Trees in Graphs With Conflict Edges or Forbidden Transitions

Mamadou Moustapha Kanté*, Christian Laforest**, and Benjamin Momège***

Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France
{mamadou.kante,laforest,momege}@isima.fr

Abstract. In a recent paper [*Paths, trees and matchings under disjunctive constraints*, Darmann et. al., Discr. Appl. Math., 2011] the authors add to a graph G a set of *conflicts*, i.e. pairs of edges of G that cannot be both in a subgraph of G. They proved hardness results on the problem of constructing minimum spanning trees and maximum matchings containing no conflicts. A *forbidden transition* is a particular conflict in which the two edges of the conflict must be incident. We consider in this paper graphs with forbidden transitions. We prove that the construction of a minimum spanning tree without forbidden transitions is still $\mathscr{NP}$-Hard, even if the graph is a complete graph. We also consider the problem of constructing a maximum tree without forbidden transitions and prove that it cannot be approximated better than $n^{1/2-\varepsilon}$ for all $\varepsilon > 0$ even if the graph is a star. We strengthen in this way the results of Darmann et al. concerning the minimum spanning tree problem. We also describe sufficient conditions on forbidden transitions (conflicts) to ensure the existence of a spanning tree in complete graphs. One of these conditions uses *graphic sequences*.

1 Introduction

In some practical situations, classical graphs are not complex enough to model all the constraints. For example, a city map can be modelled by a graph where streets are edges. However a car cannot always follow any route on this map. In some points it can be forbidden to turn left or right for example. This means that some paths in the graph are not valid. In the graph of a city with such restrictions, finding a spanning tree containing no restriction would be useful to ensure the connectivity between any pair of locations. The cars could travel on this tree submitted to no forbidden transitions. In this paper we investigate this kind of problem from a pure theoretical point of view.

In the following paragraphs we give the main definitions, notations and concepts that will be used throughout the paper. We also give some bibliographical references on related results.

* M.M. Kanté is supported by the French Agency for Research under the DORSO project.
** Ch. Laforest is supported by the French Agency for Research under the DEFIS program TODO, ANR-09- EMER-010.
*** B. Momège has a PhD grant from CNRS and région Auvergne.

Specific notions and notations. In this paper, we only consider undirected, unweighted and simple graphs. We refer to [2] for definitions and undefined notations. The vertex set of a graph G is denoted by V_G and its edge set by E_G. An edge between u and v in a graph G is denoted by uv. A *tree* is an acyclic connected graph and a *star* is a tree with a distinguished vertex adjacent to the other vertices. A complete graph (resp. star) with n vertices is denoted by K_n (resp. S_n). A path (or a cycle) of G is *Hamiltonian* if it contains all the vertices of G exactly once (all paths and cycles are elementary here).

If G is a graph, a *conflict* is a pair $\{e_1, e_2\}$ of edges of G. A conflict $\{e_1, e_2\}$ is called a *forbidden transition* if e_1 and e_2 are incident. In a forbidden transition $\{uv, vw\}$, the vertex v is called its *centre* and the vertices u and w its *extremities*. We denote by $(G, \mathcal{C})$ (resp. $(G, \mathcal{F})$) a graph G with a set of conflicts $\mathcal{C}$ (resp. with a set of forbidden transitions $\mathcal{F}$). (We use the notation $\mathcal{C}$ to denote conflicts and $\mathcal{F}$ for forbidden transitions.) A *spanning tree* T in $(G, \mathcal{C})$ is a spanning tree in G without conflicts, i.e., for any e, e' of T, $\{e, e'\} \notin \mathcal{C}$ (similarly for the other subgraph notions).

The *spanning tree problem without conflicts* (STWC) is, given $(G, \mathcal{C})$, constructs a spanning tree T in $(G, \mathcal{C})$, if one exists, otherwise say NO. We define similarly the *spanning tree problem without forbidden transitions* (STWFT). Similarly, the *Hamiltonian path (or cycle) problem without forbidden transitions* is denoted by HPWFT (or HCWFT). The problem of constructing a tree without forbidden transitions of maximum size will be denoted by MTWFT.

Works involving forbidden transitions. Graphs with forbidden transitions have already been investigated and several problems known to be polynomial in graphs have been shown to be intractable in graphs with forbidden transitions. For instance it is proved in [10] that knowing whether there exists a path between two nodes avoiding forbidden transitions is $\mathscr{NP}$-complete and a line between tractable and intractable cases have been identified. The problem of finding two-factors[1] is considered in [4] and a dichotomy between tractable and intractable instances is also given. In a very recent paper [7] we propose an exact exponential time algorithm that checks the existence of paths without forbidden transitions between two vertices; we also generalise the notion of cut in such graphs.

