Tradeoffs when optimizing Lightpaths Reconfiguration in WDM networks

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

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Tradeoffs when optimizing Lightpaths Reconfiguration in WDM networks

Nathann Cohen — David Coudert — Dorian Mazauric — Napoleão Nepomuceno — Nicolas Nisse

N° 7047

September 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7047--FR+ENG

Tradeoffs when optimizing Lightpaths Reconfiguration in WDM networks

Nathann Cohen* , David Coudert* , Dorian Mazauric* , Napoleão Nepomuceno* , Nicolas Nisse*

Thème COM — Systèmes communicants
Projets Mascotte

Rapport de recherche n° 7047 — September 2009 — 22 pages

Abstract: In this report, we study the problem of rerouting a set of lightpaths in WDM networks. The reconfiguration issue arises for instance when it is necessary to improve the usage of resources or when a maintenance operation is planned on a particular link of the network. In order to avoid service interruptions, old lightpaths should not be torn down before the new ones are set up. However, this may not be possible since establishing the new routes of lightpaths may require the release of resources previously seized by old routes. Then it could be important for the operator to minimize 1) the total number of temporarily disrupted lightpaths, and/or 2) the number of concurrent disrupted lightpaths. 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. Finally, we investigate instances from various networks through simulations.

Key-words: Reconfiguration, WDM, process number.

This work was partially funded by Région PACA, ANR AGAPE, ANR JCJC DIMAGREEN, and European project IST FET AEOLUE.

* MASCOTTE, INRIA, I3S, CNRS, Univ. Nice Sophia, Sophia Antipolis, France. firstname.lastname@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Compromis dans la reconfiguration de connexions dans les réseaux WDM

Résumé : Ce papier étudie le problème du re-routage de connections dans les réseaux WDM. Ce problème apparaît par exemple lorsque l'amélioration de l'utilisation des ressources devient nécessaire, ou lorsqu'une opération de maintenance est planifiée sur un lien du réseau. Dans le but d'éviter des interruptions de service, les anciennes routes ne devraient *a priori* pas être interrompues avant que les nouvelles routes ne soient établies. Cependant, cela n'est pas toujours possible puisque l'établissement de certaines nouvelles routes peut nécessiter la libération de ressources utilisées par d'anciennes routes. Dans ce contexte, il est important pour l'opérateur du réseau de minimiser 1) le nombre total de connections interrompues temporairement, et/ou 2) le nombre maximum de connections interrompues simultanément. Dans ce papier, nous étudions le compromis entre ces deux objectifs. Plus précisèment, nous montrons que pour certaines instances, minimiser l'un, altère arbitrairement la valeur de l'autre. De plus, nous prouvons que cela est possible même en se restreignant à des topologies simples, mettant également en exergue des classes d'instances pour lesquelles de bons compromis sont atteignables.

Mots-clés : Reroutage, process number, vertex separation, largeur de chemins

1 Introduction

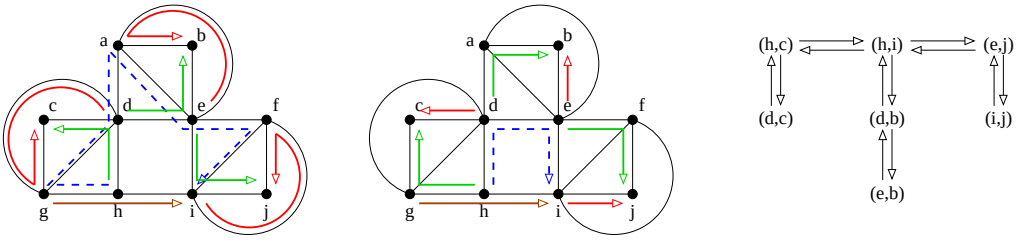
In Wavelength Division Multiplexing (WDM) backbone networks, a *connection* is a end-to-end logical service provided to a client that corresponds to the establishment of a *lightpath*. The problem of determining the set of lightpaths fulfilling all clients connections, called a *configuration* of the WDM network, has been a challenging issue for many years [24, 25, 23, 27, 28, 32, 6, 26]. In particular, critical issues are to satisfy all changes in the clients requirements (e.g., addition/termination of connections), and to ensure the service continuity when the network is facing topological modifications (e.g., failures, maintenance operations).

Changes in the clients requirements are usually dealt with using an online and dynamic process. In particular, greedy algorithms such as shortest paths computation are used to determine the lightpath to set up for serving a new connection. Terminating connections are simply torn down. Although such online processes are quite convenient in practice, they may lead to a poor usage of the overall network resources and eventually cause to refuse new connections [20, 13]. Therefore, network operators have to regularly improve the usage of resources, and so to change the network configuration, using offline methods.

Similarly, when a maintenance operation is planned on a network link, lightpaths using this link in the current configuration have to be rerouted for the duration of this operation. The operator can schedule this by using offline processes.

Such an offline re-optimization process requires to answer two questions: (1) *"how to compute the new configuration knowing the current one?"* and (2) *"how to perform the effective switching of lightpaths from the current configuration to the target one?"*. More precisely, different lightpaths will be assigned to some connections in the new configuration, and each lightpath change may induce traffic disturbance for the corresponding connection. Hence, the new configuration should be chosen in such way it induces as few lightpath changes as possible. Furthermore, it may be not possible to set up all new lightpaths before tearing down all the old ones. An important issue is to determine the best way of setting up and tearing down lightpaths to move from the current configuration to the new one while minimizing traffic disturbance.

Above questions arise in several kind of circuit-switched networks such as telephone [1, 14] or MPLS [16, 4, 18]. In the context of WDM network, heuristics and exact algorithms, mainly based on large ILPs, have been proposed for the first question [2]. The second question, that was let explicitly open for a long time [2], has only been considered recently [15, 8, 7, 31]. Some authors have also addressed both questions jointly. The most common approach is the Mote-To-Vacant (MTV) scheme that consists, starting from the current configuration, in determining a sequence of lightpath set up / tear down operations, allowing to reach a suitable configuration [5, 22, 19]. However, the MTV scheme fails when the network has not enough free resources. In this paper, we focus on the second question, namely the lightpaths reconfiguration problem in WDM networks.



(a) Initial lightpaths configuration C_1.

(b) New lightpaths configuration C_2.

(c) Dependency digraph D.

Figure 1: Example of an 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 established. Fig. 1(a) depicts the initial configuration C_1, Fig. 1(b) the new configuration C_2, and Fig. 1(c) the dependency digraph from C_1 to C_2.

1.1 Lightpaths Reconfiguration Problem.

The lightpaths reconfiguration problem consists in switching a set of connections from the old lightpaths configuration to a new pre-determined one. The problem can be stated as follows:

input: a network with a set of connections, an old lightpath and a new lightpath for each connection.

output: a sequence of *set up* and *tear down* operations resulting in the establishment of all new lightpaths while all old ones have been torn down, under the constraint that the connections are switched one by one to their final routes.

The favorable situation during a reconfiguration step occurs when all resources required by a new lightpath are released before the corresponding old lightpath is torn down. In this case, the new route using available resources is established before effectively switching the old lightpath (*Make-before-Break*). However, it may happen during the reconfiguration that a new lightpath to be set up requires an old lightpath to be torn down before, leading to deadlock. For instance, in the example depicted in Fig. 1, it is not possible to apply the Make-before-Break policy, because of some cyclic dependencies (e.g., between connections (h,i) and (d,b)). In such a case, a lightpath must be interrupted before establishing the new route (*Break-before-Make*) introducing traffic disruption. When such interruptions cannot be avoided, it may be desirable to minimize the total number of temporarily disrupted connections [15]. We refer to it as the MIN-TOTAL-DISRUPTION problem. Another possible objective for the network operator is to minimize the maximum number of concurrent interruptions [8, 7, 31, 30]. Following [30], we refer to this problem as the MIN-MAX-DISRUPTION problem.

