Be Scalable and Rescue My Slices During Reconfiguration

Adrien Gausseran, Frédéric Giroire, Brigitte Jaumard, Joanna Moulierac

▶ **To cite this version:**

Adrien Gausseran, Frédéric Giroire, Brigitte Jaumard, Joanna Moulierac. Be Scalable and Rescue My Slices During Reconfiguration. ICC 2020 - IEEE International Conference on Communications, Jun 2020, Dublin, Ireland. pp.1-6, 10.1109/ICC40277.2020.9148871 . hal-02945405

HAL Id: hal-02945405
https://hal.science/hal-02945405

Submitted on 22 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Be Scalable and Rescue My Slices During Reconfiguration

A. Gausseran*, F. Giroire*, B. Jaumard †, J. Moulierac *
* Université Côte d'Azur, I3S, CNRS, Inria, University of Nice Sophia Antipolis, France
† Concordia University, Canada

***Abstract*—Modern 5G networks promise more bandwidth, less delay, and more flexibility for an ever increasing number of users and applications, with Software Defined Networking, Network Function Virtualization, and Network Slicing as key enablers. Within that context, efficiently provisioning network and cloud resources of a wide variety of applications with dynamic users' demands is a real challenge. In this work, we consider the problem of network slice reconfiguration. Reconfiguring from time to time network slices allows to reduce the network operational costs and to increase the number of slices that can be managed within the network. However, it impacts users' *Quality of Service* during the reconfiguration step. To solve this issue, we study solutions implementing a *make-before-break* scheme. We propose new models and scalable algorithms (relying on column generation techniques) that solve large data instances in few seconds.**

I. Introduction

The Network Function Virtualization (NFV) paradigm is a major technology of 5G networks. Over the past decade, it has been widely deployed and a large number of studies investigated its use and benefits. Its core principle is to break the dependence on dedicated hardware like traditional expensive middleboxes by allowing network functions (e.g., firewall, load balancing, Virtual Private Network (VPN) gateways, content filtering) to be virtualized and implemented in software, and executed on generic servers. Virtual Network Functions (VNFs) can be instantiated and scaled on demand without the need to install new equipment, increasing flexibility with user demands [1]. In parallel, we also saw the emergence of Software-Defined Networking (SDN) that simplifies network monitoring and management. By decoupling the control plane from the data plane and abstracting network intelligence into a central controller, SDN allows a global vision and control of the network [2]. Combination of SDN and NFV leads to dynamic, programmable and flexible networks in which the network infrastructure and resources are shared between network services.

The 5G technology is envisioned to allow a multi-service network supporting a wide range of communication scenarios with a diverse set of performance and service requirements. The concept of network slicing has been proposed to address these diversified service requirements. A network slice is an end-to-end logical network provisioned with a set of isolated virtual resources on a shared physical infrastructure [3], [4]. Moreover, slicing allows an efficient usage of resources, as VNFs can be instantiated and released on demand by slices. Besides, slices can be deployed whenever there is a service request, reducing the network operator costs [4]. With all these key features, Network slicing will thus be a fundamental feature of 5G networks [3].

Dynamic resource allocation is one of the key challenges of network slicing. In a dynamic scenario, the network state changes continuously due to the arrival and departure of flows. As the granting of new flows is done without impacting the ongoing ones, we may end up with a fragmented provisioning, and thus with an inefficient resource usage. Therefore, network operators must adjust network configurations in response to changing network conditions to fully exploit the benefits of the SDN and NFV paradigms, and to minimize the operational cost (e.g., software licenses, energy consumption, and Service Level Agreement (SLA) violations).

We here consider the problem of both rerouting traffic flows and improving the mapping of network functions onto nodes in the presence of dynamic traffic, with the objective of bringing the network back to a close to optimal operating state, in terms of resource usage. Rerouting demands and migrating VNFs take several steps. Usually, network carriers/operators cannot afford traffic interruption, due to their SLAs, as it may have a non-negligible impact on the Quality of Service (QoS) experienced by the users. Their strategy is then to perform the reconfiguration by using a two–phase approach. First, a new route is established while keeping the initial one enabled (i.e., two redundant data streams are both active in parallel). Then, the transmission moves on the new route and the resources used by the initial one are released. This strategy is often referred to as *make-before-break*. In this work, to the best of our knowledge, we are the first to propose scalable models to reconfigure network slices while implementing such mechanisms to avoid QoS degradation.