It is worth noticing that the $\mathscr{NP}$-hardness of the connectedness of two vertices in graphs with forbidden transitions does not imply the $\mathscr{NP}$-hardness of STWFT. Indeed, a classical result in graph theory (see [2]) states that a graph is connected if and only if it contains a spanning tree. Unfortunately, this is not the case anymore if we take into account $\mathcal{F}$ and STWFT. The simplest proof of this fact is the following. Consider a complete graph K_n with $n \geq 3$ where each possible transition is forbidden. Each pair of vertices is connected by a path with one edge, i.e. without forbidden transitions, but any spanning tree contains at least two edges (since $n \geq 3$) thus a forbidden transition.

Works involving conflicts. Since forbidden transitions are special cases of conflicts, the $\mathscr{NP}$-hard problems considered in [4, 10] remains $\mathscr{NP}$-hard in

[1] a subgraph such that for any vertex its in-degree and its out-degree is exactly one.

graphs with conflicts. But notice that dichotomy theorems in [4, 10] are no longer valid when dealing with conflicts. Some tractable cases of the path problem have been investigated in [8]. Another set of problems have been considered in the literature. For example authors of [9] considered the problem of constructing a scheduling such that two conflicting tasks cannot be executed on the same machine or packing problems under the condition that two conflicting items cannot be packed together. In [3] the authors proved that STWC is $\mathscr{NP}$-complete. They proved also similar results for the maximum matching problem.

Summary. We prove in Section 2 that STWFT is $\mathscr{NP}$-complete and characterise (in)tractable cases. Hence our result is stronger since we prove the hardness for a more restrictive type of conflicts. We go further by proving the hardness even in complete graphs with forbidden transitions. We also show that HPWFT and HCWFT are also $\mathscr{NP}$-complete in complete graphs with forbidden transitions. We furthermore prove that MTWFT cannot be approximated, even in stars with forbidden transitions. In Section 3.1 we adapt and use a result on graphical sequences to give a sufficient condition to ensure an always YES instance for STWFT in polynomial time when restricting instances to $(K_n, \mathcal{F})$. In Section 3.2, we prove that STWFT is polynomial in $(K_n, \mathcal{F})$ where each vertex is in a bounded number of forbidden transitions. We also prove that if each edge is involved in at most one conflict, then there always exist an Hamiltonian path in $(K_n, \mathcal{C})$. Finally, in Section 3.3 we describe a polynomial time process to transform any instance $(G, \mathcal{C})$ into an instance $(G_f, \mathcal{C}_f)$ containing less edges and conflicts and ensuring that $(G, \mathcal{C})$ is a YES instance for STWC iff $(G_f, \mathcal{C}_f)$ is.

2 Hardness Results

2.1 $\mathscr{NP}$-Hardness of STWFT, HPWFT and HCWFT

If $(G, \mathcal{C})$ is a graph with conflicts, we can associate with it a *conflict* graph that has as edge set $\mathcal{C}$ and as vertex set edges involved in $\mathcal{C}$. A 2-*ladder* is a disjoint union of edges and a 3-*ladder* is a disjoint union of paths with 3 vertices. We recall the following from [3].

Theorem 1 ([3]). STWC *is $\mathscr{NP}$-complete, even if conflict graphs are* 3*-ladders. However,* STWC *is polynomial in $(G, \mathcal{C})$ with 2-ladder conflict graphs.*

A slight modification of the proof of Theorem 1 gives the following result for forbidden transitions. Its proof is given for completeness.

Theorem 2. STWFT *is $\mathscr{NP}$-complete, even in bipartite graphs with* 3*-ladders as conflict graphs.* STWFT *is polynomial in $(G, \mathcal{F})$ with* 2*-ladder conflict graphs.*

Proof. The second statement follows directly from Theorem 1.

As in the proof of the $\mathscr{NP}$-completeness of STWC, we will reduce the (3,B2)-SAT to STWFT with 3-ladders as forbidden transitions. We recall that

a (3,B2)-SAT instance is a 3-SAT instance such that each variable occurs exactly four times, twice positive and twice negated.

