Efficient Make-Before-Break Layer 2 Reoptimization

Huy Duong, Brigitte Jaumard, David Coudert, Romualdas Armolavicius

Efficient Make-Before-Break Layer 2 Reoptimization

Huy Duong[1,2], Brigitte Jaumard[2], David Coudert[3], and Ron Armolavicius[4]

[1]Centre de Recherche Informatique de Montréal (CRIM), Montréal, Qc, Canada
[2]Department of Computer Science and Software Engineering, Concordia University, Montreal (QC) Canada
[3]Université Côte d'Azur, Inria, CNRS, I3S, France
[4]CIENA

June 30, 2021

Abstract

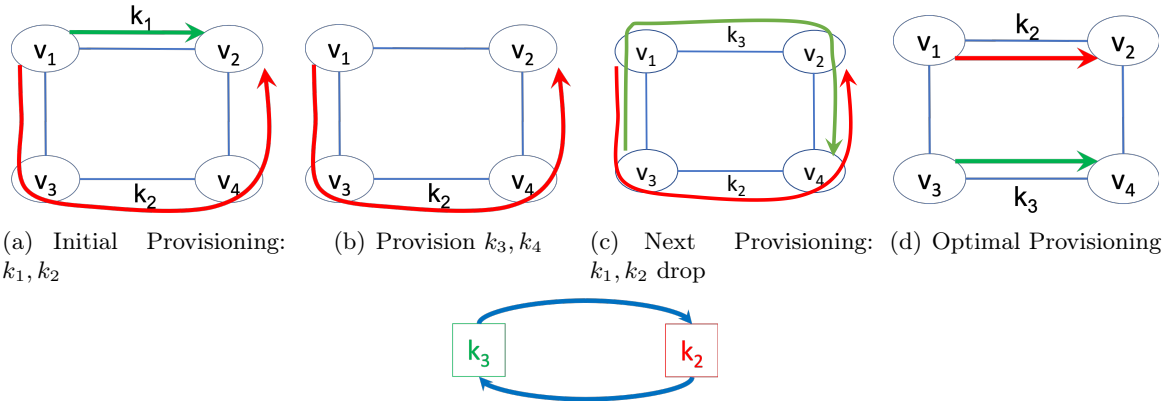
Optical multilayer optimization periodically reorganizes layer 0-1-2 network elements to handle both existing and dynamic traffic requirements in the most efficient manner. This delays the need for adding new resources in order to cope with the evolution of the traffic, thus saving CAPEX.
The focus of this paper is on Layer 2, i.e., on capacity reoptimization at the optical transport network (OTN) layer when routes (e.g., LSPs in MPLS networks) are making unnecessarily long detours to evade congestion. Reconfiguration into optimized routes can be achieved by re-defining the routes, one at a time, so that they use the vacant resources generated by the disappearance of services using part of a path that transits the congested section.
To maintain the Quality of Service, it is desirable to operate under a Make-Before-Break (MBB) paradigm, with the minimum number of reroutings. The challenge is to determine the best rerouting order while minimizing the bandwidth requirement.
We propose an exact and scalable optimization model for computing a minimum bandwidth rerouting scheme subject to MBB in the OTN layer of an optical network. Numerical results show that we can successfully apply it on networks with up to 30 nodes, a very significant improvement with respect to the state of the art. We also provide some reoptimization analysis in terms of the bandwidth requirement vs. the number of reroutings.

Keywords: Network reconfiguration, rerouting, reoptimization, make-before-break.

1 Introduction

Network reconfiguration is required in order to adapt to traffic changes, network failures, or new deployment of network resources. It occurs at the optical layer in order to make sure that the upper layer traffic, e.g., IP layer traffic, can be efficiently carried. In such a case, we deal with lightpath reconfigurations and the primary objective is to reduce disruptions to user traffic carried by existing lightpaths, measured by the number of disrupted lightpaths or the duration of lightpath disruptions [1]. Network reconfiguration may also appear in the logical layer, in order to attain a better resource utilization [2]. In heavily loaded networks, dynamic connection request addition and drop actions may result in a set of connection requests where some paths are not the shortest possible ones, leading to poor resource utilization compared to an optimal or at least an optimized state. Thus, global connections rerouting is proposed at certain time intervals (e.g., daily, weekly) to improve the network performance.

Researchers have investigated this connections rerouting along two directions. The first one consists in computing an optimized provisioning of the connection requests with respect to resource utilization, and then finding a sequence of connection rerouting operations in order to migrate from the current network provisioning to the optimized one with the minimum number of disruptions [3, 4, 5, 6, 7, 8]. These studies usually have the constraint that a connection request can be rerouted at most once (i.e., from legacy to optimized route). The existence of a strategy using only make-before-break (MBB) is not always possible due to the presence of dependency cycles. Consider the example in Figure 1 in which the state of Figure 1(c) results from add and drop requests of states represented in Figures 1(a) and 1(b). In order to reach the optimal provisioning of Figure 1(d), connection request k_2 needs to be rerouted before k_3 because a link of the new route of k_3 belongs to the current route of k_2, and for similar reasons, k_3 needs to be rerouted before k_2, as illustrated in Figure 1. In order to find rerouting strategies, authors have proposed to use the break-before-make (BBM) paradigm that allows for the temporary interruption of connection requests, and so for breaking dependency cycles [3, 4, 6, 7].



(a) Initial Provisioning: k_1, k_2 (b) Provision k_3, k_4 (c) Next Provisioning: k_1, k_2 drop (d) Optimal Provisioning

(e) Dependency Graph

Figure 1: Optimal Provisioning is not MBB Reachable if all links and connections are with unit capacities and requirements, respectively

The idea of the second direction is to compute the best provisioning that is reachable from the legacy provisioning by a sequence of connection reroutings with no disruption, i.e., under the so-called MBB paradigm. While many studies have investigated the first direction, this second direction has received very little attention [9]. In this paper, we propose a scalable optimization model, called REOPT_SIM, for this NP-Complete optimization problem. Numerical results show that we improve very significantly against the state-of-the-art [9], enabling to solve instances on networks with up to 30 nodes.

The paper is organized as follows. We briefly review in Section 2 the papers related to reoptimization in the OTN layer, as well as the model of Klopfenstein [9], which is the only previously proposed optimization model for rerouting subject to MBB. We next describe in Section 3 our proposed decomposition model, called REOPT_SIM, which requires in practice a much smaller number of variables and constraints than the model of Klopfenstein [9]. In Section 4, we explain how to solve efficiently the proposed REOPT_SIM model with the REOPT_MBB algorithm, which contains a polynomial time algorithm for the generation of the rerouting configurations. In Section 6, we show how to use parallel reroutings for reducing the number of rerouting events or, in other words, the duration of each reoptimization event as output by REOPT_SIM (see Section 2.2 for a concise definition of rerouting/reoptimization events). Numerical results are presented in Section 7. Conclusions are drawn in the last section.

2 State of the Art

Note that we focus on Layer 2, while being aware that a lot of work has been recently made on Layer 0 in the context of flexible optical networks. But capacity reoptimization differs from spectrum defragmentation as there is no need to take care of continuity or contiguity constraints, and therefore we omit references related to Layer 0.

2.1 Literature Review

Several studies have been devoted to network reconfiguration with the minimum number of disruptions, following the strategy of migrating from a legacy ineffective provisioning to a given pre-computed optimized/optimal one. As a result, it usually prevents the existence of a strategy using only MBB due to the presence of dependency cycles as explained in the introduction. In order to find a rerouting strategy, authors have then proposed to use the Break-Before-Make (BBM) paradigm sparingly to allow temporary interruption of connection requests, and so to break dependency cycles. For instance, Jose and Somani [5] propose heuristics for minimizing the total number of BBMs used in the rerouting strategy, and Coudert *et al.* [3, 10] and Solano and Pióro [7] provide scalable exact algorithms to minimize the concurrent number of BBMs. Tradeoffs between these two conflicting objectives are investigated by Cohen *et al.* [4] and Solano [6].

To further reduce the total or concurrent number of BBMs, Kadohata *et al.* [8] propose to use spare wavelengths to reroute a connection request to a temporary route rather than using a BBM. For example, assume that the current connection k needs to be rerouted from path p to path p', but such a rerouting cannot be MBB due to resource dependence. Then one unavoidable BBM reroute is performed. However, using an intermediate reroute, it may be possible to reroute k under the MBB paradigm. For instance, assume that there exists a path p'' such that the reroutings from p to p'' and from p'' to p' satisfy the MBB condition. In other words, one BBM can be avoided at the expense of performing two MBBs.

