An Exact Algorithm to Check the Existence of (Elementary) Paths and a Generalisation of the Cut Problem in Graphs with 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. A *graph with forbidden transitions* is a pair (G, F_G) where $G := (V_G, E_G)$ is a graph and F_G is a subset of the set $\{(\{y, x\}, \{x, z\}) \in E_G^2\}$. A *path* in a graph with forbidden transitions (G, F_G) is a path in G such that each pair $(\{y, x\}, \{x, z\})$ of consecutive edges does not belong to F_G. It is shown in [S. Szeider, Finding paths in graphs avoiding forbidden transitions, DAM 126] that the problem of deciding the existence of a path between two vertices in a graph with forbidden transitions is NP-complete. We give an exact exponential time algorithm that decides in time $O(2^n \cdot n^5 \cdot \log(n))$ whether there exists a path between two vertices of a given n-vertex graph with forbidden transitions. We also investigate a natural extension of the *minimum cut* problem: we give a polynomial time algorithm that computes a set of forbidden transitions of minimum size that disconnects two given vertices (while in a minimum cut problem we are seeking for a minimum number of edges that disconnect the two vertices). The polynomial time algorithm for that second problem is obtained via a reduction to a standard minimum cut problem in an associated *allowed line graph*.

1 Introduction

Algorithms manipulating graphs are often used to solve concrete situations in many applied fields. Finding a path between two given points/vertices is a fundamental basic tool that often serves as subroutine in many more complex algorithms and software (for example in flows (improving paths between a source and a sink), in scheduling (notion of constraint and critical path), in network for routing operations, etc.). Several well-known polynomial time algorithms are able to do this task: DFS, BFS (to find shortest paths in unweighted graphs), Dijkstra (for weighted graphs). They are widely available in software packages

* 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.

(like Maple and Mathematica[1] for example) and are taught in most of first level computer science or engineering courses all around the world (see a reference book on algorithms like [6]).

A path (simple or elementary) P in a graph G is just a list of consecutive (incident) edges (or arcs) of G between a given first vertex s and a final vertex t. To construct such a path P given G, s and t, the classical algorithms use all the potentiality of the graph: namely, when a given intermediate vertex u is reached, there is no restrictions on the following vertex that can be reached: *any* neighbour of u. Of course, BFS for example does not explore an already explored (and marked) vertex w but, as w is a neighbour of u, this possibility is taken into account among all possibilities.

This is a strong hypothesis in several real applications. Indeed, in some concrete networks, it is *not* possible, coming from a point a towards a point b to continue towards point c. For example in several large streets of cities, it is forbidden to turn left at point b (towards point c) and to cross a road (if one come from point a preceding b). Many such transits are forbidden in all the countries; several other restrictions exist (no "U" turn for example).

All these concrete limitations are due to the system modelled by graphs (routes, systems of production, etc.) in which all the paths are *not* possible because they have a local transition that is not allowed.

These *forbidden* transitions must be added into the knowledge of the graph. Then the algorithms that are supposed to construct paths between two vertices must take them into account. The hard part of the work starts at this point. Indeed, unlike the classical situation where there are several algorithms mentioned above, adding forbidden transitions strongly increases the complexity of the situation: In [8], Szeider shows that knowing whether there exists a path between two nodes avoiding forbidden transitions is NP-Complete while it is polynomial without these constraints (see [6]).

Several other studies have been done on graphs with forbidden transitions. For example, in [3] the authors want to find an Eulerian path in a graph representing biological data (from DNA sequencing) where all transitions between these biological elements are not allowed. Later, in [4], this practical problem serves as a motivation for a graph theoretic study. The authors prove, among other results, that finding an Eulerian path is NP-Complete when the graph contains forbidden transitions. In [7] the problem of finding two-factors[2] is considered. The author gives conditions on the type of transitions under which deciding whether there is a two-factor avoiding forbidden transitions in G is polynomial or NP-complete. In [1,9] the authors investigate a more general form of transitions; they propose polynomial time algorithms to find a shortest path avoiding forbidden *subpaths*. However, their paths are not elementary (we will call *walks* these objects later). This is a huge difference with the study of [8] where Szeider consider elementary paths.

