Computer Communications 144 (2019) 175–187

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Clustered robust routing for traffic engineering in software-defined networks

Davide Sanvito [a,*], Ilario Filippini [a], Antonio Capone [a], Stefano Paris [b], Jérémie Leguay [b]

[a] Politecnico di Milano, Italy
[b] France Research Center, Huawei Technologies Co. Ltd, France

ARTICLE INFO

Keywords:
Adaptive traffic engineering
Traffic matrix clustering
Software defined networks
Robust routing
Intent monitor and reroute

ABSTRACT

One of the key advantages of Software-Defined Networks (SDN) is the opportunity to integrate Traffic Engineering modules able to optimize network configuration according to traffic. Ideally, the network should be dynamically reconfigured as traffic evolves, so as to achieve remarkable gains in the efficient use of resources with respect to traditional static approaches. Unfortunately, reconfigurations cannot be too frequent due to a number of reasons related to route stability, forwarding rules instantiation, individual flows dynamics, traffic monitoring overhead, etc.

In this paper, we focus on the fundamental problem of deciding whether, when and how to reconfigure the network during traffic evolution. We propose a new approach to cluster relevant points in the multi-dimensional traffic space taking into account similarities in multiple domains and not only in traffic values. Moreover, to provide more flexibility to the decisions on when to apply a reconfiguration, we allow some overlap between clusters that can guarantee a good-quality routing even in case of smooth transitions.

We compare our algorithm with state-of-the-art approaches in realistic network scenarios. Results show that our method significantly reduces the number of reconfigurations with a negligible deviation of the network performance with respect to the continuous update of the network configuration.

Moreover, we present an experimental platform where our solution is implemented in a production-ready SDN controller.

1. Introduction

Traffic Engineering (TE) [1] plays a crucial role for service providers since it permits to optimize network performance, reduce operational costs, and load balance the utilization of network resources. However, the dynamic nature of the traffic due to ordinary daily fluctuations and unpredictable events stirs up the trade-off between optimality of the routing configuration and network reconfiguration rate. The traditional approach of service providers has been to design the routing considering the "worst case" traffic condition, thus privileging rare network reconfigurations. The resulting overprovisioning inevitably leads to a suboptimal underutilization of network resources.

Software-Defined Networks (SDNs) [2] provide the needed flexibility to update more frequently TE policies to pursue optimality. Having a global view of the network status, SDN controllers can integrate TE algorithms [3–5] to continuously optimize the network with an online twist. As the system evolves, new configurations are applied to the network equipment to optimize network performance according to traffic variations. The solutions that have been devised to cope with these traffic variations can be broadly classified into three main classes: *dynamic TE*, *static TE*, and *semi-static TE*. While different in the way they calculate network configurations, all techniques require the use of Traffic Matrices (TMs) periodically collected by a network monitoring tool.

The example in Fig. 1, where a TM composed of only two demands is considered, shows the main differences of the two extreme approaches, namely *dynamic TE* and *static TE*, which require, respectively, to always and never reconfigure the network every time the traffic fluctuates.

Dynamic TE, like [3,4,6,7], uses past TMs to predict the next system state and compute the corresponding optimal routing configuration using linear programming [8] or fast approximation algorithms [9]. As illustrated in Fig. 1, the accuracy of the prediction highly affects the optimality of the computed solution, since prediction errors can lead to congestion or infeasible configurations. This inevitably results in suboptimal solutions for the actual traffic realization. Furthermore, frequent network reconfigurations due to the evolution of the traffic and corresponding routing configuration result in control plane congestion due to the low speed of flow programming [4] in hardware. Although

* Corresponding author.
E-mail addresses: davide.sanvito@polimi.it (D. Sanvito), ilario.filippini@polimi.it (I. Filippini), antonio.capone@polimi.it (A. Capone), stefano.paris@huawei.com (S. Paris), jeremie.leguay@huawei.com (J. Leguay).

https://doi.org/10.1016/j.comcom.2019.06.002
Received 19 November 2018; Received in revised form 20 April 2019; Accepted 1 June 2019
Available online 4 June 2019
0140-3664/

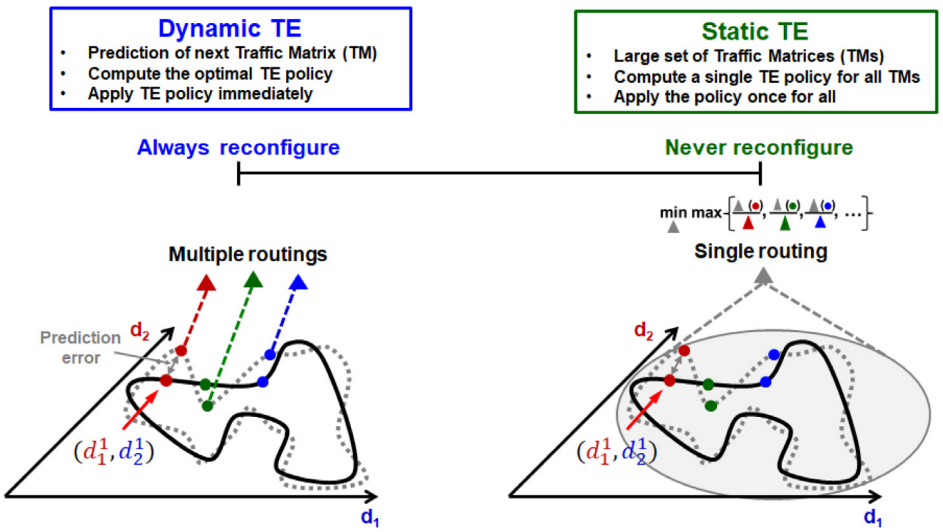

Fig. 1. Dynamic TE vs. Static TE. The traffic matrix (TM) consists of only two demands. Dynamic TE computes a routing configuration for each sampled/predicted TM, whereas Static TE computes a single solution for all sampled/predicted TMs. Points on the solid black line represent the real TMs, while gray dotted line contains the predicted TMs.

several methods have been proposed to reduce this burden by prioritizing [10] or pre-filtering [11] network updates, they cannot overcome the intrinsic limit of *Dynamic TE*: the computation of each routing solution does not take into account any robustness consideration.

In contrast, *static TE*, such as *oblivious routing* [12] and *robust routing* [13–15], monitors TMs over a long period of time and computes the TE configuration that minimizes the worst deviation with respect to the sequence of all per-TM optimal configurations. Such a deviation, which is called performance ratio, is defined as the ratio between the objective function of the static TE configuration applied to a TM and its optimal solution as illustrated in Fig. 1. It represents the loss due to a suboptimal configuration. The class of *static TE* policies keeps the network configuration stable, but it inevitably suffers from low optimality during most of the operational time.

Semi-static TE approaches such as [16,17] combine both static and dynamic TE to approximate the optimal sequence of configurations with a limited set of routing solutions computed over clusters of TMs. Clusters of TMs are formed either by statically dividing time in different intervals or by finding similarities in the traffic domain. However, the arbitrary splitting of the time domain results in significant performance loss when sharp traffic variations are temporally close. Similarly, using the same routing configuration for TMs that are close in the traffic domain (i.e., their entries have the same magnitude) but far in the time domain can lead to frequent network reconfigurations. In addition, TMs different in the traffic domain do not necessarily have a different optimal routing configuration. A crucial aspect has been neglected so far: the controller needs to decide whether and when to reconfigure. Transitory traffic fluctuations should be ignored to avoid system oscillations and the network should be reconfigured only when it is evolving towards a new state.

In this paper, we study the fundamental problem faced by SDN controllers of deciding whether, when and how to reconfigure the network after a traffic evolution. To provide an answer we study and address the problem of building a set of robust routing configurations associated to clusters of TMs that overlap in time, traffic and routing domains. Time overlap refers to the amount of time we can use a routing configuration even for TMs that are just before/after the associated cluster with minimal efficiency degradation. Traffic overlap denotes the similarity in the traffic space of TMs within the same cluster, whereas routing overlap indicates how similarly perform two routing configurations associated to two different clusters.

Given the interplay between TM clusters and routing solutions, we decouple the problem into two subproblems, namely TM clustering and robust routing. To this aim, we propose Clustered Robust Routing (CRR), an iterative algorithm that achieves three objectives: (1) covering the entire TM space so that a feasible routing configuration is available for any traffic condition, (2) reducing the number of routing changes by creating a small set of robust routing configurations that can be used for a minimum duration each time one of them is applied, and (3) maintaining a minimum time overlap between adjacent clusters that can be exploited to decide whether to reconfigure the network. These three objectives map to four properties of the optimization model that will be discussed more in depth at the beginning of Section 4.

This work extends a previous conference paper [18] by introducing the optimization model which jointly computes the members of the clusters together with the corresponding robust routing configurations and extending the evaluation of the effects of overlapped clusters. We analyze our algorithm on a realistic network scenario and compare its performance against state of the art approaches of the three TE classes discussed above. Finally, we integrate our algorithm in a production-ready SDN controller and evaluate it in an experimental testbed.

This paper is structured as follows: Section 2 presents the related work. Section 3 describes the system model and the assumptions we made in the formulation of our problem. Section 4 presents the algorithm to build clustered robust configurations considering the time continuum, the traffic space and routing similarities. Numerical results are discussed in Section 6. An experimental testbed is described in Section 7. Finally, concluding remarks are presented in Section 8.

2. Related work

The simplicity of controlling SDNs has brought back to light mitigation and scheduling network reconfiguration [19,20] problems, motivated by service providers' concern about possible network outages caused by failures of route updates or sudden traffic changes. The networking research community has developed three classes of techniques to handle traffic change: (i) *dynamic TE*, which reconfigures the network each time a new event occurs, (ii) *static TE*, which uses a single precomputed configuration that minimizes the worst deviation to the optimum, and (iii) *semi-static TE*, which reconfigures the network at predefined time instants (e.g., twice per day at noon and midnight) to further improve network performance. A comparative analysis of the performance of these classes of routing policies has been recently carried out over a SDN prototype in [21].

Examples of *dynamic TE* includes methods like [3,4,6,7], where sophisticated techniques are used to compute the best network configuration any time the traffic changes. However, reconfiguring the network too frequently can affect its stability, since programming hardware equipment with new flow rules can take longer than the reconfiguration period [4], thus causing the overflow of flow rules. Methods that reduce this burden have been proposed by prioritizing [10] or pre-filtering [11] network updates. Nonetheless, the computation of each routing solution is not robust against prediction errors on the next TM.

