Journal of Algorithms **36,** 89–101 (2000)
doi:10.1006/jagm.2000.1086, available online at http://www.idealibrary.com on IDEAL®

A New Approximation Algorithm for the Steiner Tree Problem with Performance Ratio 5/3

Hans Jürgen Prömel

Institut für Informatik, Humboldt Universität zu Berlin, 10099 Berlin, Germany
E-mail: proemel@informatik.hu-berlin.de

and

Angelika Steger

Institut für Informatik, Technische Universität München, 80290 München, Germany
E-mail: steger@in.tum.de

Received June 19, 1998

In this paper we present an RNC approximation algorithm for the Steiner tree problem in graphs with performance ratio 5/3 and RNC approximation algorithms for the Steiner tree problem in networks with performance ratio $5/3 + \epsilon$ for all $\epsilon > 0$. This is achieved by considering a related problem, the minimum spanning tree problem in weighted 3-uniform hypergraphs. For that problem we give a fully polynomial randomized approximation scheme. Our approach also gives rise to conceptually much easier and faster (though randomized) sequential approximation algorithms for the Steiner tree problem than the currently best known algorithms from Karpinski and Zelikovsky which almost match their approximation factor.

1. INTRODUCTION

In recent years, the Steiner tree problem in graphs has attracted considerable attention, both from the theoretical point of view and 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 *et al*. [13]. After a long period without any progress Zelikovsky [22], Berman and Ramaiyer [2],

0196-6774/00 $35.00

Zelikovsky [23], and Karpinski and Zelikovsky [12] improved the approximation factor step by step from 2 to 1.644.

In this paper we present an RNC approximation algorithm for the Steiner problem with approximation ratio $5/3 + \epsilon$ for all $\epsilon > 0$. The number of processors needed by these algorithms is polynomially bounded in ϵ^{-1} and n. For the Steiner problem in graphs these algorithms specialize to an RNC approximation algorithm with approximation ratio $5/3$. Our approach also gives rise to sequential approximation algorithms conceptually much easier and faster (though randomized) than the Karpinski–Zelikovsky approach which almost match the approximation factor of these algorithms.

The core of our algorithm is a fully polynomial randomized approximation scheme for finding a minimum spanning tree in weighted 3-uniform hypergraphs.

2. MINIMUM SPANNING TREES IN 3-UNIFORM HYPERGRAPHS

A *hypergraph* $H = (V, F)$ is a generalization of a graph 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 only 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 if and only if 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 contrast to graphs, not every connected hypergraph contains a spanning tree!)

Minimum Spanning Tree Problem (*MST*).

Input: A weighted hypergraph $H = (V, F; w)$, where $w : F \to \mathbf{N}$,
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 4-uniform hypergraphs even deciding whether there *exists* a spanning tree is $\mathcal{NP}$-complete. The status of the case $k = 3$, however, is not yet completely resolved. For the unweighted case Lovász [15] provided

a very complicated $\mathcal{O}(n^{17})$ algorithm, which was later improved to an $\mathcal{O}(n^4)$ algorithm by Gabow and Stallmann [10]. Here, n denotes the number of vertices of the hypergraph. Also, Lovász [16] 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 *et al.* [5] generalized the approach of Lovász [16] to obtain a randomized pseudo-polynomial time algorithm. Narayanan *et al.* [18] combined this approach with ideas from Mulmuley *et al.* [17] to obtain an RNC^2 algorithm for the weighted problem, where the number of processors is polynomial in n and $w_{\max}$ (the maximum weight of an edge). In fact, neither [16] nor [5] or [18] studies the minimum spanning tree problem directly. They all consider the (more general) matroid parity problem over linearly representable matroids, which contains the minimum spanning tree problem in 3-uniform hypergraphs as a special case, as we will point out shortly. In this section we quickly recall the main ideas from [5, 16, 18] and generalize them to obtain an approximation scheme for finding a minimum spanning tree in weighted 3-uniform hypergraphs, or, more generally, for the weighted matroid parity problem over linearly representable matroids.

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 $E = \{e_f, \tilde{e}_f \mid f \in F\}$.

FACT 2.1. *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 or none are in the tree.

