Tradeoffs in process strategy games with application in the WDM reconfiguration problem

Nathann Cohen
Mascotte, INRIA, I3S, CNRS, Université de Nice Sophia
Sophia Antipolis, France
nathann.cohen@inria.fr

David Coudert
Mascotte, INRIA, I3S, CNRS, Université de Nice Sophia
Sophia Antipolis, France
david.coudert@inria.fr

Dorian Mazauric
Mascotte, INRIA, I3S, CNRS, Université de Nice Sophia
Sophia Antipolis, France
dorian.mazauric@inria.fr

Napoleão Nepomuceno
Institut for Matematik og Datalogi, Syddansk Universitet
Campusvej 55, DK-5230 Odense, Denmark
napoleao@imada.sdu.dk

Nicolas Nisse
Mascotte, INRIA, I3S, CNRS, Université de Nice Sophia
Sophia Antipolis, France
nicolas.nisse@inria.fr

Abstract

We consider a variant of the graph searching games that models the routing reconfiguration problem in WDM networks. In the digraph processing game, a team of agents aims at *processing*, or clearing, the vertices of a digraph D. We are interested in two different measures: 1) the total number of agents used, and 2) the total number of vertices occupied by an agent during the processing of D. These measures respectively correspond to the maximum number of simultaneous connections interrupted and to the total number of interruptions during a routing reconfiguration in a WDM network.

Previous works have studied the problem of independently minimizing each of these parameters. In particular, the corresponding minimization problems are APX-hard, and the first one is known not to be in APX. In this paper, we give several complexity results and study tradeoffs between these conflicting objectives. In particular, we show that minimizing one of these parameters while the other is constrained is NP-complete. Then, we prove that there exist some digraphs for which minimizing one of these objectives arbitrarily impairs the quality of the solution for the other one. We show that such bad tradeoffs may happen even for a basic class of digraphs. On the other hand, we exhibit classes of graphs for which good tradeoffs can be achieved. We finally detail the relationship between this game and the routing reconfiguration problem. In particular, we prove that any instance of the processing game, i.e. any digraph, corresponds to an instance of the routing reconfiguration problem.

Keywords: graph searching games; process number; routing reconfiguration problem

1 Introduction

In this paper, we study the *digraph processing* game, analogous to graph searching games [12]. This game aims at *processing*, or clearing, the vertices of a contaminated directed graph D. For this, we use a set of agents which are sequentially put and removed from the vertices of D. We are interested in two different measures and their tradeoffs: the minimum number of agents required to *clear* D and the minimum number of vertices that must be *covered* by an agent. The digraph processing game has been introduced in [6] for its relationship with the routing reconfiguration problem in Wavelength Division Multiplexing (WDM) networks. In this context, the goal is to reroute some connections that are established between pairs of nodes in a communication network, which can lead to interruptions of service. Each instance of this problem may be represented by a directed graph, called its *dependency digraph*, such that the reconfiguration problem is equivalent to the clearing of the dependency digraph. More precisely, the two measures presented above respectively correspond to the maximum number of simultaneous disruptions, and to the total number of requests disrupted during the rerouting of the connections. The equivalence between these two problems is detailed in Section 5.

The digraph processing game is defined by the three following operations (or rules), which are very similar to the ones defining the *node search number* [2, 9, 12, 15, 17] of a graph, and whose goal is to *process*, or to clear, all the vertices of a digraph D.

- R_1 Put an agent at a vertex v of D;
- R_2 Remove an agent from a vertex v of D if all its outneighbors are either processed or occupied by an agent, and process v;
- R_3 Process an unoccupied vertex v of D if all its outneighbors are either processed or occupied by an agent.

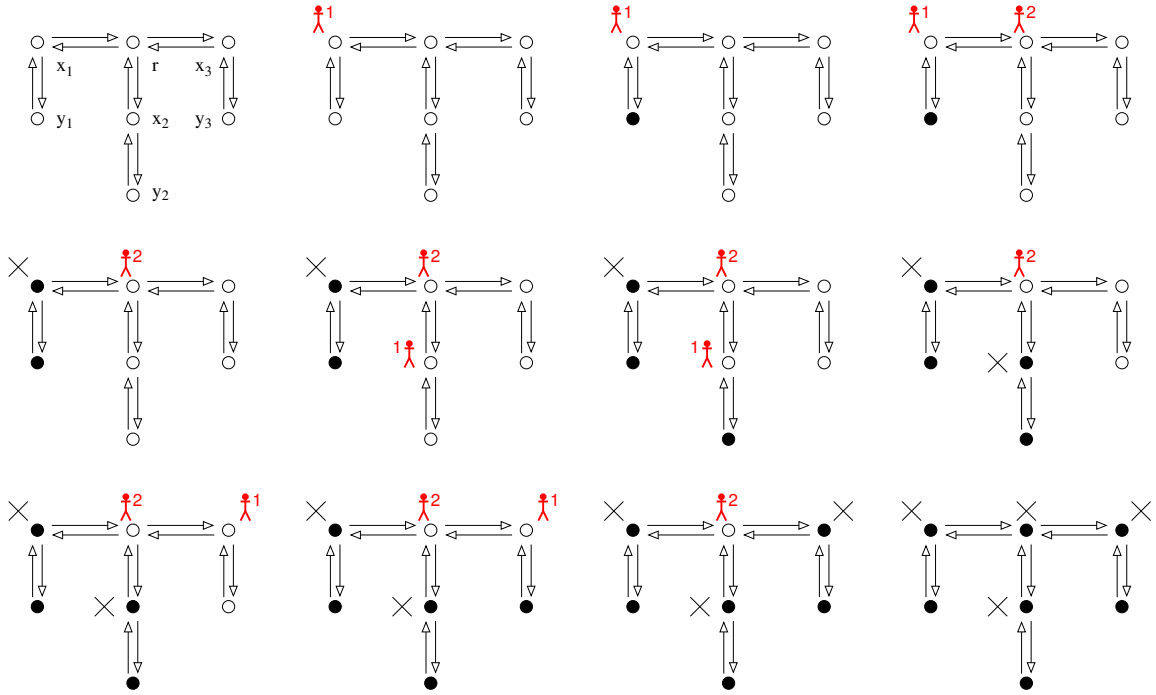
A digraph whose vertices have all been processed is said *processed*. A sequence of such operations resulting in processing all vertices of D is called a *process strategy*. Note that, during a process strategy, an agent that has been removed from a (processed) vertex can be reused. The number of agents used by a strategy on a digraph D is the maximum number of agents present at the same time in D during the process strategy. A vertex is *covered* during a strategy if it is occupied by an agent at some step of the process strategy.

Figure 1 illustrates two process strategies for a symmetric digraph D of 7 vertices. The strategy depicted in Figure 1(a) first puts an agent at vertex x_1 (rule R_1), which let y_1 (rule R_3) be processed. A second agent is then put at r (rule R_1) allowing the vertex x_1 to be processed, and the agent on it to be removed (rule R_2). The procedure goes on iteratively, until all the vertices are processed. The depicted strategy uses 2 agents and covers 4 vertices. Another process strategy is depicted in Figure 1(b) that uses 3 agents and covers 3 vertices. Note that this latter strategy consists in putting agents at the vertices of a feedback vertex set[1] of minimum size.

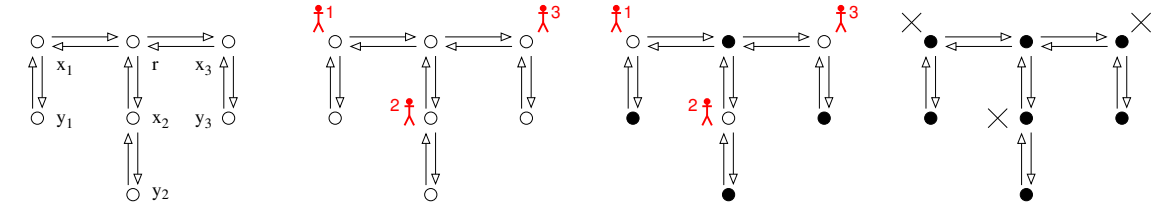
Clearly, to process a digraph D, it is sufficient to put an agent at every vertex of a feedback vertex set F of D (rule R_1), then the vertices of $V(D) \setminus F$ can be sequentially processed using rule R_3, and finally the vertices of F can be processed and all agents can be removed (rule R_2). In particular, a Directed Acyclic Graph (DAG) can be processed using 0 agent and thus covering no vertices. Indeed, to process a DAG, it is sufficient to process sequentially its vertices starting from the leaves (rule R_3). Note that any process strategy for a digraph D must cover all the vertices of a feedback vertex set of D (not necessarily simultaneously). Indeed, otherwise there exists a cycle such that none of its vertices are covered during the strategy. Therefore its vertices cannot be processed since neither rule R_1 nor rule R_3 can be applied. Obviously, for any process strategy, the number of covered vertices is always at least the number of agents used.

The minimum number of agents required to process a digraph D (without constraint on the number of covered vertices) is called the *process number* [5–7], while the minimum number of covered vertices required to process D (without constraint on the number of agents) equals the size of a *minimum feedback vertex set* (MFVS) of D. In this paper, we are interested in tradeoffs between the minimum number of agents used by a process strategy and the minimum number of vertices it covers.