The network reconfiguration problem has also been investigated in logical layers, and in particular for MPLS [2, 11, 12] and SDN [13] networks, with the same constraints and objectives as above. Surprisingly, it has been found that deciding if the problem can be solved using MBB only can be done in polynomial time for Layer 0 [14], but this decision problem is NP-Complete for logical layers, i.e., for Layer 2 [12].

On the other hand, while many studies have investigated rerouting strategies both at the optical and the logical layers, very few studies have looked at rerouting subject to the MBB paradigm (i.e., joint computation of optimal provisioning and rerouting strategy subject to MBB, or, in other words, we perform a sequence of MBB reroutings to achieve an optimal provisioning in terms of minimum bandwidth requirement). Klopfenstein's study [9] is the only one proposing an optimization model, but unfortunately it is not scalable. Still, we recall it in Section 2.3 as an introduction to our decomposition model in Section 3.

2.2 Notations

We consider a network represented by a directed multi-graph $G = (V, L)$, where V is the set of nodes (indexed by v) and L is the set of links (indexed by ℓ). Different links may exist between two nodes in order to model different logical links, with e.g., different types of traffic. We denote $\omega^-(v)$ (resp. $\omega^+(v)$) the set of incident links incoming to (resp. outgoing from) node $v \in V$. Let C_ℓ denote the transport capacity of link ℓ.

Let K be the set of connection requests (indexed by k). Connection request $k \in K$ is characterized by its source s_k, its destination d_k, and its bandwidth requirement b_k.

In what follows, we call *rerouting operation* the action of rerouting a connection request $k \in K$, and *rerouting event* the action of either performing a single rerouting operation, or a set of parallel rerouting operations (see Section 6 for the details on the conditions under which we conduct parallel rerouting). A *reoptimization event* is an ordered sequence of rerouting events, and so of rerouting operations. Let T, indexed by t, be the set of rerouting events of a reoptimization event. Hence, t designates one rerouting event. Observe that under parallel rerouting, t designates the set of rerouting operations performed in parallel (again see Section 6 for the details).

2.3 Klopfenstein's Model (2008)

The model of Klopfenstein [9] consists in finding the best possible rerouting strategy, while guaranteeing it can be reached within a Make-Before-Break (MBB) policy.

Before developing further, we define the set of variables.

- $x^t_{k\ell} = 1$ if connection request $k \in K$ uses link $\ell \in L$ in its routing at step $t \in T$, 0 otherwise.
- $\pi^t_k = 1$ if connection request k is rerouted at step t, 0 otherwise.

Indeed, Klopfenstein [9] proposed a very general network resource utilization function subject to a parameter α and that can be written as follows:

$$\mathrm{OBJ}_\alpha = \frac{1}{1-\alpha} \sum_{\ell \in L} \left(C_\ell - \underbrace{\sum_{k \in K} b_k x^{|T|}_{k\ell}}_{\text{link load}} \right)^{1-\alpha}, \tag{1}$$

In the sequel, we will adopt OBJ_0. The objective is then to maximize the overall spare capacity:

$$\max \qquad \left(\mathrm{OBJ}_0 = \sum_{\ell \in L} C_\ell - \sum_{k \in K} b_k \left(\sum_{\ell \in L} x^{|T|}_{k\ell} \right) \right). \tag{2}$$

These objectives and variables are decided by the set of below constraints:

$$\sum_{\ell \in \omega^-(v)} x^t_{k\ell} - \sum_{\ell \in \omega^+(v)} x^t_{k\ell} = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}$$
$$k \in K, v \in V, t \in T \tag{3}$$

$$\sum_{k \in K} b_k x^t_{k\ell} \leq C_\ell \qquad \ell \in L, t \in T \tag{4}$$

$$\sum_{k \in K} \pi^t_k \leq 1 \qquad t \in T \tag{5}$$

$$x^t_{k\ell} - x^{t-1}_{k\ell} \leq \pi^t_k \qquad k \in K, \ell \in L, t \in T \tag{6}$$

$$x^t_{k\ell} \in \{0,1\} \qquad k \in K, \ell \in L, t \in T \tag{7}$$

$$\pi^t_k \in \{0,1\} \qquad k \in K, t \in T. \tag{8}$$

Constraints (3) are flow constraints and are used in order to establish a route for each connection request, at each rerouting event. Due to the subsequent constraints, the set of paths at rerouting event t will differ by at most one path from the set of paths at rerouting event $t-1$. Constraints (4) enforce the transport capacity constraints. Constraints (5) impose that at

most one connection request is rerouted per rerouting event. Constraints (6) are used to detect rerouted connection requests. More precisely, if the route of connection request k at rerouting event t uses a link ℓ that was not previously used to route that connection request (i.e., at time $t-1$), we have $x^t_{k\ell} = 1$, $x^{t-1}_{k\ell} = 0$, and so $\pi^t_k = 1$, which indicates that connection request k has been rerouted at rerouting event t. Now, if the route of connection request k is not changed, we have $x^t_{k\ell} = x^{t-1}_{k\ell}$ and $\pi^t_k \in \{0, 1\}$, in which case the value of π^t_k is forced to 0 by Constraints (5) if another request k' is rerouted at rerouting event t. Observe that the case $\pi^t_k = 0$, $x^t_{k\ell} = 0$ and $x^{t-1}_{k\ell} = 1$ cannot happen. Indeed, it would imply that the routing of connection request k at rerouting event $t-1$ is not simple (i.e., is not loop-free), which is prevented by the objective. Hence, when $\pi^t_k = 0$, we have $x^t_{k\ell} = x^{t-1}_{k\ell}$ and so the routing of connection request k is unchanged. The last two sets of constraints define the domain of the variables. Note that if the rerouted path has a link in common with the original one, there is no need to double the capacity reservation corresponding to the considered connection request [9].

The largest data instance on which experiments were conducted in [9] was on a network with 10 nodes and about 40 links with capacity C_ℓ. The author was not able to obtain optimal solutions for more than 5 rerouting operations within the time limit imposed (30 minutes).

3 A Decomposition Model: REOPT_SIM

In this section, we propose a decomposition model, called REOPT_SIM, based on a set of rerouting operations, where each rerouting operation proposes a potential MBB rerouting of a single connection request, i.e., a connection request for which there exists an alternate route with enough spare bandwidth for its routing. The model is parameterized by $|T|$, a bound on the number of rerouting operations. A solution of REOPT_SIM is an ordered sequence of at most $|T|$ rerouting operations leading to the best provisioning reachable from the legacy provisioning. Observe that less than $|T|$ rerouting operations might be sufficient to reach that optimized provisioning. No external connection setup or release requests are granted between these rerouting operations. The objective is to minimize the bandwidth requirements of the best reachable MBB provisioning within at most T reroutings. Observe that there is no guarantee to reach the best provisioning with a monotonous sequence, i.e., such that the overall bandwidth requirement decreases after each single rerouting operation (of a connection request). It may happen that the overall bandwidth requirement increases after a given rerouting operation before decreasing again in order to reach the best reachable MBB provisioning.

Let P be the overall set of potential rerouting operations, with $P = \bigcup_{k\in K} \bigcup_{t\in T} P^t_k$, where P^t_k is the set of possible routes for connection request $k \in K$ at rerouting event $t \in T$.

The integer linear programming (ILP) formulation of REOPT_SIM uses the following binary variables:

- $z^t_{kp} = 1$ if route $p \in P^t_k$ is selected at rerouting event $t \in T$ for the rerouting of $k \in K$, 0 otherwise.
- C^t_ℓ = required bandwidth on link $\ell \in L$ at rerouting event $t \in T$.

It also uses the following parameters:

- $a^0_{k\ell} = 1$ if link $\ell \in L$ is used in the initial routing of connection request $k \in K$, 0 otherwise.
- $C^0_\ell = \sum_{k\in K} b_k a^0_{k\ell}$ = initial bandwidth usage on link $\ell \in L$.
- $\delta^p_\ell = 1$ if path $p \in P$ uses link $\ell \in L$, 0 otherwise.