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 2.2. *Let $f_1, \dots, f_n$ be n edges in F. Then*

$$\left|\det\left(a_{f_1}|b_{f_1}|\cdots|a_{f_n}|b_{f_n}\right)\right| = \begin{cases} 1 & \text{if } f_1, \dots, f_n \text{ form a spanning tree in } H, \\ 0 & \text{otherwise.} \end{cases}$$

The (weighted) matroid parity problem can be stated as follows: given m pairs of vectors over $\mathbf{Q}^{2n}$, say a_i, b_i for $i = 1, \dots, m$, and a weight w_i for each pair, find a parity base of minimum weight; i.e., find n pairs such that the corresponding vectors are linearly independent and the sum of their weights is minimized with respect to this property. (Actually, it is not clear that such a parity basis exists. For the sake of this paper we will, however, tacitly assume this. This is no real restriction, as by adding additional pairs with sufficiently high weight the existence of a parity base can always be achieved.) Obviously, the derivation above shows that the minimum spanning tree problem in weighted 3-uniform hypergraphs is a special case of the matroid parity problem. In [5, 16] the solution of the matroid parity problem is essentially reduced to the computation of the determinant of appropriately defined matrices. Namely, they show (the notation | just means concatenation of column vectors):

LEMMA 2.1 [5, 16]. *Let $m \geq n$ and $a_1, b_1, \dots, a_m, b_m$ be vectors in $\mathbf{Q}^{2n}$ and let $x_1, \dots, x_m$ be m indeterminants. Then the $2n \times 2n$ matrix*

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

is skew-symmetric and satisfies

$$\det(A) = \left[\sum_{1 \leq i_1 < \cdots < i_n \leq m} x_{i_1} \cdot \ldots \cdot x_{i_n} \cdot \det\left(a_{i_1}|b_{i_1}|\cdots|a_{i_n}|b_{i_n}\right)\right]^2.$$

After replacement of the indeterminants x_i by, e.g., 2^{w_i} this implies immediately an algorithm for the matroid parity problem whenever the minimum parity base is *unique*.

COROLLARY 2.1. *Assume that the weights w_i of the matroid parity problem are such that there exists a unique parity base B_0 of minimum weight, say w_0. Then the matrix*

$$A = \sum_{i=1}^{m} 2^{w_i}\left(a_i b_i^T - b_i a_i^T\right)$$

satisfies that $\det(A) \neq 0$ and 2^{2w_0} is the highest power of 2 that divides $\det(A)$. Moreover, if we let $A_i = A - 2^{w_i}(a_i b_i^T - b_i a_i^T)$ for all $i = 1, \dots, m$,

then

$$a_i, b_i \in B_0 \text{ if and only if } \frac{\det(A_i)}{2^{2w_0}} \text{ is even.}$$

To achieve uniqueness of the solution for the general matroid parity problem [18] we used the Isolating Lemma from [17]. This lemma states that a family of subsets of an m element set $\mathcal{S}$ contains, with probability at least $1/2$, a *unique* set of minimum weight, provided that the weight of a set is the sum of the weights of the elements contained in it and if, additionally, the weights of the elements are chosen randomly and independently from the set $\{1, \ldots, 2m\}$. In the application to the parity problem the set $\mathcal{S}$ consists of all pairs (a_i, b_i) and the family of subsets corresponds to all parity bases. Replacing the original weights w_i with $w_i' := (2mn + 1)w_i + r_i$, where r_i is a randomly and independently chosen value from $\{1, \ldots, 2m\}$, one deduces easily that a minimum parity base with respect to the weights w_i' is also a minimum parity base with respect to the weights w_i and that, with probability at least $1/2$, the minimum parity base with respect to the weights w_i' is unique. Combining this with the well-known fact that computing the determinant of a matrix is in NC^2 (cf., e.g., [19]) one obtains the following result from [18]. (We omit the rather messy precise statement of the processor bound, but emphasize that the entries in the matrices A and A_i depend *exponentially* on $w_{\max} := \max_i w_i$ and that the number of processors therefore has to depend on $w_{\max}$.)

