IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2019, accepted July 15, 2019, date of publication July 18, 2019, date of current version August 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929651

On Load Balancing via Switch Migration in Software-Defined Networking

F. AL-TAM AND N. CORREIA
Center for Electronic, Optoelectronic and Telecommunications (CEOT), Faculty of Science and Technology, University of Algarve, 8005-139 Faro, Portugal
Corresponding author: F. Al-Tam (ftam@ualg.pt)

This work was developed within the Center for Electronic, Optoelectronic and Telecommunications (CEOT), and supported by the UID/MULTI/00631/2019 project of the Portuguese Science and Technology Foundation (FCT).

ABSTRACT Switch-controller assignment is an essential task in multi-controller software-defined networking. Static assignments are not practical because network dynamics are complex and difficult to predetermine. Since network load varies both in space and time, the mapping of switches to controllers should be adaptive to sudden changes in the network. To that end, switch migration plays an important role in maintaining dynamic switch-controller mapping. Migrating switches from overloaded to underloaded controllers brings flexibility and adaptability to the network but, at the same time, deciding which switches should be migrated to which controllers, while maintaining a balanced load in the network, is a challenging task. This work presents a heuristic approach with solution shaking to solve the switch migration problem. Shift and swap moves are incorporated within a search scheme. Every move is evaluated by how much benefit it will give to both the immigration and outmigration controllers. The experimental results show that the proposed approach is able to outweigh the state-of-art approaches, and improve the load balancing results up to $\approx$ 14% in some scenarios when compared to the most recent approach. In addition, the results show that the proposed work is more robust to controller failure than the state-of-art methods.

INDEX TERMS Load balancing, software-defined networking, switch migration, heuristic.

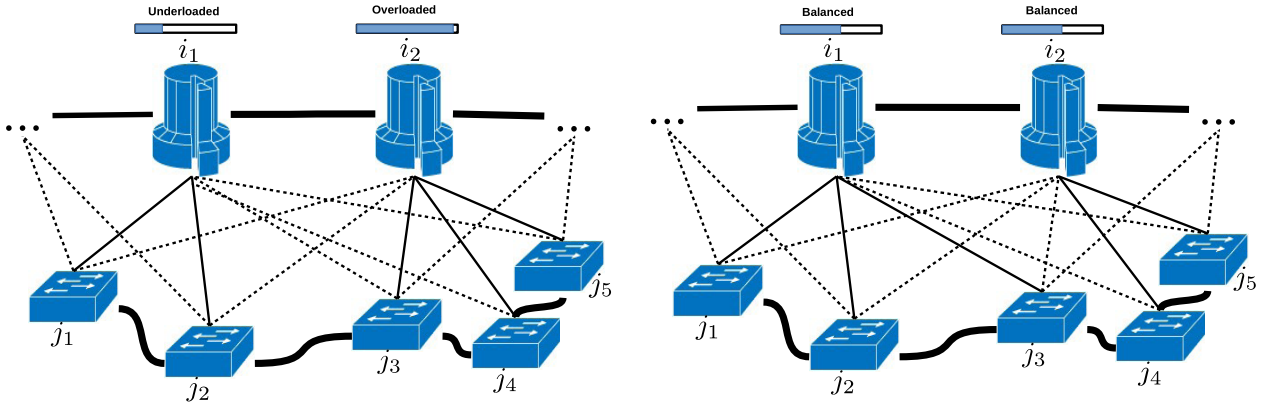
I. INTRODUCTION

In networking, data travels according to predefined policies. These are defined in the management plane, enforced by the control plane and carried out by the data plane. Therefore, policies basically render into forwarding rules that are stored in the forwarding devices. Upon the arrival of a flow, the forwarding device consults the stored rules in its table (or tables) to decide what to do with packets. In traditional IP networks, the data and control functionalities are integrated in the same device, which is meant to create resilient networks in the first place but, at the same time, yields rigid paradigms that eventually lead to a phenomenon called ''network ossification'' [1], i.e., stubborn to modification. In addition, network dynamics requires complex policies, which in turn require more configuration and management efforts, especially when having heterogeneous devices in the same network. The need to simplify network management via programmable devices has led to active networking (AN) [2] and to software-defined networking (SDN) [3]–[6].

