Complexity and approximation of the Constrained Forest problem

Cristina Bazgan [1] Basile Couëtoux [1] Zsolt Tuza [2,*]

[1] Université Paris-Dauphine, LAMSADE
Place du Marechal de Lattre de Tassigny, 75775 Paris Cedex 16, France

[2] Computer and Automation Institute, Hungarian Academy of Sciences
H–1111 Budapest, Kende u. 13–17, Hungary
and
Department of Computer Science, University of Pannonia
H–8200 Veszprém, Egyetem u. 10, Hungary

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 [17, 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 [18].

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. Indeed, since every feasible solution of Min CF(p) 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. Conclusions are provided in the final section.

2 Preliminaries

We begin with some basic definitions, summarized in three 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.

Reductions. The notion of L-reduction was introduced by Papadimitriou and Yannakakis in [15]. Let A and B be two optimization problems. Then A is said to be *L-reducible* to B if there are two constants $\alpha, \beta > 0$ such that

1. there exists a function, computable in polynomial time, which transforms each instance x of A into an instance x' of B such that $opt_B(x') \leq \alpha \cdot opt_A(x)$,

2. there exists a function, computable in polynomial time, which transforms each solution y' of x' into a solution y of x such that $|v(x,y) - opt_A(x)| \leq \beta \cdot |v(x',y') - opt_B(x')|$.

For us the important property of this reduction is that if a maximization problem A is L-reducible to a minimization problem B and B is c-approximable then A is $\frac{1}{1-\alpha\beta(c-1)}$-approximable.

3 Complexity for $p \geq 3$

Min CF(3) was proved *NP*-hard even on planar bipartite graphs of maximum degree three [14]. In this section we prove that Min CF(3) is intractable for every $p \geq 4$ for the same restricted classe 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.

