Theory of Computing Systems (2020) 64:626–644
https://doi.org/10.1007/s00224-020-09965-w

Approximating Node-Weighted k-MST on Planar Graphs

Jarosław Byrka[1] · **Mateusz Lewandowski**[1] · **Joachim Spoerhase**[2]

Published online: 3 February 2020

Abstract

We study the problem of finding a minimum weight connected subgraph spanning at least k vertices on planar, node-weighted graphs. We give a $(4+\varepsilon)$-approximation algorithm for this problem. We achieve this by utilizing the recent Lagrangian-multiplier preserving (LMP) primal-dual 3-approximation for the node-weighted prize-collecting Steiner tree problem by Byrka et al. (SWAT'16) and adopting an approach by Chudak et al. (Math. Prog. '04) regarding Lagrangian relaxation for the edge-weighted variant. In particular, we improve the procedure of picking additional vertices (tree merging procedure) given by Sadeghian (2013) by taking a constant number of recursive steps and utilizing the limited guessing procedure of Arora and Karakostas (Math. Prog. '06). More generally, our approach readily gives a $(4/3 \cdot r + \varepsilon)$-approximation on any graph class where the algorithm of Byrka et al. for the prize-collecting version gives an r-approximation. We argue that this can be interpreted as a generalization of an analogous result by Könemann et al. (Algorithmica '11) for partial cover problems. Together with a lower bound construction by Mestre (STACS'08) for partial cover this implies that our bound is essentially best possible among algorithms that utilize an LMP algorithm for the Lagrangian relaxation as a black box. In addition to that, we argue by a more involved lower bound construction that even using the LMP algorithm by Byrka et al. in a *non-black-box* fashion could not beat the factor $4/3 \cdot r$ when the tree merging step relies only on the solutions output by the LMP algorithm.

Keywords Approximation algorithms · Node-weighted k-MST · Lagrangian relaxation · LMP · Planar graphs

This article belongs to the Topical Collection: *Special Issue on Approximation and Online Algorithms 2018*
Guest Editors: Leah Epstein and Thomas Erlebach

The authors were supported by the NCN grant number 2015/18/E/ST6/00456.

✉ Jarosław Byrka
jby@cs.uni.wroc.pl

Extended author information available on the last page of the article.

1 Introduction

We consider the node-weighted variant of the well-studied k-MST problem. Given a graph $G = (V, E)$ with non-negative node weights $c\colon V \to \mathbb{R}_+$ and a positive integer k, we consider the problem of finding a minimum cost connected subgraph of G spanning at least k vertices. In analogy to the edge-weighted case, we call this problem node-weighted k-MST (NW-k-MST) because the solution can be assumed to be a tree. In fact, we focus on the rooted variant in which a given vertex r has to be included in the final solution. To obtain the unrooted version, simply use the resulting algorithm for each choice of root vertex.

It was already observed that this problem is $\Omega(\log n)$-hard to approximate [17]. However, the problem becomes easier when we restrict G to be a planar graph. It is still NP-hard, as the edge-weighted variant is NP-hard even on planar graphs [18]. To this end, consider the following reduction from edge-weighted variant to the node-weighted variant. Each original vertex gets weight 0. Now, each edge e is replaced with a new vertex v_e of weight equal to the cost of e. Moreover, v_e is connected by two edges with original endpoints of e. Finally, each original vertex is connected to l new leaves of weight 0 where l is a parameter. It is easy to see, that for $l > |E|$, solutions for k-MST instances correspond to solutions to node-weighted $(k \cdot l + k - 1)$-MST instances after reduction and vice-versa.

The above reduction preserves planarity. Therefore, the focus of this work is to provide an approximation algorithm with small factor for planar NW-k-MST.

1.1 Related Work

1.1.1 Edge-Weighted k-MST

The standard, edge-weighted k-MST problem has been thoroughly studied. In a sequence of papers [1, 9, 10] the 2-approximation algorithm for prize-collecting Steiner tree problem [11] was used to finally obtain a 2-approximation algorithm for k-MST. These results can be, to some extent, explained as in the work of Chudak et al. [7] in terms of Lagrangian Relaxation.

In particular, a 5-approximation algorithm follows the framework known mostly from Jain and Vazirani's work on the k-median problem [12]. In these algorithms, the Lagrangian multiplier preserving (LMP) property plays a crucial role. The LMP property is also satisfied by the Goemans-Williamson algorithm for the prize-collecting Steiner tree problem (PC-ST). Intuitively, the LMP property of an α-approximation algorithm for some prize-collecting problem, means that the solutions it produces would also be not more expensive than α times optimum value even if we would have to pay α times more for penalties.

1.1.2 Node-Weighted k-MST

The NW-k-MST problem was already studied in the more general quota setting, where each node has an associated profit, and the goal is to find the minimum cost connected set of vertices having total profit at least Π. In particular, an

$O(\log n)$-approximation was given in [17]. However, this result was based on their invalid $O(\log n)$-approximation for NW-PC-ST. Recently, Chekuri et al. [6] and also independently Bateni et al. [2] proposed correct algorithms for generalizations of NW-PC-ST, but without LMP guarantee. The result on the quota problem was finally restored by Könemann et al. [14] who developed an LMP algorithm. In the related master thesis [19], Sadeghian gives also an alternative way of picking vertices[1] in the reduction for the quota problem. In these results, the constant lost in the process was not optimized.

1.1.3 Node-Weighted Planar Steiner Problems

Recently, the planar variants of Steiner problems received increased attention. In particular, Demaine et al. [8] obtained a 6-approximation for the node-weighted Steiner forest problem. The factor was further improved to 3 by Moldenhauer [16]. Both results rely on the moat-growing algorithm similar to that of Goemans and Williamson [11]. Currently the best result for this problem is the 2.4 approximation by Berman and Yaroslavtsev [3] who use a different oracle for determining violated sets.

More general network design problems on planar graphs where also studied by Chekuri et al. [5]. Finally, the result of Moldenhauer was generalized to the prize-collecting variant by Byrka et al. [4], resulting in an LMP 3-approximation for NW-PC-ST on planar graphs. We note that our result highly relies on this last algorithm.

1.1.4 Partial Cover

It can be seen, that our problem on arbitrary graphs generalizes the *partial cover* problem. In this problem we are given a set cover instance along with a positive integer k. The objective is to cover at least k ground elements by a family of sets of minimum cost. In the *prize-collecting* version of the problem every element has a penalty and the objective is to minimize the sum of costs of the chosen sets and the penalties of the elements that are not covered. Könemann et al. [13] describe a unified framework for partial cover. They show how to obtain an approximation algorithm for a class $\mathcal{I}$ of partial cover instances if there is an r-approximate LMP algorithm for the corresponding prize-collecting version. In particular, their result implies a $(\frac{4}{3}+\varepsilon)r$-approximation algorithm for the class $\mathcal{I}$. Mestre [15] shows that no algorithm that uses an LMP algorithm as a black box can obtain a ratio better than $\frac{4}{3}r$ so these results are essentially optimal.

