Maximum Series-Parallel Subgraph

Gruia Călinescu[1,⋆], Cristina G. Fernandes[2,⋆⋆], and Hemanshu Kaul[3]

[1] Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA
calinescu@iit.edu

[2] Department of Computer Science, University of São Paulo, Rua do Matão, 1010, 05508-090 São Paulo, Brazil
cris@ime.usp.br

[3] Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL 60616, USA
kaul@iit.edu

Abstract. Consider the NP-hard problem of, given a simple graph G, to find a series-parallel subgraph of G with the maximum number of edges. The algorithm that, given a connected graph G, outputs a spanning tree of G, is a $\frac{1}{2}$-approximation. Indeed, if n is the number of vertices in G, any spanning tree in G has $n-1$ edges and any series-parallel graph on n vertices has at most $2n-3$ edges. We present a $\frac{7}{12}$-approximation for this problem and results showing the limits of our approach.

1 Introduction

The Maximum Series-Parallel Subgraph (MSP) problem is: given a simple graph G, find a series-parallel subgraph of G with the maximum number of edges. This problem is known to be NP-hard [3].

The algorithm that, given a connected graph G, outputs a spanning tree of G, is a $1/2$-approximation. Indeed, if n is the number of vertices in G, any spanning tree in G has $n-1$ edges and any series-parallel graph on n vertices has at most $2n-3$ edges. We present a $7/12$-approximation for this problem.

We apply a method, previously used for the Maximum Planar Subgraph problem [4], of producing a subgraph whose blocks (maximal 2-connected components) have a very simple structure. The way to produce such a subgraph also has similarities to some approximation algorithms for the Minimum Steiner Tree problem [1,6].

A novelty of this work is that we allow blocks to have unbounded size. Indeed, using only blocks of bounded size does not lead to an improvement (as we show later). This is a main difference to the works on Maximum Planar Subgraph and Minimum Steiner Tree [1,4,6]. A second difference, when compared to the

⋆ Research supported in part by NSF grant CCF-0515088, and performed in part while on sabbatical at University of Wisconsin Milwaukee.

⋆⋆ Research supported in part by CNPq 312347/2006-5, 485671/2007-7, and 486124/2007-0.

C. Paul and M. Habib (Eds.): WG 2009, LNCS 5911, pp. 54–65, 2010.

Maximum Planar Subgraph algorithms, is that, to assure a good performance, our algorithm has to sometimes throw away or shrink previously selected blocks. We show ahead a family of examples that indicates that such an approach is necessary.

We call *spruces* the very simple series-parallel graphs that we admit as non-bridge blocks in the subgraph we produce. (We define spruces in the next subsection; a *bridge* consists of two adjacent vertices.) We prove that a subgraph whose non-bridge blocks are spruces, and with maximum number of edges among such subgraphs, achieves a ratio of 2/3, and this ratio is tight. Unfortunately, computing such a subgraph is NP-hard, as we also show. So our algorithm in fact computes only a large such subgraph. The ratio our algorithm achieves is 7/12, which happens to be the average between 1/2 and 2/3. This is a coincidence though, because our analysis compares directly the algorithm's output to an optimal solution.

In a related work, Cai [2] considered the variant of the problem where one is given a complete weighted graph, and wants to find a maximal series-parallel graph of minimum weight. He presented a 1.655-approximation for this variant when the input graph is a set of points in the plane with their distances as weights.

1.1 Preliminaries

Two edges of a multigraph are *parallel* if they have the same endpoints, and they are *series* edges if there is some vertex of degree two incident to both of them. A multigraph is *series-parallel* if it arises from a forest by repeated replacing edges by parallel or series edges [7].

All of our graphs are undirected and simple, unless otherwise specified. From the definition above, one can see that a maximal series-parallel graph can be constructed by the following procedure. Start with two adjacent vertices s and t, and then repeat the following: add one new vertex and make it adjacent to two existing adjacent vertices. (Such graphs are also called *2-trees* in the literature, and series-parallel graphs are also known as *partial 2-trees*.)

Based on the construction above, a *normalized* tree decomposition of a maximal series-parallel graph is built as follows (see Fig. 1 for an example). Start with one node with bag $\{s, t\}$, the root of our tree decomposition. We maintain the invariant that, for any edge of the series-parallel graph, there is exactly one node in the tree decomposition whose bag consists of the endpoints of the edge. Whenever a vertex z is added to the series-parallel graph, and made adjacent to existing adjacent vertices x and y, add to the tree decomposition three nodes: one with bag $\{x, y, z\}$, child of the node with bag $\{x, y\}$, and two "twin" children of this new node, with bags $\{x, z\}$ and $\{y, z\}$. In this tree decomposition, all even-level nodes have bags of size two, all odd-level nodes have bags of size three, and no leaf is in an odd level. For a normalized tree decomposition T of a maximal series-parallel graph H with $|V(H)| = n$, there are exactly $n-2$ odd-level nodes in T.