One of the first techniques of *static TE* is oblivious routing [12,22], and its recent extension called *valiant routing* [23], which randomly selects paths to connect source–destination pairs using a small subset of preselected intermediate nodes. Being totally oblivious to any traffic information, oblivious routing shows high performance loss as the network size grows. Exploring a partial knowledge of the traffic can reduce the performance loss. For example, COPE [15] considers only the most likely TMs for computing the optimal configuration and add a penalty term to avoid large deviation for unlikely TMs. The technique proposed in [24] expands the most likely polytope by including TMs of normal operations in the direction of a predicted anomaly. The method proposed in [13] introduces different models for traffic uncertainness by expressing the maximum load that can be expected over a link in the pipe model or an upper bound on the traffic originating from a source node and directed to a destination node in the hose model.

Semi-static TE [16,17,25,26] provides a limited set of routing configurations with guaranteed performance loss. These works divide the TM polytope in two subsets according to the time dimension and compute a robust routing for each subset. While representing a first attempt to split the TM domain in multiple parts, these works present several limitations: (i) the slicing direction is arbitrary, (ii) the number of created subsets is limited, and (iii) the partition is performed either in the traffic domain or in the time domain. Other *Semi-static TE* approaches, like affine and multi-polar routing [27,28], have been more recently proposed to overcome these limitations. In particular, these techniques compute a set of routing configurations that can be easily combined together to generate a routing configuration for a new traffic realization. However, the combination of multiple configurations can results in the utilization of a large number of paths and a flow split ratio that might not be handled by network devices.

Although semi-static TE approaches have the potential to optimize network performance using a limited set of routing configurations, traffic, time, and routing spaces/dimensions should be jointly considered when building clusters in order to avoid oscillations between routing configurations when TMs are close in the traffic space but far in the time dimension. Furthermore, clusters should not be sharply separated, since instantaneous routing changes are impossible even in SDNs. This work is the first attempt to address these problems and decide the best trade-off between reconfiguration rate and routing optimality.

3. System model

In this section, we present the traffic and routing system models that we consider in the design of our CRR algorithm.

We consider a system composed of two main stages: (i) a cluster-maintenance stage where we group TMs into clusters and compute robust routing configurations over these clusters, and (ii) a cluster-activation stage where we track the traffic evolution and reconfigure the network accordingly. The target is to minimize the average Maximum Link Utilization (MLU) over time, which is motivated in the domain of datacenter interconnection and enterprise networks, where the goal is to minimize the network congestion.

We model the network infrastructure as an undirected graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents the set of network nodes and $\mathcal{L}$ models the set of links $e = (i, j)$, connecting network nodes $i, j \in \mathcal{N}$. Each link $e \in \mathcal{L}$ has a limited capacity c_{ij} that represents the maximum amount of traffic that the link can transmit. The set of active demands, also known as Origin–Destination (OD) flows, that need to be routed through the network, is represented as a Traffic Matrix (TM): a $|\mathcal{N}|\times|\mathcal{N}|$ matrix $T = [t_{ij}]$ where each element t_{ij} denotes the amount of traffic transmitted from source node i to destination node j. Since the traffic evolves over time, we consider a dynamic TM $T(\tau) = [t_{ij}^{\tau}]$, where τ denotes the time dimension. We assume that time is discretized and we have M samples of the TM (i.e., $\tau = 1, \ldots, M$).[1] To simplify the notation, TM are usually represented as a $|\mathcal{N}|^2 \times 1$ demand vector $D(\tau) = [d_h^{\tau}]$ where each element d_h^{τ} unequivocally corresponds to an element t_{ij}^{τ} of the TM and $\mathcal{H}$ represents the set of demands.

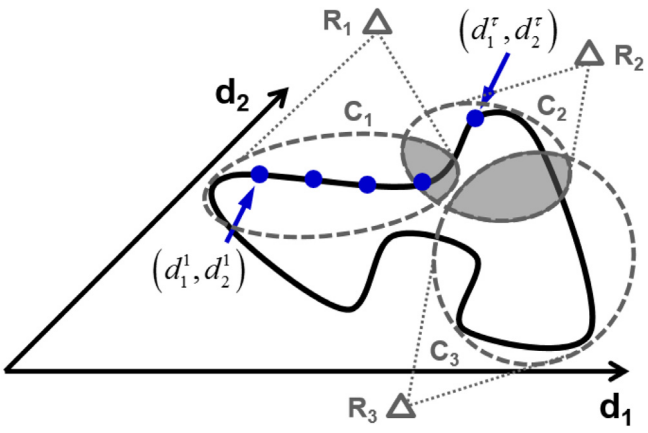

Fig. 2. Clustering of TMs. The solid black line represents the evolution of the two OD flows in the TM. Each blue point represents a sampled TM. Dashed ellipsoids denote the clusters of TMs, whereas triangles identify the corresponding routing configurations.

Cluster maintenance. Fig. 2 graphically illustrates the time evolution of a TM composed of only two demands, d_1 and d_2. The solid line represents the continuous evolution of the TM, whereas solid dots corresponds to periodic samples measured by a traffic monitoring system. As illustrated in the figure, an offline stage splits the TM domain into N clusters, denoted as C_i, and computes for each subset of TMs a routing configuration R_i, which is robust against any possible traffic variation within the cluster C_i: the routing is not customized on a specific TM, but has to be robust to cope with demands uncertainty (due to imperfect measurements/predictions or traffic anomalies). To avoid oscillations between routing configurations, a cluster C_i is built with a minimum time length L that results in a minimum utilization of the same routing configuration R_i. Furthermore, a temporal overlap O (the gray intersection in Fig. 2) is imposed between two adjacent clusters C_i and C_j to guarantee the feasibility of the corresponding robust routing configurations R_i and R_j outside their clusters. The overlap O provides further robustness against inaccurate cluster transition and can potentially leave some time to the real-time SDN controller decision on whether to reconfigure the network.

Cluster activation. The different precomputed routing configurations are then activated by the SDN controller, which follows the evolution of the traffic matrix. By receiving an estimate of actual traffic conditions (and possibly a short term prediction) from the monitoring system, it can decide whether to fetch and activate a better robust routing configuration in switches.

Clearly, the performance of this approach depends on the size and the number of clusters. Many small clusters result in a high reconfiguration rate, which may harm the network behavior itself. In contrast, too few clusters will provide a small gain over static-TE solutions (e.g., oblivious routing). This clustering approach will always lead to a better performance than the single-route case, as the network is no

[1] In the rest of the paper we will interchangeably refer to TM $T(\tau)$ at time τ with the time instant τ itself.

Table 1
Input parameters of our algorithms.

Parameter	Description
$\mathcal{N}$	Set of nodes (network devices)
$\mathcal{L}$	Set of edges (network links)
$\mathcal{H}$	Set of demands
$\mathcal{T}$	Set of time instants
$\mathcal{R}$	Set of routing configurations
c_{ij}	Edge capacity (in capacity units) $(i,j) \in \mathcal{L}$
d_h^τ	Rate of OD demand $h \in \mathcal{H}$ measured at time $\tau \in \mathcal{T}$
N	Number of robust routing configurations
M	Number of traffic matrices
L	Minimum holding time of a routing configuration
O	Temporal overlap between two adjacent clusters

Table 2
Variables of **CRR** model.

Variable	Description
$\gamma_\tau^r \in \mathbb{R}$	Network MLU when rth RC is applied to TM $T(\tau)$
$f_{ij}^{hr} \in \mathbb{R}$	Amount of demand $h \in \mathcal{H}$ routed on link $(i,j) \in \mathcal{L}$ under rth RC
$x_\tau^r \in \mathbb{B}$	TM $T(\tau)$ to rth RC assignment
$y_\tau^r \in \mathbb{B}$	Forward difference of x_τ^r variable
$w_\tau^r \in \mathbb{B}$	Backward difference of x_τ^r variable
$\gamma_\tau^{-,r} \in \mathbb{R}$	Network MLU when a RC is applied to TMs before the beginning of the cluster
$\gamma_\tau^{+,r} \in \mathbb{R}$	Network MLU when a RC is applied to TMs after the end of the cluster

Table 3
Variables of **SP** model.

Variable	Description
$x_\tau^r \in \mathbb{B}$	TM $T(\tau)$ to rth RC assignment
$y_\tau^r \in \mathbb{B}$	Forward difference of x_τ^r variable
$z^r \in \mathbb{B}$	rth RC selection
$w_\tau^r \in \mathbb{B}$	Backward difference of x_τ^r variable

Table 4
Variables of **RR** model.

Variable	Description
$\gamma_\tau \in \mathbb{R}$	Network MLU when routing TM $T(\tau)$
$f_{ij}^{hr} \in \mathbb{R}$	Amount of demand $h \in \mathcal{H}$ routed on link $(i,j) \in \mathcal{L}$ under rth RC

longer forced to always support the worst-case traffic demand. Indeed, the correct solution will be applied when the corresponding worst-case (among the possibly many that lie in different regions) appears.

For a smooth network reconfiguration when the TM evolution enters into a new cluster, we compute clusters with a minimum time overlap, represented by the intersection of two clusters in Fig. 2. This allows us to mitigate the effect of a sharp boundary between time contiguous clusters, which requires an instantaneous reconfiguration of the network.

In the next section, we show how our algorithm, Clustered Robust Routing (CRR), solves the problem of achieving a good trade-off between routing stability and optimality by maintaining a set of routing configurations for overlapping clusters of TMs.

Tables 1–4 summarize the notation used throughout the paper. The first column of Tables 2–4 also reports the domain of the decision variables: $\mathbb{R}$ and $\mathbb{B}$ define a variable with real and binary values, respectively ($\mathbb{B} = \{0,1\}$).

4. Clustered robust routing

The CRR algorithm is implemented as a module of the network controller. It takes as input a set of TMs representative of the period in which robust routing configurations have to be designed. These TMs can be obtained in several ways: they can be measurements from past network conditions, or the outcome of a TM prediction module, or even synthetically generated. For the sake of clarity, we neglect the effect of prediction errors in the description of the algorithm, however, we investigate the impact of inaccurate TMs within numerical results. The impact is indeed rather limited for realistic error values because the clustering intrinsically generates robust solutions.