1.2 Our Result and Techniques

We give a polynomial-time $(4+\varepsilon)$-approximation algorithm for the NW-k-MST problem on planar graphs. Our result extends to an algorithm for the quota node-weighted Steiner tree problem on planar graphs with the same factor.

[1] By picking vertices we mean augmenting the smaller solution with some vertices of larger solution. This is an important ingredient for the Lagrangian Relaxation technique

The main technique we use is the Lagrangian relaxation framework (as mentioned in the section above) where two solutions — one with fewer and the other with more than k nodes — are combined to obtain a feasible tree. The overview of our algorithm is as follows:

1. guess a skeleton and prune the instance
2. using the LMP algorithm [4], find trees T_1, T_2 with $\leq k$ and $\geq k$ nodes, respectively
3. combine T_1 and T_2 into a single tree with exactly k vertices

This is the standard design (although guessing step is not always necessary) of algorithms based on Lagrangian relaxation framework. However, in order to optimize the constant we employ additional ideas and techniques.

The first guessing step bears some similarities to that of Arora and Karakostas [1] where they improve Garg's 3-approximation for edge-weighted k-MST to $2+\varepsilon$. This additional guessing allows them to pay $\varepsilon \cdot \mathrm{OPT}$ instead of OPT for connecting a single set of vertices to the rest of the solution. Here, we provide a node-weighted variant of this idea and also use it more extensively, because we have to buy multiple (but still a constant number of) such connections. In our approach, we guess a set of vertices from optimum solution and call it a skeleton. Then, we can safely prune the instance ensuring that each remaining node will be not too far away from the skeleton. The guessing step is described in Section 2.

For the second step, we have to slightly modify the primal-dual LMP 3-approximation algorithm [4], so it returns solutions containing the guessed skeleton. This modification is technical and is described — together with the method used to find suitable T_1 and T_2 — in Section 4.

In the third step, we combine T_1 and T_2 by extending the procedure of picking vertices of Sadeghian [19]. He finds some cost-effective subset of vertices, which is two times larger than needed. We show that by picking vertices in certain order and applying recursion a *constant* number of times, we are able to pick almost exactly the number of nodes that is needed. Although, the number of components of this set of nodes might be arbitrary, we need to buy only a constant number of connections to restore connectivity. This is our main contribution and is described in the Section 3.

The resulting approximation factor of our algorithm is $(4+\varepsilon)$. Additionally, we show some evidence that our combining step is in some sense optimal. More precisely, we show that no other algorithm, using LMP 3-approximation as a black-box and which does not use planarity can give better constant than 4. This is obtained by interpreting our algorithm in terms of the results for the partial cover problem. The optimality of our algorithm within this framework is discussed in Section 5.

2 Pruning the Instance

First, we assume that we know OPT up to a factor $1+\varepsilon$ by using standard guessing techniques [9]. A node v is called *ε-distant* to a node set $U \subseteq V$ if there exists a path P in G from v to a node $u \in U$ of node weight $c(V(P) \setminus \{u\}) \leq \varepsilon \cdot \mathrm{OPT}$.

Lemma 1 *Consider an optimum solution T and an $\varepsilon > 0$. Then there exists a set $W \subseteq V(T)$ of size at most $1/\varepsilon$ such that each node in T is ε-distant to $W \cup \{r\}$.*

Proof Consider T as a tree rooted at r. For any node u in this tree let T_u denote the subtree hanging from u. A subtree T_u is called *good* if for any node in T_u the total weight of the unique path from this node to u within T_u (including the weight of the end nodes) is at most $\varepsilon \cdot \mathrm{OPT}$.

We traverse T in a bottom-up fashion starting with the leaves. We maintain the invariant (by removing subtrees) that for all nodes u visited so far and still being in T, the subtree T_u is good. To this end, when we encounter a node u such that T_u is good we just continue with the traversal. If T_u is bad, however, then there must be a path P within T_u ending in u of node weight $c(P) \geq \varepsilon \cdot \mathrm{OPT}$. We include u into W and assign P as a *witness* to u. Because of our invariant for all (if any) children v of u, we have that T_v is good. This means in particular that for all nodes z in T_u the node weight (*excluding* the weight of u) of the path from z to u is at most $\varepsilon \cdot \mathrm{OPT}$. Finally, remove T_u from T and continue with the traversal. We stop when we reach the root r at which point we remove the remaining tree (for the sake of analysis).

First, note that the set W has cardinality at most $1/\varepsilon$ because we assigned to each node in W a witness path of weight at least $\varepsilon \cdot \mathrm{OPT}$ and because the witness paths are pairwise node-disjoint. Second, observe that whenever we removed a node z from T as part of a subtree T_u, the node weight (excluding the weight of u) of the path from z to u was at most $\varepsilon \cdot \mathrm{OPT}$. Hence, for every node in T there exists such a path to a node in $W \cup \{r\}$ at the end of the tree traversal since every node was removed. □

In the sequel, we will call such a set W whose existence is provided by the above lemma an *ε-skeleton*.

In a pre-processing, we iterate over all $n^{\mathcal{O}(1/\varepsilon)}$ many sets $W' \subseteq V$ with $|W'| \leq 1/\varepsilon$ thereby guessing the ε-skeleton W whose existence is guaranteed by the above lemma. Moreover, we prune all nodes u from the instance that are not ε-distant to $W \cup \{r\}$.

3 The $(4 + \varepsilon)$-Approximation Algorithm

Sadeghian [19, Chapter 3] describes a $O(\log n)$ approximation for node-weighted quota Steiner tree problem. His result is established using a framework of [7], repeated also in [17] where a primal-dual LMP approximation algorithm for the prize-collecting Steiner tree problem can be used along with the Lagrangian relaxation method to obtain an approximation algorithm for the quota version of the problem. Sadeghian loses some large constant factor in the process. Direct application of his result would yield two digit approximation factor for our problem.

We now show that carefully injecting the LMP 3-approximation algorithm for NW-PC-ST on planar graphs given in [4] into his analysis yields a $(4 + \varepsilon)$-approximation. However, in the process, we need a more efficient way to pick additional vertices. We show that it is possible to pick a cheap set of these vertices.

Although it will not be connected, only a *constant* number of additional ε-distant vertices will suffice to connect the picked vertices.

For ease of the presentation, we will focus on the NW-k-MST problem. The algorithm for quota version can be then easily deduced by arguments of Bateni et al. [2]

The analysis relies on the following lemma.

