WDM network re-optimization avoiding costly traffic disruptions

Fernando Solano · Michał Pióro

Published online: 23 August 2011

© The Author(s) 2011. This article is published with open access at Springerlink.com

Abstract *Network re-optimization* is a process that must be triggered periodically in order to improve the inefficient resource allocation of online routing heuristics due to the uncertainty of online lightpath demand arrivals and departures. Network re-optimization involves two tasks: a) finding new lightpaths for a (sub)set of current demands, i.e. *rerouting*, and b) migrating the current traffic to the new configuration diminishing traffic disruptions, i.e. *lightpath reconfiguration*. If not controlled, excessive traffic disruptions may be a cause of violations of clients' Service Level Agreement, which should be compensated by the network operator with penalization fees.

So far, rerouting and reconfiguration tasks of a re-optimization process have been done separately, hence, efforts in trying to achieve the best network performance (rerouting) yields to solutions incurring on unacceptable traffic disruptions (reconfiguration) and vice-versa. In this paper, given a time-disruption threshold for reconfiguring every demand, we present a novel methodology consisting on two procedures that collaboratively find the best network performance without incurring on penalization fees.

Our numerical results are extremely encouraging: in our scenarios, it is always possible to achieve an optimal routing performance without incurring on penalization fees.

Keywords Network reoptimization · Network reconfiguration · Routing · Optical networks · WDM · Lightpath · Traffic disruption

1 Introduction

In wavelength-switched networks, demands for lightpath establishments may arrive at any time, and the duration of the connection is unknown most of the times. This fact causes network performance to degrade, regardless of the employed online routing algorithm, as time passes by.

Therefore, with the objective of maximizing revenues, the network operator may need to trigger a *Network Re-optimization* process seeking a more efficient allocation of network resources for the existing traffic. In WDM networks, network reoptimization (also named *network consolidation*) consists of two tasks:

- **REROUTING**: given a *subset of traffic demands* and a map of *available resources* in the network, find a set of (new) lightpaths whose allocation improves the overall performance metric of the network. The set of 'available resources' consists of *free resources* (those that are not employed at the moment by any lightpath) and, depending on the reconfiguration strategy, *re-allocable resources* (those that are currently used by the given subset of demands).
- **RECONFIGURATION**: Here, we are given two lightpath configurations (or virtual topologies): a) the working configuration of lightpaths and b) the new configuration of lightpaths that improves the overall resource utilization of the network; and we need to find the best sequence of steps in order to migrate the traffic from one configuration to the other. The consideration of re-allocable resources at the rerouting stage *may* cause that some resources seized by working lightpaths must be used for some of the new

F. Solano (✉) · M. Pióro
Institute of Telecommunications, Warsaw University
of Technology, Warsaw, Poland
e-mail: fs@tele.pw.edu.pl

M. Pióro
e-mail: mpp@tele.pw.edu.pl

M. Pióro
Department of Electro and Information Technologies,
Lund University, Lund, Sweden

lightpaths, generating a complex sequence of steps that may originate traffic disruptions, violations of SLAs and penalization fees for the network operator.

We explain these two tasks and their trade-off in the following example.

1.1 Illustrative example

Let us consider a topology with seven nodes and four connections: a , b , c and d . Let us assume that there is one wavelength channel per fiber. Due to the dynamism of connection and disconnection requests, let us assume that the online routing algorithm has allocated resources for these four demands as depicted in Fig. 1(a); this is our working configuration. Let us assume in our example that our network performance metric is wavelength occupancy, for ease of explanation. In this sense, we say that the routing cost of this configuration is 12 (as 12 wavelengths are occupied).

In order to obtain a better network performance, let us assume that we can count on all existing wavelengths as re-allocable resources. In this way, we find the optimal configuration shown in Fig. 1(b) with routing cost 6.

However, reconfiguring to the optimal configuration is tricky, if we assume that the wavelengths depicted for both configurations are the same. For instance, the reconfiguration of connections a and b are in a deadlock since each of their new lightpaths need the resources that are allocated to the working lightpath of the other connection: wavelengths $2 \rightarrow 3$ and $5 \rightarrow 4$. The same situation is faced by connections c and d (observe wavelengths $5 \rightarrow 6$ and $2 \rightarrow 1$). In addition, the new lightpath for c needs the wavelength in $2 \rightarrow 5$ currently allocated to a . The same situation is experienced by lightpaths d and b on wavelength $5 \rightarrow 2$.

The solution consists on the temporarily disruption of two connections—let us assume connections a and c —out of the four connections, in order to proceed with the reconfiguration, which is shown in Fig. 2.

If we assume that the time needed to signal the establishment or the disconnection of a lightpath is 2 units of time¹ per hop, 6 units of time are needed for disconnecting the working lightpath of connection a (3 hops in Fig. 1(a)) and 2 more units to establish its new lightpath (1 hop in Fig. 1(b)). In summary, the reconfiguration of connections a and c would need 8 and 10 units of time, respectively. If we assume that the SLA allows a maximum disruption time of 9 units of time within a fixed time-window, a penalization fee proportional to 1 unit of time must be reimbursed to the client of connection c .

¹In practice one unit of time corresponds to the time needed by an optical switch to adjust its components, which is approximately 50 ms if it is a MEMS-based implementation.

