T U M

I N S T I T U T F Ü R I N F O R M A T I K

RNC-Approximation Algorithms for the Steiner Problem

Hans Jürgen Prömel
Angelika Steger

TUM-I9633
Dezember 96

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-12-I9633-50/1.-FI

Druck: Fakultät für Mathematik und
Institut für Informatik der
Technischen Universität München

RNC-Approximation Algorithms for the Steiner Problem

Hans Jürgen Prömel
Institut für Informatik, Humboldt Universität zu Berlin,
10099 Berlin, Germany; *proemel@@informatik.hu-berlin.de*

Angelika Steger
Institut für Informatik, TU München,
80290 München, Germany; *steger@@informatik.tu-muenchen.de*

Abstract. In this paper we present an $\mathcal{RNC}$-algorithm for finding a minimum spanning tree in a weighted 3-uniform hypergraph, assuming the edge weights are given in unary, and a fully polynomial time randomized approximation scheme if the edge weights are given in binary. From this result we then derive $\mathcal{RNC}$-approximation algorithms for the Steiner problem in networks with approximation ratio $(1+\epsilon)\,5/3$ for all $\epsilon > 0$.

1 Introduction

In recent years, the Steiner tree problem in graphs attracted considerable attention, as well from the theoretical point of view as from its applicability, e.g., in VLSI-layout. It is rather easy to see and has been known for a long time that a minimum Steiner tree spanning a given set of terminals in a graph or network can be approximated in polynomial time up to a factor of 2, cf. e.g. Choukhmane [6] or Kou, Markowsky, Berman [14]. After a long period without any progress Zelikovsky [23], Berman and Ramaiyer [2], Zelikovsky [24], and Karpinski and Zelikovsky [13] improved the approximation factor step by step from 2 to 1.644.

In this paper we present $\mathcal{RNC}$-approximation algorithm for the Steiner problem with approximation ratio $(1+\epsilon)\,5/3$ for all $\epsilon > 0$. The running time of these algorithms is polynomial in ϵ^{-1} and n. Our algorithms also give rise to conceptually much easier and faster (though randomized) sequential approximation algorithms than the Karpinski-Zelikovsky approach which almost match their approximation factor.

The core of our algorithm is a new $\mathcal{RNC}$-algorithm for finding a minimum spanning tree in 3-uniform hypergraphs. The problem of finding a minimum spanning tree in a given graph is well studied and known to be sequentially solvable almost in linear time [10]. Moreover, this problem can also be solved efficiently in parallel [21]. On the other hand, the problem of finding a minimum spanning tree in a k-uniform hypergraph is known to be $\mathcal{NP}$-hard whenever $k \geq 4$. The status of the case $k = 3$, however, is not yet completely decided. For the unweighted case Lovász [16] provided a very complicated $\mathcal{O}(n^{17})$ algorithm, which was later improved to a $\mathcal{O}(n^4)$ algorithm by Gabow and Stallmann [11]. Here, n denotes the number of vertices of the hypergraph. Also, Lovász [17] presented a conceptionally simple randomized algorithm for the unweighted case by reducing it to a sequence of computations of determinants of appropriate matrices. For the weighted case Camerini, Galbiati and Maffioli [5] generalized the approach of Lovász [17] to obtain a randomized pseudo-polynomial time algorithm.

In this paper we present an $\mathcal{RNC}$-algorithm for finding a minimum spanning tree in a weighted 3-uniform hypergraph with edge-weights in *unary*. This algorithm simplifies to an $\mathcal{RNC}$-algorithm for deciding the existence of a spanning tree in an unweighted 3-uniform hypergraph and implies a fully polynomial approximation scheme if the edge-weights are given in binary.

To achieve this result, we combine ideas from Mulmuley, Vazirani, Vazirani [18] which they used to obtain an $\mathcal{RNC}$-algorithm for finding a perfect matching in a given graph with ideas from Lovász [17] resp. Camerini, Galbiati and Maffioli [5], where they presented a random pseudo-polynomial time algorithm for the general problem of finding a base of specified value in a weighted represented matroid subject to parity conditions.

2 Minimum Spanning Trees in 3-Uniform Hypergraphs

A *hypergraph* $H = (V, F)$ is a generalization of graphs where F is an arbitrary family of subsets of V (and not just a family of 2-element subsets). An r-*uniform* hypergraph is a hypergraph all of whose edges have cardinality exactly r.

Many notions and results of graph theory generalize to hypergraphs. Here we just need cycles and trees. A *cycle* (of length $l \geq 2$) in H is a sequence $x_1, e_1, \ldots, x_{l-1}, e_{l-1}, x_l, e_l$ of vertices and edges such that the x_i are distinct vertices, the e_i are distinct edges, $x_1 \in e_1 \cap e_l$ and $x_i \in e_{i-1} \cap e_i$ for all $i = 2, \ldots, l$. A hypergraph H is a tree iff H is connected and contains no cycles. A *spanning tree* of a hypergraph H is a subhypergraph T of H that is a tree and satisfies $V(T) = V(H)$. (In contrary to graphs, not every connected hypergraph contains a spanning tree!)

