Covering a graph with a constrained forest (extended abstract)

Cristina Bazgan[1], Basile Couëtoux[1], and Zsolt Tuza[2⋆]

[1] Université Paris-Dauphine, LAMSADE, France
{bazgan,couetoux}@lamsade.dauphine.fr
[2] Computer and Automation Institute, Hungarian Academy of Sciences, Budapest and Department of Computer Science, University of Veszprém, Hungary
tuza@sztaki.hu

Abstract. Given an undirected graph on n vertices with weights on its edges, Min WCF(p) consists of computing a covering forest of minimum weight such that each of its tree components contains at least p vertices. It has been proved that Min WCF(p) is *NP*-hard for any $p \geq 4$ (Imielinska *et al.*, 1993) but $(2 - \frac{1}{n})$-approximable (Goemans and Williamson, 1995). While Min WCF(2) is polynomial-time solvable, already the unweighted version of Min WCF(3) is *NP*-hard even on planar bipartite graphs of maximum degree 3. We prove here that for any $p \geq 4$, the unweighted version is *NP*-hard, even for planar bipartite graphs of maximum degree 3; moreover, the unweighted version for any $p \geq 3$ has no ptas for bipartite graphs of maximum degree 3. The latter theorem is the first-ever APX-hardness result on this problem. On the other hand, we show that Min WCF(p) is polynomial-time solvable on graphs with bounded treewidth, and for any p bounded by $O(\frac{\log n}{\log \log n})$ it has a ptas on planar graphs.

1 Introduction

Let $G = (V, E)$ be a graph with $|V| = n$ vertices. An edge cover of G is a subset of the edge set E such that every vertex is incident with at least one edge in the covering set. Finding the minimum size, $\rho(G)$, of an edge cover of a graph is a fundamental problem. As proved by Gallai [8], it is strongly related to determining the maximum size, $\nu(G)$, of a matching in G. A famous result of [8] states that any graph G without isolated vertices satisfies the identity $\nu(G)+\rho(G) = n$. As a matter of fact, the relation is much more than quantitative: every maximum matching of a graph can be extended to a minimum edge cover, and also conversely, every minimum edge cover contains a maximum matching. In this way, one can derive a minimum-size edge cover from a maximum matching M just by adding an arbitrary incident edge for each vertex missing from M. Hence, a minimum-size edge cover can be found in polynomial time.

⋆ Research supported in part by the Hungarian Scientific Research Fund, OTKA grant T-049613.

In the case where the graph $G = (V, E)$ has weights on its edges, the minimum-weight edge cover problem can be reduced to the problem of finding a minimum-weight perfect matching; a simple reduction is described e.g. in the first volume of Schrijver's monograph [15, Section 19.2]. As a consequence, an optimal solution can be found in $O(n^3)$ time by the results of Edmonds and Johnson [7]. It should be noted, however, that the relation between maximum matchings and minimum edge covers does not remain valid for weighted graphs, neither for uniform hypergraphs of edge size greater than two [16].

A problem that generalizes the minimum-weight edge cover problem in a very natural way is the Min Weighted Constrained Forest problem, denoted by Min WCF(p) in the sequel. It consists of computing a spanning forest of G of minimum weight such that every tree component contains at least p vertices, for a given integer p. Although traditionally p is assumed to be a constant, the methods proving our positive results will allow us to take p as a function of n, too.

Monnot and Toulouse [14] proved that the unweighted version of Min WCF(3) is *NP*-hard even on planar bipartite graphs of maximum degree three. Imielinska *et al.* [10] showed that Min WCF(p) is *NP*-hard for $p \geq 4$, and that a greedy algorithm achieves a 2-approximation. Interestingly enough, a different algorithm studied by Laszlo and Mukherjee [12] has exactly the same tight worst-case ratio of 2, as well as a common generalization of those two approaches [13]. With the methods of Goemans and Williamson [9], just a slightly better ratio $2 - \frac{1}{n}$ can be achieved.

Let us denote by Min CF(p) the unweighted version of the problem. Until now nothing was known about the complexity of Min CF(p) for $p \geq 4$. We settle this problem by showing that Min CF(p) is *NP*-hard for any $p \geq 4$, already on planar bipartite graphs with maximum degree three.