Let I be an instance of the (3,B2)-SAT with m clauses $C_1, \ldots, C_m$ and n variables $X_1, \ldots, X_n$. We let $(G, \mathcal{F})$ with

$$V_G := \{r\} \cup \{c_j \mid C_j \text{ is a clause}\} \cup \{x_i, \overline{x_i}, r_i, s_i \mid X_i \text{ is a variable}\},$$
$$E_G := \{rr_i, r_i x_i, r_i \overline{x_i}, x_i s_i, \overline{x_i} s_i \mid i \in \{1, \ldots, n\}\} \cup$$
$$\{x_i c_j \mid X_i \text{ occurs positively in } C_j\} \cup \{\overline{x_i} c_j \mid X_i \text{ occurs negatively in } C_j\}.$$

G is bipartite (colour "black" the vertices: $c_1, \ldots, c_m, r_1, \ldots, r_n, s_1, \ldots, s_n$ which are independent and "white" the other also independent remaining vertices). The structure of the graph G is the same as in [3] but the conflicts we define are now forbidden transitions.

$$\mathcal{F} := \bigcup_{i \in \{1,\ldots,n\}} \{\{c_j x_i, x_i s_i\}, \{c_k x_i, x_i s_i\} \mid c_j x_i \in E_G \text{ and } c_k x_i \in E_G\} \cup$$
$$\bigcup_{i \in \{1,\ldots,n\}} \{\{c_j \overline{x_i}, \overline{x_i} s_i\}, \{c_k \overline{x_i}, \overline{x_i} s_i\} \mid c_j \overline{x_i} \in E_G \text{ and } c_k \overline{x_i} \in E_G\}$$

One easily checks that the conflict graph of $\mathcal{F}$ is a 3-ladder. We now prove that I is satisfiable iff there exists a spanning tree of $(G, \mathcal{F})$.

Assume I is satisfiable. Then there is a mapping $\delta : \{X_1, \ldots, X_n\} \to \{0, 1\}$ such that each clause is satisfied and for each clause there is a variable X_i such that $\delta(X_i)$ allows to satisfy it. Let T be the graph formed with the following edges:

$$\bigcup_{i \in \{1,\ldots,n\}} \{rr_i, r_i x_i, r_i \overline{x_i}\} \cup \bigcup_{i \in \{1,\ldots,n\}} \{s_i x_i \mid \delta(X_i) = 1\} \cup$$
$$\bigcup_{i \in \{1,\ldots,n\}} \{s_i \overline{x_i} \mid \delta(X_i) = 0\} \cup$$
$$\bigcup_{i \in \{1,\ldots,n\}} \{x_i c_j, x_i c_k \mid \delta(X_i) = 1 \text{ and } X_i \text{ occurs positively in } C_j \text{ and in } C_k\} \cup$$
$$\bigcup_{i \in \{1,\ldots,n\}} \{\overline{x_i} c_j, \overline{x_i} c_k \mid \delta(X_i) = 0 \text{ and } X_i \text{ occurs negatively in } C_j \text{ and in } C_k\}$$

One checks that T spans G, is acyclic, is connected and does not contain a forbidden transition. Therefore, T is a spanning tree of $(G, \mathcal{F})$.

Assume now that $(G, \mathcal{F})$ has a spanning tree T without forbidden transitions. For each $i \in \{1, \ldots, n\}$ for which exactly one the edges $x_i s_i$ or $\overline{x_i} s_i$ is in E_T, we do the following assignment $\delta : \{X_1, \ldots, X_n\} \to \{0, 1\}$

$$\delta(X_i) := \begin{cases} 1 & \text{if } \overline{x_i} s_i \in E_T \\ 0 & \text{if } x_i s_i \in E_T. \end{cases}$$

The other variables X_i receive arbitrary assignments. We claim that the assignment δ satisfies the instance I. Let us consider any clause C_j. There exists

some $i \in \{1, \ldots, n\}$ such that $x_i c_j \in E_T$ or $\overline{x_i} c_j \in E_T$. If $x_i c_j \in E_T$, then $x_i s_i \notin E_T$ and therefore $\overline{x_i} s_i \in E_T$. By the definition of δ, we have $\delta(X_i) = 1$ and then C_j is satisfied by $\delta(X_i)$. Similarly, if $\overline{x_i} c_j \in E_T$, we have $\overline{x_i} s_i \notin E_T$ and then $x_i s_i \in E_T$. Again, by the definition of δ, we have $\delta(X_i) = 0$, so C_j is satisfied by $\overline{\delta(X_i)}$. We conclude that I is satisfied by δ. □

If we take as parameter the number of conflicts a vertex (an edge) is involved in graphs with forbidden transitions, Theorem 2 gives a sharp line between tractable and intractable cases. We leave open the question for a dichotomy between tractable and intractable cases with respect to conflict graphs as done in [4, 10]. In the following, we show that when restricting to complete graphs, the $\mathscr{NP}$-completeness of STWFT remains true. For any $(G, \mathcal{F})$ such that G has $n \geq 3$ vertices, construct the complete graph K_n with the same set of vertices than G (and with all possible edges) and $\overline{\mathcal{F}(G)} := \mathcal{F} \cup \{\{e, f\} \mid e \in E_{K_n} \setminus E_G, f \in E_{K_n}, e \neq f$ and e and f incident in $K_n\}$.