[1] A set F of nodes of D is a feedback vertex set if the removal of all nodes in F makes D acyclic.



(a) A $(2, 4)$-process strategy for D.



(b) A $(3, 3)$-process strategy for D.

Figure 1: Different process strategies for a symmetric digraph D.

Note that an empirical study of this tradeoff has been conducted in [21] using a heuristic algorithm designed for determining process strategies with minimum number of agents. The conclusion of this empirical study is non surprisingly that the number of covered vertices could be far from the MFVS.

1.1 Definitions and Previous Results

Let D be a n-node directed graph. In the following, a (p, q)-*process strategy* for D denotes a process strategy for D using at most p agents and covering at most q vertices. When the number of covered vertices is not constrained, we write (p, ∞)-process strategy. Similarly, when the number of agents is not constrained, we write (∞, q)-process strategy.

Process Number The problem of finding the *process number* of a digraph D, was introduced in [6] as a metric of the routing reconfiguration problem (see Section 5). Formally,

Definition 1. *The process number of D, denoted by $pn(D)$, is the smallest p such that there exists a (p, ∞)-process strategy for D.*

For instance, the digraph D of Figure 1 satisfies $pn(D) = 2$. Indeed, Figure 1(a) describes a process strategy using 2 agents, and it is easy to check that there is no process strategy using at most 1 agent. Digraphs whose process number is equal to 0 or 1 can easily be identified, as they respectively correspond to acyclic digraphs, and to graphs whose strongly connected components have a feedback vertex set of size at most 1 (which can be checked in linear time [7]). In [7] is also given an polynomial algorithm to recognize digraphs whose process number is equal to 2. However the problem of computing the process number of general digraphs is NP-complete and not in APX (i.e., admitting no polynomial-time approximation algorithm up to a constant factor, unless $P = NP$) [6]. A distributed polynomial-time algorithm to compute the process number of trees (or forests) with symmetric arcs has been proposed in [4]. Furthermore, general heuristics to compute the process number of a digraph are described in [5,21]. In [19], Solano writes that "near-optimal solutions can be quickly found in polynomial time" when computing the process number if the set of covered vertices is given as part of the input. We show that computing the process number of a digraph remains not in APX (and so is NP-complete) in this situation (see Theorem 1), and that the gap with the process number could be arbitrarily large. When considering symmetric digraphs, which can be thought of as a directed version of an undirected graph, one notices that the process number is closely related to two other graph invariants, the *node search number* and the *pathwidth*. The node search number of a graph G, denoted by $sn(G)$, is the smallest p such that rules R_1 and R_2 (R_3 is omitted) are sufficient to process G using at most p agents. See [2, 9, 12, 15, 17] for more details. The pathwidth of a (undirected) graph G, denoted by $pw(G)$, was introduced by Robertson and Seymour in [18]. It has been proved in [10] by Ellis *et al.* that the pathwidth and the node search number are equivalent, that is for any graph G, $pw(G) = sn(G) - 1$. The relationship between these parameters and the process number has been described in [6]: $pw(G) \leq pn(G) \leq pw(G) + 1$ (and so $sn(G)-1 \leq pn(G) \leq sn(G)$), where $pn(G)$ is the process number of the digraph built from G by replacing each edge by two opposite arcs. Since computing the pathwidth of a graph is NP-complete [16] and not in APX [8], determining these parameters is as hard.

Minimum Feedback Vertex Set Given a digraph D, the problem of finding a process strategy that minimizes the number of nodes covered by agents is equivalent to the one of computing a *minimum feedback vertex set* (MFVS) of D. Computing such a set is well known to be NP-complete and APX-hard [14]. A 2-approximation algorithm is known in undirected graphs [1] and in symmetric digraph (where a feedback vertex set is a vertex cover of the underlying graph). As far as we know, the best approximation algorithm for computing a MFVS in general n-node digraphs has ratio $\log n \log\log n$ [11].

We define below the parameter $mfvs(D)$, using the notion of (p, q)-process strategy, corresponding to the size of a MFVS of D.

Definition 2. *Let $mfvs(D)$ denote the smallest q such that there exists a (∞, q)-process strategy for D.*

As an example, the digraph D of Figure 1 satisfies $mfvs(D) = 3$. Indeed for $i \in \{1, 2, 3\}$, it is easy to see that either x_i or y_i must be in any feedback vertex set (FVS) of D because of the cycle (x_i, y_i, x_i). Furthermore the removal of x_1, x_2, and x_3 from D is sufficient to break all the cycles. Thus these three nodes form a MFVS of D, and so $mfvs(D) = 3$. The corresponding strategy, covering $mfvs(D) = 3$ nodes by agents, is described in Figure 1(b).

As mentioned above, $mfvs(D) \geq pn(D)$. Moreover, the gap between these two parameters may be arbitrarily large. For example consider a symmetric path P_n composed of $n \geq 4$ nodes $u_1, u_2, \ldots, u_n$ with symmetric arcs between u_i and u_{i+1} for $i = 1, \ldots, n-1$. We get $mfvs(P_n) = \lfloor \frac{n}{2} \rfloor$ while $pn(P_n) = 2$. Indeed either u_i or u_{i+1} must be in any FVS of P_n, and so we deduce that nodes $u_2, u_4, u_6, \ldots$ form a MFVS of P_n. Furthermore $pn(P_n) \geq 2$ because P_n is strongly connected and $mfvs(P_n) > 1$. We then describe a process strategy for P_n using 2 agents: we put the first agent at u_1 (R_1), we put the second agent at u_2 (R_1), we process u_1 removing the agent from it (R_2), we put this agent at u_3 (R_1), we process u_2 removing the agent from it (R_2), we put an agent at u_4 (R_1), and so on.

Remark that this process strategy for P_n uses the optimal number of agents, $pn(D) = 2$, but all the n nodes are covered by an agent at some step of the process strategy. For this digraph P_n, it is possible to describe a $(pn(D) = 2, mfvs(D) = \lfloor \frac{n}{2} \rfloor)$-process strategy, that is a process strategy for P_n minimizing both the number of agents and the total number of covered nodes. We put the first agent at u_2 (R_1), we process u_1 (R_3), we put the second agent at u_4 (R_1), we process u_3 (R_3), we process u_2 removing the agent from it (R_2), we put this agent at u_6 (R_1), and so on. Unfortunately such good tradeoffs are not

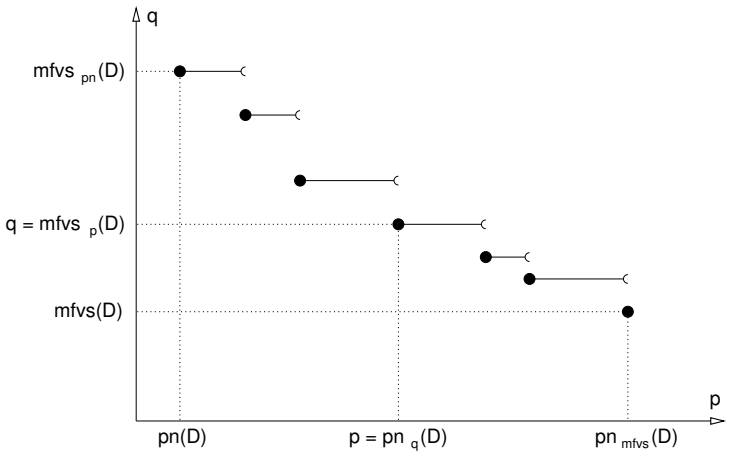


Figure 2: $mfvs_p(D)$ function of p for a digraph D. Filled circles represent minimal values of D.

always possible (it is the case for the digraph of Figure 1 as explained later). Actually, we prove in this paper that there exist some digraphs for which minimizing one of these objectives arbitrarily impairs the quality of the solution for the other one. In the following, we define formally the tradeoff metrics we will now study.

Tradeoff Metrics We introduce new tradeoff metrics in order to study the loss one may expect on one parameter when adding a constraint on the other. In particular, what is the minimum number of vertices that must be covered by a process strategy for D using $pn(D)$ agents? Similarly, what is the minimum number of agents that must be used to process D while covering $mfvs(D)$ vertices?

Definition 3. *Given an integer $q \geq mfvs(D)$, we denote by $pn_q(D)$ the minimum p such that a (p,q)-process strategy for D exists. We write $pn_{mfvs+r}(D)$ instead of $pn_{mfvs(D)+r}(D)$, $r \geq 0$.*

Definition 4. *Given an integer $p \geq pn(D)$, we denote by $mfvs_p(D)$ the minimum q such that a (p,q)-process strategy for D exists. We write $mfvs_{pn+r}(D)$ instead of $mfvs_{pn(D)+r}(D)$, $r \geq 0$.*

Intuitively $pn_{mfvs}(D)$ is the minimum number of agents required by a process strategy minimizing the number of covered vertices, and $mfvs_{pn}(D)$ is the minimum number of vertices that must be covered by a process strategy using the minimum number of agents. Note that, $pn_{mfvs}(D)$ is upper bounded by the maximum MFVS of the strongly connected components of D. Another straightforward remark is that $mfvs_{mfvs}(D) = mfvs(D)$ for any digraph D.

