SOLVING COLORING, MINIMUM CLIQUE COVER AND KERNEL PROBLEMS ON ARC INTERSECTION GRAPHS OF DIRECTED PATHS ON A TREE

OLIVIER DURAND DE GEVIGNEY, FRÉDÉRIC MEUNIER*, CHRISTIAN POPA, JULIEN REYGNER, AND AYRIN ROMERO

Abstract. Let $T = (V, A)$ be a directed tree. Given a collection $\mathcal{P}$ of dipaths on T, we can look at the arc-intersection graph $I(\mathcal{P}, T)$ whose vertex set is $\mathcal{P}$ and where two vertices are connected by an edge if the corresponding dipaths share a common arc. Monma and Wei, who started their study in a seminal paper on intersection graphs of paths on a tree, called them DE graphs (for directed edge path graphs) and proved that they are perfect. DE graphs find one of their applications in the context of optical networks. For instance, assigning wavelengths to set of dipaths in a directed tree network consists in finding a proper coloring of the arc-intersection graph.

In the present paper, we give

- a simple algorithm finding a minimum proper coloring of the paths.
- a faster algorithm than previously known ones finding a minimum multicut on a directed tree. It runs in $O(|V||\mathcal{P}|)$ (it corresponds to the minimum clique cover of $I(\mathcal{P}, T)$).
- a polynomial algorithm computing a kernel in any DE graph whose edges are oriented in a clique-acyclic way. Even if we know by a theorem of Boros and Gurvich that such a kernel exists for any perfect graph, it is in general not known whether there is a polynomial algorithm (polynomial algorithms computing kernels are known only for few classes of perfect graphs).

1. Introduction

Consider a directed tree $T = (V, A)$ (with n vertices) and a collection $\mathcal{P}$ of dipaths in this tree. The intersection graph $I(\mathcal{P}, T)$ is a graph whose vertex set is $\mathcal{P}$ and where two vertices are connected by an edge if the corresponding dipaths share a common arc. T with the collection $\mathcal{P}$ is called a *representation* of $I(\mathcal{P}, T)$.

These graphs generalize interval graphs, and also, as we will see, line-graphs of bipartite graphs.

Monma and Wei, who started their study in a seminal paper on intersection graphs of paths on a tree, called them DE graphs (for directed edge path graphs) [16]. The purpose of their paper is an extensive study of the intersection graph by considering various situations: the tree could be directed or undirected, and the paths could be identified with their vertex set or their edge set. They proved various properties and explained how such graphs can be efficiently recognized. In particular, they explained how a tree T with a collection $\mathcal{P}$ representing a given DE graph can be computed in polynomial time.

Afterwards, DE graphs received attention in papers with applications to optical fibers (see for instance [2, 3, 5, 8]), or from a combinatorial optimization point of view [6, 7]. The purpose of these studies consists often in the design of polynomial and efficient algorithms to compute various things, such as a minimum proper coloring of $I(\mathcal{P}, T)$ (which can be interpreted as assignement of wavelengths with a minimum number of distinct wavelengths) or a minimum clique cover of $I(\mathcal{P}, T)$ (which is the minimum multicut problem).

The purpose of the present paper is to continue this quest of algorithmic properties of DE graphs. When working on a DE graph, we will always assume that a tree with a collection of dipaths

*Corresponding author.

representing it is given. It is not a restriction in terms of polynomiality, since such a representation can be computed in polynomial time. Anyway, the time complexities will be expressed in terms of the size of the representation.

1.1. **Definitions.** Let $G = (V, E)$ be a (undirected) graph. A *stable set* of G is a subset $S \subseteq V$ such that no edge of G connects two vertices of S. The *stability number* is the maximum size of a stable set of G and is denoted by $\alpha(G)$.

A *clique* of G is a set of vertices any two of which are adjacent. The *clique number* is the maximum size of a clique and is denoted by $\omega(G)$.

A *coloring* of G is a color assignment on the vertices of G. If any two vertices connected by an edge get different colors (or equivalently, a partition of V into stable sets), we say that we have a *proper coloring*. The minimum number of colors in a proper coloring of G is called the *chromatic number* and is denoted by $\chi(G)$.

A *clique cover* of G is a partition of V into cliques. The minimum number of cliques in a clique cover of G is called the *clique cover number* of G and is denoted by $\bar{\chi}(G)$.

The following relations are immediate (recall that $\bar{G}$ denotes the complementary graph of G, in which edges become non-edges and conversely):

$$\alpha(G) = \omega(\bar{G}),\ \chi(G) = \bar{\chi}(\bar{G}),\ \alpha(G) \leq \bar{\chi}(G),\ \omega(G) \leq \chi(G).$$

G is said to be *perfect* if $\chi(H) = \omega(H)$ for all subgraphs H of G, or, equivalently, if $\alpha(H) = \bar{\chi}(H)$ for all induced subgraphs H of G, since according to the well-known theorem of Lovász [12], G is perfect if and only if $\bar{G}$ is perfect.

Let $D = (V, A)$ be a directed graph. Given a list of p ordered pairs of terminal vertices (s_i, t_i), with $i = 1, \ldots, p$, a *multicut* is a subset of arcs intersecting, for each $i = 1, \ldots, p$, all dipaths going from s_i to t_i. If D is a directed tree, note that for each ordered pair (s_i, t_i), there is at most one dipath from s_i to t_i.

A *cycle* of length s in an oriented graph is a sequence of arcs $(v_1, v_2), (v_2, v_3), \ldots, (v_s, v_1)$. A clique-acyclic orientation of G is an orientation $\vec{G}$ of the edges of G in such a way that each clique becomes acyclic. Said differently, there is no cycle of length 3 in the graph $\vec{G}$.

A *kernel* in an oriented graph $D = (V, A)$ is a stable subset $K \subseteq V$ such that for each $v \in V \setminus K$, there is a $w \in K$ such that $(v, w) \in A$.

Let $G = (V, E)$ be a bipartite graph with a total order $\preceq_v$ on the set of edges incident to v for each vertex $v \in V$. A *matching* M (a subset $M \subseteq E$ of disjoint edges) is said to be *stable* if whenever there is an edge $e = uv \in E \setminus M$, there is an edge $f \in \delta(u) \cap M$ such that $f \succeq_u e$ or $f \in \delta(v) \cap M$ such that $f \succeq_v e$, where $\delta(u)$ refers to the set of edges incident to the vertex u. We have the celebrated marriage theorem by Gale and Shapley.