Each TM describes the expected traffic conditions at specific time instants. Therefore, TMs can be temporally ordered and the set of TM IDs can be used as time axis. The result of the algorithm is a set of Routing Configurations (RCs, denoted as R_i in Fig. 2) and the corresponding clusters of TMs (C_i in Fig. 2). Each RC will be activated in the network as soon as the traffic enters the corresponding cluster. We have designed the CRR algorithm relying on the following main features:

- ($F1$) *Routing-based clustering*: TMs are grouped together not only in time and traffic domains, but also in the routing performance domain. Considering only similarities between the OD demands might be inefficient because TMs with similar demands can have very different good quality routings. In order to take into consideration the routing, we need to consider its ultimate effect, i.e. the network congestion resulting from applying a given RC to a given TM. Considering just similarities in the routing variables given by the optimum routings can misuse the number of RCs (a resource we do not want to waste) because, due to network scenario symmetries, two RCs can provide a comparable level of congestion despite having different routing variables. Only TMs characterized by a small congestion with a same RC must be grouped together in a cluster;
- ($F2$) *In-cluster robust routing*: due to imperfect measurements or predictions, the actual TMs we need to deal with might deviate from the TM input set. The RC thus cannot be strongly customized on a specific TM of the cluster and we need a routing solution which is robust against uncertainty in the traffic matrix subspace defined by the members of the cluster. One possibility is to consider the worst case TM in the continuous set defined by the convex hull of all the discrete TMs. Since the optimization process is quite complicated [22,29] and the considered TM might be potentially rare in practice, we rely on the discrete space described by the set of representative TMs of each cluster and adopt a multi-TM robust optimization approach, like those in [30]. The decoupled nature of the CRR allows to easily change the routing computation building block with other approaches to robust routing from literature. If, for example, significant anomalies come into play, we can adopt the approach presented in [15] which, despite still focusing on the set of representative TMs, guarantees a bounded penalty gap over the remaining traffic scenarios without changing the TM clustering logic;
- ($F3$) *Routing configuration holding time*: a same RC has to be kept for a guaranteed minimum time. Even if SDN allows to dynamically change network routes, we cannot change routes too frequently, to avoid incurring into route flapping problems. Assuming the TM input set comes from an uniform sampling, this requirement translates into enforcing a minimum size for each cluster;
- ($F4$) *Adjacent clusters overlap*: transitions between two RCs are not instantaneous even in SDN. By considering an overlap among adjacent clusters we can make the transition smoother. This means that routing configurations of adjacent clusters will be reasonably good also with the TMs that are expected to be close to a route transition. This gives more time to the SDN controller to decide when the update should occur, enabling a switching point decision based on the past, current and predicted traffic, considering anticipatory networking aspect as well. Overlaps also help consistent update techniques [31,32] that need multiple steps to guarantee a correct and congestion-free update.

4.1. CRR problem formulation

The problem we want to solve is to find the best assignment of M TMs to N robust RCs in order to find N clusters of contiguous TMs having a minimum length of L (feature *F3*) and an overlap of O TMs (feature *F4*). The TMs grouped into a cluster will be routed with the RC associated to the cluster. Moreover, since TM IDs are temporally ordered, the solution also provides the best cluster transition instants, thus routing reconfiguration points. Note that the TMs of a cluster are required as input of the intra-cluster robust routing (feature *F2*), which, in turn, is required to drive the TM clustering, through the estimated congestion using that robust routing (feature *F1*). Therefore, this two aspects must be jointly addressed to obtain an optimal solution. For the sake of simplicity, we temporarily neglect the requirement of having an overlap O. The problem is described by the following MINLP model:

$$[\mathbf{CRR}]: \min. \sum_{\tau\in\mathcal{T}, r\in\mathcal{R}} \gamma_\tau^r \qquad \text{s.t.:} \tag{1}$$

$$\sum_{(i,j)\in\mathcal{L}} f_{ij}^{hr} - \sum_{(j,i)\in\mathcal{L}} f_{ji}^{hr} = \begin{cases} 1 & \text{if } i = \text{origin}(h) \\ -1 & \text{if } i = \text{destination}(h) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i \in \mathcal{N}, h \in \mathcal{H}, \forall r \in \mathcal{R} \tag{2}$$

$$\gamma_\tau^r \geq \frac{\sum_{h\in\mathcal{H}} d_h^\tau f_{ij}^{hr}}{c_{ij}} \cdot x_\tau^r \qquad \forall \tau \in \mathcal{T}, \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{L} \tag{3}$$

$$y_\tau^r \geq x_{(\tau+1)_{|\mathcal{T}|}}^r - x_\tau^r \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{4}$$

$$\sum_{\tau\in\mathcal{T}} y_\tau^r \leq 1 \qquad \forall r \in \mathcal{R} \tag{5}$$

$$\sum_{r\in\mathcal{R}} x_\tau^r = 1 \qquad \forall \tau \in \mathcal{T} \tag{6}$$

$$\sum_{\tau\in\mathcal{T}} x_\tau^r \geq L \qquad \forall r \in \mathcal{R} \tag{7}$$

$$x_\tau^r, y_\tau^r \in \{0,1\} \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{8}$$

$$0 \leq \gamma_\tau^r \leq 1 \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{9}$$

$$0 \leq f_{ij}^{hr} \leq 1 \qquad \forall (i,j) \in \mathcal{L}, \forall h \in \mathcal{H}, r \in \mathcal{R} \tag{10}$$

The Clustered Robust Routing (CRR) takes in input a set of TMs $\mathcal{T} = \{T(1), \ldots, T(M)\}$, the network graph $G = (\mathcal{N}, \mathcal{L})$, and the index set $\mathcal{R} = \{1, \ldots, N\}$, which identifies the number of possible clusters/routing configurations. The TMs in $\mathcal{T}$ are characterized by the same demand set $\mathcal{H}$, but different demand values, d_h^τ, varying according to the traffic time evolution.

The goal of **CRR** model is to minimize the average Maximum Link Utilization (MLU)[2] over time. Variables γ_τ^r store the MLU experienced in the network when the rth RC is applied to TM $T(\tau)$. Routing variables f_{ij}^{hr} express the percentage of demand h routed over link (i,j) with the rth RC. We consider here a more general splittable routing because it can be easily implemented in SDN switches, however the model can be easily modified to consider unsplittable routing solutions as well by defining f_{ij}^{hr} in constraints (10) as binary variables.

The set of constraints (2) ensures flow balance is guaranteed for each r of the N computed RCs, thus defining each RC, while constraints (3) ensure that each γ_τ^r is set to the maximum utilization among network links when TM $T(\tau)$ is routed via the rth RC. Indeed, variables x_τ^r are the clustering variables, and define the assignment of TM $T(\tau)$ to the rth RC.

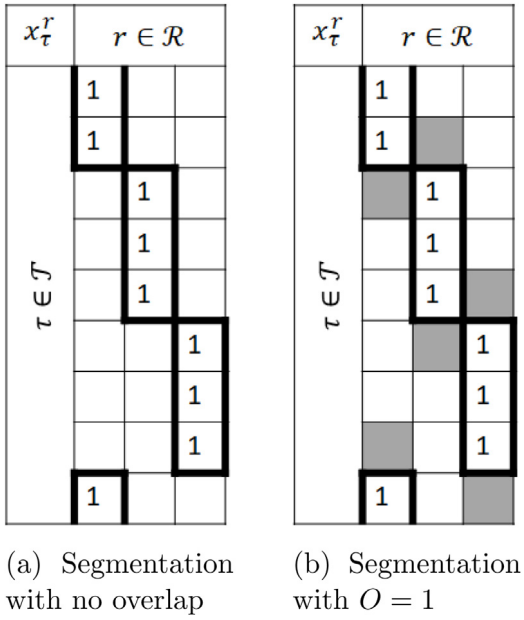

Fig. 3. Segmentation Problem, x_τ^r matrix.

If we visualize the matrix corresponding to variables x_τ^r (see Fig. 3a), the solution of the problem is a set of N contiguous column-sequences of 1's with minimum length L. These sequences must be unique in a column, and correspond to a set of TMs forming a cluster. In order to identify the beginning of each cluster, we rely on variables y_τ^r, defined as forward differences of variables x_τ^r. With $y_\tau^r = 1$, the model identifies the initial TM of the cluster r: $T(\tau+1)$.

The set of constraints (4)[3] force variables y_τ^r to be 1 whenever the forward differences of variables x_τ^r are 1. Constraints (5) guarantee a unique cluster beginning for each column.[4] The set of constraints (6) imposes each TM $T(\tau)$ to be assigned to one and only one cluster. Constraints (7) force a minimum number of TMs assigned to a cluster r, which, combined with the previous constraints, imposes a unique and compact sequence of 1's of minimum length L. Finally, constraint (9), together with (3), ensures that link capacities are not exceeded, when considering the most congested links for every TM $T(\tau)$ under the rth RC. This implicitly guarantees the same for all network links.

The solution of the model defines the optimal network configuration: variables x_τ^r define the TMs of each cluster r, whose RC is represented by variables f_{ij}^{hr}. Finally, variables y_τ^r provide the routing activation time, $\tau+1$, of each rth RC.

In order to consider an overlap between adjacent clusters, formulation **CRR** must be amended to capture the fact that O TMs beyond the boundaries of the clusters could be routed with the RC associated to the cluster. We model this by stating that each of the O TMs in overlap (gray cells in Fig. 3b) provides a MLU cost that is the average between the one of the assigned cluster and the one using the RC of the overlapping cluster.[5] The following constraints must be added:

$$w_\tau^r \geq x_{(\tau-1)_{|\mathcal{T}|}}^r - x_\tau^r \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{11}$$

$$\sum_{\tau\in\mathcal{T}} w_\tau^r \leq 1 \qquad \forall r \in \mathcal{R} \tag{12}$$

[2] Note that we decided to use the MLU as it is a commonly-used metric that directly expresses the network congestion, however the model is general enough to consider other types of metrics related to traffic. We can for example minimize the time-averaged Average Link Utilization (ALU), delay or routing cost. This can be captured by changing the objective function and the constraints (3).

[3] The notation $(\cdot)_m$ indicates the modulo-m operator.

[4] Note that clustering can be made non-circular (i.e. with the first cluster starting exactly at the beginning of the day, without spanning the end) by simply adding $\sum_{r\in R} y_{|\mathcal{T}|}^r = 1$ and $\sum_{r\in R} w_1^r = 1$ to the set of constraints.

[5] The model can capture other assumptions by simply changing some of the coefficients in the formulation.

$$\gamma_k^{-,r} \geq \frac{\sum_{h\in\mathcal{H}} d_h^k f_{ij}^{hr}}{c_{ij}} \cdot y_\tau^r$$
$$\forall \tau \in \mathcal{T}, \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{L}, \forall k : (\tau - O < k \leq \tau)_{|\mathcal{T}|} \quad (13)$$

$$\gamma_k^{+,r} \geq \frac{\sum_{h\in\mathcal{H}} d_h^k f_{ij}^{hr}}{c_{ij}} \cdot w_\tau^r$$
$$\forall \tau \in \mathcal{T}, \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{L}, \forall k : (\tau \leq k < \tau + O)_{|\mathcal{T}|} \quad (14)$$

$$w_\tau^r \in \{0,1\} \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \quad (15)$$

$$0 \leq \gamma_\tau^{-,r} \leq 1 \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \quad (16)$$

$$0 \leq \gamma_\tau^{+,r} \leq 1 \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \quad (17)$$

and a new objective function must be introduced in **CRR**:

$$\min \sum_{\tau\in\mathcal{T}, r\in\mathcal{R}} x_\tau^r \gamma_\tau^r + \frac{1}{2} \sum_{\tau\in\mathcal{T}, r\in\mathcal{R}} y_\tau^r \left(\sum_{(\tau-O<k\leq\tau)_{|\mathcal{T}|}} \gamma_k^{-,r} - \sum_{(\tau+1\leq k\leq\tau+O)_{|\mathcal{T}|}} \gamma_k^r \right) + \frac{1}{2} \sum_{\tau\in\mathcal{T}, r\in\mathcal{R}} w_\tau^r \left(\sum_{(\tau\leq k<\tau+O)_{|\mathcal{T}|}} \gamma_k^{+,r} - \sum_{(\tau-O\leq k<\tau)_{|\mathcal{T}|}} \gamma_k^r \right) \quad (18)$$

Constraints (11) define variables w_τ^r as backward differences of x_τ^r and, similarly to constraints (4)–(5), guarantee a unique cluster for each column, when combined with constraints (12). Defining the end of the compact column-sequences of 1's in Fig. 3, w_τ^r must satisfy the same uniqueness requirements as y_τ^r.