Minimum Spanning Tree Problem (MST)
Input: A weighted hypergraph $H = (V, F; w)$, where $w : F \to \mathbb{N}_0$,
Output: Find a spanning tree of H of minimum weight.

Restricted to 2-uniform hypergraphs (that is, to graphs) the minimum spanning tree problem is easily solved efficiently sequentially as well as in parallel. On the other hand, a trivial reduction from Exact Cover by 3-Sets shows that for unweighted 4-uniform hypergraphs even deciding whether there *exists* a spanning tree is $\mathcal{NP}$-complete. The status of the minimum spanning tree problem for 3-uniform hypergraphs is not yet completely resolved. Lovász [16] showed that the problem is in $\mathcal{P}$ for unweighted 3-uniform hypergraphs. His algorithm, however, is quite complicated and not very efficient. Later, Gabow and Stallmann [11] reduced the complexity from $\mathcal{O}(n^{17})$ to $\mathcal{O}(n^4)$. In [17] Lovász presented a conceptionally simple randomized algorithm for the unweighted case by reducing it to a sequence of computations of determinants of appropriate matrices. Camerini, Galbiati and Maffioli [5] generalized this approach of Lovász to obtain a randomized pseudo-polynomial time algorithm for the weighted case[1].

In this section we develop a randomized parallel algorithm for the minimum spanning tree problem in 3-uniform hypergraphs. It is also based on the algebraic approach of Lovász [17]. The parallelization is based on ideas from Mulmuley, Vazirani, Vazirani [18]. We start with some definitions.

Let $A = (a_{ij})$ be a skew-symmetric matrix (*i.e.*, $A^T = -A$) of size $2n \times 2n$ and let $\mathcal{P}$ be the set of all partitions of $\{1, \ldots, 2n\}$ into two element sets. For an element $p = \{\{i_1, i_2\}, \ldots, \{i_{2n-1}, i_{2n}\}\}$ of $\mathcal{P}$ we denote by $\sigma(p)$ the sign of the permutation

$$\begin{pmatrix} 1 & 2 & \cdots & 2n-1 & 2n \\ i_1 & i_2 & \cdots & i_{2n-1} & i_{2n} \end{pmatrix}$$

and by $\rho(p)$ the product

$$\rho(p) := \prod_{j=1}^{n} a_{i_{2j-1}\, i_{2j}}.$$

One easily verifies that $\sigma(p) \cdot \rho(p)$ is independent of the order of the classes and the order within the classes of p. Therefore

$$\mathrm{pf}(A) = \sum_{p \in \mathcal{P}} \sigma(p) \cdot \rho(p)$$

is well defined. It is called the *pfaffian* of A. A well-known result from linear algebra (cf. e.g. [15]) says:

Lemma 1 *If A is a skew-symmetric matrix A of size $2n \times 2n$ and B an arbitrary $2n \times 2n$ matrix, then*

$$\det(A) = [\mathrm{pf}(A)]^2 \qquad \textit{and} \qquad \mathrm{pf}(BAB^T) = \det(B) \cdot \mathrm{pf}(A). \qquad \square$$

Lemma 2 [5, 17] *Let $m \geq n$ and $a_1, b_1, \ldots, a_m, b_m$ be vectors in $\mathbb{R}^{2n}$, and let $x_1, \ldots, x_m$ be m indeterminants. Then the $2n \times 2n$ matrix*

$$A = \sum_{i=1}^{m} x_i (a_i b_i^T - b_i a_i^T)$$

is skew-symmetric and satisfies

$$\mathrm{pf}(A) = \sum_{1 \leq i_1 < \ldots < i_n \leq m} x_{i_1} \cdot \ldots \cdot x_{i_n} \cdot \det(a_{i_1} | b_{i_1} \cdots | a_{i_n} | b_{i_n}).$$

(With the notation $|$ we just mean the concatenation of the column vectors a_{i_j} and b_{i_j}.)

Let $H = (V, F)$ be a 3-uniform hypergraph on $2n+1$ vertices. For every edge $f = \{i, j, k\}$ in F we pick one vertex arbitrarily, say i, and let $e_f = \{i, j\}$ and $\tilde{e}_f = \{i, k\}$. Let $G = (V, E)$ be a (multi-)graph on the same vertex set as H and with edge set $E = \{e_f, \tilde{e}_f \mid f \in F\}$.

[1] In fact, neither Lovász [17] nor Camerini, Galbiati and Maffioli [5] study the minimum spanning tree problem directly. They both consider the matroid parity problem which contains the minimum spanning tree problem in 3-uniform hypergraphs as a special case.

Fact 3 *A set* $\{f_1, \ldots, f_n\} \subseteq F$ *forms a spanning tree in* H *if and only if* $\{e_{f_1}, \tilde{e}_{f_1}, \ldots, e_{f_n}, \tilde{e}_{f_n}\}$ *forms a spanning tree in* G. □

That is, the problem of finding a minimum spanning tree in a 3-uniform hypergraph is equivalent to the problem of finding a minimum spanning tree in a (multi-)graph, where the edges are "paired",*i.e.*, either both are in the tree or none.