Lemma 1. *T is a spanning tree of $(G, \mathcal{F})$ iff T is a spanning tree of $(K_n, \overline{\mathcal{F}(G)})$.*

Proof. It is clear that any spanning tree of $(G, \mathcal{F})$ is also a spanning tree of $(K_n, \overline{\mathcal{F}(G)})$. Conversely, assume now that T is a spanning tree without forbidden transitions of $(K_n, \overline{\mathcal{F}(G)})$. Since $n \geq 3$, T does not contain any "non-edge" of G, otherwise T would contain a forbidden transition. Therefore, T is a spanning tree without forbidden transitions of $(G, \mathcal{F})$. □

From Theorem 2 and Lemma 1, we can prove the following.

Theorem 3. STWFT *is $\mathscr{NP}$-complete in complete graphs with forbidden transitions.*

It is well-known that K_n contains many Hamiltonian paths or cycles that can be computed in polynomial time (if $n \geq 3$). This is not the case in complete graphs with forbidden transitions.

Theorem 4. HPWFT *and* HCWFT *are $\mathscr{NP}$-complete in complete graphs with forbidden transitions.*

Proof. We reduce the Hamiltonian path (or cycle) problem to HPWFT (or HCWFT). Let G be an n-vertex graph without forbidden transitions with $n \geq 3$ and let $(K_n, \overline{\mathcal{F}(G)})$ be the complete graph with forbidden transition associated with it (see definition of $\overline{\mathcal{F}(G)}$ before Lemma 1). One can easily show that G contains an Hamiltonian path (or cycle) if and only if $(K_n, \overline{\mathcal{F}(G)})$ contains an Hamiltonian path (or cycle) containing no forbidden transitions. But the problem of determining whether a graph G contains an Hamiltonian path or cycle is $\mathscr{NP}$-complete (see [5]). □

2.2 Inapproximability of MTWFT

The previous results show that constructing a spanning tree without forbidden transitions is a hard problem. We investigate here the optimisation version. Given $(G, \mathcal{F})$, we denote by $\alpha(G, \mathcal{F})$ the maximum size of a tree in $(G, \mathcal{F})$. Notice that if $(G, \mathcal{F})$ is a YES instance for STWFT, then $\alpha(G, \mathcal{F}) = |V|$.

Theorem 5. *Let $(G, \mathcal{F})$ and let n be the number of vertices of G. Then $\alpha(G, \mathcal{F})$ cannot be approximated with a ratio better than $n^{1/2-\varepsilon}$ for all $\varepsilon > 0$ even if G is a star.*

Proof. We will reduce the maximum clique problem to MTWFT in stars. Let G be a graph with n vertices. Construct the star G' with vertex set $V_G \cup \{r\}$ and edge set $\{ru \mid u \in V_G\}$ (r is a new vertex), and let $\mathcal{F} := \{\{ru, rv\} \mid uv \notin E_G\}$. We claim that T is a tree of size k in $(G', \mathcal{F})$ if and only if $T \setminus r$ induces a clique of size $k-1$ in G.

Let T be a tree of size k in $(G', \mathcal{F})$. Hence, T is a star S_k with distinguished vertex r and $k-1$ other vertices from V_G. As T contains no forbidden transitions, for all u and all v in $T \setminus r$, we have $uv \in E_G$. Therefore, $T \setminus r$ induces a clique of size $k-1$ in G.

Conversely, let $C := \{u_1, \ldots, u_k\}$ be a clique of size k in G. Then in G' none of the edges $ru_1, \ldots, ru_k$ is involved in a pair in $\mathcal{F}$. Therefore, $C \cup \{r\}$ induces a tree of size $k+1$ in $(G', \mathcal{F})$.

Now, using the fact that one cannot construct a clique of maximum size in an n-vertex graph with a better approximation ratio than $n^{1/2-\varepsilon}$ for all $\varepsilon > 0$ (see [1]) we get the desired result. □

3 Constructive Results in Complete Graphs

From proof of Theorem 2, we deduce the $\mathscr{NP}$-completeness of STWFT even if the number of forbidden transitions an edge or vertex is involved is bounded. However, the reduction in the proof of Theorem 3 does not preserve this property. We will see in this Section 3.2 that bounding the number of conflicts an edge or vertex is involved implies a polynomial time algorithm for STWFT in complete graphs. We will also provide some other sufficient conditions.