The associate editor coordinating the review of this manuscript and approving it for publication was Yulei Wu.

Unlike traditional IP networks, SDN paradigm decouples the management, control, and data planes, and allows forwarding devices in the data plane to be programmable. This way, traffic engineering in SDN can be accomplished efficiently [7], and network management and provisioning can be achieved more easily [8]. The brain of any SDN is the control plane represented by a logically centralized (can be physically distributed) entity called controller. The main roles of the controller are to provide a consolidated network view and enforce forwarding rules, which can be achieved efficiently because the controller offers programmable northbound interfaces to the management plane, and southbound interfaces (e.g., OpenFlow [9]) to the data plane.

Whenever a new packet arrives at a forwarding device (e.g., an OpenFlow switch), a lookup process is initiated. The packet is forwarded if it matches a rule in the flow table of the switch, or simply dropped in case it does not have any match. However, when there is a `table-miss` rule (available from OpenFlow V1.3 [10]), the typical action is to encapsulate the unmatched packet in a `packet-in` message for it to be forwarded to the responsible controller.



(a) Before switch migration

(b) After migrating switch j_3 from i_2 to i_1

FIGURE 1. A schematic representation of load balancing via SMP. Thick solid lines are the intra-plane connections, thin solid lines are the inter-plane master assignments, and thin dashed lines are the inter-plane slave assignments. (a) Before switch migration. (b) After migrating switch j_3 from i_2 to i_1.

The controller reactively calculates the path of this new packet and replies via `packet-out` and/or `flow-mod` to install the required forwarding rules at the flow tables of the affected switches. In addition, proactive and hybrid modes are also possible. In the former, the rules are pre-installed in the switches (i.e., static [11]), while in the latter the rules are pre-installed but can be updated [12].

Early proposed SDN architectures assume a single controller which can easily offer a uniform network wide-view, since all information about the network is centralized in a single controller. However, in this scheme, the controller is the bottleneck of the network. That is, when the controller fails to meet the service level agreements (SLAs), the advantages of SDN will be lost [13]. On the other hand, multi-controller SDNs are becoming more popular due to their scalability and resilience advantages [12], [14]. Moreover, static mapping of switches to controllers has shown to be inefficient under network dynamics [15], and adaptive mapping is required. However, maintaining dynamic switch-controller assignments under traffic fluctuations is challenging because traffic oscillates spatially and temporally. A solution is to reactively move (migrate) switches from one controller domain to another to adapt to network changes. The problem of determining the set of switches to be migrated between controllers is known as the switch migration problem (SMP).

Our aim in this article is to develop an approach that improves the state-of-art load balancing results via switch migration. The main contributions of this work are:

- A switch migration model;
- A heuristic algorithm, which is able to improve the state-of-art load balancing results;
- A comprehensive comparison with the state-of-art load balancing methods.

The main advantages of the proposed work are:

- Unlike existing approaches, the proposed algorithm does not halt the search whenever a switch migration is not possible. Instead, it searches for more complex moves like swapping two switches to further improve the results. Which increases its robustness;
- A controlled solution shaking (noising) approach is developed to obtain even further improvements, which gradually reduces shaking as the solution matures.
- The proposed search scheme can be tuned easily to obtain speed-performance tradeoff.

Such approach, up to our humble knowledge, has not been studied in the literature for solving the SMP. The results confirm that our search scheme is able to improve the load balancing state-of-art results.

This article is organized as follows: The switch migration problem is described in Section II. The related work is briefly discussed in Section III. Section IV presents the proposed mathematical model. The proposed heuristic algorithm is described in Section V. Section VI presents the experimental results, and the article is concluded in Section VII.

II. SWITCH MIGRATION PROBLEM

A key-enabler to achieve dynamic mapping is to monitor the resources of the controllers and move (migrate) switches from overloaded to underloaded controllers (Figure 1). Switch migration became possible after the multi-controller support in OpenFlow protocol V1.2 [9] and the development of a switch migration four-phase protocol [15]. In multi-controller SDN, a controller can be master, slave, or equal. The master controller is the only controller in the domain that is able to update the switches' flow tables. That is, each switch can not have more than a master controller, but can have one or more equal and slave controllers [9]. When a master controller fails, an equal or slave controller can replace it. The task of changing the master controller of a switch is called switch migration.