Theorem 1 (Stable marriages theorem, [9])**.** *Let $G = (V, E)$ be a bipartite graph with a total order $\preceq_v$ on the set of edges incident to v for each vertex $v \in V$. Then there is a stable matching in G and it can be computed in $O(|V||E|)$.*

1.2. **The intersection graph $I(\mathcal{P}, T)$ as a perfect graph.** Monma and Wei proved various results for DE graphs. To ease the discussion, we summarize some of these results in a theorem.

Theorem 2 (Monma and Wei [16])**.** *DE graphs are perfect graphs, they can be recognized in polynomial time and, given a DE graph G, it is possible to compute in polynomial time a representation of it, i.e. a tree T and a collection of dipaths $\mathcal{P}$ such that $G = I(\mathcal{P}, T)$.*

Let $\mathcal{P} = \{P_1, \ldots, P_p\}$. The source of P_i is denoted s_i and its target t_i. In our context, $\alpha(I(\mathcal{P}, T))$ is the maximum number of arc-disjoint dipaths, $\omega(I(\mathcal{P}, T))$ is the maximum number of dipaths passing through a common arc, $\chi(I(\mathcal{P}, T))$ is the minimum number of colors that can be assigned to the dipaths in such a way that two dipaths sharing a common arc get different colors, and $\bar{\chi}(I(\mathcal{P}, T))$

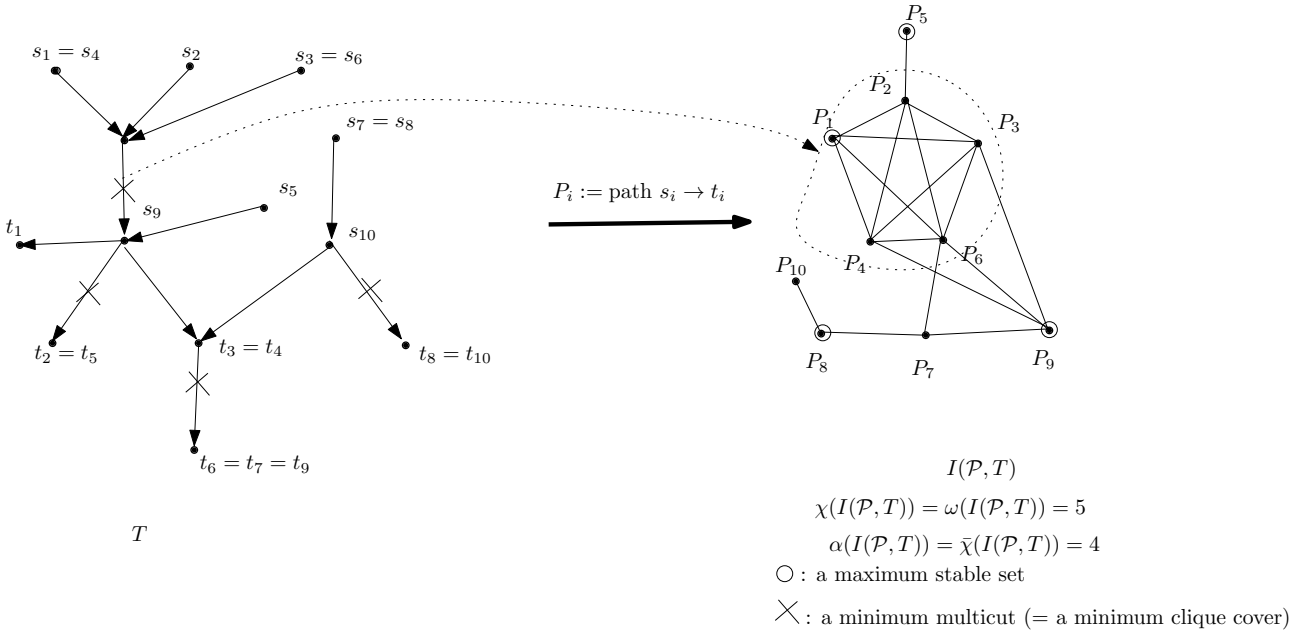

FIGURE 1. Illustration of the perfectness of the (arc)-intersection graph of dipaths on a directed tree.

is the minimum number of arcs intersecting all dipaths of $\mathcal{P}$, that is a minimum multicut of T with the (s_i, t_i) as ordered pairs of terminals. For the interpretation of $\omega(I(\mathcal{P}, T))$ and $\bar{\chi}(I(\mathcal{P}, T))$, we use the fact that $\mathcal{P}$ has the Helly property, noted first by Monma and Wei. In the present context, the Helly property means that, given a collection $P_{i_1}, \ldots, P_{i_s}$ of paths in $\mathcal{P}$ such that they pairwise contain a common arc, then the whole collection contains a common arc.

The equality $\alpha = \bar{\chi}$ says that the maximum number of arc-disjoint dipaths equals the minimum multicut, and the equality $\omega = \chi$ says that the maximum number of dipaths passing through a common arc is equal to the minimum colors that can be assigned to the dipaths in such a way that two dipaths sharing a common arc get different colors. Figure 1 is an illustration: the network on the left hand side is a representation of the DE graph on the right hand side; there are ten paths P_i, each of them is the unique dipath going from s_i to t_i.

All these values can be computed in polynomial time with the ellipsoid method and semidefinite programming [11]. Monma and Wei gave combinatorial algorithms for the computation of a minimum coloring and a maximum stable set of $I(\mathcal{P}, T)$. Since they developed a clique decomposition for their recognition algorithm, a classical method of Tarjan [20] allows to solve this problem polynomially (and recursively). For the maximum clique, it is straightforward using the representation: just check each arc one after each other (time complexity $O(n|\mathcal{P}|)$). The complexity of the minimum clique cover problem was left as an open question. Costa, Létocart and Roupin in their survey about multicut and integer multiflow [7] noted that the maximum stable set and the minimum clique cover problems admit a linear programming formulation with totally unimodular matrices, and hence can be solved polynomially (for instance with the ellipsoid method, or the interior-points method). For the maximum stable set problem, the best algorithm is that of Garg, Vazirani and Yannakakis [10].

