Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®
Discrete Applied Mathematics 134 (2004) 51–65
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

On the complexity of graph tree partition problems

Roberto Cordone[a,*], Francesco Maffioli[b]

[a] *Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano, Via Bramante 65, I-20613 Crema, Italy*

[b] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy*

Received 12 December 2000; received in revised form 1 March 2002; accepted 18 March 2002

Abstract

This paper concerns the optimal partition of a graph into p connected clusters of vertices, with various constraints on their topology and weight. We consider different objectives, depending on the cost of the trees spanning the clusters. This rich family of problems mainly applies to telecommunication network design, but it can be useful in other fields. We achieve a complete characterization of its computational complexity, previously studied only for special cases: a polynomial algorithm based on a new matroid solves the easy cases; the others are strongly $\mathcal{NP}$-hard by direct reduction from *SAT*. Finally, we give results on special graphs.

Keywords: Tree partition; Computational complexity; Greedy algorithm

1. Introduction

The general *Graph Tree Partition Problem* (*GTPP*) may be defined as follows. Let $G(V,E)$ be an undirected graph, consisting of $n=|V|$ vertices and $m=|E|$ edges. A cost function $c:E\to\mathbb{N}$ is defined on the edges of the graph. Without any loss of generality, we assume G to be connected; otherwise, fictitious edges of suitable cost can be added to it. The aim of the problem is to partition the vertex set V into a given number p of suitable disjoint *clusters* U_r, and to build on each of them a spanning

* Corresponding author.
E-mail addresses: roberto.cordone@unimi.it (R. Cordone), maffioli@elet.polimi.it (F. Maffioli).

0166-218X/03/$ - see front matter
doi:10.1016/S0166-218X(03)00340-8

tree T_r. Notice that the clusters U_r are never empty, but they can reduce to a single element: an isolated vertex is considered as a tree with an empty edge set. The feasible solutions $F(V,X)$ are all the spanning forests which consist of p trees $T_r(U_r,X_r)$ and satisfy suitable side constraints. We take into account three families of such constraints:

1. *Root constraints*: given a collection $\mathscr{R}$ of p *root sets* $R_r \subseteq V$, each tree T_r must contain at least one of the vertices in the corresponding root set R_r,

$$U_r \cap R_r \neq \emptyset \quad (r = 1, \ldots, p). \tag{1}$$

2. *Inclusion constraints*: given a collection $\mathscr{B}$ of p *boundary sets* $B_r \subseteq V$, each tree T_r must be completely included into the corresponding boundary set B_r,

$$U_r \subseteq B_r \quad (r = 1, \ldots, p). \tag{2}$$

   Each B_r should induce a connected subgraph on G, lest the problem is unfeasible.
3. *Weight constraints*: given a weight function $w : V \to \mathbb{N}$ defined on the vertices of the graph, the total weight of each tree $w(U_r) = \sum_{v \in U_r} w_v$ must belong to a given interval $[W_r^-; W_r^+]$,

$$W_r^- \leqslant w(U_r) \leqslant W_r^+ \quad (r = 1, \ldots, p). \tag{3}$$

Roughly speaking, the weight constraints limit the size of the trees, while the root and inclusion constraints limit their spatial distribution in the graph. In particular, the root constraints require the trees to touch given regions, the inclusion constraints forbid it.

As for the objective functions, we are concerned with those which depend on the costs of the single trees $c(X_r) = \sum_{e \in X_r} c_e$, since they take into account the structure of the solutions. In particular, we focus on four classical cases: *min–sum* and *max–sum* problems (respectively, minimize and maximize the total cost of the forest), *min–max* problems (minimize the cost of the most expensive tree) and *max–min* problems (maximize the cost of the cheapest tree).

The *GTPP*, though a general model, is a fairly good approximation of practical cases in various application fields. This is especially true for local telecommunication networks (servers, cable TV companies, and so on) where connection costs, the need for a balance between subnetworks and requirements on their location must all be taken into account. Electric or radio broadcasting networks present similar cost-balance trade-offs, while electoral districting concerns the partition of a given geographical area into connected and balanced regions centred on a chief town. In the end, various classical algorithms for Cluster Analysis provide solutions with a forest structure (see, for instance, the *single-linkage algorithm* [8]).

In this paper, we completely characterize the computational complexity of the *GTPP*s. Of course, if no side constraint exists, one obtains the *Minimum* (or *Maximum*) *Spanning Forest Problem*, which is solved exactly by the Greedy Algorithm [13]. We generalize this algorithm to the root constrained case by refining its *independence test* in order to take into account the additional constraints. This provides a matroid which, to our knowledge, is unknown in the literature. By contrast, the inclusion constrained, the

weight constrained and all min–max and max–min problems, however constrained, are strongly $\mathcal{NP}$-hard. We prove this by reduction from the Satisfiability Problem (SAT).

The paper is organized as follows. First, we review the literature on related subjects. Section 3 summarizes the new results on the computational complexity of the *GTPP*. The following sections discuss them in detail: in particular, Section 4 deals with the easy cases, Section 5 with the hard ones and with some remarks on their approximability. Section 6, in the end, tackles some special cases, namely bipartite graphs and grid graphs.