Moreover, we study non-approximability of these problems for the first time. In this direction we prove that dropping the condition of planarity, Min CF(p) becomes *APX*-hard for any $p \geq 3$ on bipartite graphs, and even on those with maximum degree three. It also turns out that this weakening in the condition necessarily has to appear in non-approximability results, since we can design a polynomial-time approximation scheme for Min WCF(p) on planar graphs. In this result we may allow p to be bounded by $O(\frac{\log n}{\log \log n})$. An important tool in the proof is an algorithm computing an optimal solution for Min WCF(p) on any input graph of treewidth at most k in $O(k^{ck} p^{2k+2} n)$ time, for some constant c. This time bound is valid without any restrictions on the growth of p, hence applicable also in graph classes which cannot be treated with Courcelle's powerful method via monadic second-order logic [4]. (The latter would require to fix p as a constant.)

It is worth noting that the unweighted case admits a much simpler approach than the weighted one, with the following improved approximation ratio:

Remark 1. Contrary to the weighted case, it is very easy to find increasingly better approximations for Min CF(p) as p gets large. Indeed, since every feasible solution has at most n/p connected components, the optimum can never have

a value smaller than $n - n/p = \frac{p-1}{p}n$, and hence any spanning tree gives a $(1 + \frac{1}{p-1})$-approximation.

Our results on NP- and APX-hardness are proved in Section 3, while efficient algorithms are presented in Section 4. Due to space limitations, some proofs are not given in the paper.

2 Preliminaries

We begin with some basic definitions, summarized in two groups.

Problems. First, we formally define the problem we will study in the sequel.

Min Weighted Constrained Forest p (Min WCF(p))
Input: An undirected graph $G = (V, E)$ with non-negative weights on its edges.
Output: A spanning forest of minimum weight where every tree is of order at least p (i.e., it contains at least p vertices).

The *unweighted* version of Min WCF(p) will be denoted by Min CF(p).

In our proofs we shall use the following problems:

3-Dimensional Matching (3DM)
Input: Three disjoint sets A, B, C of the same size q, and a set $\mathcal{T} \subseteq A \times B \times C$ of triplets.
Question: Does $\mathcal{T}$ contain a perfect matching, that is a subset $\mathcal{M} \subseteq \mathcal{T}$ such that $|\mathcal{M}| = q$ and no two elements of $\mathcal{M}$ agree in any coordinate (i.e., for any $(a, b, c), (a', b', c') \in \mathcal{M}$ we have $a \neq a'$, $b \neq b'$, and $c \neq c'$)?

We can associate a bipartite graph with any instance of 3DM as follows. We take an 'element-vertex' for each element of $A \cup B \cup C$, and one 'triplet-vertex' for each triplet in $\mathcal{T}$. There is an edge connecting a triplet-vertex to an element-vertex if and only if the element is a member of the triplet. Moreover, we say that the instance is planar if the graph associated with it is planar.

Max 3-Dimensional Matching (Max 3DM)
Input: Three disjoint sets A, B, C of the same size and a set $\mathcal{T} \subseteq A \times B \times C$ of triplets.
Output: A matching (set of mutually disjoint triplets) $\mathcal{M} \subseteq \mathcal{T}$ of maximum size.

The restricted versions of 3DM and of Max 3DM where each element of $A \cup B \cup C$ appears in exactly two triplets will be denoted by $3DM_2$ and Max $3DM_2$, respectively.

Approximability. Given an instance x of an optimization problem A and a feasible solution y of x, we denote by $v(x, y)$ the value of the solution y, and by $opt_A(x)$ the value of an optimum solution of x. The *performance ratio* of y is

$$R(x, y) = \max\left\{\frac{v(x, y)}{opt_A(x)}, \frac{opt_A(x)}{v(x, y)}\right\}.$$

For a constant $c > 1$, an algorithm is a *c-approximation* if for any instance x of the problem it returns a solution y such that $R(x, y) \leq c$. An optimization problem is said to be *constant approximable* if, for some $c > 1$, there exists a polynomial-time c-approximation for it. The class of problems which are constant approximable is denoted by APX. An optimization problem has a *polynomial-time approximation scheme* (a *ptas*, for short) if, for every constant $\varepsilon > 0$, there exists a polynomial-time $(1 + \varepsilon)$-approximation for it.