[1] See http://www.maplesoft.com/ and http://www.wolfram.com/

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

Summary. In Section 2 of our paper, we give preliminary definitions, notations and concepts. In Section 3 we propose an exponential time algorithm based on inclusion-exclusion principle having a complexity of $O(2^n \cdot n^5 \cdot \log(n))$ (where n is the number of vertices) to decide if the graph contains or not a (elementary) path between two given vertices avoiding forbidden transitions. In Section 4 we investigate an equivalent problem to the minimum cut problem: we propose a polynomial time algorithm to compute the minimum number of allowed transitions to transform into forbidden ones to disconnect a given vertex s to a given vertex t (instead of cutting edges as in the original well-known cut problem).

2 Preliminaries

In this paper we consider only simple graphs. If A and B are two sets, $A \backslash B$ denotes the set $\{x \in A \mid x \notin B\}$. The size of a set A is denoted by $|A|$.

We refer to [5] for graph terminology not defined in this paper. The vertex-set of a graph G (directed or not) is denoted by V_G and its edge-set (or arc-set if it is directed) by E_G. An edge between two vertices x and y in an undirected graph G is denoted by $\{x, y\}$ and an arc from $x \in V_G$ to $y \in V_G$ in a directed graph G is denoted by (x, y). For a vertex $x \in V_G$ we let $E_G(x)$ be the set of edges or arcs incident with x; the *degree* of x, defined as $|E_G(x)|$, is denoted by $d_G(x)$.

If $\mathcal{C}$ is a class of graphs, we denote by $\mathcal{C}^{\text{ind}}$ the class of graphs $\{H \mid H$ is an induced subgraph of some graph $G \in \mathcal{C}\}$. We denote by P_3, K_3, $2K_2$, P_4 and L_4 the undirected graphs depicted in Fig. 1.

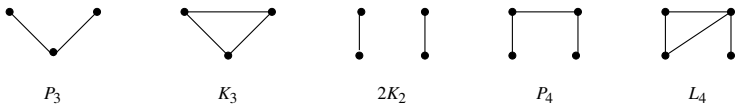


Fig. 1.

In the **undirected case**, a *graph with forbidden transitions* is a pair (G, F_G) where G is a graph and F_G is a subset of the set $T_G := \{(\{y, x\}, \{x, z\}) \in E_G^2\}$ of *transitions* of G. Transitions of F_G are called the *forbidden transitions* and the transitions of $A_G := T_G \setminus F_G$ are called the *allowed transitions*. We will also denote a transition $(\{y, x\}, \{x, z\})$ by the triplet (y, x, z). If for all the transitions $(\{y, x\}, \{x, z\}) \in F_G$ we have $(\{y, x\}, \{x, z\}) \in F_G \Leftrightarrow (\{x, z\}, \{y, x\}) \in F_G$ the forbidden transitions are called *symmetric* and in this case we define for each vertex x a *transition graph* A_x with $V_{A_x} := E_G(x)$ and $E_{A_x} := \{\{e_i, e_j\} \subseteq V_{A_x} \mid (e_i, e_j) \in A_G\}$. The set $\mathcal{A}_G := \{A_x \mid x \in V_G\}$ is called the system of allowed transitions.

A *walk* between s and t or an (s, t)-walk in (G, F_G) is a sequence of vertices $(s = x_1, x_2, \ldots, x_k = t)$ such that $\{x_i, x_{i+1}\} \in E_G$ for every $1 \leq i \leq k - 1$

and $(\{x_{i-1}, x_i\}, \{x_i, x_{i+1}\}) \in A_G$ for every $2 \leq i \leq k-1$. Such a walk is called a walk on k vertices and it may also be represented by the sequence of edges $(\{x_1, x_2\}, \ldots, \{x_i, x_{i+1}\}, \ldots \{x_{k-1}, x_k\})$.