2. Previously known facts

The literature has considered a number of problems dealing with trees and forests, which have more or less tight relations to the *GTPP*. Most of the time, they refer to weight constraints. To our knowledge, inclusion constraints have never been taken into account, and root constraints usually reduce to fixing a root for each tree in the forest ($|R_r| = 1$ for all r). This is rather surprising, if one considers their simple and natural meaning, both in physical applications and Cluster Analysis.

First, we mention some special cases of the *GTPP*. Guttmann-Beck and Hassin propose approximation algorithms for the min–sum and the min–max problem with very specific weight constraints, as well as a specific weight function: $w_v = 1$ for all v in V, and $W_r^- = W_r^+ = n/p$ for $r = 1, \ldots, p$, that is all of the trees should have the same number of vertices [9,10]. These algorithms come within a factor of $(2p - 1)$ to the optimum, and their computational complexity is $O(n^2)$. The approximation algorithm for the min–sum problem can be extended to deal with unequally sized trees, keeping the $(2p - 1)$ approximation ratio, but its complexity becomes exponential in p. The method cannot be applied when the triangle inequality does not hold. In this case, however, the problem is no longer approximable (see also Corollary 6).

Yamada et al. describe an exact algorithm for the min–max *GTPP* when the root sets are singletons [17]. This is a depth-first branch-and-bound method, branching on the edge variables: the upper bound derives from a heuristic based on the exchange of subtrees among trees, and the reevaluation of the minimum tree spanning each cluster [16].

Ali and Hwang tackle the min–sum *GTPP* with special weight constraints: $w_v = 1$ for all v in V, $W_r^- = \lfloor n/p \rfloor$ and $W_r^+ = \lceil n/p \rceil$ for $r = 1, \ldots, p$ [2]. So the trees must all have approximately the same number of vertices, one more one less. The authors formulate the problem by *clique inequalities*, to impose the cardinality upper bound, and *augmented clique inequalities*, to impose the lower bound. They relax the inequalities in a lagrangean fashion and update the multipliers by dual ascent.

Other problems differ from the *GTPP* in some fundamental feature. For instance, Imielińska et al. consider a sort of min–sum *GTPP* with a uniform weight constraint from below on all trees, but they do not fix the number of trees in the partition [12]. They prove that this problem is $\mathcal{NP}$-hard by reduction from the three-dimensional matching problem and propose a simple greedy heuristic with an approximation ratio of 2.

Goemans and Williamson introduce a very general approximation algorithm to solve min–sum spanning forest problems with various side constraints [7]. The algorithm runs in $O(n^2 \log n)$ time and comes within a factor of $2(1 - 1/n)$ from the optimum. It is a greedy algorithm, employing auxiliary costs on the edges and updating them step by step. The performance guarantee is based on the simultaneous construction of a primal and a dual heuristic solution. Since these differ at most by a factor of 2, the same factor applies between the primal approximate solution and the optimum. If the triangle inequality holds, the algorithm can be adapted to the *GTPP* with unit weights ($w_v = 1$ for all v in V) and identical uniform lower and upper bounds on the weights of the trees ($W_r^- = W_r^+ = n/p$), that is to the problem in which the cardinality of the trees is fixed. The performance guarantee is $4(1 - p/n)(1 - 1/n)$.

In the end, the *Capacitated Minimum Spanning Tree* problem (*CMST*), which has a wide variety of applications in the design of teleprocessing networks, deals with a rooted spanning tree whose subtrees have a weight not exceeding a uniform upper bound W. The number of the subtrees is not specified. This problem is strongly $\mathcal{NP}$-hard [14], but it admits approximate algorithms running in $O(n^2)$ time: if $w_v=1$ for all v in V, the guaranteed performance ratio is $(3 - 2/W)$; otherwise, it is 4 [3].

3. Summary of complexity results

In this section, we discuss the computational complexity of all *GTPP*s. It is easy to prove that their recognition version is always in $\mathcal{NP}$. Table 1 summarizes the new results while the known ones are given as notes. As for the min–sum and max–sum objective:

- The unconstrained problem is solved by the Greedy Algorithm, as it is the *Minimum* (*Maximum*) *Spanning Forest* problem [13].
- The root constrained problem can be solved by the Greedy Algorithm in the special case in which all root sets are singletons ($R_r = \{v_r\}$ for $r = 1, \ldots, p$), by adding a dummy vertex to the graph and connecting all roots to it with very low (or high) cost edges. We prove that the problem is always easy by generalizing the Greedy Algorithm (Corollary 3).
- The weight constrained problems are known to be strongly $\mathcal{NP}$-hard [10] when the weight function is uniform ($w_v = 1$ for all v in V) and the lower weight bound on each tree is equal to the corresponding upper bound ($W_r^- = W_r^+$ for $r = 1, \ldots, p$), that is to say when the cardinality of each tree is fixed [16]. We prove that the problem is strongly $\mathcal{NP}$-hard in general by reduction from the Satisfiability Problem (*SAT*) (Theorem 5). We will also prove that this holds in special graphs (Propositions 7 and 9).
- We prove that all inclusion constrained problems are strongly $\mathcal{NP}$-hard, even on special graphs, by reduction from *SAT* (Theorem 8) and the Steiner Tree Problem (Theorem 10).