3 Complexity for $p \geq 3$

In this section we prove that the constrained forest problem is intractable for every $p \geq 4$, even on some very restricted classes of problem instances. We first deal with time complexity and then with approximation hardness.

3.1 *NP*-hardness for $p \geq 4$, planar bipartite unweighted graphs

Theorem 1. *For any $p \geq 4$,* Min CF(p) *is NP-hard, even on planar bipartite graphs with maximum degree 3.*

Proof. First, we prove *NP*-hardness for $p = 4$. We construct a polynomial reduction from 3DM to Min CF(4). Let $I = (A, B, C, \mathcal{T})$ be an instance of 3DM where $|A| = |B| = [C| = q$ and $\mathcal{T} = \{T_1, \ldots, T_m\}$. Assume, without loss of generality, that each element occurs in at least one triplet (otherwise I admits no perfect matching at all).

The graph instance $G(I) = (V, E)$ of Min CF(4) is constructed as follows; see an illustration in Figure 1 with a particular instance I of 3DM. For each triplet $T_\ell = (a_i, b_j, c_k) \in \mathcal{T}$ we create a copy of a star with 3 branches, that we shall call *star gadget*, with vertices $a_i^\ell, b_j^\ell, c_k^\ell, d^\ell$ and edges $(a_i^\ell, d^\ell), (b_j^\ell, d^\ell), (c_k^\ell, d^\ell)$. These stars are assumed to be vertex-disjoint. For their union we denote

$$V_T = \bigcup_{T_\ell = (a_i, b_j, c_k) \in \mathcal{T}} \{a_i^\ell, b_j^\ell, c_k^\ell, d^\ell\} \qquad \text{and}$$

$$E_T = \bigcup_{T_\ell = (a_i, b_j, c_k) \in \mathcal{T}} \{(a_i^\ell, d^\ell), (b_j^\ell, d^\ell), (c_k^\ell, d^\ell)\}.$$

To the elements of $A \cup B \cup C$ we assign *linking gadgets*; if an element appears in h triplets, then $h - 1$ gadgets will be associated with it.

Consider first the set A. We define the set $J_A \subset A \times \mathbb{N} \times \mathbb{N}$ to be the collection of all (a_i, r, s) with the following properties: $a_i \in A$, and r, s are two consecutive indices of triplets containing a_i, in the sense that $a_i \in T_r$ and $a_i \in T_s$ but $a_i \notin T_\ell$ for any $r < \ell < s$. (In this general setting we allow that some a_i occur in just one member of $\mathcal{T}$ — hence generating no linking gadgets — although in such a situation I would easily be reducible to a smaller instance.) Each $(a_i, r, s) \in J_A$

defines a linking gadget inducing a 'claw' in $G(I)$, with vertices $a_i^r, a_i^s, a_i^{r,s}, \overline{a}_i^{r,s}$ and edges $(a_i^r, a_i^{r,s}), (a_i^{r,s}, a_i^s), (a_i^{r,s}, \overline{a}_i^{r,s})$. For their union we denote

$$V_A = \bigcup_{(a_i,r,s)\in J_A} \{a_i^{r,s}, \overline{a}_i^{r,s}\} \quad \text{and} \quad E_A = \bigcup_{(a_i,r,s)\in J_A} \{(a_i^r, a_i^{r,s}), (a_i^{r,s}, a_i^s), (a_i^{r,s}, \overline{a}_i^{r,s})\}.$$