3.1 A Sufficient Condition to Contain a STWFT

For $(K_n, \mathcal{F})$, we will construct a graph G with the same set of n vertices and containing only the edges of G that are not in any forbidden transitions of $\mathcal{F}$. If G is connected, then we are done since we can just take any spanning tree of G, and it will be of course a spanning tree of $(K_n, \mathcal{F})$. So, we will assume that G is not connected and let us denote by $C_1, \ldots, C_k$ its $k > 1$ connected components and n_i the number of vertices of C_i. In the following we will also use C_i to denote the set of the n_i vertices of the component C_i. Some components

are not necessarily complete graphs and some of them may be composed of only one vertex.

We call *Edge Between Components*, noted EBC, an edge having its two extremities in two different components. The general idea is the following. If it is possible to connect the k components of G using EBC, in a "meta-tree" of components, in such a way that each vertex of each component is incident to at most one such EBC, then $(K_n, \mathcal{F})$ contains a spanning tree. Indeed, this "meta-tree" is a tree of the k components, it is connected and cycle-free. Inside each component, it is sufficient to take any spanning tree. These k trees connected by these EBC form a spanning tree T of $(K_n, \mathcal{F})$ (T is connected and is cycle-free). Indeed, T contains no forbidden transitions because, by construction, edges of components are part of no forbidden transitions and EBC are pairwise non incident by construction. In Theorem 7 we give a sufficient condition under which it is possible to do this construction. Before going further, we need some notions, definitions and preliminary results.

A sequence $n_1, \ldots, n_k$ of positive integers ($n_i \geq 1$) is a SDT (*Sequence of Degrees Tree*) if there exists a tree T of k vertices denoted by $u_1, \ldots, u_k$ such that $d_T(u_i) \leq n_i$. We will use the following theorem.

Theorem 6 ([6]). *Let $n_1, \ldots, n_k$ be a sequence of positive integers, $k \geq 2$. There exists a tree with k vertices having degrees $n_1, \ldots, n_k$ if and only if $\sum_{i=1}^{k} n_i = 2k - 2$.*

We underline the fact that the proof of Theorem 6 in [6] describes a polynomial time algorithm to construct the tree from the sequence.

Lemma 2. *The sequence of positive integers $n_1, \ldots, n_k$, $k \geq 2$, is a* SDT *if and only if $\sum_{i=1}^{k} n_i \geq 2k - 2$.*

Proof. We suppose first that $\sum_{i=1}^{k} n_i \geq 2k - 2$ and we show that $n_1, \ldots, n_k$ is a SDT. We decrease the value of some n_i (keeping them strictly positive) to obtain a sum equal to $2k - 2$. This operation can easily be done in polynomial time. Then we can apply Theorem 6 on this new sequence. In the corresponding tree T the degrees are less than $n_1, \ldots, n_k$ and hence this sequence is a SDT.

Let us show now that if $n_1, \ldots, n_k$ is a SDT, then $\sum_{i=1}^{k} n_i \geq 2k - 2$. As $n_1, \ldots, n_k$ is a SDT, there exists a tree T whose k vertices $u_1, \ldots, u_k$ are such that $d_T(u_i) \leq n_i$ (for $i = 1, \ldots, k$). But, it is well-known that in any graph, the sum of degrees of vertices is equal to two times the number of edges and in a tree the number of edges is equal to the number of vertices minus 1. This gives here $\sum_{i=1}^{k} d_T(u_i) = 2(k-1)$ and we get the expected inequality. □

Theorem 7. *Let $(K_n, \mathcal{F})$ and let $n_1, \ldots, n_k$ be the number of vertices of the k connected components induced by the n vertices of K_n and all the edges that are not in any forbidden transitions. If $\sum_{i=1}^{k} n_i \geq 2k - 2$, then $(K_n, \mathcal{F})$ contains a spanning tree that can be constructed in polynomial time.*

Proof. If $\sum_{i=1}^{k} n_i \geq 2k - 2$, then by Lemma 2 there exists a tree T_C with k vertices $u_1, \ldots, u_k$ such that $d_{T_C}(u_i) \leq n_i$. Now, replace each vertex u_i by the

connected component C_i having n_i vertices. For each edge of T_C, between C_i and C_j choose a vertex u in C_i and a vertex v in C_j and connect them by an EBC (this EBC exists since G is a complete graph). These two vertices u and v will not be used in any other connections between components. The number of vertices in each component is sufficient to ensure that property. Now construct in each component C_i any tree spanning its n_i vertices. The whole graph composed of these k trees plus the selected EBC forms a spanning tree of $(K_n, \mathcal{F})$. Note that, as the proofs of Theorem 6 and Lemma 2 are constructive and polynomial, there is a polynomial time algorithm to construct it. □

3.2 Other Sufficient Conditions for a Polynomial Testing

In this section we study the case where each vertex is the extremity of a limited number of forbidden transitions. We have shown in Theorem 4 that deciding whether $(K_n, \mathcal{F})$ contains an Hamiltonian Path is $\mathscr{NP}$-complete. We first notice that when each edge is in at most one conflict, it is possible to construct one (recall that in [3] the authors proved a polynomial testing for STWC in such graphs).