By a theorem of Boros and Gurvich [4], we know that in any clique-acyclic orientation of a perfect graph, there is a kernel. In particular, in any clique-acyclic orientation of a DE graph $I(\mathcal{P}, T)$, there is a kernel. Using the Helly property, we get that a clique-acyclic orientation in a DE graph $I(\mathcal{P}, T)$ is equivalent to the existence of a reflexive and antisymmetric binary relation $\mathcal{R}$ defined on the set

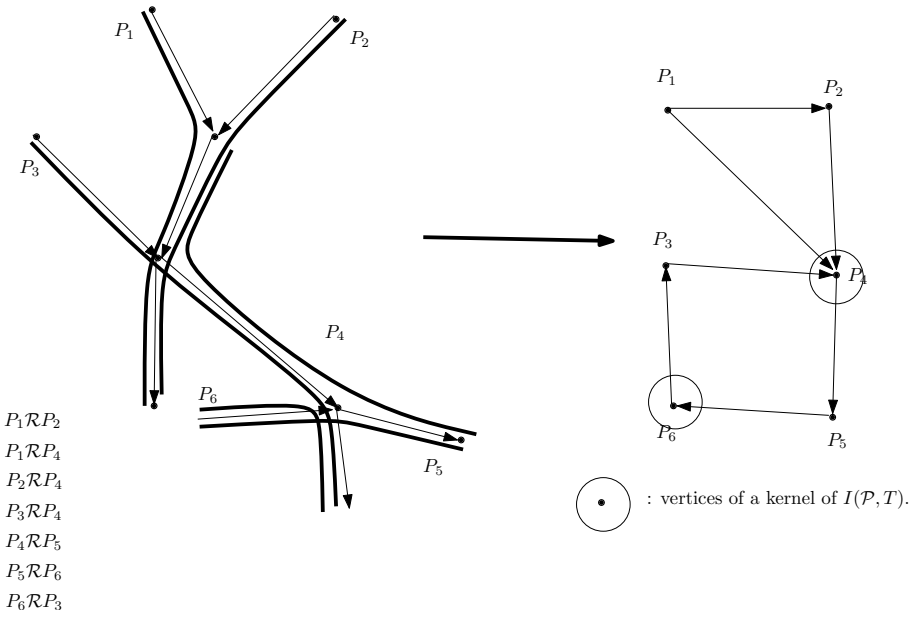

FIGURE 2. A kernel in the arc-intersection graph of the dipaths.

of dipaths $\mathcal{P}$ that induces for each arc a a total order $\preceq_a$ on the set of dipaths containing a: we orient an edge uv from u to v in the DE graph if the dipaths P_u (giving u) and P_v (giving v) of its representation are such that $P_u \mathcal{R} P_v$, and conversely.

See Figure 2 for an illustration.

Motivated by the question of computing a Nash equilibrium, Megiddo and Papadimitriou defined in [15] the complexity class TFNP, consisting exactly of all search problems in NP for which every instance is guaranteed to have a solution. A subclass of the TFNP class is the PPAD class for which the existence is proved through the following argument :

> *In any directed graph with one unbalanced node (node with outdegree different from its indegree), there must be another unbalanced node.*

This class was defined by Papadimitriou in 1994 [17] in a paper in which he also shows that there exists PPAD-complete problem, that is, problem in PPAD for which the existence of a polynomial algorithm would imply the existence of a polynomial algorithm for any problem in PPAD. Finding a kernel in a clique-acyclic orientation of a perfect graph belongs to the class PPAD, since it is a consequence of a theorem of Scarf that belongs to PPAD (see [1]). Nevertheless, even if the search problem associated to the Scarf theorem is known to be PPAD-complete, it is not known whether the problem of finding a kernel in a clique-acyclic orientation of a perfect graph is PPAD-complete. Hence, examples of families of perfect graphs for which this problem is polynomial are of great importance.

Nevertheless, algorithms are known for few classes of perfect graphs: chordal graphs (straightforward), bipartite graphs [18], line graphs of bipartite graphs (as noted by Maffray [13], a stable matching in a bipartite graph is a kernel in its line graph and Theorem 1 shows that it is polynomially computable).

1.3. **Results.** We show how to find a minimum coloring and a minimum clique cover of $I(\mathcal{P}, T)$ directly on the tree T, neither using clique decompositions nor using linear programming. We get a time complexity of $O(n|\mathcal{P}|)$ for both the minimum coloring and the minimum clique cover, where n is the number of vertices in T. Our algorithms are simpler than all previously known algorithms,

and clearly faster in the case of the minimum clique cover problem. Recall that the minimum coloring problem on the tree corresponds to the minimum wavelength assignement problem, and that the minimum clique cover problem to the minimum multicut problem. Hence, we get

Theorem 3. *Given a list of p ordered pairs of terminal vertices on a directed tree with n vertices, a minimum multicut can be computed in $O(np)$.*

We give an $O(n|\mathcal{P}|^2)$ algorithm that computes a kernel in any clique-acyclic orientation of $I(\mathcal{P}, T)$. Since given a DE graph, a representation in terms of T and $\mathcal{P}$ can be computed in polynomial time (Theorem 2), it provides a new class of perfect graphs for which the problem of finding a kernel is known to be polynomial.

Theorem 4. *The computation of a kernel in a clique-acyclic orientation of a DE graph can be done in polynomial time.*

1.4. **Plan.** Section 2 explains the basic observation, noted first by Monma and Wei in their paper, which will be useful for the description of the algorithms. Namely, that if the tree is a star, all problems can be reformulated in terms of a bipartite graph: coloring of the paths becomes coloring of the edges, arc-disjoint paths become matching, and a kernel becomes a stable matching. A tree can be then seen as a collection of intricate bipartite graphs that can be processed in a consistent way.

Section 3 is devoted to the the simple coloring algorithm that compute a minimum coloring of the dipaths in such a way that two dipaths get distinct colors.

Section 4 is devoted to the minimum multicut in a directed tree and Section 5 to the kernel problem.

2. Starting observation: stars and bipartite graphs

In this section, we explain how any of the problems we are interested in can be encoded on a bipartite graph. It was noted first by Monma and Wei (Theorem 6 (c) of [16]). This remark will be used many times in the paper.

Let $G = (V(G), A(G))$ be an oriented star, and denote by v the central vertex (a *star* is a vertex complete to a stable set). Denote by $u_1, \ldots, u_s$ the other vertices. Consider a set $\mathcal{P}$ of dipaths on this star.

Adding vertices $u'_1, \ldots$, we can assume that all dipaths have length 2. Build the bipartite graph B whose vertices are the u_i and the u'_i (vertices with outgoing arcs on one side, vertices with ingoing arcs on the other side) and whose edges connect vertices in the same dipath (with multiplicities). Hence there is a one-to-one correspondence between dipaths of G and edges of B. See Figure 3 for an illustration of the construction. Note that $I(\mathcal{P}, G)$ is the line graph of B.