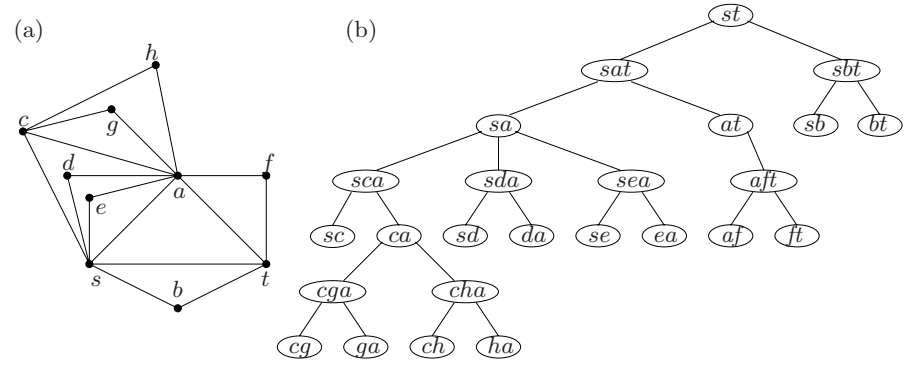

Fig. 1. (a) A maximal series-parallel graph, obtained by starting with the two adjacent vertices s and t, and then adding in order vertices a, b, c, d, e, f, g, h. (b) Its normalized tree decomposition.

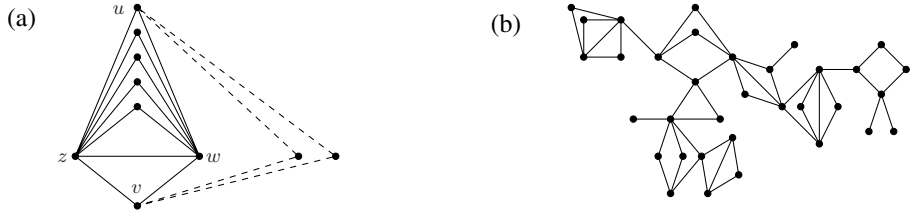

Fig. 2. (a) A graph with several spruces. (b) A connected spruce structure.

A *spruce* is a graph that has exactly two *base* vertices and at least one *tip* vertex, in which every tip vertex is adjacent to exactly the two base vertices. If the two base vertices are adjacent, the spruce is *complete*; otherwise it is incomplete. The *gain* of a spruce S is its cyclomatic number, and it is denoted $gain(S)$; this is the number of tips for complete spruces, and one less than the number of tips for incomplete spruces.

Fig. 2(a) depicts in solid lines a complete spruce with base vertices z and w, and six tip vertices including u and v. Another spruce contained in the same graph has base vertices u and v, and four tips including z and w; this second spruce is incomplete.

A *spruce cactus* is a graph such that each of its blocks is a spruce. A *spruce structure* is a graph each of whose blocks is a spruce or a bridge edge. See an example in Fig. 2(b).

Fact 1. *Spruce cactuses/structures are series-parallel graphs.*

We can view a spruce cactus as a collection of spruces — those giving the blocks of the spruce cactus. A spruce cactus is *well-behaved* if it is a collection of spruces that do not share tips. We define the *gain* of a spruce cactus to be its cyclomatic number.

Fact 2. *The gain of a spruce cactus equals the sum of the gains of its spruces.*

Before we proceed with the algorithm, we first elaborate on the need of spruces of unbounded size. First, if the input graph is a complete spruce with $n-2$ tips (and $2n-3$ edges), any approach which uses blocks of size bounded by, say, k, results in an output with gain at most $k-2$ and a total of $n+k-3$ edges. With n large and k fixed, this is only a $1/2$-approximation.

Our algorithm discards and shrinks selected spruces. Why one has to do this becomes clear from the following example, depicted in Fig. 3(a). The optimum has n vertices and $2n-3$ edges. It contains a spruce with base vertices x and y and circa $\sqrt{n}$ tips. For each of its tips v, there are two complete spruces, one with base vertices x and v, and the other with base vertices v and y, each with circa $\sqrt{n}/2$ tips. If an algorithm mistakenly (or greedily) selects the spruce with base vertices x and y, then it cannot add any more spruces and it ends up with circa $n+\sqrt{n}$ edges — asymptotically not better than a $1/2$-approximation.