We introduce two additional variables $\gamma_\tau^{-,r}$ and $\gamma_\tau^{+,r}$, similar to γ_τ^r in (13)–(14), to compute the MLU caused by the application of cluster r's RC to O TMs, respectively, before the beginning and after the end of cluster r. They are used to consider congestion contribution of adjacent clusters and ensure, thanks to constraints (16)–(17), that the RC of overlapping clusters is feasible for the TMs in the overlap region as well.

The new objective function (18) computes the MLU of TMs in overlap by removing half of the MLU cost with the assigned cluster's RC and adding half of the MLU cost resulting from the application of the overlapping cluster's RC. For sake of completeness, the linear version of the full model with overlap is detailed in Appendix.

Unfortunately, the problem described by **CRR** model is strongly combinatorial and it has revealed to be very hard to solve. State-of-the-art integer programming solvers, like Gurobi Solver[6] or IBM CPLEX,[7] could not provide a solution in reasonable times: small instances of tens of TMs require several days to get the optimum.

5. Two-steps CRR heuristic

The hardness of the joint problem calls for the development of a heuristic approach to split the overall complexity in more affordable subproblems. In this perspective, we propose the two-steps Clustered Robust Routing (CRR) algorithm, whose scheme is drawn in Fig. 4. In the first step, a *Segmentation Problem* is solved: provided an initial set of W RCs, the best assignment of M TMs to up to N of the given RCs is computed, considering the minimum holding time constraint and the overlap. In the second step, a *Robust Routing Problem* is computed for each of the N clusters, in order to create new RCs more suitable for the selected TMs. The new RCs are introduced in the set of W available RCs of the Segmentation Problem and the two steps are repeated for a given number of iterations.

[6] www.gurobi.com.
[7] www.ibm.com/software/commerce/optimization/cplex-optimizer/.

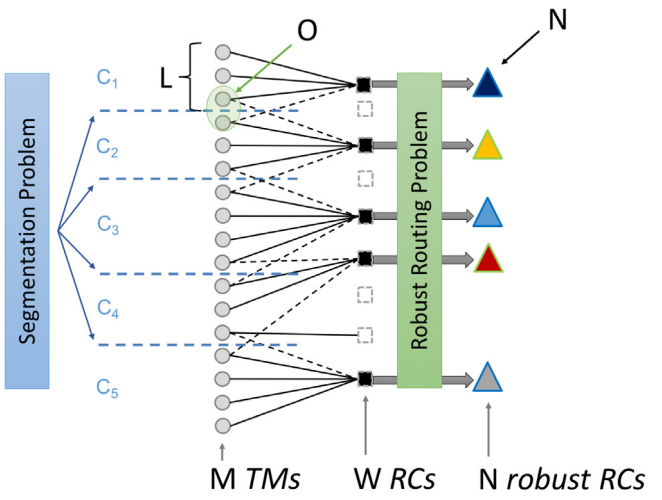

Fig. 4. High-level view of the proposed two-steps CRR algorithm.

5.1. STEP 1 - Segmentation Problem

The Segmentation Problem takes in input a set of TMs $\mathcal{T} = \{T(1), \ldots, T(M)\}$ and a set of W RCs $\mathcal{R} = \{R_1, \ldots, R_W\}$. Its goal is to assign each TM $T(\tau)$ to a RC R_r such that the overall association cost is minimized and the number of used RCs is not larger than N. The TM–RC association cost δ_τ^r can be precomputed and corresponds to the network Maximum Link Utilization (MLU) when TM $T(\tau)$ is routed through RC R_r, which is given. The solution of the Segmentation Problem provides a set of N TM clusters and RCs satisfying minimum length and overlap constraints.

We model the Segmentation Problem as an ILP model that considers a subset of variables and constraints of **CRR** model presented in the previous section, where variables γ_τ^r have been replaced by input parameters δ_τ^r. Indeed, since the W RCs are precomputed, and thus input parameters, constraints (2), (3), (9), (10), and variables f_{ij}^{hr} and γ_τ^r must not be included. Again, for the sake of simplicity we temporarily neglect the overlap O.

The Segmentation Problem has the following formulation:

$$[\mathbf{SP}] : \min. \sum_{\tau\in\mathcal{T}, r\in\mathcal{R}} x_\tau^r \delta_\tau^r \qquad \text{s.t.:} \quad (19)$$

$$y_\tau^r \geq x_{(\tau+1)_{|\mathcal{T}|}}^r - x_\tau^r \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \quad (20)$$

$$\sum_{\tau\in\mathcal{T}} y_\tau^r \leq z^r \qquad \forall r \in \mathcal{R} \quad (21)$$

$$\sum_{r\in\mathcal{R}} x_\tau^r = 1 \qquad \forall \tau \in \mathcal{T} \quad (22)$$

$$\sum_{\tau\in\mathcal{T}} x_\tau^r \geq L \cdot z^r \qquad \forall r \in \mathcal{R} \quad (23)$$

$$\sum_{r\in\mathcal{R}} z^r \leq N \quad (24)$$

$$x_\tau^r, y_\tau^r, z^r \in \{0,1\} \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \quad (25)$$

Variables x_τ^r and y_τ^r have the same meaning as in **CRR** formulation, while a new set of variables z^r has been introduced. Since considered RCs are typically more than the required N, z^r set to 1 means that RC R_r has been selected, and constraint (24) ensures that no more than N can be selected overall. Constraints (20)–(23) have the same goal of (4)–(7), but considering only selected RCs.

In order to consider an overlap between adjacent clusters, formulation **SP** has to be modified in a similar way as discussed for the formulation **CRR**. The following constraints:

$$w_\tau^r \geq x_{(\tau-1)_{|\mathcal{T}|}}^r - x_\tau^r \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \quad (26)$$

$$\sum_{\tau\in\mathcal{T}} w_\tau^r \le z^r \qquad \forall r\in\mathcal{R} \tag{27}$$

$$w_\tau^r \in \{0,1\} \qquad \forall \tau\in\mathcal{T}, r\in\mathcal{R} \tag{28}$$

and a new objective function must be introduced in **SP**:

$$\min \sum_{\tau\in\mathcal{T},r\in\mathcal{R}} x_\tau^r \delta_\tau^r + \frac{1}{2}\sum_{\tau\in\mathcal{T},r\in\mathcal{R}} y_\tau^r \left(\sum_{(\tau-O<k\le\tau)_{|\mathcal{T}|}} \delta_k^r - \sum_{(\tau+1\le k\le\tau+O)_{|\mathcal{T}|}} \delta_k^r \right) + \frac{1}{2}\sum_{\tau\in\mathcal{T},r\in\mathcal{R}} w_\tau^r \left(\sum_{(\tau\le k<\tau+O)_{|\mathcal{T}|}} \delta_k^r - \sum_{(\tau-O\le k<\tau)_{|\mathcal{T}|}} \delta_k^r \right) \tag{29}$$

Constraints (26)–(27) are exactly the same as (11)–(12). The new objective function (29) and the objective function (18) seem to resemble each other, however there is an important difference: since δ_τ^r are now precomputed parameters, no new variables, like $\gamma_\tau^{-,r}$ and $\gamma_\tau^{+,r}$, are required.

5.2. STEP 2 - Robust Routing Problem

Once TMs have been clustered around a RC in STEP 1, STEP 2 computes a new robust RC R_c considering the TMs in the cluster. This will likely provide a better tailored routing. In addition, being a robust routing, it makes CRR intrinsically robust against noisy TM measurements.

For each cluster c, we compute RC R_c as robust routing that minimizes the average MLU over time when the set of TMs in the cluster, denoted as $\mathcal{T}_c$, is routed via R_c. The TMs in $\mathcal{T}_c$ are characterized by the same demand set $\mathcal{H}$, but different demand values, d_h^τ, varying according to the traffic time evolution. R_c will be defined by flow variables f_{ij}^h, which indicate the percentage of demand h's flow of every TM in $\mathcal{T}_c$ must be routed along the link (i,j).

The formulation of the Robust Routing Problem is:

$$[\mathbf{RR}]: \min. \sum_{\tau\in\mathcal{T}_c} \gamma_\tau \qquad \text{s.t.:} \tag{30}$$

$$\sum_{(i,j)\in\mathcal{L}} f_{ij}^h - \sum_{(j,i)\in\mathcal{L}} f_{ji}^h = \begin{cases} 1 & \text{if } i=\text{origin}(h) \\ -1 & \text{if } i=\text{destination}(h) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i\in\mathcal{N}, h\in\mathcal{H} \tag{31}$$

$$\gamma_\tau \ge \frac{\sum_{h\in\mathcal{H}} d_h^\tau f_{ij}^h}{c_{ij}} \qquad \forall \tau\in\mathcal{T}_c, (i,j)\in\mathcal{L} \tag{32}$$

$$0 \le f_{ij}^h \le 1 \qquad \forall h\in\mathcal{H}, (i,j)\in\mathcal{L} \tag{33}$$

$$0 \le \gamma_\tau \le 1 \qquad \forall \tau\in\mathcal{T}_c \tag{34}$$

The model shares part of the constraints of the **CRR** model, but it computes a single RC for a single cluster of TMs. Constraints (31) are standard flow conservation constraints for splittable routing. Constraints (32) makes γ_τ the MLU when routing TM $T(\tau)$ and, together with constraints (34), guarantee that the routing of each demand does not exceed the link capacity c_{ij} for any TM.

As a final remark, the set $\mathcal{R}$ used in STEP 1 is initialized by considering W RCs obtained by the solution of the Robust Routing problem over W sequential groups of TMs spanning the entire set $\mathcal{T}$. At the end of STEP 1, STEP 2 computes a set of RCs over the N generated clusters, which will be included in the initial set $\mathcal{R}$. This provides $\mathcal{R}$ with more refined RCs, which could be selected in the solution of the Segmentation Problem of the next iteration.