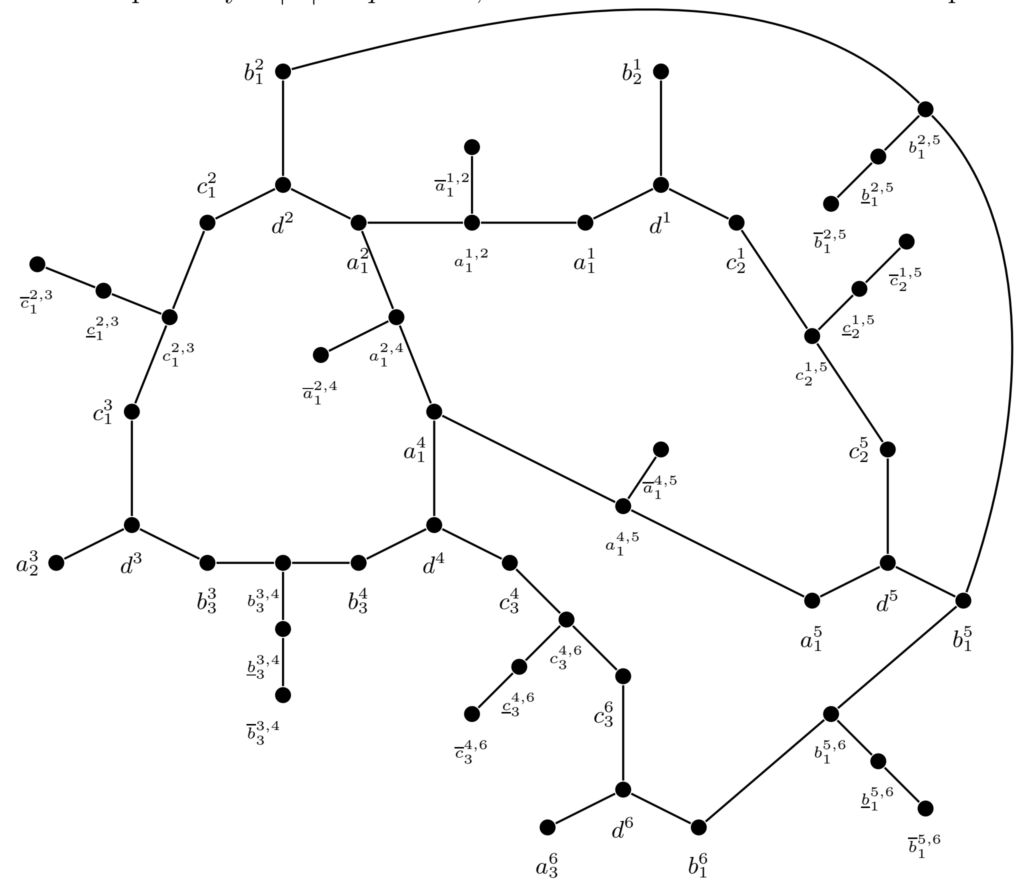

Figure 1: Graph $G(I)$ derived from instance $I = (A, B, C, \mathcal{T})$ of 3DM with $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $C = \{c_1, c_2, c_3\}$, and $\mathcal{T} = \{T_1, \ldots, T_6\}$ with $T_1 = (a_1, b_2, c_2)$, $T_2 = (a_1, b_1, c_1)$, $T_3 = (a_2, b_3, c_1)$, $T_4 = (a_1, b_3, c_3)$,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

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 $f_\mathcal{M}(a_i) = f_\mathcal{M}(b_j) = f_\mathcal{M}(c_k) = \ell$, 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

$$\begin{aligned} F_{a_i} = & \bigcup_{\substack{r<f_\mathcal{M}(a_i)\\(a_i,r,s)\in J_A}} \{(d^r, a_i^r), (a_i^r, a_i^{r,s}), (a_i^{r,s}, \overline{a}_i^{r,s})\} \quad \cup \\ & \bigcup_{\substack{s>f_\mathcal{M}(a_i)\\(a_i,r,s)\in J_A}} \{(d^s, a_i^s), (a_i^s, a_i^{r,s}), (a_i^{r,s}, \overline{a}_i^{r,s})\}, \\ F_{b_j} = & \bigcup_{\substack{r<f_\mathcal{M}(b_j)\\(b_j,r,s)\in J_B}} \{(b_j^r, b_j^{r,s}), (b_j^{r,s}, \underline{b}_j^{r,s}), (\underline{b}_j^{r,s}, \overline{b}_j^{r,s})\} \quad \cup \\ & \bigcup_{\substack{s>f_\mathcal{M}(b_j)\\(b_j,r,s)\in J_B}} \{(b_j^s, b_j^{r,s}), (b_j^{r,s}, \underline{b}_j^{r,s}), (\underline{b}_j^{r,s}, \overline{b}_j^{r,s})\}. \end{aligned}$$

The set F_{c_k} is defined like F_{b_j}. Let $F_\mathcal{M} = \bigcup_{e\in A\cup B\cup C} F_e \cup F_T$. 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 where each connected component has at least 4 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 $p = 4$.

In order to prove the theorem for $p > 4$, we also construct a reduction from 3DM to Min CF(p). Given an instance I of 3DM, the graph $G'(I)$ instance of Min CF(p) is obtained from the previous $G(I)$ by replacing the edges between a_i^ℓ and d^ℓ, $\underline{b}_j^{r,s}$ and $\overline{b}_j^{r,s}$, and $\underline{c}_j^{r,s}$ and $\overline{c}_j^{r,s}$ by a path of length $p - 3$ (see Figure 2). More precisely, we add to $G(I)$ the following vertices and edges:

$$V_T(p) = \bigcup_{a_i^\ell \in V_T} \{a_i^\ell(1), \ldots, a_i^\ell(p-4)\}$$

$$E_T(p) = \bigcup_{a_i^\ell \in V_T} \{(a_i^\ell, a_i^\ell(1)), (a_i^\ell(1), a_i^\ell(2)), \ldots, (a_i^\ell(p-4), d^\ell)\}$$

$$V_B(p) = \bigcup_{b_j^{r,s} \in V_B} \{b_j^{r,s}(1), \ldots, b_j^{r,s}(p-4)\}$$

$$E_B(p) = \bigcup_{b_j^{r,s} \in V_A} \{(\underline{b}_j^{r,s}, b_j^{r,s}(1)), (b_j^{r,s}(1), b_j^{r,s}(2)), \ldots, (b_j^{r,s}(p-4), \overline{b}_j^{r,s})\}$$

The sets $V_C(p)$ and $E_C(p)$ are defined analogously.