For every pair of edges e_f, $\tilde{e}_f$ we define two $2n$-dimensional vectors a_f and b_f as follows:

$$(a_f)_i = \begin{cases} 1 & \text{if } i \in e_f, \\ 0 & \text{otherwise,} \end{cases} \qquad (b_f)_i = \begin{cases} 1 & \text{if } i \in \tilde{e}_f, \\ 0 & \text{otherwise.} \end{cases}$$

(Note that the a_f's and b_f's are essentially the incidence vectors of e_f and $\tilde{e}_f$ – except that the $(2n+1)$st component has been cut off.) The following fact resembles a well-known property of the incidence matrix of a graph.

Fact 4 *Let* $f_1, \ldots, f_n$ *be* n *edges in* F. *Then*

$$|\det(a_{f_1}|b_{f_1}|\ldots|a_{f_n}|b_{f_n})| = \begin{cases} 1 & \textit{if } f_1, \ldots, f_n \textit{ form a spanning tree in } H, \\ 0 & \textit{otherwise.} \end{cases}$$ □

Finally, let A denote the $2n \times 2n$ matrix

$$A = \sum_{f \in F} 2^{w(f)} (a_f b_f^T - b_f a_f^T). \tag{1}$$

With the above facts and notation at hand it is now relatively straightforward to design an algorithm for constructing a minimum spanning tree whenever this tree is unique.

Lemma 5 *Assume* H *is a 3-uniform hypergraph on* $2n+1$ *vertices and* $w : F \to \mathbb{N}_0$ *is a weight-function such that* H *has a* <u>*unique*</u> *spanning tree* T_0 *of minimum weight, say,* w_0. *Then* $\det(A) \neq 0$ *and* 2^{2w_0} *is the highest power of 2 that divides* $\det(A)$. *Moreover, if we let* $A_f = A - 2^{w(f)}(a_f b_f^T - b_f a_f^T)$ *for all* $f \in F$, *then*

$$f \in T_0 \qquad \textit{if and only if} \qquad \frac{\det(A_f)}{2^{2w_0}} \quad \textit{is even.}$$

Proof. Combining Lemma 1, 2, and Fact 4 we deduce that

$$\det(A) \;=\; [\mathrm{pf}(A)]^2 \;=\; \left[\sum_T 2^{w(T)} \cdot \delta_T\right]^2,$$

where the sum is over all spanning trees T of H and $\delta_T \in \{-1, +1\}$ for all such trees. Hence,

$$\det(A) = \sum_{i \geq 0} c_i 2^{2w_0 + i} \qquad \text{with } c_0 = 1 \text{ and appropriate } c_1, c_2, \ldots \in \mathbb{Z}.$$

The first part of the theorem follows. For the second part just observe that A_f is the matrix corresponding to the hypergraph $H - f$. The reasoning above therefore implies that

$$\det(A_f) = \begin{cases} 2^{2w_0} + c_f \cdot 2^{2w_0+1} & \text{if } f \notin T_0, \\ c_f \cdot 2^{2w_0+2} & \text{if } f \in T_0, \end{cases}$$

for appropriate constants $c_f \in \mathbb{Z}$. □

To achieve the uniqueness of a minimum spanning tree in general hypergraphs we use randomization.

Lemma 6 [18] *Let $H = (V, F)$ be a hypergraph on n vertices. For every vertex $v \in V$ choose uniformly and independently at random an integer $r(v)$ from $[1, 2n]$ and define the weight $w(f)$ of an edge $f \in F$ as $w(f) = \sum_{v \in f} r(v)$. Then*

$$\text{Prob}[\textit{There exists a unique edge of minimum weight}] \geq \frac{1}{2}. \qquad \square$$

Corollary 7 *Let $H = (V, F; w)$ be a weighted 3-uniform hypergraph on $2n+1$ vertices containing at least one spanning tree. For every edge $f \in F$ choose uniformly and independently at random an integer $r(f)$ from $[1, 2\binom{2n+1}{3}]$ and define a weight $w' : E \to \mathbb{N}$ as follows:*

$$w'(f) := 3n^4 \cdot w(f) + r(f).$$

Then

$$\text{Prob}[\textit{There exists a unique minimum spanning tree with respect to } w'] \geq \frac{1}{2}.$$

Proof. Construct a hypergraph H' as follows. The vertex set of H' consists of all edges of H, and the edges of H' correspond to all spanning trees of H. Apply Lemma 6 with respect to the hypergraph H'. Then the weight of an edge in H' is at most $n \cdot 2\binom{2n+1}{3} \leq \frac{8}{3}n^4$ and with probability at least $\frac{1}{2}$ there exists a minimum weight edge.