In the **directed case**, a *directed graph with forbidden transitions* is a pair (G, F_G) where G is a graph and F_G is a subset of the set $T_G := \{((y,x),(x,z)) \in E_G^2\}$ of *transitions* of G. Transitions of F_G are called the *forbidden transitions* and the transitions of $A_G := T_G \setminus F_G$ are called the *allowed transitions*. We will also denote a transition $((y,x),(x,z))$ by the triplet (y,x,z). A *walk* from s to t or an (s,t)-walk in (G, F_G) is a sequence of vertices $(s = x_1, x_2, \ldots, x_k = t)$ such that $(x_i, x_{i+1}) \in E_G$ for every $1 \leq i \leq k-1$ and $((x_{i-1}, x_i), (x_i, x_{i+1}))$ is a transition of A_G for every $2 \leq i \leq k-1$. Such a walk is called a walk on k vertices and it may also be represented by the sequence of arcs $((x_1, x_2), \ldots, (x_i, x_{i+1}), \ldots, (x_{k-1}, x_k))$.

In the two cases, a *shortest* (s,t)-walk is a walk for which k is minimum. A *path* is a walk where each vertex appears once. If there is an (s,t)-walk in G we say that G is (s,t)-connected; s and t are called disconnected if there is no (s,t)-walk.

We can apply these definitions to a "classic" graph G by noting that G is a graph with forbidden transitions where $F_G = \emptyset$.

The *path problem* in graphs with forbidden transitions consists in, given a graph with forbidden transitions (G, F_G), and two vertices s and t, asking for the existence of an (s,t)-path.

Until the end of this section we consider only undirected graphs with symmetric forbidden transitions.

Theorem 1 ([8]). *Let $\mathcal{C}$ be a class of graphs closed under isomorphism and let $\mathcal{G}(\mathcal{C})$ be the set of graphs with symmetric forbidden transitions (G, F_G) with $\mathcal{A}_G \subseteq \mathcal{C}$. The path problem is* NP*-complete in $\mathcal{G}(\mathcal{C})$ if $\mathcal{C}^{\text{ind}}$ contains at least one of the sets $\{K_3, 2K_2\}$, $\{K_3, 2K_2\}$, $\{P_4\}$, $\{L_4\}$. In all other cases the path problem is solvable in linear time, and a path can be constructed in linear time.*

Remark 2. *One can even prove that Theorem 1 is still true if we consider bipartite graphs with symmetric forbidden transitions. Indeed, let (G, F_G) be a graph with symmetric forbidden transitions, and let $(G', F_{G'})$ where*

$$
\begin{aligned}
V_{G'} &:= V_G \cup E_G, \\
E_{G'} &:= \{\{x, \{x,y\}\} \mid \{x,y\} \in E_G\}, \\
F_{G'} &:= \{(\{\{x,y\}, y\}, \{y, \{y,z\}\}) \mid (\{x,y\}, \{y,z\}) \in F_G\}.
\end{aligned}
$$

One easily proves that $P := (x_1, \ldots, x_k)$ is a path in (G, F_G) if and only if $(x_1, \{x_1, x_2\}, x_2, \ldots, x_{k-1}, \{x_{k-1}, x_k\}, x_k)$ is a path in $(G', F_{G'})$.

An easy corollary of Theorem 1 is the following.

Corollary 3. *Let (G, F_G) be a graph with symmetric forbidden transitions such that, for every vertex x of G, the transition graph A_x is a complete graph whenever $d_G(x) \geq 4$. Then, for every two vertices x and y, one can construct in linear time a path between x and y, if one exists.*

Proof. Let x be a vertex of G. If $d_G(x) \geq 4$, then A_x is a complete graph, and each graph in the set $\{P_3,\ P_4,\ L_4,\ 2K_2\}$ is not an induced subgraph of A_x. If $d_G(x) \leq 3$, then $P_4,\ L_4$ and $2K_2$ cannot be induced subgraphs of A_x. Therefore, any of these sets $\{K_3, 2K_2\},\ \{K_3, 2K_2\},\ \{P_4\},\ \{L_4\}$ is included in $\mathcal{A}_G^{\text{ind}}$. And by Theorem 1, one can find a path, if it exists, between every two vertices of G in linear time. □

3 Exact Exponential Time Algorithm

Our algorithm will count the number of paths between two vertices using the principle of inclusion-exclusion as in [2]. Let us introduce some notations.