Coloring the dipaths in G is equivalent to color the edges in B. Finding a set of arc-disjoint dipaths in G is equivalent to find a matching in B. Finally, finding a set of arc-disjoint dipaths in G that provides a kernel in the intersection graph $I(\mathcal{P}, G)$ is equivalent of finding a stable matching in B. Indeed, $I(\mathcal{P}, G)$ is the line graph of B and, as we have already said, Maffray [13] has noted that a stable matching in B is a kernel in its line graph (for two edges e and f of B sharing a common vertex v, the edge ef of the line graph is oriented from e to f if $e \preceq_v f$).

The construction of B can be done in linear time.

Remark : DE graphs are the common generalization of the line graph of bipartite graphs (as it is explained in the present section) and of the interval graphs (the tree of the representation is a directed path).

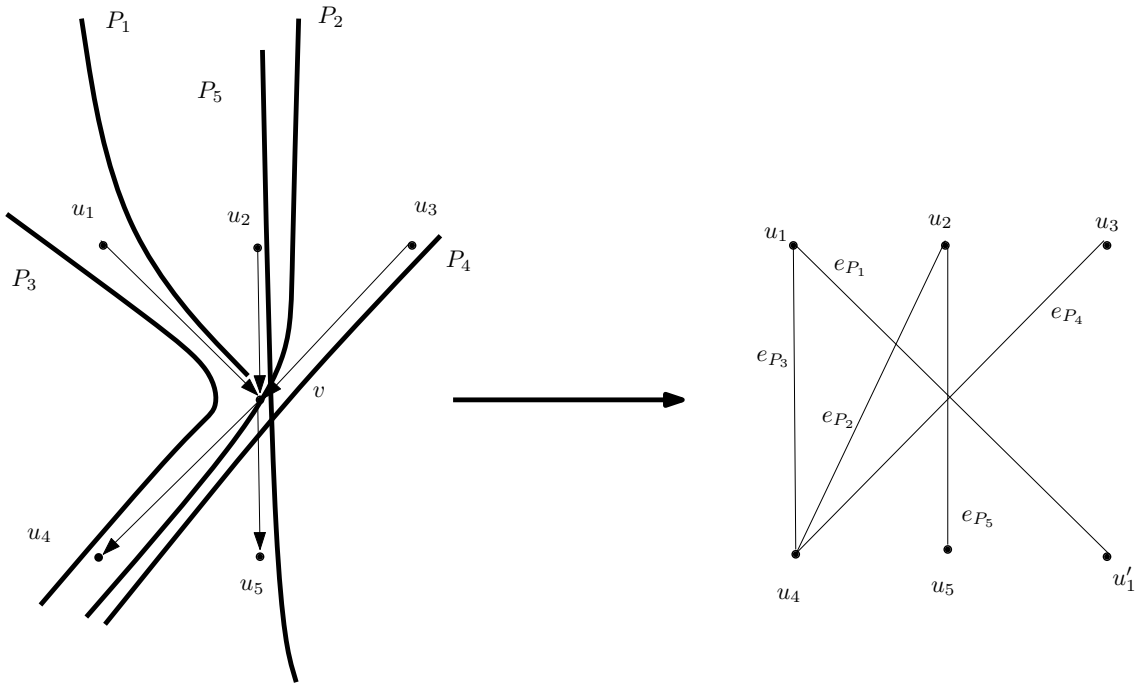

FIGURE 3. Illustration of the construction of the bipartite graph that represents the structure of the dipaths passing through a given vertex.

3. An easy algorithm to find a minimum proper coloring of the dipaths

There is an easy algorithm computing a minimum proper coloring of a DE graph using its tree representation. It provides a simpler algorithm than the one given by Monma and Wei, which uses the clique-tree decomposition and an approach by Tarjan, and an easy (and algorithmic) proof of the perfectness of DE graphs.

The strategy to color the dipaths is simple. Choose any vertex v of T. Consider the set of paths $\mathcal{P}_v$ passing through v. Coloring these paths is equivalent of coloring the edges of a bipartite multigraph B_v as explained in the previous section. Coloring the edges of a bipartite multigraph can be done in $O(\deg(v)|\mathcal{P}_v|)$, as explained by Schrijver in his book [19] (Theorem 20.13, p.333). Recall that according to Kőnig's edge-coloring theorem the minimum number of colors is equal to the maximal degree of the bipartite graph.

Once these paths are colored, start again with one of the neighbors of v, consider the set of paths passing through this neighbor w, color the paths of the associated bipartite multigraph B_w without changing the colors of the paths already colored (those passing through the arc vw), in $O(\deg(w)|\mathcal{P}_w|)$. We can color the dipaths while coloring the edges of the bipartite graphs B_u, taking for u the vertices of T one after the other, with the additional requirement that a new vertex must be the neighbor of a vertex already processed. Stop when all vertices have been processed. The tree structure of T explains why no color conflict can happen: when a new bipartite graph B_u is considered, the edges already colored emanate all from a common vertex (the vertex already processed) and it is easy to see that such a coloring can be extended on the whole bipartite graph B_u. At the end, one gets an admissible coloring, and the number of colors is equal to the maximum degree of the bipartite graphs B_u, which is exactly the maximum number of dipaths passing through a common arc.

The whole complexity is then in $O\left(\sum_{v\in V}\deg(v)|\mathcal{P}_v|\right) \leq O\left(\sum_{v\in V}\deg(v)|\mathcal{P}|\right) = O(n|\mathcal{P}|)$ since $\sum_{v\in V}\deg(v) = 2n$.

Note that this proof leads also to the equality of the chromatic number and the clique number for DE graphs, giving an algorithmic proof of the perfectness of DE graphs (already proved by Monma and Wei).

4. Minimum multicut in directed trees (minimum clique cover in DE graphs)

A theorem of Grötschel, Lovász and Schrijver [11] explains how to derive a minimum clique cover in a perfect graph once we know how to compute a maximum clique and a maximum stable set. This method would provide an $O(n \times \mathrm{DP})$ complexity for our problem, where DP is the complexity for finding a maximum set of arc-disjoint dipaths. The best algorithm for this last purpose is the one proposed by Garg, Vazirani and Yannakakis [10], and whose complexity is $O(n^{1/2}|\mathcal{P}|)$ at best (the exact complexity is not easy to determine). Actually, their algorithm is written for undirected trees, in which case the minimum multicut problem is NP-hard. Nevertheless, one would get with this approach a $O(n^{3/2}|\mathcal{P}|)$ time algorithm at best for the minimum clique cover of $I(\mathcal{P}, T)$.