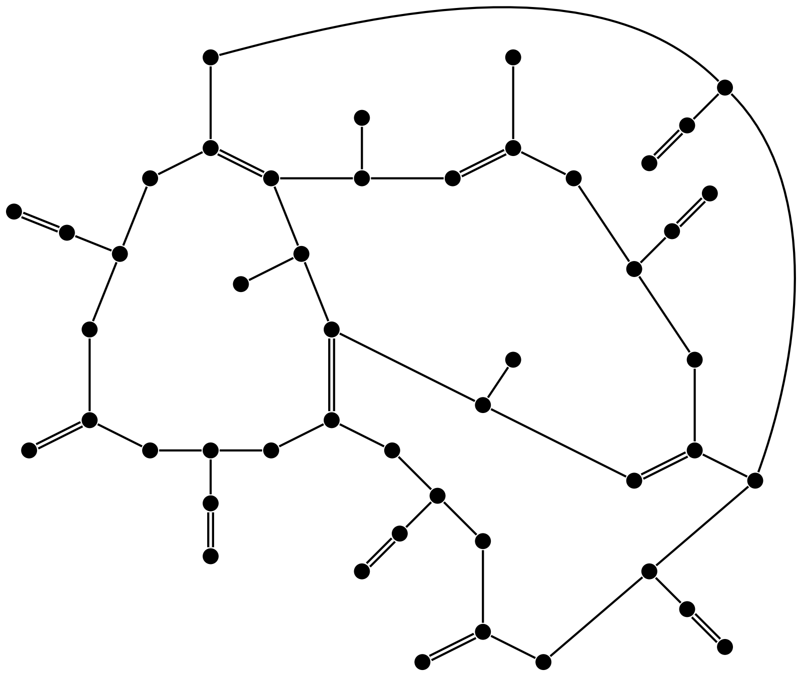

Figure 2: Graph $G'(I)$ derived from instance I described in Figure 1. Double lines represent paths of length $p-3$.

We show next that I contains a perfect matching if and only if $G'(I)$ contains a forest of weight at most $(p-1)(3|\mathcal{T}|-2q)$ where every tree is of order at least p. Given a perfect matching $\mathcal{M}$ on the instance I, in addition to the edges chosen for such a solution in $G(I)$ for $p=4$, we take every edge in $E_B(p), E_C(p)$ and $E_T(p)$. Thus we obtain a forest with the same number of trees $3|T|-2q$ and all these trees are of order at least p. Conversely, given a forest F of weight $(p-1)(3|\mathcal{T}|-2q)$ on the instance $G'(I)$, we can consider the trees which are not covering vertices of linking gadgets. Such a collection of trees describes a perfect matching of size q on I, which is therefore optimal. □

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.*

Proof. We first give a detailed argument for $p = 4$. For this, we construct an L-reduction from Max 3DM$_2$ to Min CF(4). For any instance I of Max 3DM$_2$ we construct $G(I)$ as in Theorem 1. Since any element of A, B, C appears exactly twice in T, for any a_i there exist only one vertex of the form $a_i^{r,s}$ and one of $\overline{a}_i^{r,s}$; locally in this proof we rename them as a_i and $\overline{a_i}$. We shall use $b_j, \underline{b}_j, \overline{b}_j, c_k, \underline{c}_k, \overline{c}_k$ in an analogous meaning. Note that the number of vertices in $G(I)$ is exactly $16q$.