On the other hand, after scaling the weights $w(f)$ of the edges in H by $3n^4$ the weight of a spanning tree in H is a multiple of $3n^4$. That is, by adding the values $r(f)$ we maintain the order relation on the spanning trees of H of different weight. We just disturb the order of the spanning trees in H which have the same weight "a little" – just enough to reach uniqueness with respect to w'. $\square$

Algorithm 8 (MINIMUM SPANNING TREES IN 3-UNIFORM HYPERGRAPHS)
`Input`: A weighted 3-uniform hypergraph $H = (V, F; w)$ on $2n + 1$ vertices.
`Output`: A spanning tree T of H or FAILURE.

1. Compute the weight function w' as defined in Corollary 7;
2. Compute the matrix A as defined in (1) and let w_0 be the largest integer such that 2^{2w_0} divides $\det(A)$;
   Let $T := \emptyset$;
3. For every edge $f \in F$ do in parallel:
   Compute $\frac{det(A_f)}{2^{2w_0}}$;
   if $\frac{\det(A_f)}{2^{2w_0}}$ is even **then** $T := T \cup \{f\}$;
4. **if** T is a spanning tree of H **then return** T
   else return FAILURE.

Observe that step 3 involves most of the computational effort. Here, one can use e.g. Pan's [19] randomized matrix-inversion algorithm which requires $\mathcal{O}(\log^2 n)$ time and $\mathcal{O}(n^{3.5} \cdot l)$ processors for computing the determinant of an $n \times n$ matrix whose entries are l bit integers. As the entries of the matrix A are of size exponential in $\mathcal{O}(n^4 \max_{f \in F} w(f))$, step 3 needs thus in total $\mathcal{O}(\log^2 n)$ time and $\mathcal{O}(m \cdot n^{7.5} \cdot \max_{f \in F} w(f))$ processors. Combining this observation with Lemma 5 and Corollary 7 we obtain the following result.

Theorem 9 *For all 3-uniform hypergraphs which contain at least one spanning tree Algorithm 8 returns a minimum spanning tree with probability at least $\frac{1}{2}$. The running time of the algorithm is $\mathcal{O}(\log^2 n)$ and it uses at most $\mathcal{O}(m \cdot n^{7.5} \cdot max_{f \in F}\, w(f))$ processors. In particular, if the weight function w is polynomially bounded in n or if the weights are given in unary, this yields an $\mathcal{RNC}$-algorithm for finding a minimum spanning tree in a 3-uniform hypergraph.* □

Corollary 10 *For every $\epsilon > 0$ there exists a randomized parallel algorithm with running time $\mathcal{O}(\log^2 n)$ and $\mathcal{O}(\frac{1}{\epsilon} \cdot m^2 \cdot n^{8.5})$ processors that returns, for all weighted 3-uniform hypergraphs H with at least one spanning tree, a spanning tree T such that $w(T) \leq (1+\epsilon) mst(H)$ with probability at least $\frac{1}{2}$. (Here $mst(H)$ denotes the length of a minimum spanning tree in the hypergraph H.)*

Proof. We apply the usual scaling technique. Let $H = (V, F; w)$ be a weighted 3-uniform hypergraph on n vertices and let $w_{\max} := \max_{f \in F} w(f)$. For a given $\epsilon > 0$ we set $t := \epsilon \cdot w_{\max}/n$ and define a new hypergraph $H' = (V, F; w')$ with the same vertex and edge set and weight function w' given by

$$w'(f) := \left\lceil \frac{w(f)}{t} \right\rceil \qquad \text{for all } f \in F.$$

Observe that, by construction,

$$mst(H') \leq \frac{1}{t} mst(H) + \frac{n-1}{2}$$

and that $w'_{\max} = \max_{f \in F} w'(f)) = \mathcal{O}(\frac{n}{\epsilon})$. Hence, by Theorem 9, Algorithm 8 returns a minimum spanning tree T in H' with probability at least $\frac{1}{2}$, in $\mathcal{O}(\log^2 n)$ time, using $\mathcal{O}(\frac{1}{\epsilon} \cdot m \cdot n^{8.5})$ processors. If T is indeed a minimum spanning tree in H' we can bound its weight in H as follows:

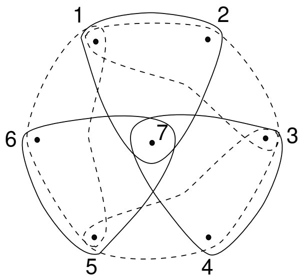
$$w(T) \leq t \cdot w'(T) = t \cdot mst(H') \leq mst(H) + \frac{1}{2} tn \leq mst(H) + \frac{1}{2} \epsilon w_{\max}.$$

That is, if we could guarantee that $mst(H) \geq w_{\max}$ we would be home. Unfortunately this is not true in general. But another trick helps here.