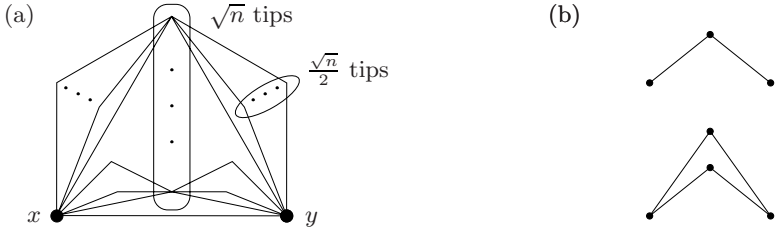

Fig. 3. (a) A graph where a naive greedy strategy that does not discard previously selected spruces fails to achieve a ratio better than 1/2. (b) The only two types of degenerate spruces.

For the weighted version of our problem, the algorithm that returns a maximum weight spanning tree is a 1/2-approximation. This follows from Lemma 3, which is also used in the analysis of our algorithm. Precisely, for any subgraph H' of an edge-weighted graph H, let $w(H')$ denote the sum of $w(e)$ for all e in $E(H')$. The proof of the next lemma follows closely that of Lemma 17 in [5].

Lemma 3. *Let F be a maximum weight forest in weighted simple series-parallel graph H. Then $w(H) \leq 2\,w(F)$, with the inequality being strict if $w(H) > 0$.*

Proof. We use the greedy algorithm to construct F, first sorting the edges of H into non-increasing order by weight. Let E_h be the set of the first h edges in this ordering, $1 \leq h \leq m$, where $m = |E(H)|$. By w_h we denote the weight of the h^{th} edge in this ordering and we put $w_{m+1} = 0$. Starting with $F = \emptyset$, the greedy spanning tree algorithm scans the edges in the given order and adds an edge to F as long as it does not create any cycles.

Let F be the set of edges chosen by the greedy algorithm and let $F_h = E_h \cap F$. Then, by rearranging the terms,

$$w(F) \;=\; \sum_{h=1}^{m} |F_h|(w_h - w_{h+1}), \text{ and } \; w(H) \;=\; \sum_{h=1}^{m} |E_h|(w_h - w_{h+1}).$$

It is therefore enough to show that $|E_h| < 2\,|F_h|$ for $1 \le h \le m$. If this holds, of course $w(H) \le 2\,w(F)$, and if $w_1 > 0$, the inequality is strict.

Choose an h such that $1 \le h \le m$. Let $p_1, p_2, \ldots, p_k$ be the number of vertices in the non-trivial components of F_h. Of course, $|F_h| = \sum_{z=1}^{k}(p_z - 1)$. Also note that $k \ge 1$, as F_h has at least one edge. Any edge of E_h must have its two endpoints in the same component of F_h. (Otherwise, the edge could have been selected by the greedy algorithm, merging two components of F_h.) Obviously this component is non-trivial. We associate each edge of E_h with the (non-trivial) component of F_h which contains both of its endpoints. The edges of E_h associated with a component of F_h are a subset of the edges of the graph induced in H by the vertices of this component. Thus, the number of edges associated with the z^{th} non-trivial component is at most $2p_z - 3$, because this graph is series-parallel . But then, as $k \ge 1$, we have that $|E_h| \le \sum_{z=1}^{k}(2p_z - 3) < \sum_{z=1}^{k} 2(p_z - 1) = 2\,|F_h|$. ■

2 A Local Improvement Algorithm

We may assume the input graph G is connected. Our local improvement algorithm, when running on G, keeps a set Q of spruces in G that form a well-behaved spruce cactus. We abuse notation and sometimes think of Q as the spruce cactus it forms.

The algorithm uses a slightly modified notion of gain. (One could also get an approximation ratio higher than 1/2 by only using gain in the algorithm, but we get a higher ratio.) For a spruce S, the *adjusted gain* of S is denoted by $\widehat{gain}(S)$, and is defined as $\widehat{gain}(S) = gain(S)$ if S is complete, and $\widehat{gain}(S) = gain(S) - 1$ if S is incomplete. We call a spruce *degenerate* if its adjusted gain is non-positive. See Fig. 3(b).