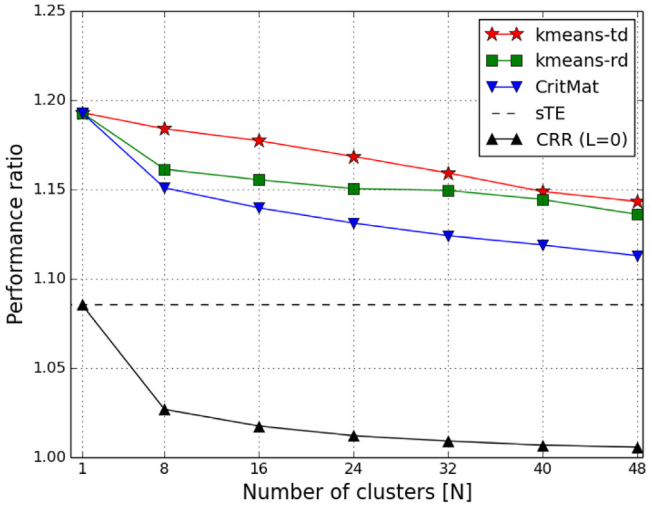

Fig. 5. Performance comparison of different TM clustering approaches.

6. Numerical results

In order to assess the performance of the proposed algorithm, we consider a daily scenario in which we compare our two-steps CRR algorithm to different routing solutions within the Abilene Network [33], whose traffic requests are described by a set of TMs with granularity 5 min (288 TMs for the entire day). Abilene network was one of the first high-performance backbone networks, connecting 11 cities across United States with 14 links. Nowadays, Abilene is one of the very few real data sets in which network TMs and routing are publicly available. We imagine a scenario in which the optimization of clusters and RCs for the day after is run during the night, on the basis of daily TM predictions. Unless differently indicated, we average obtained results over a week and run the algorithm for 10 iterations. The CRR algorithm has been implemented in Python, using Gurobi Solver language interface.

Given the values of the set of routing variables f_{ij}^{hr} of a particular routing configuration RC, we compute for each τ in $\mathcal{T}$ the Maximum Link Utilization (MLU) as

$$MLU_{RC}[\tau] = \max_{(i,j)\in\mathcal{L}} \frac{\sum_{h\in\mathcal{H}} d_h^k f_{ij}^{hr}}{c_{ij}} \tag{35}$$

The performance ratio is then defined as

$$PR = \frac{\sum_{\tau\in\mathcal{T}} MLU_{RC}[\tau]}{\sum_{\tau\in\mathcal{T}} MLU_{DynTE}[\tau]} \tag{36}$$

where $MLU_{DynTE}[\tau]$ is the MLU obtained considering as routing policy the Dynamic TE, which solves a min-MLU Multi-Commodity Flow (MCF) problem for each TM. The performance ratio allows to compare different routing policies over a same network topology and set of traffic matrices. Its value is always greater or equal to 1 and it lets us evaluate how far a solution is from the performance of the ideal routing, i.e. the one allowing to change the routing configuration at every TM.

6.1. Clustering approaches comparison

Fig. 5 shows the comparison of different clustering techniques. In the first set of experiments we neglect the minimum cluster length L and the overlap O to assess the maximum achievable gain with respect to state-of-the-art approaches, which do not consider these constraints. We measure the performance in terms of ratio between the time-average MLU of a particular routing policy (included our CRR scheme) and the ideal value achievable using Dynamic TE. On the x-axis, the number of generated clusters is shown. We tested our CRR algorithm against different alternative approaches:

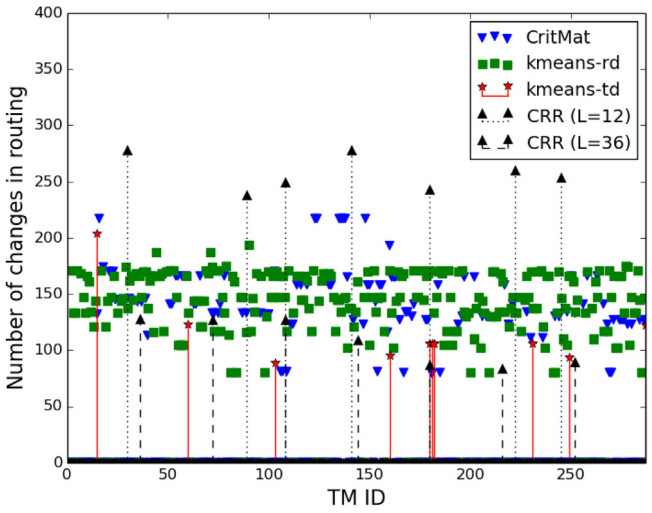

Fig. 6. Number of routing changes when different clustering approaches are applied. Example of solutions with 8 clusters..

- *sTE*: A static TE solution where a robust routing is computed over the entire TM set. The result is a single daily RC and no reconfiguration is required, like in the case of oblivious routing [12].
- *CritMat*: This approach, presented in [17], consists in clustering TMs according to dominating cluster heads, which are synthetic TMs including the maximum of each demand among the TMs grouped into the cluster. The RC associated to the cluster is the MLU-optimum routing for the cluster head.
- *K-means clustering*: Most of the TM clustering approaches in literature are based on a variant of the well-known k-means technique [34]. We consider the following two alternative clustering schemes based on k-means
  - *kmeans-td*: TMs are clustered considering their similarity in the traffic domain. The distance function used by k-means to compute the centroids is the Euclidean distance of the traffic values of the OD flows in the TMs. The distance between TMs τ_1 and τ_2 is defined as $dist(\tau_1, \tau_2) = \sqrt{\sum_{h \in \mathcal{H}} (d_h^{\tau_1} - d_h^{\tau_2})^2}$.
  - *kmeans-rd*: TMs are clustered according to their similarity in the routing domain. In this case the Euclidean distance is computed using as entries the routing variables obtained solving the min-MLU Multi-commodity Flow (MCF) problem for each TM. The distance between TMs τ_1 and τ_2 is defined as $dist(\tau_1, \tau_2) = \sqrt{\sum_{(i,j) \in \mathcal{L}} \sum_{h \in \mathcal{H}} (f_{ij}^{h\tau_1} - f_{ij}^{h\tau_2})^2}$.

  In both cases, once clusters are eventually created, we compute a MLU-optimum routing over the dominating TM of each cluster.

We can note how the proposed CRR algorithm outperforms all other alternatives. The curves' trend shows that *CritMat* dominating TM appears to be over-conservative, as the resulting congestion is even worse than that of *sTE*. Indeed, the outcome RCs can address such a large set of potential TMs that their working points are largely suboptimal when RCs are applied to specific TMs. Even the TM clustering approach based on the similarity among the optimal routing (as *kmeans-rd*) does not provide the best result: due to scenario symmetries, different RCs can provide the same congestion, therefore clustering on the mere basis of RC topology may waste clusters to separate TMs with different optimum routings, which could be equivalently well routed by another unique RC. In order to better include the routing effects in the cluster selection, we need to consider the ultimate effect of the routing, that is the network congestion resulting from applying a given RC to a given TM. Only TMs that are characterized by a small congestion with the same RC must be grouped together into the cluster associated to the specific RC.

Fig. 6 shows the reconfiguration intensity of the clustering approaches compared in Fig. 5. For the sake of clarity, we plot just the case in which $N = 8$ clusters are generated. The figure considers a typical day, the time is expressed in terms of ordered TM IDs and each ID corresponds to a 5-minute interval. Each point in the plot indicates the number of links that change their routing coefficients (f_{ij}^h variables in formulation **RR**) with respect to the routing in the previous TM. If this value is 0, it means that the previous RC is maintained. This roughly corresponds to measure the overall number of rule updates across all network switches. We can notice that *CritMat* and the k-means with routing-domain clustering (*kmeans-rd*) produce many reconfigurations, frequently changing activated RCs. K-means with TM-domain clustering (*kmeans-td*), instead, results more stable, however it still exhibits two main drawbacks. First, although being designed to use 8 clusters, it produces more reconfigurations (up to 10–11), as the 8 associated RCs are reused. Second, there are some reconfiguration bursts where RCs change after few minutes (e.g. around TM 180). The CRR algorithm with $L = 0$, which provides the best performance in terms of congestion, is characterized by an unstable routing behavior as well. Therefore, we need to explicitly provide a minimum cluster length guarantee to avoid route flapping problems, which, as we will see, comes at the cost of a small congestion increase. This guarantees results in a fixed number of transitions, each separated by the desired length L. In the figure, results for $L = 12$ and $L = 36$ are shown.

Fig. 7 reports routing change instants of Fig. 6: a bar at time τ indicates that at least one routing variable has changed from the previous instant $\tau - 1$. Even if all the five approaches produce no more than $N = 8$ distinct routing configurations, *CritMat* and kmeans produce a set of clusters that does not satisfy the constraints of minimum length (minimum holding time) and time contiguity (no routing repetitions), while CRR guarantees these two properties. We can observe that CRR, depending on L, M, N parameters, might produce clusters of non-uniform size if $N < M/L$ and this allows to better tailor RCs to particular TM sequences. For example, for a same value of $N = 8$, $CRR(L = 12)$ produces smaller clusters than $CRR(L = 36)$ and gets better performance (cfr. Fig. 8).

6.2. Impact of minimum cluster length

In Fig. 8, we assess the performance of CRR algorithm when the minimum cluster length constraint is activated with different values of L. The x-axis shows the number N of clusters in a day, while different curves represent different values of L. Note that the values of L and N are not independent: as N clusters are generated in one day, N cannot be larger than the ratio 24 h / L (in hours). Therefore curves with larger L stops at smaller N values. We can see that the minimum length constraint impacts on the performance of the clustering algorithm. With realistic N values, the MLU performance ratio increases by 4% approaching 1.06, which is still very close to the ideal dynamic TE. This performance is obtained with $N = 8$ and $L = 36$ results in keeping the same routing configuration for at least 3 consecutive hours and changing only 8 times the RC during the next day.