(a) Working configuration

(b) Optimal rerouting with routing cost 6

(c) Constrained rerouting with routing cost 8

Fig. 1 Example of re-optimization solutions

In Fig. 1(c) we show routing configuration of lower (sub-optimal) cost of 8. Even though not depicted, in this sequence no SLAs are violated since all the disruption times are allowed by the SLA, i.e., below 9 units of time.

Summarizing, we have shown the trade-off between routing cost and penalization fees during a re-optimization process: two new configurations in which the one with the lowest routing cost incurs in a (higher) penalization fee than the other configuration with a higher routing cost.

Fig. 2 Representation of the reconfiguration sequence of the routing solution of Fig. 1(b). The order in time of the sequence is from left to right. In this case, working lightpath for *a* is torn down, then the new lightpath for *b* is established, then the working lightpath for *b* is disconnected, and so on. This sequence incurs in 1 SLA violation

In this paper we present a methodology that seeks the optimal rerouting without exceeding a threshold in the disruption time for some demands. The importance of this work is that network operators will be able to reoptimize the network—obtaining the best usage of their resources as possible—bounding the violation of the Service Level Agreements (SLA) of the clients. *It is important to highlight that previously several Routing and Wavelength Assignment (RWA) methods have been proposed to find an optimal configuration, but none of them restricts the aforementioned counter-effect on traffic disruption, as we do.*

This paper is organized as follows. In Sect. 3 we provide an analysis of the problem of finding the sequence of steps in a reconfiguration that minimizes the penalization fee. In Sect. 4 we explain our proposed methodology for reoptimizing the network without penalization fees. In Sect. 5 we present some numerical results depicting metrics trade-offs. We provide conclusions and further work in Sect. 6.

2 Related work

The most common way of dealing with this problem in the literature is re-optimizing one lightpath at a time, as it is the case in e.g., [1–3]. In [4] the authors consider that backup lightpaths can be taken as free resources during reoptimization of working lightpaths, and vice versa. In all these cases, the network performance achieved is not the best. The reader is referred to [5] for a survey of the literature of years prior 2004 concerning reoptimization. The remaining of this section focuses on the state of the art of the reconfiguration problem solely, i.e., excluding the possibility of computing the new routes in the problem.

In earlier years (prior to 2004), Jose and Somani in [6] studied a different objective function: minimizing the total disruption during reconfiguration. This problem directly maps to the Minimum Feedback Vertex Set (MFVS) [7], whose complexity is well-known. The most efficient algorithm solving this problem was proposed by in [8].

Most of the research in the area of WDM reconfiguration problems have been done by Coudert and by the authors intensively since 2004. The objective function studied was the

minimization of the maximum disruption experienced during the reconfiguration of the network. While Coudert has focused on the analysis and modeling of some reconfiguration problems (see, e.g., [9–11]), the authors have opted for proposing heuristics and evaluating its performance on real problem instances (see, e.g., [12] and [13]).

With the aim of modeling the problem closer to the real needs of network operators, the authors have opted for analyzing a different objective function, which has not been studied previously.

3 Analysis of lightpath reconfiguration and penalization fees

In this section, we explain how to calculate the penalization fees (and disruption times) of a reconfiguration process. For this, we first introduce the necessary notion of a dependency graph and a feedback vertex set.

3.1 Background: dependency graph

We explain the concept of a *dependency graph* [6], which is a building-stone for circuit reconfiguration problems. A dependency graph is a directed graph that is built as follows:

- map each connection of the reconfiguration problem to a vertex of the dependency graph
- if the new lightpath for a connection, say *a*, needs some resources that are currently allocated to the working lightpath of another connection, say *b*, then create an arc from the vertex representing the first connection, *a*, to the vertex representing the second connection, *b*, in the dependency graph.²

In Fig. 3 we show the dependency graph considering the working configuration (see Fig. 1(a)) and the optimal rerouting configuration (see Fig. 1(b)) depicted in the previous example.

²Due to the way in which optical switches and signaling protocols operate, a new lightpath of a connection can reuse the resources that are allocated to its working lightpath without disruption its traffic.

Fig. 3 Dependency graph and feedback vertex set for the working and optimal rerouting configuration

All connections can be reconfigured (without disruptions) if and only if the dependency graph is acyclic [6]. In fact, the reconfiguration steps can be computed efficiently with a simple polynomial algorithm [6], if the condition holds.

In graph theory, a subset of vertices of a digraph that breaks all the cycles in the graph is called a *feedback vertex set*. Given a cyclic dependency graph, the only way to perform the reconfiguration is by finding a feedback vertex set and disrupting the associate connections [6]. In the example depicted previously, one of the feedback vertex sets with minimal cardinality in Fig. 3 are the two shaded vertices, i.e., a and c .

3.2 Problem modeling

Given a dependency graph $G(D, E)$, let $F \subseteq D$ be a feedback vertex set of G . As pointed out previously, let us assume that all the working lightpaths of demands associated with vertices in the given feedback vertex set F are torn down initially, causing a traffic disruption. For all the other demands, i.e. $D \setminus F$, we first establish the new lightpath—namely *reconnection* operation henceforth—before tearing down the working one—namely a *disconnection* operation. This yields the following two rules of succession relationship in any cost optimal reconfiguration sequence:

- $\forall d \in D \setminus F$, the disconnection operation follows *immediately* after its reconnection operation;
- a reconnection operation is performed *just* after *all* the necessary disconnection operations are performed.