Our contributions in this paper are as follows:

- We propose two *scalable* models, `rescue-ILP` and `rescue-LP`, with `rescue` standing for "REconfiguration of network Slices with ColUmn gEneration without interruption". Both are based on a decomposition model and are solved using column generation. Our algorithms reconfigure a given set of network slices from an initial routing and placement of network functions to another solution that improves the usage of the network resources

This work has been supported by the French government through the UCA JEDI (ANR-15-IDEX-01) and EUR DS4H (ANR-17-EURE-004) Investments in the Future projects, and by Inria associated team EfDyNet.

(both in terms of links and VNFs). Our solutions scale on large networks as we succeeded in solving data instances with 65 nodes and 108 links, and a hundred of network slices in few seconds, a lot faster than with a classic compact Integer Linear Program (ILP) formulation such as `slow-rescue`.

- We show that our solutions allow *the decrease of the network cost* without degrading the QoS (as the network slices are not interrupted thanks to the *make-before-break* approach) in moderate running times. Moreover, we can manage more network slices when the network is congested compared to solutions without any reconfiguration.

II. Related Work

In the last years, a large corpus of works has studied the deployment and management of network services, see [5] and [6] for surveys. In particular, the problem of jointly routing demands and provisioning their needed VNFs has attracted a lot of attention. A large number of efficient algorithms and optimization models have been proposed in order to minimize setup cost [7], [8] or take into account the chaining constraints [9], [10]. Most of these works have only considered scenarios in which, when a service is deployed, its route and used virtual resources are not changed during its lifetime. However, even with an optimal service deployment, this may lead to a sub-optimal use of network resources after some time, when some services have left.

Inspired by the classic defragmentation mechanism in optical networks [11], it has been proposed to carry out reconfigurations of network and virtual resources regularly in order to bring the network closer to an optimal state of operation. The goals can be diverse: optimizing network usage, granting more requests, modifying the capacities of flows already allocated on the network or even to overcome network failures.

The readjustment of Service Function Chains (SFCs) has been studied in [12]. The authors formulate an ILP and a column generation model in order to jointly optimize the deployment of the SFCs of new users and the readjustment of the SFCs already provisioned in the network while considering the trade-off between resource consumption and operational overhead. [13] studies the trade-off between the reconfiguration of SFCs and the optimality of the reconfigured routing and placement solution.

Gao and Rouskas [14] considered the reconfiguration of virtual networks. They proposed online algorithms to minimize the maximum utilization of substrate nodes and links while bounding the number of virtual nodes that have to be migrated.

Recently, the problem has been studied for network slices. [15] proposes a hybrid slice reconfiguration mechanism. The goal of the authors is to optimize the profit of a network slice provider, i.e., the total utility gained by serving slices minus the resource consumption and reconfiguration cost. The reconfiguration overhead of a slice includes two aspects: service interruption and reconfiguration resource cost.

Similarly, all works on reconfigurations of virtual resources (virtual networks, slices or service function chains) include a cost expressing the degradation of the client's QoS. On the contrary, our goal is to *avoid this QoS degradation* by proposing a *make-before-break* mechanism, in which the new route is reserved and the new virtual resources are installed before the slice is reconfigured. A similar mechanism has been proposed in [16]. However, we are the first to propose a scalable decomposition model based on column generation.

III. Problem Statement and Notations

We consider the network as a directed capacitated graph $G = (V, E)$ where V represents the node set and E the link set. C_u is the resource node capacity (e.g., CPU, memory, and disk) of node $u \in V$. C_ℓ is the bandwidth link capacity and Γ_ℓ is the link delay of link $\ell \in E$. $t \in \{1, T\}$ is the number of steps used for the reconfiguration. Δ_f is the number of bandwidth units required by function $f \in F$.