Theorem 8. *Let $(K_n, \mathcal{C})$ be such that the conflict graph associated with $\mathcal{C}$ is a 2-ladder. Then $(K_n, \mathcal{C})$ contains an Hamiltonian path and it can be constructed in polynomial time.*

Proof. We construct the Hamiltonian path in $(K_n, \mathcal{C})$ step by step, by adding one by one the vertices and keeping the property that the chosen vertices form a path in $(K_n, \mathcal{C})$. We begin with any two vertices of K_n. Suppose now that we have constructed a path in $(K_n, \mathcal{C})$ with $p \geq 2$ vertices, denoted by H. We denote by a and b the two extremities of H and a' (resp. b') the unique neighbour of a (resp. b) in H. Consider a vertex c outside H.

Case 1. If one of $\{a'a, ac\}$ or $\{b'b, bc\}$ is not a conflict, then we can add c as a new extremity (by adding the edge ac or bc) of H that becomes a path with $p+1$ vertices in $(K_n, \mathcal{C})$.
Case 2. In the other cases, this means that $\{a'a, ab\}$ is not a conflict (otherwise the edge $a'a$ would be involved into 2 conflicts) and similarly for $\{ab, bc\}$. One can construct a path H' composed in this order of: $b', \ldots, a', a, b, c$ which is a path with $p+1$ vertices in $(K_n, \mathcal{C})$.

We can therefore conclude that $(K_n, \mathcal{C})$ contains an Hamiltonian path. Since at each step the construction can be done in polynomial time, an Hamiltonian path can be constructed in polynomial time. □

We now look at the case where each vertex is in a limited number of forbidden transitions.

Fact 1 *Let $(K_n, \mathcal{F})$ be given. We suppose that each vertex is the extremity of at most k forbidden transitions and that $n \geq k+1$. If $(K_n, \mathcal{F})$ contains a tree*

T with $k+1$ vertices, then one can extend it to a spanning tree of $(K_n, \mathcal{F})$ in polynomial time.

Proof. If $n = k+1$ then T is already a spanning tree of $(K_n, \mathcal{F})$. If $n > k+1$, consider a vertex u of K_n outside T. As u is the extremity of at most k forbidden transitions and as T contains $k+1$ vertices, T contains a vertex v which is the centre of no forbidden transitions of extremity u. One can complete T by connecting u to v. This induces no forbidden transitions in this new tree that now has $k+2$ vertices. We do the same process with this new tree containing $k+2$ vertices, etc. At each step one can connect any vertex outside the current tree to this tree by adding no forbidden transitions. One can continue until obtaining a spanning tree. If the initial T is given, then completing it into a spanning tree can be done in polynomial time. □

Fact 2 *Let k be a fixed positive integer. There is a polynomial time algorithm that constructs in $(G, \mathcal{F})$ a tree with $k+1$ vertices if and only if there exists one.*

Proof. We apply here a brute force method: Generate all the subsets with $k+1$ vertices; Each subset induces a graph with $k+1$ vertices; Test in each such graph all the possible spanning trees. When one such tree without forbidden transitions is found, stop and return it. If none is found, this means that there is no such tree (since the process is exhaustive).

Generating all the subsets with $k+1$ vertices can be done in $O(n^{k+1})$. We generate all the trees with $k+1$ vertices in such induced subgraphs. There are at most $(k+1)^{k-1}$ trees (this is a well-known result, see [2]). Each such tree can be generated and tested in polynomial time. But, as k is a constant, $(k+1)^{k-1}$ is also a constant. The whole process is a polynomial time algorithm able to construct a required tree if and only if there exists one. □

Theorem 9. *Let k be a fixed positive integer. Let $(K_n, \mathcal{F})$ be given. If each vertex of K_n is the extremity of at most k forbidden transitions, then in polynomial time one can decide whether there exists a spanning tree in $(K_n, \mathcal{F})$ and, in this case, construct one in polynomial time.*

Proof. If K_n contains at most $k+1$ vertices, then the technique used in the proof of Fact 2 shows that one can determine and construct in polynomial time a spanning tree in $(K_n, \mathcal{F})$ if and only if there exists one. Let us consider now the case where K_n contains more than $k+1$ vertices.