Let us use the variable t_i^- and t_i^+ for representing the instant of time in which the disconnect and reconnect operations, respectively, start for demand $i \in D$. It is clear that: a) $t_i^- < t_i^+$ for those demands i in F , and b) $t_i^- > t_i^+$ for those demand i not in F . We will use later the nomenclature t_i' as a variable to symbolize either a reconnection or disconnection operation on i and, if present in the same expression, t_i'' to symbolize the opposite operation over the same demand.

Let us denote by δ_i^- and δ_i^+ the amount of time required for tearing down the working connection and establishing the new connection, respectively, of demand i .

The aforementioned two rules can be formulated as follows (written down in the same order):

$$\forall i \in D \setminus F, \quad t_i^- = t_i^+ + \delta_i^+ \quad (1)$$

$$\forall i \in D, \quad t_i^+ = \max_{j \in a(i)} \{t_j^- + \delta_j^-\}, \quad (2)$$

with $a(i)$ being the set of vertices with incoming arcs from vertex i in the dependency graph.

The elapsed time in which a connection is disrupted can be described easily as $(t_i^+ - t_i^-)$, $\forall i \in F$, or generally as $\max\{t_i^+ - t_i^-, 0\}$, $\forall i \in D$. We consider that the penalized time is, hence, the amount of disrupted time that exceeds a given threshold, and its penalization fee is proportional to the penalized time. Concretely, we define it as follows:

$$\omega_i \cdot \max\{t_i^+ + \delta_i^+ - t_i^- - \mu_i, 0\}, \quad \forall i \in D \quad (3)$$

where μ_i is the time threshold for which demand i is allowed to be disrupted without being penalized and ω_i represents the unitary cost for its penalization (if the disruption time exceeds the threshold μ_i). As a consequence, the total penalization fee of a reconfiguration sequence is:

$$c(F) = \sum_{i \in F} \omega_i \cdot \max\{t_i^+ + \delta_i^+ - t_i^- - \mu_i, 0\} \quad (4)$$

As the reader could observe, the value of $c(F)$ —hence the optimality of our solution—depends only on the value of t_i^- for $i \in F$, since all other values—i.e., t_i^+ , $\forall i \in D$, and t_i^- , $\forall i \in D \setminus F$ —are set by the previously mentioned two rules.

The optimal penalization fee—or which is the same, the optimal starting times of the disconnection operations of the connections represented by the vertices in F —can be computed with a simple linear problem, according to the equations mentioned previously. At this point, we conjecture that the problem discussed in this section, i.e., computing the values of t_i^- such that minimizes $c(F)$, can be solved in polynomial time by an algorithm. In this section we present an algorithm that finds a good solution in polynomial time.

3.3 Algorithmic solution

Along our explanation, we will use the dependency graph shown in Fig. 4 for better visualization of the concepts and the algorithm. The example consists of five demands: a , b , c , d and e that are dependent on each other for reconfiguration in a double chain structure, creating four simple cycles in the dependency graph.

Concepts In any cost optimal reconfiguration sequence, there is at least one required operation that *immediately* follows in time the next operation. For instance, recalling (2), a reconnection operation is only performed once all required disconnections have been finished. However, it may happen

that some of these disconnections are finished at different times, creating a gap between the reconnection operation and some of its previous disconnection operations. We say that an operation t'_i is *tightly bound* to t''_j when $t'_i = t''_j + \delta''_j$ and $j \in a(i)$. In the example solution of Fig. 5(a), t_b^+ is tightly bound to t_a^- but *not* to t_c^- —note that there are 4 units of gap between $t_c^- + \delta_c^-$ and t_b^+ .

W.l.o.g., we assume that an operation can be tightly bound to at most one (different) operation. Because of this, a set of operations that are tightly bound to each other conform a tree structure where the root of the tree is a disconnection operation. We will use $s(t'_i)$ to denote the set of operations that are tightly bound to t'_i and

$$S(t'_i) = \bigcup_{t''_j \in s(t'_i)} S(t''_j) \cup s(t'_i)$$

to denote the set of operations conforming the tree assuming t'_i as a root. In the example of Fig. 5(a), $S(t_a^-) = \{t_b^+, t_c^-, t_d^+, t_e^+\}$, $S(t_c^-)$ is empty, and $S(t_e^-) = \{t_d^+, t_e^+\}$.

Initial solution We start with an initial feasible solution, where all disconnections of the demands in F occur at time 0. The time for all other operations are computed following the succession rules previously described. This solution is the one depicted in Fig. 5(a).

Fig. 4 Dependency graph example. The values in parenthesis correspond to $(\delta_i^-, \delta_i^+, \mu_i, \omega_i)$

Iteration We compute for each demand $i \in F$ the change in the value of $c(F)$, namely Δ_i , that would occur if we delay the disconnection operation of demand d one unit of time later. The value of Δ_i could change after delaying the disconnection operation one unit of time, for instance, as it depends on the time gaps between related operations. For simplification, we use κ_i to symbolize the maximum length of time for which Δ_i is still accurate. In each iteration we delay the demand that reduces the value of $c(F)$ the most, i.e., the one with the largest negative value of $\Delta_i \cdot \kappa_i$. Once delayed, we recompute the values of Δ_i and κ_i and repeat the procedure while there is a demand with a negative value of Δ_i . To complete our explanation, we proceed to explain how Δ_i and κ_i are computed.