Let $\mathcal{M}$ be any solution, say of size t, on I ; that is, $\mathcal{M}$ is a matching but not necessarily a perfect matching. From $\mathcal{M}$ we construct a forest F of $G(I)$. Let

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

$$F_R = \bigcup_{x_\ell = (a_i, b_j, c_k) \notin \mathcal{M}} \{(a_i^\ell, d^\ell), (a_i^\ell, a_i), (b_j^\ell, b_j), (c_k^\ell, c_k)\}$$

$$F_A = \bigcup_{a_i \in A} \{(a_i, \overline{a}_i)\}, \quad F_B = \bigcup_{b_j \in B} \{(b_j, \underline{b}_j), (\underline{b}_j, \overline{b}_j)\}, \quad F_C = \bigcup_{c_k \in C} \{(c_k, \underline{c}_k), (\underline{c}_k, \overline{c}_k)\}.$$

Then, let $F = F_T \cup F_R \cup F_A \cup F_B \cup F_C$. We note that the trees occurring in F are of orders at least 4. In fact, F contains t trees on the star gadgets corresponding to $\mathcal{M}$, and one tree of order four on each linking gadget. Therefore F contains $t + 3q$ trees, and then the weight of F is $13q - t$. Thus, $opt(G(I)) \leq 13q - opt(I)$.

Let F be a forest of weight v, covering $G(I)$ with trees of orders at least 4. We consider, as solution $\mathcal{M}$ for I, the triplets corresponding to those tree components of F which have order 4 and so cover star gadgets. Since F contains $16q - v$ trees and at most $3q$ of these trees cover linking gadgets, we obtain that $opt(I) \geq |\mathcal{M}| \geq 13q - v$. Thus, $v(\mathcal{M}, I) \geq 9q - v(F, G(I))$ and $opt(G(I)) = 13q - opt(I)$.

Since I contains $2q$ triplets and each of them shares an element with at most three triplets, we have $opt(I) \geq \frac{q}{2}$ (a lower bound valid for every inclusion-wise maximal solution), and consequently $opt(G(I)) \leq 25 opt(I)$. Moreover, by the observations above, we also obtain $opt(I) + opt(G(I)) = 13q \leq v(F, G(I)) + v(\mathcal{M}, I)$ and so $opt(I) - v(\mathcal{M}, I) \leq v(F, G(I)) - opt(G(I))$.

It is NP-hard to achieve an approximation factor of $\frac{95}{94}$ for Max 3DM$_2$ [3], and consequently Min CF(4) is not $\frac{2376}{2375}$-approximable if $P \neq NP$.

The proof for $p > 4$ applies the same modifications as it has been done in Theorem 1 for NP-hardness, replacing some edges by paths of length $p - 3$.

For $p = 3$, we essentially replace the star gadget by a path of length 3 $(b_j^\ell, a_i^\ell, c_k^\ell)$, by removing the vertex d^ℓ and we also remove the vertices $\overline{b}_j, \overline{c}_j$ (see an illustration in Figure 3 with a particular instance I of Max 3DM$_2$). □

4 Efficient algorithms

In this section we design efficient algorithms for some restricted classes of graphs.

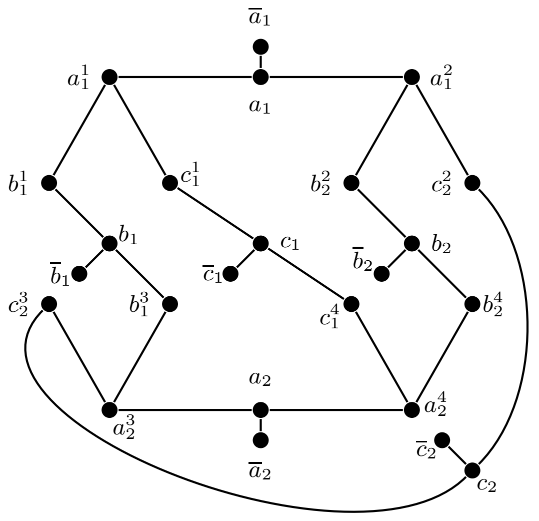

Figure 3: Graph obtained from instance $I = (A, B, C, \mathcal{T})$ of Max 3DM$_2$ with $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, $C = \{c_1, c_2\}$, and $\mathcal{T} = \{T_1, \ldots, T_4\}$ with $T_1 = (a_1, b_1, c_1)$, $T_2 = (a_1, b_2, c_2)$, $T_3 = (a_2, b_1, c_2)$, $T_4 = (a_2, b_2, c_1)$.