Lemma 2 *We can produce trees T_1 and T_2 containing all the vertices W from the ε-skeleton and the root r of sizes $|T_1| \leq k \leq |T_2|$, such that for $\alpha_1, \alpha_2 \geq 0$ with $\alpha_1 + \alpha_2 = 1$ and $\alpha_1|T_1| + \alpha_2|T_2| = k$ we have that*

$$\alpha_1 c(T_1) + \alpha_2 c(T_2) \leq (3 + \varepsilon) OPT$$

The construction of these trees T_1 and T_2 and the proof of above lemma is described in Section 4.

Let now $q = k - |T_1|$ be the number of vertices that are missing from the tree T_1. We will now show, that these vertices can be picked from $T_2 \setminus T_1$ without paying too much.

Lemma 3 *It is possible to find a (not necessarily connected) set S of at least q vertices in $T_2 \setminus T_1$ of cost at most $(1 + \varepsilon_2)\alpha_2 c(T_2)$, which can be connected to T_1 by connecting additionally $\mathcal{O}(\log(1/\varepsilon_2))$ many ε-distant vertices to the ε-skeleton, where ε_2 is any constant.*

Proof Here, we substantially extend the analysis in [19]. Consider a graph T_2' constructed from T_2 by contracting all vertices from $T_1 \cap T_2$ to a single vertex r'. Define the cost of this vertex r' to 0 (we will buy T_1 anyway). From now on, whenever we count the cardinality of some subset S of vertices in T_2', we do not count vertex r'. □

Definition 1 A subset of vertices S is cost-effective if $\frac{c(S)}{|S|} \leq \frac{c(T_2')}{|T_2'|}$.

Lemma 4 *If cost-effective set S has size $(1 + \varepsilon_2)q$ then its cost is at most $(1 + \varepsilon_2)\alpha_2 c(T_2)$.*

Proof

$$c(S) \leq |S| \frac{c(T_2')}{|T_2'|} \leq (1 + \varepsilon_2) q \frac{c(T_2)}{|T_2| - |T_1|} \leq (1 + \varepsilon_2)\alpha_2 c(T_2),$$

where we used the fact that $\alpha_2 = \frac{k - |T_1|}{|T_2| - |T_1|}$.

So now, our goal is to find a cost-effective set S in T_2' of size only slightly larger that q. First, we start with a procedure for picking at most $2q$ vertices as in [19]. Initialize graph H with any spanning tree of T_2'. Observe that H is cost-effective. Consider any edge e of H. Let X and Y be the two components that would be created

after removing the edge e from H. At least one of these two components must be cost-effective. For any cost-effective component from this two, say X, do the following. If X has enough vertices, i.e. $|X| \geq q$, remove Y from H and continue. Otherwise, contract vertices of X to a single super vertex and set its cost to the sum of all vertices in X. We consider that the new super-vertex has *super-cardinality* equal to $|X|$.

It can be seen that after repeating this procedure as many times as possible, the graph H will be a star graph with super-cardinality of each leaf at most q. Let p be the number of leaves of H. In the case when $p \leq 1$ it is easy to see, that taking the whole graph H would result in a cost-effective set of vertices of size at most $2q$. Therefore, assume now that $p \geq 2$. Then, there exists a central vertex of the star graph H, call it c, which is not a super vertex. Moreover, every leaf v must be cost-effective (otherwise either we would remove v, or H would consist of two nodes). Observe also, that the super-cardinality of each leaf is at most q. Hence adding leaves to S one by one, would eventually lead to the set S with super-cardinality at most $2q$ (and at least q). Finally, S could be connected to T_1 by a single path from vertex c.

We now modify this procedure of adding leaves. First, consider them in the order of decreasing super-cardinalities. To this end, let $v_1, v_2, \ldots v_p$ be leaves of H and $s_1 \geq s_2 \geq \cdots \geq s_p$ be the corresponding super-cardinalities. Find the smallest i such that $\sum_{j=1}^{i} s_j + s_{i+1} \geq q$. If $s_{i+1} = 1$, then the desired set S consist of all vertices in $v_1, v_2, \ldots v_{i+1}$ and it has exactly q vertices. Otherwise, add the first i leaves to the set S. Let $t = \sum_{j=1}^{i} s_j$ be the number of vertices added to S. Now, instead of adding to S all vertices in the super vertex s_{i+1}, we expand this super vertex back to the original graph and repeat the above process with the new number of vertices to pick equal to $q' = q - t$. Observe that, because of sorting we have that $t \geq \frac{1}{2}q$, which also implies that $q' \leq \frac{1}{2}q$. This process is repeated recursively up to l times—where l is a parameter— but in the last call we take the last leaf completely (Fig. 1).

Let now $q_1, q_2, \ldots, q_l$ be the numbers of vertices to pick in respective recursive calls (note that $q_1 = q$ and $q_j \leq \frac{1}{2}q_{j-1}$). The total number of picked vertices is then at most $q + 2q_l \leq (1 + 2^{-l+2})q$. Therefore, to find the desired set S of at most $(1+\varepsilon_2)q$ vertices, we need only a constant number of recursive calls — parameter l is only $\mathcal{O}(\log(1/\varepsilon_2))$. Moreover all the vertices of S can be connected to T_1 by buying paths from the central nodes of all the l star graphs that appeared in the process. This finishes the proof. □

To construct a feasible solution, take the set S guaranteed by the above lemma and connect it to T_1 by the $\mathcal{O}(\log(1/\varepsilon_2))$ shortest paths to the ε-skeleton. Denote this solution by SOL_1. Let also SOL_2 be the entire tree T_2. Our algorithm outputs cheaper of the two solutions SOL_1 and SOL_2.

This enables us to prove the following.

Lemma 5 *Assuming $\varepsilon \leq 1$, the cost of the cheaper of the two solutions SOL_1 and SOL_2 is $(4 + O(\sqrt{\varepsilon})) \cdot \mathrm{OPT}$.*

Proof To bound the cost of the cheaper of two solutions SOL_1 and SOL_2 we employ the following Lemma by Könemann et al. [13].