For each component C of Q, the algorithm keeps a weighted tree T_C whose vertex set is $V(C)$ and edge set is as follows. For each spruce S in C with base vertices x and y, and tips $v_1, v_2, \ldots, v_k$, there is an edge xy in T_C and edges xv_i for $i = 1, \ldots, k$. The weight of the edges is given as follows: $w(xy) = \widehat{gain}(S)$, and $w(xv_i) = 1$ for all i. Note that T_C is indeed a tree. For any two vertices x and y of C, let $index_Q(x, y)$ be an edge in T_C of minimum weight in the path in T_C from x to y. If x and y are in different components of Q, then let $index_Q(x, y)$ be undefined and consider its weight to be zero.

Let $v_1, v_2, \ldots, v_k$ be all vertices isolated in Q that are adjacent in G to both x and y. If $k \ge 1$, let $S_Q(x, y)$ be the spruce with base vertices x and y, tips $v_1, v_2, \ldots, v_k$, and the edge xy if it exists in G. Otherwise let $S_Q(x, y)$ be undefined.

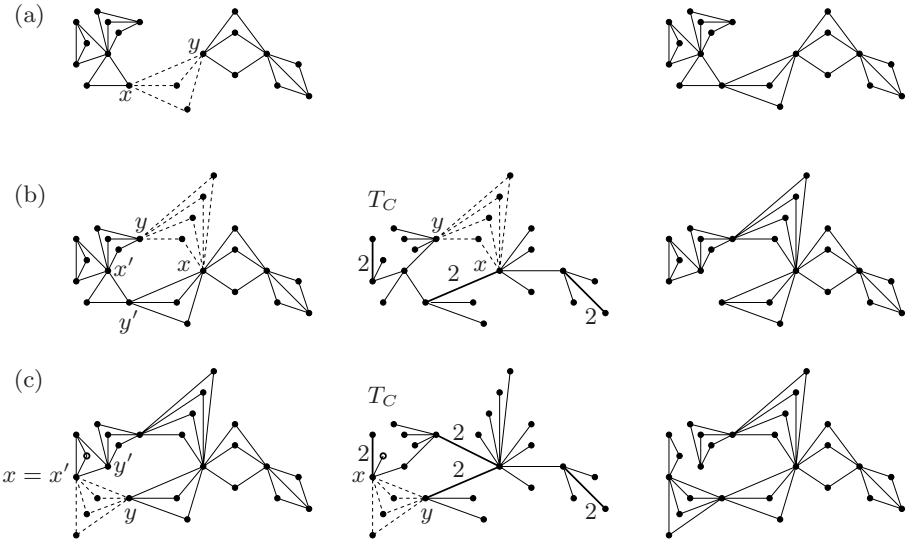

Fig. 4. Examples of local improvement, with $S_Q(x, y)$ given by the dashed lines in each case. (a) For such x and y, line 4 of the algorithm is executed resulting in Q as shown in the right. (b) For such x and y, line 7 of the algorithm is executed resulting in Q as shown in the right. The weighted tree T_C before the improvement is in the middle, with weights 1 except for those written in the figure. (c) For such x and y, line 7 and line 13 of the algorithm are executed resulting in Q as shown in the right.

The algorithm is shown in pseudocode later. We exemplify some of its cases in Fig. 4. Initially $Q = \emptyset$. The algorithm proceeds in iterations, each doing a local improvement. In each iteration, Q is updated as follows. If there are two vertices x and y of G for which $S_Q(x, y)$ is defined and $\widehat{gain}(S_Q(x, y)) > w(index_Q(x, y))$, then obtain a new Q' as follows, else go to the final phase. If $index_Q(x, y)$ is undefined, then let Q' be obtained from Q by adding $S_Q(x, y)$, and start a new iteration with Q' in the place of Q. Otherwise, let x' and y' be the endpoints of $index_Q(x, y)$, and C be the component of Q containing x, x', y, and y'. Let S' be the spruce in Q containing x' and y'. Note that such spruce exists by the construction of T_C. If x' and y' are the base vertices of S', then remove S' from Q and add $S_Q(x, y)$ to obtain Q'. Otherwise, by the construction of T_C, between x' and y' one is a base vertex of S', and the other is a tip of S'. Exchange x' and y' if needed so that x' is a base vertex of S'. Remove from S' the two edges incident to y'. If the resulting S' is degenerate or is a single edge, then remove S' from Q. Moreover, add $S_Q(x, y)$ to obtain Q', and start a new iteration with Q' in the place of Q.

Observe that, in this iterative part of the algorithm, we maintain the invariant that Q is a set of non-degenerate spruces that form a spruce cactus. Indeed, this follows by induction. It is enough to note that $\widehat{gain}(S_Q(x, y)) > 0$, and x and y are in different components, either from the start, or after we removed part or all of the spruce S' from Q.