To illustrate the pertinence of these tradeoff metrics, consider the digraph D of Figure 1. Recall that $pn(D) = 2$ and $mfvs(D) = 3$. We can easily verify that there does not exist a $(2,3)$-process strategy for D, that is a process strategy minimizing both p and q. On the other hand, we can exhibit a $(2,4)$-process strategy (Figure 1(a)) and a $(3,3)$-process strategy (Figure 1(b)) for D. Hence, we have: $pn_{mfvs}(D) = 3$ while $pn(D) = 2$, and $mfvs_{pn}(D) = 4$ while $mfvs(D) = 3$. Intuitively for these two process strategies, we can not decrease the value of one parameter without increasing the other.

We generalize this concept through the notion of *minimal values* of a digraph D. We say that (p,q) is a minimal value of D if $p = pn_q(D)$ and $q = mfvs_p(D)$. Note that $(pn(D), mfvs_{pn}(D))$ and $(pn_{mfvs}(D), mfvs(D))$ are both minimal values by definition (and may be the same). For the digraph of Figure 1, there are two minimal values: $(2,4)$ and $(3,3)$. Figure 2 depicts the variations of the minimum number q of vertices covered by a p-strategy for a digraph D ($p \geq pn(D)$), i.e., $mfvs_p(D)$ as a function of p. Clearly, it is a non-increasing function upper bounded by $mfvs_{pn}(D)$ and lower bounded by $mfvs(D)$.

Filled circles of Figure 2 represent the shape of minimal values of D. Clearly for a given digraph D, the number of minimal values is at most linear in the number of nodes. We now give an example of a family of n-node digraph for which the number of minimal value is $\Omega(\sqrt{n})$. Intuitively, it means that, in those digraphs D, starting from the optimal number of agents $pn(D)$, each extra agent added allows to strictly decrease the number of covered vertices, until the optimal, $mfvs(D)$, is reached. Let

H_n be the symmetric directed star with $n \geq 3$ branches of length 2 (for instance, H_3 is the digraph of Figure 1), and let G_k be the graph that consists of the disjoint union of $H_3, \cdots, H_k$, $k \geq 3$. Then, for any $0 \leq i \leq k-2$, $(pn(G_k)+i, mfvs(G_k)+k-2-i) = (2+i, (k(k+1)/2)-5+k-i)$ are minimal values (this can be easily proved using the easy results described in Section 2.1).

1.2 Our Results

Our results consist in an analysis of the behaviour of the two given tradeoff measures both in general digraphs and in symmetric digraphs. As mentioned above, in general, no process strategy minimizes both the number of agents and the number of covered vertices (see example in Figure 1). Hence, we are interested in the loss on one measure when the other is constrained. In particular, we are interested in the ratios $\frac{pn_{mfvs}(D)}{pn(D)}$ and $\frac{mfvs_{pn}(D)}{mfvs(D)}$. This study involves various theorems on the complexity of estimating this loss (Section 2) and the existence of digraphs for which it can be arbitrarily large (Section 3). We also study in Section 4 the case of symmetric digraphs. Finally we describe in Section 5 the relation between the routing reconfiguration problem and the processing game.

More precisely, we first prove that computing the process number of a digraph is not in APX (and thus NP-complete), even when the subset of vertices of the digraph at which an agent will be put is given (Theorem 1). Then, we prove that for all $\alpha, \beta \geq 0$, the problems of determining the parameters $\alpha \cdot pn_{mfvs}(D) + \beta \cdot pn(D)$ and $\alpha \cdot mfvs_{pn}(D) + \beta \cdot mfvs(D)$ are NP-complete (Theorem 2). In particular, the problem of determining $pn_{mfvs}(D)$ is not in APX and the problem of determining $mfvs_{pn}(D)$ is APX-hard (Theorem 2). Then, we prove that for any $q \geq 0$ (resp. for any $p \geq 0$), the ratio $\frac{pn_{mfvs+q}(D)}{pn(D)}$ (resp. $\frac{mfvs_{pn+p}(D)}{mfvs(D)}$) is not bounded even in the class of bounded process number digraphs (Theorem 3 and Theorem 4). However we prove that $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq pn(D)$ for any symmetric digraph D (Lemma 5).

In Section 5, we detail the relationship between the processing game and the reconfiguration routing problem. In this context, any instance of the routing reconfiguration problem may be represented by a directed graph, called the dependency digraph of this instance, such that the routing reconfiguration problem is equivalent to the processing of this digraph. We prove the reverse, that is, any digraph is the dependency digraph of an instance of the reconfiguration problem (Theorem 7).

2 Complexity Results

This section is devoted to the study of the complexity of the problems related to the parameters introduced in Section 1.1. First, we need to define some digraphs.

2.1 Definition of some useful digraphs.

Let H_n be a symmetric directed star with $n \geq 3$ branches each of which contains two vertices (the root r being at distance 2 from any leaf), with a total of $2n+1$ vertices. H_3 is represented in Figure 1. It is easy to check that $pn(H_n) = 2$. Indeed 1 agent is obviously not sufficient and there exists a $(2, n+1)$-process strategy for H_n: an agent is put at the central node r, then we successively put an agent at a vertex x adjacent to r, the remaining neighbor of x (different from r) is processed, and we process x itself relieving the agent on it. Then, the same process is applied until all vertices adjacent to r are processed, and finally we process r. Figure 1(a) represents a $(2,4)$-process strategy for H_3. Moreover, the single MFVS of H_n is the set X of the n vertices adjacent to r. It is easy to check that the single process strategy occupying only the vertices of X consists in putting n agents at all vertices of X. No agent can be removed while all agents have not been put. Thus this strategy is a (n,n)-process strategy, and $pn_{mfvs}(H_n) = n$. See Figure 1(b) for such a process strategy for H_3. To summarize, the two minimal values of H_n are $(pn(H_n), mfvs_{pn}(H_n)) = (2, n+1)$ and $(pn_{mfvs}(H_n), mfvs(H_n)) = (n,n)$.

Let K_n be a symmetric complete digraph of n nodes. It is easy to check that the unique minimal value of K_n is $(pn(K_n), mfvs(K_n)) = (n-1, n-1)$.

Let $D = (V, A)$ be a symmetric digraph with $V = \{u_1, \ldots, u_n\}$. Let $\hat{D} = (V', A')$ be the symmetric digraph where $V' = V \cup \{v_1, \ldots, v_n\}$, and $\hat{D}$ is obtained from D by adding two symmetric arcs between u_i and v_i for $i = 1, \ldots, n$. It is easy to show that there exists an optimal process strategy for $\hat{D}$ such

that the set of occupied vertices is V. Indeed, note that, for all i, at least one of u_i or v_i must be covered by an agent (any FVS of $\hat{D}$ contains at least one of v_i or u_i). Furthermore if some step of a process strategy for $\hat{D}$ consists in putting an agent at some vertex v_i, then the process strategy can be easily transformed by putting an agent at u_i instead. In particular, $mfvs_{pn}(\hat{D}) = n$.

2.2 NP-completeness.

Before proving that computing the tradeoff parameters introduced in Section 1.1 are NP-complete, we prove the hardness of computing the process number even if the subset of vertices that must be covered is given.

Indeed a possible approach for computing the process number, proposed by Solano in [19], consists of the following two phases: 1) finding the subset of vertices of the digraph at which an agent will be put, and 2) deciding the order in which the agents will be put at these vertices. Solano then said that the difficulty of the problem mainly lies in finding the accurate subset of vertices that must be covered. We show that the second phase is also NP-complete and not in APX.

Theorem 1. *Computing the process number of a digraph is not in APX (and thus NP-complete), even when the subset of vertices of the digraph at which an agent will be put is given.*

Proof. Let D be any symmetric digraph. Let us consider the problem of computing an optimal process strategy for $\hat{D}$ when the set of vertices covered by agents is constrained to be V. By the remark in Section 2.1, such an optimal strategy always exists. It is easy to check that this problem is equivalent to the one of computing the node search number (and so the pathwidth) of the underlying undirected graph of D which is NP-complete [16] and not in APX [8]. □

Theorem 2. *Let $\alpha, \beta \geq 0$ be fixed, with $\max\{\alpha, \beta\} > 0$. The problem that takes a digraph D as an input and that aims at determining:*

- $\alpha \cdot pn_{mfvs}(D) + \beta \cdot pn(D)$ *is not in APX,*
- $\alpha \cdot mfvs_{pn}(D) + \beta \cdot mfvs(D)$ *is APX-hard.*

Proof. The two cases for $\alpha = 0$ and $\beta > 0$ clearly holds from the literature. Now, let us assume $\alpha > 0$.

- We start with $\alpha \cdot pn_{mfvs}(D) + \beta \cdot pn(D)$.

  Let us first consider the case $\beta = 0$. That is, let us show that the problem of determining pn_{mfvs} is not in APX. Indeed, let $\mathcal{D}$ be the class of all digraphs $\hat{D}$ obtained from some symmetric digraph D. For any symmetric digraph D, the problem of computing $pw(D)$ (where $pw(D)$ is the pathwidth of the underlying graph of the symmetric digraph of D) is not in APX, and $pn(\hat{D}) = pn_{mfvs}(\hat{D}) = pw(D) + 1$ (see Theorem 1). Hence, the problem of determining pn_{mfvs} is not in APX.

