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

Nathann Cohen, David Coudert, Dorian Mazauric, Napoleao Nepomuceno, Nicolas Nisse

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

Nathann Cohen[1], David Coudert[1], Dorian Mazauric[1], Napoleão Nepomuceno[1,2], Nicolas Nisse[1]

[1]MASCOTTE, INRIA, I3S, CNRS, UNS, Sophia Antipolis, France.
[2]Universidade Federal do Ceará, Fortaleza-CE, Brazil.

Abstract

We consider a variant of the graph searching games that is closely related to the routing reconfiguration problem in WDM networks. In the digraph processing game, a team of agents is aiming at clearing, or *processing*, the vertices of a digraph D. In this game, two important measures arise: 1) the total number of agents used, and 2) the total number of vertices occupied by an agent during the processing of D. Previous works have studied the problem of minimizing each of these parameters independently. In particular, both of these optimization problems are not in APX. In this paper, we study the tradeoff between both these conflicting objectives. More precisely, we prove that there exist some instances 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 in the case of basic network topologies. On the other hand, we exhibit classes of instances where good tradeoffs can be achieved. We also show that minimizing one of these parameters while the other is constrained is not in APX.

Keywords: Graph searching, process number, routing reconfiguration.

1 Introduction

In this paper, we study the *digraph processing* game analogous to graph searching games [9]. This game aims at *clearing* the vertices of a contaminated directed graph D. For this, we use mobile agents that are sequentially put to and removed from the vertices of D. We are interested in two different measures and their tradeoffs: the minimum number of vertices that must be *covered* (i.e., visited by an agent) and the minimum number of agents required to *clear* D. This game is closely related to the routing reconfiguration problem in Wavelength Division Multiplexing (WDM) networks. In

*This work was partially funded by région PACA, ANRs AGAPE, and DIMAGREEN

this context, the goal is to reroute some connections that are established between pairs of nodes in a communication network, which unfortunately can lead to interruptions of service. Each instance of this problem may be represented by a directed graph called its *dependency digraph* D such that the reconfiguration problem is equivalent to the clearing of D. More precisely, the two measures presented above respectively correspond to the total number of requests disrupted during the rerouting of the connections, and to the number of simultaneous disruptions during the whole process. The equivalence between these two problems is detailed in Section 5.

The digraph processing game has been introduced in [5] for its relationship with the routing reconfiguration problem. This game is defined by the three following operations (or rules), which are very similar to the ones defining the *node search number* [1, 9, 12, 14] of a graph and whose goal is to clear, or to *process*, all the vertices of a digraph D :

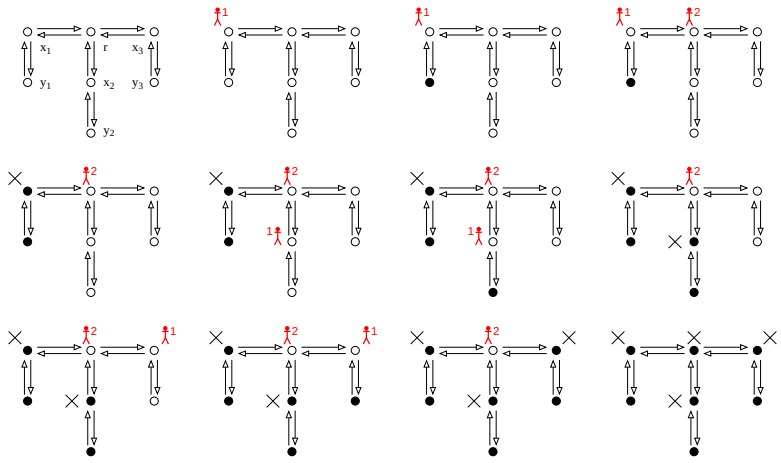
- R_1 Put an agent at a vertex v of D;
- R_2 Remove an agent from a vertex v of D if all its outneighbors are either processed or occupied by an agent, and process v;
- R_3 Process an unoccupied vertex v of D if all its outneighbors are either processed or occupied by an agent.

A graph 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 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 vertices of 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.