THEOREM 2.1 [18]. *There exists a randomized parallel algorithm for the matroid parity problem that computes a minimum parity base in* $\mathcal{O}((\log n)^2)$ *time using poly*$(n + w_{\max})$ *many processors with probability at least* $1/2$.

Note that if $w_{\max}$ is polynomially bounded in n or if the weights are given in unary the algorithm from Theorem 2.1 is fact an RNC^2 algorithm.

Recalling that the minimum spanning tree problem in 3-uniform hypergraphs is a special case of the matroid parity problem we obtain the following algorithm:

ALGORITHM 2.2 (Min Spanning Trees in 3-Unif, 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 a weight function w' by $w'(f) := (2mn + 1)w(f) + r_f$, where r_f is a randomly chosen weight from $\{1, \ldots, 2|F|\}$;

2. Compute the matrix A as defined in Corollary 2.1 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 $\dfrac{\det(A_f)}{2^{2w_0}}$;
   if $\dfrac{\det(A_f)}{2^{2w_0}}$ is even, **then** $T := T \cup \{f\}$;
4. **if** T is a spanning tree of H, **then** **return** T
   else **return** FAILURE.

COROLLARY 2.2. *Algorithm* 2.2 *computes a minimum spanning tree for hypergraphs containing at least one spanning tree with probability at least* $1/2$ *in* $\mathcal{O}((\log n)^2)$ *time using poly*$(n + w_{\max})$ *many processors.*

We now combine the algorithm from Theorem 2.1 with a scaling technique to obtain RNC^2 algorithms which approximate the minimum weight parity base arbitrarily well.

THEOREM 2.3. *There exists a randomized parallel approximation scheme for the weighted matroid parity problem that computes for every* $\epsilon > 0$ *in time* $\mathcal{O}((\log n)^2)$ *using poly*$(n + \epsilon^{-1})$ *many processors, with probability at least* $1/2$, *a parity base* B *that satisfies* $w(B) \le (1 + \epsilon)w(B_{\text{opt}})$. *Here* $w(\cdot)$ *denotes the weight of a parity base and* B_{opt} *denotes a parity base of minimum weight.*

Proof. Consider an input to the matroid parity problem $a_1, b_1, \ldots, a_m, b_m$ in $\mathbf{Q}^{2n}$ with arbitrary weights $w_1, \ldots, w_m$. Let $w_{\max} := \max_{1 \le i \le m} w_i$. For a given $\epsilon > 0$ we set $t := \epsilon \cdot w_{\max}/n$ and define a new matroid parity problem with the same pairs of vectors but weights w_i' given by

$$w_i' := \left\lceil \frac{w_i}{t} \right\rceil \text{ for all } 1 \le i \le m.$$

Let B'_{opt} denote a parity base of minimum weight with respect to the weights w_i'. We also use the notation $w'(\cdot)$ to denote the weight of a parity base with respect to the weights w_i'.

Observe that, by construction,

$$w'(B'_{\text{opt}}) \le w'(B_{\text{opt}}) \le \frac{1}{t} w(B_{\text{opt}}) + n$$

and that $w'_{\max} = \max_{1 \le i \le m} w_i' = \mathcal{O}(\frac{n}{\epsilon})$. Hence, we can apply the algorithm from Theorem 2.1 to the scaled problem to find, with probability at least $\frac{1}{2}$, a parity base B such that $w'(B) = w'(B'_{\text{opt}})$. Note also, that the number of necessary processors is now polynomially bounded in n and ϵ^{-1}. Assume

B is indeed a minimum parity base with respect to the scaled weights w'_i. Then we can bound its weight with respect to the original weights w_i as follows:

$$w(B) \le t \cdot w'(B) = t \cdot w'(B'_{\text{opt}}) \le w(B_{\text{opt}}) + tn \le w(B_{\text{opt}}) + ew_{\max}.$$

If we could guarantee that $w(B_{\text{opt}}) \ge w_{\max}$ we would be home. Unfortunately this is not true in general. But another trick helps here.