  Assume now that $\beta > 0$. To prove that determining $\alpha \cdot pn_{mfvs}(D) + \beta \cdot pn(D)$ is not in APX, let D_1 be the disjoint union of H_n and any n-node digraph D. First, let us note that $pn_{mfvs}(D_1) = pn_{mfvs}(H_n)$ because $pn_{mfvs}(D) \leq n - 1$ and $pn_{mfvs}(H_n) = n$. Note that the process number of any digraph is the maximum for the process numbers of its strongly connected components, thus $pn(D_1) = \max\{pn(D), pn(H_n)\}$. Therefore, since $pn(H_n) = 2$, we get that $\alpha \cdot pn_{mfvs}(D_1) + \beta \cdot pn(D_1) = \alpha \cdot n + \beta \cdot \max\{pn(D), 2\}$. So, the NP-completeness comes from the NP-completeness of the process number problem.

- We now consider $\alpha \cdot mfvs_{pn}(D) + \beta \cdot mfvs(D)$.

  When $\beta = 0$, let us prove that the problem of determining $mfvs_{pn}$ is APX-hard. Let D_2 be the disjoint union of K_n and any n-node digraph D. First let us note that $pn(D_2) = \max\{pn(K_n), pn(D)\}$ because the process number of any digraph is the maximum for the process numbers of its strongly connected components. It is easy to show that $pn(D_2) = pn(K_n) = n - 1$ because $pn(D) \leq n - 1$. Hence, when D must be processed, $n-1$ agents are available. So, in order to minimize the number of nodes covered by agents, the agents must be placed on a MFVS of D. Thus $mfvs_{pn}(D_2) = n - 1 + mfvs(D)$, and the result follows because computing $mfvs(D)$ is APX-hard.

  Assume now that $\beta > 0$. To prove that determining $\alpha \cdot mfvs_{pn}(D) + \beta \cdot mfvs(D)$ is APX-hard, let D_3 be the disjoint union of K_n, H_n, and D. Again, $pn(D_3) = \max\{pn(K_n), pn(H_n), pn(D)\}$. It is

easy to show that $pn(D_3) = pn(K_n) = n-1$ because $pn(H_n) = 2$ and $pn(D) \leq n-1$. Moreover, any process strategy of D_3 using $n-1$ agents must cover $n-1$ nodes of K_n, $n+1$ nodes of H_n ($mfvs(H_n) = n$ but one extra agent is needed to cover only n nodes), and $mfvs(D)$ nodes of D (because $n-1$ agents are available and $mfvs(D) \leq n-1$). Hence, $mfvs_{pn}(D_3) = (n-1) + (n+1) + mfvs(D)$. Furthermore $mfvs(D_3) = (n-1) + n + mfvs(D)$ because $mfvs(K_n) = n-1$ and $mfvs(H_n) = n$. Thus $\alpha \cdot mfvs_{pn}(D_3) + \beta \cdot mfvs(D_3) = (\alpha+\beta)(mfvs(D) + 2n) - \beta$. The result follows the APX-hardness of the MFVS problem.

□

Corollary 1. *For an input digraph D and two integers $p \geq 0$ and $q \geq 0$, and any $\alpha, \beta \geq 0$ ($\{\alpha, \beta\} \neq \{0, 0\}$) the problems of determining:*

- *$\alpha.pn_{mfvs+q}(D) + \beta.pn(D)$ are not in APX,*
- *$\alpha.mfvs_{pn+p}(D) + \beta.mfvs(D)$ are APX-hard.*

3 Behaviour of ratios in general digraphs

In this section, we study the behaviours of parameters introduced in Section 1.1 and their ratios, showing that, in general, good tradeoffs are impossible.

Theorem 3. *For any $C > 0$ and any integer $q \geq 0$, there exists a digraph D such that $\frac{pn_{mfvs+q}(D)}{pn(D)} > C$.*

Proof. Consider the symmetric directed star H_n defined in Section 2.1. Let now D be the digraph consisting of $q+1$ pairwise disjoint copies of H_n. So D has $q+1$ strongly connected components. We get $mfvs(D) = (q+1)n$. By definition, any $(pn_{mvfs+q}(D), mfvs(D)+q)$-process strategy for D covers at most $q(n+1)+n$ nodes. Therefore, there exists at least one of the $q+1$ strongly connected components for which at most n nodes must be covered. Hence, the corresponding connected component requires at least n agents to be processed, and actually n agents are sufficient because (n, n) is a minimal value of H_n (see Section 2.1). Hence, $pn_{mfvs+q}(D) = n$ while $pn(D) = 2$. Taking $n > 2C$, we get $\frac{pn_{mfvs+q}(D)}{pn(D)} > C$. □

Note that if it is allowed to cover $mfvs(D) + q + 1$ nodes during the process strategy (instead of $mfvs(D)+q$), then the number of agents required is $pn(D)$. In other words, for the digraph D described in the proof of Theorem 3, we get $\frac{pn_{mfvs+q+1}(D)}{pn(D)} = 1$ while $\frac{pn_{mfvs+q}(D)}{pn(D)} = \frac{n}{2}$.

Corollary 2. *For any $C > 0$, there exists a digraph D such that $\frac{pn_{mfvs}(D)}{pn(D)} > C$.*

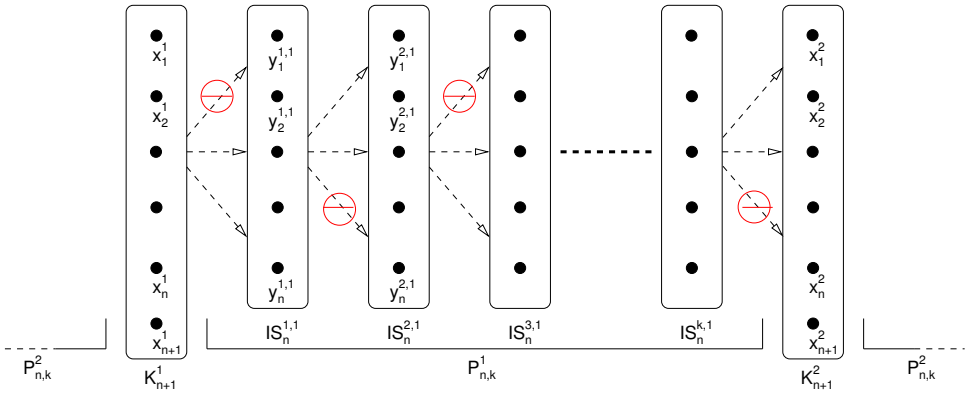
In the sequel, we present similar results for the second ratio.

Theorem 4. *For any $C > 0$ and any integer $p \geq 0$, there exists a digraph D such that $\frac{mfvs_{pn+p}(D)}{mfvs(D)} > C$.*

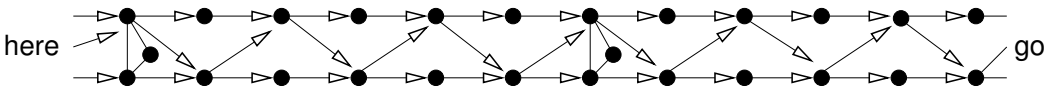
Proof. Let $n \geq 2$ and let $k \geq 1$ be an odd integer. Let us consider the digraph $D_{n,k}$ built as follows. Let $IS_n^1, \cdots, IS_n^k$ be k independent sets, each IS_n^t ($1 \leq t \leq k$) having n vertices: $y_1^t, y_2^t, \ldots, y_n^t$. Let $P_{n,k}$ be the digraph obtained from the k independent sets IS_n^t ($1 \leq t \leq k$) by adding the arcs from y_i^t to y_j^{t+1}, for $1 \leq j \leq i \leq n$ and $t = 1, 3, \ldots, k-2$, and from y_i^t to y_j^{t+1}, for $1 \leq i \leq j \leq n$ and $t = 2, 4, \ldots, k-1$. Let K_{n+1} be the symmetric clique with $n+1$ nodes: $x_1, x_2, \ldots, x_{n+1}$.

The digraph $D_{n,k}$ is obtained from two copies $P_{n,k}^1, P_{n,k}^2$ of $P_{n,k}$ and two copies K_{n+1}^1, K_{n+1}^2 of K_{n+1}, by adding the following arcs. In what follows, $y_j^{t,a}$ denotes the j^{th} vertex in the t^{th} independent set of $P_{n,k}^a$, where $j \leq n$, $t \in \{1, k\}$, $a \in \{1, 2\}$, and x_j^a denotes the j^{th} vertex of K_n^a, where $j \leq n+1$, $a \in \{1, 2\}$. There are arcs from x_i^a to $y_j^{1,a}$, for $1 \leq i \leq j \leq n$ and $a = 1, 2$, and from $y_i^{k,a}$ to x_j^b, for $1 \leq i \leq j \leq n$, $a = 1, 2$ and $b = 3 - a$. Finally there is an arc from each node of $V(D_{n,k}) \setminus V(K_{n+1}^1)$ to each node of $V(K_{n+1}^1)$. Note that these last arcs are not needed to obtain the results but help to make the proof less technical.

Figure 3(a) shows the general shape of $D_{n,k}$, where the red symbol $\ominus$ represents the inexistence of arcs between these subgraphs. $D_{2,5}$ is depicted in Figure 3(b). For not overloading the figures, the arcs from $V(D_{n,k}) \setminus V(K_{n+1}^1)$ to $V(K_{n+1}^1)$ are not represented.