When the disconnection operation of a demand i is delayed, say one unit of time, all tightly bounded operations related to i and its descendants—namely $S(t_i^-)$ —are delayed as well one unit of time. This may affect the disruption times of some other demands in F , if the reconnection operation of those demands is tightly bound through several operations to the disconnection of i . For example, if t_a^- is delayed, it will increase the disruption time of demand c . A similar situation would occur with the disruption time of demand c , if t_e^- is delayed more than 2 units of time, since t_c^+ would become tightly bound to t_d^- after delaying t_e^- one unit of time.

According to our previous observation, the value of Δ_i can be given by:

$$\Delta_i = \begin{cases} 0, & \text{if } t_i^+ \leq t_i^- + \mu_i \text{ or} \\ & t_i^+ \in S(t_i^-) \text{ or } i \notin F \\ \sum_{j, t_j^+ \in S(t_i^-)} \omega_j - \omega_i, & \text{otherwise} \\ t_j^+ > t_j^- + \mu_j \end{cases} \quad (5)$$

(a) Initial solution. The cost of this solution is $(10 - 0 - 3) \cdot 10 + (9 - 0 - 3) \cdot 5 + (8 - 0 - 3) \cdot 1 = 105$.

(b) Optimal solution. The cost of this solution is $(10 - 0 - 3) \cdot 10 + (9 - 3 - 3) \cdot 5 + (9 - 1 - 3) \cdot 1 = 90$.

Fig. 5 Initial and optimal solution for the problem of finding the optimal reconfiguration sequence

Table 1 Value of the main variable algorithms through the iterations when solving the problem depicted in Fig. 4

Iteration	t_c^-	t_e^-	Total disruption	Demand of delayed disconnection (i)	Δ_i	κ_i
Initial	0	0	105	c	$0 - \omega_c = -5$	$t_d^+ - t_c^- - \delta_c^- = 2$
1st	2	0	95	c	$\omega_e - \omega_c = -4$	$t_c^+ - t_d^- - \delta_d^- = 1$
2nd	3	0	91	e	$0 - \omega_e = -1$	$t_d^+ - t_e^- - \delta_e^- = 1$
3rd	3	1	90	–	–	–

where j refers to those demands in F whose reconnection operation is tightly bound through several operation to t_i^- and contribute positively to $c(F)$. In our example in Fig. 5(a), both Δ_a and Δ_e are zero as their reconnection operation is tightly bounded through several operations to their respective disconnection operation; hence, any delay in the disconnection time of, say, a will not affect the disruption time of a . However, the situation is different for demand c . In this case, Δ_c has the value of -5 (equal to $-\omega_c$), since delaying the disconnection of c will not affect the disruption times of any other demand (because no other operation is tightly bound to t_c^-) but only itself at this point. Since Δ_c is the only variable with a negative value, the algorithm will opt for delaying the disconnection time of c some units of time in its first iteration.

Now we have to calculate *how much time* each connection i could be delayed while preserving the same gain ratio Δ_i . The idea is that the disconnection of i can be *pushed* as long as it does not push other operations outside $S(t_i^-)$. As mentioned before, this is represented by the variable κ_i . It can be computed as:

$$\kappa_i = \min_{j, t_j^- \in S(t_i^-) \cup t_i^-} \left(\min_{k \in a(j) \setminus \{i\}} t_k^+ - t_j^- - \delta_j^- \right), \quad \forall i, \Delta_i < 0 \quad (6)$$

In our example, $\kappa_c = \min_{k \in a(c) \setminus \{c\}} t_k^+ - t_c^- - \delta_c^- = \min\{t_b^+, t_d^+\} - t_c^- - \delta_c^-$ in the first iteration, which takes the value of 2.

The cardinality of the set $S(t_i^-)$ increases, in each iteration as the disconnection operation of i is delayed, by adding the adjacent operations to the operations in $S(t_i^-)$ that set the value of κ_i in the previous expression.

It is worth noticing that since the cardinality of the set $S(t_i^-)$ increases through iterations, the value of κ_i tends to reduce, and the value of Δ_i increases (towards 0). As a way to show it, and for completeness, we present in Table 1 the values of the main variables of our algorithm when the problem instance of Fig. 4 is considered. The optimal final solution³ can be observed in Fig. 5(b). We omit the values for connection a as it experiences no changes through the algorithm's iterations.

³Note that delaying one unit of time more t_c^- and t_e^- would provide a solution of equal cost.

4 Reoptimizing considering penalization fees

We have analyzed and described the way of computing a reconfiguration sequence of minimum penalization fee given a dependency graph and a feedback vertex set. In this section, we focus on the embracing problem of how to reoptimize the network—by calculating the new lightpaths—in such a way that the minimum penalization fee of the resulting reconfiguration problem is zero, if possible.

The methodology consists of two procedures that jointly find the best solution to the constrained re-optimization problem. The first procedure is an ILP model for rerouting, named *Constrained Rerouting* (CR) and the second is a branching procedure, named *Deadlock Reconfiguration* (DR), that finds the best feedback vertex set. These two procedures are run one after the other, several times, until the best solution is found. We proceed to explain how these two procedures collaborate with each other.