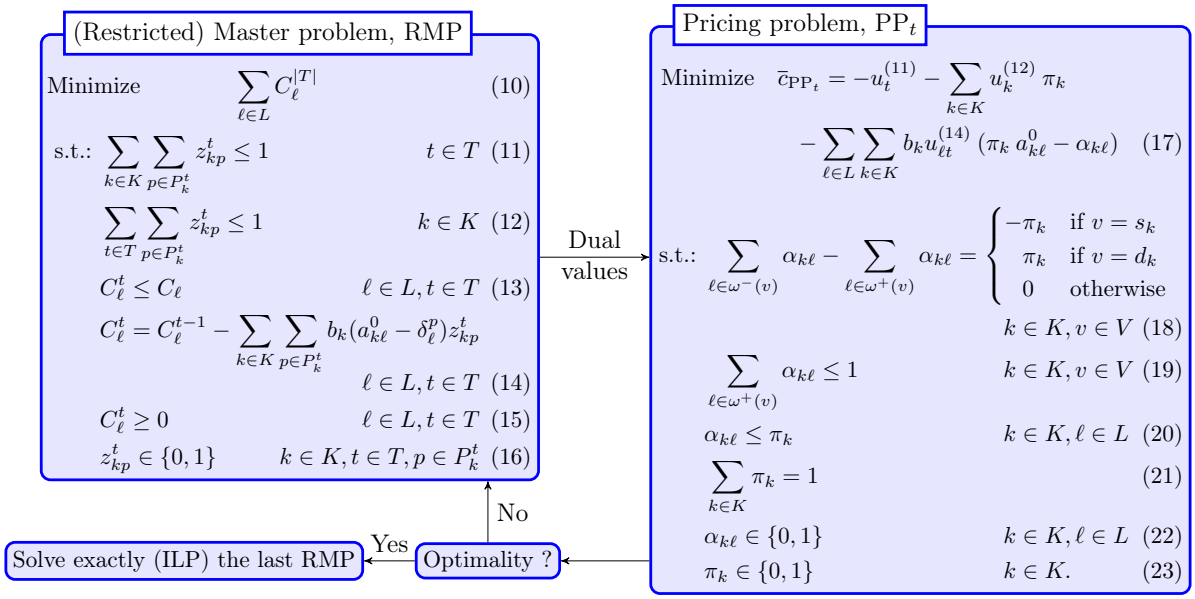


Figure 2: Flow chart of decomposition model REOPT_SIM

The objective of REOPT_SIM is to maximize the spare capacity

$$\max \sum_{\ell\in L}\left(C_\ell - C_\ell^{|T|}\right). \tag{9}$$

As $\sum_{\ell\in L} C_\ell$ is a constant value, the objective can be re-expressed as minimizing the bandwidth usage:

$$[\text{REOPT\_SIM}] \quad \min \sum_{\ell\in L} C_\ell^{|T|} \qquad \text{subject to} \quad (11)-(16).$$

Constraints (11) in Figure 2 prevent the selection of more than one rerouting operation at each rerouting event. Note that $|T| \le |K|$ is an upper bound on the number of rerouting operations as we cannot predict a priori the number of required MBB reroutings. Constraints (12) ensure that a connection request is rerouted at most once. Constraints (13) make sure that transport capacities are never exceeded at any rerouting event. Constraints (14) update the bandwidth usage on link ℓ at rerouting event t, taking into account the unique connection request that has been rerouted at t. Constraints (15)-(16) define the domain of the variables.

Since less that $|T|$ rerouting operations might be necessary to reach the best possible provisioning reachable from the legacy provisioning after at most $|T|$ rerouting operations, it might be desirable to ensure that rerouting operations are performed with consecutive indexes. This can be done adding Constraints (24) to the model (10)-(16). Indeed, Constraints (24) prevent performing a rerouting operation at rerouting event $t+1$ if no rerouting operation is performed at rerouting event t.

$$\sum_{k\in K}\sum_{p\in P_k^{t+1}} z_{kp}^{t+1} \le \sum_{k\in K}\sum_{p\in P_k^t} z_{kp}^t \qquad t\in T. \tag{24}$$

4 Solution Process

4.1 Generic Process

The model (10)-(16) has an exponential number of variables, and therefore column generation [15] is required in order to efficiently solve its linear relaxation.

This technique consists of decomposing the original problem into a Restricted Master Problem (RMP), i.e., model (10)-(16) with a very restricted number of variables, and one or several pricing problems (PPs). In the particular case of model (10)-(16), we will show in the next section that the pricing problem can be decomposed into $|K| \times |T|$ independent smaller pricing problems, each denoted by PP_t^k. The RMP and the PP(s) are solved alternately. Solving the RMP consists in selecting the best connection reroutings, while solving the PPs allows for the generation of new columns, i.e., potential connection routes, and more precisely routes such that, if added to the current RMP, improve the optimal value of its linear relaxation. The process continues until the optimality condition is satisfied, that is, all the so-called reduced costs that define the objective function of the pricing problems are positive (see [15] if not familiar with linear programming concepts). An ε-optimal solution is derived by solving exactly the ILP model associated with the last RMP, with ε defined as follows:

$$\varepsilon = \left(\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}}\right) / z^{\star}_{\text{LP}}, \tag{25}$$

where $z^{\star}_{\text{LP}}$ and $\tilde{z}_{\text{ILP}}$ denote the optimal LP value and the optimal ILP value of the last RMP, respectively. The solution process is illustrated in the flowchart of Figure 3.

4.2 REOPT_SIM Algorithm

Pricing Problem PP_t

Let $u_t^{(11)} \leq 0$, $u_k^{(12)} \leq 0$ and $u_{\ell t}^{(14)} \gtreqless 0$ be the values of the dual variables associated with constraints (11), (12) and (14), respectively.

We use the following binary variables:

- $\pi_k = 1$ if $k \in K$ is selected for rerouting, 0 otherwise.
- $\alpha_{k\ell} = 1$ if $\pi_k = 1$ and the route of $k \in K$ uses link $\ell \in L$, 0 otherwise.

The goal of PP_t (Figure 2) is to select a unique request for potential rerouting, the one with a new route of minimum cost (Objective (17)). Constraints (18) take care of identifying the best possible route, using flow constraints, for the request that is rerouted, i.e., the unique request k such that $\pi_k = 1$. Constraints (19) make sure that we only output simple paths, with no loops. Constraints (20) make sure that routing variables $\alpha_{k\ell}$ are null if request k is not selected for rerouting during rerouting event t, i.e., the rerouting event associated with the PP_t pricing problem. Constraints (21) ensure that each PP_t selects exactly one connection for potential rerouting at rerouting event t. Constraints (22)-(23) define the domain of the variables.

Observe that we can decompose each PP_t into $|K|$ elementary pricing problems PP_t^k, each examining the option of rerouting request $k \in K$ at rerouting event t by setting $\pi_k = 1$ in PP_t. Then, the solution of PP_t is given by

$$\overline{c}_{\text{PP}_t} = \min_{k \in K} \{\overline{c}^k_{\text{PP}_t} : k \text{ is rerouted at rerouting event } t\}, \tag{26}$$

where $\overline{c}^k_{\text{PP}_t}$ denotes the reduced cost of PP_t^k, that we next describe.

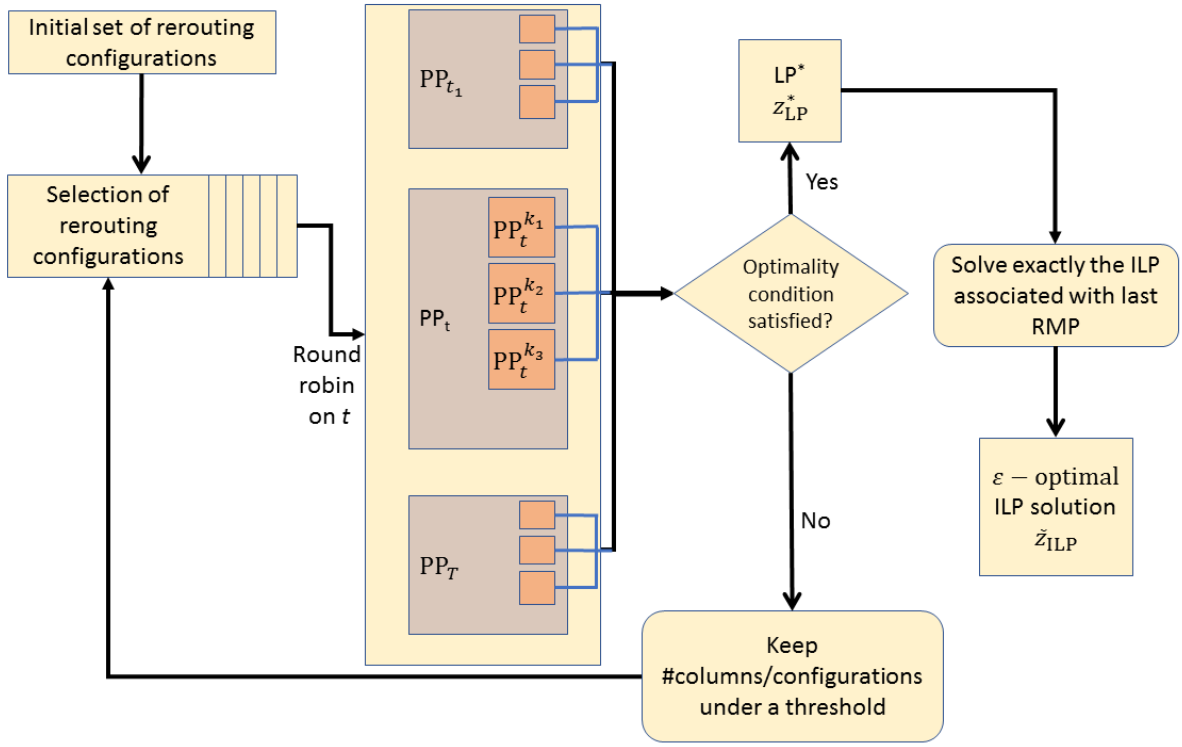


Figure 3: Flowchart of the Proposed Solution Scheme

Elementary Pricing Problem PP_t^k

In each elementary pricing problem PP_t^k, we examine the option of rerouting request k at rerouting event t by setting $\pi_k = 1$ in PP_t. Thus, the ILP formulation of PP_t^k is as follows.