(a) $D_{n,k}$ of Theorem 4 and Corollary 3 (Case k odd). The red symbol $\ominus$ represents the inexistence of arcs between these subgraphs. The arcs from $V(D_{n,k}) \setminus V(K^1_{n+1})$ to $V(K^1_{n+1})$ are not represented.



(b) $D_{2,5}$ in Cor. 3 where the arcs from all vertices to triangle K^1_3 have been omitted.

Figure 3: Digraph $D_{n,k}$ described in Theorem 4 and Corollary 3.

Clearly, $mfvs(D_{n,k}) = 2n$, and any MFVS consists of $\{x^1_1, \ldots, x^1_n\}$ plus n vertices of K^2_{n+1}.

First, note that to process one vertex of K^1_{n+1}, there must be a step of any process strategy for $D_{n,k}$ where n agents are simultaneously occupying n nodes of K^1_{n+1}. Hence, $pn(D_{n,k}) \geq n$. Note that, similarly, any process strategy for $D_{n,k}$ must occupy n vertices of K^2_{n+1}. Moreover, because of the arcs from $V(D_{n,k}) \setminus V(K^1_{n+1})$ to $V(K^1_{n+1})$, any agent that is placed at some vertex in $V(D_{n,k}) \setminus V(K^1_{n+1})$ can only be removed when all vertices of K^1_{n+1} are occupied or processed. Consider any process strategy S for $D_{n,k}$ (in particular, S uses at least n agents) and let s_0 be the first step of S that does not consist in placing an agent at some vertex of K^1_{n+1}. By the above remarks, after step $s_0 - 1$ of S, n agents are occupying n vertices of $V(K^1_{n+1})$. Up to reorder the first $s_0 - 1$ steps of S, we obtain a process strategy for $D_{n,k}$ that starts by placing n agents at n vertices of $V(K^1_{n+1})$, without increasing the number of agents used nor the number of vertices occupied by S. Moreover, if the vertex of $V(K^1_{n+1})$ that is not occupied is x^1_i with $i < n+1$, it means that an agent is placed at x^1_{n+1} during the first n steps of the strategy. Replacing this operation by the placement of an agent at x^1_i instead of x^1_{n+1} does not modify the remaining part of the strategy (but the operation "remove the agent from x^1_{n+1}" which is replaced by "remove the agent from x^1_i") since the vertex x^1_{n+1} can be processed immediatly when the n other vertices of K^1_{n+1} are occupied. Hence, we may assume that S starts by placing agents at $\{x^1_1, \ldots, x^1_n\}$ and then processes x^1_{n+1}.

Second, any process strategy for any graph can easily be modified, without increasing (possibly decreasing) the number of used agents nor the number of occupied vertices, in such a way that the strategy processes all possible vertices before placing or removing agents. In other words, the rule R_3 can be made prioritary without increasing the considered parameters. Therefore, any process strategy S for $D_{n,k}$ can be modified, without increasing the number of agents used nor the number of vertices occupied by S, into a strategy that first places n agents at $\{x^1_1, \ldots, x^1_n\}$, then processes x^1_{n+1} and all vertices of $P^2_{n,k}$, and finally that mimicks S. Such a strategy is called a *good* process strategy for $D_{n,k}$.

Third, $pn(D_{n,k}) \leq n+1$ as proved by the following strategy S^*. First, place n agents at $\{x^1_1, \ldots, x^1_n\}$, then process x^1_{n+1}, and then process all vertices of $P^2_{n,k}$. In the next sentence, $y^{0,1}_i$ denotes x^1_i and $y^{k+1,1}_i$ denotes x^2_i, $i \leq n$. Then, for $j = 1 \ldots k+1$, the j^{th} phase of S^* consists of the following: for $i = 1 \ldots n$, place an agent at $y^{j,1}_{n-i+1}$ if j odd (resp., at $y^{j,1}_i$ if j even) and remove the agent at $y^{j-1,1}_{n-i+1}$ (resp., at $y^{j-1,1}_i$ if j even). Finally, process all vertices of K^2_{n+1}.

Let p, $0 \leq p \leq n-2$ (we choose $n \geq p-2$). Let S be a good process strategy for $D_{n,k}$ that uses $n+1+p$ agents (which exists by the previous remarks). We assume that S minimizes the number q of independent sets $IS_n^{t,1}$ of $D_{n,k}$ for which a vertex is occupied during the execution of S. Such an independent set is said *touched*. Note that the transformation that makes a strategy *good* does not increase the number of touched independent sets. Therefore, $2n + q \leq mfvs_{n+1+p}(D_{n,k})$ since any strategy occupies n vertices in each clique plus at least one vertex per touched independent set. In the sequel, we will prove that $q \geq k$, i.e., all independent sets of $P_{n,k}^1$ must be touched, and then, taking $k > 2n(C-1)$, we get that $\frac{mfvs_{pn+p}(D_{n,k})}{mfvs(D_{n,k})} = \frac{mfvs_{pn+p}(D_{n,k})}{2n} \geq \frac{mfvs_{n+1+p}(D_{n,k})}{2n} \geq \frac{2n+k}{2n} > C$.

It remains to prove that S touches all the k independent sets of $P_{n,k}^1$. To do so, we will modify S, possibly increasing the number of occupied vertices but without increasing the number of touched independent sets.

Since S is good, it first places n agents at $\{x_1^1, \ldots, x_n^1\}$, then processes x_{n+1}^1 and all vertices of $P_{n,k}^2$. We set $x_i^1 = y_i^{0,1}$, for all $i \leq n$. Let $S^0 = S$. For $0 < j < k$, let S^j be the strategy that mimicks the j first phases of S^* and then performs in the same order those movements of S^0 that concern the unprocessed vertices at this step. We prove by induction on $j < k$ that S^j can be transformed into the good process strategy S^{j+1} for $D_{n,k}$ satisfying the desired properties without increasing the number of touched independent sets. Clearly, S^0 is a good process strategy for $D_{n,k}$ that satisfies these properties.

Assume that, for some $0 \leq j < k-1$, S^j is a good process strategy that satisfies the desired properties. Then, S^j starts by occupying the vertices of $\{x_1^1, \ldots, x_n^1\}$, processes x_{n+1}^1 and the vertices of $P_{n,k}^2$ and then occupies and processes successively all vertices of $IS_n^{r,1}$, $r = 1 \ldots j$ until all vertices of $IS_n^{j,1}$ are occupied. Let s_j be the step of S^j when it occurs. We first prove that S^j touches $IS_n^{j+1,1}$. Indeed, if j is even, there are n vertex-disjoint paths from $y_n^{j+1,1}$ (resp., from $y_1^{j+1,1}$ if j is odd) to $x_1^2, \ldots, x_n^2$. While $y_n^{j+1,1}$ (resp., from $y_1^{j+1,1}$ if j is odd) is not processed, no agent in $IS_n^{j,1}$ can be removed, and thus only $p+1 \leq n-1$ agents are available. Therefore, the only way to process $y_n^{j+1,1}$ (resp., from $y_1^{j+1,1}$ if j is odd) is to place an agent at it. Hence, there is a step of S^j (hence, of S^0) that consists of placing an agent at $y_n^{j+1,1}$ (resp., $y_1^{j+1,1}$ if j is odd). Hence, S^0 touches $IS_n^{j+1,1}$. To conclude, we modify S^j by adding after step s_j the $j+1^{th}$ phase of S^*. That is, after step j, the strategy successively occupies the vertices of $IS_n^{j+1,1}$ removing the agents at $IS_n^{j,1}$ until all vertices of $IS_n^{j+1,1}$ are occupied and all vertices of $IS_n^{j,1}$ have been processed. Then, the strategy mimicks the remaining steps of S^j. The strategy obtained in such a way is clearly S^{j+1} that satisfies all desired properties. In particular, the obtained strategy is a good process strategy for $D_{n,k}$ that touches the same independent sets as S^0. □

Note that there exists a $(pn(D)+p+1, mfvs(D))$-process strategy for the digraph $D_{n,k}$ described in the proof of Theorem 4 whereas the minimum q such that a $(pn(D)+p, q)$-process strategy for $D_{n,k}$ exists, is arbitrarily large.

Corollary 3. *For any $C > 0$, there exists a digraph D such that $\frac{mfvs_{pn}(D)}{mfvs(D)} > C$.*

We obtain this result by considering the digraph $D_{n,k}$ described in Figure 3(a), with $n = 2$ and $k \geq 1$ (Figure 3(b) represents $D_{2,5}$). This digraph is such that $pn(D_{2,k}) = 3$ and $mfvs(D_{2,k}) = 4$ while $\frac{mfvs_{pn}(D_{2,k})}{mfvs(D_{2,k})} = \frac{k+4}{4}$ is unbounded.

Lemma 5 in Section 4 shows that, in the class of symmetric digraphs with bounded process number, $\frac{mfvs_{pn}(D)}{mfvs(D)}$ is bounded.

4 Behaviour of ratios in symmetric digraphs

We address in this section the behaviour of $\frac{mfvs_{pn}(D)}{mfvs(D)}$ for symmetric digraphs D. Note that the behaviours of $\frac{pn_{mfvs+q}(D)}{pn(D)}$ and $\frac{pn_{mfvs}(D)}{pn(D)}$ have already been studied in Section 3 for symmetric digraphs with bounded process number.