Let (G, F_G) be a graph (directed or not) with forbidden transitions and let s and t be two vertices of G. For every positive integer ℓ, we denote by $W_\ell(s,t)$ the set of (s,t)-walks on ℓ vertices, and by $P_\ell(s,t)$ the set of paths in $W_\ell(s,t)$. For $Y \subseteq V_G$, we denote by $W_{\ell,Y}(s,t)$ the walks in $W_\ell(s,t)$ that do not intersect Y and similarly for $P_{\ell,Y}(s,t)$.

We propose Algorithm 1 to verify the existence of an (s,t)-path.

Algorithm 1: PathProblem(G, F_G)

Data: A graph with forbidden transitions (G, F_G) and two vertices s and t.
Result: Does there exist an (s,t)-path in (G, F_G)? If yes, how many vertices the shortest path contain?

```
begin
1   | Let n be |V_G|
2   | for ℓ ← 1 to n do
3   |   | R := 0
4   |   | foreach A ⊆ V_G with |A| ≥ n − ℓ do
5   |   |   | R := R + C(|A|, n−ℓ) · (−1)^{|A|−(n−ℓ)} · |W_{ℓ,A}(s,t)|
    |   | end
6   |   | if R ≥ 1 then
7   |   |   | return (YES, ℓ)
    |   | end
    | end
8   | return NO
end
```

Theorem 4. *Algorithm 1 is correct and runs in time $O(2^n \cdot n^5 \cdot \log(n))$ for every n-vertex graph with forbidden transitions.*

The rest of this section is devoted to the proof of Theorem 4. So, we assume that we are given an n-vertex graph with forbidden transitions (G, F_G) and two vertices s and t.

Lemma 5. *Let ℓ be a fixed positive integer. Then*

$$|P_\ell(s,t)| = \sum_{\substack{Y \subseteq V_G \\ |Y|=n-\ell}} \sum_{X \subseteq V_G \setminus Y} (-1)^{|X|} \cdot |W_{\ell, Y \cup X}(s,t)|.$$

Proof. Since a path on ℓ vertices is a walk that goes through ℓ vertices and avoids $n-\ell$ other vertices, we have that

$$|P_\ell(s,t)| = \sum_{\substack{Y \subseteq V_G \\ |Y|=n-\ell}} |P_{\ell,Y}(s,t)|. \tag{1}$$

A walk on ℓ vertices that is not a path is a walk that repeats at least one vertex. Then,

$$|P_{\ell,Y}(s,t)| = |W_{\ell,Y}(s,t)| - \left| \bigcup_{x \in V_G \setminus Y} W_{\ell, Y \cup \{x\}}(s,t) \right|.$$

And, by the inclusion-exclusion principle

$$\left| \bigcup_{x \in V_G \setminus Y} W_{\ell, Y \cup \{x\}}(s,t) \right| = \sum_{\substack{X \subseteq V_G \setminus Y \\ X \neq \emptyset}} (-1)^{|X|} \cdot |W_{\ell, Y \cup X}(s,t)|.$$

Therefore,

$$|P_{\ell,Y}(s,t)| = \sum_{X \subseteq V_G \setminus Y} (-1)^{|X|} \cdot |W_{\ell, Y \cup X}(s,t)|.$$

By Eq. (1), we can conclude. □

As a corollary of Lemma 5, we get the following which proves the correctness of the algorithm.

Corollary 6. *Let ℓ be a fixed positive integer. Then,*

$$|P_\ell(s,t)| = \sum_{\substack{A \subseteq V_G \\ |A| \geq n-\ell}} \binom{|A|}{n-\ell} \cdot (-1)^{|A|-(n-\ell)} \cdot |W_{\ell,A}(s,t)|.$$

Proof. Choosing a subset Y of V_G of size $n-\ell$ and then choosing a subset of $V_G \setminus Y$, is similar to choosing a subset A of V_G of size at least $n-\ell$, and then choosing a subset Y of A of size $n-\ell$. Hence,

$$\sum_{\substack{Y \subseteq V_G \\ |Y|=n-\ell}} \sum_{X \subseteq V_G \setminus Y} (-1)^{|X|} \cdot |W_{\ell,Y \cup X}(s,t)|$$
$$= \sum_{\substack{A \subseteq V_G \\ |A| \geq n-\ell}} \binom{|A|}{n-\ell} \cdot (-1)^{|A|-(n-\ell)} \cdot |W_{\ell,A}(s,t)|.$$

By Lemma 5, we can conclude. □

It remains now to bound the time complexity of Algorithm 1.

