theoretically fastest. As discussed in Section 2, we did not implement u-scaling for reasons of performance. Instead, we provided ApproximateCongestion3 with the (near) optimal value of u computed by our new algorithm. ε was set to its theoretically maximum value of $1/36$. The number of iterations was restricted to 100. We ran the online algorithm with various values of $A \in (1, 1.5]$ and listed the best outcome (which – except for three cases – occurred for $A = 1.5$. In fact, choosing A larger than theoretically allowed always decreased $c_{\max}$ dramatically). The values for ApproximateCongestion3 refer to the fractional relaxation.

Our test instances were grid graphs with rectangular holes (see Fig. 1). Such grid graphs typically arise in the design of VLSI logic chips where the holes represent big arrays on the chip. They are considered as hard instances for path- as well as for tree-packing problems. Multicast requests of size 50 to 1000 were chosen by a random generator[1]. Columns 3–5 show the computed congestion and in $\widetilde{\text{brackets}}$ the minimum number of iterations that gave the tabulated congestion. $\widetilde{\text{Dual}}$ approximates $\max\{\sum_{i=1}^{k} l(\tilde{\text{MST}}(S_i)/l(G) \mid l : E \to \mathbb{R}_{\geq 0}\}$, where $\tilde{\text{MST}}$ is computed by Mehlhorn's algorithm (*i.e.* $\widetilde{\text{Dual}} \leq 2 \cdot \text{OPT}$). The superior performance of our heuristic algorithm is evident.

Acknowledgement. We would like to thank the referees for their detailed comments.

[1]Interested readers can download our test examples and the source codes of our algorithms from `http://www.numerik.uni-kiel.de/~asrdfg9/multicast/`

TABLE 1. Experimental comparison of the algorithms.

grid	#requests/ #terminals	ONLINE	APPROXIMATE CONG.3 (#it.)	New Alg. (# it.)	$\widetilde{\text{Dual}} \geq$
1	50 / 4	11	12 (1), 2.5 (100)	2 (1)	1.04
1	100 / 4	16	17 (1), 4.4 (100)	3 (1)	2.01
1	150 / 4	29	29 (1), 6.1 (100)	5 (1), 4 (2)	2.86
1	200 / 4	43	44 (1), 8.0 (100)	7 (1), 5 (2)	3.85
1	300 /4	55	57 (1), 11.5 (100)	9 (1), 7 (3)	5.77
1	500 / 4	78	79 (1), 19.9 (100)	16 (1), 12 (2)	10.00
1	1000 / 4	156	161 (1), 36.5 (100)	29 (1), 21 (69)	20.00
1	2000 / 4	373	383 (1), 76.1 (100)	60 (1), 44 (2)	41.00
1	500 / 2–100	178	178 (1), 69.1 (100)	41 (1), 32 (9)	31.00
1	1000 / 2–100	326	326 (1), 100.5 (100)	79 (1), 65 (3)	64.00
2	50 / 5–10	15	16 (1), 7.2 (100)	7 (1), 5 (2)	4.01
2	50 / 10–20	16	18 (1), 8.6 (100)	7 (1), 6 (2)	4.58
2	100 / 5–15	32	33 (1), 15.2 (100)	12 (1), 9 (3)	8.08
2	100 / 10–20	37	38 (1), 17.6 (100)	4 (1), 11 (4)	9.94
2	150 / 10–20	57	57 (1), 24.7 (100)	22 (1), 15 (4)	13.37
2	150 / 30–50	55	55 (1), 29.2 (100)	27 (1), 21 (4)	19.33
2	200 / 10–20	64	69 (1), 36.7 (100)	34 (1), 29 (6)	27.87
2	200 / 30–50	65	65 (1), 38.6 (100)	34 (1), 27 (9)	25.78
2	300 / 10–20	115	115 (1), 48.58 (100)	41 (1), 29 (4)	27.03
2	300 / 30–50	116	116 (1), 59.94 (100)	51 (1), 40 (15)	38.69
4	50 / 5–10	13	14 (1), 3.5 (100)	3 (1)	1.85
4	50 / 20–100	13	13 (1), 7.4 (100)	7 (1), 6 (2)	4.54
4	100 / 5–10	19	20 (1), 7.0 (100)	6 (1), 5 (2)	3.44
4	100 / 20–100	28	28 (1), 13.5 (100)	13 (1), 10 (8)	9.03
4	200 / 5–10	35	35 (1), 12.0 (100)	10 (1), 8 (6)	6.66
4	200 / 20–100	46	46 (1), 28.0 (100)	24 (1), 19 (42)	18.01
4	300 / 5–10	66	71 (1), 20.0 (100)	17 (1), 12 (2)	10.05
4	300 / 20–100	67	67 (1), 41.0 (100)	35 (1), 28 (10)	26.23
4	500 / 5–10	83	86 (1), 31.5 (100)	26 (1), 19 (2)	16.15
4	500 / 20–100	114	115 (1), 67.6 (100)	59 (1), 46 (23)	44.46
3	50 / 5–10	11	11 (1), 2.8 (100)	2 (1)	1.01
3	50 / 20–100	12	12 (1), 4.3 (100)	4 (1), 3 (20)	2.40
3	100 / 5–10	15	14 (1), 2.4 (100)	4 (1), 3 (2)	2.02
3	100 / 20–100	23	25 (1), 9.0 (100)	7 (1), 6 (3)	4.94
3	200 / 5–10	28	29 (1), 7.9 (100)	6 (1), 5 (2)	3.77
3	200 / 20–100	44	44 (1), 16.9 (100)	15 (1), 11 (3)	9.34
3	300 / 5–10	41	41 (1), 11.5 (100)	8 (1), 7 (2)	5.43
3	300 / 20–100	56	57 (1), 24.8 (100)	21 (1), 16 (4)	14.10
3	500 / 5–10	72	76 (1), 18.5 (100)	15 (1), 11 (4)	9.06
3	500 / 20–100	98	97 (1), 41.6 (100)	33 (1), 26 (4)	23.56