Table 1
The computational complexity of the *GTPP*s

Side constraints	Min–sum Max–sum	Min–max	Max–min
None	$\mathscr{P}$[a]	Strongly $\mathscr{NP}$-hard	Strongly $\mathscr{NP}$-hard
Root constraints	$\mathscr{P}$[b]	Strongly $\mathscr{NP}$-hard[c]	Strongly $\mathscr{NP}$-hard
Inclusion constraints	Strongly $\mathscr{NP}$-hard	Strongly $\mathscr{NP}$-hard	Strongly $\mathscr{NP}$-hard
Weight	Strongly $\mathscr{NP}$-hard[d]	Strongly $\mathscr{NP}$-hard[d]	Strongly $\mathscr{NP}$-hard

[a] *Minimum Spanning Forest* [13].
[b] In $\mathscr{P}$ for $|R_r| = 1$ (a special case of the *Minimum Spanning Forest*) [13].
[c] Simply $\mathscr{NP}$-hard for $|R_r| = 1$ [16].
[d] Strongly $\mathscr{NP}$-hard for $w_v = 1$ and $W^- = W^+ = W/p$ [9,10].

As for min–max problems:

- When all root sets are singletons, the problem is known to be $\mathscr{NP}$-hard [17]. We prove that it is strongly $\mathscr{NP}$-hard even if unconstrained.
- When all trees must have the same cardinality ($w_v = 1$ for all v in V and $W_r^- = W_r^+$ for $r = 1, \ldots, p$), the problem is strongly $\mathscr{NP}$-hard [9]. We prove that it is strongly $\mathscr{NP}$-hard in general, by reduction from *SAT* (Theorem 5), and $\mathscr{NP}$-hard in special graphs (Propositions 7 and 9).

As for max–min problems:

- We prove that these problems are all strongly $\mathscr{NP}$-hard, by reduction from *SAT* (Theorem 5), and $\mathscr{NP}$-hard in special graphs (Propositions 7 and 9).

4. The root-constrained problem

The min–sum *GTPP* with no side constraints reduces to the *Minimum Spanning Forest* problem. The Greedy Algorithm, terminated as soon as the current solution consists of p trees, solves it exactly.

We generalize this algorithm to the root-constrained problem. A *well-rooted forest* is a spanning forest with $\tilde{p} \geqslant p$ connected components (trees) $T_r(U_r, X_r)$ $(r = 1, \ldots, \tilde{p})$ such that $U_r \cap R_r \neq \emptyset$ for $r = 1, \ldots, p$. We denote the first p components as *rooted components*, since one of the vertices in the intersection $(U_r \cap R_r)$ can be considered as the root of the tree. The other components will be referred to as *unrooted components*. If a well-rooted forest consists of exactly p trees, it is a feasible solution to the *GTPP*. If it has more, we can obtain a feasible solution, provided the graph is connected,

by adding suitable edges to the forest, so as to link the unrooted components to the rooted ones.

We prove that the edge set E and the collection $\mathscr{X}$ of the edge sets of all well-rooted forests form a matroid, whose bases are the feasible solutions to the root-constrained *GTPP*. Thus, the Greedy Algorithm can be applied, as long as the independence test takes into account both the prohibition of cycles and the prohibition of bad root locations. One needs a suitable test for the second condition, which the Greedy Algorithm could apply step by step. This is easy, as it is a problem of matching trees to subsets on an auxiliary bipartite graph $\tilde{G}^X$. On one shore of the graph, p vertices represent the root sets R_r, on the other shore $q=n-|X|$ vertices represent the connected components U_s in the current spanning forest $F(V,X)$. The edges of graph $\tilde{G}^X$ connect the "root set" vertices and the "component" vertices such that the corresponding subsets intersect in graph G.

Theorem 1. *Given a connected undirected graph $G(V,E)$ and a collection $\mathscr{R}$ of root sets, let $\mathscr{X}$ be the collection of the edge sets of all well-rooted spanning forests $F(V,X)$. Either $\mathscr{X}$ is empty, or the system set $(E,\mathscr{X})$ is a matroid, where $\mathscr{X}$ is its family of independent sets.*

Proof. In order to be a matroid, the system set $(E,\mathscr{X})$ must satisfy the three following conditions [15]:

1. $\emptyset \subseteq \mathscr{X}$,
2. if $X \in \mathscr{X}$ and $Y \subset X$, then $Y \in \mathscr{X}$,
3. $\forall X, Y \in \mathscr{X}$ such that $|X| = |Y| + 1$, there exists $e \in X \setminus Y$ such that $Y \cup \{e\} \in \mathscr{X}$.

The first condition derives from the second one, if $\mathscr{X}$ is not empty. In fact, the empty set is a proper subset of any other set of edges X.

The second condition states that if X induces a well-rooted spanning forest, any of its subsets Y also does. Of course, if X induces no cycles, this holds also for Y. The auxiliary graphs $\tilde{G}^X$ and $\tilde{G}^Y$ have the same root set vertices R_r on one side, whereas each vertex U^X on the other side corresponds to one or more vertices U^Y linked to the same root set vertices as U^X. Therefore, any p-cardinality matching on $\tilde{G}^X$ determines a p-cardinality matching on $\tilde{G}^Y$.