The final phase consists of the following. Let Q now be the set of non-degenerate spruces produced by the iterative phase. Obtain a spanning connected subgraph of G from Q by adding bridges and let it be the output of the algorithm.

```
CONSTRUCT-SPRUCE-STRUCTURE (G)
1   Q ← ∅
2   while there are x and y such that S_Q(x,y) is defined
        and gain^(S_Q(x,y)) > w(index_Q(x,y)) do
3       if index_Q(x,y) is undefined
4           then Q ← Q ∪ {S_Q(x,y)}
5           else let x' and y' be the endpoints of index_Q(x,y)
6                let S' be the spruce in Q containing x' and y'
7                Q ← Q \ {S'} ∪ {S_Q(x,y)}
8                if x' or y' is a tip of S'
9                    then let z be one between x' and y' that is a tip of S'
10                        let {e,f} be the two edges of S' incident to z
11                        S ← S' − {e,f}
12                        if S is not degenerate neither a single edge
13                            then Q ← Q ∪ {S}
14  add bridges to Q to obtain a connected spanning subgraph of G
15  return Q
```

2.1 Running Time Analysis

The main result of this section is the very technical Lemma 4 below, which shows that each iteration makes some "progress". Unfortunately, the definition of "progress" is not straightforward, for the following reason.

A natural measure of progress would be the gain of Q (that is, its cyclomatic number). If $gain(Q)$ increased in every iteration, then it would have been easy to conclude that the algorithm runs a polynomial number of iterations. However this is not the case, and a more careful analysis is required. Let us give some intuition in this paragraph. One can check that, in most of the cases, the gain of Q increases. Also, it never decreases and, in the iterations in which the gain of Q is maintained, the number of components increases — more components are helpful since more, or bigger, spruces become eligible to improve the current Q.

Define $\Phi(Q) = 3\,gain(Q) + c(Q)$, where $c(Q)$ is the number of components of Q when Q is seen as a spanning subgraph of G.

Lemma 4. *Every iteration of the algorithm increases the parameter Φ.*

From this lemma, whose proof we omit, we conclude that the number of iterations is polynomially bounded, because $\Phi(Q)$ is a non-negative integer and $gain(Q) \leq (2n-3) - (n-1) = n-2$, which means $\Phi(Q)$ is bounded by $3(n-2) + n = 4n-6$.

Also, each iteration can be easily implemented in polynomial time, as there are only $O(n^2)$ pairs x, y for which $S_Q(x,y)$ must be computed and, if possible, used in updating Q.

2.2 Approximation Ratio Analysis

Let m be the number of edges in the graph Q returned by the algorithm. Then

$$m \;=\; n-1+\textstyle\sum_{S\in Q} gain(S).$$

Let A be an optimal solution for G and q be such that A has $2n-3-q$ edges. Thus, the algorithm achieves a ratio that is a constant greater than $1/2$ if

(i) $\sum_{S\in Q} gain(S)$ is at least a fraction of n, or
(ii) q is at least a fraction of n.

The analysis aims to prove that (i) or (ii) holds. Precisely, it will be shown that

$$6\textstyle\sum_{S\in Q} gain(S)+3q \;\geq\; n-2. \tag{1}$$

From this, it is easy to derive the 7/12 ratio:

$$m \;=\; n{-}1+\textstyle\sum_{S\in Q} gain(S) \;\geq\; n{-}1+\frac{1}{6}(n-2-3q) \;\geq\; \frac{7}{12}(2n-3-q).$$

The proof of Inequality (1) is not straightforward. We start by giving an overview. First we will derive a set M of spruces from A and prove that

$$\textstyle\sum_{S\in M} \widehat{gain}(S)+3q \;\geq\; n-2.$$

This is done in Lemma 5, later. Then, to achieve Inequality (1), it remains to prove that

$$6\textstyle\sum_{S\in Q} gain(S) \;\geq\; \sum_{S\in M} \widehat{gain}(S). \tag{2}$$

Consider Q to be the set of spruces when the algorithm finishes the iterations, and before the final phase (of adding bridges). Let t be the number of components of Q, and n' be the number of vertices in spruces of Q. Inequality (2) is a consequence of the following two inequalities:

$$4\textstyle\sum_{S\in Q} gain(S) \;\geq\; \sum_{S\in M} \widehat{gain}(S)-(n'-t),$$

which is given by Lemma 6, below, and

$$\textstyle\sum_{S\in Q} gain(S) \;\geq\; \frac{1}{2}(n'-t),$$

which is given by Lemma 7.