4.1 Exact algorithms for bounded treewidth graphs

In the following 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 4. The time bound is polynomial in n whenever $k(\log k + \log p) = O(\log n)$ and in particular if $k = O\left(\frac{\log n}{\log \log n}\right)$ 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 5. *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.*

Proof. One has to consider each type of nodes separately.

If x is a *start* node, then the entire set of values $f(x, F, \mathcal{P}, I)$ can be determined for x in $O(k)$ time. Indeed, in this case $G(x)$ has at most $k+1$ vertices, the classes of $\mathcal{P}$ are exactly the vertex sets of the connected components of F which can be found in $O(k)$ time, and $i_r = \min\{|C_r|, p\}$ holds for every class C_r of $\mathcal{P}$.

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, where B_k is the kth Bell number. 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.

If x is an *introduce* node, let E_v be the set of edges incident to v in F. Let $F_v = F - v$ be the forest without the vertex v; the removal of v may have divided a tree component of F into several components. We consider all possible partitions $\mathcal{P}_v$ over the elements of X_1, such that every class of $\mathcal{P}$ which does not contain v is a class in $\mathcal{P}_v$, too; and the class C_s

which contains v is divided into q classes where q is the number of components created by the removal of v and such that every remaining component is in a different class. The i-values associated with those q classes must sum up to $i_s - 1$. Then

$$f(x, F, \mathcal{P}, I) = \min_{\mathcal{P}_v, I_v} f(x_1, F_v, \mathcal{P}_v, I_v) + w(E_v).$$

This step needs at most $B_k \cdot O(p^{k+1})$ time.

If x is a *forget* node, let F_v be obtained through the addition of v to F such that all components of F_v which are adjacent to v in F belong to the same class of $\mathcal{P}$. Assume further that $\mathcal{P}_v$ has the same classes as $\mathcal{P}$ except that we add v to the class which contains the vertices it is connected to in F_v; if there is none, then we add the class $\{v\}$ to $\mathcal{P}$ as a singleton. Now I_v is either I if v is not alone in its class in $\mathcal{P}_v$, or $I \cup \{p\}$. Then

$$f(x, F, \mathcal{P}, I) = \min_{F_v} f(x_1, F_v, \mathcal{P}_v, I_v).$$

This needs at most $B_k \cdot O(p^{k+1})$ time. □

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 6. *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 $p_i \le p$, 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

In the following we propose ptas on the class of planar graphs, using the polynomial time algorithm for graphs of bounded treewidth.

Theorem 7. *For $p = O\left(\frac{\log n}{\log \log n}\right)$, 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 $k = \left\lceil \frac{4(p-1)}{\epsilon} \right\rceil$. Let X_t be the set of vertices of level t and let H_i, $0 \le i \le k - 1$, 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 $\bigcup_{t+1 \le j \le t+k} X_j$, 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 $\bigcup_{t+1 \le j \le t+k} X_j$ is k-outerplanar, and so is H_i, too.