Lemma 7. *Let ℓ be a fixed positive integer and let Y be a subset of V_G. Then one can compute $|W_{\ell,Y}(s,t)|$ in time $O(n^4 \cdot \log(n))$.*

Proof. We can assume that $\ell \geq 3$ since the statement is clear for $\ell \leq 2$. If $\{s,t\} \cap Y \neq \emptyset$, then $|W_{\ell,Y}(s,t)| = 0$. So, assume that s and t do not belong to Y. Let us use the notation $W_{\ell,Y}(s,x,t)$ to denote walks in $W_{\ell,Y}(s,t)$ having x as penultimate vertex. Then,

$$|W_{\ell,Y}(s,t)| = \sum_{x \in V_G \setminus Y} |W_{\ell,Y}(s,x,t)|.$$

If we know the multiset $\{|W_{\ell,Y}(s,x,t)| \mid x \in V_G \setminus Y\}$, then we can compute $|W_{\ell,Y}(s,t)|$ in time $O(n)$. We will prove that the multiset $\{|W_{\ell,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ can be computed in time $O(n^4 \cdot \log(n))$. If $\ell = 3$, then $|W_{\ell,Y}(s,x_0,x_1)| = 1 \Leftrightarrow (s,x_0,x_1) \in A_G$. Assume now that $\ell \geq 4$. Then

$$|W_{\ell,Y}(s,x_0,x_1)| = \sum_{\substack{y \in V_G \setminus Y \\ (y,x_0,x_1) \in A_G}} |W_{\ell-1,Y}(s,y,x_0)|. \tag{2}$$

We will compute the multiset $\{|W_{\ell,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ by dynamic programming using Eq. (2). We first compute $\{|W_{3,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ in time $O(n^2 \cdot \log(n))$ (there are $O(n^2)$ possible transitions and a search can be done in time $O(\log(n))$ if T_G is lexicographically ordered). Now, if $\{|W_{\ell-1,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ is known, one can compute $|W_{\ell,Y}(s,x_0,x_1)|$, for some pair $(x_0,x_1) \in (V_G \setminus Y)^2$, in time $O(n \cdot \log(n))$ (by using Eq. (2)), and then one can compute the multiset $\{|W_{\ell,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ from $\{|W_{\ell-1,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ in time $O(n^3 \cdot \log(n))$ and from $\{|W_{3,Y}(s,x_0,x_1)| \mid (x_0,x_1) \in (V_G \setminus Y)^2\}$ in time $O(n^4 \cdot \log(n))$. This finishes the proof. □

Proof (Proof of Theorem 4). By Corollary 6, Algorithm 1 is correct. And, by Corollary 6 and Lemma 7, the time complexity is bounded by

$$\sum_{\ell=1}^{n} \sum_{i=n-\ell}^{n} \binom{n}{i} \cdot O(n^4 \cdot \log(n)) \leq O(2^n \cdot n^5 \cdot \log(n)).$$

This concludes the proof. □

4 Generalisation of the Minimum Cut Problem

Paths and walks are related to notions of connectivity. A classical problem in graph theory is to cut a minimum number of edges to disconnect two given vertices s and t. In our context, there is another way to disconnect s and t: we can just transform some allowed transitions into forbidden ones. In this section we propose a polynomial time algorithm to find a minimum number of such transitions that must be turned into forbidden.

Let (G, F_G) be a **directed** graph with forbidden transitions, and s, t two non-adjacent vertices of G. We first define a new graph associated to (G, F_G), and s, t that we will use in the rest of this section.

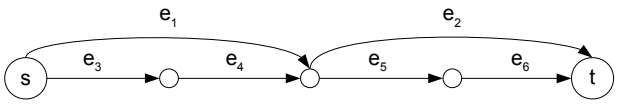
Definition 1. *We denote by $G^*_{s,t}$ the directed graph defined by:*

$$V_{G^*_{s,t}} := E_G \cup \{s'\} \cup \{t'\},$$
$$E_{G^*_{s,t}} := A_G \cup \{(s', (s, v)) \mid (s, v) \in E_G\} \cup \{((v, t), t') \mid (v, t) \in E_G\}$$

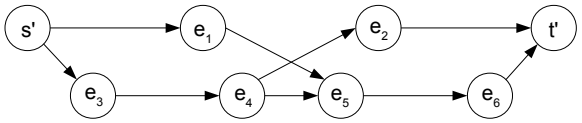
In the following we simply denote $G^*_{s,t}$ by G^*.