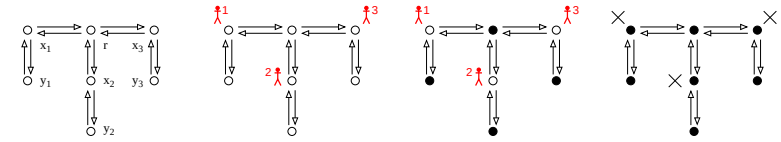
Fig. 1 illustrates two process strategies for a symmetric digraph D of 7 vertices. The strategy depicted in Fig. 1(a) first put an agent at vertex x_1 (R_1), which enables to process y_1 (R_3). A second agent is then put at r allowing the vertex x_1 to be processed, and the agent on it to be removed (R_2). The procedure goes on iteratively, until all the vertices are processed after 11 steps. The depicted strategy uses 2 agents and covers 4 vertices. Another process strategy is depicted, Fig. 1(b), uses 3 agents and covers 3 vertices. Note that the latter strategy consists in placing agents at the vertices of a feedback vertex set (FVS)[1] of minimum size.

Clearly, any digraph can be processed by placing simultaneously an agent at every node. However, Rule R_3 allows to process some vertices without placing an agent on it. More precisely, to process a digraph D, it is sufficient to put an agent at every vertex of a feedback vertex set F of D, then the vertices of $V(D) \setminus F$ can be processed using Rule R_3, and finally all agents

[1] $F \subseteq V$ of a digraph $D = (V, A)$ is a FVS if removing all vertices 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.

can be removed. In particular, a Directed Acyclic Graph (DAG) can be processed using 0 agents and thus covering no vertices. Indeed, to process a DAG, it is sufficient to process sequentially its vertices starting from the leaves. Remark that any process strategy for a digraph D must cover all vertices of a feedback vertex set of D (not necessarily simultaneously). In general, 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, 6, 4], 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* of D.

We are interested in tradeoffs between the minimum number of agents used by a strategy and the minimum number of vertices covered during it.

1.1 Definitions and Previous Results

Let D be a n-node digraph. A (p, q)-*process strategy* 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* instead of (p, n)-process strategy. Similarly, when the number of

agents is not constrained, *q-process strategy* replaces (n, q)-process strategy.

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

Definition 1 *Let $pn(D)$ denote the smallest p such that there exists a p-process strategy for D.*

For instance, the digraph Fig. 1 satisfies $pn(D) = 2$. Indeed, Fig. 1(a) describes a process strategy using 2 agents, and it is easy to check that no strategy can process D using at most 1 agent. While digraphs with process number 0, 1, and 2 can be recognized in polynomial time [6], computing the process number is NP-complete and not in APX (i.e., admitting no polynomial-time approximation algorithm up to a constant factor, unless $P = NP$) [5]. A distributed polynomial-time algorithm to compute the process number of trees (or forests) with symmetric arcs has been proposed in [3]. Furthermore the first heuristic for computing the process number of any digraph is described in [4]. In [16], Solano conjectured that computing the process number of a digraph can be solved in polynomial time if the set of covered vertices is given as part of the input. We disprove this conjecture, showing that computing the process number of a digraph remains out of APX (and so is NP-complete) even when the subset of vertices at which an agent will be put is given (see Theorem 1).

The *node search number* and the *pathwidth* are graph invariants closely related to the notion of process number for undirected graph. 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 [1, 9, 12, 14] for more details. The notion of pathwidth was introduced by Robertson and Seymour in [15]. It has been proved in [8] by Ellis *et al.* that the pathwidth and the node search number are equivalent, that is for any graph G, $pw(G) = sn(G) - 1$, and in [5] that $pw(G) \leq pn(G) \leq pw(G) + 1$ (and so $sn(G) - 1 \leq pn(G) \leq sn(G)$), where the graph G is considered as a symmetric digraph. Since the problem of determining the pathwidth of a graph is NP-complete [13] and not in APX [7], these two parameters behave similarly.

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 similar to the one of computing the size of a *minimum feedback vertex set* (MFVS) of D. Computing such a set is well known to be NP-complete and not in APX [10]. 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, for the digraph of Fig. 1, $mfvs(D) = 3$. As mentioned above, $mfvs(D) \geq pn(D)$. Moreover, the gap between these two parameters may be arbitrarily large. For example a symmetric path P_n of $n \geq 4$ nodes is such that: $mfvs(P_n) = \lfloor \frac{n}{2} \rfloor$ while $pn(P_n) = 2$.

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 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 define $pn_{mfvs}(D) = pn_q(D)$ when $q = mfvs(D)$.*

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 define $mfvs_{pn}(D) = mfvs_p(D)$ when $p = pn(D)$.*

Note that $mfvs_{pn}(D)$ is precisely the minimum number of vertices that must be covered by a process strategy using the minimum number of agents, and that $pn_{mfvs}(D)$ is the minimum number of agents required by a process strategy minimizing the number of covered vertices.