We show in this section how to adapt the algorithm of Garg, Vazirani and Yannakakis in order to give simultaneously the maximum stable set and the minimum clique cover of $I(\mathcal{P}, T)$, that is a maximum set of arc-disjoint dipaths and a minimum set of arcs intersecting all dipaths of $\mathcal{P}$ (i.e. a minimum multicut for the list of endpoints of the dipaths of $\mathcal{P}$). Our approach improves the complexity, since the algorithm for the arc-disjoint dipaths problem is applied only once, and we finally get a $O(n|\mathcal{P}|)$ time algorithm. Before proving this, we need to prove a technical lemma about vertex-cover in bipartite graph, which is very similar to Kőnig's theorem, both for the statement and the proof.

Lemma 1. *Let $G = (V, E)$ be a bipartite graph, of color classes U and W, and suppose that we are given a vertex $x \in W$ of G contained in every maximum matching. Suppose given a maximum matching M. Then one can find in linear time a minimum vertex cover C having the following properties:*

(i): *$x \in C$ and*
(ii): *each edge e incident to x and contained in no maximum matching has its other endpoint contained in C, too.*

Proof. Let M be a maximum matching. One can assume that there is a neighbor y of x that is not covered by M; indeed, if not, add a new vertex, call it y, and add an edge between y and x. One cannot improve the cardinality of the maximum matching with this new edge yx, since, otherwise, this would mean that x is not in all maximum matching of G. Hence a minimum vertex cover of this new graph is a minimum vertex cover of the former one.

Make the classical construction of the directed graph D_M (see [19]) by orienting each edge $e = uw$ of G (with $u \in U$ and $w \in W$) as follows :

- if $e \in M$, then orient e from w to u.
- if $e \notin M$, then orient e from u to w.

Let U_M and W_M be the sets of vertices in U and W (respectively) missed by M, and define R_M to be the set of vertices reachable in D_M from U_M. So $R_M \cap W_M = \emptyset$. Then each edge uw in M is either contained in R_M or disjoint from R_M (that is $u \in R_M \Leftrightarrow w \in R_M$). Moreover, no edge of G connects $U \cap R_M$ and $W \setminus R_M$, as no arc of D_M leaves R_M. So $C := (U \setminus R_M) \cup (W \cap R_M)$ is a vertex cover of G. Since C is disjoint from $U_M \cup W_M$ and since no edge in M is contained in C, one has $|C| \leq |M|$. Therefore, C is a minimum-size vertex cover and contains x since $xy \in E$, $y \in U_M$ and $U_M \cap C = \emptyset$.

Now, consider the edge e as in the statement of the lemma. The other endpoint of e cannot be in R_M since otherwise it would be possible to switch between edges in M and edges not in M along a path or a cycle containing both e and x, without decreasing the size of the matching, and e would

be in a maximum matching of the graph. Hence e is necessarily between $U \setminus R_M$ and $x \in W \cap R_M$. Both endpoints of e are in C. □

Proof of Theorem 3. The algorithm we describe now follows the scheme proposed by Garg, Vazirani and Yannakakis [10].

Root the tree at a vertex r. Denote by $\mathcal{Q}_v$ all dipaths that contain the arc uv (independently of its orientation) where u is the father of v in the rooted tree, and by T_v the subtree rooted at v. Now consider the subproblem of finding the maximal number of arc-disjoint dipaths entirely contained in T_v, and denote by α_v this maximal number. Define $\mathcal{B}_v$ to be the set of *bad dipaths*, that is, those dipaths P of $\mathcal{Q}_v$ such that, when P is selected, the maximum number of arc-disjoint dipaths entirely contained in T_v and arc-disjoint from P is not α_v. Said differently, for any maximum subset of arc-disjoint dipaths in T_v, there is a dipath in this subset that shares a common arc with P. By convention, we set $\mathcal{Q}_r := \emptyset$ and $\mathcal{B}_r := \emptyset$.

The algorithm works in two passes, an upward one and a downward one. During the upward pass, all sets $\mathcal{B}_v$ are determined. During the downward pass, using the sets $\mathcal{B}_v$, a set $\mathcal{S}$ of arc-disjoint dipaths, and a set $\mathcal{C}$ of arcs are updated at each vertex v. These two sets are such that, at the end, $|\mathcal{S}| = |\mathcal{C}|$ and each dipath of $\mathcal{P}$ contains at least an arc of $\mathcal{C}$ (it is the multicut we are looking for). We will check by induction that any dipath fixed in $\mathcal{S}$ is not bad.

Upward pass: Process first the leaves. Clearly, $\mathcal{B}_v = \emptyset$. Then choose a vertex v whose sons have all been processed. Denote by u its father. Note that if P in $\mathcal{Q}_v$ has v as an endpoint, then clearly P is not in $\mathcal{B}_v$. Hence, to determine $\mathcal{B}_v$, we will consider dipaths using an arc wv (independently of its orientation), with w a son of v.

In order to determine the elements of $\mathcal{B}_v$, consider the star G_v induced by v and its neighbors, and consider on this star the dipaths induced by the non-bad dipaths of its sons, i.e. dipaths of $(\bigcup_{w \text{ is a son of } v} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{B}_w)$. So, the bad dipaths of the sons of v are not in G_v. According to Section 2, the question of disjoint paths in G_v is in one-to-one correspondence with the question of matching in a bipartite graph B, in which path P induces an edge $e_P = uw_P$, for some vertex w_P that is a son of v. Select such a path P. It is not a bad path of a son of v since they have been removed before making the construction of B. Then, such a P is in $\mathcal{B}_v$ if and only if there is no maximum matching in $B - u$ that avoids this w_P. Note moreover that if such a P is not in $\mathcal{B}_v$, it means that all maximum matchings of B cover u.

The other elements of $\mathcal{B}_v$ are the dipaths of $\mathcal{Q}_v$ that are already in a $\mathcal{B}_w$, for w a son of v.

Hence, it is easy to determine $\mathcal{B}_v$. It can be computed in linear time once the matching is computed.

See Figure 4 for an illustration. When the upward pass is over, $\mathcal{B}_v$ is known for each vertex v. We run now the downward pass.