The sets J_B and J_C are defined in a similar way. Each $(b_j, r, s) \in J_B$ defines a linking gadget inducing a 'claw' in $G(I)$, with vertices $b_j^r, b_j^s, b_j^{r,s}, \overline{b}_j^{r,s}, \underline{b}_j^{r,s}$ and edges $(b_j^r, b_j^{r,s}), (b_j^{r,s}, b_j^s), (b_j^{r,s}, \underline{b}_j^{r,s}), (\underline{b}_j^{r,s}, \overline{b}_j^{r,s})$. For their union we denote

$$V_B = \bigcup_{(b_j,r,s)\in J_B} \{b_j^{r,s}, \underline{b}_j^{r,s}, \underline{b}_j^{r,s}\} \qquad \text{and}$$

$$E_B = \bigcup_{(b_j,r,s)\in J_B} \{(b_j^r, b_j^{r,s}), (b_j^{r,s}, b_j^s), (b_j^{r,s}, \underline{b}_j^{r,s}), (\underline{b}_j^{r,s}, \overline{b}_j^{r,s})\}.$$

The same is done for the set C in complete analogy to B, hence introducing the linking gadgets $(c_k, r, s) \in J_C$ and obtaining vertex set V_C and edge set E_C.

After all these preparations, let the graph $G(I) = (V, E)$ has vertex set $V = V_T \cup V_A \cup V_B \cup V_C$ and edge set $E = E_T \cup E_A \cup E_B \cup E_C$. Since an element of $A \cup B \cup C$ appearing in h triplets has h copies and $h-1$ linking gadgets in $G(I)$, we can see that the total number of vertices is precisely $12|\mathcal{T}| - 8q$. Hence, the transformation is linear with respect to input size.

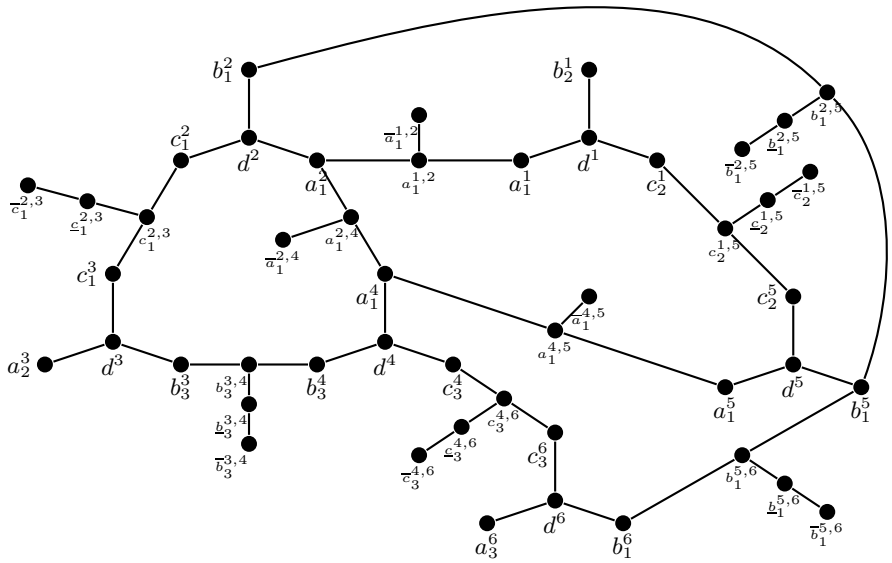

Fig. 1. Graph $G(I)$ derived from instance $I = (A, B, C, \mathcal{T})$ of 3DM with $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $C = \{c_1, c_2, c_3\}$, and $\mathcal{T} = \{T_1, \ldots, T_6\}$ with $T_1 = (a_1, b_2, c_2)$, $T_2 = (a_1, b_1, c_1)$, $T_3 = (a_2, b_3, c_1)$, $T_4 = (a_1, b_3, c_3)$, $T_5 = (a_1, b_1, c_2)$, $T_6 = (a_3, b_1, c_3)$.

We are going to show that I contains a perfect matching if and only if $G(I)$ contains a forest of weight at most $9|\mathcal{T}| - 6q$ in which every tree component is of order at least 4.