To illustrate the pertinence of these tradeoff metrics, consider the digraph D of Fig. 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 (Fig. 1(a)) and a $(3, 3)$-process strategy (Fig. 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 one value without increasing the other.

We generalize this concept through the introduction of the notion of minimal values for a digraph D. We say that (p, q) is a minimal value for a digraph D if $p = pn_q(D)$ and $q = mfvs_p(D)$. Remark that $(pn(D), mfvs_{pn}(D))$ and $(pn_{mfvs}(D), mfvs(D))$ are both minimal values by definition (and may be the same). Clearly for a given digraph D, the number of minimal values is linear in the number of nodes $n = |V(D)|$. For the digraph of Fig. 1, there are two minimal values: $(2, 4)$ and $(3, 3)$. Fig. 2 represents the shape of minimal values for a digraph D. More precisely, Fig. 2 depicts the variations of the minimum number q of vertices covered by a p-strategy for D ($p \geq pn(D)$, i.e., $mfvs_p(D)$ as a function of p. Clearly, it is a non-increasing function greater than by $mfvs(D)$.

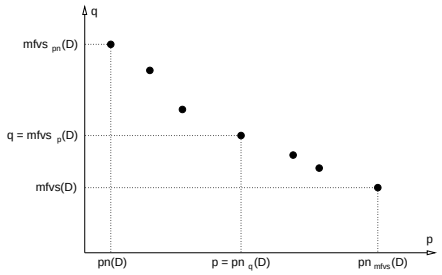


Figure 2: Representation of minimal values.

1.2 Our Results

Our results constitute an analysis of the behaviour of the two given measures both in general digraphs and in symmetric digraphs. In general, as mentioned above, no process strategy minimizes both the number of agents and the number of covered vertices (see, e.g., Fig. 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 (Sec. 2) and the existence of digraphs for which it can be arbitrarily large (Sec. 3 and Sec. 4). More precisely, we first disprove the conjecture of Solano [16] (Th. 1). Then, we prove that all parameters $pn_{mfvs}(D), mfvs_{pn}(D)$, $\frac{pn_{mfvs}(D)}{pn(D)}$, and $\frac{mfvs_{pn}(D)}{mfvs(D)}$ are not in APX (Th. 2). Then, we prove that $\frac{pn_{mfvs}(D)}{pn(D)}$ and $\frac{mfvs_{pn}(D)}{mfvs(D)}$ are not bounded in general digraphs even in the class of bounded process number digraphs (Th. 3 and 4). However, $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq pn(D)$ for any symmetric digraph D (Lemma 1). Due to lack of space most of the proofs are sketched and can be found in [2].

2 Complexity Results

Before proving that computing the tradeoff parameters introduced in Section 1.1 are NP-complete and not in APX, we disprove a conjecture of Solano about the complexity of computing the process number of a digraph D.

Indeed a possible approach for computing the process number, proposed in [16], consists of 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 are put at these vertices. Solano conjectures that the complexity of the process number problem resides in Phase 1 and that Phase 2 can be solved or approximated in polynomial time [16]. We disprove this conjecture.

Theorem 1 *Computing the process number of a digraph D is not in APX (and thus NP-complete), even when the subset of covered vertices is given.*

Sketch of the Proof. Let $D = (V, A)$ be a symmetric digraph with $V = \{u_1, \ldots, u_n\}$. Let $D' = (V', A')$ be the symmetric digraph with $V' = V \cup \{v_1, \ldots, v_n\}$ obtained from D by adding 2 symmetric arcs between u_i and v_i $(i = 1, \ldots, n)$. It is easy to show that there exists an optimal process strategy for D' such that the set of occupied vertices is V. Now, consider the problem of computing an optimal process strategy for D' when the set of vertices covered by agents is constrained to be V. 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 [13] and not in APX [7]. ■

Theorem 2 *Given a digraph D, the problems of determining $pn_{mfvs}(D)$, $mfvs_{pn}(D)$, $\frac{pn_{mfvs}(D)}{pn(D)}$, and $\frac{mfvs_{pn}(D)}{mfvs(D)}$ are not in APX (and thus NP-complete).*

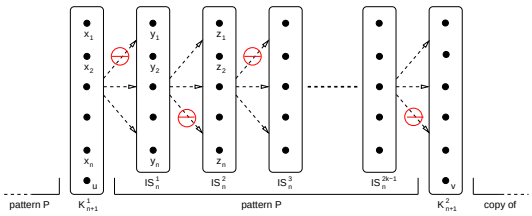
Proof. From Theorem 1, we know that the problem of determining pn_{mfvs} is not in APX. Indeed, in the class of graphs D' defined in the proof of Theorem 1, $pn(D') = pn_{mfvs}(D') = pw(D)+1 = sn(D)$ (where the relationship between D and D' is described in this proof).