In order to check the consistency of the obtained results, we also considered a second topology from SNDlib dataset [35]. The Nobel Network is the German research network operated by the DFN-Verein. It includes 17 nodes interconnected by 26 links and routes traffic for 257 demands. Since the size of the model is driven by the number of links and demands, this topology provides a much bigger instance to validate our approach. Compared to Abilene, it indeed comprises almost 2 times links and more than 2 times demands. Results refer to the single day (288 TMs in total) available in the dataset. Fig. 9 reports the results of the same analysis of the clustering techniques and of the impact of minimum cluster length performed on the Abilene Network

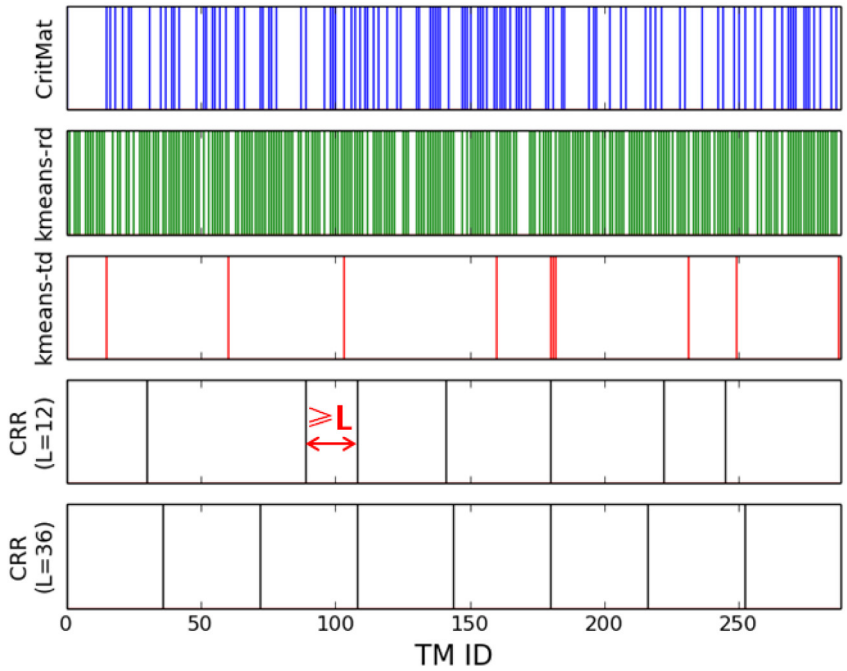

Fig. 7. Routing change instants. A vertical bar represents a routing reconfiguration and the space between two vertical lines represents a cluster, which maintains the same routing. CRR guarantees that the minimum cluster size is $\geq L$.

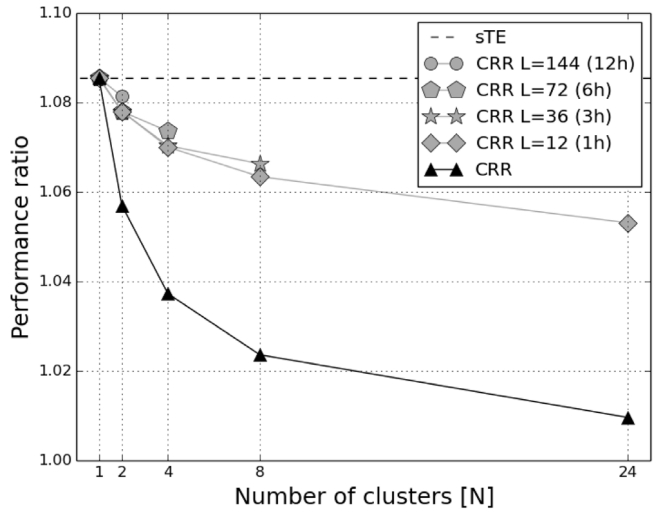

Fig. 8. Impact of different minimum cluster lengths on the CRR performance.

and reported in Figs. 5 and 8. The figure on the top right corner is the zoomed-in version of the one on the left. Since the results over these two topologies present a comparable trend, in the rest of the paper we focus just on Abilene Network.

6.3. Cluster overlap and robustness to switching time errors

In Fig. 10, we analyze the performance of the CRR algorithm varying the degree of overlap O. The figure shows on the x-axis the minimum length L imposed to the cluster, while different curves are plotted for multiple values of O. We can see the impact of the overlap is significant only for short clusters, while it becomes quickly negligible when the minimum cluster length increases. Moreover, note that each TM included in the overlap provides an overlap extension of 5 min on each side. Therefore, considering $O = 1, \ldots, 6$ corresponds to transition periods from 10 min to 1 h, which can reasonably include both the uncertainty on the transition instant and the signaling delay needed to anticipate reconfigurations.

The output of the CRR provides the set of routing configurations with corresponding cluster activation times. We define an activation time to be *correct* if the transition point lies in the middle of the overlapped regions. Fig. 11 evaluates the effect of a wrong (i.e. shifted) activation time. We selected a random activation time within the overlap region for each cluster transition: for example, evaluating the effect of a wrong activation time for overlap $O = 3$ implies trying to advance/delay the cluster activation time of up to 3 TMs (i.e. the time shift amounts to $0, \pm 1, \pm 2$ and ± 3 time instants). With respect to the objective function optimized by the **SP** model (and reported in Fig. 10), which considers an averaged congestion for the TMs in the overlapped region, here we take a decision on the specific activation time and evaluate the resulting objective function assigning each TM to exactly one of the two overlapping clusters, according to the shifted activation time under analysis. The red bars in Fig. 11 report the boxplot of the objective function for different shifted activation times and for different values of overlap O and minimum cluster length L on a typical day. The number of possible sequences of shifted activation times grows like $(2 \cdot O + 1)^{(K_{max}-1)}$ where $K_{max} = M/L$. Since we cannot afford to completely explore them, the boxplots results from 5000 instances of randomly picked activation times sequences. We perform the same kind of evaluation also for the routing configurations obtained by solving the CRR without overlap and report the results in the blue bars. The green line reports instead the sTE solution. If we focus on the bars as a whole, performance decreases as the overlap amount increases, confirming the results of the previous figure. Focusing instead of a single pair of red/blue bars for a given value of overlap O and minimum cluster length L, we can appreciate that including non-zero overlap in the CRR can effectively help in limiting the worst performance degradation in case of a wrong activation time. Put it in another way, the overlap makes the reconfiguration decision less critical, since the performance deviation measured as the gap between the largest and smallest Performance Ratio of the box-plots in the figure is rather limited.

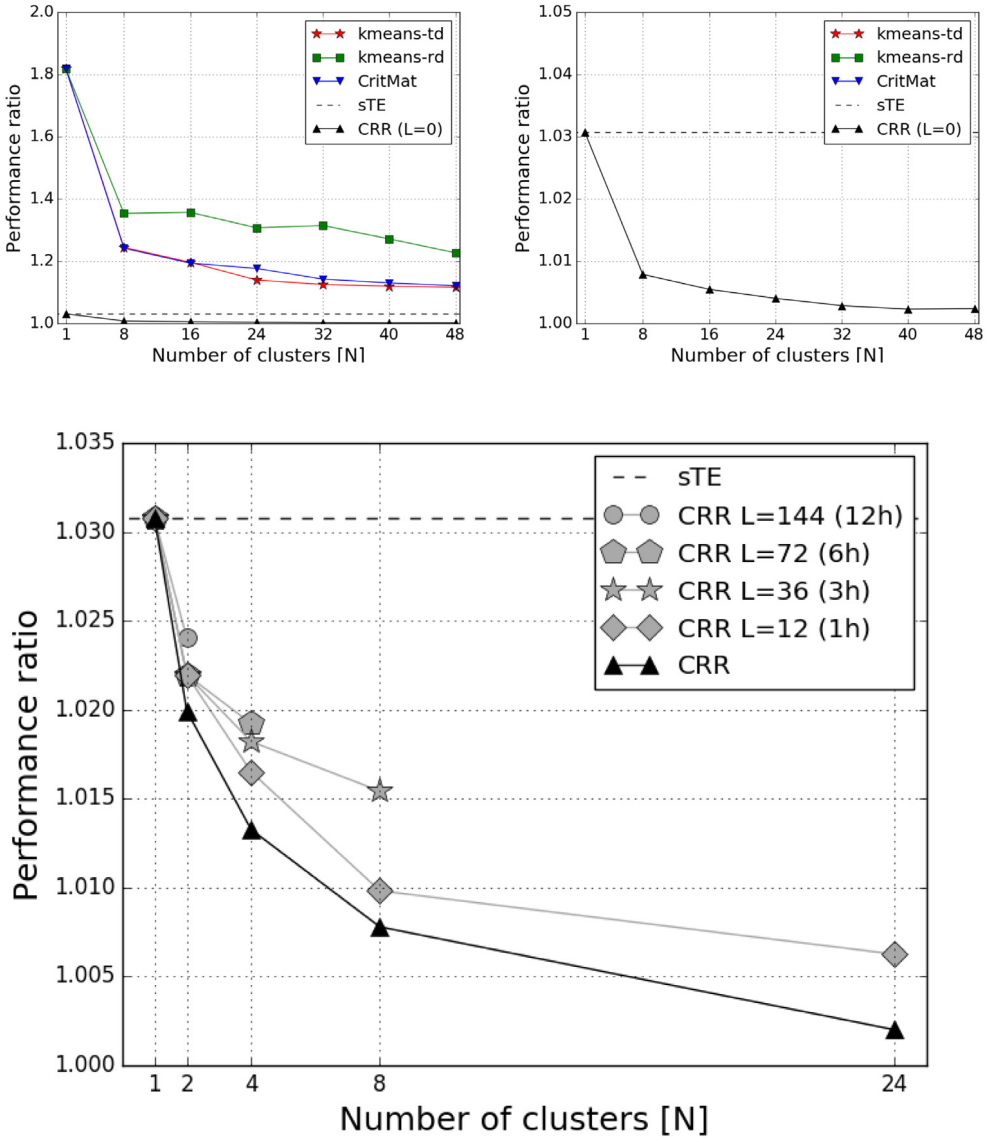

Fig. 9. Performance comparison of different TM clustering approaches and impact of different minimum cluster lengths on the CRR performance for Nobel Topology.

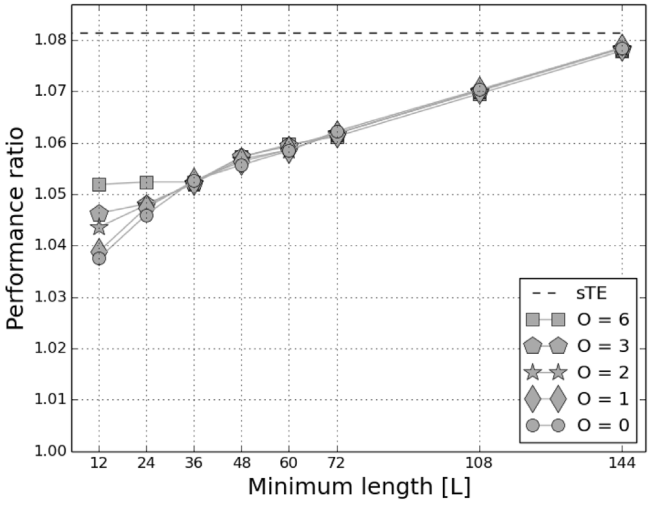

Fig. 10. Impact of different degrees of overlap on the CRR performance.

6.4. Impact of prediction error

In the previous sections, we have analyzed the performance of the CRR algorithm when the clustering and the related RCs are computed over a set of TMs and applied to the same set. This corresponds to assume perfect TM prediction and provide the potential performance achievable by the algorithm. In this section, we relax this assumption and analyze the impact of prediction errors.