Suppose first that $\mathcal{M}$ is a perfect matching of I. For a $T_\ell = (a_i, b_j, c_k) \in \mathcal{M}$ we write $f_M(a_i) = f_M(b_j) = f_M(c_k) = \ell$, and further denote by F_T the set of edges of the star gadgets corresponding to triplets of $\mathcal{M}$:

$$F_T = \bigcup_{T_\ell = (a_i, b_j, c_k) \in \mathcal{M}} \{(a_i^\ell, d^\ell), (b_j^\ell, d^\ell), (c_k^\ell, d^\ell)\}$$

and

$$F_{a_i} = \bigcup_{\substack{r < f_M(a_i) \\ (a_i, r, s) \in J_A}} \{(d^r, a_i^r), (a_i^r, a_i^{r,s}), (a_i^{r,s}, \overline{a}_i^{r,s})\} \quad \cup$$

$$\bigcup_{\substack{s > f_M(a_i) \\ (a_i, r, s) \in J_A}} \{(d^s, a_i^s), (a_i^s, a_i^{r,s}), (a_i^{r,s}, \overline{a}_i^{r,s})\},$$

$$F_{b_j} = \bigcup_{\substack{r < f_M(b_j) \\ (b_j, r, s) \in J_B}} \{(b_j^r, b_j^{r,s}), (b_j^{r,s}, \underline{b}_j^{r,s}), (\underline{b}_j^{r,s}, \overline{b}_j^{r,s})\} \quad \cup$$

$$\bigcup_{\substack{s > f_M(b_j) \\ (b_j, r, s) \in J_B}} \{(b_j^s, b_j^{r,s}), (b_j^{r,s}, \underline{b}_j^{r,s}), (\underline{b}_j^{r,s}, \overline{b}_j^{r,s})\}.$$

The set F_{c_k} is defined like F_{b_j}. Let $F_M = \bigcup_{e \in A \cup B \cup C} F_e \cup F_T$. Forest F_M covers $G(I)$ with trees of order 4. Thus, it is a forest of weight $9|\mathcal{T}| - 6q$.

Conversely, let F be a covering forest of $G(I)$ of weight at most $9|\mathcal{T}| - 6q$, where each tree is of order at least 4. Remark that since the graph $G(I)$ has exactly $12|\mathcal{T}| - 8q$ vertices, a covering forest without isolated edges and isolated vertices is of weight at least $9|\mathcal{T}| - 6q$. Thus, every tree in F is of order exactly 4, and F is of weight $9|\mathcal{T}| - 6q$. All edges $(a_i^{r,s}, \overline{a}_i^{r,s})$ are in F since this is the only edge incident to $\overline{a}_i^{r,s}$. Since F has only trees of order 4, just one of the edges $(a_i^{r,s}, a_i^r)$ and $(a_i^{r,s}, a_i^s)$ is in F. Therefore in the family $\{a_i^\ell\}_\ell$ there exists exactly one vertex that is not incident to a linking gadget in F. Since forest F is of weight $9|\mathcal{T}| - 6q$, it contains $3|\mathcal{T}| - 2q$ trees. Since for any element of A, B, C the number of linking gadgets associated is equal to the number of occurrences of this element in $\mathcal{T}$ minus 1, there are $3|\mathcal{T}| - 3q$ linking gadgets in $G(I)$. It means that there are q trees of F which do not cover any linking gadget. Each of these remaining trees therefore covers a star gadget of the form $a_i^\ell, b_j^\ell, c_k^\ell, d^\ell$. From these star gadgets we extract a collection of q triplets (a_i, b_j, c_k), which is a valid solution in I since every triplet appears in $\mathcal{T}$ by construction and every element of the triplets appears in exactly one occurrence.

Since 3DM is *NP*-hard even on planar instances [5] and the previous reduction preserves planarity, we can restrict Min CF(4) to planar bipartite graphs of maximum degree 3. This completes the proof for $p = 4$. The construction of $G(I)$ for $p > 4$ consists of replacing some edges of $G(I)$ for $p = 4$ by paths of length $p - 3$. □

3.2 *APX*-hardness for $p \geq 3$, unweighted bipartite graphs

Applying L-reduction from MAX 3DM$_2$, the following result can be obtained.

Theorem 2. *For any $p \geq 3$,* MIN CF(p) *is* APX*-hard. Moreover,* MIN CF(p) *for $p \geq 3$ is not $\frac{95(8p-7)+1}{95(8p-7)}$-approximable on bipartite graphs of maximum degree 3, unless P=NP.*

4 Efficient algorithms

4.1 Exact algorithms for bounded treewidth graphs

In this section we present an efficient algorithm solving MIN WCF(p) on graph classes of bounded treewidth. For undefined details on tree decomposition we refer to [11].