Let H_n be a symmetric directed star with n branches each of which contains two vertices, excluding the central node r (e.g., Fig. 1 for $n = 3$). Let K_n be a symmetric n-node clique digraph, and D be any n-node digraph.

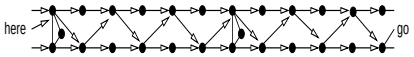
Let D' be the disjoint union of K_n and D. Clearly, $pn(D') = pn(K_n) = n - 1$.Thus $mfvs_{pn}(D') = n - 1 + mfvs(D)$ since when we process D we can use $n - 1$ agents. Since computing $mfvs(D)$ is not in APX, computing $mfvs_{pn}$ is not in APX. To show that $\frac{pn_{mfvs}}{pn}$ is not in APX, let D' be the digraph composed of two components H_n and D. Let us do some trivial remarks: (1) the neighbors of r belong to any MFVS of D'. (2) Moreover, r does not belong to a MFVS of D'. Hence, to process r while occupying at most $mfvs(D')$ vertices, all neighbors of r must be simultaneously occupied. This leads to $pn_{mfvs}(D') = n$. To conclude, it is sufficient to remark that $pn(D') = \max\{pn(D), pn(H)\}$. Hence, $\frac{pn_{mfvs}(D')}{pn(D')} = \frac{n}{\max\{pn(D),2\}}$. However, computing $pn(D)$ is not in APX [5].

To prove that $\frac{mfvs_{pn}}{mfvs}$ is not in APX, let D' be the digraph composed of K_n, H_n, and D. It is easy to show that $pn(D') = pn(K_n) = n - 1$. Hence, $\frac{mfvs_{pn}(D')}{mfvs(D')} = \frac{(n-1)+(n+1)+mfvs(D)}{(n-1)+n+mfvs(D)}$. Indeed to process H_n using $n-1$ agents, we must cover $n + 1$ nodes by agents: the central node r and successively its n neighbors (see Fig. 1 for $n = 3$). Furthermore, the minimum number of nodes covered by agents when we process D is $mfvs(D)$ because we have $n - 1$ available agents. Thus $\frac{mfvs_{pn}(D')}{mfvs(D')} = \frac{2n+mfvs(D)}{2n-1+mfvs(D)}$. To get this ratio we must compute $mfvs(D)$ which is not in APX. ■

Corollary 1 *Let $p \geq pn(D)$, $q \geq mfvs(D)$ be integers, and D a digraph. Computing $pn_q(D)$, $mfvs_p(D)$, $\frac{pn_q(D)}{pn(D)}$, or $\frac{mfvs_p(D)}{mfvs(D)}$ is not in APX.*



(a) D of Theorems 4 and 5.



(b) D of Theorem 4 ($n = 2$, $k = 3$).

Figure 3: Digraph D described in Theorems 4 and 5.

3 Behaviour of ratios in general digraphs

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

Theorem 3 *$\forall C > 0$, $q \in \mathbb{N}$, there exists a digraph D s.t. $\frac{pn_{mfvs+q}(D)}{pn(D)} > C$.*

Sketch of the Proof. Let H_n be a symmetric directed star with $n \geq 3$ branches each of which containing two vertices, excluding the central node r. H_3 is represented in Fig. 1. It is easy to check that $pn(H_n) = 2$ (e.g., Fig. 1(a)). Moreover, since the single MFVS of H_n is the set X of the n vertices adjacent to r, it is easy to check that $pn_{mfvs}(H_n) = n$ (e.g., Fig. 1(b)). We now build D with $q+1$ copies of H_n. Hence, $pn_{mfvs+q}(D) = n$ while $pn(D) = 2$. Taking $n > 2C$, we get $\frac{pn_{mfvs+q}(D)}{pn(D)} > C$. ■