$$\min \bar{c}^k_{\mathrm{PP}_t} = -u_t^{(11)} - u_k^{(12)} - \sum_{\ell \in L} b_k u_{\ell t}^{(14)} (a^0_{kl} - \alpha_{k\ell}) \tag{27}$$

$$\sum_{\ell \in \omega^-(v)} \alpha_{k\ell} - \sum_{\ell \in \omega^+(v)} \alpha_{k\ell} = \begin{cases} -1 & \text{if } v = s_k \\ 1 & \text{if } v = d_k \\ 0 & \text{otherwise} \end{cases}, \quad v \in V \tag{28}$$

$$\sum_{\ell \in \omega^+(v)} \alpha_{k\ell} \leq 1 \qquad v \in V \tag{29}$$

$$\alpha_{k\ell} \in \{0,1\} \qquad k \in K, \ell \in L. \tag{30}$$

Now, observe that PP_t^k is a weighted shortest *simple* path problem in a graph with possibly negative weight cycles. This problem is NP-hard by a reduction from the longest simple path problem (see [16, Section 24.1] or [17, Chapter 5]), and so cannot be solved using only the Bellman-Ford-Moore (BFM) algorithm, even if it takes care of the negative weights. However, we can still use the BFM algorithm, but with some additional tool. As we cannot enforce the simple path condition, the BFM algorithm may fail to output a path with a negative reduced cost, due to the discovery of a negative cycle. In such a case, we then recourse to the solution of the ILP formulation of PP_t^k ((27)-(30)), which includes constraints to enforce the simple path condition.

It is worth noting that calls to the BFM algorithm can be grouped by sources. So $|V|$ calls to BFM suffice to solve PP_t.

Theorem 1. *All pricing problems can be investigated with at most $O(|V|)$ runs of the BFM algorithm, leading to an $O\left(|L| \times (|K| + |V|^2) \times |T|\right)$ time complexity.*

Proof. Note that the reduced cost in (27) can be rewritten:

$$\overline{c}^k_{\mathrm{PP}_t} = \underbrace{-u^{(11)}_t - u^{(12)}_k - \sum_{\ell \in L} b_k u^{(14)}_{\ell t} a^0_{kl}}_{\text{constant}} + \sum_{\ell \in L} b_k u^{(14)}_{\ell t} \alpha_{k\ell}. \tag{31}$$

This entails that the solution of PP^k_t can be reduced to the solution of:

$$[\mathrm{PP}^{\text{generic}}_{tk}] \qquad \min \sum_{\ell \in L} u^{(14)}_{\ell t} \varphi_\ell \quad \text{subject to:} \quad (28) - (30).$$

Observe that problem $\mathrm{PP}^{\text{generic}}_{tk}$ is equivalent to a shortest simple path problem with negative weights, without any guarantee that it contains no negative cycles. Taking into account that *(i)* the coefficients of the objective function are independent of k, and *(ii)* the BFM algorithm can be easily modified in order to output a shortest path tree from a given source node, see, e.g., [16], (i.e., it computes all the (weighted) shortest paths from a given source node), we can then use $|V|$ calls of the BFM algorithm, one from each possible source node, for a given t. Then, for each connection k, we can compute $\overline{c}^k_{\mathrm{PP}_t}$ using (31) and check whether the reduced cost is negative, and, if so, generate a new potential rerouting. For a given t, computing $\overline{c}^k_{\mathrm{PP}_t}$ for all k can be done in $O(|K| \times |L|)$ time, once all $|V|$ BFM calls have been made, and each call to BFM requires time $O(|V| \times |L|)$, hence the overall complexity

$$O\left(|L| \times (|K| + |V|^2) \times |T|\right).$$

□

Solution Process

In order to be done efficiently, the resolution of the $|K|$ Elementary Pricing Problems (PP^k_t) for a given t require their grouping as seen in the previous paragraph. In the context of a column generation model with a large set of different pricing problems, the efficiency of the solution depends on the best combination of linear program re-optimization (i.e., solution of the current Restricted Master Problem), and the solution of the whole set or a subset of pricing problems. We consider the following three options.

1. Re-optimize the current RMP after solving all the elementary PP^k_t associated with a given t, and perform a round-robin on t. Re-optimization of the RMP (Restricted Master Problem) is performed with all the rerouting configurations with a negative reduced cost for a given t.

2. Solve all PP_t with the same set of dual values, and add all the new improving columns simultaneously. Again, either solve exactly each PP_t or stop their solution as soon as one PP^k_t has a negative reduced cost.

3. Solve all the elementary PP^k_t associated with a given t, and add to the RMP the rerouting associated with the smallest reduced cost. Perform a round-robin on t. This strategy focuses on k, instead of on t as in the two previous options. It consists of solving all the elementary PP^k_t associated with a given k, and add to the RMP the rerouting associated with the smallest reduced cost. Perform a round robin on k. The connections can be sorted according to the length of their paths.

Based on a careful analysis of the pros. and cons., we went on with the third option. Details are available in Algorithm 1. Therein, function CostBFM(k, t, RMP) computes the weighted shortest path for connection k with weights defined by the dual values associated with the optimal solution of the current RMP. If we cannot find a shortest path due to a negative cycle (leveraging the CostBFM function), k is included into the set K^{NEG}. When there is no positive reduced cost found by the BFM algorithm and the set K^{NEG} is not empty, we try the exact ILP model (CostILP) for the connections in K^{NEG}, Lines 8-11.

Algorithm 1 Solution Process

Require: RMP_{LP}, Current Linear Relaxation of Restricted Master Problem (RMP)

1: **repeat**
2: **for** $t \in T$ **do**
3: $k' \leftarrow \arg\max_{k \in K}(\text{CostBFM}(k, t, \text{RMP}_{\text{LP}}))$
4: **if** reduced cost of $k' < 0$ **then**
5: add a new column to RMP_{LP}
6: optimize RMP_{LP}
7: **else**
8: $k' \leftarrow \arg\max_{k \in K^{\text{NEG}}}(\text{CostILP}(k, t, \text{RMP}_{\text{LP}}))$
9: **if** reduced cost of $k' < 0$ **then**
10: add a new column to RMP_{LP}
11: optimize RMP_{LP}
12: **until** no new column is found

Generation of an Initial Set of Columns

We use two complementary strategies to generate the initial set of columns. The first strategy is to identify the subset $K_I \subseteq K$ of connection requests that can be rerouted on a shortest path. So, none of the connection requests $k \in K_I$ is routed on a shortest path in the legacy routing. Then, we arbitrarily select at most $|T|$ of these connection requests, and all of them if $|K_I| \leq |T|$, to form the initial set of columns. The first strategy is described in Algorithm 2. Therein, function Shortest_Avail_Path(G, k) finds the shortest path with current spare resources (it may be different from the shortest path without capacity constraints). Note that if the shortest path is sharing links with the current path, the resources on those links are not required twice. The second strategy is used when $|T|$ is large. Let T_1, T_2 and T_3 be pairwise disjoint sets of rerouting events such that $T = T_1 \cup T_2 \cup T_3$ and $|T| = |T_1| + |T_2| + |T_3|$. We use the first strategy to find a subset K_{I_1} of initial columns, with $K_{I_1} \leq |T_1|$. We then use K_{I_1} as initial columns for running the reopt_mbb algorithm with a bound $|T_1|$ on the number of rerouting events. We next obtain as an output a subset $K_1 \subseteq K$ of connection requests. We go on using the first strategy to find a subset K_{I_2} of initial columns, with $K_{I_2} \leq |T_2|$ and $K_1 \cap K_{I_2} = \emptyset$. Then, we use K_{I_2} to run the reopt_mbb algorithm with a bound $|T_2|$ on the number of rerouting events and the extra constraints that connection requests in K_1 cannot be rerouted (i.e., we set $\pi_k = 0$ for all $k \in K_1$). We obtain a subset $K_2 \subseteq K$ of connection requests such that $K_1 \cap K_2 = \emptyset$. We repeat the same procedure to find a subset $K_3 \subseteq K$ of connection requests such that $K_1 \cap K_3 = \emptyset$ and $K_2 \cap K_3 = \emptyset$. Finally, we use $K_1 \cup K_2 \cup K_3$ as the initial set of columns for running the reopt_mbb algorithm with the bound $|T|$ on the number of rerouting events. Observe that $|K_1 \cup K_2 \cup K_3|$ might be less than $|T|$ depending on the instance.