Suppose that the algorithm in the proof of Fact 2 constructs a tree T. Thanks to Fact 1 one can extend it into a spanning tree of $(K_n, \mathcal{F})$. Both operations are done in polynomial time.

Suppose now the opposite, that is the algorithm in Fact 2 constructs no tree. In this case, $(K_n, \mathcal{F})$ contains no spanning tree. Indeed, if it contains one, T, then one can easily extract from T a tree on $k+1$ vertices, without forbidden transitions. This is in contradiction with Fact 2. □

We notice that it is challenging to reduce the time complexity of the procedure in Fact 2 from $O(n^{k+1})$ to $O(f(k) \cdot n^c)$ for some constant c not depending

in k and n. One just notices that a naive local search from a given vertex that maximises at each step the number of neighbours of the current vertex will not work since one can reduce the problem of finding a maximum clique in a graph to a such local search.

Theorem 10. *Let k be a fixed positive integer and let $(K_n, \mathcal{F})$ be given. If each vertex of K_n is the extremity of at most k forbidden transitions and if $n \geq 2k+1$, then $(K_n, \mathcal{F})$ necessarily contains a spanning tree that can be constructed in polynomial time.*

Proof. The global idea is the following: First construct a tree T without forbidden transitions on $k+1$ vertices in polynomial time and then use Fact 1 with this tree T to extend it and finish the proof.

If u is a vertex of K_n we denote by $Ext(u)$ the set of vertices of K_n which are centres of a forbidden transition having u as an extremity. Hence, by hypothesis, for all u, $|Ext(u)| \leq k$.

Let us construct T by selecting first its vertices. Take any vertex u_1. Take any vertex $u_2 \neq u_1$ which is not in $Ext(u_1) : u_2 \notin Ext(u_1) \cup \{u_1\}$, etc., take u_{i+1} a vertex not already taken and not in $Ext(u_i)$; $u_{i+1} \notin Ext(u_i) \cup \{u_1, \ldots, u_i\}$, etc. until obtaining a vertex u_{k+1}. One can always choose at each step a new vertex u_{i+1} because u_{i+1} is any vertex outside the set $Ext(u_i) \cup \{u_1, \ldots, u_i\}$ but $|\{u_1, \ldots, u_i\}| \leq k$ and $|Ext(u_i)| \leq k$; as $n \geq 2k+1$, u_{i+1} can always be selected.

The tree T is then the path $u_1, u_2, \ldots, u_{k+1}$ spanning the selected vertices. The only transitions in T are of the form $\{u_{i-1}u_i, u_iu_{i+1}\}$. But such a transition is not forbidden since u_{i+1} was selected outside $Ext(u_i)$ (if the transition $\{u_{i-1}u_i, u_iu_{i+1}\}$ is forbidden, this means that u_{i+1} would have been the extremity of a forbidden transition with centre u_i, which is not the case by construction).

The path/tree T contains no forbidden transitions and has $k+1$ vertices. This tree T always exists and can be constructed in polynomial time. We end the construction of the spanning tree by using the polynomial time constructing process of Fact 1 while T is given here. □

3.3 Simplification of $(G, \mathcal{C})$

We describe a process to simplify an instance $(G, \mathcal{C})$ (if it is possible) by suppressing edges and conflicts to obtain a new (reduced) instance $(G_f, \mathcal{C}_f)$ in which there is a spanning tree if and only if there is a spanning tree in $(G, \mathcal{C})$. $(G_f, \mathcal{C}_f)$ is constructed iteratively, step by step. Let $G_0 = G$ and $\mathcal{C}_0 = \mathcal{C}$. For each i, we let H_i be the subgraph of G_i composed of all the vertices of G and only edges that are not involved in a conflict in $\mathcal{C}_i$ and let S_i be the set of edges of G_i that are in a conflict of $\mathcal{C}_i$ **and** whose two extremities are in the same connected component of H_i. The shape of the algorithm is in Fig. 1.

As at each step some edges are removed, the algorithm terminates and is polynomial. We denote by $(G_f, \mathcal{C}_f)$ its final result. The graph G_f contains all the vertices of the initial graph G and a subset of its edges. Moreover $\mathcal{C}_f \subseteq \mathcal{C}$.