Theorem 3. *The problem* MIN WCF(p) *on graphs with treewidth bounded by k can be solved in time $O(k^{ck}p^{2k+2}n)$ for some constant c, where n is the number of vertices.*

Remark 2. The time bound is polynomial in n whenever $k(\log k + \log p) = O(\log n)$ and in particular if $k = O(\frac{\log n}{\log\log n})$ and $p = O(\log n)$.

Proof. Let G be a graph of treewidth k, and $(T_G, \{X \mid x \in V(T_G)\})$ a tree decomposition of G with width k. As a notational convention, the vertex subset of G associated with node x of T_G is denoted by the corresponding upper-case letter X, and we shall use the same subscript for them where necessary. We view T_G as a *rooted* tree. By assumption, the set X associated with any node x of T_G contains at most $k+1$ vertices of $V(G)$. We shall assume further that every node x is of one of the following types:

- a *start* node that has no children (a leaf in T_G),
- a *join* node that has two children x_1, x_2 and $X_1 = X_2 = X$,
- an *introduce* node that has one child x_1 and $X_1 = X \setminus \{v\}$ for some $v \in V(G)$,
- a *forget* node that has one child x_1 and $X_1 = X \cup \{v\}$ for some $v \in V(G)$.

This is called a "nice tree decomposition" in the literature, see pp. 149–150 of [11]. As it is well known, every graph admits a tree decomposition of size $O(n)$ satisfying these conditions. It is also easy to see that any tree decomposition of width k can be modified to one with the properties above in $O(kn)$ time.

For any node x of T_G, let $T_G(x)$ be the rooted subtree of T_G containing exactly the node x and its descendants. The vertices of G which appear in the sets associated with the nodes of $T_G(x)$ define a subgraph of G, denoted by $G(x)$. Remark that $T_G(x)$ is a tree decomposition of $G(x)$.

Consider any node x of T_G. For each spanning forest F (possibly with many vertices of degree zero in F) of the subgraph $G[X]$ induced by X in G, we consider all partitions $\mathcal{P} = (C_1, \ldots, C_\ell)$ of X and all ℓ-tuples $I = (i_1, \ldots, i_\ell) \in \{1, \ldots, p\}^\ell$ such that the following conditions are met:

- if $v_i, v_j \in X$ are in the same tree component of F, then v_i, v_j belong to the same partition class C_r,
- if $|C_r| \leq p$, then $|C_r| \leq i_r \leq p$, and otherwise $i_r = p$.

Let $f(x, F, \mathcal{P}, I)$ be defined as the value of a minimum-weight spanning forest of $G(x)$ in which X induces precisely F, and in which two vertices belong to the same tree component if and only if they are in the same class of $\mathcal{P}$.

Lemma 1. *For any vertex x of T_G, for any forest F over the subgraph $G[X]$, for any partition $\mathcal{P}$ of X, and for any ℓ-tuple I of integers satisfying the conditions above, we can determine $f(x, F, \mathcal{P}, I)$ in $O(k^{ck}p^{2k+2})$ time if the corresponding values f are available for the child(ren) of x.*

Sketch of Proof. One has to consider each type of nodes separately. Due to space limitations, we give the argument for *join* nodes only, which is the most time-consuming case. If x is a *join* node, then $\mathcal{P}$ has to be generated from two finer partitions over X, one for X_1 and the other for X_2. We can do this by artificially completing F with sets E_{blue}, E_{red} of blue and red edges (not necessarily edges of G) to obtain a forest, each tree component of which spans a class of $\mathcal{P}$. We then consider the forests F_{blue} and F_{red} which are respectively the forests defined by $F \cup E_{blue}$ and $F \cup E_{red}$. The tree components of these forests define the partitions $\mathcal{P}_{blue}$ and $\mathcal{P}_{red}$ over the vertices of X. The vectors I_1 and I_2 are such that, for every partition class C_j of $\mathcal{P}$, i_j is equal to the minimum of p and the sum of the values of the blue and red classes included in C_j minus $|C_j|$. Then,

$$f(x, F, \mathcal{P}, I) = \min_{F_{blue}, F_{red}, I_1, I_2} f(x_1, F, \mathcal{P}_{blue}, I_1) + f(x_2, F, \mathcal{P}_{red}, I_2) - w(F).$$