As an example, a way to reconfigure the instance depicted in Fig. 1 may be to interrupt connections $(h, c), (d, b), (e, j)$, then set up the new lightpaths of all other connections, tear down their old lightpaths, and finally, set up the new lightpaths of connections $(h, c), (d, b), (e, j)$. Such a strategy interrupts a total of 3 connections. Another strategy may consists of interrupting the connection (h, i), then sequentially: interrupt connection (h, c), reconfigure (d, c) in a Make-before-Break manner, set up the new lightpath of (h, c), then reconfigure in the same way first (d, b) and (e, b), and then (e, j) and (i, j). Finally, set up the new lightpath of (h, i). The second strategy implies the interruption of 4 connections, but at most 2 connections are interrupted simultaneously. Actually, the first strategy is optimal for the MIN-TOTAL-DISRUPTION problem while the second one is optimal for the MIN-MAX-DISRUPTION problem.

Some natural questions arise here. How can we combine both objectives? For instance, is there any way to reconfigure the instance of Fig. 1 with only 3 interruptions while at most 2 connections are interrupted simultaneously? It is easy to check it is not possible, and Theorem 4 proves a more general result.

1.2 Objectives and results.

In this paper, we consider the tradeoff between both these conflicting objectives. We need some notations. Let $\mathcal{I}$ be an instance of the reconfiguration problem. Throughout the paper, $MFVS(\mathcal{I})$ denotes the optimal solution of the MIN-TOTAL-DISRUPTION problem on $\mathcal{I}$, i.e., it denotes the smallest total number of interruptions needed for the reconfiguration. Also, $PN(\mathcal{I})$ denotes the optimal solution of the MIN-MAX-DISRUPTION problem on $\mathcal{I}$, i.e., it denotes the smallest maximum number of concurrent interruptions needed for the reconfiguration. Moreover, $MFVS_{PN}(\mathcal{I})$ denotes the minimum total number of connections that have to be disrupted during the reconfiguration, under the constraint that the maximum number of concurrent interruptions is $PN(\mathcal{I})$. Finally, let $PN_{MFVS}(\mathcal{I})$ denote the smallest maximum number of concurrent interruptions that occur during a reconfiguration while minimizing the total number of interruptions, i.e., interrupting $MFVS(\mathcal{I})$ connections.

We start by giving two general results on the reconfiguration problems (Theorem 1) and on the MIN-MAX-DISRUPTION problem (Theorem 2). Then, the main topic of this paper concerns the ratios $\lambda(\mathcal{I}) = \frac{MFVS_{PN}(\mathcal{I})}{MFVS(\mathcal{I})}$ and $\mu(\mathcal{I}) = \frac{PN_{MFVS}(\mathcal{I})}{PN(\mathcal{I})}$. We first prove that neither λ nor μ are bounded in general. More precisely, for any $C > 0$, we exhibit an instance $\mathcal{I}$ such that $\lambda(\mathcal{I}) > C$ (Theorem 4) and an instance $\mathcal{J}$ with $PN(\mathcal{J}) = 3$ such that $\mu(\mathcal{J}) > C$ (Theorem 5). Since it is interesting for an operator to optimize the MIN-MAX-DISRUPTION problem without degrading the total number of disrupted connections, we then focus on λ. We exhibit a class of instances such that $\lambda(\mathcal{I}) \leq PN(\mathcal{I})$ for any instance $\mathcal{I}$ in this class (Lemmas 1 and 2). Then, we consider networks consisting of a directed path when wavelength conversion is not allowed. Finally, we prove that all previous general results apply to this particular class of simple instances (Theorem 6): in particular, no good tradeoff may be expected. Finally, we consider the parameter λ in instances from various networks through simulations.

Related Work.

The concept of rerouting has originally introduced in the context of circuit-switched telephone networks [1, 14]. This problem has also been tackled in the context of WDM networks [15, 8, 7, 31]. A classical approach to handle the reconfiguration problem is based on the Move-to-Vacant scheme [5, 22, 19]. In a more recent work [15], Jose and Somani introduced the notion of *dependency digraph* to propose heuristics to tackle the MIN-TOTAL-DISRUPTION problem. Using this notion, Coudert et al. [8] expressed the MIN-MAX-DISRUPTION problem as a cops-and-robber game, similar to the game-theoretical model of the *pathwidth* of a graph [29, 17]. Coudert et al. [7] and Solano [30] take advantage of this model to propose heuristics for the MIN-MAX-DISRUPTION problem.

The reconfiguration problem also appears in Multi-Protocol Label Switching (MPLS) networks [16, 3, 4, 18]. In [18], Klopfenstein proposes a Linear Program for computing the new routes in such a way that the reconfiguration phase can be done without disturbance. Józsa and Makai present some sufficient conditions over the links' capacities for allowing a rerouting process without service interruptions [16].

2 Model and definitions

Following [15, 8], the MIN-TOTAL-DISRUPTION and MIN-MAX-DISRUPTION problems can be expressed as a theoretical game on the dependency digraph [15]. Given an initial lightpaths configuration and the new configuration we want to reach, 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 lightpath of connection v uses resources that are needed by the new lightpath of connection u. Fig. 1 shows an example of an instance of the reconfiguration problem and corresponding dependency digraph. In Fig. 1(c), there is an arc from vertex (d, c) to vertex (h, c), because the new lightpath used by connection (d, c) (Fig. 1(b)) uses resources seized by connection (h, c) in the initial configuration (Fig. 1(a)). Other arcs are built in the same way.

The next theorem somehow proves the equivalence between instances of the reconfiguration problem and dependency digraphs. Note that, obviously, a digraph may be the dependency digraph of various instances of the reconfiguration problem.

Theorem 1. *Any digraph is the dependency digraph of some instance of the reconfiguration problem.*

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 Fig. 2 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.

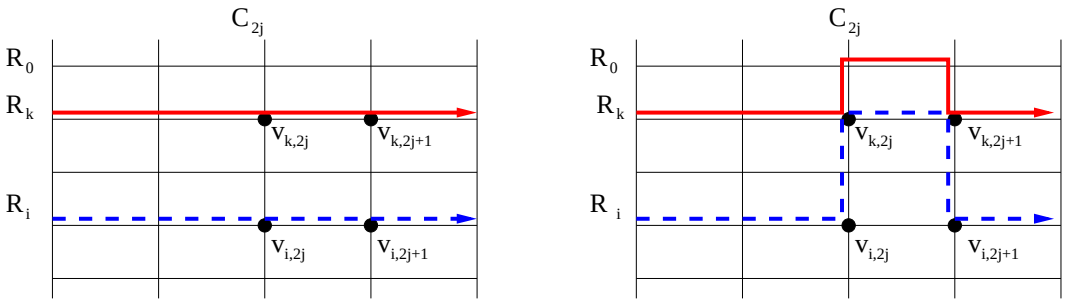


Figure 2: Scheme of the transformation in the proof of Theorem 1

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, $0 < i \leq n$, connection i, corresponding to c_i in D, occurs between $v_{i,1} \in V(G)$ the leftmost vertex of R_i and $v_{i,2m} \in V(G)$ 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, $0 < i \leq n$, 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 Fig. 2. For any $\ell \notin \{i, k\}$, $P_\ell^j = P_\ell^{j-1}$. Now, P_i^j is defined by replacing 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. □

Since any digraph may be the dependency digraph of a realistic instance of the reconfiguration problem, Theorem 1 shows the relevance of studying these problems through dependency digraph notion. In particular, the maximum out-degree of a dependency digraph is correlated with the length of associated lightpaths. Therefore, it has sense to investigating random digraphs with bounded out-degree.