The number of subsets of rerouting events used in the second strategy can be adjusted depending on the value of T. In our experiments, we used only the first strategy when $|T| = 50$, the second strategy with two subsets of size 50 when $|T| = 100$, and the second strategy with

three subsets of size 50 when $|T| = 150$. Reported computation times include the time needed for generating the initial columns and the resolution of the REOPT_MBB algorithm with 50 rerouting events. The second strategy is described in the Algorithm 3. In this algorithm, we demonstrate the case where T is divided into three smaller non-intersecting subsets.

Algorithm 2 Initial Set of Columns - Strategy 1

Require: G, current network state
Ensure: K_I, initial set of columns
1: $K_I \leftarrow \emptyset$
2: #reroutes $\leftarrow 0$
3: **for** $k \in K$ **do**
4: $p^{\text{NEW}} \leftarrow$ SHORTEST_AVAIL_PATH(G, k)
5: **if** p^{NEW} is shorter than k's current path **then**
6: change k's current path to p^{NEW}
7: $K_I \leftarrow K_I \cup \{k\}$
8: #reroutes $\leftarrow$ #reroutes + 1
9: update G's state // route of k is changed
10: **if** #reroutes $= |T|$ **then**
11: break
12: **return** K_I

Algorithm 3 Initial Set of Columns - Strategy 2

Require: G, current network state
Require: $T = T_1 \cup T_2 \cup T_3$, set of rerouting events
Ensure: K_I, initial set of columns
1: $K_I \leftarrow \emptyset$
2: $\#reroutes \leftarrow 0$
3: **for** i from 1 to 3 **do**
4: $K_{I_i} \leftarrow$ initial set of columns using Strategy 1 with G, K, T_i
5: remove K_{I_i} from G
6: $K \leftarrow K \setminus K_{I_i}$
7: $K_I \leftarrow K_I \cup K_{I_i}$
8: **return** K_I

5 Minimum Rerouting: REOPT_MIMO

In this section, we show how to modify the REOPT_SIM model in order to minimize the total number of rerouting operations required to obtain a solution satisfying a given bandwidth requirement (i.e., $BW^\star$ is used in this section). The REOPT_MIMO model, is formalized as follows.

Master Problem

$$\min \quad \sum_{t \in T} \sum_{k \in K} \sum_{p \in P_k^t} z_{kp}^t \tag{32}$$

subject to Constraints (11)-(16) and:

$$\sum_{l \in L} C_l^{|T|} \leq BW^\star. \tag{33}$$

Constraint (33) ensures that the solution must be at least as good as the given bandwidth requirement $BW^\star$, and the Objective (32) is to minimize the total number of rerouting operations.

Observe that the REOPT_MIMO model might not admit a feasible solution if the given upper bound on the bandwidth requirement or on the number of rerouting events is too small. However, if these bounds are at least the objective value ($BW^\star$) obtained from the REOPT_SIM model with the same number $|T|$ of rerouting events, then the REOPT_MIMO model always admits a feasible solution (at least the solution of REOPT_SIM).

Pricing Problem
Let $u_t^{(11)} \leq 0$, $u_k^{(12)} \leq 0$ and $u_{\ell t}^{(14)} \gtreqless 0$ be the values of the dual variables associated with constraints (11), (12) and (14), respectively.

We use the following binary variables:

- $\pi_k = 1$ if $k \in K$ is selected for rerouting, 0 otherwise.

- $\alpha_{k\ell} = 1$ if $\pi_k = 1$ and the route of $k \in K$ uses link $\ell \in L$, 0 otherwise.

$$[\text{PP}^{\text{MIMO}}] \begin{cases} \min & \bar{c}_{\text{PP}_t} = 1 - u_t^{(11)} - \sum\limits_{k \in K} u_k^{(12)} \pi_k - \sum\limits_{\ell \in L} \sum\limits_{k \in K} b_k u_{\ell t}^{(14)} (\pi_k a_{k\ell}^0 - \alpha_{k\ell}). \\ \text{Subject to} & \text{Constraints (18)-(23)} \end{cases}$$

In the new pricing problem PP^{MIMO}, the additional term (to the original pricing problem PP) is a constant, as the variables z_{kp}^t are introduced to the master objective function. Thus, this additional term does not change the pricing problem principle, i.e., the solution process of the REOPT_MBB algorithm can be applied completely to the REOPT_MIMO model.

6 Parallel Rerouting

Enabling parallel rerouting operations allows for reaching either a better provisioning within the specified bound on the number $|T|$ of rerouting events since more individual rerouting operations can be performed, or the same provisioning as REOPT_SIM within less rerouting events. In this section, we explore the latter advantage of parallel rerouting operations.

In the decomposition model REOPT_SIM, Constraints (11) restrict the number of rerouting operations per rerouting events to 1, although some rerouting operations could be safely done in parallel with respect to the MBB paradigm. Indeed, a subset $K' \subseteq K$ of the connection requests can be rerouted in parallel, or concurrently, at rerouting event t if the sum of their bandwidth requirements on the old and new routes does not exceed any link's capacity.

More formally, let $K_r \subseteq K$ be the subset of connection requests that are rerouted by REOPT_SIM, i.e., K_r contains each connection request $k \in K$ such that $\sum_{t \in T} \sum_{p \in P_k^t} z_{kp}^t = 1$. For convenience, we assume that $|K_r| = |T|$. Let $p_k \in \cup_{t \in T} P_k^t$ be the route selected for the rerouting of connection request $k \in K_r$. Recall that a connection request can be rerouted at most once. Let B, indexed by rerouting event $t \in T$, be a set of bins. The bin B^t corresponds to a subset of the connection requests that can be safely rerouted in parallel at rerouting event t. Given a solution of REOPT_SIM, the problem of packing the connection rerouting operations into the minimum number of bins can be formalized as the ILP (34)-(42), using the following variables.

- $q_k^t = 1$ if $k \in K_r$ is packed into bin B^t, for some $t \in T$, 0 otherwise.

- $q^t = 1$ if bin B^t is used.

Observe that the rerouting operations of the connection requests in bin B^t must be performed before those of bin B^{t+1}. Hence, our problem combines a bin packing problem with a scheduling problem.

$$\text{Minimize} \qquad \sum_{t \in T} q^t \tag{34}$$

Subject to:

$$\sum_{t \in T} q_k^t = 1 \qquad k \in K_r \tag{35}$$

$$q_k^t \leq q^t \qquad k \in K_r, t \in T \tag{36}$$

$$q^{t+1} \leq q^t \qquad t \in T \tag{37}$$

$$C_\ell^t \leq C_\ell \qquad \ell \in L, t \in T \tag{38}$$

$$C_\ell^{t-1} + \sum_{k \in K_r} b_k \delta_\ell^{p_k} (1 - a_{k\ell}^0) q_k^t \leq C_\ell \qquad \ell \in L, t \in T \tag{39}$$

$$C_\ell^t = C_\ell^{t-1} - \sum_{k \in K_r} b_k (a_{k\ell}^0 - \delta_\ell^{p_k}) q_k^t \qquad \ell \in L, t \in T \tag{40}$$

$$q_k^t \in \{0, 1\} \qquad k \in K_r,\ t \in T \tag{41}$$

$$q^t \in \{0, 1\} \qquad t \in T \tag{42}$$

$$C_\ell^t \geq 0 \qquad \ell \in L,\ t \in T \tag{43}$$

The goal of this ILP is to minimize the number of bins (Objective (34)), hence minimizing the number of rerouting events needed to perform all parallel operations. Constraints (35) ensure that each connection request is packed into a single bin. Constraints (36) are used to identify used bins. Constraints (37) are used to break symmetries in the solution, ensuring that bin B^{t+1} can be used only if bin B^t is used. Constraints (38)-(40) make sure that the capacity of a link is never exceeded. In particular, Constraints (39) ensure that the "make" part of the MBB operations of the connection requests in bin B^t respect the capacity constraints, i.e., there is enough capacity to establish all the new routes before releasing the capacity used by the legacy routes. Finally, Constraints (40) set the link capacities after the rerouting operations of rerouting event t are made. Constraints (41)-(43) define the domain of the variables.