Since H_i is k-outerplanar, it has treewidth at most $3k-1$ [2]. We construct graph H_i' from H_i by attaching $p-1$ pendant edges to each vertex on the boundary (that means vertices in X_{t+1}, X_{t+k} with $t \equiv i \pmod{k}$), and put weights 0 on these edges. On applying Theorem 3, we can efficiently determine an optimal forest S_i on H_i' which is a solution on H_i such that each vertex is either in a tree that contains a boundary vertex or in a tree of size at least p. We complete this solution 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 $F_0, \dots, F_{k-1}$ is an $(1+\epsilon)$-approximation of the optimal value on G. We will use two lemmas and some notation for the proof. For any subset of edges that forms a 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 8. *Let F be a spanning forest of G, and $V' = \{v_1, \dots, v_r\}$ 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 $w(F) + \sum_{v \in V'} g(v)$.*

Proof. For short, let us call a tree component small if it has fewer than p vertices, and call it large otherwise. By induction we suppose that the assertion is true when we have at most r small trees in the forest (the case $r = 0$ is trivial). Let F be a forest with $r+1$ small components, such that some vertices $v_1, \dots, v_{r+1}$ belong to distinct small trees; and let T_{r+1} be the small tree in F containing v_{r+1}. Let A_{r+1} be the tree which realizes the optimum of $g(v_{r+1})$. Since T_{r+1} is small and A_{r+1} is large, and v_{r+1} belongs to both trees, there exists an edge of A_{r+1} which has exactly one endpoint in T_{r+1}. We add this edge to forest F in order to obtain F'. Forest F' is of weight at most $w(F) + g(v_{r+1})$ since the added edge belongs to A_{r+1}. Thus, in F', tree T_{r+1} is linked to another tree which implies that the number of small trees has decreased at least by one (either by joining two small trees to become just one component of any size, or by joining T_{r+1} to a large component). Moreover $v_1, \dots, v_r$ are sufficient to cover the remaining small trees. Therefore we can construct a covering forest in which each tree is of order at least p, and which has weight at most

$$\begin{aligned} w(F') + \sum_{v \in V' \setminus \{v_r+1\}} g(v) &\le w(F) + g(v_{r+1}) + \sum_{v \in V' \setminus \{v_r+1\}} g(v) \\ &= w(F) + \sum_{v \in V'} g(v) \end{aligned}$$

□

Lemma 9. $\sum_{v\in V} g(v) \leq 2(p-1)\, opt(G)$.

Proof. Suppose, for simplicity, that all weights are distinct. (This can be achieved by a slight modification of the weights, by adding $\varepsilon, 2\varepsilon, 3\varepsilon, \dots$ to them with a very small ε such that the increase of sum on each tree of order p is smaller than the smallest positive edge-weight difference.)

Let F^* be an optimal solution on G. For $v \in V$, let $T^*(v) \subset F^*$ be a subtree of order exactly p which contains v and minimizes the largest weight of its edges. Let $e^*(v)$ be the edge of maximum weight in $T^*(v)$. Remark that $w(e^*(v)) \geq g(v)$. For any edge $e \in F^*$, let

$$T(e) = \bigcup_{e^*(v)=e} T^*(v).$$

By definition, $T(e)$ is a tree since it is included in F^*. Moreover, $w_{max}(T(e)) = w(e)$ if $T(e)$ is not empty. If $|V(T(e))| \geq 2p-1$ then, if we remove e from $T(e)$, we obtain two trees and at least one of them, say T', is large. Therefore T' is in F^* and $w_{max}(T') < w_{max}(T(e))$ which is a contradiction with $e^*(v) = e$ for any vertex in T'. Thus $|(e^*)^{-1}(e)| \leq 2(p-1)$.

Consequently,

$$\begin{aligned}\sum_{v\in V} g(v) &\leq \sum_{v\in V} w(e^*(v)) \\ &= \sum_{e\in F^*} |(e^*)^{-1}(e)| w(e) \\ &\leq 2(p-1) \sum_{e\in F^*} w(e) \\ &= 2(p-1) opt(G)\end{aligned}$$

□

Now we are in a position to complete the proof the theorem. Let $V_i = \bigcup_{t\equiv i \pmod k}(X_{t+1} \cup X_{t+k})$. By Lemma 8, 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 9 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

$$\min_{1\leq i\leq k} w(F_i) \leq w(F_{i_0}) \leq \left(1 + \frac{4(p-1)}{k}\right) opt(G) \leq (1+\epsilon)\, opt(G).$$

The overall running time of the algorithm is k times what we need for graphs of treewidth at most k, that is $O(n(\frac{4p}{\varepsilon})^{dp/\varepsilon})$, where d is a constant. Hence, since $p = O\left(\frac{\log n}{\log\log n}\right)$ we have a ptas. □

4.3 Speeding up the ptas for planar graphs

Planar graphs of bounded treewidth allow to organize dynamic programming more efficiently than the method described in Section 4.1. In the following result we give such a speed-up in the running time, although it does not improve the $O\left(\frac{\log n}{\log \log n}\right)$ bound for p from Theorem 7.