In what follows, we present the description of the set M of spruces, and proceed to Lemmas 5, 6, and 7.

Let A^+ be a maximal series-parallel graph containing A. Call the edges of A^+ not in A of *missing* edges. As A^+ is maximal, it can be obtained by the incremental procedure described in the preliminaries. For each edge xy of A^+ for which this procedure added at least one new vertex adjacent to x and y, consider a spruce S^+_{xy} in A^+ that has x and y as base vertices, and as tips all the vertices adjacent to x and y that were added in the procedure. As an example, in Fig. 1(a), spruce S^+_{as} has a and s as base vertices, and tips c, d, e. Let S_{xy} be a maximal spruce of A contained in S^+_{xy}, if such a spruce exists. Let $M=\{M_1, M_2, \ldots, M_k\}$ be the set of all such spruces S_{xy}. First, note that the spruces in M do not share tips. Also,

Lemma 5. $\sum_{S\in M} \widehat{gain}(S) + 3q \;\geq\; n-2$.

Proof. Observe that, as all S^+_{xy} are complete, the sum of $gain(S^+_{xy})$ for all x and y (for which S^+_{xy} is defined) equals the cyclomatic number of A^+, which is $2n-3-(n-1) = n-2$. Let us first argue that $\sum_{S\in M} gain(S) \geq n-2-2q$. Indeed each missing edge e decreases the sum of $gain(S^+_{xy})$ by at most two, because the edge e might appear in two spruces S^+_{xy} (once as xy and once as an edge incident to a tip of S^+_{xy}). Note also that a spruce S^+_{xy} for which S_{xy} is not a spruce corresponds to a term in the sum of $gain(S^+_{xy})$ that will become zero or negative after these discounts, so it does not hurt to drop it from the sum. Finally, the sum $\sum_{S\in M} \widehat{gain}(S)$ is equal to the sum $\sum_{S\in M} gain(S)$ minus the number of incomplete spruces in M, which is bounded above by q. Therefore, the lemma holds. ■

We proceed to Lemma 6.

Lemma 6. $4\sum_{S\in Q} gain(S) \geq \sum_{S\in M} \widehat{gain}(S) - (n'-t)$.

Proof. For $i = 1, 2, \ldots, k$, let U_i be the set of tips of M_i that are in some spruce of Q. Let S_i be obtained from M_i after the removal of its tip vertices in U_i. Note that S_i might not be a spruce (it might be empty or a single edge). If S_i is a spruce, then $\widehat{gain}(S_i) = \widehat{gain}(M_i) - |U_i|$. To simplify, set $\widehat{gain}(S_i) = 0$ if S_i is not a spruce.

The proof of this lemma has two steps. The first one consists of the following simple observation. As $\sum_i |U_i| \leq n'$, we have that

$$\sum_{S\in M} \widehat{gain}(S) \;=\; \sum_i \widehat{gain}(M_i) \;\leq\; n' + \sum_i \widehat{gain}(S_i), \tag{3}$$

because the spruces M_i do not share tips.

Let x and y be the base vertices of a spruce M_i from M. If x and y are in different components of Q, then S_i has to be a degenerate spruce or it is not a spruce (otherwise the algorithm would have included it in Q).

For each component C of Q, consider the following weighted simple graph $H = H_C$ on its set of vertices. For two vertices x and y in C that are the base vertices of a spruce S_i, the edge xy is present in H and it has weight $w(xy) = \widehat{gain}(S_i)$. Observe that H is a simple series-parallel graph. (It is a subgraph of A^+.)

Now, for the second step, let F_C be such a maximum weight forest in H. Recall that the algorithm constructs a weighted tree T_C on the same set of vertices; we treat the edges of T_C as distinct from the edges of F_C though both sets of edges have weight w. For each two vertices x and y with xy in F_C, there is a spruce S_i such that $w(xy) = \widehat{gain}(S_i)$. Now, the spruce $S_Q(x,y)$ was considered by the algorithm. Since Q is the set of spruces just before the final phase of the algorithm, $S_Q(x,y)$ was not added to Q and therefore $\widehat{gain}(S_Q(x,y)) \leq w(index_Q(x,y))$. Note that $\widehat{gain}(S_Q(x,y)) \geq \widehat{gain}(S_i)$ as all the tips of S_i, being isolated vertices in Q, are also in $S_Q(x,y)$. Thus, putting all this together, we have that $w(xy) = \widehat{gain}(S_i) \leq \widehat{gain}(S_Q(x,y)) \leq w(index_Q(x,y))$, for every x

and y such that $xy \in F_C$. But then, in the multigraph whose vertex set is C and the edge set is the disjoint union of $E(F_C)$ and $E(T_C)$, the tree T_C is a maximum weight tree [8]. Also, as F_C is a forest in this multigraph, we have that $w(F_C) \leq w(T_C)$.