The third condition requires that, given two well-rooted forests induced by X and Y, if X contains one more edge than Y, at least one of the edges in X can be added to Y, yielding a well-rooted forest. First of all, let us observe that $\tilde{G}^Y$ has one more component vertex than $\tilde{G}^X$, and in particular one more unmatched component. Consider an unmatched component U^Y in $\tilde{G}^Y$: let U_i^X be the components of $\tilde{G}^X$ intersecting it (possibly, a single one), and let U^{XY} be their union. Of course, U^{XY} includes U^Y, so that there are three possible cases:

1. $U^Y \subset U^{XY}$,
2. $U^Y = U^{XY}$ and all components U_i^X are unmatched,
3. $U^Y = U^{XY}$ and at least one component U_i^X is matched.

In the first case, at least one component U_i^X includes vertices in U^Y and out of U^Y. So, at least one edge e in X has a single extreme in U^Y. Adding e to Y, U^Y merges with another component, and the original matching in $\tilde{G}^Y$ is untouched. Thus, $Y \cup \{e\} \in \mathscr{X}$. We now prove that at least one of the unmatched components U^Y falls under this case.

In the second case, in fact, we can remove U^Y from $\tilde{G}^Y$ and all components U_i^X from $\tilde{G}^X$: both of the matchings are untouched, and the number of unmatched components is still higher in $\tilde{G}^Y$. Of course, this cannot hold for all unmatched components U^Y.

Finally, in the third case, we modify the matching in $\tilde{G}^Y$. Suppose that U_i^X matches with R_r in $\tilde{G}^X$: then, we can match U^Y with R_r also in $\tilde{G}^Y$, generating a new unmatched component. As the number of unmatched components remains higher in $\tilde{G}^Y$, this cannot always hold. □

So, the Greedy Algorithm exactly solves the min–sum (respectively, the max–sum) root constrained problems. The following *GreedyMatching* algorithm describes its adaptation to this particular case. First, it checks whether the problem is unfeasible. If it is not, the algorithm sets the edges in non-decreasing (respectively, non increasing) cost order, and fixes them one by one, discarding those which generate cycles or bad root locations. Notice that in the unconstrained case the second test is always satisfied, and we get back to the Greedy Algorithm.

Algorithm 2. ***Greedy Matching*** $(G, c, p, \mathscr{R})$.

$X := \emptyset$;
Build the auxiliary graph $\tilde{G}^X$;
If $\tilde{G}^X$ *does not admit a p-cardinality matching* ***return*** *Unfeasible*;
Set the elements of E in non increasing cost order ($c_{e_{j-1}} \leqslant c_{e_j}$ *for* $j = 2, \ldots, |E|$);
$j := 1$;
While $|X| < n - p$ ***do***
  If $X \cup \{e_j\}$ *is an acyclic graph* ***and*** $\tilde{G}^{X \cup \{e_j\}}$ *admits a p-cardinality matching*
    then $X := X \cup \{e_j\}$; {*This also updates* $\tilde{G}^X$}
$j := j + 1$;
EndWhile;
Return X.

Corollary 3. *Algorithm* GreedyMatching *exactly solves the root constrained Graph Tree Partition Problem, or it proves that no solution exists.*

The complexity of *GreedyMatching* is $O(m(\log m + \alpha(m,n) + \gamma(m,n)))$ overall. The first term, $m \log m$, derives from the ordering initial phase. The second one from the test on the existence of cycles: function $\alpha(m,n)$ is defined as $\min\{i\colon A(i, m/n) \geqslant \log_2 n\}$ and it is almost constant, since the Ackermann function $A(i,j)$ increases with extremely high speed [1]. The dominating term is the last one, $\gamma(m,n)$, which is the complexity of solving a Maximum Bipartite Matching Problem, presently $O(m\sqrt{n})$ [11].

Actually, the matching need not be recomputed from scratch at each step: it can be updated.

Proposition 4. *The worst-case time complexity of Algorithm* GreedyMatching *is* $O(mnp)$.

Proof. Let a p-cardinality matching be determined on the auxiliary graph, and suppose to add a new edge to the forest in the original graph. If the edge links two unrooted components, or a rooted and an unrooted one, the current matching is still valid. If both of the components are rooted, on the contrary, two edges of the matching conflict. We remove one of them, obtaining a matching of cardinality $p-1$. It is well known that a larger matching exists if and only if an *augmenting path* can be found [4]. This is a path starting from an unmatched vertex, ending into another unmatched vertex, and made up of an odd number of edges, alternatively in and out of the matching. There is a single unmatched root set vertex from which an augmenting path could possibly start. A simple labelling algorithm permits to determine whether it exists, taking into account each edge in $\tilde{G}$ at most once. Finally, $\tilde{G}$ has at most np edges. □

5. $\mathcal{NP}$-hard problems

The inclusion constrained, the weight constrained and all min–max and max–min problems, however constrained, are strongly $\mathcal{NP}$-hard. We prove this by reduction from the Satisfiability Problem (SAT) [6].

We first take into account the weight constraints, and the min–max and max–min objective functions. We prove that the three problems have a common special case, and then reduce SAT to that special case [6]. The proof is still valid if $w_v = 1$ for all v, $p = 2$, and the triangle inequality holds. We neglect the inclusion constrained problems since Section 6 will prove that they are strongly $\mathcal{NP}$-hard even on special graphs. However, the same graph construction employed for weight constrained, min–max and max–min $GTPP$s can be used for inclusion constrained $GTPP$s.