Remark 8. *We call the subgraph of G^* induced by E_G the* allowed line graph *of (G, F_G). It admits as vertex-set E_G and as edge-set A_G.*

For example for the following graph with forbidden transition (G, F_G) with $F_G = \{(e_1, e_2)\}$:



we obtain the following graph G^*:



The proof of the following proposition is straightforward.

Proposition 9. *The function*

$$f : \begin{cases} \{(s,t)\text{-walks in } G\} \to \{(s',t')\text{-walks in } G^*\} \\ (e_1 \in E_G, ..., e_k \in E_G) \mapsto (s', e_1 \in V_{G^*}, ..., e_k \in V_{G^*}, t') \end{cases}$$

is well defined and bijective.

So we have the following immediate corollary.

Corollary 10. *G is (s,t)-connected if and only if G^* is (s',t')-connected.*

Remark 11. *We see that f is a bijection between the shortest (s,t)-walks in G and the shortest (s',t')-walks in G^* and therefore to obtain in polynomial time a shortest (s,t)-walk in G (if it exists) we can find a shortest path in G^* (if it exists) and return its fibre under f.*

Lemma 12. *There exists an (s',t')-cut (*i.e. *a set of arcs disconnecting s' and t') in G^* having no outgoing arc of s' and no incoming arc of t'. By definition the size of a cut is its number of arcs.*

Proof. Since s and t are not adjacent in G an (s,t)-walk in G takes at least one transition. By making these transitions forbidden we obtain a new graph which is not (s,t)-connected. Hence by removing the corresponding arcs in G^* we obtain a graph which is not (s',t')-connected by Corollary 10. Finally these arcs form a cut and as they correspond to transitions of G, there is no outgoing arc of s' or incoming arc of t'. □

Lemma 13. *In G^*, the arcs of a minimum (s',t')-cut having no outgoing arc of s' and no incoming arc of t', correspond to a minimum set of allowed transitions in G sufficient to forbid to disconnect s and t, and reciprocally.*

Proof. G^* deprived of these arcs is no longer (s',t')-connected and so G deprived of the corresponding transitions is also no longer (s,t)-connected according to Corollary 10 and reciprocally. Moreover as the size of this cut is equal to the size of the set of corresponding transitions, if one is minimum, the other is also minimum. □

Theorem 14. *Let m be the number of arcs of G^*. Assign to each arc of G^* a capacity equal to one except for the outgoing arcs of s' and the incoming arcs of t' for which it is taken equal to m. The arcs of an (s',t')-cut of minimum capacity correspond to a minimum set of allowed transitions in G sufficient to forbid to disconnect s and t.*

Proof. According to Lemma 12, there exists an (s',t')-cut in G^* having no outgoing arc of s' and no incoming arc of t'. This cut has less than m arcs (each having a capacity equal to one) and thus admits a capacity less than m. Thus, the minimum capacity of an (s',t')-cut is strictly less than m, and therefore an (s',t')-cut of minimum capacity contains no outgoing arc of s' or incoming arc of t' because its capacity would be greater than or equal to m. Now for such a cut, the capacity is equal to its size and if its size is minimum, it corresponds to a minimum set of allowed transitions in G sufficient to forbid to disconnect s and t by Lemma 13. □

We are now able to give an algorithm that takes as input a graph with forbidden transitions (G, F_G), and $s,t \in V_G$ two non-adjacent vertices, and returns a minimum set of allowed transitions sufficient to forbid to disconnect s and t.

Algorithm 2: GeneralisedCutProblem(G, F_G)

Data: A directed graph with forbidden transitions (G, F_G), and two non-adjacent vertices s and t.