In order to reproduce the effect of unideal predictions, we run the CRR algorithm over a noisy version of the daily set of TMs to compute clusters and RCs. Then, we apply the RCs to the original set of TMs, which represent the real traffic behavior. Therefore, the TM sets used to compute and evaluate the clustering approach differ. Each noisy TM has been obtained from the original one by adding a uniform relative error $[-\alpha, \alpha]\%$ to every OD demand d_h^τ. The results of these experiments are shown in Fig. 12, where the performance achievable with different cluster lengths L and prediction errors α is reported. Similarly to previous analyses, the performance is computed as the ratio between the average network MLU over the ideal case of applying dynamic TE in perfect prediction conditions.

We can clearly note that the absolute performance of CRR is negatively affected by the presence of prediction errors. However the

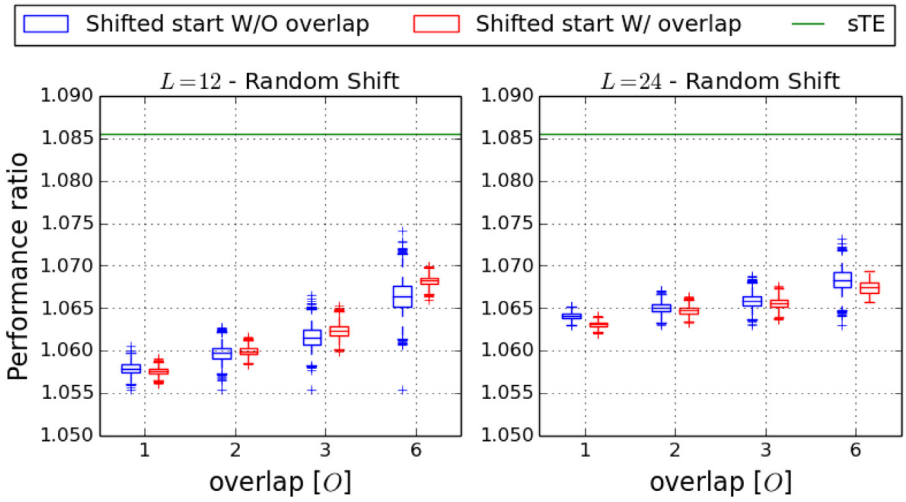

Fig. 11. Impact of wrong cluster activation times. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

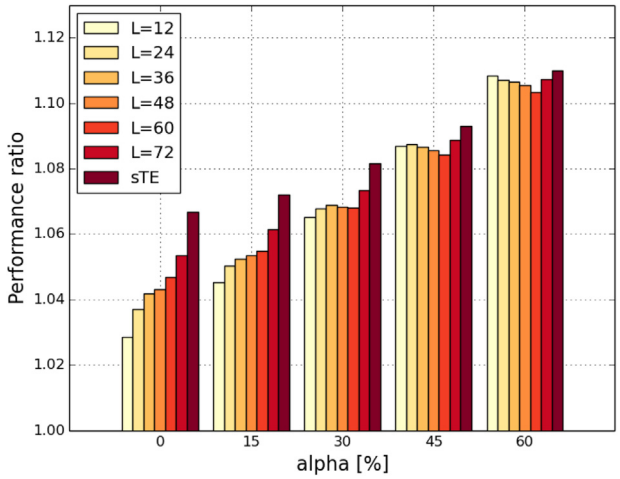

Fig. 12. CRR performance when prediction error is considered. Results are expressed as performance ratio with respect to the ideal optimum routing.

intrinsic robustness of the clustered approach limits the performance decrease. Even with large errors, the gap with respect to the ideal Dynamic TE is within 10%–11%.

It can be further observed that the gain of CRR with small clusters fades away as the noise increases. Indeed, when the prediction accuracy of TMs decreases, considering robust RCs computed over larger sets of TMs provides the best performance ratio. Taking this to the extreme, when we have very low-quality predictions, no clustering can be helpful, because the representative set of TMs and the actual traffic will have little correlation. Therefore, a trade-off between cluster length and prediction accuracy exists. In case of good predictions, the size of the clusters drives the performance. In contrast, if predictions are affected by large errors, the impact of TM uncertainness completely overwhelms the effect of cluster sizes.

7. Experimental testbed

In order to assess the feasibility of our approach in a Software-Defined Network, we have integrated the CRR algorithm in Open Network Operating System (ONOS) [36], a production-ready open source SDN network operating system built for Service Provider networks.

ONOS provides high performance, scalability and availability thanks to its distributed core and proper abstractions to configure the network. Intents represent the highest level of abstraction: developers can focus on *what* should be done, rather than *how* it should be done, by expressing their "intentions" via high-level policies. For example, users can request connectivity between a pair of elements in the network. ONOS supports different types of intents, each one supplied with a compiler which enables ONOS core to translate the high-level policy to low-level rules to be installed in network devices. The component of ONOS responsible of handling intents is the Intent Framework. We selected intents as the most suitable mean to integrate CRR in ONOS because they allow to decouple the definition of connectivity requests (in terms of endpoints, i.e. TM endpoints) from the actual routing decision (in terms of specific path realizing the communication). In this way the application submitting intents is independent from the algorithm (CRR) deciding *when* and *how* to update the paths. In addition, our routing logic can be reused by any ONOS application based on intents.

The CRR algorithm determines the paths used by the Intent Framework when compiling the intents. Integrating such a computationally heavy component in the same machine which runs the controller can deeply affect the high performance requirements of ONOS. We thus developed a new service to complement ONOS with an external plug&play routing logic: ONOS Intent Monitor and Reroute (IMR) service[8] [37]. This service orchestrates the monitoring (flow statistics collection) and the rerouting (path changes) of the intents and communicates with an off-platform application (OPA), running the CRR, via a set of REST APIs. The details of the interactions between the different components, as depicted in Fig. 13, can be found in [37].

We selected SDN-IP [38] as an example of intent-based ONOS application whose performance can be enhanced by exploiting CRR through our IMR service. SDN-IP is an ONOS application which enables SDN network to connect to legacy IP networks using Border Gateway Protocol (BGP) while appearing externally as a traditional Autonomous System (AS).

We modified the SDN-IP application to require the IMR service to monitor the intents related to AS-to-AS traffic and to expose their statistics to an OPA running our CRR algorithm. We refer to this modified application as *Extended SDN-IP*. We implemented an experimental testbed based on Mininet [39], replicating the Abilene topology and attaching to each node an external BGP speaker with a single host. We replayed a subset of a 3-day TM set generating data using *iperf* traffic tool.

The CRR framework learns the traffic pattern from previous days and adapts the routing strategy by creating multiple clusters for the

[8] IMR has been included in ONOS *Nightingale* version 1.13 as an official contribution.

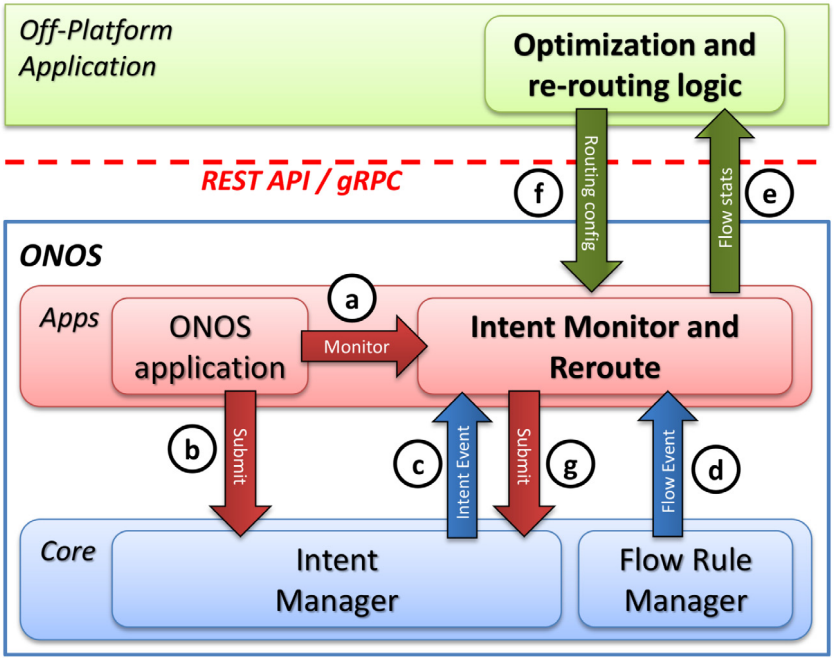

Fig. 13. IMR interactions with ONOS and with the OPA.

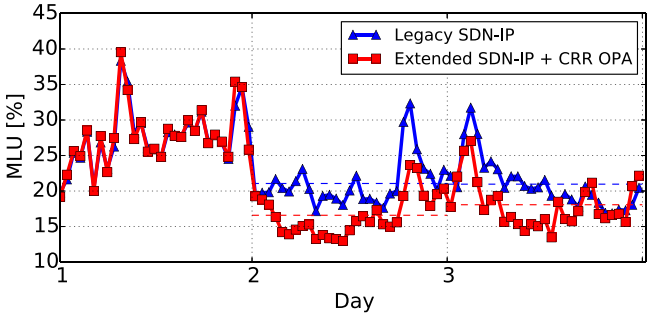

Fig. 14. SDN-IP application enhanced by CRR – Average MLU comparison.

next day. More specifically, during the first day, intents are compiled by the standard Intent Framework and statistics are collected and exposed to the OPA. At the end of the first training period (first day of data), collected TMs are fed to the CRR algorithm presented in Section 4 to compute a set of three robust routing configurations for the following day. During the second day, the OPA schedules the activation of the routing configurations and at the same time keeps collecting statistics in order to potentially refine the clusters and routing configurations for the following periods. Fig. 14 shows the MLU during the 3 days and compares the daily-average MLU (dashed line) obtained with the *Legacy SDN-IP* (in blue) against the one obtained using our *Extended SDN-IP* application enhanced by the CRR algorithm. During the first day the trend of the MLU is identical because the two applications both rely on the standard Intent Framework. After the first day of training, we can appreciate a 5% decrease in the average MLU for the *Extended SDN-IP*. We can also observe that, even if routing configurations have been computed over TM measurements from the previous day, the robust nature of each routing configuration can limit MLU peaks.

8. Conclusion

In this paper we investigated how robust routing approaches can be made adaptive in the SDN context. Assuming the availability of traffic predictions, we designed an off-line method to split the traffic space into smaller partitions and build routing configurations that are robust against any real-time traffic variation within the partition. Differently from previous solutions proposed in the literature, our clustering approach considers both technical and practical constraints in the modeling, like the minimum holding time of a routing configuration and a temporal overlap that makes routing changes smooth. The results showed that routing configurations based on TM clustering can achieve a performance very close to the optimal routing only if a good clustering domain is chosen. Our proposal based on the estimation of the congestion caused by the activation of a given routing outperforms the other candidate solutions.