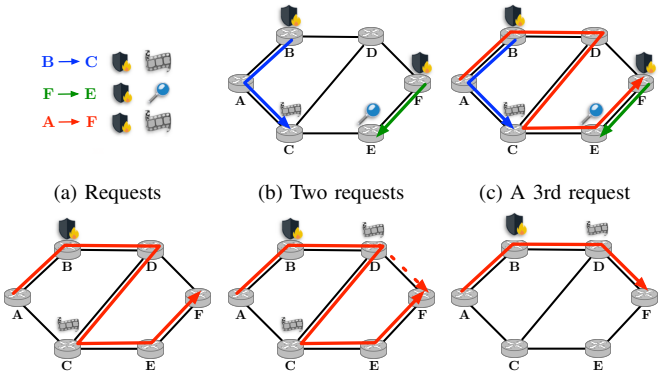
A slice can be modeled by a set of demands following for example [17], [18]. Each demand $d \in D$ is modeled with a quintuplet: v_s the source, v_d the destination, c_d the ordered sequence of network functions that need to be performed, where $f_l^{c_d}$ is the $l-th$ function of the chain of c_d. BW_d the required units of bandwidth, and γ_d the delay requirement.

In a dynamic scenario with no information on future traffic, each demand is routed individually while *minimizing the network operational cost* defined by the weighted sum of link bandwidth and VNF usage costs (licenses, energy consumption, etc). As requests come and leave over time, allocations that are locally optimal at a given instant can bring the network in a global sub-optimal state. Our goal is to reconfigure the network to improve resource usage and therefore the operational costs. In doing this, we use the *make-before-break* mechanism to avoid network service disruption due to traffic rerouting.

1) Example: Figure 1 illustrates an example for the reconfiguration of a request using a *make-before-break* process. Two requests, B to C and F to E are routed during step (b). Four VNFs have been installed in B, C, E and F to satisfy the needs of these requests. To avoid the usage cost of new VNFs, the route from A to F with minimum cost is a long 5–hops route (step (c)). When requests from B to C and from F to E leave, the request is routed on a non-optimal path (step (d)) which uses more resources than necessary. We compute one optimal 3-hop path and reroute the request on it (step (f)) with an intermediate make-before-break step (step (e)) in which both routes co-exist. In this example, the reconfiguration can be done with only one step of reconfiguration, but we will consider in the following up to 3 steps of reconfiguration.

IV. Optimization models

As we will see in Section V, although effective, the compact ILP model `slow-rescue` (the complete description may be found in [19]) does not scale on large networks or with many slices. We therefore propose an alternative using column generation: `rescue-ILP` and `rescue-LP` (for REconfiguration of network Slices with ColUmn gEneration with ILP or LP pricing).



(a) Requests (b) Two requests (c) A 3rd request

(d) The first two requests leave (e) Reconfiguration phase (f) Optimal routing for the third request

Fig. 1: An example of the reconfiguration of a request using a *make-before-break* approach with one step.

A. Description of our CG-based algorithms: `rescue-ILP` and `rescue-LP`

Column generation (CG) is a model allowing to solve an optimization model without explicitly introducing all variables, see Figure 2 for an explanation. It thus often allows to solve larger instances of the problem than a compact model, in particular, with an exponential number of variables. In our model, the master problem (MP) seeks a possible global reconfiguration for all slices with a path-formulation. In the restricted master problem (RMP), only a subset of potential paths is used for each slice. At the initialization, the set of paths is the one used before reconfiguration. Each pricing problem (PP) then generates a new path for a request, together with the placement of the VNFs. During a reconfiguration, slices are migrated from one path to another. Note that, as the execution of each pricing problem is independent of the others, their solutions can be obtained in parallel.

Layered graph. Our models are based on the concept of a layered graph presented in [19]. In order to model the chaining constraint of a demand, we associate to each demand d a layered graph $G^L(d)$ with $|c_d|$ layers where $|c_d|$ denotes the number of VNFs in the chain of the demand. Each layer is a duplicate of the original graph and the capacities of both nodes and links are shared among layers. A path on the layered graph starts at layer 0 and ends at layer c_d and corresponds to an assignment of both a path and the locations where functions are being run (the links between layers).

1) Master Problem of `rescue-ILP` *and* `rescue-LP`*:*

Variables:

- $\varphi_p^{d,t} \in [0,1]$ is the amount of flow of demand d on path p at time step t.
- $y_p^{d,t} \in [0,1]$ is the maximum amount of flow of demand d on path p between time step $t-1$ and t.
- δ_l^p is the number of times the link ℓ appears on path p.
- $\theta_{i,u}^p$ = 1 if node u is used as a VNF on path p on layer i.