2.1 MIN-TOTAL-DISRUPTION problem.

This section is devoted to express the MIN-TOTAL-DISRUPTION problem in terms of a classical invariant of the corresponding dependency digraph [15]. A reconfiguration can be done without interrupting any connection (i.e., using only the Make-before-Break policy) if and only if the corresponding dependency digraph is a Directed Acyclic Graph (DAG) [15]. To see it, the reconfiguration consists in sequentially rerouting the connections corresponding to the dependency digraph's vertices without out-neighbors. As an example, the reader can check that it is not possible to go from C_1 to C_2, in Fig. 1, without any interruption. Indeed, the corresponding dependency digraph depicted in Fig. 1(c) is not a DAG.

In the same vein, solving the MIN-TOTAL-DISRUPTION problem is equivalent to compute a *minimum feedback vertex set* (MFVS) of the corresponding dependency digraph D, i.e., to find a minimum-cardinality set of vertices whose removal makes the digraph acyclic. More precisely, given a MFVS of D, a possible reconfiguration starts by tearing down the connections represented by the vertices of the MFVS. Then, the dependency digraph of the remaining connections forms a DAG and it is thus possible to reroute them in a Make-before-Break manner. Finally, all new lightpaths of the connections in MFVS are set up. Now, let $\mathcal{I}$ be an instance of the reconfiguration problem, and let D be the corresponding dependency digraph. From the above discussion, $MFVS(\mathcal{I}) = mfvs(D)$ where $mfvs(D)$ denotes the cardinality of a MFVS of D. Moreover, the problem of computing $mfvs$ is well known to be NP-complete and APX-complete [12], and the complexity of the MIN-TOTAL-DISRUPTION problem follows.

2.2 MIN-MAX-DISRUPTION problem.

The MIN-MAX-DISRUPTION problem can be modeled by a game using agents on the dependency digraph D [8]. In this game, interrupting a connection is represented by placing an agent on the corresponding vertex in D. A vertex of D is said *processed* when the corresponding connection has been rerouted. Given a dependency digraph D, a *process strategy* on D consists of a sequence of the following three operations that results in processing all vertices of D:

- R_1 Place an agent at a vertex v of D (tear down the old lightpath of connection v);
- R_2 Remove an agent from a vertex v of D if all its out-neighbors are either processed or occupied by an agent, and process v (set up the new lightpath of connection v when needed resources are available);
- R_3 Process an unoccupied vertex v of D if all its out-neighbors are either processed or occupied by an agent (set up the new lightpath of v and tear down the old one)

The number of agents used by a process strategy is the maximum number of agents occupying the vertices of D over any step of the strategy. A *p-process strategy* for D is a strategy which processes D using p agents, and the *process number* of a digraph D, denoted by $pn(D)$, is

the smallest p such that a p-process strategy for D exists. A process strategy for D using $pn(D)$ agents is said *optimal*.

During a process strategy, the set of vertices actually occupied by an agent corresponds to the set of the disrupted connections. Clearly, given an instance $\mathcal{I}$ of the reconfiguration problem and the corresponding dependency digraph D, $PN(\mathcal{I}) = pn(D)$. In Sec. 1.1, we have described a 3 and a 2-process strategy for the example of Fig. 1(c). One can easily check that the digraph of Fig. 1(c) has process number 2.

While digraphs with process number 0, 1, and 2 can be recognized in polynomial time [9], computing the process number is NP-complete in general [8] and not APX (i.e., admitting no approximation polynomial-time algorithm up to a constant factor, unless $P = NP$) [11]. In [10], a distributed polynomial-time algorithm is described that computes the process number of any digraph in the class of trees with symmetric arcs. The first heuristic for computing the process number of any dependency digraph is described in [7]. A possible approach for computing the process number, proposed by Solano [30], consists of two phases: 1) finding the disruption set, i.e., to find the subset of vertices of the dependency digraph at which an agent will be placed, and 2) sorting the disruptions, i.e., given the set of vertices computed in Phase 1, deciding the order in which the agents will be placed 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 [30]. We disprove this conjecture.

Theorem 2. *If the set of disrupted connections is given, then the reconfiguration problem remains NP-complete and not APX.*

Proof. Let $D = (V, A)$ be a symmetric digraph with $V = \{u_1, \ldots, u_n\}$. Let $D' = (V', A')$ be the digraph with $V' = V \cup \{v_1, \ldots, v_n\}$ obtained from D by adding two symmetric arcs between u_i and v_i for any $i \leq n$. Obviously, any process strategy for D' must place an agent either at u_i or v_i for any $i \leq n$. Moreover, it is easy to show that there exists an optimal process strategy for D' such that the set of occupied vertices is V. Indeed, if some step of a process strategy for D' consists in placing an agent at some vertex v_i, then the strategy can easily be transformed by placing an agent at u_i instead.

Now, consider the problem of computing an optimal process strategy for D' when the disruption set is constrained to be V. It is easy to check that this problem is equivalent to the one of computing an optimal path-decomposition of the underlying undirected graph of D which is NP-complete [21] and not APX [11]. □

3 Tradeoffs

This section is devoted to prove some tradeoffs between MIN-TOTAL-DISRUPTION and MIN-MAX-DISRUPTION problems. More precisely, we are interesting in bounding the ratios λ and μ defined in Section 1.2. We prove that both problems are conflicting, i.e., λ

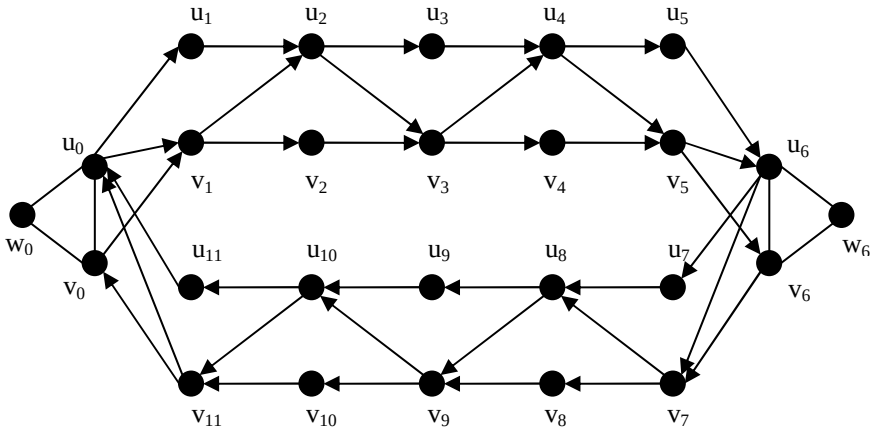


Figure 3: Digraph D described in proof of Th. 5 with $n = 3$ and $\lambda(D) = \frac{9}{4}$.

and μ are not bounded in general. On the positive side, we prove that λ is bounded in the class of instances of the reconfiguration problem whose dependency digraphs are symmetric.