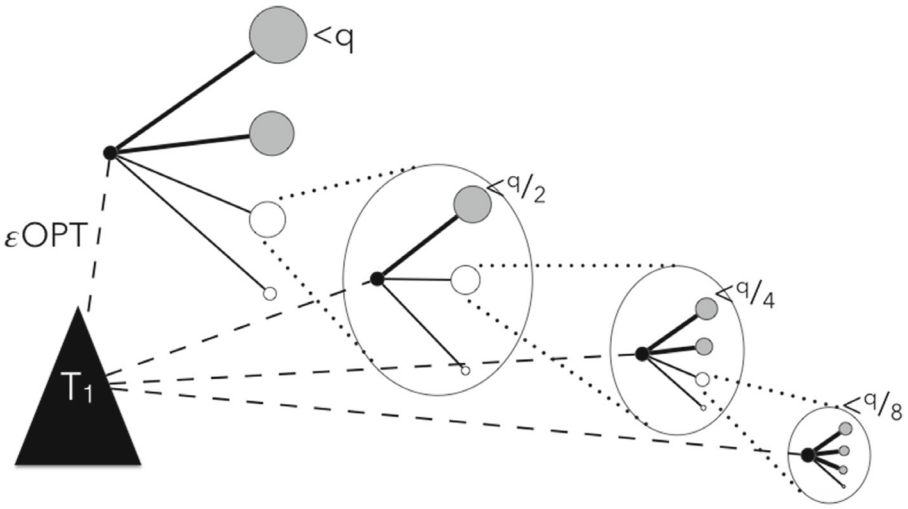

Fig. 1 Construction given in Lemma 3. The star graphs are build recursively. From each star graph, some vertices are picked (gray color) in the decreasing order of super-cardinality. Later, the cost-effective set of gray vertices is connected to T_1 via a constant number of shortest paths from centers of stars, forming SOL_1

Lemma 6 (Lemma 6 in [13]) *For any $r > 1$ and $\delta > 0$, we have*

$$\max_{\substack{\alpha \in (0,1)\\ \beta \in [0,r]}} \min\left\{\frac{r(1+\delta)-(1-\alpha)\beta}{\alpha}, r(1+\delta)+\alpha\beta\right\} = \left(\frac{4}{3}+O(\sqrt{\delta})\right) r\,.$$

The above equation comes from an optimization problem which can be solved by case analysis and balancing variables. For the proof, we refer the reader to the work of Könemann et al. [13]. Below, we will show how to use Lemma 6 to prove Lemma 5.

Let $\alpha = \alpha_2$ and $\beta = \frac{c(T_1)}{OPT}$. With this notation we obtain in a similar way as Könemann et al. [13]

$$\begin{aligned} c(\mathrm{SOL}_1) &\le c(T_1) + (1+\varepsilon_2)\alpha \cdot c(T_2) + \varepsilon \cdot \mathcal{O}(\log(1/\varepsilon_2)) \cdot \mathrm{OPT} \\ &\le \alpha \cdot c(T_1) + (1-\alpha) \cdot c(T_1) + (1+\varepsilon_2)\alpha \cdot c(T_2) + \varepsilon \cdot \mathcal{O}(\log(1/\varepsilon_2)) \cdot \mathrm{OPT} \\ &\le (3(1+\varepsilon_2)+\alpha\beta) \cdot \mathrm{OPT} + \varepsilon \cdot \mathcal{O}(\log(1/\varepsilon_2)) \cdot \mathrm{OPT}, \end{aligned}$$

and

$$\begin{aligned} c(\mathrm{SOL}_2) &= c(T_2) = \tfrac{\alpha \cdot c(T_2)}{\alpha} \\ &\le \tfrac{(3+\varepsilon)\mathrm{OPT} - (1-\alpha)c(T_1)}{\alpha} \\ &\le \tfrac{3(1+\varepsilon)-(1-\alpha)\beta}{\alpha} \cdot \mathrm{OPT}\,. \end{aligned}$$

By setting $r = 3$ and $\delta = \varepsilon = \varepsilon_2$ we obtain via Lemma 6 that the better of the two solutions has cost no more than $(4 + O(\sqrt{\varepsilon} + \varepsilon \log 1/\varepsilon)) \cdot \mathrm{OPT} = (4 + O(\sqrt{\varepsilon})) \cdot \mathrm{OPT}$ completing the proof. □

3.1 Generalization to Non-Planar Graph Classes

Note that in our algorithm, we use planarity exclusively by exploiting that the LMP algorithm of Byrka et al. [4] for the prize-collecting version has ratio 3 on planar graphs. Their algorithm, however, can be executed on an arbitrary graph class (e.g. H-minor-free graphs). Thus all our calculations can be carried through by replacing 3 with any factor $r \geq 1$ thereby obtaining the following generalization.

Corollary 1 *The above algorithm has performance* $(4/3 + \varepsilon)r$ *for any graph class where the algorithm of Byrka et al.* [4] *has a performance ratio of* r.

4 Lagrangian Relaxation and Moat Growing on Planar Graphs

In this section we prove Lemma 2. The proof utilizes Lagrangian Relaxation and follows a framework similar to the one in [7].

We start with the following LP relaxation for the NW-k-MST problem, where solutions are additionally constrained to contain all guessed vertices W of the ε-skeleton. For each vertex v we have the x_v variable indicating whether we will include this vertex in the solution. The z variables are indexed by sets of vertices not containing the root and the guessed vertices. There exists optimum integral solution, such that only the one z_X variable is set to 1. This would be for the set X of vertices not included in the final solution.

$$
\begin{aligned}
\min \quad & \textstyle\sum_{v \in V \setminus \{r\}} x_v c_v && (LP)\\
s.t. \quad & \\
& \textstyle\sum_{v \in \Gamma(S)} x_v + \sum_{\substack{X \,:\, S \subseteq X \\ X \cap W = \emptyset}} z_X \geq 1 && \forall S \subseteq V \setminus \{r\}\\
& x_v + \textstyle\sum_{\substack{X \,:\, v \in X \\ X \cap W = \emptyset}} z_X \geq 1 && \forall v \in V \setminus \{r\}\\
& \textstyle\sum_{X \subseteq V \setminus \{r\}} |X| z_X \leq n - k && (1)\\
& x_v \geq 0 && \forall v \in V \setminus \{r\}\\
& z_X \geq 0 && \forall X \subseteq V \setminus \{r\}
\end{aligned}
$$

The first two types of constraints guarantee connectivity of the solution to the root vertex and skeleton W. The $\Gamma(S)$ denotes the neighborhood of the set S, i.e. the set of vertices that are not in S, but have a neighboring vertex in S.

The constraint (1) ensures that the final solution will have at least k vertices and introduces difficulties. Therefore, we move it to the objective function obtaining the following Lagrangian Relaxation:

$$\begin{aligned}
&\min \ \textstyle\sum_{v\in V\setminus\{r\}} x_v c_v + \lambda\left(\sum_{X\subseteq V\setminus\{r\}} |X| z_X - (n-k)\right) \quad (LR(\lambda))\\
&s.t.\\
&\qquad \textstyle\sum_{v\in\Gamma(S)} x_v + \sum_{\substack{X\,:\,S\subseteq X\\ X\cap W=\emptyset}} z_X \geq 1 \qquad \forall S\subseteq V\setminus\{r\}\\
&\qquad x_v + \textstyle\sum_{\substack{X\,:\,v\in X\\ X\cap W=\emptyset}} z_X \geq 1 \qquad \forall v\in V\setminus\{r\}\\
&\qquad x_v \geq 0 \qquad \forall v\in V\setminus\{r\}\\
&\qquad z_X \geq 0 \qquad \forall X\subseteq V\setminus\{r\}
\end{aligned}$$

The above LP (ignoring the constant $-\lambda(n-k)$ term in the objective function) is exactly the LP for the node-weighted prize-collecting Steiner tree (NW-PC-ST in which the penalty of each vertex in $V' = V \setminus W$ is equal to the parameter λ) with a slight modification that the subset of vertices W is required to be in the solution.