Let $w_1 < \ldots < w_s$ be the occurring weights in H sorted in increasing order. For every $1 \leq i \leq s$ we define a hypergraph $H_i = (V, F_i; w)$ by deleting all edges from H which have weight larger than w_i. That is, we let $F_i := \{f \in F \mid w(f) \leq w_i\}$. Furthermore, let T_{opt} be an (arbitrary) minimum spanning tree in H and let i_0 be defined such that w_{i_0} is the weight of an edge of maximum weight in T_{opt}. Then, clearly, $w_{i_0} \leq mst(H_{i_0}) = mst(H)$. That is, if use the scaling technique outlined above to compute (in parallel) a minimum spanning tree in all hypergraphs H_i and return of the at most s spanning trees those with minimum weight, this tree T will be, with probability at least $\frac{1}{2}$, a spanning tree in H such that $w(T) \leq (1 + \frac{\epsilon}{2}) mst(H)$. The running time of this modified algorithm is, of course, still $\mathcal{O}(\log^2 n)$ while the number of processors increases by a factor of at most m to $\mathcal{O}(\frac{1}{\epsilon} \cdot m^2 \cdot n^{8.5})$. □

Corollary 11 *There exists a fully polynomial randomized (parallel and sequential) approximation scheme for finding a minimum spanning tree in 3-uniform hypergraphs.* □

Remark. It is not true that if Algorithm 8 outputs a spanning tree that this tree is then necessarily a *minimum* spanning tree. The following hypergraph $H = (V, F)$ provides an example:



Here the solid edges $\{1,2,7\}$, $\{3,4,7\}$, and $\{5,6,7\}$ have weight 2, while the dashed edges $\{1,2,3\}$, $\{3,4,5\}$, and $\{5,6,1\}$ have weight 1. H has exactly 3 minimum spanning trees (of weight 4). Hence, the algorithm correctly determines $w_0 = 4$. Now consider H minus an edge f. If f is an edge of weight 2, then $H - f$ contains 2 minimum spanning trees. So $\det(A_f)/2^{2w_0}$ is even for all solid edges. On the other hand, if f is an edge of weight 1 then $H - f$ contains only one minimum spanning tree. So $\det(A_f)/2^{2w_0}$ is odd for all dashed edges. Hence, with respect to the weight function $w' = w$ the algorithm returns the three solid edges – which do form a spanning tree, but not a minimum one.

3 Approximation Algorithms for the Steiner Problem

Let $G = (V, E)$ be a graph and $K \subseteq V$ be a subset of the vertex set. A subgraph T of G is a *Steiner tree* for K, if T is a tree containing all vertices of K (*i.e.*, $K \subseteq V(T)$) such that all leaves of T are elements of K. A *Steiner minimum tree* for K in G is a Steiner tree T such that $|E(T)|$ (in unweighted graphs) resp. $w(T)$ (in weighted graphs) is minimum.

STEINER PROBLEM IN NETWORKS (SPN)
Input: A network (a weighted graph) $N = (V, E; w)$, and a set $K \subseteq V$.
Output: A Steiner minimum tree for K in N, that is a Steiner tree T such that $w(T) = \min\{w(T') \mid T' \text{ a Steiner tree for } K \text{ in } N\}$.

If the input is restricted to *unweighted* graphs $G = (V, E)$ and sets $K \subseteq V$ we speak of the STEINER PROBLEM IN GRAPHS (SPG). Given a graph $G = (V, E)$ or network $N = (V, E; w)$ and a terminal set K we denote by $smt(G, K)$ resp. $smt(N, K)$ the length of a Steiner minimum tree for K in G resp. N. Note that for $K = V$ the Steiner problem is exactly the minimum spanning tree problem. In the following we denote the length of a minimum spanning tree in a hypergraph H by $mst(H)$.

The computational complexity of the Steiner tree problem in graphs and networks varies considerably with the structure of the underlying graph and the cardinality of the terminal set K. While it can be easily solved whenever $|K| = 2$ or $K = V$ and for certain classes of graphs, it is rather difficult in general.

The Steiner tree problem in networks was among the first problems shown to be $\mathcal{NP}$-hard in the seminal paper of Karp [12]. Bern and Plassmann [3] then proved that even the special problem with edge weights restricted to the values 1 and 2 is

$\mathcal{MAXSNP}$-hard. A consequence of the new characterization of the class $\mathcal{NP}$ by Arora, Lund, Motwani, Sudan and Szegedy [1] is thus that there exists no polynomial time approximation scheme for the Steiner tree problem, unless $\mathcal{P}$=$\mathcal{NP}$. Hence, unless $\mathcal{P}$=$\mathcal{NP}$ the best performance ratio attainable by a polynomial time algorithm is a constant larger than 1. It remains a challenging question how close to 1 this performance ratio can be.

Let $N = (V, E; w)$ be a network and $K \subseteq V$ a terminal set. With respect to this Steiner problem we define for all $r \geq 2$ a weighted hypergraph $H_r(N, K) = (K, F_r; w_r)$ on the vertex set K as follows. The edge set F_r consists of all subsets of K of cardinality at most r, and the weight $w_r(f)$ of an edge $f \in F_r$ is the length of a Steiner minimum tree for f in the network N.