Let $\mathcal{I}$ be an instance of the reconfiguration problem, and let D be its dependency digraph. In the following, $pn_{mfvs}(D)$ denotes the smallest number of agents used by a process strategy for D subject to the fact that the number of occupied vertices is minimized. Let $\mu(D) = \frac{pn_{mfvs}(D)}{pn(D)}$. Clearly, $\mu(D) = \mu(\mathcal{I})$. Conversely, let $mfvs_{pn}(D)$ be the smallest total number of vertices that must be occupied by an optimal process strategy for D, and let $\lambda(D) = \frac{mfvs_{pn}(D)}{mfvs(D)}$. Again, $\lambda(D) = \lambda(\mathcal{I})$.

Theorem 3. *The problems of determining pn_{mfvs}, $mfvs_{pn}$, λ and μ are NP-complete and not APX.*

Proof. Note that Theorem 2 proves that pn_{mfvs} is NP-complete and not APX. Indeed, in the class of graphs D' defined in the proof of Theorem 2, $pn(D') = pn_{mfvs}(D') = pwd(D)$ (where the relationship between D and D' is described in the proof of Theorem 2, and *pwd* denotes the pathwidth of D).

Similarly, $mfvs_{pn}$ is NP-complete and not APX. To see this, let D be any n-node digraph, and let D^* be the digraph obtained by adding symmetric arcs between one vertex of D and one vertex of a symmetric $n+1$-node clique. Then, it is easy to show that $pn(D^*) = n$ and $mfvs_{pn}(D^*) = mfvs(D) + n$.

To show that μ is NP-complete, let G^0 be any strongly connected n-node digraph. For any v in G^0, let us add a vertex w_v with two arcs (v, w_v) and (w_v, v), let G^1 be the resulting graph and let V^1 be the set of vertices of G^1 that corresponds to a vertex in G^0. Let H be a symmetric directed star with $n+1$ branches each of which containing two vertices (excluding

the central node). Fig. 1(c) shows an example with $n = 3$ and the central node (h, i). Let ℓ be a neighbor of the central node r. Now, we build G as follows. Start with three copies of G^1, and let a, b and c be three vertices, in each copy of V^1 . Finally, add three vertices x, y and z together with symmetric arcs between ℓ and x, ℓ and y, ℓ and z, a and x, b and y, and c and z. Let us do some trivial remarks: (1) the set of vertices that consists of the vertices of the three copies of G^0 and the neighbors of r is a MFVS of G, and thus $mfvs(G) = 4n+1$. (2) Moreover, r does not belong to a MFVS of G. Hence, to clear r by occupying at most $mfvs(G)$ vertices, all neighbors of r must be simultaneously occupied. This leads to $pn_{mfvs} \geq n + 1$. And (3), because G^0 is strongly connected, $pn(G) \geq pn(G^1) + 1$. Indeed, after the first copy of G^1 has been cleared and before the second copy of G^1 is cleared, then at most one agent must occupy some vertex in $V(H) \cup \{x, y, z\}$. Thus, for the second copy of G^1 to be cleared, $pn(G^1)$ other agents must be available. To conclude, it is sufficient to remark that $pn(G) \leq pn(G^1) + 1 = pwd(G^0) + 1$ where $pwd(G^0)$ denotes the pathwidth of the undirected graph underlying G^0 (see the proof of Theorem 2 for the equality) and $pn_{mfvs} \leq n+1$ (corresponding strategies are easy to compute). Hence, $\mu = \frac{pwd(G^0)+1}{n+1}$ which is NP-complete and APX to compute. □

Theorem 4. *For any $C > 0$, there is an instance $\mathcal{I}$ of the reconfiguration problem such that $\mu(\mathcal{I}) > C$.*

Proof. By Theorem 1, it is sufficient to prove that there is a digraph D such that $\mu(D) > C$. Let D be a symmetric directed star with $n \geq 2$ branches each of which containing two vertices (excluding the central node). Fig. 1(c) shows an example with $n = 3$ and the central node (h, i). From [9], $pn(D) \geq 2$ (D is not a DAG and any minimum feedback vertex set of D contains n nodes). Then it is easy to check that $pn(D) = 2$: place an agent at the central node v, and then successively place an agent at a vertex x adjacent to v, process the other vertex adjacent to x and then process x itself relieving the agent on it, until all vertices adjacent to v are processed, and finally process v. Moreover, the single MFVS of D is the set X of the n vertices adjacent to v. It is easy to check that the single process strategy occupying only the vertices of X consists in placing agents at any vertex of X. No agent can be removed while all agents have not been placed. Hence, $pn_{MFVS}(D) = n$. Taking $n > 2C$, we get $\mu(D) > C$ and so $\mu(\mathcal{I}) > C$. □

Theorem 5. *For any $C > 0$, there is an instance $\mathcal{I}$ of the reconfiguration problem such that $\lambda(\mathcal{I}) > C$.*

Proof. By Theorem 1, it is sufficient to prove that there is a digraph $D = (V, A)$ such that $\lambda(D) > C$. Let $n \geq 2$. We set $V = \{w_0, w_{2n}\} \cup \{u_i, 0 \leq i \leq 4n-1\} \cup \{v_i, 0 \leq i \leq 4n-1\}$ and:

- $\forall i \in \{0, 2n\}$, then $\{(u_i, v_i), (v_i, u_i)\} \in A$;
- $\forall i \in \{0, 2n\}$, then $\{(u_i, w_i), (u_i, w_i)\} \in A$;
- $\forall i \in \{0, 2n\}$, then $\{(v_i, w_i), (w_i, v_i)\} \in A$;

- $\forall i \in \{i, 0 \leq i \leq 4n-1\}$, then $(u_i, u_{i+1}) \in A$;
- $\forall i \in \{i, 0 \leq i \leq 4n-1\}$, then $(v_i, v_{i+1}) \in A$;
- $\forall i \in \{2i, 0 \leq i \leq 2n-1\}$, then $(u_i, v_{i+1}) \in A$;
- $\forall i \in \{2i+1, 0 \leq i \leq 2n-1\}$, then $(v_i, u_{i+1}) \in A$;
- $(w_0, v_1) \in A$;
- $(v_{4n-1}, w_0) \in A$;
- $(w_{2n}, v_{2n+1}) \in A$;
- $(v_{2n-1}, w_{2n}) \in A$.

See Fig. 3 for an example of such a graph for $n = 3$.

First, $mfvs(D) = 4$ because any MFVS of D must contains at least 2 vertices in each of the symmetric triangles (u_0, v_0, w_0) and (u_{2n}, v_{2n}, w_{2n}). Moreover, $\{u_0, v_0, u_{2n}, v_{2n}\}$ is a feedback vertex set of D.