Theorem 5. *The weight constrained, the min–max and the max–min* GTPP*s are strongly $\mathcal{NP}$-hard, even if all vertices have the same weight, $p = 2$, and the triangle inequality holds.*

Proof. Given any instance of the SAT problem, it is possible to build an auxiliary graph $G(V,E)$, such that the answer to the former (*Is there a truth assignment to the boolean variables u_i such that all clauses C_j be satisfied?*) is positive if and only if G can be partitioned into two trees satisfying suitable weight and cost conditions. Thus, any instance of SAT is equivalent to a particular instance of the $GTPP$. The vertex set V is composed of:

- a subset U containing a pair of vertices for each boolean variable, labelled as u_i and $\bar{u}_i$,

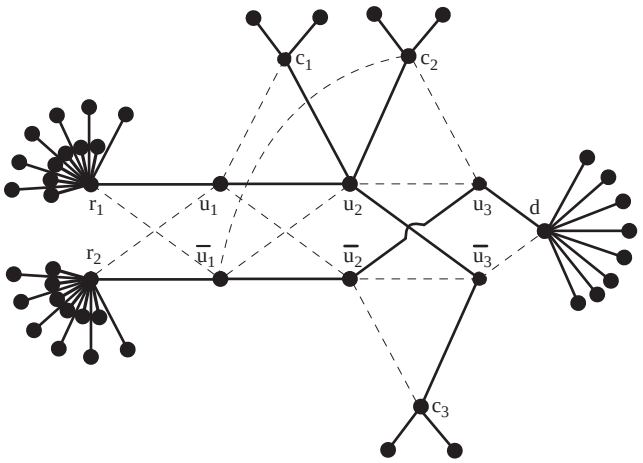

Fig. 1. Graph construction to prove the $\mathcal{NP}$-hardness of the weight constrained, the min–max and the max–min *GTPP*.

- a subset C_j for each clause, containing n vertices, one of which is labelled as c_j,
- a subset D containing mn vertices, one of which is labelled as d,
- two subsets R_1 and R_2 containing $n + mn + 1$ vertices each; one of the vertices in R_1 is labelled as r_1, one of the vertices in R_2 as r_2.

All the vertices in the graph have the same weight: $w_v = 1$ for all v in V. As for the edges, r_1 is connected to all of the other vertices in R_1, r_2 to all of the other vertices in R_2, d to all of the other vertices in D and each vertex c_j is connected to all of the other vertices in the corresponding subset C_j. In addition, r_1 and r_2 are connected to u_1 and $\bar{u}_1$. The vertices associated to each boolean variable, u_i and $\bar{u}_i$, are linked with those associated to the following one, u_{i+1} and $\bar{u}_{i+1}$. Vertices u_n and $\bar{u}_n$ are connected to d. Finally, each vertex c_j is connected either to u_i or to $\bar{u}_i$, according to which of the two corresponding literals appears in clause C_j. All of these edges have cost $c_e = 1$. The graph is completed by adding edges of cost γ. The triangle inequality holds as long as $1/2 \leqslant \gamma \leqslant 2$. See Fig. 1 for the graph corresponding to the boolean formula $(u_1 + u_2)(\bar{u}_1 + u_2 + u_3)(\bar{u}_2 + \bar{u}_3)$.

We now discuss separately the three partition problems, imposing conditions on the cost and weight of the trees such that they all reduce to the same special case: The edges of cost γ are not reported for the sake of clarity.

1. *Min–sum weight constrained problem*: we impose $\gamma = 2$, an upper or a lower bound (or both) $W = 2n + 2mn + 1$ on the weight of each tree, and an upper bound $K = 4n + 4mn$ on the total cost.
2. *Max–sum weight constrained problem*: we impose $\gamma = 1/2$, an upper or a lower bound (or both) $W = 2n + 2mn + 1$ on the weight of each tree, and a lower bound $K = 4n + 4mn$ on the total cost.

3. *Min–max problem*: we impose $\gamma = 2$ and an upper bound $K = 2n + 2mn$ on the cost of each tree.
4. *Max–min problem*: we impose $\gamma = 1/2$ and a lower bound $K = 2n + 2mn$ on the cost of each tree.

Since $|V| = 4n + 4mn + 2$, any spanning forest of two trees is composed of $4n + 4mn$ edges. Thus, the cost bound prevents the use of the edges costing γ, which can be removed. Moreover, in the min–max and the max–min problem each of the two trees must contain exactly $2n + 2mn$ edges. As well, in the weight constrained problem, whether the bound is imposed from below or from above or both, the two trees are composed of exactly $2n + 2mn + 1$ vertices and $2n + 2mn$ edges. So we are actually looking for two trees of the same cardinality, and employing only the edges of cost 1. Now, we show that this problem is equivalent to *SAT*.