We then investigated the behavior of our solution when the accuracy of traffic predictions varies. It showed an interesting trade-off between cluster sizes and prediction errors that opens a new research direction for the dynamic orchestration of routing configurations.

We finally showed how the proposed solution can be readily implemented in a production-ready SDN controller.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Linear CRR model

The linearized version of **CRR** problem, including overlap, is described by the following MILP model:

$$
\begin{aligned}
&[\textbf{Linear CRR}] : \min. \sum_{\tau\in\mathcal{T},r\in\mathcal{R}} \gamma_\tau^r + \\
&\frac{1}{2}\sum_{\tau\in\mathcal{T},r\in\mathcal{R}}\left(\sum_{(\tau-O<k\le\tau)_{|\mathcal{T}|}} \gamma_\tau^{-,r} - \sum_{(\tau+1\le k\le\tau+O)_{|\mathcal{T}|}} \gamma_k^r\right) + \\
&\frac{1}{2}\sum_{\tau\in\mathcal{T},r\in\mathcal{R}}\left(\sum_{(\tau\le k<\tau+O)_{|\mathcal{T}|}} \gamma_\tau^{+,r} - \sum_{(\tau-O\le k<\tau)_{|\mathcal{T}|}} \gamma_k^r\right) \qquad \text{(A.1)} \\
&\sum_{(i,j)\in\mathcal{L}} f_{ij}^{hr} - \sum_{(j,i)\in\mathcal{L}} f_{ji}^{hr} = \begin{cases} 1 & \text{if } i = O_h \\ -1 & \text{if } i = D_h \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

$$\forall i \in \mathcal{N}, h \in \mathcal{H}, \forall r \in \mathcal{R} \tag{A.2}$$

$$\gamma_\tau^r \geq \frac{\sum_{h\in\mathcal{H}} d_h^\tau f_{ij}^{hr}}{c_{ij}} - A \cdot (1 - x_\tau^r)$$

$$\forall \tau \in \mathcal{T}, \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{L} \tag{A.3}$$

$$\gamma_\tau^{-,r} \geq \frac{\sum_{h\in\mathcal{H}} d_h^k f_{ij}^{hr}}{c_{ij}} - B \cdot (1 - y_\tau^r)$$

$$\forall \tau \in \mathcal{T}, \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{L}, \forall k : (\tau - O < k \leq \tau)_{|\mathcal{T}|} \tag{A.4}$$

$$\gamma_\tau^{+,r} \geq \frac{\sum_{h\in\mathcal{H}} d_h^k f_{ij}^{hr}}{c_{ij}} - C \cdot (1 - w_\tau^r)$$

$$\forall \tau \in \mathcal{T}, \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{L}, \forall k : (\tau \leq k < \tau + O)_{|\mathcal{T}|} \tag{A.5}$$

$$y_\tau^r \geq x_{(\tau+1)_{|\mathcal{T}|}}^r - x_\tau^r \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{A.6}$$

$$w_\tau^r \geq x_{(\tau-1)_{|\mathcal{T}|}}^r - x_\tau^r \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{A.7}$$

$$\sum_{\tau\in\mathcal{T}} y_\tau^r \leq 1 \qquad \forall r \in \mathcal{R} \tag{A.8}$$

$$\sum_{\tau\in\mathcal{T}} w_\tau^r \leq 1 \qquad \forall r \in \mathcal{R} \tag{A.9}$$

$$\sum_{r\in\mathcal{R}} x_\tau^r = 1 \qquad \forall \tau \in \mathcal{T} \tag{A.10}$$

$$\sum_{\tau\in\mathcal{T}} x_\tau^r \geq L \qquad \forall r \in \mathcal{R} \tag{A.11}$$

$$x_\tau^r, y_\tau^r, w_\tau^r \in \{0,1\} \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{A.12}$$

$$0 \leq \gamma_\tau^r \leq 1 \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{A.13}$$

$$0 \leq \gamma_\tau^{-,r} \leq 1 \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{A.14}$$

$$0 \leq \gamma_\tau^{+,r} \leq 1 \qquad \forall \tau \in \mathcal{T}, r \in \mathcal{R} \tag{A.15}$$

$$0 \leq f_{ij}^{hr} \leq 1 \qquad \forall (i,j) \in \mathcal{L}, \forall h \in \mathcal{H}, r \in \mathcal{R} \tag{A.16}$$

A, B and C are needed to deactivate constraints (A.4)–(A.6) when x_τ^r, y_τ^r and w_τ^r are respectively set to 0. We can simply select

$$A = B = C = \frac{\max_{\tau\in\mathcal{T}} \sum_{h\in\mathcal{H}} d_h^\tau}{\min_{(i,j)\in\mathcal{L}} c_{ij}}$$

References

[1] N. Wang, K. Ho, G. Pavlou, M. Howarth, An overview of routing optimization for internet traffic engineering, IEEE Commun. Surv. Tutor. 10 (1) (2008) 36–56.

[2] D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.

[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with software-driven wan, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 15–26.

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al., B4: Experience with a globally-deployed software defined wan, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 3–14.

[5] M. Malboubi, L. Wang, C.N. Chuah, P. Sharma, Intelligent sdn based traffic (de)aggregation and measurement paradigm (istamp), in: Proc. IEEE INFOCOM, 2014.

[6] T. Benson, A. Anand, A. Akella, M. Zhang, Microte: Fine grained traffic engineering for data centers, in: Proc. ACM CoNext, 2011, p. 8.

[7] M. Roughan, M. Thorup, Y. Zhang, Traffic engineering with estimated traffic matrices, in: Proc. ACM IMC, 2003.

[8] K. Murakami, H.S. Kim, Optimal capacity and flow assignment for self-healing atm networks based on line and end-to-end restoration, IEEE/ACM Trans. Netw. 6 (2) (1998) 207–221.

[9] C. Albrecht, Global routing by new approximation algorithms for multicommodity flow, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 20 (5) (2001) 622–632.

[10] X. Jin, H.H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, R. Wattenhofer, Dynamic scheduling of network updates, ACM SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 539–550.

[11] S. Paris, A. Destounis, L. Maggi, G.S. Paschos, J. Leguay, Controlling flow reconfigurations in sdn, in: Proc. IEEE INFOCOM, 2016.

[12] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, H. Racke, Optimal oblivious routing in polynomial time, in: ACM Symp. on Theory of Computing, 2003, pp. 383–388.

[13] V. Tabatabaee, A. Kashyap, B. Bhattacharjee, R.J. La, M.A. Shayman, Robust routing with unknown traffic matrices, in: Proc. IEEE INFOCOM, 2007, pp. 2436–2440.

[14] M. Kodialam, T. Lakshman, S. Sengupta, Efficient and robust routing of highly variable traffic, in: Proc. HotNets, 2004.

[15] H. Wang, H. Xie, L. Qiu, Y.R. Yang, Y. Zhang, A. Greenberg, Cope: traffic engineering in dynamic networks, ACM SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 99–110.

[16] P. Casas, L. Fillatre, S. Vaton, Multi hour robust routing and fast load change detection for traffic engineering, in: Proc. IEEE ICC, 2008, pp. 5777–5782.

[17] Y. Zhang, Z. Ge, Finding critical traffic matrices, in: Proc. IEEE DSN, 2005.

[18] D. Sanvito, I. Filippini, A. Capone, S. Paris, J. Leguay, Adaptive robust traffic engineering in software defined networks, in: Proc. IFIP Networking, 2018.

[19] T. Holterbach, S. Vissicchio, A. Dainotti, L. Vanbever, Swift: Predictive fast reroute, in: Proc. ACM SIGCOMM, 2017.

[20] S. Brandt, K.-T. Förster, R. Wattenhofer, On consistent migration of flows in sdns, in: Proc. IEEE INFOCOM, 2016, pp. 1–9.

[21] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C.L. Lim, R. Soulé, Semi-oblivious traffic engineering: The road not taken, in: USENIX NSDI, 2018.

[22] D. Applegate, E. Cohen, Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs, in: Proc. ACM SIGCOMM, 2003.

[23] R. Zhang-Shen, Valiant load-balancing: Building networks that can support all traffic matrices, in: Algorithms for Next Generation Networks, Springer London, 2010, pp. 19–30.

[24] P. Casas, F. Larroca, S. Vaton, Robust routing mechanisms for intradomain traffic engineering in dynamic networks, in: Proc. IEEE LANOMS, 2009, pp. 1–10.

[25] W. Ben-Ameur, M. Żotkiewicz, Robust routing and optimal partitioning of a traffic demand polytope, Intl. Trans. Oper. Res. 18 (3) (2011) 307–333.

[26] M. Silva, M. Poss, N. Maculan, Solving the bifurcated and nonbifurcated robust network loading problem with k-adaptive routing, Networks 72 (1) (2018) 151–170.

[27] M. Poss, C. Raack, Affine recourse for the robust network design problem: Between static and dynamic routing, Networks 61 (2) (2013) 180–198.

[28] W. Ben-Ameur, M. Żotkiewicz, Multipolar routing: where dynamic and static routing meet, Electron. Notes Discrete Math. 41 (2013) 61–68.

[29] W. Ben-Ameur, H. Kerivin, Routing of uncertain traffic demands, Opt. Eng. 6 (3) (2005) 283–313.

[30] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, D. Towsley, On optimal routing with multiple traffic matrices, in: Proc. IEEE INFOCOM, Vol. 1, 2005, pp. 607–618.

[31] M. Reitblatt, N. Foster, J. Rexford, D. Walker, Consistent updates for software-defined networks: Change you can believe in!, in: Proc. Work. on Hot Topics in Networks, ACM, 2011, p. 7.

[32] W. Wang, W. He, J. Su, Y. Chen, Cupid: Congestion-free consistent data plane update in software defined networks, in: Proc. IEEE INFOCOM, 2016, pp. 1–9.

[33] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E.D. Kolaczyk, N. Taft, Structural analysis of network traffic flows, ACM SIGMETRICS PER 32 (1) (2004) 61–72.

[34] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, Oakland, CA, USA, 1967, pp. 281–297.

[35] S. Orlowski, M. Pióro, A. Tomaszewski, R. Wessäly, SNDlib 1.0–survivable network design library, in: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium, 2007.

[36] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, Onos: Towards an open, distributed sdn os, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, in: HotSDN '14, ACM, New York, NY, USA, 2014, pp. 1–6.

[37] D. Sanvito, D. Moro, M. Gulli, I. Filippini, A. Capone, A. Campanella, Onos intent monitor and reroute service: enabling plug&play routing logic, in: IEEE NetSoft, 2018, pp. 272–276.

[38] P. Lin, et al., Seamless interworking of sdn and ip, in: ACM SIGCOMM Computer Communication Review, 2013, pp. 475–476.

[39] B. Lantz, et al., A network in a laptop: Rapid prototyping for software-defined networks, in: ACM Hotnets-IX, 2010, p. 19.