*Definition 1 (**switch migration problem - SMP**):* Given a non-optimal network state at time t (e.g., low network utility

or controller failure), the SMP is to determine at time $t+1$ which set of switches should be migrated from outmigration controllers to immigration controllers so that a set of merit functions (e.g., network utilization) is maximized, and a set of loss functions (e.g., control plane overhead) is minimized, while obeying some constraints (e.g., maximum capacity of controllers).

Therefore, a solution to SMP is supposed to find three unknown sets: outmigration controllers, switches to migrate, and immigration controllers, which is an NP-hard problem [16] and requires a heuristic algorithm to solve it within a reasonable time-frame.

Load balancing is an issue point in multi-controller [17], [18] and, in fact, load balancing is one of the main reasons to migrate switches. Therefore, in order to handle the SMP properly, it becomes important to identify the main tasks that handling them constitutes the load at controllers. Mainly, the load at a controller is comprised by handling these tasks:

- Processing the received `packet-in` messages and installing forwarding rules [19], [20];
- Topology management;
- Communication with other controllers to synchronize network state [19];
- Network traffic statistics and other signaling events [21];
- Update of rules due to change in network policy [22].

Handling each of these tasks contributes to load at a controller, but the processing of `packet-in` messages is the major component [20]. Therefore, switch migration is mainly engineered with respect to the number of received `packet-in` messages at controllers [17].

Another important aspect is to determine when is switch migration necessary. Most of the literature is coined around defining a load imbalancing threshold, and once this threshold is reached the switch migration is triggered [16], [21], [23]. Although switch migration is required in many cases, it is possible to state that it is mainly triggered by the following situations:

- Due to network traffic dynamics, some controllers can be turned off to save communication cost and energy, i.e., to obtain a feasible number of controllers that are able to meet the SLAs while utilizing the available resources [16], [24]–[26];
- When all controllers are fully utilized, new controllers need to join the control plane pool [16], [25], [26];
- When load imbalance occurs, some controllers become overcommitted (hot spots), while others are underloaded (cold spots). Therefore, some switches need to be migrated to balance the load and maximize the utility of the network. In fact, an overloaded controller can fail and lead to a cascade failure [27];
- When a switch is not able to communicate to its master controller due to node/link failure, this "orphaned" switch needs to be reassigned to another controller;
- Switch migration can also take place for security reasons [28];

III. RELATED WORK

SMP has been firstly addressed in [15] and is now a hot topic [16], [28], [29]. The publication of [15] has triggered many papers since then. This section reviews the most relevant and recent work related to the proposed approach.

In the literature, the general scheme in solving the SMP includes sorting the load of controllers and then greedily migrate switches from overloaded to underloaded controllers [16], [23], [26], [30]. Other approaches based on simulated annealing can be found in [45], [47]. In [23], centralized and distributed algorithms are presented to solve SMP in order to balance the load of controllers in data centers. A greedy algorithm is developed in [26] to balance the load and guide the controller selection problem. Load diversity between the controllers is used in [16] to detect load imbalance, and a greedy algorithm is developed to solve the SMP. In [28], the SMP is formulated as an integer linear programming inspired from the earth mover distance (EMD) and a heuristic algorithm that solves simplified linear and integer programming problems is also developed to solve the SMP. In [40], we have studied SMP under fractional assignment. Moreover, in [31] we have developed an SMP model but no heuristic algorithm was proposed. A two-phase algorithm is developed in [33] to find a switch-controller assignment. The assignment problem is treated as a college admission problem, where in the second stage, a game theory method is used to improve the initial assignment and obtain a Nash stable solution. The load balancing problem is treated as a graph partitioning problem in [43] and a heuristic of three stages called Balanced Controller (BalCon) is developed. The first stage monitor and detect congestion, the second stage cluster and evaluate the migration, while the final task perform the migration. In [32] a bidirectional matching strategy (BMS) is developed which takes into account the matching preferences of the switches and controllers in a similar way to [33]. More recently, an on-line controller load balancing algorithm (OCLB) [29] was developed to solve the SMP. This algorithm iteratively reduces the load imbalance until no further switch migration is willing to improve the solution. In [41], a non-cooperative game among multiple controllers is used to solve the SMP. The SMP is modeled as a reinforcement learning problem in [44] and the Q-learning is used to solve it. In [42] algorithms for in-band and out-band traffic offloading using switch migration is proposed, where a so called control flow table is also proposed to realize fractional flow shifting. Breadth-first search (BFS) is used in [48] to develop a control-domain adjustment algorithm (CDAA), where BFS is used to select the outmigration switches based on the switch distance to the its master controller and current traffic load. Here in this article, we go further and improve the sate-of-art results on load balancing via switch migration using a local search approach.