Subsets R_1, R_2 and D, as well as each subset C_j, are fully included into one of the two trees. The cardinality constraint forces R_1 and R_2 to reside in different trees, respectively, T_1 and T_2. Suppose, with no loss of generality, that D belong to T_2: the only way to connect D to R_2 is through a path traversing exactly one vertex in each pair $(u_i, \bar{u}_i)$, as the cardinality constraint forbids to use vertices in C and to touch more than n vertices in U. The rest of U, and the whole of C, belong to T_1. A feasible solution to the *GTPP* determines a satisfying truth assignment, and viceversa. The theorem follows. □

5.1. Non approximability results

Among the $\mathcal{NP}$-hard *GTPP*s, two very special cases are known to be approximable, under the triangle inequality assumption: the min–sum and the min–max problems where the cardinality of the trees is fixed [9,10]. If the triangle inequality does not hold, it can be proved that no constant approximation ratio is possible, unless $\mathcal{P} = \mathcal{NP}$. We extend this proof to other min–sum and min–max versions of the *GTPP*, while the max–sum and max–min versions remain open.

Corollary 6. *If the triangle inequality does not hold, the inclusion-constrained and the weight-constrained min–sum* GTPP, *as well as the min–max* GTPP *admit no polynomial approximation algorithm with bounded error guarantee, even if all vertices have the same weight and* $p = 2$, *unless* $\mathcal{P} = \mathcal{NP}$.

Proof. Suppose, to the contrary, that there exists a polynomial approximation algorithm and a constant α such that, for every instance of one of these problems, the algorithm finds a solution $\tilde{F}$ satisfying $c(\tilde{F}) \leqslant \alpha c(F^*)$, where F^* is an optimal solution. Consider an instance of *SAT* and the corresponding construction used in Theorem 5. Set $\gamma = \alpha(4n + 4mn)$. Any feasible non-optimal solution employs at least one of the edges costing γ. Therefore, it is much more expensive. Thanks to its guaranteed performance ratio, an α-approximation algorithm would avoid these solutions, and determine an optimal one. Thus, it would solve *SAT* in polynomial time. To be concise, we do not provide the proof for the inclusion constrained case, which employs a very similar graph construction. □

Table 2
The computational complexity of the *GTPP*s on bipartite and grid graphs

Objective function	No side constraints	Root constraints	Inclusion constraints	Weight constraints
Min–sum Max–sum	$\mathscr{P}$	$\mathscr{P}$	Strongly $\mathscr{NP}$-hard	$\mathscr{NP}$-hard
Min–max	$\mathscr{NP}$-hard	$\mathscr{NP}$-hard	Strongly $\mathscr{NP}$-hard	$\mathscr{NP}$-hard
Max–min	$\mathscr{NP}$-hard	$\mathscr{NP}$-hard	Strongly $\mathscr{NP}$-hard (bipartite graphs) $\mathscr{NP}$-hard (grid graphs)	$\mathscr{NP}$-hard

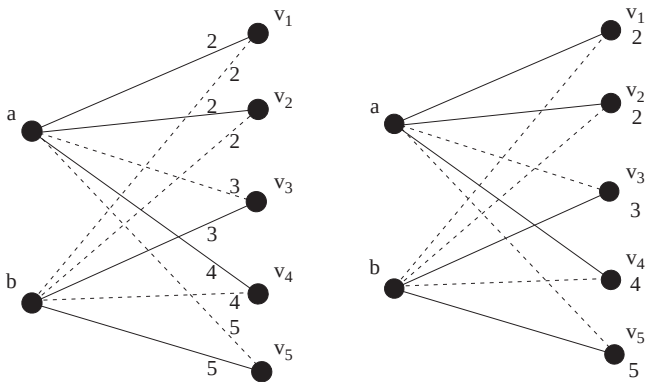

Fig. 2. The equivalence between graph partition and partition on bipartite graphs, with min–max or max–min objective (on the left) or with weight constraints (on the right).

6. Special cases

We briefly discuss in the following the computational complexity of the *GTPP*s on special graphs, namely bipartite graphs and grid graphs. Table 2 sums up the results.

Proposition 7. *Any min–max or max–min* GTPP, *as well as any weight-constrained* GTPP *is* $\mathscr{NP}$*-hard on bipartite graphs, even if* $p = 2$.

Proof. The proof extends one by Yamada et al. [16] for the min–max *GTPP* with singleton root sets (that is, fixed roots). Given a generic instance of the Partition Problem, in which a set $S = \{s_i\}$ of integers must be divided into two subsets of equal sum, build a complete bipartite graph $\hat{G}(\hat{V}, \hat{E})$, with $\hat{V} = \{a, b\} \cup V$.

For the min–max and the max–min problems, set $c_{av_i} = c_{bv_i} = s_i$ (see the left side of Fig. 2). If these problems were easy, one could tell in polynomial time whether

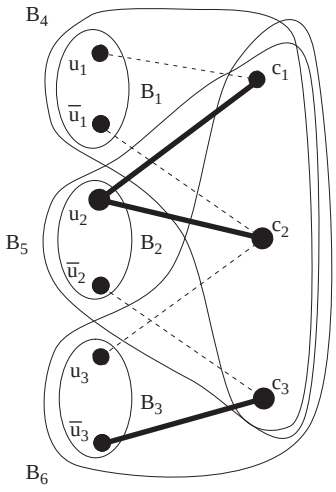

Fig. 3. The equivalence between inclusion constrained graph partition and *SAT* on bipartite graphs.