Theorem 10. *On planar graphs, for any $\epsilon > 0$, there exists an algorithm that returns a $(1+\epsilon)$-approximation in $O(\frac{1}{\varepsilon} n p^4 (16p^3)^{4p/\varepsilon})$ time.*

Proof. It is known [16] that for any planar graph G with treewidth at most k there exists a branch decomposition of width at most k. A *branch decomposition* of G is a couple (T, μ) such that T is a tree with all internal nodes of degree 3 and μ is a bijection from the edges of G to the leaves of T. For any edge e of T, the removal of e creates two trees $T_1(e)$ and $T_2(e)$. Let $(E_1(e), E_2(e))$ be the partition of the edge set of G naturally defined by μ in the way that $e' \in E_i(e)$ if and only if $\mu(e')$ is a leaf of $T_i(e)$, $i \in \{1, 2\}$. Denote by $mid(e)$ the set of vertices of G which are incident with an edge from $E_1(e)$ and also from $E_2(e)$. The maximum size of $mid(e)$ over all $e \in E(T)$ is called the *width* of the branch decomposition.

Given a planar embedding of G, the result of Dorn *et. al.* [6] enables us to obtain a sphere-cut decomposition of width at most k from this branch decomposition. A *sphere-cut decomposition* is a branch decomposition (T, μ) such that for any edge $e \in E(T)$ there exists a closed Jordan curve (not intersecting itself), called *noose*, which passes through all vertices of $mid(e)$ and does not cross any edge of G, moreover in the planar embedding of G every edge of $E_1(e)$ is in the interior of the noose and every edge of $E_2(e)$ is in the exterior of the noose. We view the noose as a virtual cycle on $mid(e)$, whose edges are embedded inside the faces which contain any two consecutive members of $mid(e)$ on their boundaries.

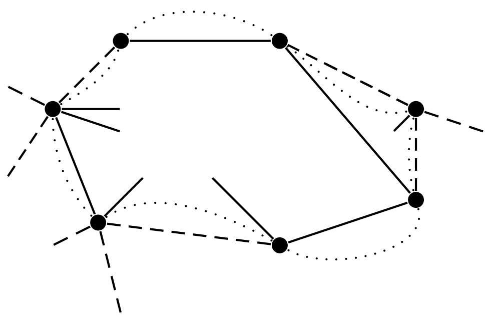

Figure 4: The straight lines and the dashed lines are the edges of $E_1(e)$ and $E_2(e)$, respectively; the dotted closed curve indicates the virtual cycle of the noose

Consider any internal node in the tree of the sphere-cut decomposition. It has degree 3, and the cycles associated to its incident edges will look as shown in Figure 5.

For any node x of T, denote $C(x) = mid(e_1) \cup mid(e_2) \cup mid(e_3)$ where e_1, e_2, e_3 are the three incident edges at x in T. The tree decomposition will be built on the same tree T what we are using for the sphere-cut decomposition. The sets associated with the nodes of the tree decomposition are defined as follows:

- to any internal node x of T we associate the set of vertices of $C(x)$,
- to any leaf x of T we associate the endpoints of the edge $\mu^{-1}(x)$.