Downward pass: We process first r. Consider the star centered at r and the dipaths induced by $(\bigcup_{w \text{ is a son of } r} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } r} \mathcal{B}_w)$. Find a maximum matching M in the corresponding bipartite graph (Section 2) and a minimum vertex cover C. Define $\mathcal{S}$ to be the dipaths corresponding to M and $\mathcal{C}$ to be the arcs corresponding to C. We have clearly $|\mathcal{S}| = |\mathcal{C}|$.

Now, take a vertex v whose father u has been processed. By induction, no element of $\mathcal{S}$ is bad. Consider the star centered at v with the dipaths induced by $(\bigcup_{w \text{ is a son of } v} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{B}_w)$. Denote by B the bipartite graph that corresponds. Two situations may occur:

- One of these dipaths has already been fixed in $\mathcal{S}$: Denote by f the edge in B that corresponds to the already fixed dipath.

  Find a maximum matching M that uses f. It exists since f stems from a non-bad dipath. Find a minimum vertex cover C provided by Lemma 1, whose vertex x is played by vertex u

of T. Since f corresponds to a non-bad dipath, u is in all maximum matchings, as required by Lemma 1.

Add to $\mathcal{S}$ all dipaths that correspond to edges in M. These dipaths are not bad, by definition of B. It makes $|M|-1$ new dipaths. Add to $\mathcal{C}$ all arcs vw with $w \in C$ except arc vu. It makes $|M|-1 = |C|-1$ new arcs. $\mathcal{S}$ and $\mathcal{C}$ are increased by the same quantity. Each dipath in $(\bigcup_{w \text{ is a son of } v} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{B}_w)$ but not in $\mathcal{Q}_v \setminus \mathcal{B}_v$ contains at least one of these new arcs (each dipath in $\mathcal{B}_v \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{B}_w)$ contains one of the new arcs, according to the second point of Lemma 1).

- None of these dipaths has already been fixed in $\mathcal{S}$: Delete from B all edges induced by the dipaths in $\mathcal{Q}_v \setminus \mathcal{B}_v$.

  Find a maximum matching M containing no edge induced by a path in $\mathcal{B}_v$ (it is possible by definition of $\mathcal{B}_v$) and a minimum vertex cover C in B.

  Add to $\mathcal{S}$ all dipaths that correspond to edges in M. These dipaths are not bad, by definition of B. It makes $|M|$ new dipaths. Add to $\mathcal{C}$ all arcs that correspond to vertices of C. It makes $|M| = |C|$ new arcs. $\mathcal{S}$ and $\mathcal{C}$ are increased by the same quantity. Each dipath in $(\bigcup_{w \text{ is a son of } v} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{B}_w)$ but not in $\mathcal{Q}_v \setminus \mathcal{B}_v$, contains at least one of these new arcs.

During the downward pass, when the processing of v is over, no dipath in $\mathcal{S}$ is bad and $|\mathcal{C}| = |\mathcal{S}|$.

At the end, we get a set of arc-disjoint dipaths $\mathcal{S}$ and a set of arcs $\mathcal{C}$ of same cardinality. It remains to prove that each dipath contains at least one arc of $\mathcal{C}$. It is enough to show that for each dipath P, there is a vertex v such that either

- we have simultaneously $P \in \mathcal{Q}_w \setminus \mathcal{B}_w$ and $P \in \mathcal{B}_v$ where w is a son of v or
- we have simultaneously $P \in \mathcal{Q}_w \setminus \mathcal{B}_w$ and v an endpoint of P, where w is a son of v or
- we have simultaneously $P \in \mathcal{Q}_w \setminus \mathcal{B}_w$ and $P \in \mathcal{Q}_{w'} \setminus \mathcal{B}_{w'}$ with w and w' sons of v.

Indeed we have seen that such dipaths always contain at least one arc of $\mathcal{C}$. Such a vertex v exists: either there is vertex v' such that $P \in \mathcal{B}_{v'}$, and then take v the farthest vertex of P from r that is such that $P \in \mathcal{B}_v$, or define v to be the nearest vertex of P from r.

The whole complexity is $O(n|\mathcal{P}|)$, since at each vertex v, the complexity is $O(\deg(v)|\mathcal{P}_v|)$ where $\mathcal{P}_v$ denotes the set of dipaths passing through v (complexity of computing a maximum matching in the bipartite graph). □

Note that this proof leads also to the equality of the stability number and the cardinality of a minimum clique cover for DE graphs, giving another algorithmic proof of the perfectness of DE graphs.

5. Kernel

In this section, we prove that the computation of a kernel in a clique-acyclic orientation of a DE graph can be done in polynomial time (Theorem 4).

Before proving Theorem 4, we state a theorem about stable matchings in bipartite graphs.

Theorem 5 ([14])**.** *Any two distinct stable matchings cover the same set of vertices.*

The algorithm is based on the following lemma, which derives directly from Theorem 5.

Lemma 2. *Let $B = (V(B), E(B))$ be a bipartite graph with a total order $\preceq_v$ on the set of edges incident to v for each vertex $v \in V(B)$. Choose a vertex u. Define $U \subseteq \delta(u)$ to be the set of edges $e \in \delta(u)$ that are in no stable matching of $(B \setminus \delta(u)) \cup \{e\}$. Then for all $f \in \delta(u) \setminus U$ and all $U' \subseteq U$, the edge f is in all stable matchings of $(B \setminus \delta(u)) \cup \{f\} \cup U'$.*