$\hat{V}$ can be partitioned into S_1 and S_2, so that the cost of both trees is not higher than $S^* = 1/2 \sum_{s_i \in S} s_i$ (in the min–max case) or not lower than S^* (in the max–min case). Should vertices a and b belong to the same tree, this would cost $\sum_{s_i \in S} s_i = 2S^*$ and the other one would reduce to a single vertex (with zero cost). This is unfeasible. Then, the two trees are stars centred in a and b and they both cost S^*, yielding a solution to the Partition Problem. Conversely, any solution to that problem also solves the *GTPP*.

For the weight constrained *GTPP*, set $w_{v_i} = s_i$ for $i = 1, \ldots, n$ (see the right side of Fig. 2). Impose an upper bound (or a lower bound, or both) equal to $S^* = 1/2 \sum_{i \in S} s_i$ on the weight of both trees. If the *GTPP* were easy, we could determine in polynomial time whether it is possible to partition $\hat{V}$ into two subsets whose weight would respect the S^* threshold. By construction, both of the trees would weigh S^* and the Partition Problem would be solved. □

Determining a feasible solution to an inclusion constrained *GTPP*, with any cost function, is strongly $\mathcal{NP}$-hard by reduction from *SAT*. Fig. 3 reports the graph corresponding to the boolean formula $(u_1 + u_2)(\bar{u}_1 + u_2 + u_3)(\bar{u}_2 + \bar{u}_3)$.

Theorem 8. *All inclusion constrained* GTPP*s are strongly $\mathcal{NP}$-hard on bipartite graphs.*

Proof. Let $\hat{G}(\hat{V}, \hat{E})$ be a bipartite graph, with $2n$ vertices on one shore (representing the literals $u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n$) and m vertices on the other shore (representing the logical clauses $C_1, \ldots, C_m$). Each clause vertex is linked to the literal vertices satisfying the clause. The first n boundary sets include each couple of literal vertices: $B_i = \{u_i, \bar{u}_i\}$ for $i = 1, \ldots, n$. The last n boundary sets include two literal vertices and all of the

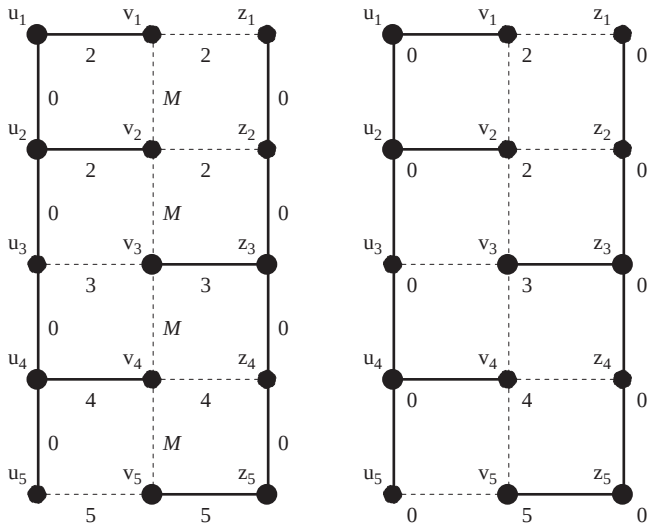

Fig. 4. The equivalence between graph partition and partition on grid graphs with min–max or max–min objective function (on the left) or with weight constraints (on the left).

clause vertices: $B_{i+n} = C \cup \{u_i, \bar{u}_i\}$ for $i = 1, \ldots, n$. The problem consists in determining a feasible partition of $\hat{G}$ into $2n$ trees.

Since the graph is bipartite, the first n trees reduce to single literal vertices. The other n literal vertices are disjoint from one another, and they can be considered as roots of the remaining n trees. If it is possible to connect the clause vertices to them in order to obtain a feasible solution, then each clause admits a satisfying literal assignment. Conversely, if such an assignment exists, the *GTPP* admits a feasible solution. □

Proposition 9. *The min–max and the max–min* GTPP, *as well as the weight-constrained* GTPP *are* $\mathcal{NP}$*-hard on grid graphs, even if* $p = 2$.

Proof. Consider a grid graph consisting of three columns of n vertices: $U = \{u_i\}$, $V = \{v_i\}$ and $Z = \{z_i\}$.

For the min–max (max–min) problem (see Fig. 4 on the left), let the edges between the vertices in U and those between the vertices in Z have zero cost; let the edges between the vertices in V have a very high cost $M \geqslant \sum_i s_i$ (or a very low cost $M \leqslant -\sum_i s_i$). In the end, let $c_{u_i v_i} = c_{v_i z_i} = s_i$. Let us impose an upper bound $S^* = 1/2 \sum_i s_i$ on the cost of each tree. This forbids the use of the edges costing M. As $p = 2$, either none of the vertices in V is isolated or a single one is. In the former case, at least n edges are incident in V and they saturate the cost bound. Each tree costs exactly S^* and is connected through the zero cost path along U and Z. In the latter case, let vertex v_{i^*} be isolated: all of the other vertices are connected, and the corresponding shortest spanning tree contains one edge for each vertex in $V \setminus \{v_{i^*}\}$ and both edges

for one of them. Its cost is $\sum_i s_i - s_{i^*} + \min_i s_i$ and it cannot be feasible, unless in trivial cases.

Conversely, if the Partition Problem admits a solution, one can build a feasible graph partition by considering a subset of (u_i, v_i) edges having total cost equal to S^* and the path linking all of the vertices in U. The complementary (v_i, z_i) edges and the path linking all of the vertices in Z determine the second feasible tree.