We then prove that $pn(D) = 3$. First, note that, for any process strategy for D, there must be a step when two agents are occupying two vertices in $\{u_0, v_0, w_0\}$, resp., $\{u_{2n}, v_{2n}, w_{2n}\}$. Hence, let us assume w.l.o.g. that the first such step occurs when 2 vertices in $\{u_0, v_0, w_0\}$ are occupied simultaneously. At such a step, if no other vertex of D is currently occupied by an agent, then a third agent is necessary to perform the next operation. Hence, $pn(D) > 2$. We now describe a 3-process strategy for D. First, 3 agents are placed on u_0, v_0, and v_1. Then, w_0 is processed, and $\{u_{4n-1}, v_{4n-1}, \ldots, u_{2n+1}, v_{2n+1}\}$ are processed sequentially (without extra agent). We process v_0 removing the agent from it. We place an agent on u_2. Then, for $0 \leq j \leq n-2$, the agent at u_{2j} is released and placed at v_{2j+3}, and then the agent at v_{2j+1} is released and placed at $u_{2(j+2)}$. Finally, the agent at v_{2n-3} is released and placed at u_{2n}, and the agent at u_{2n-2} is released and placed at v_{2n}.

To conclude, we prove that the strategy described above minimizes the total number of occupied vertices among all 3-process strategies for D. W.l.o.g we suppose that w_0 is processed before w_{2n}. We consider the case when no agent is placed at w_0. If v_0 is the node not covered by an agent (so u_0 and w_0 are necessary covered by an agent), it is exactly the same case because v_0 and w_0 have exactly the same neighborhood. If u_0 is not covered by agent (so v_0 and w_0 are necessary covered by an agent), then it is necessary to use 4 agents. Thus when w_0 is processed, we have two agents on u_0, and on v_0. Furthermore since w_{2n} is not yet processed, v_1 is necessary covered by an agent because otherwise v_1 has to be yet processed and so u_2 and v_2 have also to be yet processed or covered by an agent. It is impossible because there exist two nodes disjoint paths from u_2 to w_{2n} and from v_2 to w_{2n} and so we need at least two agents to disconnect u_2 and v_2 from w_{2n} because by assumption w_{2n} is not yet processed. Thus v_1 is covered by an agent and any node $x \in \{u_i, 1 \leq i \leq 2n\} \cup \{v_i, 2 \leq i \leq 2n\} \cup \{w_{2n}\}$ is not yet processed. We can assume that the process strategy starts to put three agents on u_0, v_0, and v_1 because doing it, we

process $\{u_{4n-1}, v_{4n-1}, \ldots, u_{2n+1}, v_{2n+1}\}$ sequentially without extra agent. Then we process v_0 removing the agent from it. Then two nodes can be covered by the free agent in order to continue the 3-process strategy: u_1 and u_2. To minimize $mfvs_{pn}(D)$, we choose u_2 and we put the free agent on it. Then we process u_1 (without agent on it), and we process u_0 removing the agent from it. Using the same reasoning, we get that for $i \in [1, 2n-1]$, exactly one node among u_i and v_i is covered by an agent. Finally, we have to put two agents on u_{2n} and v_{2n} in order to process w_{2n}. Thus we have $mfvs_{pn}(D) = 2n+3$. Taking $n > \frac{4C-3}{2}$, we get $\lambda(D) > C$ and so $\lambda(\mathcal{I}) > C$. □

The digraphs we built to prove Theorem 5 have process number 3 while λ is unbounded. The following lemma shows that, in the class of symmetric digraphs with bounded process number, λ is bounded.

Lemma 1. *For any symmetric digraph D, $\lambda(D) \leq pn(D)$.*

Proof. Let S be a process strategy for $D = (V, E)$, using $pn(D)$ agents and occupying the fewest vertices as possible. 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, $X \cup Z$ is an independent set because it is the complementary of a MFVS. Since the vertices not occupied by S have all their neighbors occupied, $W \cup Z$ is an independent set. Given $S \subseteq V$, $N(S)$ denotes the set of neighbors of the vertices in S.

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. 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 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

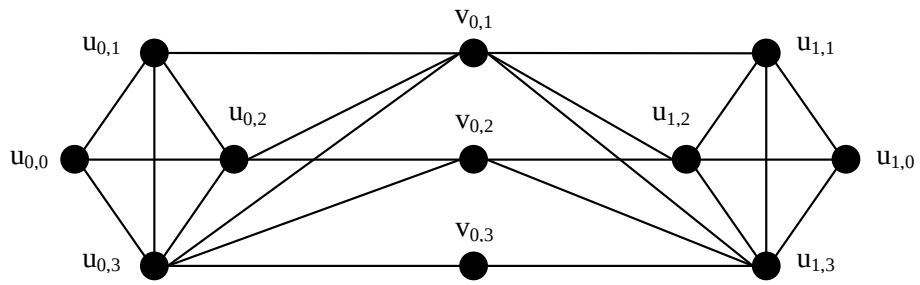


Figure 4: Sym. digraph G of Lem. 2 with $n = 3$, $k = 2$ and $\lambda(G) = \frac{4}{3}$.

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, $|N_{i+1} \setminus N_i| < pn(D)$.

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

Lemma 2. *$\forall \varepsilon > 0$, there exists a symmetric digraph D such that $\lambda(D) \geq 2 - \epsilon$.*

Proof. Let $G = (V, E)$ be a symmetric directed graph. Let $K^0_{n+1} = (V_0, E_0), \ldots, K^k_{n+1} = (V_k, E_k)$ be $k+1$ disjoint complete graphs with $V_1 = \{u_{0,0}, \ldots, u_{0,n}\}, \ldots, V_2 = \{u_{k,0}, \ldots, u_{k,n}\}$. Let $F^0_n, \ldots, F^{k-1}_n$ be k disjoint sets of n nodes with $F^0 = \{v_{0,1}, \ldots, v_{0,n}\}, \ldots, F^{k-1} = \{v_{k-1,1}, \ldots, v_{k-1,n}\}$. We set $V = \cup_{i=0}^k V_i \cup_{i=0}^k F_i$, and $E = \cup_{i=0}^k E_i \cup_{i=0}^{k-1} \cup_{j=1}^n \{(u_{i,j}, v_{i,h}), (u_{i+1,j}, v_{i,h});\ 1 \leq h \leq j\}$.

See Fig. 4 for an example of such a directed graph with $n = 3$ and $k = 2$.

We now prove that $pn(G) = n + 1$.

• $pn(G) \geq n$: k complete graphs of $n + 1$ nodes each are subgraphs of G and recall that the process number of a complete graph of size $n + 1$ is n. Since computing the process number is a subgraph closed problem, then we get the inequality.

• $pn(G) \neq n$: assume that there exists a n-process strategy for G. W.l.o.g assume that $u_{0,0}$ is processed before $u_{1,0}$ and it is never covered by an agent. When $u_{0,0}$ is processed, the nodes $u_{0,1}, \ldots, u_{0,n}$ are covered by n agents. Since $u_{1,0}$ is not processed (by assumption), then the nodes $v_{0,1}, \ldots, v_{0,n}, u_{1,1}, \ldots, u_{1,n}$ are not processed because there is a path between any node among the previous list and $u_{1,0}$. Finally it is impossible to process a node $u \in \{u_{0,1}, \ldots, u_{0,n}\}$ covered by an agent, and so impossible to remove one agent. Thus we have to use a $(n + 1)$th agent to process G.