Consider now, the dual of the $LR(\lambda)$:

$$\begin{aligned}
&\max \ \textstyle\sum_{S\subseteq V\setminus\{r\}} y_S + \sum_{v\in V\setminus\{r\}} p_v - \lambda(n-k) \quad (DLR(\lambda))\\
&s.t.\\
&\qquad \textstyle\sum_{S:v\in\Gamma(S)} y_S + p_v \leq c_v \qquad \forall v\in V\setminus\{r\}\\
&\qquad \textstyle\sum_{X\subseteq S} y_X + \sum_{v\in S} p_v \leq \lambda|S| \qquad \forall S\subseteq V'\setminus\{r\}\\
&\qquad y_S \geq 0 \qquad \forall S\subseteq V\setminus\{r\}
\end{aligned}$$

Now, we can leverage the slightly modified primal-dual LMP 3-approximation for (NW-PC-ST) given in [4].

Lemma 7 *There exists a polynomial time algorithm that given graph G with penalties λ and a subset of vertices W returns a tree T^λ and a dual solution (y^λ, p^λ) for $DLR(\lambda)$ such that:*

$$c(T^\lambda) + 3\lambda(n - |T^\lambda|) \leq 3\left(\sum_{S\subseteq V\setminus\{r\}} y_S^\lambda + \sum_{v\in V\setminus\{r\}} p_v^\lambda\right), \tag{2}$$

where T^λ contains all vertices of W.

The proof of the above lemma is deffered to Section 4.1.

Let us now see how we can use it to finish the proof of Lemma 2. We proceed essentially as in [19] and [7]. By subtracting $3\lambda(n-k)$ from both sides of

inequality (2) and simplifying the notation so that $\mathrm{DS}_\lambda = \sum_{S\subseteq V\setminus\{r\}} y_S^\lambda + \sum_{v\in V\setminus\{r\}} p_v^\lambda$ denotes the value of a dual solution we have that

$$\begin{aligned} c(T^\lambda) + 3\lambda(k - |T^\lambda|) &\leq 3\,(\mathrm{DS}_\lambda - \lambda(n-k)) \\ &\leq 3 \cdot \mathrm{DLR}(\lambda) \leq 3 \cdot \mathrm{OPT}. \end{aligned}$$

Observe that for $\lambda = 0$ the algorithm could output a tree with at least k vertices (because of moats growing around vertices in W, see next subsection). In this case the resulting tree is a 3-approximation so we do not need the merging procedure described in Section 3. Otherwise, for some large λ, e.g. the maximum cost of a vertex, the resulting tree would contain all the vertices. Therefore, we do the binary search for λ such that $|T^\lambda|$ is close to k. In a lucky event $|T^\lambda| = k$ and then we don't need the merging procedure described in Section 3. Otherwise, we obtain λ_1 and λ_2 such that $|T^{\lambda_1}| < k < |T^{\lambda_2}|$. By making enough steps of the binary search we can ensure that $\lambda_2 - \lambda_1 \leq \frac{\varepsilon\cdot\mathrm{OPT}}{3n}$. Let these trees be T_1 and T_2. Now, by setting $\alpha_1 = \frac{|T_2|-k}{|T_2|-|T_1|}$ and $\alpha_2 = \frac{k-|T_1|}{|T_2|-|T_1|}$ and using inequality (2) twice we have that

$$\begin{aligned} \alpha_1 c(T_1) + \alpha_2 c(T_2) &\leq 3\,(\alpha_1 \mathrm{DS}_1 + \alpha_2 \mathrm{DS}_2 - \alpha_1\lambda_1(n - |T_1|) - \alpha_2\lambda_2(n - |T_2|)) \\ &\leq 3\,(\alpha_1 \mathrm{DS}_1 + \alpha_2 \mathrm{DS}_2 - \lambda_2(n-k) + (\lambda_2 - \lambda_1)(n - |T_1|)) \\ &\leq 3\,(\mathrm{OPT} + (\lambda_2 - \lambda_1)n) \\ &\leq (3+\varepsilon)\,\mathrm{OPT}, \end{aligned}$$

where we used the fact that the convex combination of DS_1 and DS_2 is a feasible solution for $\mathrm{DLR}(\lambda_2)$.

4.1 Moat Growing

In this section we describe the slight technical modification needed in the primal-dual algorithm for NW-PC-ST problem on planar graphs given in [4] and give the proof of Lemma 7. Observe, that there are two differences in the LPs used (comparing above $DLR(\lambda)$ to DLP_{PCST} in [4]).

First, we have to include some guessed vertices W in the solution. However, this can be easily guaranteed by assigning to them a sufficiently large penalty. Second, we have additional constraints and corresponding dual variables p_v. This is due to the fact, that in our setting all vertices can have both nonzero penalty and cost, while in the previous setting the reduction step was employed so that each vertex is a terminal with some penalty and zero cost or a Steiner vertex with zero penalty. However, this reduction can also be done as follows. For every vertex $v \in V$, set p_v to the minimum of cost and penalty. Then define the reduced costs and reduced penalties. This does not influence the approximation factor, nor the LMP guarantee.

For completeness, we now give a description of the resulting LMP primal-dual algorithm. First, we assign an infinite penalty to guessed vertices W. Then, we eliminate p_v variables as described above.

The algorithm maintains a set of moats, i.e., a family of disjoint sets of vertices. In each step, these moats can be viewed as the components of the graph induced by the so far bought nodes. Each moat has an associated potential equal to the total penalty

of vertices inside this moat minus the sum of the dual variables for all the subsets of this moat. The moats with positive potential are active, with the exception that the moat containing the root is always inactive.

The algorithm raises simultaneously the dual variables of all the active moats. For the growth of a moat we pay with its potential. We can have two events.

In the first event, some vertex goes tight, i.e., the inequality for this vertex in the dual program becomes tight. In this case we buy this vertex and merge all the neighboring moats, setting the potential accordingly to the sum of all previous moats' potentials. We declare this new moat inactive whenever it contains a root vertex.