We assume an initial configuration is provided with fixed values for $\varphi_p^{d,0}$. The optimization model is written as follows.

Objective: minimize the amount of network resources consumed during the last reconfiguration time step T, which is the sum of the bandwidth used (BW) added to the sum of the costs of the deployed VNFs multiplied by a factor β.

$$\min \sum_{d\in D}\sum_{p\in P_d}\sum_{\ell\in E} \mathrm{BW}_d\, \varphi_p^{d,T}\, \delta_\ell^p + \beta \sum_{u\in V^{\mathrm{VNF}}}\sum_{f\in F} c_{u,f}\, z_{u,f} \quad (1)$$

Constraints:

One path constraint. For $d \in D$, time step $t \in \{0,...,T\}$.

$$\sum_{p\in P_d} \varphi_p^{d,t} = 1 \quad (2)$$

Path usage over two consecutive time periods. For $d \in D$, $p \in P_d$, $t \in \{1,...,T\}$.

$$\varphi_p^{d,t} \le y_p^{d,t} \text{ and } \varphi_p^{d,t} \le y_p^{d,t-1} \quad (3)$$

Make Before Break - Node capacity constraints. The capacity of a node u in V is shared between each layer and cannot exceed C_u considering the resources used over two consecutive time periods. For $u \in V^{\mathrm{VNF}}$, $t \in \{1,...,T\}$.

$$\sum_{d\in D}\sum_{p\in P_d}\sum_{i=0}^{|c_d|-1} y_p^{d,t}\cdot\theta_{i,u}^p\cdot \mathrm{BW}_d\cdot\Delta_{f_i^{c_d}} \le C_u \quad (4)$$

Make Before Break - Link capacity constraints. The capacity of a link $\ell \in E$ is shared between each layer and cannot exceed C_ℓ considering the resources used over two consecutive time periods. For $\ell \in E$, $t \in \{1,...,T\}$,

$$\sum_{d\in D}\sum_{p\in P_d} \mathrm{BW}_d\, y_p^{d,t}\, \delta_\ell^p \le C_\ell. \quad (5)$$

Function activation. To know which functions are activated on which nodes in the final routing. For $u \in V$, $f \in F$, $d \in D$, $i \in \{0,...,|c_d|-1\}$,

$$y_p^{d,T}\, \theta_{i,u}^p \le z_{u,f_i^{c_d}}. \quad (6)$$

2) ILP Pricing Problem of `rescue-ILP`*:* The pricing problem searches for a possible placement for the slice. Since a reconfiguration can be done in several steps, a pricing problem is launched for each demand, at each time step.

Parameters:

- μ are the dual values of the master's constraints. The number written in upperscript is the reference of the master's constraints.

Variables:

- $\varphi_{\ell,i} \in [0,1]$ is the amount of flow on link ℓ in layer i.
- $\alpha_{u,i} \in [0,1]$ is the amount of flow on node u in layer i.

Objective: minimize the amount of network resources consumed for the demand d at time t.

$$\min \sum_{\ell\in E}\sum_{i=0}^{|c_d|} \varphi_{\ell,i}\,\mathrm{BW}(1+\mu_{l,t}^{(5)}) + \mathrm{BW}\sum_{u\in V^{\mathrm{VNF}}} \mu_{u,t}^{(4)} \sum_{i=0}^{|c_d|-1} \Delta_{f_i^{c_d}}\,\alpha_{u,i} - \mu_{d,t}^{(2)} + \beta \sum_{u\in V^{\mathrm{VNF}}}\sum_{f\in F} c_{u,f}\, z_{u,f}\, \mu_{d,u,f}^{(6)} \quad (7)$$