Lemma 5. *For any symmetric digraph D, $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq pn(D)$.*

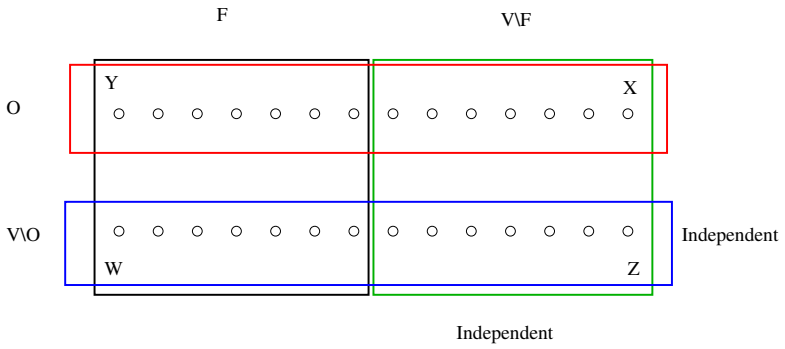


Figure 4: Proof of Lemma 5

Proof. Without loss of generality, we prove the lemma for a connected digraph D. Let S be a $(pn(D), mfvs_{pn}(D))$-process strategy for $D = (V, E)$. Let $O \subseteq V$ be the set of vertices occupied by an agent during the execution of S. Let F be a MFVS of D. Let us partition V into $(Y, X, W, Z) = (O \cap F, O \setminus F, F \setminus O, V \setminus (O \cup F))$. Since D is symmetric, $V \setminus F$ is an independent set because it is the complementary of a MFVS. Since the vertices not occupied by S have all their neighbors occupied, $V \setminus O$ is an independent set. Given $V' \subseteq V$, $N(V')$ denotes the set of neighbors of the vertices in V'. The partition is illustrated in Figure 4.

First, note that $|N(W) \cap X| \leq pn(D)|W|$, because, for any vertex $v \in W$ to be processed, all its neighbors must be occupied by an agent. Thus, the maximum degree of v is $pn(D)$.

Then, we prove that $|X \setminus N(W)| \leq (pn(D) - 1)|Y|$. Let $R = X \setminus N(W)$. Because $X \cup Z$ is an independent set, for any $v \in R$, $N(v) \subseteq Y$. Let $T = N(R) \subseteq Y$. Note that $N(T) \cap R = R$ because D is connected and symmetric. Let us order the vertices of $T = \{v_1, \cdots, v_t\}$ in the sequence in which they are processed (when the agents are removed) when executing S. For any i, $1 \leq i \leq t$, let $N_i = \bigcup_{j \leq i} N(v_j) \cap R$. We aim at proving that $|N_1| < pn(D)$ and $|N_{i+1} \setminus N_i| < pn(D)$ for any $i < t$. Hence, we obtain $|N_t| = |R| \leq (pn(D) - 1)|T| \leq (pn(D) - 1)|Y|$.

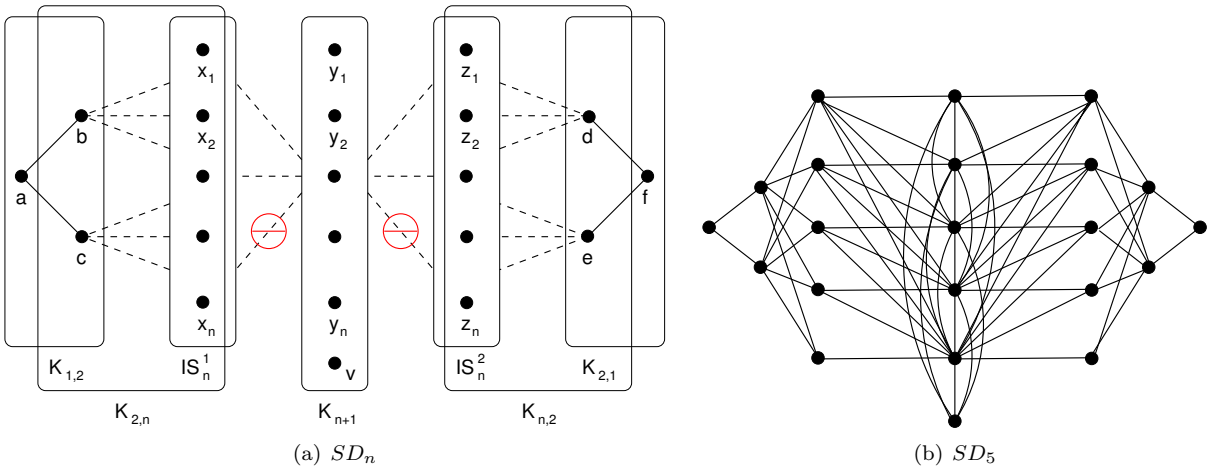
Let us consider the step of S just before an agent is removed from v_1. Let $v \in N_1 \neq \emptyset$. Since the agent will be removed from v_1, either v has already been processed or is occupied by an agent. We prove that there is a vertex in $N(v) \subseteq T$ that has not been occupied yet and thus v must be occupied. Indeed, otherwise, all neighbors of v are occupied (since, at this step, no agents have been removed from the vertices of T) and the strategy can process v without placing any agent on v, contradicting the fact that S occupies the fewest vertices as possible. Therefore, just before an agent is to be removed from v_1, all vertices of N_1 are occupied by an agent. Hence, $|N_1| < pn(D)$.

Now, let $1 < i \leq t$. Let us consider the step of S just before an agent is removed from v_i. Let $v \in N_i \setminus N_{i-1}$ if such a vertex exists. Since the agent will be removed from v_i, either v has already been processed or is occupied by an agent. We prove that there is a vertex in $N(v) \subseteq T \setminus N_{i-1}$ that has not been occupied yet and thus v must be occupied. Indeed, otherwise, all neighbors of v are occupied (since, at this step, no agents have been removed from the vertices of $T \setminus N_{i-1}$) and the strategy can process v without placing any agent on v, contradicting the fact that S occupies the fewest vertices as possible. Therefore, just before an agent to be removed from v_i, all vertices of $N_{i+1} \setminus N_i$ are occupied by an agent. Hence, $|N_{i+1} \setminus N_i| < pn(D)$.

To conclude: $mfvs_{pn}(D) = |O| = |Y| + |X|$ and $|X| = |X \setminus N(W)| + |N(W) \cap X|$. Hence, $mfvs_{pn}(D) \leq pn(D)(|Y| + |W|) = pn(D)|F| = pn(D) \cdot mfvs(D)$. □

Lemma 6. *For any given $\varepsilon > 0$, there exists a symmetric digraph D such that $3 - \varepsilon \leq \frac{mfvs_{pn}(D)}{mfvs(D)} < 3$.*

Proof. Let $n \geq 1$. Let us consider the digraph SD_n built as follows. Let IS_n^1 and IS_n^2 be two independent sets of n nodes each: respectively $x_1, \ldots, x_n$ and $z_1, \ldots, z_n$. Let K_{n+1} be a symmetric clique of $n + 1$ nodes $y_1, \ldots, y_n, y_{n+1} = v$. The digraph SD_n is built starting from the disjoint union of IS_n^1, IS_n^2, K_{n+1} and 6 isolated vertices $\{a, b, c, d, e, f\}$ by adding the following arcs. There are symmetric arcs between the nodes x_i and y_j and the nodes z_i and y_j, for any $1 \leq i \leq j \leq n$. Furthermore, all symmetric arcs



(a) SD_n (b) SD_5

Figure 5: Symmetric digraph SD_n of Lemma 6 (Figure 5(a)) and instance of SD_n when $n = 5$ (Figure 5(b)). The red symbol $\ominus$ represents the absence of arcs.

of the complete bipartite graph with partitions $\{b, c\}$ and IS_n^1 are added. Similarly, all symmetric arcs of the complete bipartite graph with partitions $\{d, e\}$ and IS_n^2 are added. Finally, the symmetric arcs $(a, b), (a, c), (d, f), (e, f)$ are added. The general shape of SD_n is depicted in Figure 5(a). The digraph SD_5 is represented in Figure 5(b).

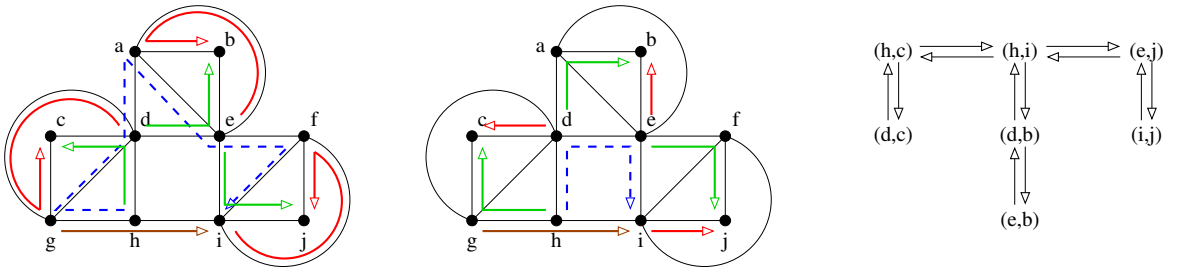
Note that the set $F = \{y_1, \ldots, y_n, b, c, d, e\}$ is a feedback vertex set of SD_n, with $|F| = n + 4$. Thus $mfvs(SD_n) \leq n + 4$ (actually, one can easily check that F is a minimum feedback vertex set of SD_n). Clearly, $pn(SD_n) \geq n$. In what follows, we prove that any strategy using $n + 1$ agents needs to cover at least $3n + 2$ vertices, and we present a $(n + 1, 3n + 2)$-process strategy for SD_n. Since $mfvs_{n+1}(D) \leq mfvs_{pn}(D)$ for any digraph D, the result follows.