CR is a path-based formulation for a traditional multi-commodity flow problem (minimum-cost routing of all the demands in the network) with some additional combinatorial constraints that relates it to DR. DR is a tool for determining which demands of the last solution of CR are exceeding the disruption time threshold during reconfiguration. The main idea is simple: DR detects which lightpaths cause penalization fees at iteration qt , and then, we ask CR to find a new lightpath configuration by forbidding this combination of lightpaths in iteration $q + 1$.

The main iteration of our procedure is drafted as follows:

1. Run CR considering that all the resources are re-allocable. We will obtain a new lightpath configuration, namely $L^{(0)}$.
2. Set q to 0.
3. Build the dependency graph G_q by taking into account the input working configuration, namely L , and $L^{(q)}$.
4. If G_q is acyclic, return $L^{(q)}$.
5. Run DR and find which new lightpaths are causing deadlock reconfigurations with unacceptable disruption times. If none found, return $L^{(q)}$.
6. Let $\mathcal{C} \in 2^D$ be a family of sets of demands, where each element $C_j \in \mathcal{C}$, $0 < j \leq |\mathcal{C}|$ represents a set of connection for which there is deadlock incurring in unacceptable disruption time.

Fig. 6 Pseudo-code of the complete routine

7. Run CR constraining the selection of all the new lightpaths in C_v at the same time, for $0 < v \leq |\mathcal{C}|$.
8. Increase q and go to 3.

The routine can be seen in Fig. 6.

Let us name $z_{L^{(q)}}$ the routing cost of configuration solution at iteration q , i.e., $L^{(q)}$. It is not hard to prove that after q iterations:

$$z_{L^{(0)}} \leq z_{L^{(1)}} \leq \dots \leq z_{L^{(q-1)}} \leq z_{L^{(q)}} \leq z_L \quad (7)$$

We proceed to detail the CR ILP and the DR algorithm.

4.1 Constrained rerouting

In CR we are given a set of candidate paths for every demand and we have to choose a subset of them that minimizes a performance metric cost and satisfies some reconfiguration conditions. In this article we consider: a) that the performance metric is the total number of wavelength occupancy in the network and b) that there are no wavelength conversion capabilities. However, the methodology can be easily extended towards other objective functions and the support of wavelength conversion.

The set $\mathcal{C}^{(q)}$ consists of a set of path combinations that cannot occur in any routing solution of CR at iteration q or later; we name them the *set of forbidden combinations*. This set is updated at every iteration with information coming from DR, as previously mentioned. We explain how elements are added to $\mathcal{C}^{(q)}$ in the next subsection.

We assume that each demand $i \in D$ is of one lightpath connection using the capacity of one wavelength channel. The ILP model presented here works with aggregated demands, viz., we use a parameter to quantify how many wavelength channels are demanded between a pair of nodes: $h_{(n,m)}$ counts the number of demands that have n as source and m as destination.

We employ the following constants and variables in our model. Constant $\rho_{(n,m)}^{p,f}$ is set to 1 when fiber f is used by the p -th candidate path from n to m . The variable $x_{(n,m)}^{w,p}$ is set to 1 when a lightpath using wavelength w and following the p -th candidate path from n to m is taken as part of the output configuration $L^{(q)}$, 0 otherwise.

$$z_{L^{(q)}} = \min \sum_{f,w,p,n,m} \rho_{(n,m)}^{p,f} \cdot x_{(n,m)}^{w,p} \quad (8a)$$

$$\sum_{w,p} x_{(n,m)}^{w,p} \geq h_{(n,m)}, \quad \forall (n,m) \quad (8b)$$

$$\sum_{p,n,m | \rho_{(n,m)}^{p,f}=1} x_{(n,m)}^{w,p} \leq 1, \quad \forall f, w \quad (8c)$$

$$\sum_{w,p,n,m | \{w,p,n,m\} \in C_v} x_{(n,m)}^{w,p} \leq |C_v| - 1, \quad \forall C_v \in \mathcal{C}^{(q)} \quad (8d)$$

$$x_{(n,m)}^{w,p} \in \{0, 1\}, \quad \forall w, p, n, m \quad (8e)$$

Equation (8a) is our objective function. Constraint (8b) assures that there are $h_{(n,m)}$ lightpaths between n and m . Constraint (8c) constrains the usage of the same wavelength-fiber by two lightpaths. Constraint (8d) forbids the usage of some combination of lightpaths, as given by the previous iteration of DR.

4.2 Reconfiguration fees detection

We are given two configurations and we build its dependency graph. The recursive procedure in Fig. 7 finds the feedback vertex set with minimum penalization fee, i.e., considering (4) as a cost function.

The procedure works as follows. It receives a feedback vertex set F . It attempts to create another feedback vertex

Input: Graph $G(D, E)$, Vertex subset $F \subseteq D$
 Obtain $c(F)$;
 Let $Z \leftarrow \arg \min_{X=\{F, Z\}} c(X)$;
for $i \in F$ *such that* $\Delta_i > 0$ **do**
 Let $F' \leftarrow F \setminus \{i\}$;
 if F' *is a feedback vertex set of* G *and is not visited*
 then
 Obtain $c(F')$;
 if $c(F') \leq c(F)$ **then**
 branch-on(G, F');
 mark feedback vertex set F' as visited;

Fig. 7 Branching for finding the minimum penalization fee

set, named F' , formed by all the elements of F except by one arbitrarily chosen vertex. The conditions that: a) the set F' is a feedback vertex set, b) the cost of F' is not worse than the cost of F , and c) F' has not been checked before, are necessary conditions for the recursive invocation of the procedure using F' .