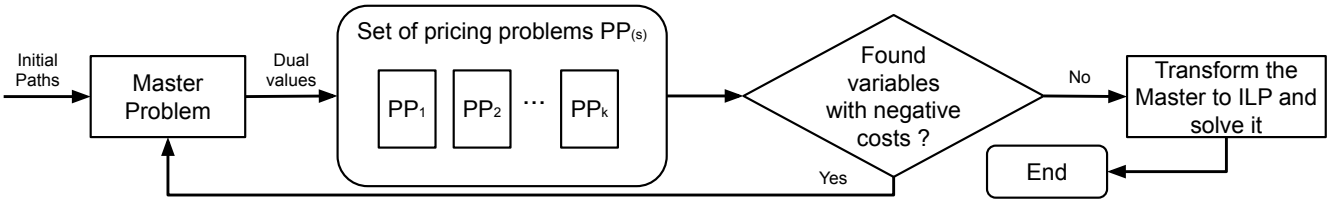


Fig. 2: CG is a decomposition method dividing an optimization model into two parts: a master problem and a (set of) pricing problem(s) (PP). The restricted master problem (RMP) solves a fractional relaxation of the problem with a restricted set of columns. Then the PPs compute the best columns to be added, based on prices given by the dual variables of the RMP. The RMP and PP are then iteratively solved until no more columns can improve the solution of the RMP. Last, the original problem is solved with the integrality constraint using the columns of the RMP.

where $\mu^{(6)}_{d,u,f} = 0$ when $t \neq T$, see constraints (6).

Constraints:

Flow conservation constraints for the demand d. For $u \in V^{\text{VNF}}$.

$$\sum_{\ell \in \omega^+(u)} \varphi_{\ell,0} - \sum_{\ell \in \omega^-(u)} \varphi_{\ell,0} + \alpha_{u,0} = \begin{cases} 1 \text{ if } u = v_s \\ 0 \text{ else} \end{cases} \quad (8)$$

$$\sum_{\ell \in \omega^+(u)} \varphi_{\ell,|c_d|} - \sum_{\ell \in \omega^-(u)} \varphi_{\ell,|c_d|} - \alpha_{u,|c_d|-1} = \begin{cases} -1 \text{ if } u = v_d \\ 0 \text{ else} \end{cases} \quad (9)$$

$$\sum_{\ell \in \omega^+(u)} \varphi_{\ell,i} - \sum_{\ell \in \omega^-(u)} \varphi_{\ell,i} + \alpha_{u,i-1} - \alpha_{u,i-1} = 0 \quad 0 < i < |c_d| \quad (10)$$

Delay constraints. The sum of the link delays of the flow must not exceed the delay requirement of demand d.

$$\sum_{i=0}^{|c_d|} \varphi_{\ell,i} \, \Gamma_\ell \leq \gamma_d \quad (11)$$

Function activation. To know which functions are activated on which nodes. For $u \in V^{\text{VNF}}$, $f \in F$, layer $i \in \{0, ..., |c_d| - 1\}$

$$\alpha_{u,i} \leq z_{u,f_i^{c_d}} \quad (12)$$

Location constraints. A node may be enabled to run only a subset of the virtual network functions. For $u \in V^{\text{VNF}}$, $i \in \{0, ..., |c_d|-1\}$, if the $(i+1)^{\text{th}}$ function of c_d cannot be installed on u, we have

$$\alpha_{u,i} = 0. \quad (13)$$

3) LP Pricing Problem of `rescue-LP`*:* The difference between `rescue-ILP` and `rescue-LP` comes from the pricing problem, which is integer for `rescue-ILP` and fractional for `rescue-LP`. Indeed, the execution time of the CG algorithm is divided into the resolutions of: (1) the multiple PPs, (2) the multiple relaxations of the RMP, and (3) the ILP of the MP. In our experiments, the time spent in (1) represents more than 90% of the whole execution time. To reduce this computational time, we propose `rescue-LP` that solves a relaxation of the pricing problem with fractional flows. The Master Problem of `rescue-LP` is the same as previously described. In the vast majority of cases, even with no constraint to force integral flows, the PP outputs an integral path that can be directly integrated into the RMP. If the LP gives a fractional flow, we use the ILP PP of `rescue-ILP` to get an integral path.

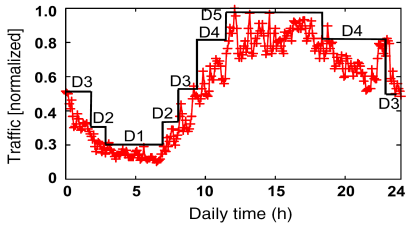


Fig. 3: Period approximation of traffic variation

Slice Types	VNF chain	Latency	BW (Mbps)
Video Streaming	NAT-FW-TM-VOC-IDPS	High	256
Web Service	NAT-FW-TM-WOC-IDPS	Medium	100
VoIP	NAT-FW-TM-FW-NAT	Low	64
Online Gaming	NAT-FW-VOC-WOC-IDPS	Very low	50

TABLE I: Characteristics of network slices

V. Numerical Results

A. Data sets

We conduct simulations on three real-world topologies from SNDlib [20] of different sizes: `pdh` (11 nodes, 34 links), `ta1` (24 nodes, 55 links), and `ta2` (65 nodes, 108 links).