First, we prove by contradiction that all process strategies for SD_n using $n+1$ agents must start by processing either the nodes b and c or the nodes d and e, and so by placing the $n + 1$ agents either at vertices a and $x_1, \ldots, x_n$ or at vertices f and $z_1, \ldots, z_n$.

Suppose that the first vertex to be processed is either a or belongs to IS_n^1, and it is processed at step s. Therefore, the vertices b and c must be occupied by agents at this step (such that a or a vertex in IS_n^1 can be processed thereafter). Without loss of generality, let us assume that b is processed, say at step s', before c. Since at most $n - 1$ agents are available while c and b are occupied, no vertex of the clique K_{n+1} can be processed before step s'. On the other hand, at step s', all vertices of IS_n^1 are processed or occupied by agents such that b can be processed. Let X be the subset of vertices of IS_n^1 that are occupied at step s', and let $\bar{X} = V(IS_n^1) \setminus X$. For any $x_i \in \bar{X}$, y_i must be occupied at step s' (since x_i is processed and y_i is not). Hence, at step s', at least $2 + |X| + |\bar{X}| = n + 2$ agents are occupying some vertices, a contradiction. By symmetry, f and any vertex of IS_n^2 cannot be the first vertex to be processed.

Now suppose that the first vertex to be processed is $y_i \in K_{n+1}$, $i \leq n + 1$. Note that all vertices of K_{n+1}, but $y_{n+1} = v$, have at least $n + 2$ outneighbors. Therefore, $i = n + 1$. When v is processed, the n vertices of $K_{n+1} \setminus \{v\}$ must be occupied, leaving at most one free agent. But now, all vertices of K_{n+1} but v have at least 2 unprocessed outneighbors. Whatever be the placement of the last agent, no other vertex can be processed and no agents can be released. Hence, the strategy fails, a contradiction.

Hence, any process strategy using $n + 1$ agents must start by processing b, c, d or e. Without loss of generality, (by symmetry), let us assume that the first vertex to be processed is b. Hence, the strategy must start by placing agents at any vertex in $\{a\} \cup V(IS_n^1)$. At this step, the strategy processes b and c without covering them. Then a can be processed and the agent at it is released. At this step, no other vertex can be processed. Moreover, the only move that can be done is to place the free agent at y_n. Indeed, any other move would let all agents blocked. Then the free agent is placed at node y_n and x_n can be processed and the agent occupying it can be released. Similarly, the strategy sequentially places an agent at y_{n-i}, processes x_{n-i} and removes the corresponding agent, for $1 \leq i \leq n - 1$. It



(a) Old set of routes S_1.

(b) New set of routes S_2.

(c) Dependency digraph D from S_1 to S_2.

Figure 6: Instance of the reconfiguration problem consisting of a network with 10 nodes and symmetric arcs, 8 connections $(h,i),(h,c),(d,c),(d,b),(e,b),(e,j),(i,j),(g,i)$ to be reestablished. Figure 6(a) depicts the old set of routes S_1, Figure 6(b) the new set S_2, and Figure 6(c) the dependency digraph from S_1 to S_2.

is easy to check that any variation of this would make the strategy immediately fail. Once all vertices $y_1, \cdots, y_n$ are occupied, then v can be processed without being covered. Then, the strategy goes on being highly constrained: for $1 \leq i \leq n$, the free agent occupies z_i, allowing to process y_i and to free the agent occupying it. Finally, when all vertices of IS_n^2 are occupied, the free agent must occupy f, and all remaining vertices may be processed. Again, all these moves are forced for, otherwise, the strategy would be blocked.

Such a strategy covers $3n+2$ nodes. Therefore, $mfvs_{pn}(SD_n) \geq mfvs_{n+1}(SD_n) = 3n+2$. Hence, $\frac{mfvs_{pn}(SD_n)}{mfvs(SD_n)} \geq \frac{3n+2}{n+4}$. For $n > \frac{10}{\epsilon} - 4$, we get $\frac{mfvs_{pn}(SD_n)}{mfvs(SD_n)} \geq 3 - \epsilon$. Moreover, since SD_n has $3n+7$ vertices, we get $mfvs_{pn}(SD_n) \leq 3n+6$, and so $\frac{mfvs_{pn}(SD_n)}{mfvs(SD_n)} < 3$. □

Conjecture 1. *For any symmetric digraph D, $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq 3$.*

5 Process Strategy out of the Routing Reconfiguration Problem

The *routing reconfiguration problem* occurs in connection-oriented networks such as telephone, MPLS, or WDM [3, 5–7, 19, 21]. In such networks, a connection corresponds to the transmission of a data flow from a source to a destination, and is usually associated with a capacited path (or a wavelength in WDM optical networks). A *routing* is the set of paths serving the connections. In the context of all optical WDM networks without wavelength conversion, we not only consider paths but lightpaths, that is a path in the network and its specific wavelength. Without loss of generality, we assume here that each arc of the network has capacity one, and that each connection requires one unit of capacity (each arc has one wavelength). Consequently, no two paths can share the same arc (this is a valid assumption in WDM networks where no two lightpaths can use the same wavelength on the same fiber). When a link of the network needs to be repaired, it might be necessary to change the routing of the connection using it, and incidentally to change the routing of other connections if the network has not enough free resources. Computing a new viable routing is a well known hard problem, but it is not the concern of this paper. Indeed, this is not the end of our worries: once a new routing not using the unavailable links is computed, it is not acceptable to stop all the connections going on, and change the routing, as it would result in a bad quality of service for the users (such operation requires minutes in WDM networks). Instead, it is preferred that each connection first establishes the new path on which it transmits data, and then stops the former one. This requires a proper scheduling to avoid conflicts in accessing resources (resources needed for a new path must be freed by other connections first). However, cyclic dependencies might force to interrupt some connections during that phase. The aim of the routing reconfiguration problem is to optimize tradeoffs between the total number and the concurrent number of connections to interrupt.

As an example, a way to reconfigure the instance depicted in Figure 6 may be to interrupt connections $(h,c),(d,b),(e,j)$, then set up the new paths of all other connections, tear down their old routes, and

finally, set up the new paths of connections $(h, c), (d, b), (e, j)$. Such a strategy interrupts a total of 3 connections and these ones are interrupted simultaneously. Another strategy may consist of interrupting the connection (h, i), then sequentially: interrupt connection (h, c), reconfigure (d, c) without interruption for it, set up the new route of (h, c), then reconfigure in the same way first (d, b) and (e, b) without interruption for these two requests, and then (e, j) and (i, j). Finally, set up the new route of (h, i). The second strategy implies the interruption of 4 connections, but at most 2 connections are interrupted simultaneously.

Indeed, possible objectives are (1) to minimize the maximum number of concurrent interruptions [5, 6,19–21], and (2) to minimize the total number of disrupted connections [13]. Following [6,13], these two problems can be expressed through the theoretical game described in Section 1.1, on the dependency digraph [13]. Given the initial routing and the new one, the dependency digraph contains one node per connection that must be switched. There is an arc from node u to node v if the initial route of connection v uses resources that are needed by the new route of connection u. Figure 6 shows an instance of the reconfiguration problem and its corresponding dependency digraph. In Figure 6(c), there is an arc from vertex (d, c) to vertex (h, c), because the new route used by connection (d, c) (Figure 6(b)) uses resources seized by connection (h, c) in the initial configuration (Figure 6(a)). Other arcs are built in the same way.

Given the dependency digraph D of an instance of the problem, a (p, q)-process strategy for D corresponds to a valid reconfiguration of the connections where p is the maximum number of concurrent disruptions and q is the total number of interruptions. Indeed the three rules can be viewed in terms of reconfiguration of requests:

R_1 Put an agent at a vertex v of D;
Interrupt the request corresponding to v;

R_2 Remove an agent from a vertex v of D if all its outneighbors are either processed or occupied by an agent, and process v;
Route an interrupted connection when final resources are available;

R_3 Process an unoccupied vertex v of D if all its outneighbors are either processed or occupied by an agent;
Reroute a non-interrupted connection when final resources are available.

Clearly, any instance of the routing reconfiguration problem may be represented by its dependency digraph. Therefore the next theorem proves the equivalence between instances of the reconfiguration problem and dependency digraphs.

Theorem 7. *Any digraph D is the dependency digraph of an instance of the routing reconfiguration problem whose network is a grid.*

Proof. Roughly, consider a grid network where each initial lightpath of any connection is some row of the grid. If two connections i and k are linked by an arc (i, k) in the dependency digraph, then we build the new lightpaths of both connections as depicted in Figure 7 which actually create the desired dependence. Note that the lightpath of connection k is deported on an additional row, i.e., a row corresponding to no connection. For each arc of the dependency digraph, we can use different columns of the grid-network, in such a way that these transformations may be done independently.