The procedure is called initially considering all the vertices in the dependency graph as our initial feedback vertex set, i.e., $F \leftarrow D$. Through iterations, the feedback vertex set with the best cost is stored in the global variable Z . The iterations of this algorithm are shown in Fig. 8. We prove in the Appendix that this algorithm finds the optimal solution to the problem.

Similarly to the MFVS problem, this reconfiguration problem can be divided into several subproblems, one for each *Strongly Connected Component* (SCC) of the dependency graph. We can solve each subproblem separately and later merge subsolutions. By considering subproblems (SCC) we: 1) decrease the time needed by DR to find the minimum penalization fee and 2) provide tighter bounds for the set of forbidden combinations of CR. The latter occurs because forbidden combinations resulting from a SCC apply as well to the initial (larger) dependency graph, hence it can be added in the next invocation of CR.

5 Numerical results

The purpose of this section is to show the efficacy of the proposed method in this paper. The numerical results presented in this section are focused on characterizing the hardness of reconfiguring the network considering some reconfiguration problem instances, and not about the achieved performance network metric, as the latter policy may vary from operator to operator.

Table 2 Types of demands used in the numerical results

Class	Percentage of demands	Disruption threshold	Cost of disruption $\times$ time unit
Gold	25%	0	10
Premium	50%	$2 N $	5
Economic	25%	1000	1

5.1 Scenario description

In order to provide comparable results, we generate problem instances in which between 15% and 25% of the resources can be saved after the re-optimization; this depicts similar re-optimization results as [14]. For this, we blocked nearly 50% of the wavelengths—in order to fragment resources—and executed CSPF algorithm routing one demand between any two random nodes in the network at a time. The demand model is increasing, therefore no demand leaves the network. Every time 20 new lightpaths are established in the network, the proposed methodology in this paper is executed considering that all the lightpaths can be rerouted and that all the network resources are available (including the previously blocked wavelengths).

Three types of SLA concerning disruption thresholds and costs of disruption are considered. These are shown in Table 2. The intuition behind the parameters of *gold* SLA is that no disruption is tolerated, and if it occurs a high fee must be reimbursed. Gold demands should be reoptimized without penalization fee if only no disruption is made. For *premium*, disruption of only that connection is permitted; if the disruption is longer, a lower fee must be reimbursed. *Economic* SLA reflect more the idea of backup lightpaths which can be disrupted with low chances of affecting the service.

The network topology used is the German topology with 50 nodes found in SNDLIB library project.⁴ Each fiber can be demultiplexed in 16 wavelength channels. As mentioned before, no wavelength conversion is allowed. For CR, we generated several paths using the k -shortest paths algorithm. We observed that 10 paths were enough to achieve the optimal solution to most of the problem instances.

Due to the exponential nature of the exact procedure proposed in Sect. 4.2, this reconfiguration procedure is executed for one minute at most. Therefore, the results shown here should be interpreted as upper bounds. We use a Pentium IV computer with 2 GB of RAM using Cplex 11 as solver and Java.

We run each experiment 400 times. In total we obtained approximately 5000 re-optimization problem instances considering different number of lightpaths between 20 and 300.

⁴<http://sndlib.zib.de/>.

Fig. 8 Iterations of the branching procedure. Darker vertices represent the feedback vertex set. Solutions in a circle have a cost of 0, other solutions have a cost of 1. We do not show solutions where a FVS is not obtained, since they are not explored

How much routing cost must be sacrificed in order to obtain a zero penalization fee? We observe in nearly our 5000 re-optimization problem instances that, after several iterations, an equal-cost routing lightpath configuration of lowest routing cost without penalization fee exists, that is:

$$z_{L^{(0)}} = z_{L^{(q)}}$$

This means that it was always possible to find the optimal routing solution without any degradation of its cost because of avoiding penalization fees.

How many iterations are needed to obtain a zero penalization fee? Throughout all our experiments, although extremely unlikely, we observe that a maximum of 395 iterations are needed in order to find a zero penalization fee re-optimization for 140, 200 and 240 lightpaths. We proceed to analyze the distribution of the number of re-optimization problem instances according to the number of iterations required to achieve a zero penalization fee reconfiguration.

We classify each experiment according to the maximum number of iterations that were needed in order to find a zero penalization fee. We have categories for experiments incurring in a number of iterations between $10x + 1$ and $11x$, where x is the number of the category starting at 0. In figure Fig. 9, we show the distribution of experiments according to these categories. When 200 lightpaths are considered, We observe that nearly 91% of the experiments need less than 30 iterations to find a zero penalization fee re-optimization.

Fig. 9 Distribution of the number of iterations needed to obtain a zero penalization fee using our method with a load of 140, 200 and 240 lightpaths

We observe also that varying the number of lightpaths alters the number of iterations required to obtain a zero penalization fee reconfiguration. For instance, 30% of the experiment using 240 lightpaths required 10 or less iterations, while this percentage for 140 lightpaths is 92%.