• We now describe a $(n+1)$-process strategy for G. First we put n agents on $u_{0,1}, \ldots, u_{0,n}$, and we process $u_{0,0}$. We put an agent on $v_{0,1}$, and we process $u_{0,1}$ removing the agent from it. We have n agents located on nodes. We put an agent on $v_{0,2}$, and we process $u_{0,2}$ removing the agent from it, and so on. We have at the end $n-1$ agents on nodes $v_{0,1}, \ldots, v_{0,n-1}$ and 1 agent on $u_{0,n}$. We put an agent on $u_{1,n}$, we process $v_{0,n}$, and we process $u_{0,n}$ removing the agent from it. Then we put an agent on $u_{1,n-1}$, and we process $v_{0,n-1}$ removing the agent from it. We have n nodes occupied by agents. We do it until having n agents on $\{u_{1,1}, \ldots, u_{1,n}\}$. We process $u_{1,0}$. Here we use the same reasoning than before to have $n-1$ agents on nodes $v_{1,1}, \ldots, v_{1,n-1}$, and after to have n agents on nodes $\{u_{2,1}, \ldots, u_{2,n}\}$. We do it until process $\{u_{k,1}, \ldots, u_{k,n}\}$.

$mfvs_{pn}(G) = 3n-1$: w.l.o.g we assume that $u_{0,0}$ is processed before $u_{1,0}$. Thus we have n agents on $\{u_{0,1}, \ldots, u_{0,n}\}$. To process $u_{1,0}$ we must have n agents located on $\{u_{1,1}, \ldots, u_{1,n}\}$. If we put an agent on $u \in \{u_{1,1}, \ldots, u_{1,n}\}$ or on $\{v_{0,2}, \ldots, v_{0,n}\}$, we cannot process any node and the $n+1$ agents are used. Thus we have to put the $(n+1)$th agent on $v_{0,1}$ (process $u_{0,1}$ removing an agent from it, so we have 1 agent available). Then we have the same constraint: If we put an agent on $u \in \{u_{1,1}, \ldots, u_{1,n}\}$ or on $\{v_{0,3}, \ldots, v_{0,3}\}$, we can do nothing after. Thus we put an agent on $v_{0,2}$ (process $u_{0,2}$ removing the agent from it, so we have 1 agent available). We have to do it until $v_{0,n-1}$. After we can directly put an agent on $u_{1,n}$ processing $v_{0,n}$ without covering it by an agent. Finally $\{v_{0,1}, \ldots, v_{0,n-1}\}$ have been covered by agents. Using the same reasoning to process the nodes $u_{2,0}, \ldots, u_{k,0}$, we get that $mfvs_{pn}(G) = k(2n-1) + n$.
$mfvs(G) = n(k+1)$ because it is necessary to break all cycles in the $k+1$ disjoint complete graphs $K^0_{n+1}, \ldots, K^k_{n+1}$ and it is sufficient to break all cycles in G.

Finally we have $pn(G) = n+1$, $mfvs_{pn}(G) = k(2n-1)+n$, and $mfvs(G) = n(k+1)$. We have $\lambda(G) = \frac{k(2n-1)+n}{kn+n}$. If $n = k > \lceil 2/\varepsilon - 1 \rceil$, then $\lambda(G) \geq 2 - \varepsilon$. □

Conjecture 1. *For any symmetric digraph D, $\lambda(D) \leq 2$.*

4 Simple network topologies

We now prove that even for simple topologies like directed paths, $MFVS, PN, \mu$ and λ may be unbounded. More generally, we prove that all previous general results apply to this particular class of simple instances.

Let us consider instances of the reconfiguration problem whose underlying networks are a directed path or a directed cycle. We say that no wavelength conversion is allowed if the lightpath of a connection keeps the same wavelength all along the path. Note that, in such topologies, there is a unique path available for any connection. Therefore, in this setting, reconfiguring a connection consists in switching the wavelength used by the connection all along the lightpath. From now on, $\mathcal{H}$ denotes the set of instance of the reconfiguration problem on a directed path (or cycle) without wavelength conversion.

As an example, consider the instance depicted in Fig. 5, with 5 connections and 3 wavelengths. The figure on the left represents the initial configuration and the figure on the right

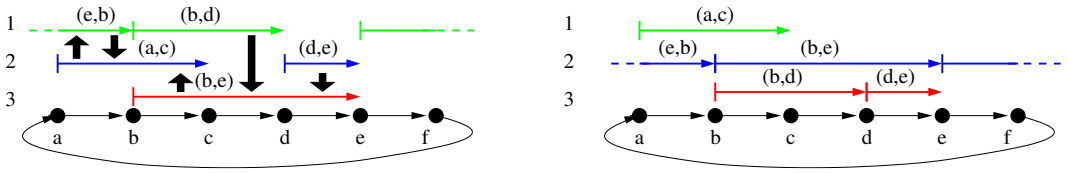


Figure 5: Example of a reconfiguration instance in $\mathcal{H}$, i.e. when the underlying network is a directed cycle and no wavelength conversion is allowed.

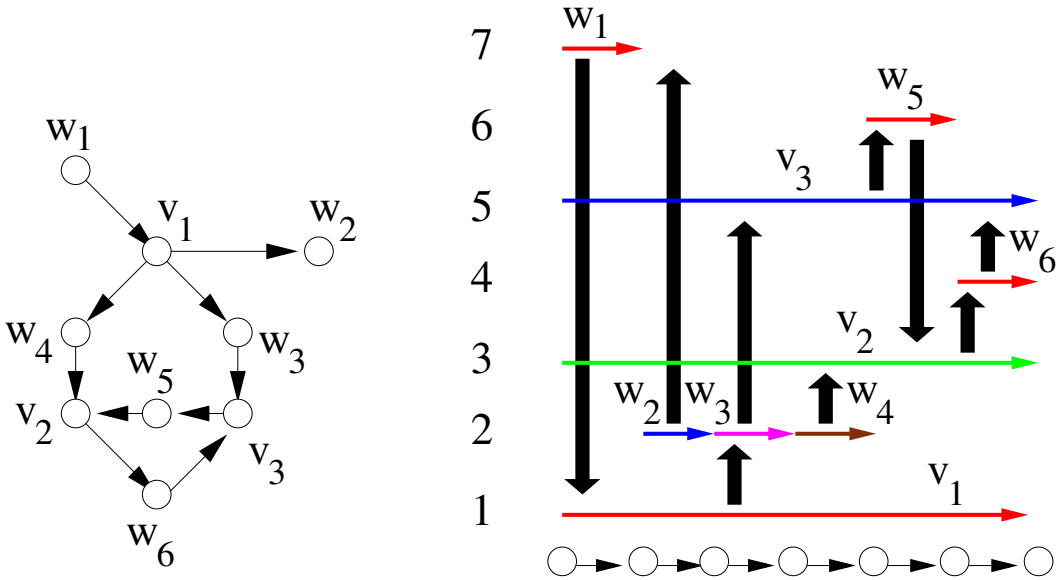


Figure 6: Example of a digraph and corresponding instance on a directed path.

represents the final one. The wild vertical arrows indicate wavelengths of new lightpaths. For example, connection (b, e) is switched from wavelength 3 to 2.

Note that, there exist some digraphs that cannot be the dependency digraph of an instance in $\mathcal{H}$.

Lemma 3. *Any digraph that contains a symmetric triangle as an induced subgraph cannot be the dependency digraph of an instance in* $\mathcal{H}$.

Proof. For purpose of contradiction, let us assume that a symmetric triangle (a, b, c) is the dependency digraph of such an instance. Then, the final wavelength ω of connection a must be the initial wavelength of b because of the arc (a, b) and ω must be the initial wavelength of c because of the arc (a, c). But then, b and c having the same intial wavelength on a directed path, their corresponding paths must be disjoint, contradicting the dependency between b and c. □

However, the next theorem shows that any digraph can be slightly modified, preserving all its properties related to the reconfiguration problems, in such a way that the modified digraph corresponds to an instance of the reconfiguration problem on a directed path without wavelength conversion.