IV. MATHEMATICAL MODEL

A. NOTATION AND ASSUMPTIONS

Consider a set of controllers and a set of switches, denoted by $\mathcal{I}$ and $\mathcal{J}$, respectively, whose physical connections are described by the binary adjacency matrix $\boldsymbol{G}_{m\times n}$. Each switch is managed by exactly one master controller and is assigned to at least one slave controller (Figure 1). The switch-controller master assignment is described by the binary matrix $\boldsymbol{S}_{m\times n}$. The set of switches having controller i as master, called a domain, is denoted by $\mathcal{J}_i$, and the set of switches having controller i as slave is denoted by $\mathcal{J}'_i$. Each switch j sends a number of `packet-in` messages, r_j, per second to its master controller in order to handle the incoming new flows, and each controller has a limited capacity to process the received requests. Further notation and variables are described in Table 1.

TABLE 1. Notation used throughout this article.

Term	Description
$\mathcal{I}$	Set of m controllers
$\mathcal{J}$	Set of n switches
$\mathcal{J}_i$	Set of switches having controller i as master (i.e., domain)
$\mathcal{J}'_i$	Set of switches having controller i as slave
$\boldsymbol{G}_{m\times n}$	Inter-plane adjacency matrix, $g_{ij}=1$ if controller i is physically connected to switch j
$\boldsymbol{S}_{m\times n}$	Switch-controller master assignment matrix, $s_{ij}=1$ if controller i currently serves as master for switch j
d_{ij}	Latency between controller i and switch j
$v_{ii'}$	Latency between controllers i and i'
γ_i	Capacity of controller i
L	Maximum allowed load (e.g., 90% of controller's capacity)
ℓ_i	Current load percentage at controller i
$\hat{\ell}_i$	Load percentage at controller i under switch migration
$\bar{\ell}$	Mean load
r_j	Number of `packet-in` messages per second generated by switch j
ϑ_j	Current importance of switch j in its domain (e.g., number of active entries in its flow table — `active_count` field)
$\theta_j^{ii'}$	Cost of migrating switch j from controller i to i'
σ_i	Deviation of the load at controller i from the mean load
ω	Migration competency index
$x_j^{ii'}$	1 if switch j is to be migrated from controller i to i', 0 otherwise

B. FORMULATION

The load of controller i is the sum of all requests originated from the switches in its domain,

$$\ell_i = \frac{1}{\gamma_i}\sum_{j\in\mathcal{J}_i} r_j, \quad \forall i \in \mathcal{I} \tag{1}$$

Under switch migration, the load of i will be the sum of the received requests from the switches in its domain plus the difference between the number of requests of the switches it outmigrates and immigrates (i.e., receives) [28]:

$$\hat{\ell}_i = \frac{1}{\gamma_i}\Big\{\sum_{j\in\mathcal{J}_i} r_j + \sum_{i'\in\mathcal{I}|i\neq i'}\sum_{j\in\mathcal{J}_{i'}} r_j x_j^{i'i} - \sum_{i'\in\mathcal{I}|i\neq i'}\sum_{j\in\mathcal{J}_i} r_j x_j^{ii'}\Big\}, \quad \forall i \in \mathcal{I}. \tag{2}$$

where x is the migration decision variable.