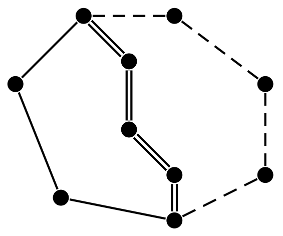

Figure 5: The structure for a node of T, composed of three nooses: dashed and straight lines form the virtual cycle of the father edge, dashed and double lines for one child edge, and straight and double lines for the other child edge

If two vertices u, v of G are adjacent, they appear together in the set associated with the leaf node $\mu(u, v)$. Moreover, the occurrences of any $v \in V(G)$ in the sets associated with the nodes of T form a subtree in T, because if $v \in \mathit{mid}(e)$ then v appears in the sets at both ends of e, thus the set of occurrences of v is precisely the subtree of T whose set of leaves is $\{\mu(v, v') \mid (v, v') \in E(G)\}$. Moreover, each $C(x)$ is composed of three cycles of length at most k each, and each virtual edge of those cycles occurs exactly twice in the union, as indicated in Figure 5. Consequently, each $C(x)$ contains fewer than $3k/2$ vertices. Therefore a tree decomposition of G is obtained with width less than $3k/2$.

Consider now any virtual cycle $C(x)$ associated with a noose. While determining the table for the corresponding node of T in the dynamic programming computation, we need to consider each partition on $C(x)$ which extends the partial partitions having been determined in the interior and exterior of $C(x)$. Consider one from the exterior, for instance. An extended partition is obtained by inserting a forest in the interior of $C(x)$. If the inserted edges cross inside $C(x)$, however, then due to planarity it implies that the two connected partition classes intersect in the exterior, hence could not be parts of separate extended classes. Consequently we only need to consider spanning forests on $C(x)$ with non-crossing edges. The number of such forests on k nodes is expressed by the *Catalan number* $C_k = \frac{1}{k+1}\binom{2k}{k}$, which is substantially smaller than the Bell number B_k.

All rows in the table for a node are determined by the combinations of the table(s) of its child(ren). Hence combining the partial partitions of the interior and exterior, we need to consider no more than $(C_{k+1})^2$ cases for each pair of non-crossing subforests. Therefore the minimum constrained forest problem on a planar graph with treewidth less than k can be solved in $O(4^{2k} \times p^{3k+3})$ time. This improvement speeds up our previous algorithm to obtain complexity $O(\frac{1}{\varepsilon} np^4 (16p^3)^{4p/\varepsilon})$. $\square$

5 Conclusion

In this paper we considered a natural generalization, Min WCF(p), of the fundamental problem of minimum-weight edge cover in graphs. The task is to find, for an edge-weighted input graph, a spanning forest in which every tree component has at least p vertices and the sum of weights is as small as possible.

On the negative side, up to now Min CF(3) and Min WCF(p) for $p \geq 4$ were proved *NP*-hard. We extended this, by proving that even the unweighted Min CF(p) is *APX*-hard for all $p \geq 3$. This result remains valid for restricted classes of graphs (e.g., bipartite graphs of maximum degree 3).

We also considered the positive side to some extent, by designing an exact algorithm for graph classes where treewidth may slowly tend to infinity, and efficient approximations for planar graphs where p may slowly grow with the number of vertices. The latter results are of interest since Min CF(p) is *NP*-hard also on planar instances with $p = 3$ and assuming maximum degree 3.

Perhaps the most challenging problem that remains open is to design a $(2-c)$-approximation for Min WCF(p) for some constant $c > 0$ and $p \geq 3$. For the unweighted case, however, it is easy to get a $\left(1 + \frac{1}{p-1}\right)$-approximation in linear time.

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 optimizationproblems. *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 Branch Decompositions. *Proceedings of the 13th Annual European Symposium (ESA 2005), LNCS 3669*, 95–106.

[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] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Science*, 43:425–440, 1991.

[16] N. Robertson and P. Seymour. Graph minors X. Obstructions to treedecomposition. *Journal of Combinatorial Theory Series B*, 52:153–190, 1991.

[17] A. Schrijver. *Combinatorial Optimization.* Springer, 2003.

[18] Zs. Tuza. Extensions of Gallai's graph covering theorems for uniform hypergraphs. *Journal of Combinatorial Theory Series B*, 52:92–96, 1991.