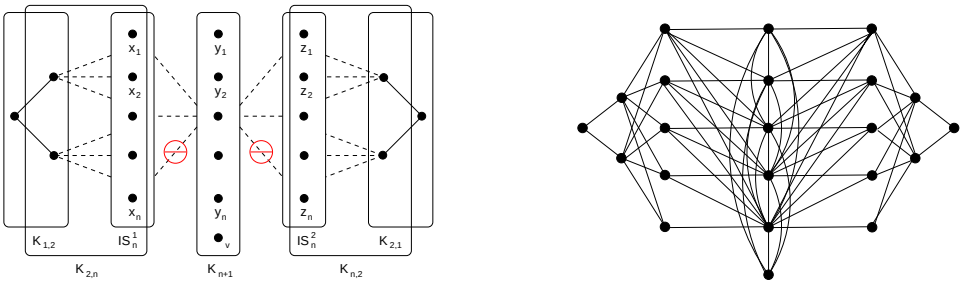
Corollary 2 *For any $C > 0$, there exists a digraph D s.t. $\frac{pn_{mfvs}(D)}{pn(D)} > C$.*

We now prove similar results for the other ratio. To do it, let us consider the digraph D of Fig. 3(a). K^1_{n+1} is a symmetric clique of $n+1$ nodes $x_1, \ldots, x_n, u$. IS^1_n and IS^2_n are two independent sets of n nodes each: respectively $y_1, \ldots, y_n$ and $z_1, \ldots, z_n$. In D, there is an arc from x_i to y_j, $i = 1, \ldots, n$, $j = 1, \ldots, n$, if and only if $j \geq i$. There is an arc from y_i to z_j, $i = 1, \ldots, n$, $j = 1, \ldots, n$, if and only if $i \geq j$. The other arcs of D are built in such a way for other independent sets $IS^3_n, \ldots, IS^{2k-1}_n$ and the symmetric clique K^2_{n+1}. These arcs and the independent sets form the pattern P (see Fig. 3(a)). Between K^2_{n+1} and K^1_{n+1}, the same pattern is built. Fig. 3(b) represents D when $n = 2$ and $k = 3$.

Theorem 4 *For any $C > 0$, there exists a digraph D s.t. $\frac{mfvs_{pn}(D)}{mfvs(D)} > C$.*

Sketch of the Proof. Let D be the digraph described in Fig. 3(a) with $n = 2$. We prove that $mfvs(D) = 4$, and that for any $(3, q)$-process strategy for D, $q \geq 2k + 3$. Taking $k > \frac{4C-3}{2}$, we get $\frac{mfvs_{pn}(D)}{mfvs(D)} \geq \frac{2k+3}{4} > C$. ■

By setting $n = p + 2$ in the digraph of Fig. 3(a) (details in [2]), we get:



(a) D (b) D when $n = 5$.

Figure 4: Symmetric digraph D of Lemma 2 (a) and D when $n = 5$ (b).

Theorem 5 *For any $C > 0$ and any integer $p \geq 0$, there exists a digraph D such that $\frac{mfvs_{pn+p}(D)}{mfvs(D)} > C$.*

The digraph described in proof of Theorem 4 has process number 3 while $\frac{mfvs_{pn}(D)}{mfvs(D)}$ is unbounded. Lemma 1 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

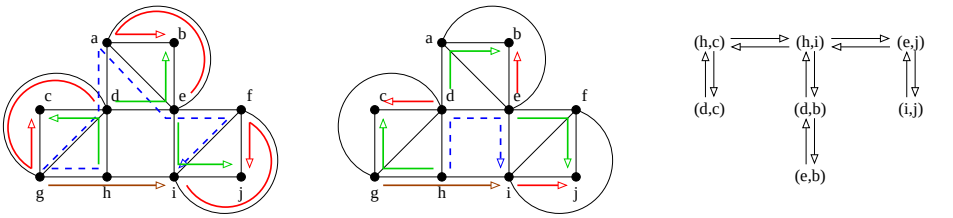
We address the behaviour of $\frac{mfvs_{pn}(D)}{mfvs(D)}$ for symmetric digraphs D. Note that the behaviours of pn_q, pn_{mfvs}, and the different ratios, have been already studied in Sec. 3 for symmetric digraphs with bounded process number. Due to lack of space the proof of Lemma 1 is omitted and can be found in [2].