Regarding the cost of migrating switch j from domain $\mathcal{J}_i$ to domain $\mathcal{J}_{i'}$, it can be defined as:

$$\theta_j^{ii'} = (1-\frac{r_j}{\gamma_i}d_{ij}) + \frac{r_j}{\gamma_{i'}}d_{i'j} + \vartheta_j v_{ii'}, \quad \forall i,i' \in \mathcal{I},\ j\in\mathcal{J} \tag{3}$$

In (3), the first term encourages the master controller to migrate switches that have high latencies, the second term encourages the slave controller to receive switches with low latencies, and the last term encourages the master controller to migrate switches that will cause low control plane overhead when the master and slave controllers exchange information about migrated switches.

Let $\bar{\ell}$ be the mean of the loads at controllers, and let the absolute deviation of the load at controller i from the mean load $\bar{\ell}$, be:

$$\sigma_i = \left|\hat{\ell}_i - \bar{\ell}\right|, \quad \forall i \in \mathcal{I}. \tag{4}$$

Having such notation in mind, the SMP can now be formulated as the following mixed integer linear programming (MILP) problem:

$$\text{Minimize (SMP)}: \sum_{i\in\mathcal{I}}\sigma_i + \sum_{i\in\mathcal{I}}\sum_{i'\in\mathcal{I}|i\neq i'}\sum_{j\in\mathcal{J}_i}\theta_j^{ii'}x_j^{ii'} \tag{5}$$

$$\text{Subject to:} \sum_{i\in\mathcal{I}}\sum_{i'\in\mathcal{I}|i\neq i'} x_j^{ii'} \leq 1, \quad \forall j\in\mathcal{J} \tag{6}$$

$$x_j^{ii'} \leq g_{i'j}, \quad \forall i,i'\in\mathcal{I},\ j\in\mathcal{J} \tag{7}$$

$$0 \leq \hat{\ell}_i \leq L \quad \forall i\in\mathcal{I} \tag{8}$$

$$\hat{\ell}_i - \bar{\ell} \leq \sigma_i, \quad \forall i\in\mathcal{I} \tag{9}$$

$$\bar{\ell} - \hat{\ell}_i \leq \sigma_i, \quad \forall i\in\mathcal{I} \tag{10}$$

$$x_j^{ii'} \in \{0,1\}, \quad \sigma_i \in [0,1],\ \forall i,i'\in\mathcal{I},\ j\in\mathcal{J} \tag{11}$$

The first term in the objective function (5) minimizes the absolute deviations from the mean load, and the second term minimizes the switch migration cost. Constraints (6), ensure that a switch can only migrate to a single controller. Constraints (7) restrict migrations to link availability between a switch j and a slave controller i'. Constraints (8) limit the loads of the controllers to the maximum allowed load. Constraints (9) and (10) represent the linearization of (4) [34]. Finally, constraints (11) define the types of the decision variables

In this mathematical programming problem the objective function and constraints are both linear and, therefore, readily solved using MILP solvers. However, this is only efficient for small networks. For this reason, a heuristic approach is presented next.

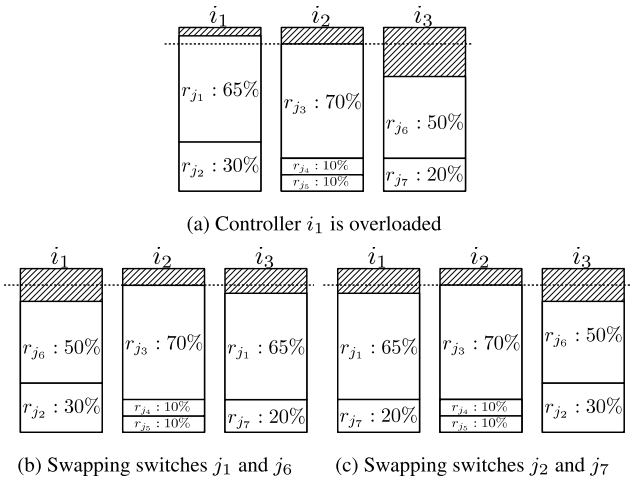
V. A HEURISTIC ALGORITHM

This section presents a heuristic approach inspired by the success of the use of simple moves like *shift* and *swap* to solve generalized assignment problems [35], [36]. The proposed heuristic in this article is called migration competency-based

load balancing (MCBLB). It uses a local search framework but, contrarily to the classical local search procedures, MCBLB can start from an infeasible solution, uses multiple evaluation functions (one for each type of move), and applies a filtering step to the potential moves.