Fact 12 *Let $N = (V, E, w)$ be a network with terminal set K and let $H_r(N, K)$ be as defined above. Then*

$$mst(H_r(N, K)) \geq smt(N, K). \tag{2}$$

Proof. Let T be a minimum spanning tree in $H_r(N, K)$. For every edge f in T choose a Steiner minimum tree T_f for f in N. Then the fact that T is a spanning tree in $H_r(N, K)$ implies that $S = \bigcup_{f \in T} T_f$ is a connected subgraph of (V, E) which contains all vertices in K. □

Reconsidering the proof of Fact 12, we observe that given a spanning tree T in $H_r(N, K)$ one can easily determine a Steiner tree for K in the original network N of at most the same length: We just take the union of Steiner minimum trees for all $f \in T$, and delete edges until the obtained graph is a tree in which all leaves are terminals.

Let ρ_r denote the least upper bound of the ratio $mst(H_r(N, K))/smt(N, K)$ for all networks $N = (V, E; w)$ and terminal sets $K \subseteq V$. One easily sees that $\rho_2 = 2$, cf. e.g. [22] or [14]. Obtaining ρ_3 is considerably more difficult. Zelikovsky [23] shows that $\rho_3 \leq 5/3$ and considering appropriate binary trees one deduces easily that in fact $\rho_3 = 5/3$. For general $r \in \mathbb{N}$, Du, Zhang, and Feng [9] proved that $\rho_{2^r} \leq 1 + \frac{1}{r}$ implying in particular that $\rho_r \to 1$ for $r \to \infty$ and, finally, Borchers and Du [4] proved that $\rho_r = ((t+1)2^t + l)/(t2^t + l)$ for $r = 2^t + l$ and $0 \leq l < 2^t$.

Observe that for all constants $r \in \mathbb{N}$ the hypergraph $H_r(N, K)$ can be constructed in polynomial time: There are less than k^r subsets of K of cardinality at most r, where $k = |K|$. For each of these subsets a Steiner minimum tree can be found in $\mathcal{O}(n^2 \log n + nm)$ with the algorithm of Dreyfus and Wagner [8]. A plausible approach for designing an approximation algorithm is therefore to solve the minimum spanning tree problem in $H_r(N, K)$ and deduce from that a Steiner tree for K in N as outlined above. This would give an approximation algorithm with ratio ρ_r. As, however, finding minimum spanning trees in r-uniform hypergraphs is $\mathcal{NP}$-complete for all $r \geq 4$ this reduction to the spanning tree problem is not really helpful[2] – except for the case $k = 3$.

Zelikovsky [23] showed that in the special case of $H_3(N, K)$ one can use a greedy approach to find a spanning tree T in $H_3(N, K)$ of length at most $\frac{1}{2}[mst(H_2(N, K)) +$

[2]In fact, this is not quite obvious at this point, as the reduction from the Steiner tree problem generates only special spanning tree problems. It is, however, easy to see that also these special spanning tree problems are $\mathcal{NP}$-hard to solve, cf. e.g. [20].

$mst(H_3(N,K))]$ and thus a Steiner tree of size at most $\frac{1}{2}(\rho_2+\rho_3) = 11/6$ times the length of a Steiner minimum tree.

Berman and Ramaiyer [2] found a different procedure which makes also use of the hypergraphs $H_r(N,K)$ for $r > 3$. They obtained an algorithm with performance ratio

$$\rho_2 - \sum_{i=3}^{h} \frac{\rho_{i-1}-\rho_i}{i-1} \approx 1.734.$$

for all $h \geq 3$. Zelikovsky [24] invented a so-called relative greedy heuristic for approximating $mst(H_r(N,K))$ that yields an approximation algorithm for the length of a Steiner minimum tree with performance ratio $1+\ln 2 \approx 1.693$. A slight further improvement, then, led to a ratio of $1.644+\varepsilon$ for any positive $\varepsilon > 0$, see Karpinski and Zelikovsky [13].

In order to use the algorithm of the previous section for solving the spanning tree problem in $H_3(N,K)$ we have to reduce the spanning tree problem in hypergraphs with edges containing *at most* three vertices to a corresponding problem in a 3-uniform hypergraph, *i.e.*, a hypergraph where all edges consist of *exactly* three vertices. This, however, can easily be achieved:

Reducing the spanning tree problem to 3-uniform hypergraphs. Let $H = (V, F; w)$ be a weighted hypergraph on n vertices such that every edge contains at most three vertices. Construct a weighted 3-uniform hypergraph $\tilde{H} = (\tilde{V}, \tilde{F}, \tilde{w})$ as follows. The vertex set $\tilde{V}$ consists of all vertices of V plus $n-1$ new vertices $z_1, \ldots, z_{n-1}$, and

$$\begin{aligned} \tilde{F} \;\; &= \;\; F \cup \{e \cup \{z_i\} \mid e \in F, |e| = 2, 1 \leq i \leq n-1\} \\ &\quad \cup \{v \cup \{z_i, z_j\} \mid v \in V, 1 \leq i < j \leq n-1\}. \end{aligned}$$

New triples containing exactly one z-vertex will be called type I triples, while those containing two z-vertices are of type II. The weight function $\tilde{w}$ is defined as follows:

$$\tilde{w}(f) = \begin{cases} w(f) & \text{if } f \in F, \\ w(e) + M & \text{if } f \text{ is a type I triple with } e \in F \text{ and } e \subseteq f, \\ 2M & \text{if } f \text{ is a type II triple.} \end{cases}$$

Fact 13 *The above reduction has the following properties:*

(i) Every spanning tree T in H gives rise to a spanning tree $\tilde{T}$ in $\tilde{H}$ of weight $\tilde{w}(\tilde{T}) = w(T) + (n-1)M$ by replacing every edge of cardinality 2 in T by a type I triple (by adding different z-vertices) and adding type II triples until every z-vertex is contained in exactly one triple.