Note that, for any spruce S in Q, the total weight of the edges of T_C obtained from S is $2\,gain(S)$, which holds both if S is complete or not. Let $\mathcal{C}$ be the collection of connected components of Q. Also, for C in $\mathcal{C}$, let Q_C be the (non-empty) set of spruces in C. By summing up for all spruces in Q, we obtain that

$$2 \sum_{C \in \mathcal{C}} gain(Q_C) \;=\; \sum_{C \in \mathcal{C}} w(T_C) \;\geq\; \sum_{C \in \mathcal{C}} w(F_C) \;\geq\; \frac{1}{2} \sum_{C \in \mathcal{C}} w(H_C) + \frac{1}{2}\, t,$$

where the last inequality comes from Lemma 3 and the fact that all weights are integers. Thus

$$2 \sum_{S \in Q} gain(S) \;=\; 2 \sum_{C \in \mathcal{C}} gain(Q_C) \;\geq\; \frac{1}{2} \sum_{i} \widehat{gain}(S_i) + \frac{1}{2}\, t.$$

and this, together with (3), implies the lemma. ∎

Now we proceed to Lemma 7.

Lemma 7. $\sum_{S \in Q} gain(S) \geq \frac{1}{2}\,(n' - t)$.

Proof. As in the previous proof, $\mathcal{C}$ is the collection of connected components of Q, and Q_C is the (non-empty) set of spruces in C, for C in $\mathcal{C}$. Let $n(C)$ be the number of vertices in C.

It is enough to prove that $gain(Q_C) \geq (n(C){-}1)/2$ for all C in $\mathcal{C}$. So, consider a C in $\mathcal{C}$, and recall that Q does not have degenerate spruces. Let us prove by induction on the number of spruces in Q_C that $gain(Q_C) \;\geq\; (n(C){-}1)/2$.

If Q_C has only one spruce S, then if S is complete, $n(S) = gain(S){+}2$, and thus $gain(S) = n(S){-}2 \geq (n(S){-}1)/2$ because $n(S) \geq 3$. If S is incomplete, $n(S) = gain(S){+}3$, and thus $gain(S) = n(S){-}3 \geq (n(S){-}1)/2$ because, as S is not degenerate, $n(S) \geq 5$.

Now suppose that Q_C has more than one spruce, and let S be a spruce in Q_C with at most one vertex in common with the others spruces in Q_C. (There is always one such spruce because Q_C is a spruce cactus.) Let C' be the connected subgraph of Q corresponding to the union of the spruces in $Q_{C'} = Q_C \setminus \{S\}$. By induction, $gain(Q_{C'}) \geq (n(C'){-}1)/2$. If S is complete, $n(C) = n(C') + gain(S) + 1$, and $gain(Q_C) = gain(Q_{C'}) + gain(S) \geq (n(C'){-}1)/2 + gain(S) = (n(C) - gain(S) - 2)/2 + gain(S) = (n(C) + gain(S) - 2)/2 \geq (n(C){-}1)/2$, because $gain(S) \geq 1$. If S is incomplete, $n(C) = n(C') + gain(S) + 2$, and $gain(Q_C) = gain(Q_{C'}) + gain(S) \geq (n(C'){-}1)/2 + gain(S) = (n(C) - gain(S) - 3)/2 + gain(S) = (n(C) + gain(S) - 3)/2 \geq (n(C){-}1)/2$, because $gain(S) \geq 2$, as S is non-degenerate. ∎

Having finished this proof, based on the discussion at the beginning of the subsection, we obtain the main result of the paper:

Theorem 1. *There is a polynomial-time $\frac{7}{12}$-approximation for Maximum Series-Parallel Subgraph.*

As an aside, observe that if we allowed the algorithm to include in Q the degenerate spruce which is a 4-cycle, then Lemma 7 would not hold anymore. Yet a weaker version of it would, with 1/3 instead of 1/2, and this would also lead to an approximation ratio greater than 1/2. We introduced the adjusted gain concept specifically to forbid 4-cycles, so that Lemma 7 holds with 1/2.