We consider four different types of slices corresponding to four services: Video Streaming, Web Service, VoIP, and online gaming. The characteristics of each service are reported in Table I and are taken from [21]. They differ in terms of VNF chains, bandwidth usage, and latency requirement. Each slice has to implement a chain of 5 VNFs and requires a specific amount of bandwidth. The latency requirements are expressed in terms of maximum stretch, i.e., the ratio between the path delay compared to the shortest path between the source and destination. Simulations have been conducted on an Intel Xeon E3-1271 v3 with 32GB of RAM.

Our goal was to study the impact of reconfiguration for different network usages. Indeed, when the traffic is low or medium, all slices can be served and reconfigurations improve the network usage (links and VNFs). However, when the traffic is high and if some links are congested, reconfiguration also helps to prevent denying slices. To model the typical daily variation of traffic in an ISP network, we used the traffic distribution from a trace of the Orange network (Fig. 3). We adapted the churn rate of slices during time in order to obtain

a similar level of traffic. Each level of traffic corresponds to a different average number of slices: from 30 for D1, 68 for D2, 105 for D3, 158 for D4 to 180 for D5 for pdh.

We evaluate and compare 5 different algorithms:

- no-reconf places and removes the slices without reconfiguring the network,
- slice-wreck reconfigures regularly the network, but with interruptions. This algorithm gives a bound of the best we can reach with the make-before-break approach,
- slow-rescue: our compact ILP that reconfigures slices without interruptions,
- rescue-ILP: our CG based algorithm with ILP pricing,
- rescue-LP: our CG based algorithm with LP pricing.

B. Efficiency of our algorithms with different traffic matrices

In this section, we consider the pdh and ta1 networks for five different levels of traffic during the day, as shown in Fig. 3. All the slices of the traffic matrix are placed and routed, and one reconfiguration with two steps is achieved to reroute the slices in order to improve the network usage. First all the slices of $D1$ are placed one by one, and all reconfigured at once. Then, the same process is repeated for $D2$ until $D5$.

1) Execution times: We report the execution times of a reconfiguration in two steps for slow-rescue, rescue-ILP, and rescue-LP in Figure 4. Each value is an average over 10 experiments. We set a time limit of one hour. For pdh, slow-rescue finds the optimal solution only for the period D1. For all the other ones, it reaches the time limit. However, it succeeded to find a feasible solution (which was already efficient) for all the time periods, as can be seen in Fig 5. For the larger network ta1, the compact ILP was not able to find any feasible solution, even for D1 with few slices. On the contrary, the execution times of the column generation models are a low (below 120 s for both networks for any time period). Moreover, the models scale well as their execution times increase in a linear way. rescue-LP needs from 2 s to around 45 s, while the execution times of rescue-ILP are between 10 s and 120 s. Thus, as expected, rescue-LP is a lot faster than rescue-ILP.

2) Gains in network cost: We now compare in Figure 5 the improvement in terms of network cost obtained after a reconfiguration for each time period for pdh and ta1. For pdh, we observe that the results of rescue-ILP and rescue-LP are very close to the ones of slow-rescue: the difference is about 5% for D1 and less than 1% for the other periods. It shows that the column generation models achieve very good results, while being a lot faster than the ILP model.

For both networks, we see that rescue-ILP and rescue-LP achieve comparable results. As rescue-LP is faster, we use it as our *preferred solution* in the following. Last, we compare the results of our models with slice-wreck, which does not use the make-before-break mechanism. slice-wreck can achieve a better network improvement but at the cost of breaking slices and, thus, of a degraded QoS for users. We report its results as an upper bound on what our algorithms can achieve. We see that rescue-ILP and rescue-LP results are within few percent of the ones of slice-wreck, showing their efficiency. The difference is higher for heavy load periods (D4 and D5). Indeed, when the traffic is high, some links are almost saturated, it is harder to ensure that the bandwidth for both the current path and the path targeted by the reconfiguration can be reserved during the process.