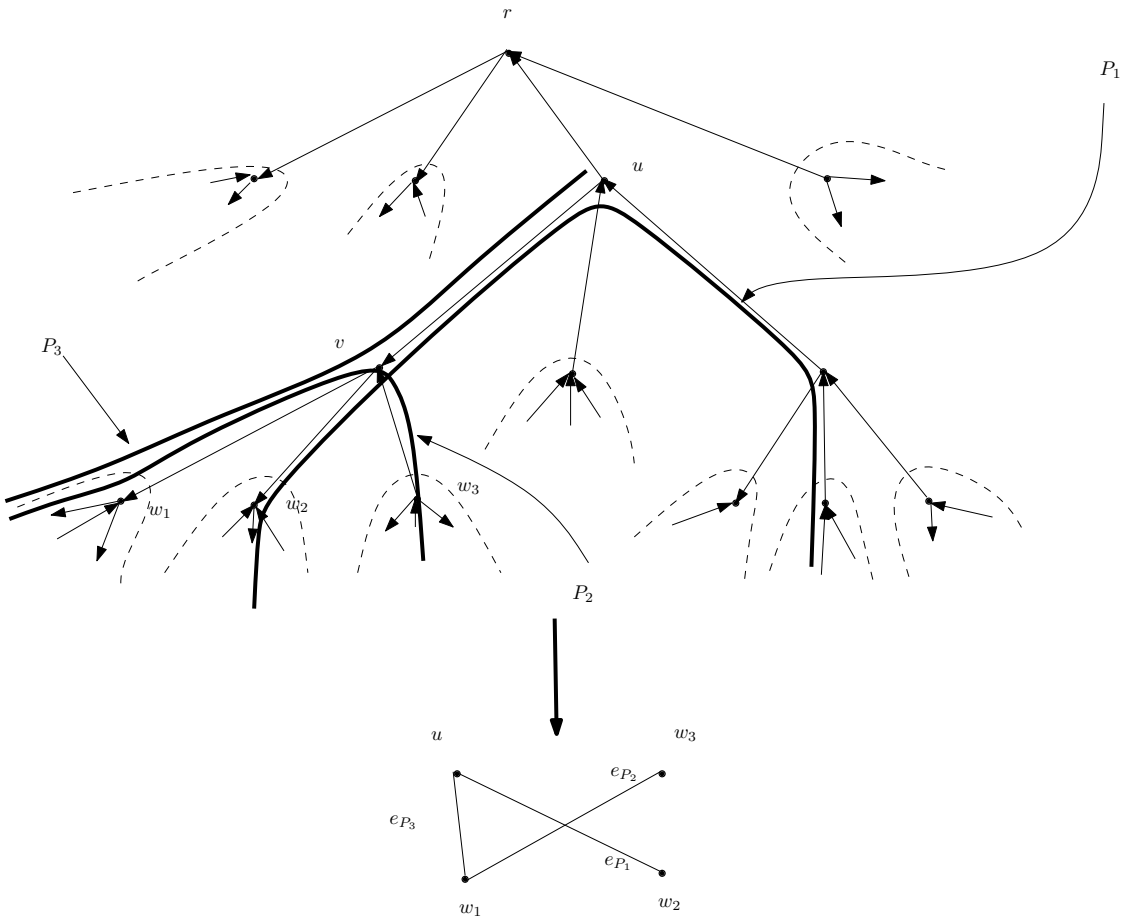

FIGURE 4. We are in the upward pass of the algorithm for multicut. Assume that $P_1 \in \mathcal{Q}_{w_2} \setminus \mathcal{B}_{w_2}$, $P_2 \in (\mathcal{Q}_{w_1} \cup \mathcal{Q}_{w_3}) \setminus (\mathcal{B}_{w_2} \cup \mathcal{B}_{w_3})$, and $P_3 \in \mathcal{Q}_{w_1} \setminus \mathcal{B}_{w_1}$. The bipartite graph below is the one built when processing v. Then $P_1 \in \mathcal{Q}_v \setminus \mathcal{B}_v$ and $P_3 \in \mathcal{B}_v$.

Proof. Take $f \in \delta(u) \setminus U$ and $U' \subseteq U$. Let M be a stable matching of $(B \setminus \delta(u)) \cup \{f\} \cup U'$ (which exists according to Theorem 1). None of the edge e in U' can belong to M, otherwise M would be a stable matching of $(B \setminus \delta(u)) \cup \{e\}$, which contradicts the definition of U. Thus M is a stable matching of $(B \setminus \delta(u)) \cup \{f\}$. By definition, f is in a stable matching of $(B \setminus \delta(u)) \cup \{f\}$, hence there is a stable matching of $(B \setminus \delta(u)) \cup \{f\}$ for which u is covered. According to Theorem 5, all stable matchings of $(B \setminus \delta(u)) \cup \{f\}$ cover u, and thus, M covers u. As the only edge that can cover u in M is f, we have $f \in M$.

□

Proof of Theorem 4. As before, we root the tree at a particular vertex r and denote by $\mathcal{Q}_v$ all dipaths that contain the arc uv where u is the father of v in the rooted tree, and by T_v the subtree rooted at v.

The algorithm works in the same spirit than the algorithm of previous section. Instead of having "bad dipaths", we will define *uninteresting dipaths* $\mathcal{U}_v$, which are such that, roughly speaking, when selected, they make impossible the existence of a kernel in the subtree T_v. By convention, we set $\mathcal{Q}_r := \emptyset$ and $\mathcal{U}_r := \emptyset$.

As before, the algorithm works in two passes, an upward one and a downward one. During the downward pass, dipaths will be added to a set $\mathcal{K}$ in order to make it at the end a kernel of $I(\mathcal{P}, T)$. By induction, it will be checked that no uninteresting dipath is added to $\mathcal{K}$.

We will use extensively Theorem 1 without explicitely stating it.

Upward pass: If v is a leaf, define $\mathcal{U}_v := \emptyset$. Now, take a vertex v for which $\mathcal{U}_w$ has already be defined, for each son w of v. Denote by u its father. Let $B = (V(B), E(B))$ be the bipartite graph that corresponds to the star centered at v and all its neighbors, with the dipaths induced by $(\bigcup_{w \text{ is a son of } v} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{U}_w)$. So, the uninteresting dipaths of the sons of v give no edge in B.

For each $e \in \delta(u)$, test if e is in a stable matching of $(B \setminus \delta(u)) \cup \{e\}$. To do this, we use Theorem 5. Indeed, according to this theorem, if a stable matching contains e, all stable matchings contain e, since in this bipartite graph, u is of degree 1. It is then enough to compute any stable matching and then to check whether e is in it or not. Now let P be the dipath from which e stems (it is not an uninteresting path of a son of v since they have been removed before making the construction of B). We put P in $\mathcal{U}_v$ if e is not in a stable matching of $(B \setminus \delta(u)) \cup \{e\}$.

We put moreover in $\mathcal{U}_v$ those dipaths of $\mathcal{Q}_v$ already in a $\mathcal{U}_w$, for w a son of v.

When the upward pass is finished, we have defined $\mathcal{U}_v$ for all vertices v of the tree T.

Downward pass: We process first r. Consider the star centered at r and the dipaths induced by $(\bigcup_{w \text{ is a son of } r} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } r} \mathcal{U}_w)$. Find a stable matching in the corresponding bipartite graph. It induces a set $\mathcal{K}$ of dipaths.