References

[1] J. Aspnes, Y. Azar, A.Fiat, S. Plotkin and O. Waarts, On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. Association Computing Machinery* **44** (1997) 486–504.

[2] R. Carr and S. Vempala, Randomized Metarounding, in *Proc. of the 32nd ACM Symposium on the theory of computing (STOC '00)*, Portland, USA (2000) 58–62.

[3] N. Garg, J. Könemann, Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems, in *Proc. 39th IEEE Annual Symposium on Foundations of Computer Science* (1998) 300–309.

[4] A.V. Goldberg, A natural randomization strategy for multicommodity flow and related algorithms. *Inform. Process. Lett.* **42** (1992) 249–256.

[5] A.V. Goldberg, A.D. Oldham, S. Plotkin and C. Stein, An Implementation of a Combinatorial Approximation Algorithm for Minimum-Cost Multicommodity Flows, in *Proc. 6th Conf. on Integer Prog. and Combinatorial Optimization* (1998) 338–352.

[6] M.D. Grigoriadis and L.G. Khachiyan, Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM J. Optim.* **4** (1994) 86–107.

[7] K. Jansen and H. Zhang, An approximation algorithm for the multicast congestion problem *via* minimum Steiner trees, in *Proc. 3rd Int. Worksh. on Approx. and Random. Alg. in Commun. Netw. (ARANCE'02)*, Roma, Italy, September 21. Carleton Scientific (2002) 77–90.

[8] K. Jansen and H. Zhang, Approximation algorithms for general packing problems with modified logarithmic potential function, in *Proc. 2nd IFIP Int. Conf. on Theoretical Computer Science (TCS'02)*, Montréal, Québec, Canada, August 25–30 (2002).

[9] P. Klein, S. Plotkin, C. Stein and E. Tardos, Faster Approximation Algorithms for the Unit Capacity Concurrent Flow Problem with Applications to Routing and Finding Sparse Cuts. *SIAM J. Comput.* **23** (1994) 466–487.

[10] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos and S. Tragoudas, Fast approximation algorithms for multicommodity flow problems. *J. Comp. Syst. Sci.* **50** (1995) 228–243.

[11] D.W. Matula and F. Shahrokhi, The maximum concurrent flow problem. *J. Association Computing Machinery* **37** (1990) 318–334.

[12] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs. *Inform. Process. Lett.* **27** (1998) 125–128.

[13] S. Plotkin, D. Shmoys and E. Tardos, Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.* **20** (1995) 257–301.

[14] T. Radzik, Fast deterministic approximation for the multicommodity flow problem. *Math. Prog.* **78** (1997) 43–58.

[15] P. Raghavan, Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comp. Syst. Sci.* **38** (1994) 683–707.

[16] A. Srivastav and P. Stangier, On complexity, representation and approximation of integral multicommodity flows. *Discrete Appl. Math.* **99** (2000) 183–208.

[17] S. Vempala and B. Vöcking, Approximating Multicast Congestion, in *Proc. 10th ISAAC*, Chennai, India (1999) 367–372.

[18] G. Robins and A. Zelikovsky, Improved Steiner tree approximation in graphs, in *Proc. of the 11th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA 2000)* (2000) 770–779.