The number of relevant blue-red forests is not larger than the number B_{k+1} of partitions of a $(k+1)$-element set. For each of them, processing all combinations of $O(p^{k+1})$ records at X_1 and the same amount of data at X_2 may need $O(p^{2k+2})$ time. This yields the upper bound $B_{k+1} \cdot O(p^{2k+2})$ altogether. □

To conclude the proof of Theorem 3, we traverse T_G in postorder, and compute $f(x, F, \mathcal{P}, I)$ for all possible choices of node x, spanning forest F of $G[X]$, partition $\mathcal{P}$ of X, and sequence I satisfying the conditions given at the beginning of the proof. Then the solution of Min WCF(p) on G is equal to the smallest value of f occurring at the root of T_G for a sequence $I = (p, \ldots, p)$ of any length.

The overall upper bound is obtained by the facts that for any k, the number of choices for $\mathcal{P}$ is bounded above by B_{k+1}, the number of vertex-labeled trees of order t is t^{t-2} (this puts a bound on the choices for F), and the number of sequences I at x is at most p^{k+1}. The most time-consuming step is to compute f at a *join* node; it can be organized by taking all combinations of the values stored at x_1 and x_2. □

Remark 3. The same method can be applied to solve the more general problem where, instead of a uniform condition p for the constrained forest, the input graph $G = (V, E)$ on vertex set $V = \{v_1, \ldots, v_n\}$ is given together with a vector

$(p_1, \ldots, p_n)$ of natural numbers $p_i \leq p$, and it is required that each v_i be contained in a tree component which has at least p_i vertices in the spanning forest. The steps of the algorithm above can be adjusted for this variant.

4.2 Ptas for planar graphs

Theorem 4. *For* $p = O(\frac{\log n}{\log \log n})$, MIN WCF($p$) *on planar graphs admits a ptas.*

Proof. Given a planar embedding of an input graph, we consider the set of the vertices which are on the exterior face, they will be called *level 1 vertices*. By induction we define *level* k as the vertices which are on the exterior face when we have removed the vertices of levels smaller than k [1]. A planar embedding is *k-level* if it has no nodes of level greater than k. If a planar graph is k-level, it has a k-outerplanar embedding.

If we want to achieve an approximation within $1 + \epsilon$, let us consider $k = \lceil \frac{4(p-1)}{\epsilon} \rceil$. Let X_t be the set of vertices of level t and let H_i, $0 \leq i \leq k-1$, be the graph obtained from G by deleting all edges between the vertices X_t and X_{t+1} for all t such that $t \equiv i \pmod{k}$. The set of vertices $\bigcup_{t+1 \leq j \leq t+k} X_j$, for $t \equiv i \pmod{k}$ is therefore a component in H_i since we have deleted all edges from this set of vertices to the other vertices of the graph. Hence, the subgraph containing exactly $\bigcup_{t+1 \leq j \leq t+k} X_j$ is k-outerplanar, and so is H_i.

Since H_i is k-outerplanar, it has treewidth at most $3k-1$ [2]. By applying Theorem 3 and Remark 3, we can efficiently determine an optimal forest S_i on H_i; the condition on p remains true on vertices not incident with edges deleted, and is changed to zero on the boundary vertices of H_i, that means vertices in X_{t+1}, X_{t+k} with $t \equiv i \pmod{k}$. We complete the solution S_i obtained on H_i into a feasible solution F_i on G by choosing the useful edges of smallest cost greedily within the edges of G until every tree is of order at least p.

We are going to prove that the best forest among $F_0, \ldots, F_{k-1}$ is an $(1+\epsilon)$-approximation of the optimal value on G. We will use two lemmas and some notation for the proof. For any tree T of G, let $w_{max}(T)$ denote the maximum weight of the edges of T, $w_{max}(T) = \max_{e \in E(T)} w(e)$. Let $g : V \to \mathbb{R}$ be defined as $g(v) = \min_{T \text{ tree}, |V(T)|=p, v \in V(T)} w_{max}(T)$.