A digraph $D = (W, F)$ is obtained by *subdividing* an arc (u, v) of a digraph $D' = (V, A)$ if $W = V \cup \{w\}$ and $F = A \cup \{(u, w)\} \cup \{(w, v)\} \setminus \{(u, v)\}$.

Theorem 6. *Let D^* be any digraph, and let D be the digraph obtained from D^* by subdividing all arcs. Then,*

- *$mfvs, pn, mfvs_{pn}$ and pn_{mfvs} are equal in D and D^*.*
- *D is the dependency digraph of an instance of the reconfiguration problem whose underlying network is a directed path, and no wavelength conversion is allowed.*

Proof. The first item is straightforward, since it is well known that subdivision of arcs neither change the process number nor the minimum feedback vertex set of a digraph [].

For the second item, we prove a more general statement. Let $D = (V, A)$ be a digraph such that V can be partitioned into two independent sets U and W (a set of vertices is independent if it induces no arcs) such that all vertices of W have in-degree and out-degree at most one. Then D is the dependency digraph of an instance of the reconfiguration problem whose underlying network is a directed path, and such that no wavelength conversion is allowed.

To see this, let $U = \{v_1, \cdots, v_n\}$ and $W = \{w_1, \cdots, w_m\}$. Let us consider the network that consists of a directed path $P = \{x_1, \cdots, x_{m+1}\}$ and $2n + 1$ available wavelengths $\{\lambda_1, \cdots, \lambda_{2n+1}\}$. For any $i \leq n$, let the initial lightpath of connection v_i be P, i.e., between x_1 and x_{m+1}, on wavelength λ_{2i-1} and its final lightpath be on wavelength λ_{2i}. For any $j \leq m$, let the initial lightpath of connection w_j be the arc (x_j, x_{j+1}) on wavelength λ_{2i} if w_j has an in-neighbor v_i and on wavelength λ_{2n+1} otherwise, and let its final lightpath be on wavelength λ_{2k-1} if w_j has an out-neighbor v_k and on wavelength λ_{2n+1} otherwise. The instance described above is clearly valid and admits D as dependency digraph.

An example of the above construction is given in Fig. 4. □

Corollary 1. *The MIN-TOTAL-DISRUPTION and MIN-TOTAL-DISRUPTION problems are NP-complete in the class of instances in $\mathcal{H}$. Moreover, all results of Section 3 remain valid when the instances in $\mathcal{H}$.*

5 Simulations

In previous sections, we prove that, in general, no tradeoff can be achieved for the MIN-TOTAL-DISRUPTION and MIN-MAX-DISRUPTION problems. That is, there exist some "bad" instances for which the total number of disruptions drastically increases when we aim at minimizing the maximum number of concurrent disruption (Theorem 5). However, there exist classes of instances for which good tradeoff can be achieved (Lemma 1).This section is devoted to evaluate the tradeoff that might be expected in various classes of instances.

Recall that neither the complexity nor any heuristic or approximation algorithms are known for the computation of λ. To evaluate this ratio on several instances, we first have used the heuristic of [7] to compute a process-strategy on these instances, leading to an upper bound of $mfvs_{pn}$, i.e., the number of vertices occupied during the obtained strategy. Second, in the class of instances where it was possible, we derived some lower bounds on $mfvs$.

We investigate the class of symmetric square-grids, for a side range in $\{1, \ldots, 20\}$ (Fig 7(a)), the digraph with process number 2 characterized in [9], for a side range in $\{4, \ldots, 280\}$ and different densities (Fig 7(b)), the class of directed random graphs, for a side range in $\{10, \ldots, 100\}$ and a probability of arcs' presence in $\{0.025, 0.05, 0.075, 0.1, 0.3, 0.5, 0.7, 0.9\}$ (Fig 7(c), Fig 7(d)), and the class of symmetric random graphs with 100 nodes and a probability of arcs' existance in $\{0.025, 0.05, 0.075, 0.1, 0.3\}$ (Fig 7(e)). In Figs. 7(a) 7(b) 7(c) 7(c), for a given size n and a class $\mathcal{C}$ of digraphs, each measure depicted corresponds to the average of the measures obtained on 20 n-node digraph in the class $\mathcal{C}$. Similarly, in Fig 7(e), for any presence-probability p, each measure depicted corresponds to the average of the measures obtained on 20 symmetric random digraphs of 100 nodes with presence probability p.

For symmetric square-grids (Fig 7(a)), heuristic of [7] gives process strategies with a number of agents almost equal to the optimal (filled lines). Recall that the process number of a $n \times n$ symmetric grid is $n + 1$ (if $n > 2$). Furthermore, the total number of nodes covered by agents through these strategies are almsot equal to the number of nodes whereas the optimal value $mfvs_{pn}$ is the number of nodes over two.

Then we apply the heuristic for digraphs with process number 2 representing the number of agents used (filled lines) and the total number of agents covered by agents (dotted lines) in Fig 7(b) for different densities.

We also apply the heuristic proposed in [7] for random digraphs with different probability $p \in \{0.025, 0.05, 0.075, 0.1, 0.3, 0.5, 0.7, 0.9\}$ representing for each p the number of agents used to proces the digraph (Fig 7(c)) and the total number of nodes covered by agents (Fig 7(d)). There is an arc from node u to node v with probability p. Note that larger is the probability larger are the values the two previous numbers.

Finally we apply the heuristic for a random symmetric graph of 100 nodes, computing also the optimal value of $mfvs$, according to a probability $p \in \{0.025, 0.05, 0.075, 0.1, 0.3\}$ (two nodes u and v are linked with probability p). See Fig 7(e).

6 Conclusion

In this paper, we have investigated, through simulations, the tradeoff between MIN-TOTAL-DISRUPTION and MIN-MAX-DISRUPTION problems (i.e., the parameter λ) in the case of instances from various networks. However, our simulations show an over-estimation of the value of λ. This is inherently due to the heuristic of [7] that we use to compute a process-strategy. Indeed, this heuristic "saves" the use of unneeded agents by *sliding an agent* (place an agent on a node v and remove an agent from an in-neighbor of v) each time it is possible. This behaviour which is highly benefic to minimize the number of used agents, leads to poor results in terms of total number of occupied vertices. Next step will be the design of a suitable heuristic that computes strategies using few agents and occupying few vertices.

Finally, we have exhibit extremal instances for which the total number of disruptions increases drastically when we aim at minimizing the maximum number of concurrent disruption (Theorem 5). However, it is likely that good tradeoffs may be achieved on practical instances. This should be checked through extensive simulations.

Acknowledgment

This work has been partially supported by région PACA, ANR AGAPE and DIMAGREEN, and European project IST FET AEOLUS.

References

[1] M. Ackroyd. Call repacking in connecting networks. *IEEE Transactions on Communications*, 27(3):589–591, March 1979.

[2] D. Banerjee and B. Mukherjee. Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Transactions on Networking*, 8(5):598–607, October 2000.

[3] S. Beker, D. Kofman, and N. Puech. Off-line MPLS layout design and reconfiguration: Reducing complexity under dynamic traffic conditions. In *International Network Optimization Conference (INOC)*, pages 61–66, October 2003.

[4] S. Beker, N. Puech, and V. Friderikos. A tabu search heuristic for the off-line MPLS reduced complexity layout design problem. In *3rd International IFIP-TC6 Networking Conference (Networking)*, volume 3042 of *Lecture Notes in Computer Science*, pages 514–525, Athens, Greece, May 2004. Springer.