How many demands should be disrupted? In Fig. 10 we show the average number of demands that need to be temporarily disrupted by MFVS and our proposal. It shows that our proposal needs to disrupt between 2 and 4 times more demands. Even though our method disrupt more demands, the reconfiguration is performed without penalization fees.

Fig. 10 Comparison of the average number of demands temporarily disrupted incurred by MFVS and our proposal

Fig. 11 Comparison of the average penalization fee incurred by MFVS and our proposal

What would the penalization fee of not using this method? In Fig. 11 we show the average penalization fee when using MFVS [6] and the average penalization fee through the iterations of the method presented in this paper (labeled as PF). We can clearly observed that, on average, using MFVS may yield a penalization fee twice as big as the average found through the iterations of our methods.

6 Conclusions

In this paper we proposed a new scheme for lightpath re-optimization that bounds the possible unacceptable disruptions on the traffic caused during the resulting reconfiguration process. Our scheme works iteratively. In each iteration it finds a new configuration trying to avoid previously detected reconfiguration deadlocks that may cause penalization fees.

In all our experiments, numerical results showed that it was always possible to achieve new routing configurations with optimal performance such that the reconfiguration process do not incur on penalization fees. Considering a normal network load (70% of resource usage), nearly 90% of the times it was possible to compute this new routing configuration using a 2 GHz Intel PC in less than 30 minutes, which we consider as an accurate window time for provisioning in ASON. In addition, we observe that the penalization fees incurred by previously proposed methods are extremely high: almost twice the average of our solution.

Acknowledgements We acknowledge the Polish Ministry of Science and Higher Education because of its grants: MNiSW, grant no. N517 397334: *Optimization Models for NGI Core Networks*, and MNiSW, grant 58/N-EUREKA/2007/0: *Management Platform for Next Generation Optical Networks* within the EU CELTIC project MANGO CP4-17.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

Appendix: Proof

The proposed algorithm takes in each iteration a Feedback Vertex Set (FVS) F and explores derived solutions by removing one vertex, namely i , from F . If $F \setminus \{i\}$ is a FVS and its cost, i.e., $c(F \setminus \{i\})$, is greater than the cost of its parent solution, i.e., $c(F)$, then the algorithm does not perform consecutive explorations based on $F \setminus \{i\}$.

The key point to proof is that no other proper subset of F can have a lower cost than $c(F)$ in these circumstances. In other words, we need to proof that:

$$\text{If } \forall i \in F, c(F) < c(F \setminus \{i\}) \text{ then } c(F) \leq c(F'), \forall F' \subset F$$

In order to simplify our explanation, we will initially assume that any considered set F and any subset of F mentioned in this appendix is a FVS, unless otherwise explicitly stated. W.l.o.g., we consider the values of ω_i and μ_i set to 1 and 0, respectively.

Since reconnection operation of a connection $i \in F$ can only be performed once *all* the necessary operations are finished, we need to estimate the *longest* elapsed time since any of the disconnection operation of the connections in F necessary for reconnecting i . A lower bound for the elapsed time is obtained by finding the longest path between vertices of the feedback vertex set in the dependency graph considering $\delta_i^+ + \delta_i^-$ as the weight of the vertex i .

Definition Let us define the l -path of $k \in F$ as the longest simple⁵ path from vertex k to any other vertex in F in the

⁵With no cycles.

given graph. We represent the l -path from a vertex k in a solution F as $Q_F(k)$. Let $d_F(k) \in F$ be the destination (last vertex) in the l -path from k in the solution F , i.e., $Q_F(k)$.

Theorem 1 If $c(F \setminus \{i\}) > c(F)$, for some $i \in F$, then $Q_{F \setminus \{i\}}(k) = Q_F(k) \oplus Q_F(i)$, $\forall k \in Q_F(i) \cap F$.

Since $c(F \setminus \{i\}) > c(F)$, the length of the l -path for some vertex k in $F \setminus \{i\}$ should be greater than the length of its l -path in F , meaning, $|Q_{F \setminus \{i\}}(k)| > |Q_F(k)|$. Since the l -path from k can only get longer by not including i in the solution, vertex i must be traversed in the longest path from k , otherwise we would stay with the same l -path in the solution F . Since $d_F(k)$ is the farthest vertex from i , hence it is from k in the solution $F \setminus \{i\}$, i.e., $d_F(k) = d_{F \setminus \{i\}}(k)$. Therefore, as i is the only connecting point from k to $d_F(k)$ (excluding all other vertices in F), $Q_{F \setminus \{i\}}(k) = Q_F(k) \oplus Q_F(i)$.

Now we proceed with our final proof. Let F be a FVS for which the removal of any vertex induces a solution of higher cost. In order to prove that any subset of F also induces a solution of higher cost, we proceed by induction. We prove that if $c(F \setminus \{i\}) > c(F)$ and $c(F \setminus \{j\}) > c(F)$, for some $i, j \in F$, then $c(F \setminus \{i, j\}) > c(F)$.