The ILP formulation (34)-(43) is difficult to solve. Also, we now propose a simple greedy algorithm that packs rerouting operations into bins. Let $\sigma : T \to K_r$ be a mapping from rerouting events to connection requests, such that $\sigma(t)$ indicates the connection request that is rerouted at rerouting event t by REOPT_SIM. That is, $\sigma(t) = \sum_{k \in K_r} \sum_{p \in P_k^t} k \cdot z_{kp}^t$ since Constraints (11) ensure that, for each $t \in T$, a unique variable z_{kp}^t can be set to 1. We denote σ^{-1} the inverse mapping. So $\sigma^{-1}(k)$ is the rerouting event t at which connection request k is rerouted by REOPT_SIM.

Algorithm 4 arranges the connection requests of K_r into bins, each bin corresponding to a set of connection requests that can safely be rerouted in parallel, and so fulfills Constraints (39). It proceeds as follows. After initializing the first bin, it considers the connection requests in the rerouting ordering given by REOPT_SIM (Lines 3-8). If the addition of connection request k to the current bin results in a violation of Constraints (39), then a new bin is created and k is added to it. Then the algorithm considers the next connection requests until all connection requests have been placed into a bin.

Algorithm 4 ensures that if connection request k is placed into bin B^{t+1}, then it satisfies $\sigma^{-1}(k) > \sigma^{-1}(k')$ for all $k' \in B^t$. In other words, the connection requests in B^t are rerouted

Algorithm 4 Parallel Rerouting with respect to Original Order

Require: σ, mapping from rerouting events to connection requests
Require: T, set of rerouting events in original ordering
Ensure: B, bins containing requests rerouted in parallel

```
B^1 ← ∅                                              // Initialize the first bin
i ← 1                                                // Index of the current bin
for t from 1 to |T| do
    k ← σ(t)
    if adding k to B^i violates Constraints (39) then
        i ← i + 1
        B^i ← ∅                                      // Create a new bin
    B^i ← B^i ∪ {k}                                  // Add k to current bin
return B
```

before those in B^{t+1}, which is consistent with the solution given by REOPT_SIM. Furthermore, the link capacity after the rerouting operations of bin B^t is exactly the same as the link capacity in the solution of REOPT_SIM after the rerouting of connection request $k = \sigma(\sum_{i=1}^{t} |B^i|)$. The number of bins created by Algorithm 4 depends on the solution given by REOPT_SIM. In the worse case, when no parallel rerouting operation is possible, it creates $|T|$ bins.

7 Numerical Results

7.1 Data Sets

We consider a network with 32 nodes and 250 directed links, which corresponds approximately to a Ciena customer network. Existing network connections were used to construct a traffic matrix input to a network simulator generating realistic random connection states. Connection requests had Poisson arrivals based on the traffic matrix and random durations drawn from a common exponential distribution. Each connection had a Weibull distributed bandwidth with a coefficient of variation of 0.3. Connections were routed on the shortest path (in hop count) that had sufficient bandwidth. A load factor parameter was used to globally vary the connection arrival rates: the corresponding equilibrium (after simulation start-up transients had disappeared) connection states represent a range of congestion levels from light (0.5) to heavy (1.0). For each load factor, we considered 10 reoptimization events. Reoptimization was performed with a period of 1 mean connection duration, ensuring that sufficient connection request arrival and termination events occurred in order to produce comparably degraded connection state. Characteristics of the data sets are described in Table 1, where for each load factor, we provide the average number of granted requests right before each reoptimization and the average number of connection requests that are *not* initially routed on a shortest path.

7.2 Comparison with the Model of Klopfenstein [9]

We compared the performance of our model and algorithm with the model of Klopfenstein [9]. We use a dataset with a load factor of 0.5, $|T| = 40$ and for each reoptimization event, only 250 connections from the original connection set are taken into account. The differences are significant even for such small instances, as indicated by the results reported in Table 2. Therein, we report the results for each of the 10 reoptimization events. Since the Klopfenstein model can take enormous time to finish, we report the best solution found within a computation time limit of one hour. Note that due to these computation times, we were not able to perform comparisons

Table 1: Characteristics of the data sets

Load factor	Number of requests	# Req. not on shortest path
0.5	777.2	90.2
0.6	894.6	199.1
0.7	951.4	252.3
0.8	971.1	268.3
0.9	993.3	285.7
1.0	1,015.4	295.7

Table 2: Comparison with Klopfenstein: Traffic load 0.5, $|T| = 40$, number of connection requests = 250

Defrag. events	Bandwidth saving (%)		Accuracy ε (%)		# reroutings		Computing times (sec.) (limit = 1 hour)	
	REOPT_SIM	Klopfenstein [9]	REOPT_SIM	Klopfenstein [9]	REOPT_SIM	Klopfenstein [9]	REOPT_SIM	Klopfenstein [9]
210	5.7	5.7	0.00	0.00	16.0	21.0	14.7	2316.7
220	4.0	4.0	0.00	0.00	10.0	12.0	14.4	1020.8
230	3.4	2.9	0.00	0.55	9.0	8.0	13.4	**limit**
240	1.7	1.7	0.00	0.00	5.0	16.0	11.9	755.3
250	2.3	2.3	0.00	0.00	8.0	9.0	14.1	859.1
260	7.5	7.5	0.00	0.00	13.0	15.0	15.1	802.8
270	5.7	5.9	0.27	0.00	14.0	19.0	7.2	2249.1
280	3.0	3.0	0.00	0.00	9.0	11.0	13.9	895.1
290	2.3	2.3	0.00	0.00	6.0	13.0	13.9	479.1
300	7.1	4.5	0.00	2.81	17.0	11.0	11.6	**limit**
Average	4.3	4.0	0.03	0.34	10.7	13.5	13.0	1661.0
Ratio	1.1		0.08		0.8		0.0078 ($\approx 1/130$)	

on larger instances. Columns entitled REOPT_SIM correspond to the results obtained with the REOPT_SIM model and REOPT_MBB algorithm.

As expected, our model can be solved orders of magnitude faster than the compact ILP model of Klopfenstein [9], that is, about 130 times faster. Furthermore, the computation time limit of one hour was reached for two instances with the compact ILP model (reoptimization events indexed as 230 and 300). Moreover, our model yields on average a better accuracy (i.e., 0.03% versus 0.34%), because Klopfensteins's model was stopped by the time limitation twice, resulting in significant optimality gaps.

Obviously, when optimal solutions are obtained with both models, the bandwidth savings are equal. However, for the instances indexed 230 and 300 that were not optimally solved with the model of Klopfenstein, the solutions computed with our model offer better bandwidth savings. In addition, the solutions obtained with our model involve on average less rerouting operations. This can be explained by the facts that firstly, the minimization of the number of rerouting operations is part of none of the objective functions of the models, and that secondly, the model of Klopfenstein allows a connection request to be rerouted more than once.

7.3 Accuracy and Performance of the Reoptimization Solution

All statistics computed in this section correspond to averages over all the 10 reoptimization events performed for each load factor.

Recall that we initialize the REOPT_MBB algorithm with at most $|T|$ potential rerouting operations (columns). When $|T| = 50$, we use the first strategy presented in Section 4 for generating the initial set of columns. When $|T| = 100$ or $|T| = 150$, we use the second strategy. We have reported in Table 1 the average number of connection requests that are not routed on a

Table 3: Impact of the Initial Rerouting Configurations and Overall Number of Configurations

Load	# Initial potential reroutings			# Initial potential reroutings in the optimal solution						Overall # of generated potential reroutings		
	T = 50	T = 100	T = 150	T= 50		T = 100		T = 150		T = 50	T = 100	T = 150
0.5	43.3	68.1	70.0	18.5	(39.1%)	44.9	(63%)	42.0	(57.6%)	595.7	797.0	1401.0
0.6	50.0	99.0	145.4	15.3	(30.9%)	76.8	(77.4%)	112.8	(77.4%)	590.4	1157.7	1738.9
0.7	50.0	99.2	149.0	12.6	(25.4%)	63.9	(64.5%)	113.1	(75.8%)	546.0	1010.2	1408.2
0.8	50.0	98.6	148.5	14.1	(28.9%)	68.3	(69.4%)	110.2	(74.4%)	469.2	911.5	1348.5
0.9	50.0	99.1	149.1	12.6	(25.6%)	65.4	(66%)	122.8	(82.3%)	504.2	950.2	1283.4
1.0	50.0	99.5	149.5	12.1	(24.4%)	73.0	(73.7%)	120.5	(80.9%)	520.8	827.8	1207.8

shortest path in the legacy routing, and in Table 3 the average size of the initial sets of potential rerouting operations (initial columns) as produced by the first or second strategy. These sets are given as inputs to the REOPT_MBB algorithm. The relatively low number of initial potential reroutings for the instances with traffic load 0.5 is explained by the small number of connection requests (90.2 in average) that are not routed on a shortest path in the legacy routing.