In the second event, some moat goes tight, i.e. the inequality in the dual program becomes tight for some set of vertices. This corresponds to the situation when the potential of this moat drops to zero. In this case we declare this moat inactive and we mark all the previously unmarked terminals inside it as marked with the current time. Observe that in the dual we do not have these inequalities for sets containing guessed vertices W. This means, that all the vertices of W will be connected to the root vertex.

We repeat this process until we do not have any active moats. Then we start a pruning phase. We consider all the bought vertices in the reverse order of buying. We delete a vertex v if the removal of v would not disconnect any unmarked terminal or any terminal marked with time greater than the time of buying the vertex v. We return the pruned set of bought vertices as the solution.

We now give a more formal proof of Lemma 7.

Proof of Lemma 7 Set cost c_v of every vertex $v \in V$ to 1. Now, for every vertex $w \in W$, set its penalty π_w to infinity (or a large number, e.g. $|V|$ is enough). For all other vertices $v \in V \setminus W$, set $\pi_v = \lambda$. Then, for every vertex $v \in V$, define $p_v^\lambda = \min\{c_v, \pi_v\}$. Finally for every $v \in V$, let $c'_v = c_v - p_v^\lambda$ and $\pi'_v = \pi_v - p_v^\lambda$.

We run the algorithm from Section 2 of [4] on graph G with costs c' and penalties π'. By Theorem 1 in [4], this algorithm produces tree T^λ and dual variables y_S^λ feasible for the DLP_{PCST} such that

$$c'(T^\lambda) + 3\pi'(V \setminus T) \leq 3 \sum_{S \subseteq V \setminus \{r\}} y_S^\lambda$$

By adding $3\sum_{v \in V \setminus \{r\}} p_v$ to both sides of this inequality, we get inequality (2). Now, we claim that $W \subseteq T^\lambda$. Because of the huge potential, the moats containing vertices of W will be active until they connect to root r, therefore every $w \in W$ will remain unmarked. As pruning phase removes only some marked vertices, the claim follows. It remains to show that (y^λ, p^λ) is also a feasible solution to $DLR(\lambda)$. This however follows immediately by adding p_v^λ to both sides of inequality (1) (and respectively $\sum_{v \in S} p_v$ to inequality (2)) of DLP_{PCST} in [4]. □

5 Trying to Beat the Factor of 4: Relation to the Partial Cover

Here we draw connections to the recent work on the partial cover problems. Könemann et al. [13] showed how to obtain a $(4/3 + \varepsilon)r$-approximation algorithm

for the partial cover problems using an r-approximate LMP algorithm for the corresponding prize-collecting version as a black-box. Their approach is roughly as follows. First, the most expensive sets from the optimum solution are guessed and all sets which are more expensive are discarded. Further, the black-box algorithm is used together with binary search to find two solutions, one, say S_1, feasible but possibly expensive, and the other, say S_2, infeasible but inexpensive. Then the merging procedure is employed to obtain a solution S_3. Finally, the cheapest solution of the S_1 and S_3 is returned.

5.1 Generalizing the Algorithm of Könemann et al.

Extending a folklore reduction from set cover type problems to node-weighted Steiner tree problems, we argue that our algorithm may be interpreted as a non-trivial generalization of the above-outlined algorithm by Könemann et al. [13].

First of all, the following reduction shows that the partial covering problem can be encoded as the quota node-weighted Steiner tree problem. The reduction creates for each element a vertex with zero cost and profit 1. Then, for each set it creates a node with the same cost and zero profit and connects it to the elements covered by this set. Finally, the root vertex is added and connected to all the set-corresponding nodes. The target quota profit is set to be the same as the requirement for the partial cover problem.

For such a reduced instance, we can run the preprocessing step from Section 2 which will remove the expensive sets (we could also employ the Könemann's preprocessing beforehand). Then, we would run any LMP algorithm for the prize-collecting cover problems within the Lagrangian relaxation framework which would indicate two families of sets to merge. Putting it on the reduced instance, these would correspond to two trees to merge. More precisely, take to the tree the set-corresponding nodes, the root vertex and the elements covered by sets. Now, we can apply the merging procedure described in the Lemma 3 with a slight adjustment needed to account for quota variant. In particular we modify the notion of cost-effectiveness to account profits instead of cardinalities and we also redefine the super-cardinality to be the sum of profits. To retrieve the solution from the tree, simply take the sets corresponding to non-zero cost nodes in the tree. Finally, output the cheaper of the two feasible solutions giving a partial cover with the same quality as the one by obtained via the algorithm by Könemann et al.

We remark that the above argument does not work in the reverse direction. The graph instances that are created have a very specific structure with three node layers ensuring that any partial cover solution is automatically connected at no additional cost. Achieving connectivity for *general* graphs, however, is not implied and guaranteeing this structural property without loss in the performance guarantee of the algorithm can be seen as a main contribution of our work.

5.2 Black-Box Optimality

Now, the above reduction, together with a lower bound construction by Mestre [15] implies that our approach is best possible using the LMP algorithm as a black-box

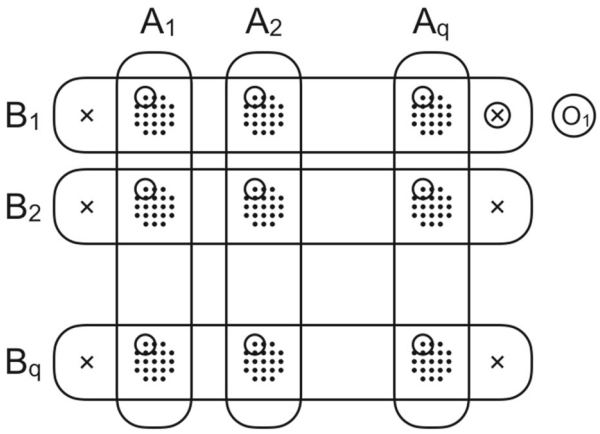

Fig. 2 The instance of partial cover given by Mestre [15]

and without referring to the underlying graph class. To see this, observe, that the Mestre's construction given in Theorem 3.1 in [15], can be transformed to an instance of quota node-weighted Steiner tree instance by using the above reduction.

Here, we repeat the Mestre's example, as we will extend it further. Fix some integer constant q. The instance consists of q^3 ground elements aligned in the grid of size q by q with q elements in each cell. Then we have q sets $A_1, A_2, \ldots A_q$, each covers all elements in the corresponding column of a grid. Analogously, we have q sets $B_1, B_2, \ldots B_q$ which cover rows. Moreover, each B_i set has two more ground elements. Then, we have q sets $O_1, O_2, \ldots O_q$, where set O_i covers i-th element from each cell of the grid and a single element which is also covered by B_i. This construction is illustrated in Fig. 2, where the set O_1 is marked with circles. Then, costs of sets are defined as follows: $c(A_i) = \frac{2}{3} \cdot \frac{r}{q}$, $c(B_i) = \frac{4}{3} \cdot \frac{r}{q}$, $c(O_i) = \frac{1}{q}$, where $r = 3$ in our case.