[5] Xiaowen Chu, Tianming Bu, and Xiang-Yang Li. A study of lightpath rerouting schemes in wavelength-routed wdm networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 2400–2405, 2007.

[6] Xiaowen Chu and Bo Li. Dynamic routing and wavelength assignment in the presence of wavelength conversion for all-optical networks. *IEEE/ACM Trans. Netw.*, 13(3):704–715, 2005.

[7] 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)*. IEEE, 2009.

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

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

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

[11] 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.

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

[13] Ornan Ori Gerstel, Galen H. Sasaki, Shay Kutten, and Rajiv Ramaswami. Worst-case analysis of dynamic wavelength allocation in optical networks. *IEEE/ACM Trans. Netw.*, 7(6):833–846, 1999.

[14] A. Girard and S. Hurtubise. Dynamic routing and call repacking in circuit-switched networks. *IEEE Transactions on Communications*, 31(12):1290–1294, 1983.

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

[16] B. G. Józsa and M. Makai. On the solution of reroute sequence planning problem in MPLS networks. *Computer Networks*, 42(2):199 – 210, 2003.

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

[18] O. Klopfenstein. Rerouting tunnels for MPLS network resource optimization. *European Journal of Operational Research*, 188(1):293 – 312, 2008.

[19] K.-C. Lee and V.O.K. Li. A wavelength rerouting algorithm in wide-area all-optical networks. *IEEE/OSA Journal of Lightwave Technology*, 14(6):1218–1229, June 1996.

[20] Ling Li and Arun K. Somani. Dynamic wavelength routing using congestion and neighborhood information. *IEEE/ACM Trans. Netw.*, 7(5):779–786, 1999.

[21] 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.

[22] G. Mohan and C.S.R. Murthy. A time optimal wavelength rerouting algorithm for dynamic trafficin WDM networks. *IEEE/OSA Journal of Lightwave Technology*, 17(3):406–417, March 1999.

[23] Ahmed Mokhtar and Murat Azizoglu. Adaptive wavelength routing in all-optical networks. *IEEE/ACM Trans. Netw.*, 6(2):197–206, 1998.

[24] B. Mukherjee. WDM-based local lightwave networks – Part I: Single-hop systems. *IEEE Network*, 6(3):12–27, May 1992.

[25] B. Mukherjee. WDM-based local lightwave networks – Part II: Multi-hop systems. *IEEE Network*, 6(4):20–32, July 1992.

[26] B. Mukherjee. *Optical WDM Networks*. Springer, 2006.

[27] Ramu S. Ramamurthy and Biswanath Mukherjee. Fixed-alternate routing and wavelength-routed optical networks. *IEEE/ACM Trans. Netw.*, 10(3):351–367, 2002.

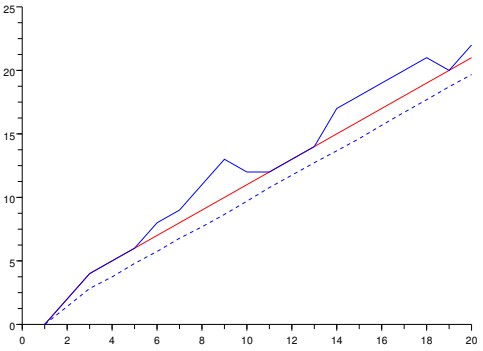
[28] Rajiv Ramaswami and Kumar N. Sivarajan. Routing and wavelength assignment in all-optical networks. *IEEE/ACM Trans. Netw.*, 3(5):489–500, 1995.

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

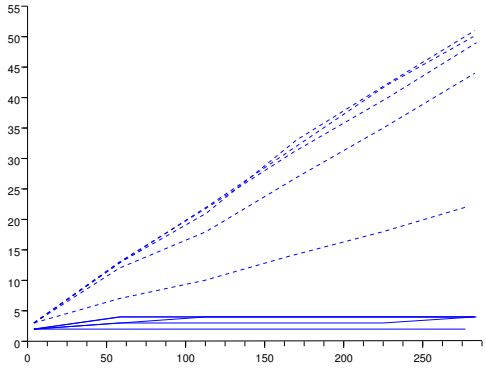
[30] F. Solano. Analyzing two different objectives of the WDM network reconfiguration problem. In *IEEE Global Communications Conference (Globecom)*, Honolulu, Hawai, USA, December 2009. to appear.

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

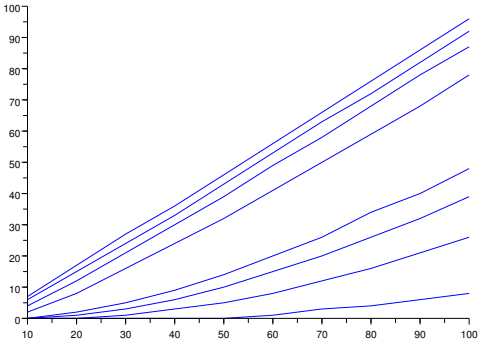
[32] Yuhong Zhu, George N. Rouskas, and Harry G. Perros. A comparison of allocation policies in wavelength routing networks. *Photonic Network Communications*, 2:265–293, 2000.



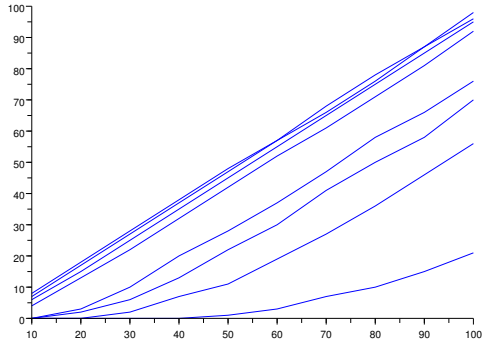
(a) symmetric square-grids: filled lines represent the number of agents used by the heuristic and the optimal value pn, dotted lines represent the square root of total number of nodes covered by agents.



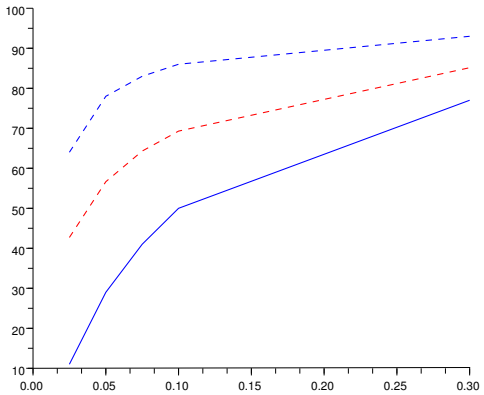
(b) digraphs with process number 2: filled lines represent the number of agents used and dotted lines represent the total number of nodes covered by agents according to the number of nodes for different densities.



(c) Random graphs: different number of agents used according to the number of nodes and the set of probabilities $\{0.025, 0.05, 0.075, 0.1, 0.3, 0.5, 0.7, 0.9\}$.



(d) Random graphs: different total number of nodes covered by agents according to the number of nodes and the set of probabilities $\{0.025, 0.05, 0.075, 0.1, 0.3, 0.5, 0.7, 0.9\}$..



(e) Random symmetric graphs: dotted line represents the number of agents used, dotted red line represents the optimal $mfvs$ and dotted blue line represents the total number of nodes covered by agents through strategies computed by the heuristic.

Figure 7: sdff.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399