The reduction to the max–min problem can be proved in a similar way, after adding a large constant M to the cost of all edges apart those between the vertices in V, in order to prevent their use. Of course, the cost bound should be correspondingly increased.

For the weight constrained problem (see Fig. 4 on the right), set to zero the weight of the vertices in U and Z ($w_{u_i} = w_{z_i} = 0$), while those in V have given integer weights $w_{v_i} = s_i$. Impose an upper bound (or a lower bound, or both) equal to S^* on the weight of each tree. Finding a feasible partition is equivalent to solving the Partition Problem. In fact, if the latter admits a solution, the vertices in the central column can be divided into two clusters of equal weight S^*. The first cluster can be connected through the path linking the vertices in U, the second cluster through the path linking the vertices in Z.

Conversely, if there exists a feasible partition of the graph into two trees, both of them must weigh exactly S^* and the weighted vertices in each tree identify a feasible solution to the Partition Problem. □

As for inclusion constraints, we can prove strong $\mathcal{NP}$-hardness for the min–max and the min–sum problems by reduction from the Steiner Tree Problem. It is still open whether the max–min and max–sum problems are simply $\mathcal{NP}$-hard, as proved above, or strongly $\mathcal{NP}$-hard.

Theorem 10. *The min–sum and the min–max inclusion constrained* GTPP*s are strongly $\mathcal{NP}$-hard on grid graphs.*

Proof. The Steiner Tree Problem requires to determine a minimum cost tree which include a given set of mandatory vertices Z. It is strongly $\mathcal{NP}$-hard on grid graphs [5]. Let us consider a generic instance of the Steiner Tree Problem on a grid graph. If one could solve any min–sum or min–max inclusion constrained *GTPP* on a grid graph, one could also determine whether there exists a tree including all of the vertices in Z and costing less than a given threshold K.

In fact, let us set $B_1 = V$ and $B_r = V \setminus Z$ for $r = 2, \ldots, p$. Only the first tree can include the vertices in Z; therefore, it must. Let us search for a feasible tree partition of this graph into p trees such that its total cost, or the cost of its most expensive tree, is lower than K. If for any p between 2 and $n - |Z|$ such a solution exists, then the first tree of the partition costs less than K. Otherwise, no tree solves the Steiner Tree Problem, since a forest made up of that tree and a number of isolated vertices would also solve the *GTPP* (Fig. 5). □

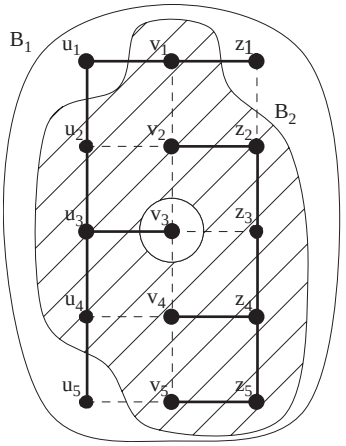

Fig. 5. The equivalence between inclusion constrained min–max grid graph partition and the Steiner Tree Problem.

References

[1] F.W. Ackermann, Zum Hilbertschen Aufban der Reelen Zahlen, Math. Ann. 99 (1928) 118–133.
[2] A.I. Ali, C.-H. Hwang, Balanced spanning forests and trees, Networks 21 (1991) 667–687.
[3] K. Altinkemer, B. Gavish, Heuristics with constant error guarantees for the design of tree networks, Management Sci. 32 (1988) 331–341.
[4] C. Berge, Graphes, 3rd Edition, Gauthier–Villars, Paris, 1983.
[5] M.R. Garey, D.S. Johnson, The rectilinear Steiner tree problem is $\mathcal{NP}$-complete, SIAM J. Appl. Math. 32 (1977) 826–834.
[6] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
[7] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, SIAM J. Comput. 24 (2) (1995) 296–317.
[8] J.C. Gower, G.J.S. Ross, Minimum spanning trees and single linkage cluster analysis, Appl. Statist. 18 (1969) 54–64.
[9] N. Guttmann-Beck, R. Hassin, Approximation algorithms for min–max tree partition, J. Algorithms 24 (2) (1997) 266–286.
[10] N. Guttmann-Beck, R. Hassin, Approximation algorithms for minimum tree partition, Appl. Math. 87 (1–3) (1998) 117–137.
[11] J.E. Hopcroft, R.M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2 (1973) 225–231.
[12] C. Imielińska, B. Kalantari, L. Khachiyan, A greedy heuristic for a minimum weight forest problem, Oper. Res. Lett. 14 (1993) 65–71.
[13] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc. 7 (1956) 48–50.
[14] C.H. Papadimitriou, The complexity of the capacitated tree problem, Networks 8 (1978) 217–230.
[15] D.J.A. Welsh, Matroid Theory, Academic Press, New York, 1976.
[16] T. Yamada, H. Takahashi, S. Kataoka, A heuristic algorithm for the mini–max spanning forest problem, European J. Oper. Res. 91 (1996) 565–572.
[17] T. Yamada, H. Takahashi, S. Kataoka, A branch-and-bound algorithm for the mini–max spanning forest problem, European J. Oper. Res. 93 (1997) 93–103.