Lemma 2. *Let F be a spanning forest of G, and $V' = \{v_1, \ldots, v_r\}$ a set of vertices such that any tree T of F which is not of order at least p contains a vertex $v_i \in V'$. Then we can construct a forest of G in which every tree component has order at least p and the total weight is at most $w(F) + \sum_{v \in V'} g(v)$.*

Lemma 3. $\sum_{v \in V} g(v) \leq 2(p-1)\, opt(G)$.

Now we are in a position to complete the proof of the theorem. Let $V_i = \bigcup_{t \equiv i \pmod{k}} (X_{t+1} \cup X_{t+k})$. By Lemma 2, starting from S_i we can construct a forest F_i satisfying the property that every tree component is as large as required, moreover $w(S_i) + \sum_{v \in V_i} g(v) \geq w(F_i)$. Since S_i is an optimal solution of a relaxed problem, we have $w(S_i) \leq opt(G)$ and so $opt(G) + \sum_{v \in V_i} g(v) \geq w(F_i)$.

Moreover, using Lemma 3 we obtain the following inequality $\sum_{1\leq i\leq k} g(V_i) = 2\sum_{v\in V} g(v) \leq 4(p-1)\,opt(G)$. Hence, there exists an i_0 such that $\sum_{v\in V_{i_0}} g(v) \leq \frac{4(p-1)}{k}\,opt(G)$, which implies $\min_{1\leq i\leq k} w(F_i) \leq w(F_{i_0}) \leq (1+\frac{4(p-1)}{k})\,opt(G) \leq (1+\epsilon)\,opt(G)$. The overall running time of the algorithm is k times what we need for graphs of treewidth at most k, that is $O((\frac{4p}{\varepsilon})^{dp/\varepsilon}n)$, where d is a constant. Hence, since $p = O(\frac{\log n}{\log\log n})$ we have a ptas. □

Remark 4. We can improve the time of the ptas given in Theorem 4 using sphere cut decomposition instead of bounded treewidth decomposition. In this way we obtain a ptas whose running time is $O((4p)^{8p/\varepsilon}n)$

References

1. B. S. Baker. Approximation algorithms for *NP*-complete problems on planar graphs. *Journal of the Association for Computing Machinery*, 41(1):153–180, 1994.
2. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.
3. M. Chlebik and J. Clebikova. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science*, 354:320–338, 2006.
4. B. Courcelle. The monadic second-order logic of graphs. III. Tree-decompositions, minors and complexity issues. *RAIRO Informatique Théorique Appliquée*, 26:257-286, 1992.
5. M. E. Dyer and A. M. Frieze. Planar *3DM* is *NP*-complete. *Journal of Algorithms*, 7:174–184, 1986.
6. F. Dorn, E. Penninkx, H. L. Bodlaender, and F. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions. *Technical Report UU-CS-2006-006.*
7. J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5:88–124, 1973.
8. T. Gallai. Über extreme Punkt- und Kantenmengen. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Mathematica*, 2:133–138, 1959.
9. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal of Computing*, 24:296–317, 1995.
10. C. Imielinska, B. Kalantari and L. Khachiyan. A greedy heristic for a minmum-weight forest problem. *Operations Research Letters*, 14:65–71, 1993.
11. T. Kloks. Treewidth. computations and approximations. *Lecture Notes in Computer Science*, 842, 1994.
12. M. Laszlo and S. Mukherjee. Another greedy heuristic for the constrained forest problem. *Operations Research Letters*, 33:629–633, 2005.
13. M. Laszlo and S. Mukherjee. A class of heuristics for the constrained forest problem. *Discrete Applied Mathematics*, 154:6–14, 2006.
14. J. Monnot and S. Toulouse. The path partition problem and related problems in bipartite graphs. *Operations Research Letters*, 35:677–684, 2007.
15. A. Schrijver. *Combinatorial Optimization.* Springer, 2003.
16. Zs. Tuza. Extensions of Gallai's graph covering theorems for uniform hypergraphs. *Journal of Combinatorial Theory Series B*, 52:92–96, 1991.