$i = 1$;
while ($S_{i-1} \neq \emptyset$) do
  let G_i be obtained from G_{i-1} by deleting edges in S_{i-1};
  let $\mathcal{C}_i$ be obtained from $\mathcal{C}_{i-1}$ by removing conflicts involving at least an edge in S_{i-1};
  $i = i + 1$;
endwhile
return $(G_{i-1}, \mathcal{C}_{i-1})$;

Fig. 1. Simplification of $(G, \mathcal{C})$

Theorem 11. *$(G, \mathcal{C})$ contains a spanning tree if and only if $(G_f, \mathcal{C}_f)$ contains a spanning tree.*

Proof. If $(G_f, \mathcal{C}_f)$ contains a spanning tree T, then T is also a spanning tree of $(G, \mathcal{C})$. Indeed, T covers all the vertices of G and it contains no conflicts of $\mathcal{C}$. Otherwise, assume that two edges e and e' T are in conflict in $\mathcal{C}$. As $\{e, e'\} \notin \mathcal{C}_f$, this means that the conflict was eliminated during the construction of $(G_f, \mathcal{C}_f)$. However the algorithm removes a conflict only if one of its edges is removed. Hence e and e' cannot be both in G_f; Contradiction.

Consider now a spanning tree T of $(G, \mathcal{C})$. This tree T covers all the vertices of G. Let us denote by $C_1, \ldots, C_k$ the k connected components of G_f in which all the edges that are in at least one conflict of $\mathcal{C}_f$ are removed. Let uv be any edge of T that has its two extremities into two different such connected components. Let us show that the edge uv is in G_f. If not, this means that it was removed at some step in the construction of $(G_f, \mathcal{C}_f)$, say step i and the two vertices u and v are in a same connected component of H_i since the algorithm deletes an edge involved in a conflict only if its two extremities are in a same connected component of H_i. However, the algorithm guarantees that an edge in a connected component of H_i will be always kept in the remaining steps $j > i$. So the edge uv is an edge in G_f.

Let I_C be the set of edges of T that are in G_f and have their two extremities between two connected components $C_1, \ldots, C_k$. Consider now the graph G' composed of all the vertices of G and of all the edges of I_C and all the edges of the connected components $C_1, \ldots, C_k$. This graph is clearly connected. Moreover, it contains no conflicts of $\mathcal{C}_f$. Indeed, let us consider any pair e and e' of edges of G'.

Case 1. There exist C_i and C_j such that $e \in C_i$ and $e' \in C_j$ (we may have $i = j$). By construction this means that they are not involved in a conflict of $\mathcal{C}_f$.
Case 2. e and e' are both in I_C. As I_C is a set of edges of a tree without conflicts in $\mathcal{C}$, edges e and e' do not form a conflict in $C_f \subseteq \mathcal{C}$.
Case 3. One edge, say e, is in a connected component and the other one, e', is in I_C. As e is in a connected component, by construction it is not involved in a

conflict of $\mathcal{C}_f$.

It is then easy to construct any spanning tree of G' (with a BFS for example) that is a spanning tree in $(G_f, \mathcal{C}_f)$. □

If $(G, \mathcal{C})$ is given, let $G(\overline{\mathcal{C}})$ be the graph containing all the vertices of G but only the edges that are not involved in a conflict of $\mathcal{C}$.

Corollary 1. *If $G_f(\overline{\mathcal{C}_f})$ is connected, then $(G, \mathcal{C})$ contains a spanning tree.*

It is easy to give instances $(G, \mathcal{C})$ in which $G(\overline{\mathcal{C}})$ is not connected while $G_f(\overline{\mathcal{C}_f})$ is connected and thus contains a trivial solution (any spanning tree). This simplification process leads to transform some instances that seem to be too complicated to solve but that are in fact trivial.

References

1. Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation.* Springer, 1999.
2. A. Bondy and U.S.R. Murty. *Graph Theory.* Springer London Ltd, 2010.
3. Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726?1735, 2011.
4. Zdeněk Dvořák. Two-factors in orientated graphs with forbidden transitions. *Discrete Mathematics*, 309(1):104–112, 2009.
5. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, 1979.
6. Gautam Gupta, Puneet Joshi, and Amitabha Tripathi. Graphics sequences of trees and a problem of frobenius. *Czechoslovak Mathematical Journal*, 57(132):49–52, 2007.
7. Mamadou Moustapha Kanté, Christian Laforest, and Benjamin Momège. An exact algorithm to check the existence of (elementary) paths and a generalisation of the cut problem in graphs with forbidden transitions. In *SOFSEM*, pages 257–267, 2013.
8. Petr Kolman and Ondřej Pangrác. On the complexity of paths avoiding forbidden pairs. *Discrete Applied Mathematics*, 157(13):2871 – 2876, 2009.
9. Ulrich Pferschy and Joachim Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009.
10. Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2-3):261–273, 2003.