For such instance, optimal prize-collecting cover is either the empty cover or $A_1, A_2, \ldots A_q$ or $B_1, B_2, \ldots B_q$ or $O_1, O_2, \ldots, O_q$. Now, the target value of k is set to $q^3 + q$ for which the optimal partial cover are O-sets. Now, if the LMP algorithm returns A-sets for some penalties λ, and B-sets for slightly larger penalties λ^+, then — as Mestre shows — the merging procedure can do no better than $\frac{4}{3} \cdot r$ approximation. Thus, without knowing the inner-workings of the LMP algorithm, one can not obtain a better approximation algorithm for partial cover using the general method of merging LMP solutions. For details regarding this construction, we refer to the original work of Mestre [15].

Consider now the corresponding quota Steiner tree instance of the above partial cover instance. This construction is not giving a complete lower bound for our problem by two reasons. First, the instance is not planar, second, the LMP algorithm is not exemplifying the proof of Mestre's Lemma 3.3 [15].[2] Indeed, the LMP algorithm

[2]This lemma states that there exists an LMP algorithm which returns either sets A or B (depending on the initial penalty λ).

would buy cheap O-vertices straight ahead. Despite these two issues, this instance is still useful in the following sense: it already shows that in order to beat the factor $4/3 \cdot r$ for NW-k-MST, we would need to consider the inner-workings of the LMP algorithm or the underlying graph class (e.g. planarity).

5.3 Inner-Workings are Not Enough

Finally, we extend Mestre's example to show that even examining the inner-workings of the algorithm of Section 4.1 without referring to the underlying graph class (such as planar graphs) in the merging procedure is not enough to beat the factor of $4/3 \cdot r$. We do this by giving a modified instance for which the LMP algorithm of Section 4.1 returns either A or B sets. We will work with the instance of node-weighted prize-collecting Steiner tree problem obtained from Mestre's construction via our reduction.

To build an intuition, we point out two main difficulties concerning the reduced instance with respect to the moat-growing algorithm.

First, the Steiner vertices corresponding to A and B sets are much more expensive related to O-vertices. Since the number of terminals adjacent to these Steiner vertices is roughly the same, the O-vertices would be bought by the algorithm first and remain in our solution. In order to solve this problem, we will introduce a so-called handicap gadget, which will allow us to enforce earlier buying time of more expensive vertices.

The second problem is a technical issue. The costs of Steiner vertices are affected by penalties before growth phase (see the reduced costs of Section 4.1). This makes deriving claims about buying time of vertices more difficult. To alleviate this inconvenience we will aggregate more potential on terminals. We now describe the two constructions solving the two above problems.

5.3.1 The Potential Aggregation

We propose the following modification of an instance. Connect by stars a large amount of new terminals to every terminal t, each t having its own set of $\gamma \gg q$ terminals. Observe now, that in the beginning of the GROW phase, the moat-growing algorithm for each terminal t will immediately form a single component consisting of t and added terminals. The total potential of this moat will be equal to $\gamma \cdot \lambda$. Therefore, the initial potential λ needed for moat-growing algorithm to grow moats reaching Steiner vertices can be arbitrarily small. This in turn, allows us to insist that reduced cost of Steiner vertices is comparable to the original cost.

5.3.2 The Handicap Gadget

We introduce a gadget which allows to significantly reduce the buying time of expensive A and B vertices so that they would go tight at the same time and also much earlier than the cheaper O-vertices would.

The gadget consist of a grid of terminals with q columns and q^2 rows. Each B-vertex is connected to every vertex of a grid. Each A-vertex is connected only to all vertices inside $\frac{q}{2}$ columns. These columns are assigned in a way that each column is

assigned to at least one A vertex. Finally each vertex O_i is connected to all vertices in column i. It can be seen that in the growth phase of the LMP algorithm, the B vertices gain their contribution to cost roughly two times faster than A vertices. Since B vertices are twice as expensive, after adding this gadget, the buying time of A and B should be now roughly the same and much smaller than that of O vertices.

5.3.3 Finishing the Construction

Here, we describe the final construction and analyze the behavior of the algorithm from Section 4.1 on this instance. We extend the instance from Section 5.2. Recall, that each set corresponds now to a Steiner vertex which is also directly connected to the root vertex. On top of that construction, add the handicap gadget.

After adding the stars for potential aggregation the number of terminals is now $\gamma \cdot (2 \cdot q^3 + 2 \cdot q)$. By handicap gadget, the buying time of A and B vertices is roughly the same. Finally set the target k appropriately, i.e. $k = (2 \cdot q^3 + q) \cdot \gamma$. Let I be the resulting instance.

Lemma 8 *There exist the initial potential λ such that, the LMP algorithm of Section 4.1 for the instance I returns the A solution, while for the slightly larger potential λ^+ it returns the B solution.*

Proof First, we claim that, there is an initial potential λ_1 for which all the A vertices will be bought, but neither B, nor O vertices. To see this, observe that the buying time of the first vertex is computed by dividing the cost of a vertex by number of neighboring terminals (moats) and taking a minimum. Let $t(A), t(B), t(O)$ be these ratios for vertices of corresponding type. We have that:

$$\begin{aligned} t(A) &= \left(\tfrac{4}{3} \cdot \tfrac{r}{q} - \lambda_1\right) \cdot \tfrac{1}{q^2 + \frac{1}{2}q^3} \\ t(B) &= \left(\tfrac{2}{3} \cdot \tfrac{r}{q} - \lambda_1\right) \cdot \tfrac{1}{q^2 + 2 + q^3} \\ t(O) &= \left(\tfrac{1}{q} - \lambda_1\right) \cdot \tfrac{1}{q^2 + 1 + q} \end{aligned}$$

Note, that by potential aggregation, λ_1 is arbitrarily small. Thus, for q large enough we have that $t(A) < t(B) < t(O)$. Now, the λ_1 that we are looking for is computed from the equation $t(A) = \gamma \cdot \lambda_1$.

Now, increase the λ value starting from λ_1 to the point just before that B-vertices get tight. Now, at some slightly larger initial potential λ^+, the B-vertices will be bought ending the growth phase. However, the pruning phase of the moat-growing algorithm will then keep all the B-vertices and prune all the A-vertices. □

Now, using analogous arguments as in the result of Mestre [15] we conclude the following.