More formally, let $D = (V, A)$ be a digraph with $V = \{c_1, \cdots, c_n\}$ and $A = \{a_1, \cdots, a_m\}$. Let us define the network G as a $(n + 2) \times (2m)$ grid such that each edge of which has capacity one. Let R_i denotes the i^{th} row of G $(0 \leq i \leq n + 1)$ and C_i its i^{th} column $(1 \leq j \leq 2m)$, and let $v_{i,j} \in V(G)$ be the vertex in $R_i \cap C_j$. For any i, $1 \leq i \leq n$, connection i, corresponding to c_i in D, occurs between $v_{i,1} \in V(G)$, being the leftmost vertex of R_i and $v_{i,2m} \in V(G)$, being the rightmost vertex of R_i, and let the initial lightpath of connection i follows R_i. Now, we present an iterative method to build the new lightpath of each connection. Initially, for any i, $1 \leq i \leq n$, the new lightpath P_i^0 of connection i equals the old lightpath R_i. Now, after the $(j-1)^{th}$ step $(0 < j \leq m)$ of the method, let P_i^{j-1} be the current value of the new lightpath of connection i and assume that in the subgraph of G induced by columns $(C_{2j-1}, \cdots, C_{2m})$, P_i^{j-1} equals R_i. Consider $a_j = (c_i, c_k) \in A$ and let us do the following transformation depicted in Figure 7. For any $\ell \notin \{i, k\}$, $P_\ell^j = P_\ell^{j-1}$. Now, P_i^j is defined by replacing

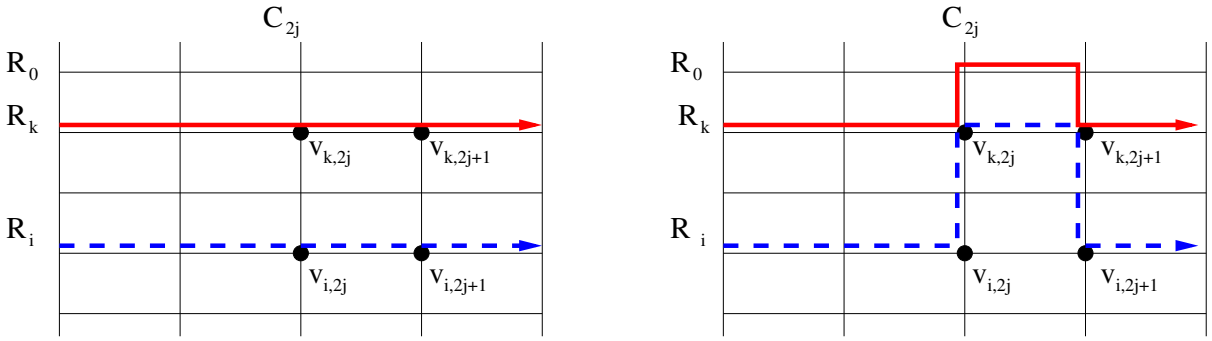


Figure 7: Scheme of the transformation in the proof of Theorem 7

the edge $(v_{i,2j-1}, v_{i,2j})$ in P_i^{j-1} by the shortest path from $v_{i,2j-1}$ to $v_{k,2j-1}$ (following C_{2j-1}), the edge $(v_{k,2j-1}, v_{k,2j})$, and the shortest path from $v_{k,2j}$ to $v_{i,2j}$ (following C_{2j}). Similarly, P_k^j is defined by replacing the edge $(v_{k,2j-1}, v_{k,2j})$ in P_k^{j-1} by the shortest path from $v_{k,2j-1}$ to $v_{n+1,2j-1}$ if $i < k$ (resp., to $v_{0,2j-1}$ if $i > k$), the edge $(v_{n+1,2j-1}, v_{n+1,2j})$ (resp., $(v_{0,2j-1}, v_{0,2j})$), and the shortest path from $v_{n+1,2j}$ to $v_{k,2j}$ (resp., from $v_{0,2j}$ to $v_{k,2j}$). It is easy to check that the grid G, the sets of initial lightpaths $\{R_1, \cdots, R_n\}$ and final lightpaths $\{P_1^m, \cdots, P_n^m\}$ admit D as dependency digraph. □

Note that a digraph may be the dependency digraph of various instances of the reconfiguration problem. Since any digraph may be the dependency digraph of a realistic instance of the reconfiguration problem, Theorem 7 shows the relevance of studying these problems through dependency digraph notion.

A feasible reconfiguration may be defined by a (p, q)-process strategy for the corresponding dependency digraph. Problem (1) is equivalent to minimize p (Section 1.1) and Problem (2) is similar to the one of minimizing q (Section 1.1). Recall that Problem (1) corresponds to minimize the maximum number of concurrent interruptions, and Problem (2) corresponds to minimize the total number of disrupted connections. Consider the dependency digraph D of Figure 6. From Section 1.1, we can not minimize both p and q, that is the number of simultaneous disrupted requests and the total number of interrupted connections. Indeed there does not exist a $(2, 3)$-process strategy while $(2, 4)$ and $(3, 3)$ exist (Figure 1(a) and Figure 1(b)).

It is now easy to establish the relationship between tradeoff metrics introduced in Section 1.1 and tradeoffs for the routing reconfiguration problem. For example, pn_{mfvs} introduced in Definition 3 represents the minimum number of requests that have to be simultaneously interrupted during the reconfiguration when the total number of interrupted connections is minimum. Also Section 2 shows that the problems of computing these new tradeoffs parameters for the routing reconfiguration problem are NP-complete and not in APX. Finally, Section 3 proves that the loss one can expect on one parameter when minimizing the other may be arbitrarily large.

6 Conclusion

In this paper, we address the routing reconfiguration problem through a game played on digraphs. We introduce the notion of (p, q)-process strategy and some tradeoff metrics in order to minimize one metric under the constraint that the other is fixed. We proved that the problems of computing these parameters are APX-hard and some are not in APX. We also proved that there exist digraphs for which minimizing one parameter may increase the other arbitrarily. For further research, we plan to continue our study for symmetric digraphs in order to (dis)prove Conjecture 1. Moreover, it would be interesting to design exact algorithms and heuristics to compute (p, q)-process strategies.

Acknowledgments

This work has been partially supported by région PACA, ANRs AGAPE and DIMAGREEN.

References

[1] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[2] R. L. Breisch. An intuitive approach to speleotopology. *Southwestern Cavers*, VI(5):72–78, 1967.

[3] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse. Tradeoffs in process strategy games with application in the WDM reconfiguration problem. In P. Boldi and L. Gargano, editors, *Fifth International conference on Fun with Algorithms (FUN 2010)*, volume 6099 of *Lecture Notes in Computer Science*, pages 121–132, Ischia Island, Italy, June 2010. Springer.

[4] D. Coudert, F. Huc, and D. Mazauric. A distributed algorithm for computing and updating the process number of a forest. *Algorithmica*, 2011, to appear.

[5] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J-S. Sereni. Routing reconfiguration/process number: Coping with two classes of services. In *13th Conference on Optical Network Design and Modeling (ONDM)*, pages 1–6, Braunschweig, Germany, February 2009. IEEE.

[6] D. Coudert, S. Perennes, Q.-C. Pham, and J.-S. Sereni. Rerouting requests in WDM networks. In *7ème Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'05)*, pages 17–20, Presqu'île de Giens, France, May 2005.

[7] D. Coudert and J-S. Sereni. Characterization of graphs and digraphs with small process number. *Discrete Applied Mathematics (DAM)*, 2011, to appear.

[8] N. Deo, S. Krishnamoorthy, and M. A. Langston. Exact and approximate solutions for the gate matrix layout problem. *IEEE Transactions on Computer-Aided Design*, 6:79–84, 1987.

[9] J. Díaz, J. Petit, and M. Serna. A survey on graph layout problems. *ACM Computing Surveys*, 34(3):313–356, September 2002.

[10] J.A. Ellis, I.H. Sudborough, and J.S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1):50–79, 1994.

[11] G. Even, J. Naor, B. Schieber, , and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. In E. Balas and J. Clausen, editors, *4th International Integer Programming and Combinatorial Optimization Conference (IPCO)*, volume 920 of *Lecture Notes in Computer Science*, pages 14–28. Springer, May 1995.

[12] F. Fomin and D. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

[13] N. Jose and A.K. Somani. Connection rerouting/network reconfiguration. In *4th International Workshop on Design of Reliable Communication Networks (DRCN)*, pages 23–30. IEEE, October 2003.

[14] V. Kann. *On the Approximability of NP-complete Optimization Problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm., 1992. see also: http://www.csc.kth.se/ viggo/wwwcompendium/node19.html.

[15] M. Kirousis and C.H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47(2):205–218, 1986.

[16] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the Association for Computing Machinery*, 35(1):18–44, 1988.

[17] T. D. Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 426–441. Springer, Berlin, May 1978.

[18] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, August 1983.

[19] F. Solano. Analyzing two different objectives of the WDM network reconfiguration problem. In *IEEE Global Communications Conference (Globecom)*, pages 1–7, December 2009.

[20] F. Solano and M. Pióro. A mixed-integer programing formulation for the lightpath reconfiguration problem. In *VIII Workshop on G/MPLS Networks (WGN8)*, October 2009.

[21] F. Solano and M. Pióro. Lightpath reconfiguration in WDM networks. *IEEE/OSA Journal of Optical Communication and Networking*, 2(12):1010–1021, December 2010.