The analysis is tight. We will describe a family of graphs that proves this. Follow the description looking at Fig. 5. There is a graph G_k in this family for each even positive integer k. The graph G_k is the union of two edge-disjoint series-parallel graphs H_1 and H_2. The first one, H_1, is a path of length $8+k$, with a triangle on top of each of its edges (for a total of $7+k$ triangles and $3(7+k)$ edges). We call this path the *defining* path of H_1. In Fig. 5, the bottom edges form the defining path of H_1. The first 7 triangles on top of this path (shown by the darker edges) play a different role than the remaining k triangles. Call *top* the vertex in each of these triangles that is not on the defining path, and *round* the tops of the last k triangles plus the first and fourth top vertices. See the white circle vertices in Fig. 5. The final k vertices of the defining path are alternately named *square* and *triangular* vertices. The second and fifth top vertices are also square vertices, and the third and sixth are also triangular vertices. See Fig. 5. We will use these marks to describe the second graph.

The second graph, H_2, consists of three big spruces on the marked vertices of H_1, with a pair of new extra vertices per tip t, each of them adjacent to t and to one of the spruce base vertices. Each spruce is on one of the types of marked vertices in H_1. Let us now describe the first of the three big spruces, the one on the round vertices of H_1. This spruce has as base vertices the two first round vertices in H_1, and has as tips each of the other round vertices in H_1, for a total of k tips. In Fig. 5, this spruce is shown by the dotted edges, plus the "round" triangle with straight edges. For this triangle, we show also the two extra new vertices — the black small circle vertices, incident to the dashed edges.

The second big spruce is on the square vertices of H_1. Its base vertices are the two square top vertices, and its tips are the other $k/2$ square vertices of H_1. The third big spruce is defined similarly on the triangular vertices of H_1. This

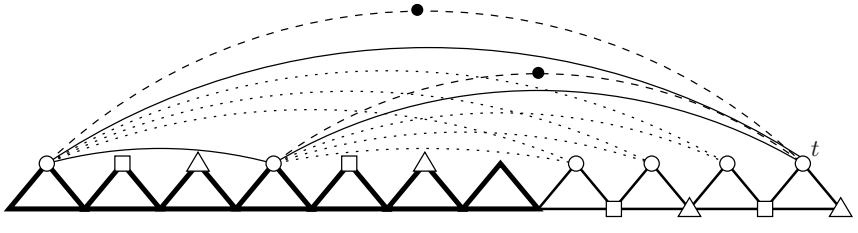

Fig. 5. Part of the graph G_4: the graph H_1 (the bottom path and the triangles on top of it), the first big spruce in H_2 (the subgraph induced by the white round vertices), and two extra vertices (the black small circle vertices)

completes the description of H_2, which, summarizing, consists of these three big spruces, plus the extra new vertices adjacent to the endpoints of the edges of these spruces incident to their tips. (In Fig. 5, we show only two of the extra vertices, the black small circle vertices.)

As we said, G_k consists of these two graphs H_1 and H_2. Note that both of them are indeed series-parallel. Thus, the number of edges in H_2, which is $(2k+1) + 2(k+1) + 8k = 12k+3$, is a lower bound on the size of a maximum series-parallel subgraph of G_k. Moreover, one can verify that our algorithm in the iterative phase can produce as Q the graph H_1, and output a graph with $|E(H_1)| + 4k = 3(7+k) + 4k = 7k+21$ edges. In this case, the ratio achieved is no more than $(7k+21)/(12k+3)$, which approaches $7/12$ as k gets large.

References

1. Berman, P., Ramaiyer, V.: Improved approximations for the Steiner tree problem. Journal of Algorithms 17, 381–408 (1994)
2. Cai, L.: On spanning 2-trees in a graph. Discrete Applied Mathematics 74(3), 203–216 (1997)
3. Cai, L., Maffray, F.: On the spanning k-tree problem. Discrete Applied Mathematics 44, 139–156 (1993)
4. Călinescu, G., Fernandes, C.G., Finkler, U., Karloff, H.: A better approximation algorithm for finding planar subgraphs. Journal of Algorithms 27(2), 269–302 (1998)
5. Călinescu, G., Fernandes, C.G., Karloff, H., Zelikovski, A.: A new approximation algorithm for finding heavy planar subgraphs. Algorithmica 36(2), 179–205 (2003)
6. Robins, G., Zelikovsky, A.: Improved steiner tree approximation in graphs. In: Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 770–779 (2000)
7. Schrijver, A.: Combinatorial Optimization, vol. A. Springer, Heidelberg (2003), `http://homepages.cwi.nl/~lex/files/dict.ps`
8. Tarjan, R.E.: Data Structures and Networks Algorithms. Society for Industrial and Applied Mathematics (1983)