(ii) If $\tilde{T}$ is a spanning tree in $\tilde{H}$ of weight $\tilde{w}(\tilde{T}) < nM$ then every z-vertex is covered by exactly one triple of $\tilde{T}$. Thus, $\tilde{T}$ corresponds to a spanning tree T of H of weight $w(T) = \tilde{w}(\tilde{T}) - (n-1)M$.

(iii) Let $M \geq n \cdot \max_{f \in F} w(f)$. Then a minimum spanning tree in $\tilde{H}$ of length less than nM corresponds to a minimum spanning tree of H, whereas the fact that the length of a minimum spanning tree in $\tilde{H}$ is at least nM indicates that H contains no spanning tree. Furthermore, if H is the complete hypergraph (e.g. the hypergraph $H_3(N,K)$ in the reduction from the Steiner tree problem) then choosing $M = \max_{f \in F} w(f)$ suffices to achieve these properties. $\square$

Algorithm 14 (STEINER TREE PROBLEM IN GRAPHS)
`Input`: A connected network $N = (V, E; w)$ and a terminal set $K \subseteq V$.
`Output`: A Steiner tree for K or FAILURE.

1. Compute the hypergraph $H_3(N, K)$;
2. Transform the hypergraph $H_3(N, K)$ to the corresponding 3-uniform hypergraph $\tilde{H}$;
3. Use the algorithm of the previous section to find a spanning tree $\tilde{T}$ in $\tilde{H}$;
   If this algorithm returns FAILURE then stop;
4. Transform $\tilde{T}$ to a spanning tree T in $H_3(N, K)$ according to Fact 13;
5. Transform T into a Steiner tree S for K in N; return S.

For an implementation of step 1 observe first that a Steiner minimum tree for a two element set $\{x, y\}$ is just a shortest x-y path. Furthermore, a Steiner minimum tree for a three element set $\{x, y, z\}$ is the union of three shortest paths. Namely, a shortest x-w path plus a shortest y-w path plus a shortest z-w path, where w is an appropriate vertex in V. As the all-pairs shortest path problem can be solved in $\mathcal{O}(\log^2 n)$ time on $\mathcal{O}(n^3)$ processors, we conclude that step 1 can certainly be achieved in $\mathcal{O}(\log^2 n)$ time and $\mathcal{O}(n^4)$ processors. Step 2 is easily implemented within the same time bound. Note that the weight of an edge in the hypergraph $H_3(N, K)$ (and hence also in the hypergraph $\tilde{H}$) is less than $n \cdot \max_{e \in E} w(e)$. Theorem 9 thus implies that $\mathcal{O}(\log^2 n)$ time and $\mathcal{O}(n^{11.5} \cdot \max_{f \in F} w(f))$ processors suffice for step 3. As steps 4 and 5 are again easily implemented within this time bound, we obtain the following theorem.

Theorem 15 *Algorithm 14 is a randomized parallel algorithm which returns with probability at least $\frac{1}{2}$ a Steiner tree S for K such that $w(S) \leq \frac{5}{3} smt(N, K)$. The algorithms runs in $\mathcal{O}(\log^2 n)$ time, using $\mathcal{O}(n^{11.5} \cdot max_{f \in F} w(f))$ processors. In particular, if the weight function w is polynomially bounded in n or if the weights are given in unary, this yields an $\mathcal{RNC}$-approximation algorithm for the* STEINER PROBLEM IN NETWORKS *with performance ratio* 5/3. □

Corollary 16 *There exists an $\mathcal{RNC}$-approximation algorithm with performance ratio* 5/3 *for the* STEINER PROBLEM IN GRAPHS. □

For networks we can also use the algorithm of Corollary 10 instead of those of Theorem 9 to obtain an $\mathcal{RNC}$-approximation scheme for performance ratio 5/3:

Corollary 17 *For every $\epsilon > 0$ there exists a randomized parallel algorithm that given a network $N = (V, E; \ell)$ on n vertices and a terminal set K returns in $\mathcal{O}(\log^2 n)$ steps, using $poly(n + \frac{1}{\epsilon})$ processors, with probability at least $\frac{1}{2}$ a Steiner tree T such that*

$$\ell(T) \leq \frac{5}{3}(1 + \epsilon) smt(N, K).$$

□

Of course, the algorithm can also be implemented as a sequential algorithm. Here one can also use the fact that the determinant of an $n \times n$ matrix containing l bit integers can be computed on a random access machine in $\mathcal{O}(n^{\alpha} l \log l)$ bit operations, where $\mathcal{O}(n^{\alpha})$ is the number of arithmetic operations required to multiply two $n \times n$ matrices. Currently, the best know value is $\alpha < 2.376$ [7].