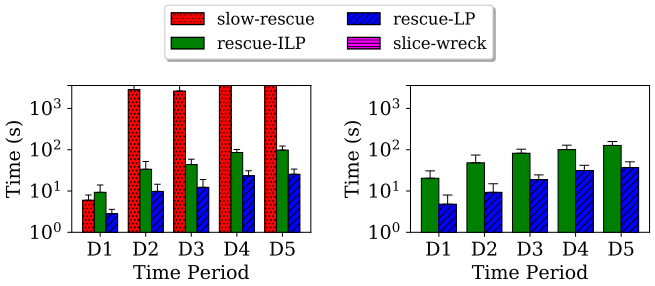


Fig. 4: Execution times for pdh (left) and for ta1 (right).

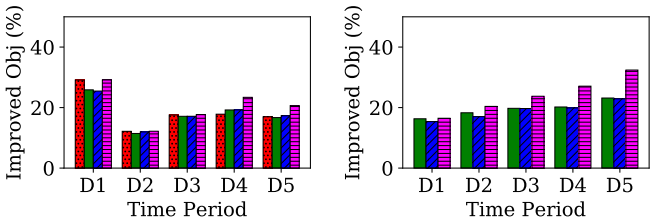


Fig. 5: Gains in network cost for pdh (left) and for ta1 (right).

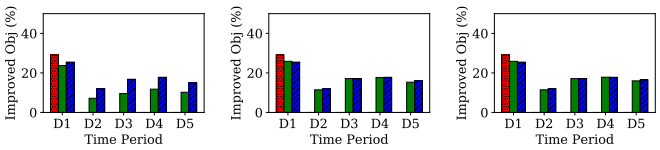


Fig. 6: Improvement due to the reconfiguration for different model time limits: 10 s (left), 60 s (middle) and 600 s (right) on pdh.

3) Time limits for the reconfiguration: The reconfiguration of the network has to be done dynamically in real time. In this context, the time to compute the reconfiguration is an important element towards the adoption of such solutions. We thus compare the results of the algorithms for pdh for different maximum execution times: 10, 60 and 600 seconds, see Figure 6. The first observation is that slow-rescue only gives a solution for D1, even with 600 s of execution. Secondly, rescue-ILP takes at least 60 s to reach its best value for any time period while rescue-LP reaches it in 10 s for all periods (except for D5 where it needs 60 seconds to reach it). It confirms that rescue-LP is the most scalable method while reaching similar performance as rescue-ILP. It thus is the best solution to use in practice.

C. Gains over Time

In the following, we are considering a scenario where reconfiguration is regularly performed, and where traffic is dynamic (requests arrive and leave during time). We now study the gains provided by the reconfiguration during time. To this

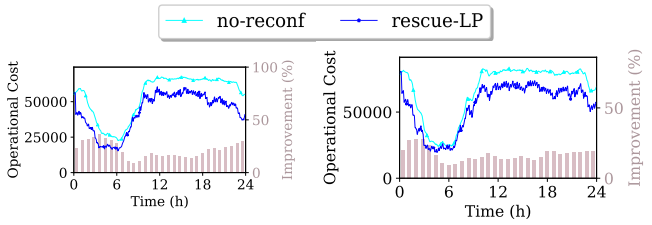


Fig. 7: Network cost for ta1 (left) and for ta2 (right).

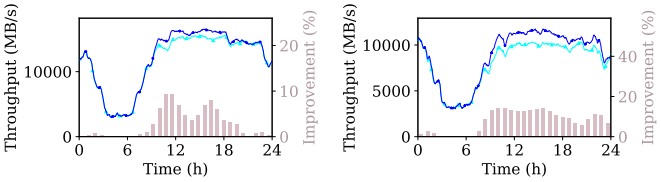


Fig. 8: Throughput for ta1 (left) and for ta2 (right).

end, we compare the results of rescue-LP with the ones of no-reconf which does not reconfigure the slices. We consider in this section two networks: the medium and large, ta1 and ta2. In our scenario, the network has periods of high congestion during which some slices may be rejected.We thus study the two metrics: the network operational cost and the throughput of the accepted slices. rescue-LP performs reconfigurations every 15 minutes. We choose this value as it seems a reasonable one for a network operator which does not want to change its routes too frequently.