Corollary 2 *For any $r > 1$ there is an infinite family of graphs where the natural moat growing algorithm for NW-PC-ST [4] has a ratio r but where any feasible*

solution to the NW-k-MST problem using only the nodes returned by this algorithm has cost at least $4/3 \cdot r$ *times that of an optimum solution.*

5.3.4 Interpretation

In the edge-weighted case of k-MST, Garg [10] was able to carefully exploit the inner workings of the Goemans-Williamson algorithm [11] for the Lagrangian relaxation to match its ratio of 2. Corollary 2 means that our approach is in a certain sense optimal and that we would need to deviate from this framework to improve on the loss of factor $4/3$ in the tree-merging step. This could possibly be achieved by exploiting structural properties of the underlying graph class or using nodes outside the solution returned by the LMP algorithm.

Even when we exploit planarity it seems to be non-trivial to beat factor 4 along the lines of Garg [9, 10]. The changes in the solutions by increasing initial potentials of vertices can be much larger than those in the edge-weighted variant. In particular, one can observe situations of node-flips in which two potentially distant vertices exchange their presence in the solution. Also, in contrast to edge-weighted variant, a single node can be adjacent to any number of moats and not only two. This in turn causes a large difference in two trees produced by the algorithm. In particular, the *old vertices* as described by Garg [9] (think of them as the vertices of $T_1 \cap T_2$) can form any number of connected components which may be expensive to connect even when the graph is planar.

6 Conclusions and Comments

The $4 + \varepsilon$ approximation factor was obtained for the NW-k-MST problem on planar graphs. In the process we used the Lagrangian Relaxation technique. Our work can be interpreted as a generalization of a work on partial cover [13]. The result by Mestre [15] implies that our factor is essentially best possible using the underlying LMP algorithm for the NW-PC-ST as a black-box. It shows that one would have to exploit planarity in the merging process to beat factor 4.

Our ultimate hope would be to match the factor of 3 of the LMP algorithm. We think that the question of whether this is possible is very interesting and challenging.

Acknowledgements We would like to thank Zachary Friggstad for initial discussions on the problem.

References

1. Arora, S., Karakostas, G.: A $(2+\varepsilon)$-approximation algorithm for the k-MST problem. Math. Program. **107**(3), 491–504 (2006). https://doi.org/10.1007/s10107-005-0693-1
2. Bateni, M., Hajiaghayi, M., Liaghat, V.: Improved approximation algorithms for (budgeted) node-weighted Steiner problems. In: Proc. 40th International Colloquium on Automata, Languages, and Programming (ICALP'13), pp. 81–92 (2013). https://doi.org/10.1007/978-3-642-39206-1_8
3. Berman, P., Yaroslavtsev, G.: Primal-dual approximation algorithms for node-weighted network design in planar graphs. In: Proc. 15th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX'12), pp. 50–60 (2012). https://doi.org/10.1007/978-3-642-32512-0_5
4. Byrka, J., Lewandowski, M., Moldenhauer, C.: Approximation algorithms for node-weighted prize-collecting Steiner tree problems on planar graphs. In: Proc. 15th Scandinavian symposium and workshops on algorithm theory (SWAT'16), pp. 2:1–2:14 (2016). https://doi.org/10.4230/LIPIcs.SWAT.2016.2
5. Chekuri, C., Ene, A., Vakilian, A.: Node-weighted network design in planar and minor-closed families of graphs. In: Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP'12), pp. 206–217 (2012). https://doi.org/10.1007/978-3-642-31594-7_18
6. Chekuri, C., Ene, A., Vakilian, A.: Prize-collecting survivable network design in node-weighted graphs. In: Proc. 15th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX'12), pp. 98–109 (2012). https://doi.org/10.1007/978-3-642-32512-0_9
7. Chudak, F.A., Roughgarden, T., Williamson, D.P.: Approximate k-MSTs and k-Steiner trees via the primal-dual method and lagrangean relaxation. Math. Program. **100**(2), 411–421 (2004). https://doi.org/10.1007/s10107-003-0479-2
8. Demaine, E.D., Hajiaghayi, M.T., Klein, P.N.: Node-weighted Steiner tree and group Steiner tree in planar graphs. ACM Trans. Algor. **10**(3), 13,1–13,20 (2014). https://doi.org/10.1145/2601070
9. Garg, N.: A 3-approximation for the minimum tree spanning k vertices. In: Proc. 37th Annual Symposium on Foundations of Computer Science (FOCS'96), pp. 302–309 (1996). https://doi.org/10.1109/SFCS.1996.548489
10. Garg, N.: Saving an epsilon: A 2-approximation for the k-MST problem in graphs. In: Proc. 37th Annual ACM Symposium on Theory of Computing (STOC'05), pp. 396–402 (2005). https://doi.org/10.1145/1060590.1060650
11. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM J. Comput. **24**(2), 296–317 (1995). https://doi.org/10.1137/S0097539793242618
12. Jain, K., Vazirani, V.V.: Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. J. ACM **48**(2), 274–296 (2001). https://doi.org/10.1145/375827.375845
13. Könemann, J., Parekh, O., Segev, D.: A unified approach to approximating partial covering problems. Algorithmica **59**(4), 489–509 (2011). https://doi.org/10.1007/s00453-009-9317-0
14. Könemann, J., Sadeghabad, S.S., Sanità, L.: An LMP $O(\log n)$-approximation algorithm for node weighted prize collecting Steiner tree. In: Proc. 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS'13), pp. 568–577 (2013). https://doi.org/10.1109/FOCS.2013.67
15. Mestre, J.: Lagrangian relaxation and partial cover. In: Proc. 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS'08), pp. 539–550 (2008). https://doi.org/10.4230/LIPIcs.STACS.2008.1315
16. Moldenhauer, C.: Primal-dual approximation algorithms for node-weighted Steiner forest on planar graphs. Inf. Comput. **222**, 293–306 (2013). https://doi.org/10.1016/j.ic.2012.10.017
17. Moss, A., Rabani, Y.: Approximation algorithms for constrained node weighted Steiner tree problems. SIAM J. Comput. **37**(2), 460–481 (2007). https://doi.org/10.1137/S0097539702420474
18. Ravi, R., Sundaram, R., Marathe, M.V., Rosenkrantz, D.J., Ravi, S.S.: Spanning trees short or small. In: Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. 23-25 January 1994, Arlington, Virginia, pp. 546–555 (1994)
19. Sadeghabad, S.: Sina: Node-weighted prize-collecting Steiner tree and applications. Master's thesis (2013)

Affiliations

Jarosław Byrka[1] · Mateusz Lewandowski[1] · Joachim Spoerhase[2]

Mateusz Lewandowski
mlewandowski@cs.uni.wroc.pl

Joachim Spoerhase
joachim.spoerhase@uni-wuerzburg.de

[1] Institute of Computer Science, University of Wroclaw, Wroclaw, Poland

[2] Lehrstuhl fur Informatik I, Universitat Wurzburg, Wurzburg, Germany