Let $\hat{w}_1 < \cdots < \hat{w}_s$ denote the (different) weights of the original problem sorted in increasing order. For every $1 \le i \le s$ we define a restricted parity problem P_i by deleting all pairs which have weights larger than $\hat{w}_i$. Note that some of these problems P_i will not contain a parity base. But because we will be concerned only with those problems which do, we do not care. Let i_0 denote the index for which $\hat{w}_{i_0}$ is the maximum weight in B_{opt}. Then, clearly, P_{i_0} does contain a parity base and the weight of a minimum parity base of problem P_{i_0} is equal to $w(B_{\text{opt}})$ and is hence greater than or equal to the maximum weight occurring in the problem P_{i_0}. That is, if we use the scaling technique outlined above to compute (in parallel) a minimum parity base for all problems P_i and of the at most s parity bases obtained, we return the one with minimum weight, this base B will be, with probability at least $\frac{1}{2}$, a parity base of the original problem such that $w(B) \le (1 + \epsilon)w(B_{\text{opt}})$. The running time of this modified algorithm is, of course, still $\mathcal{O}((\log n)^2)$. The number of processors increases by a factor of at most m. ∎

COROLLARY 2.3. *For all $\epsilon > 0$ there exists an RNC^2 algorithm that finds in 3-uniform hypergraphs $H = (V, F; w)$ which contain at least one spanning tree with probability at least $1/2$ a minimum spanning tree.*

Remark. It is not true that if Algorithm 2.2 outputs a spanning tree that this tree is then necessarily a *minimum* spanning tree. Consider, e.g., the hypergraph $H = (V, F)$ with vertex set $V = \{1, \ldots, 7\}$ and six edges $\{1, 2, 7\}$, $\{3, 4, 7\}$, $\{5, 6, 7\}$, $\{1, 2, 3\}$, $\{3, 4, 5\}$, and $\{5, 6, 1\}$, the first three having weight 2 and the remaining three having weight 1. One easily checks that H has exactly three minimum spanning trees (of weight 4) and that the algorithm therefore correctly determines $w_0 = 4$. However, by considering $H - f$ for different edges f one also finds that the algorithm will return the three edges $\{1, 2, 7\}$, $\{3, 4, 7\}$, $\{5, 6, 7\}$. These do form a spanning tree, but not a minimum one.

3. THE STEINER TREE PROBLEM

Let $G = (V, E)$ be a graph and let $K \subseteq V$ 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 $K \subseteq V$ this restricted problem is called 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. The Steiner problem in networks was among the first problems shown to be $\mathcal{NP}$-hard in the seminal paper of Karp [11]. Bern and Plassmann [3] proved that even the special problem with edge weights restricted to the values 1 and 2 is MaxSNP-hard. A consequence of the characterization of the class $\mathcal{NP}$ by Arora *et al.* [1] in terms of probabilistically checkable proofs is thus that there exists no polynomial time approximation scheme for the Steiner 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.

Basically all known approximation algorithms for the Steiner problem are in one way or another related to solving a spanning tree problem in a suitably defined hypergraph. Let $N = (V, E; w)$ be a network and let $K \subseteq V$ be 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 3.1. $mst(H_r(N, K)) \geq smt(N, K)$.

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 3.1, 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 for $mst(H_r(N, K))/smt(N, K)$ for all networks $N = (V, E; w)$ and terminal sets $K \subseteq V$. One easily sees that

$\rho_2 = 2$, cf., e.g., [21] or [13]. Obtaining ρ_3 is considerably more difficult. Zelikovsky [22] shows that $\rho_3 \leq 5/3$ and considering appropriate binary trees one deduces easily that in fact $\rho_3 = 5/3$. For general $r \in \mathbf{N}$, Du *et al.* [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 $r \in \mathbf{N}$ 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 to designing an approximation algorithm is therefore to solve the minimum spanning tree problem in $H_r(N, K)$ and to 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, already pointed out in the previous section finding minimum spanning trees in r-uniform hypergraphs is $\mathcal{NP}$-complete for all $r \geq 4$, so this reduction to the spanning tree problem is not really helpful[1]—except for the case $k = 3$. Here we can apply our algorithm from the previous section. Before we do that, however, we complete our short historical review.