1) Network Cost: In Figure 7 we study the network operational cost over time. The network cost follows the traffic variation depicted in Figure 3: the more traffic, the more network operational cost. Our solution is more reactive to traffic variations thanks to the reconfigurations that are regularly performed. Throughout the entire execution and for both networks, rescue-LP reduces significantly the network operational costs: 21% of reduction on ta1 and 18% on ta2 compared to no-reconf case. This reduction is particularly interesting when the network is loaded (between 10am and 6pm). This implies that the network is better managed and the resources are used more efficiently.

2) Throughput: The objective of our solution is to reduce operational costs. However, we should not reduce these costs at the price of rejecting slices. Therefore, we present the global throughput of the network in Figure 8. This throughput is defined as the sum of the requested bandwidth of the accepted slices. As we can see on both networks, ta1 and ta2, during the first 5 hours of execution there is almost no congestion because the traffic decreases. Therefore, no-reconf and rescue-LP accept the same number of slices, and therefore get roughly the same throughput. The next 3 hours, traffic increases and rescue-LP improves the throughput until almost 10% on ta2 when the network is the most saturated (traffic period D5). For a period of 24 hours, rescue-LP allows an average throughput improvement of 4% on ta1 and 7% on ta2. Therefore, as a conclusion on these two last figures, rescue-LP reduces network operational costs while at the same time improving the network throughput. These gains are reached without impacting users' Quality of service.

VI. Conclusion

In this work, we provide solutions, rescue-ILP and rescue-LP, to reconfigure a set of requests using a make-before-break approach. Our algorithms, based on column generation, reroute the requests to an optimal or close to optimal solution without impacting the rerouted requests. rescue-LP is the solution to be chosen in practice as we observed during simulations that it scales better with the network and the number of slices. Reconfiguring regularly the network with rescue-LP allows a slight increase in throughput when the network is congested as well as a significant reduction in operating costs of around 20%.

References

[1] M. Chiosi *et al.*, "Network functions virtualisation (NFV) network operator perspectives on industry progress," in *SDN and OpenFlow World Congress*, 2013.

[2] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, 2013.

[3] P. Rost *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications magazine*, 2017.

[4] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE INFOCOM*, 2017.

[5] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE TNSM*, 2016.

[6] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, 2016.

[7] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Transactions on Networking (TON)*, 2018.

[8] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *IEEE INFOCOM*, 2015.

[9] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, 2018.

[10] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," in *IEEE INFOCOM*, 2018.

[11] R. Wang and B. Mukherjee, "Provisioning in elastic optical networks with non-disruptive defragmentation," *IEEE Journal of Lightwave Technology*, 2013.

[12] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE TNSM*, 2017.

[13] K. A. Noghani, A. J. Kassler, and J. Taheri, "On the cost-optimality trade-off for service function chain reconfiguration," in *IEEE International Conference on Cloud Networking (CloudNet)*, 2019.

[14] L. Gao and G. N. Rouskas, "Virtual network reconfiguration with load balancing and migration cost considerations," in *IEEE INFOCOM*, 2018.

[15] G. Wang, G. Feng, T. Q. Quek, S. Qin, R. Wen, and W. Tan, "Reconfiguration in network slicing-optimizing the profit and performance," *IEEE TNSM*, 2019.

[16] A. Gausseran, A. Tomassilli, F. Giroire, and J. Moulierac, "Don't Interrupt Me When You Reconfigure my SFCs," in *IEEE International Conference on Cloud Networking (CloudNet)*, 2019.

[17] M. Leconte, G. Paschos, P. Mertikopoulos, and U. Kozat, "A resource allocation framework for network slicing," in *IEEE INFOCOM*, 2018.

[18] M. Pozza, A. Patel, A. Rao, H. Flinck, and S. Tarkoma, "Composing 5G network slices by co-locating VNFs in μslices," in *IFIP Networking Conference*, 2019.

[19] A. Gausseran, F. Giroire, B. Jaumard, and J. Moulierac, "Don't break network slices during reconfiguration," Inria, Tech. Rep., 2019.

[20] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0–survivable network design library," *Wiley Networks*, 2010.

[21] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *IEEE Conference NFV-SDN*, 2015.