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

Lemma 2 *$\forall \varepsilon > 0$, there exists a symmetric digraph D s.t. $\frac{mfvs_{pn}(D)}{mfvs(D)} \geq 3\text{-}\epsilon$.*

Sketch of the Proof. Let D be the symmetric digraph of Fig. 4(a). 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 $y_1, \ldots, y_n, v$. In D, there are two symmetric arcs between x_i and y_j, and between z_i and y_j, $i = 1, \ldots, n$, $j = 1, \ldots, n$, if and only if $j \geq i$. Furthermore the two right nodes of $K_{1,2}$ and nodes of IS_n^1 form a complete symmetric bipartite subgraph (the same construction for $K_{2,1}$ and IS_n^2). The symmetric digraph of Fig. 4(b) represents D when $n = 5$. We prove that $mfvs(D) = n + 4$, and that, any $(n+1, q)$-process strategies must cover at least $3n + 2$ nodes, that is $q \geq 3n + 2$. Taking $n > \frac{10}{\epsilon} - 4$, we get $\frac{mfvs_{pn}(D)}{mfvs(D)} \geq \frac{3n+2}{n+4} \geq 3 - \epsilon$. ■

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



(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 5: 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. Fig. 5(a) depicts the old set of routes S_1, Fig. 5(b) the new set S_2, and Fig. 5(c) the dependency digraph.

5 Processing Game out of Routing Reconfiguration

The *routing reconfiguration problem* occurs in connection-oriented networks such as telephone, MPLS, or WDM [2, 4, 5, 6, 16, 17]. 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. To avoid confusion, we assume here that each arc of the network has capacity one, and that each connection requires one unit of capacity. Consequently, no two paths can share the same arc (valid assumption in WDM networks). 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 edges 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). Furthermore, 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 Fig. 5 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. 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 total number of disrupted connections [11], and (2) to minimize the maximum number of concurrent interruptions [4, 5, 16, 17]. Following [5, 11], these two problems can be expressed through the theoretical game described in this paper, on the dependency digraph [11]. 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. Fig. 5 shows an instance of the reconfiguration problem and its corresponding dependency digraph. In Fig. 5(c), there is an arc from vertex (d, c) to vertex (h, c), because the new route used by connection (d, c) (Fig. 5(b)) uses resources seized by connection (h, c) in the initial configuration (Fig. 5(a)). Other arcs are built in the same way. The next theorem proves the equivalence between instances of the reconfiguration problem and dependency digraphs. Due to the lack of space, the proof can be found in [2].

Theorem 6 *Any digraph D is the dependency digraph of an instance of the routing reconfiguration problem.*

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, Th. 6 shows the relevance of studying these problems through the notion of dependency digraph.

A feasible reconfiguration may be defined by a (p, q)-process strategy for the corresponding dependency digraph. Problem (1) is equivalent to minimizing q (Sec. 1.1) and Problem (2) is similar to the one of minimizing p (Sec. 1.1). Consider the dependency digraph D of Fig. 5. From Sec. 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 (Fig. 1).

It is now easy to make the relation between tradeoffs 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.

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 heuristic to compute (p, q)-process strategies.

References

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

[2] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse. Tradeoffs when optimizing lightpaths reconfiguration in WDM networks. RR 7047, INRIA, 2009.

[3] D. Coudert, F. Huc, and D. Mazauric. A distributed algorithm for computing and updating the process number of a forest. In *22nd Int. Symposium on Distributed Computing (DISC)*, pages 500–501, 2008.

[4] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J-S. Sereni. Routing reconfiguration/process number: Coping with two classes of services. In *13th Conf. on Optical Network Design and Modeling (ONDM)*, 2009.

[5] D. Coudert, S. Perennes, Q.-C. Pham, and J.-S. Sereni. Rerouting requests in wdm networks. In *AlgoTel'05*, pages 17–20, mai 2005.

[6] D. Coudert and J-S. Sereni. Characterization of graphs and digraphs with small process number. Research Report 6285, INRIA, September 2007.

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

[8] 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.

[9] F. Fomin and D. Thilikos. An annotated bibliography on guaranteed graph searching. *Theo. Comp. Sci.*, 399(3):236–245, 2008.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[11] N. Jose and A.K. Somani. Connection rerouting/network reconfiguration. In *Design of Reliable Communication Networks*. IEEE, 2003.

[12] M. Kirousis and C.H. Papadimitriou. Searching and pebbling. *Theoretical Comp. Sc.*, 47(2):205–218, 1986.

[13] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. Assoc. Comput. Mach.*, 35(1):18–44, 1988.

[14] 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, 1978.

[15] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Th. Ser. B*, 35(1):39–61, 1983.

[16] F. Solano. Analyzing two different objectives of the WDM network reconfiguration problem. In *IEEE Global Communications Conference (Globecom)*, 2009.

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