Zelikovsky [22] 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 $\frac{1}{2}[mst(H_2(N, K)) + mst(H_3(N, K))]$ 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 also makes use of the hypergraphs $H_r(N, K)$ for $r > 3$ to obtain approximation algorithms with performance ratio $\rho_2 - \sum_{i=3}^{h}(\rho_{i-1} - \rho_i)/(i-1) \approx 1.734$ for all $h \geq 3$. Zelikovsky [23] invented a so-called relative greedy heuristic for approximating $mst(H_r(N, K))$ that yields approximation algorithms for the length of a Steiner minimum tree with performance ratio arbitrarily close to $1 + \ln 2 \approx 1.693$. A slight further improvement, then, led to a ratio of $1.644 + \varepsilon$ for any positive $\varepsilon > 0$; see Karpinski and Zelikovsky [12].

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

[1] 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].

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

$$\tilde{F} = \{f \in F \mid |f| = 3\} \cup \{e \cup \{z_i\} \mid e \in F, |e| = 2, 1 \le i \le n - 1\}$$
$$\times \bigcup \{v \cup \{z_i, z_j\} \mid v \in V, 1 \le i < j \le n - 1\}.$$

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

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

where M is a suitably chosen huge weight (see below).

FACT 3.2. *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 edge (by adding different z-vertices) and adding type II edges until every z-vertex is contained in exactly one edge*.

(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 edge 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 $M \ge n \cdot \max_{f \in F} w(f)$. 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 $H_3(N, K)$ arising in the reduction from some Steiner tree problem), then choosing $M = \max_{f \in F} w(f)$ suffices to achieve these properties*.

ALGORITHM 3.1 (Steiner 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 Algorithm 2.2 to find a spanning tree $\tilde{T}$ in $\tilde{H}$;
   If this algorithm returns FAILURE, then stop with FAILURE;
4. Transform $\tilde{T}$ to a spanning tree T in $H_3(N, K)$ according to Fact 3.4;
5. Transform T into a Steiner tree S for K in N; return S.

For an implementation 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. Recall also that the all-pairs shortest path problem can be solved in $\mathcal{O}((\log n)^2)$ time on $\mathcal{O}(n^3)$ processors. As steps 2, 4, and 5 are also easily implemented within the same time and processor bounds, this together with Theorem 2.2 proves the following theorem.

THEOREM 3.2. *Algorithm* 3.1 *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* $\mathcal{O}((\log n)^2)$ *time, using poly*$(n, w_{\max})$ *many processors. In particular, if the weight function w is polynomially bounded in n or if the weights are given in unary, this yields an RNC*^{2} *approximation algorithm for the Steiner problem in networks with performance ratio* $5/3$.

COROLLARY 3.1. *There exists an RNC*^{2} *approximation algorithm with performance ratio* $5/3$ *for the Steiner problem in graphs.*

Using the approximation scheme from Corollary 2.3 instead of Algorithm 2.2 also avoids the dependence on the maximum weight in Theorem 3.2, at the price of a slightly weaker performance ratio.

COROLLARY 3.2. *For every* $\epsilon > 0$ *there exists an RNC*^{2} *algorithm that given a network* $N = (V, E; w)$ *on n vertices and a terminal set K returns, with probability at least* $\frac{1}{2}$, *a Steiner tree S such that*

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

Of course, these algorithms can also be implemented as sequential algorithms. For the sake of completeness we state here an explicit time bound. To do that we 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 known value is $\alpha \approx 2.376$ [7].

COROLLARY 3.3. *For the Steiner problem in graphs there exists a randomized sequential* $\mathcal{O}(n^{8+\alpha} \log n)$ *approximation algorithm with performance ratio* $5/3$.

Proof. For the Steiner problem in graphs the maximum weight in $H_3(N, K)$ and according to part (iii) of Fact 3.4 also in $\tilde{H}$ is bounded by $\mathcal{O}(n)$. The number of edges in $\tilde{H}$ is $m = \mathcal{O}(n^3)$. The maximum of the weight function constructed in step 1 of Algorithm 2.2 is thus of the order $\mathcal{O}(n^5)$. That is, we may assume that $l = \mathcal{O}(n^5)$, neglecting the usual logarithmic factors. The running time of Algorithm 2.2 and thus also that of Algorithm 3.1 are dominated by the computations of $m + 1 = \mathcal{O}(n^3)$ determinants, each of which requires $\mathcal{O}(n^{5+\alpha} \cdot \log n)$ time. ∎

COROLLARY 3.4. *For the Steiner problem in networks there exists for every* $\epsilon > 0$ *a randomized sequential* $\mathcal{O}((\log(1/\epsilon)/\epsilon) \cdot n^{11+\alpha} \log n)$ *approximation algorithm with performance ratio* $5/3 + \epsilon$.