Concerning the choice of $|T|$, which can be as large as the cardinality of the whole connection set, we conducted our experiments for values of $|T|$ of at most 150 in order to limit the computation time of our model to about one hour.

We now analyze the results reported in Table 3 on the efficiency of our strategies for generating the initial potential reroutings. Observe that the number of potential reroutings increases with the traffic load due to the increased number of routes that are not the shortest possible ones. For $|T| = 50$, where the initial potential reroutings are produced by the first strategy only, we observe in the middle set of columns of Table 3 that at most 39.1% of the initial potential reroutings appear in the final solutions. But, for $|T| = 100$ and $|T| = 150$, where we use the second strategy, at least 57.6% and up to 82.3% of these initial potential reroutings are part of the final solutions. This means that the time spent by the second strategy in the resolution of sub-problems results in very good choices of initial potential reroutings. Furthermore, it has a strong impact on the total number of generated potential reroutings, reported in the last set of columns of Table 3, and so on the number of times the pricing problem PP is solved. Indeed, the pricing problem generates a potential rerouting only if it is expected to improve the current solution of REOPT_MBB algorithm. Hence, when many initial potential reroutings are actually part of the final solution, fewer calls to the pricing problem are needed to solve the problem.

Table 4: Number of Reroutings and Accuracy of the Solutions

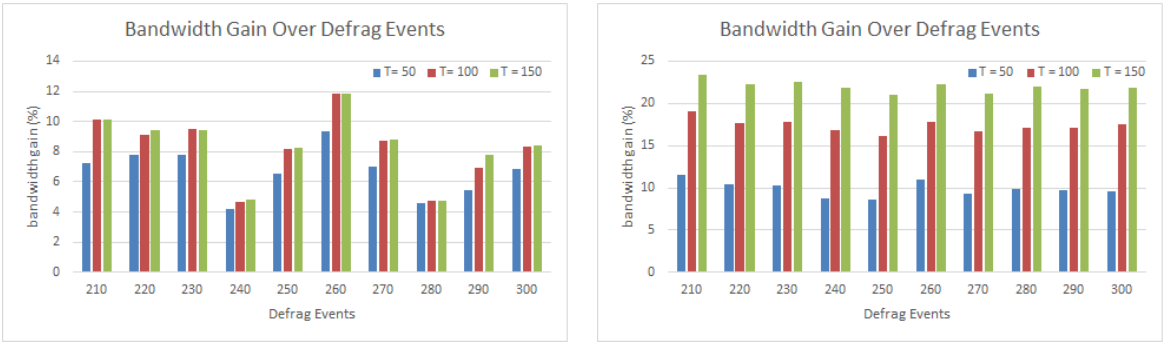
Load	Number of required rerouting			Accuracy (ε)			Computational times in minutes			ILP Pricing Problem computational times in seconds (number of solved ILPs)		
	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150
0.5	47.2	71.2	72.9	1.2	1.7	1.5	1.5	7.1	16.1	0.0 (0.0)	0.0 (0.1)	0.0 (0.2)
0.6	49.4	99.1	145.6	2.0	1.6	1.6	2.0	20.4	89.7	0.3 (1.1)	0.2 (2.8)	3.6 (1.5)
0.7	49.6	99.0	149.1	2.4	1.3	1.2	1.8	20.0	65.9	2.0 (4.6)	0.7 (2.9)	2.2 (1.8)
0.8	48.7	98.4	148.1	2.7	1.6	1.2	1.7	15.8	63.2	1.5 (3.8)	2.0 (7.5)	2.1 (9.8)
0.9	49.1	99.0	149.1	2.2	1.3	1.2	2.0	16.8	64.3	0.4 (1.3)	0.6 (2.9)	9.7 (15.6)
1.0	49.5	99.0	148.9	2.3	1.4	1.0	2.2	14.3	45.9	1.8 (4.0)	4.0 (7.1)	3.7 (4.4)

In Table 4, we report on the accuracy of the solutions. We first observe that the accuracy of computed solutions is always between 1% and 3%, which is satisfactory taking into account the additional computation time it would require for getting an optimal solution with a branch-and-price method [18], instead of the current solution process (Section 4). Furthermore, solutions obtained for larger values of $|T|$ have a better (smaller) accuracy, except for load factor 0.5. This shows that the proposed solution process performs very well even on large instances.

Computation times, which include the generation of initial potential reroutings, are also quite reasonable, although too long for a real-time reoptimization operation. However, we expect to reduce them significantly in the near future with the addition of heuristics to speed up the solution process. Moreover, we observe that the overall computation time due to the resolution of the pricing problems using ILPs, which occur when the BFM algorithm fails to find a solution, is negligible. In the last part of Table 4, the main number is the averaged computing time spent on the ILP executions and the number in the parentheses is the averaged number of executions. We need less than 10 seconds (for at most 16 executions) for solving the ILPs of the pricing problems. This supports the effectiveness of the decomposition of PP into PP_t^k subproblems, each solved independently with the BFM algorithm.

7.4 Reoptimization Performance

We investigated the reduction of the overall bandwidth requirements after each reoptimization event. We have reported in Figure 4 the reduction after each reoptimization event for the two extreme load factors, i.e., 0.5 and 1.0. As already anticipated with the results of Table 4, enabling more than $|T| = 100$ rerouting operations for the instances with load factor 0.5 does not help further reducing the overall bandwidth usage. However, with load factor 1.0, enabling more rerouting operations allows for significant reduction of the bandwidth usage. More precisely, increasing $|T|$ from 50 to 100 leads to more than 5% gain, and increasing $|T|$ from 100 to 150 provides 5% additional bandwidth gain. An example of bandwidth usage evolution during a sequence of rerouting operations is depicted in Figure 5 for the case of load factor 0.5 and $|T| = 50$. We observe that the bandwidth usage is essentially monotonically decreasing, although local increases of the bandwidth usage are possible with our model that only ensures that the final bandwidth usage is minimized.



(a) Load factor = 0.5

(b) Load factor = 1.0

Figure 4: Reduction (%) of bandwidth requirement

7.5 Multiple Rerouting

In order to confirm the benefit of enabling multiple rerouting operations per connection requests, i.e., a connection is allowed to be rerouted more than once, we applied REOPT_SIM consecutively, taking as input the routing obtained in the previous call. More precisely, after first solving the problem with $|T| = 50$, we solve it again with $|T| = 50$, starting from the routing R_1 computed by the first execution of REOPT_SIM, and then again producing R_3 taking as input the routing R_2 resulting from the second call.

In the first set of columns of Table 5, we ensure that a connection request is rerouted at most once. This is done by setting $\pi_k = 0$ for the connection requests that were previously rerouted.

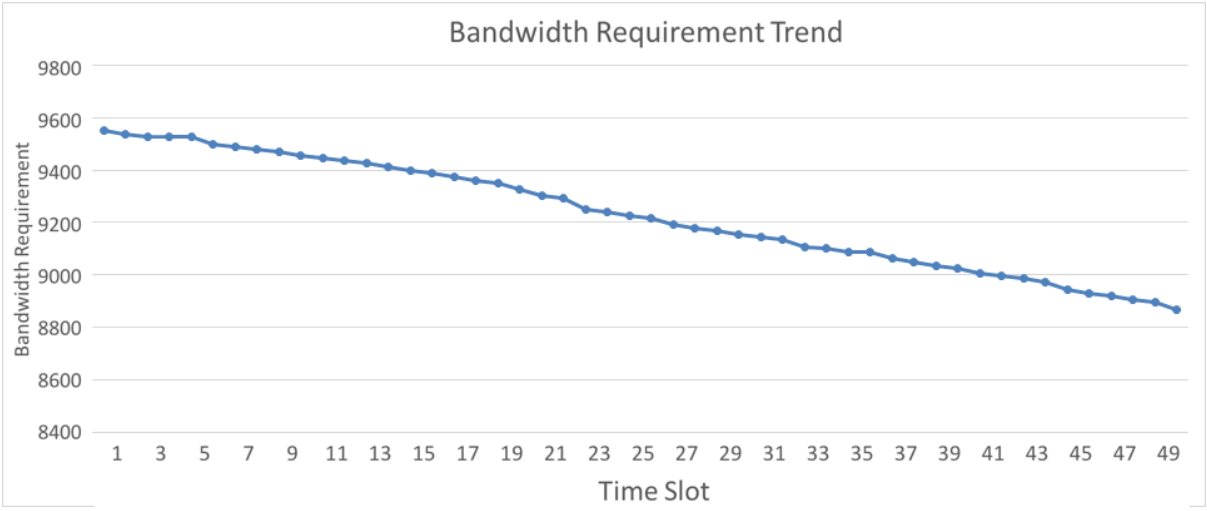


Figure 5: Trend of bandwidth usage after each rerouting operation on load 0.5 within $|T| = 50$.