Result: A minimum set of allowed transitions sufficient to forbid to disconnect s and t.

begin

1 Construct the graph G^*;

2 Assign to each arc of G^* a capacity equal to one except for the outgoing arcs of s' and the incoming arcs of t' for which it is taken equal to $|E_G|$;

3 Compute an (s', t')-cut of minimum capacity in G^* with these capacities;

4 **return** *Transitions of G corresponding to the arcs of the cut.*

end

Theorem 15. *Algorithm 2 is correct and runs in time polynomial in the size of G.*

Proof. The correctness of Algorithm 2 follows from Theorem 14. For the time complexity, the steps 1, 2, and 4 run in time $O(|E_G|^2)$ and step 3 can be done using a polynomial time algorithm which computes an (s', t')-cut of minimum capacity in G^* as the Edmonds-Karp algorithm. □

Now if (G, F_G) is an **undirected** graph with forbidden transitions and s, t two non-adjacent vertices of V_G, we introduce the directed graph with forbidden transitions G_d defined formally as follows:

$$\begin{aligned} V_{G_d} &:= V_G, \\ E_{G_d} &:= \{(v, w), (w, v) \mid \{v, w\} \in E_G\}, \\ F_{G_d} &:= \{((v, w), (w, x)) \mid (\{v, w\}, \{w, x\}) \in F_G\}. \end{aligned}$$

As G and G_d have the same walks (sequences of vertices) just apply the algorithm 2 to G_d to obtain a minimum set of allowed transitions sufficient to forbid to disconnect s and t in G.

Remark 16. *To obtain (if it exists) in polynomial time a shortest (s, t)-walk in an undirected graph with forbidden transitions G we can find (if it exists) a shortest path in G_d^* and return its fibre under f.*

5 Conclusion and perspectives

Including forbidden transitions into graphs strongly extend the capacities to model real situations but also increases the complexity of *a priori* simple tasks such as finding elementary (and shortest) paths. When the total number k of forbidden transitions in G is low one can apply a simple branch and bound algorithm to find (if there is one) such a path. Up to $k = O(\log n)$, this elementary algorithm remains polynomial. However, as k can be much larger than n, we proposed in Section 3 another method with time complexity $O(2^n \cdot poly(n))$.

In Section 4 we transposed a classical cutting problem: instead of cutting a minimum number of edges to disconnect two given vertices s and t, we proposed a polynomial time algorithm to turn a minimum number of allowed transitions into forbidden ones in such a way that there is no more walk between s and t. This is another measure of connectivity between s and t and can be used for example in preliminary studies to prevent to disconnect two parts of a city when there are temporary traffic restrictions due for example to a punctual event or due to works in streets.

In conclusion, we can see graphs with forbidden transitions as graphs with "reduced connection capabilities". Due to the potential applications, we plan to further investigate this subject. We hope to decrease the $O(2^n \cdot poly(n))$ complexity of the exact algorithm. Our result on the equivalent cutting problem in Section 4 motivates us to try to generalise the flow problem in graphs with forbidden transitions. In a more general way we intend to generalise classical algorithmic problems into graphs with forbidden transitions.

References

1. Mustaq Ahmed and Anna Lubiw. Shortest paths avoiding forbidden subpaths. In Susanne Albers and Jean-Yves Marion, editors, *STACS*, volume 3 of *LIPIcs*, pages 63–74. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
2. Eric T. Bax. Inclusion and exclusion algorithm for the hamiltonian path problem. *Inf. Process. Lett.*, 47(4):203–207, 1993.
3. Jacek Błażewicz and Marta Kasprzak. Computational complexity of isothermic dna sequencing by hybridization. *Discrete Applied Mathematics*, 154(5):718–729, 2006.
4. Jacek Błażewicz, Marta Kasprzak, Benjamin Leroy-Beaulieu, and Dominique de Werra. Finding hamiltonian circuits in quasi-adjoint graphs. *Discrete Applied Mathematics*, 156(13):2573–2580, 2008.
5. J.A. Bondy and U.S.R. Murty. *Graph Theory*. Springer London Ltd, 2010.
6. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.
7. Zdeněk Dvořák. Two-factors in orientated graphs with forbidden transitions. *Discrete Mathematics*, 309(1):104–112, 2009.
8. Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2-3):261–273, 2003.
9. Daniel Villeneuve and Guy Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107, 2005.