Proof. Consider the proof of Theorem 2.3. The maximum weight in the scaled problem is bounded by $\mathcal{O}(\frac{n}{\epsilon})$. The claimed bound follows in the same way as in the proof of Corollary 3.3, keeping in mind that we have to solve $m = \mathcal{O}(n^3)$ instead of just one spanning tree problem. ∎

In comparison, the best deterministic sequential approximation algorithm for the Steiner problem in networks of Karpinski and Zelikovsky [12] 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 [23], Karpinski and Zelikovsky [12] 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 reasonably "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, *in* "Proc. 33rd Annual IEEE Symp. Foundations of Computer Science," pp. 14–23, 1992.
2. P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem, *J. Algorithms* **17** (1994), 381–408.
3. M. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Inform. Process. Lett.* **32** (1989), 171–176.
4. A. Borchers and D.-Z. Du, The k-Steiner ratio in graphs, *in* "Proc. 27th Annual ACM Symp. on the Theory of Computing," pp. 641–649, 1995.

5. P. M. Camerini, G. Galbiati, and F. Maffioli, Random pseudo-polynomial algorithms for exact matroid problems, *J. Algorithms* **13** (1992), 258–273.
6. Choukhmane El-Arbi, Une heuristique pour le problème de l'arbre de Steiner, *RAIRO. Rech. Opér.* **12** (1978), 207–212.
7. D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *in* "Proc. 19th Annual ACM Symp. on Theory of Computing," pp. 1–6, 1987.
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, *in* "Proc. 32nd Annual IEEE Symp. on Foundations of Computer Science," pp. 431–439, 1991.
10. H. N. Gabow and M. Stallmann, An augmenting path algorithm for linear matroid parity, *Combinatorics* **6** (1986), 123–150.
11. R. Karp, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–103, Plenum, New York, 1972.
12. M. Karpinski and A. Z. Zelikovsky, New approximation algorithms for the Steiner tree problem, *J. Comb. Optim.* **1** (1997), 47–65.
13. L. Kou, G. Markowsky, and L. Berman, A fast algorithm for Steiner trees, *Acta Inform.* **15** (1981), 141–145.
14. S. Lang, "Algebra," Addison–Wesley, Reading, MA, 1993.
15. L. Lovász, The matroid matching problem, *in* "Algebraic Methods in Graph Theory," Colloquia Mathematica Societatis János Bolyai, Szeged, 1978.
16. L. Lovász, On determinants, matchings and random algorithms, *Fund. Comput. Theory* **79** (1979), 565–574.
17. K. Mulmuley, U. Vazirani, and V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica* **7** (1987), 105–113.
18. H. Narayanan, H. Saran, and V. V. Vazirani, Randomized parallel algorithms for matroid union and intersection, with applications to arboreseces and edge-disjoint spanning trees, *SIAM J. Comput.* **23** (1994), 387–397.
19. V. Pan, Fast and efficient algorithms for the exact inversion of integer matrices, *in* "Fifth Annual Foundations of Software Technology and Theoretical Computer Science Conference," Lecture Notes in Computer Science, Vol. 206, pp. 504–521, Springer-Verlag, Berlin/New York, 1985.
20. H. J. Prömel and A. Steger, "The Steiner Tree Problem: A Tour through Graphs, Algorithms, and Complexity," Vieweg Verlag, Wiesbaden, 2000.
21. H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Japon.* **24** (1980), 573–577.
22. A. Z. Zelikovsky, An 11/6-approximation algorithm for the network Steiner problem, *Algorithmica* **9** (1993), 463–470.
23. A. Z. Zelikovsky, Better Approximation Algorithms for the Network and Euclidean Steiner Tree Problems," Technical report, Kishinev, 1995.