Corollary 18 *There exists a randomized sequential $\mathcal{O}(n^{8+\alpha}\log n)$ approximation algorithm with performance ratio 5/3 for the* Steiner Problem in Graphs. □

Corollary 19 *For every $\epsilon > 0$ there exists a randomized sequential $\mathcal{O}(\frac{\log\frac{1}{\epsilon}}{\epsilon}\cdot n^{11+\alpha}\log n)$ approximation algorithm with performance ratio $5/3+\epsilon$ for the* Steiner Problem in Networks. □

In comparison, the best deterministic sequential approximation algorithm for the Steiner Problem in Networks of Karpinski and Zelikovsky [13] has a performance ratio of ≈ 1.644 which is slightly better than $5/3 \approx 1.667$. However, this performance guarantee is only achieved in the limit. More precisely, building on the work of Zelikovsky [24] , Karpinski and Zelikovsky [13] define a class $(\mathcal{A}_k)$ of approximation algorithms such that the running time of $\mathcal{A}_k$ is bounded by $\mathcal{O}(n^k)$ and the performance ratio of $\mathcal{A}_k$ tends to 1.644 for k tending to infinity. For reasonable "small" k, say k up to 20, the performance ratio is, however, still larger than 5/3.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and hardness of approximation problems, *Proc. 33rd Annual IEEE Symp. Foundations of Computer Science* (1992), 14–23.

[2] P. Berman and V. Ramaiyer, Improved Approximations for the Steiner Tree Problem, *Journal of Algorithms* **17** (1994), 381–408.

[3] M. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Information Processing Letters* **32** (1989), 171–176.

[4] A. Borchers and D.-Z. Du, The k-Steiner ratio in graphs, *Proc. 27th Annual ACM Symp. on the Theory of Computing* (1995), 641–649.

[5] P.M. Camerini, G. Galbiati, and F. Maffioli, Random pseudo-polynomial algorithms for exact matroid problems, *Journal of Algorithms* **13** (1992), 258–273.

[6] Choukhmane El-Arbi, Une heuristique pour le problème de l'arbre de Steiner, *R.A.I.R.O. Recherche opérationnelle* **12** (1978), 207–212.

[7] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *Proc. 19th Annual ACM Symp. on Theory of Computing* (1987), 1–6.

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

[9] D.-Z. Du, Y.-J. Zhang and Q. Feng, On better heuristic for Euclidean Steiner minimum trees, *Proc. 32nd Annual IEEE Symp. on Foundations of Computer Science* (1991), 431–439.

[10] H.N. Gabow, Z. Galil, T.H. Spencer, Efficient implementation of graph algorithms using contraction, *Proc. 25th Annual IEEE Symp. on Foundations of Computer Science* (1984), 347–357.

[11] H.N. Gabow and M. Stallmann, An augmenting path algorithm for linear matroid parity, *Combinatorica* **6** (1986), 123–150.

[12] R. Karp, Reducibility among combinatorial problems, in: *Complexity of computer computations* (Miller, R.E., Thatcher, J.W., eds.), Plenum Press, 1972, 85-103.

[13] M. Karpinski and A.Z. Zelikovsky, New approximation algorithms for the Steiner tree problem, *Journal of Combinatorial Optimization* **1** (1997), 47–65.

[14] L. Kou, G. Markowsky, and L. Berman, A fast algorithm for Steiner trees, *Acta Informatica* **15** (1981), 141–145.

[15] S. Lang, Algebra, Addison-Wesley Publishing Company, 1993.

[16] L. Lovász, The matroid matching problem, *Algebraic Methods in Graph Theory*, Colloquia Mathematica Societatis János Bolyai, Szeged (Hungary), 1978.

[17] L. Lovász, On determinants, matchings and random algorithms, *Fund. Comput. Theory* **79** (1979), 565–574.

[18] K. Mulmuley, U. Vazirani, and V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica* **7** (1987), 105–113.

[19] V. Pan, Fast and efficient algorithms for the exact inversion of integer matrices, *Fifth Annual Foundations of Software Technology and Theoretical Computer Science Conference* (1985), LNCS 206, 504–521.

[20] H.J. Prömel and A. Steger, *The Steiner Tree Problem. A Tour through Graphs, Algorithms, and Complexity*, Vieweg Verlag, Wiesbaden, to appear 1997.

[21] F. Suraweera and P. Bhattacharya, An $\mathcal{O}(\log m)$ parallel algorithm for the minimum spanning tree problem, *Inf. Proc. Lett.* **45** (1993), 159–163.

[22] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Japonica* **24** (1980), 573–577.

[23] A.Z. Zelikovsky, An 11/6-approximation algorithm for the network Steiner problem, *Algorithmica* **9** (1993), 463–470.

[24] A.Z. Zelikovsky, Better approximation algorithms for the network and Euclidean Steiner tree problems, Technical Report, Kishinev, 1995.