Now, take a vertex v whose father u has been processed. By induction, $\mathcal{K}$ contains no uninteresting dipath. Consider the star centered at v with the dipaths induced by $(\bigcup_{w \text{ is a son of } v} \mathcal{Q}_w) \setminus (\bigcup_{w \text{ is a son of } v} \mathcal{U}_w)$. Denote by B the bipartite graph that corresponds. Two situations can occur:

- One of these dipaths has already been fixed in $\mathcal{K}$: Denote by f the edge in B that corresponds to the already fixed dipath. Define B' to be B minus all edges that correspond to a dipath in $\mathcal{Q}_v \setminus \mathcal{U}_v$.

  Compute a stable matching in B', which necessarily contains f by definition of $\mathcal{U}_v$ and according to Lemma 2: indeed, f is in a stable matching of $(B \setminus \delta(u)) \cup \{f\}$ and the set U' of edges corresponding to the dipaths in $\mathcal{U}_v$ satisfies the requirement of Lemma 2.

  Add to $\mathcal{K}$ the dipaths whose corresponding edges are in the stable matching. Note that these dipaths are not uninteresting.
- None of these dipaths has already been fixed in $\mathcal{K}$: Define B' to be B minus all edges that correspond to a dipath in $\mathcal{Q}_v \setminus \mathcal{U}_v$.

  Compute a stable matching in B'.

  Add to $\mathcal{K}$ the dipaths whose corresponding edges are in the stable matching. By definition of $\mathcal{U}_v$, no dipath using arc uv is added to $\mathcal{K}$. Note that these dipaths are not uninteresting.

At the end, we get a set $\mathcal{K}$ of arc-disjoint dipaths. It remains to check that whenever a dipath P of $\mathcal{P}$ is not in $\mathcal{K}$, there is an arc $a \in P$, and a dipath $Q \in \mathcal{K}$ such that $a \in Q$ and $Q \succeq_a P$. It is enough to show that for each dipath P, there is a vertex v such that either

- we have simultaneously $P \in \mathcal{Q}_w \setminus \mathcal{U}_w$ and $P \in \mathcal{U}_v$ where w is a son of v or
- we have simultaneously $P \in \mathcal{Q}_w \setminus \mathcal{U}_w$ and v an endpoint of P, where w is a son of v or
- we have simultaneously $P \in \mathcal{Q}_w \setminus \mathcal{U}_w$ and $P \in \mathcal{Q}_{w'} \setminus \mathcal{U}_{w'}$ where w and w' are sons of v.

Indeed such dipaths are edges of a bipartite graph for which a stable matching is computed. As in the previous section, such a vertex v exists: either there is vertex v' such that $P \in \mathcal{U}_{v'}$, and then take v the farthest vertex of P from r that is such that $P \in \mathcal{U}_v$, or define v to be the nearest vertex of P from r.

The whole complexity is $O(n|\mathcal{P}|^2)$, since at each vertex v, the complexity is $O(\deg(v)|\mathcal{P}_v|^2)$ where $\mathcal{P}_v$ denotes the set of dipaths passing through v: we repeat at most $|\mathcal{P}_v|$ times the computation of a stable matching for each vertex v. □

Acknowledgement We are grateful to Michel Cosnard for having pointed out the question of a simple and fast algorithm for the coloring problem.

References

1. R. Aharoni and R. Holzman, *Fractional kernels in digraphs*, Journal of Combinatorial Theory (ser. B) **73** (1998), 1–6.
2. J.-C. Bermond, L. Braud, and D. Coudert, *Traffic grooming on the path*, Theoret. Comput. Sci. **384** (2007), 139–151.
3. J.-C. Bermond, M. Cosnard, D. Coudert, and S. Pérennes, *Optimal solution of the maximum all request path grooming problem*, Proceedings of the Advanced International Conference on Telecommunications, AICT06 (Guadeloupe, France), February 2006.
4. E. Boros and V. Gurvich, *Perfect graphs are perfect solvable*, Discrete Mathematics **159** (1996), 35–55.
5. I. Caragiannis, C. Kaklamanis, and P. Persiano, *Wavelength routing in all-optical tree networks: A survey*, Computers and Artificial Intelligence (2001).
6. M.-C. Costa, L. Ltocart, and F. Roupin, *A greedy algorithm for multicut and integral multiflow in rooted trees*, Operations Research Letters **31** (2003), 21–27.
7. ———, *Minimal multicut and maximal integer multiflow: A survey*, Eur. J. Op. Res. **162** (2005), 55–69.
8. T. Erlebach, T. Munchen, K. Jansen, C. Kaklamanis, and P. Persiano, *Optimal wavelength routing on directed fiber trees*, Theoretical Computer Science **221** (1999), 119–137.
9. D. Gale and L.S. Shapley, *College admissions and the stability of marriage*, American Mathematical Monthly (1962), 9–15.
10. N. Garg, V. V. Vazirani, and M. Yannakakis, *Primal-dual approximation algorithms for integral flow and multicut in trees*, Algorithmica **18** (1997), 3–20.
11. M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer, Berlin, 1988.
12. L. Lovász, *Normal hypergraphs and the perfect graph conjecture*, Discrete Math. **2** (1972), 253–267.
13. F. Maffray, *Kernel in perfect line graph*, Journal of Combinatorial Theory, Series B **55** (1992), 1–8.
14. D.G. McVitie and L.B. Wilson, *Stable marriage assignement for unequal sets*, BIT (1970), 295–309.
15. N. Meggido and C. H. Papadimitriou, *A note on total functions, existence theorems, and computational complexity*, Tech. report, IBM, 1989.
16. C. L. Monma and V. K. Wei, *Intersection graphs of paths in a tree*, J. Combin. Theory Ser. B **41** (1986), 141–181.
17. C. Papadimitriou, *On the complexity of the parity argument and other inefficient proofs of existence*, J. Comput System Sci. **48** (1994), 498–532.
18. M. Richardson, *Solutions of irreflexive relations*, Annals of Mathematics **58** (1953), 573–590.
19. A. Schrijver, *Combinatorial optimization*, Springer, 2003.
20. R. E. Tarjan, *Decomposition by clique separators*, Discrete Math. **55** (1985), 221–232.

Université Paris Est, LVMT, ENPC, 6-8 avenue Blaise Pascal, Cité Descartes Champs-sur-Marne, 77455 Marne-la-Vallée cedex 2, France.
E-mail address: frederic.meunier@enpc.fr

Ecole Polytechnique, 91128 Palaiseau Cedex, France
E-mail address: {olivier.durand-de-gevigney,christian.popa,julien.reygner,ayrin.romero-campos}@polytechnique.edu