In the second set of columns of Table 5 we allow the reroute in R_2 (resp. R_3) of a connection request that has already been rerouted in R_1 (resp. R_1 or R_2). Hence, a connection request can be rerouted up to three times. We observe a small improvement in the bandwidth gain (around 0.1% for R_3) when enabling up to three reroutings per connection request.

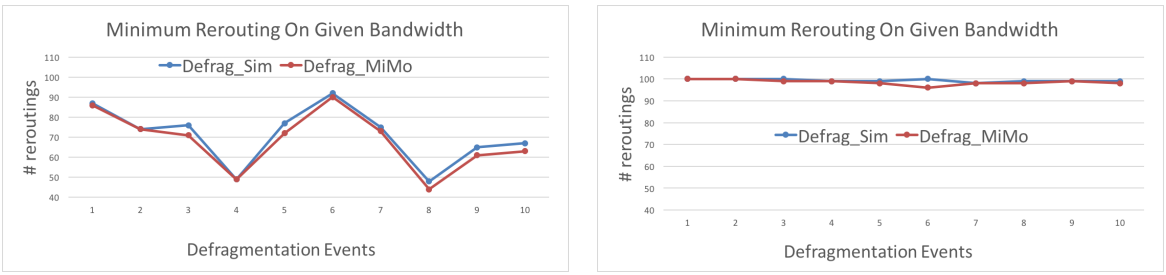
Table 5: Impact of Multiple Rerouting

Load	At most one rerouting per connection & t						Multiple rerouting per connection					
	# reroutings			bandwidth gain (%)			# reroutings			bandwidth gain (%)		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
0.5	47.2	20.9	1.9	6.7	8.0	8.2	47.2	21.0	2.5	6.7	8.1	8.2
0.6	49.4	49.6	46.4	9.4	14.9	18.4	49.6	49.3	46.3	9.4	14.8	18.5
0.7	49.6	49.6	49.8	9.8	16.6	21.0	49.1	49.7	49.8	9.8	16.6	21.1
0.8	48.7	49.9	49.9	9.9	16.8	21.5	48.7	49.5	49.6	9.9	16.8	21.4
0.9	49.1	50	50	10.1	17.2	22.1	49.4	49.6	50.0	10.1	17.2	22.2
1	49.5	50	50	9.9	17.1	21.9	49.7	49.8	50.0	9.9	17.1	21.9

7.6 Minimum Rerouting

The REOPT_MIMO model (Section 5) aims at minimizing the number of rerouting events to obtain a solution with specified quality (i.e., satisfying a given upper bound on the bandwidth usage). Figure 6 illustrates the results of REOPT_MIMO and REOPT_SIM for the instances with load factor 0.5 and 1.0, when $|T| = 100$. We observe that the reduction in the number of rerouting events offered by REOPT_MIMO over REOPT_SIM is small. With a 0.5 load factor, there are more connections which are being rerouted on their ultimate shortest path in terms of the number of links. In addition, although it is allowed to use up to $|T| = 100$ reroutes, it is usually not necessary to have so many before reaching an optimized or optimal rerouting (e.g., for events 4 and 8, only about half of the allowed number of reroutes are needed). With a 1.0 load factor, i.e., in a competitive resource environment, there are many more connections that are routed over longer paths than the shortest possible. As a result, the solution must use more re-routings (more than 95 reroutings for all events). The reason is that the objective

of minimizing the number of rerouting operations was implicitly involved into the REOPT_SIM model, since it is parameterized by $|T|$.



(a) Load factor = 0.5

(b) Load factor = 1.0

Figure 6: Number of Rerouting operations with REOPT_SIM and REOPT_MIMO

7.7 Parallel Rerouting

Table 6 demonstrates the efficiency of the parallel rerouting methodology (Section 6) in terms of the number of required rerouting events, i.e., a rerouting event will perform several rerouting operations in parallel provided that they do not violate the capacity limitation. Assuming that the duration of a rerouting event made of parallel rerouting operations is the same (or almost the same) as a single rerouting operation, applying parallel rerouting operations reduces by a factor 10 to 30 the cost of the reconfiguration procedure. In addition, the computing time for the REOPT_PAR model is remarkably less than for the REOPT_SIM model.

Table 6: Number of Parallel Rerouting Events

Load	Heuristic (Algorithm 4)			REOPT_PAR			REOPT_PAR Computational Time (min.)		
	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150	T = 50	T = 100	T = 150
0.5	7.5	12.2	12.8	4.6	6.1	5.6	3.6	5.8	6.0
0.6	9.4	16.6	20.7	5.1	6.6	7.5	3.9	8.4	13.8
0.7	9.2	16.5	21.1	5.5	7.0	6.7	3.9	8.5	14.2
0.8	9.2	16.3	21.2	5.3	7.4	6.8	3.9	8.4	12.9
0.9	8.6	16.2	20.3	4.8	6.2	6.3	4.0	8.4	12.9
1.0	10.5	17.6	21.8	5.9	6.4	6.3	4.1	8.5	13.5

8 Conclusion

We have proposed a new model for progressive reoptimization, i.e., computing a sequence of make-before-break reroutings leading to the minimum bandwidth requirements. It corresponds to a huge improvement with respect to the previous model proposed by Klopfenstein [9] as we reach up to 150 reroutings in less than a few hours, while the model of [9] was only scalable for toy problems.

We plan to further study the proposed model so that it can handle the case of more than one rerouting per (selected) connection. In addition, we plan to study the adaptation of our optimization model to other seamless migrations, e.g., to requests in Content Delivery Networks (CDNs), [19].

Acknowledgments

B. Jaumard has been supported by a Concordia University Research Chair (Tier I) and by an NSERC (Natural Sciences and Engineering Research Council of Canada) grant. H. Duong was supported by a MITACS & Ciena Converge Fellowship. D. Coudert has been supported by the French National Research Agency (ANR), through the UCA$^{\text{JEDI}}$ Investments in the Future project with the reference number ANR-15-IDEX-0001, and the Inria associated-team project EfDyNet.

References

[1] H. Li and J. Wu, "Survey of WDM network reconfiguration: topology migrations and their impact on service disruptions," *Telecommunication Systems*, vol. 60, pp. 349–366, Nov. 2015.

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

[3] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J.-S. Sereni, "Reconfiguration of the routing in WDM networks with two classes of services," in *Conference on Optical Network Design and Modeling - ONDM*, 2009, pp. 1–6.

[4] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse, "Tradeoffs in process strategy games with application in the WDM reconfiguration problem," *Theoretical Computer Science*, vol. 412, no. 35, pp. 4675–4687, 2011.

[5] N. Jose and A. K. Somani, "Connection rerouting/network reconfiguration," in *Proceedings of IEEE/VDE Workshop on Design of Reliable Communication Networks - DRCN*, 2003, pp. 23–30.

[6] F. Solano, "Analyzing two conflicting objectives of the WDM lightpath reconfiguration problem," in *IEEE Global Telecommunications Conference - GLOBECOM*, Nov. 2009, pp. 1–7.

[7] F. Solano and M. Pióro, "Lightpath reconfiguration in WDM networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 2, no. 12, pp. 1010 – 1021, December 2010.

[8] A. Kadohata, A. Hirano, F. Inuzuka, A. Watanabe, and O. Ishida, "Wavelength path reconfiguration design in transparent optical WDM networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 5, no. 7, pp. 751 – 761, July 2013.

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

[10] D. Coudert, D. Mazauric, and N. Nisse, "Experimental evaluation of a branch-and-bound algorithm for computing pathwidth and directed pathwidth," *ACM Journal of Experimental Algorithmics*, vol. 21, no. 1.3, pp. 1–23, 2016.

[11] 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)*, October 2003, pp. 61–66.

[12] D. Coudert, D. Mazauric, and N. Nisse, "On rerouting connection requests in networks with shared bandwidth," *Electronic Notes in Discrete Mathematics*, vol. 32, pp. 109–116, 2009.

[13] S. Brandt, K.-T. Förster, and R. Wattenhofer, "On consistent migration of flows in SDNs," in *IEEE Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*, 2016, pp. 1–9.

[14] D. Coudert and J.-B. Sereni, "Characterization of graphs and digraphs with small process number," *Discrete Applied Mathematics*, vol. 159, no. 11, pp. 1094–1109, Jul. 2011.

[15] V. Chvatal, *Linear Programming*. Freeman, 1983.

[16] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge: The MIT Press, 2001.

[17] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

[18] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, no. 3, pp. 316–329, 1998.

[19] J. Lai, Q. Fu, and T. Moors, "Using SDN and NFV to enhance request rerouting in ISP-CDN collaborations," *Computer Networks*, vol. 113(1), pp. 176–187, Fev. 2017.