If we focus on the solution F , there are three possible ways in which i and j could be related in this solution:

CASE 1. $d_F(i) \neq j$ and $d_F(j) \neq i$. By Theorem 1, the l -path in solution $F \setminus \{i\}$ for the vertices in $Q_F(i)$ are disjoint from the l -path in the solution $F \setminus \{j\}$ for the vertices in $Q_F(j)$. As a consequence, the cost of the solution $F \setminus \{i, j\}$ can be computed as $c(F \setminus \{i, j\}) = (c(F \setminus \{i\}) - c(F)) + (c(F \setminus \{j\}) - c(F)) + c(F) > c(F)$.

CASE 2. $d_F(i) = j$ and $d_F(j) \neq i$. Consider the solution $F \setminus \{i\}$. Following the same reasoning of the proof of Theorem 1, we can show that the l -path from the vertices in $Q_{F \setminus \{i\}}(j)$ end in $d_{F \setminus \{i\}}(j)$ when vertex j is removed from this solution. As a consequence, the cost of the solution $F \setminus \{i, j\}$ can be computed as $c(F \setminus \{i, j\}) = c(F \setminus \{i\}) + |Q_{F \setminus \{i\}}(j)| \cdot \sum_{k \in n(Q_F(i))} \omega_k > c(F \setminus \{i\}) > c(F)$.

CASE 3. $d_F(i) = j$ and $d_F(j) = i$. Vertices i and j cannot be removed at the same time from F as there would be cycles.

References

1. Chu, X., Bu, T., & Li, X. y. (2007). A study of lightpath rerouting schemes in wavelength-routed WDM networks. In *Proceedings of IEEE international conference on communications (ICC)* (pp. 2400–2405), Glasgow, Scotland, June.
2. Sreenath, N., Panesar, G., & Siva Ram Murthy, C. (2001). A two-phase approach for virtual topology reconfiguration of wavelength-routed wdm optical networks. In *Proceedings of the 9th international conf. on networks (ICON)* (pp. 371–376), Bangkok, Thailand, October.
3. Takagi, H., Zhang, Y., & Jia, X. (2000). Virtual topology reconfiguration for wide-area wdm networks. In *IEEE international conference on communications, circuits and systems and West Sino expositions* (pp. 835–839), Chengdu, China, June.
4. Xin, Y., Shayman, M., La, R., & Marcus, S. (2006). Reconfiguration of survivable mpls/wdm networks. In *Proceedings of IEEE global communications (Globecom)* (pp. 1–5), San Francisco, USA, November.
5. Golab, W., & Boutaba, R. (2004). Policy-driven automated reconfiguration for performance management in wdm optical networks. *IEEE Communications Magazine*, 42(1), 44–51.
6. Jose, N., & Somani, A. (2003). Connection rerouting/network reconfiguration. In *IEEE design of reliable communication networks (DRCN)* (pp. 23–30), Banff, Canada, October.
7. Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness. Series of Books in the Mathematical Sciences*. New York: Freeman.
8. Lin, H.-M., & Jou, J.-Y. (1999). Computing minimum feedback vertex sets by contraction operations and its applications on cad. In *IEEE int. conf. on computer design (ICCD)* (pp. 364–369), Austin, USA, October.
9. Coudert, D., Pérennes, S., Pham, Q.-C., & Sereni, J.-S. (2005). Rerouting requests in WDM networks. In *AlgoTel'05* (pp. 17–20), Presqu'île de Giens, France, May.
10. Coudert, D., & Mazauric, D. *Network reconfiguration using cops-and-robber games*. (Research Report RR-6694). INRIA, 2008.
11. Couder, D., Huc, F., Mazauric, D., Nisse, N., & Sereni, J.-S. (2009). Reconfiguration of the routing in wdm networks with two classes of services. In *Proc. of the optical network design and modeling conference* (pp. 146–151), Braunschweig, Germany, February.
12. Solano, F. (2009). Analyzing two different objectives of the WDM network reconfiguration problem. In *IEEE global communications (Globecom)*, Honolulu, USA, December.
13. Solano, F., & Pioro, M. (2010). Lightpath reconfiguration in WDM networks. *IEEE Journal of Optical Communications and Networking*, 2(12), 1010–1021.
14. Ahmed, J., Solano, F., Monti, P., & Wosinska, L. (2011). Traffic re-optimization strategies for dynamically provisioned wdm networks. In *Proc. of the IEEE international conference on optical networking design and modeling*, Bologna, Italy, February.

Fernando Solano received the PhD degree with emphasis in Information Technologies from the University of Girona, Spain, in 2007. He received the BSEE degree with honors from Universidad del Norte, Barranquilla, Colombia, in March 2003.

Currently, he is an Assistant Professor at the Institute of Telecommunications of the Warsaw University of Technology. His research interests include Optical Transport Networks design and analysis, Sensor Networks, Network Optimization and

Algorithms. He has participated in the Mango Celtic Project and he is the coordinator of the Goldfish project funded by FP7.

Michał Pióro is a full professor and the head of the Computer Networks and Switching Division in the Institute of Telecommunications at the Warsaw University of Technology, Poland, and a full professor at Department of Electrical and Information Technology at Lund University, Sweden. He received a PhD degree in telecommunications in 1979 and a DSc degree, also in telecommunications, in 1990, both from the Warsaw University of Technology. In 2002 he received a Polish State Professorship. His research inter-

ests concentrate on modeling, design and performance evaluation of telecommunication networks. He is an author of four books and more than 150 technical papers presented in the telecommunication and OR journals and conference proceedings. He has lead many national and international research projects for telecom industry and EC in the field.