A. MOTIVATION

Heuristics proposed in the literature (e.g., [16], [28]) assume that the shift move (i.e., single migration) is always possible, which is not always the case. "Heavy" switches (usually causing hot-spots) can not be migrated by just shifting them to underloaded controllers, because these switches may overload the target underloaded controllers [23]. Therefore, it is more robust to consider more types of moves when solving SMP. For example, let us consider the scenario in Figure 2 where controller i_1 is overloaded and, at the same time, there is no feasible single migration to neither i_2 nor i_3 to offload i_1. However, an effective move would be to swap switch j_1 with j_6 or j_2 with j_7. The swap choice depends on the migration cost.



(a) Controller i_1 is overloaded

(b) Swapping switches j_1 and j_6 (c) Swapping switches j_2 and j_7

FIGURE 2. A load scenario where the maximum allowed load L is 90%. In this case, there is no single migration (single shift) that will produce a feasible solution. However, when considering a swap move, i_1 can easily be offloaded.

In addition to shift and swap moves, some complex moves like injection-chains have been successfully used to solve problems similar to SMP [35]. However, in the SMP these complex moves can easily result into prohibitive time complexities. Therefore, in this work, only shift and swap moves are considered.

B. MOVES AND EVALUATION FUNCTIONS

When migrating a switch j from controller i to controller i', the value of the objective function in (5) will be affected by the change of σ_i, $\sigma_{i'}$ and $\theta_j^{ii'}$, while the other terms not involved in migrating j will be intact. Therefore, the proposed approach associates a sub-function to each type of move, each sub-function deals with the terms that are affected by the migration of switches.

1) SHIFT

To measure the improvement in the load balancing when migrating j from i to i', the migration competency index (MCI) is defined as:

$$\omega(j,i,i') = \frac{\left(\ell_i - \frac{r_j}{\gamma_i} - \bar{\ell}\right)^2 + \left(\ell_{i'} + \frac{r_j}{\gamma_{i'}} - \bar{\ell}\right)^2}{\left(\ell_i - \bar{\ell}\right)^2 + \left(\ell_{i'} - \bar{\ell}\right)^2}, \quad \ell_i \neq \ell_{i'} \tag{12}$$

MCI in (12) can be interpreted as a measure on how much load balancing improvement will be obtained when applying a move (numerator) compared to the load balancing without applying the move (denominator). Therefore, MCI can be used to filter out the "non-productive" moves, and a switch j is considered for migration from i to i' iff $\omega(j,i,i') < 1$.

The cost function associated with shifting j from i to i' can be defined as:

$$f_{\text{shift}}(j,i,i') = \frac{\omega(j,i,i') + \theta_j^{ii'}}{|\ell_i - \ell_{i'}|}, \quad \omega(j,i,i') < 1 \tag{13}$$

where the denominator is the difference between the loads at the controllers, encouraging the selection of controllers with large load gaps.

2) SWAP

This move is used to migrate a switch j_1 from i to i' and migrate a switch j_2 from i' to i simultaneously. The MCI in this case will be:

$$\omega(j_1,j_2,i,i') = \frac{\left(\ell_i + \frac{r_{j_2} - r_{j_1}}{\gamma_i} - \bar{\ell}\right)^2 + \left(\ell_{i'} + \frac{r_{j_1} - r_{j_2}}{\gamma_{i'}} - \bar{\ell}\right)^2}{\left(\ell_i - \bar{\ell}\right)^2 + \left(\ell_{i'} - \bar{\ell}\right)^2}, \quad \ell_i \neq \ell_{i'} \tag{14}$$

and the evaluation function of swapping switches j_1 and j_2 will be:

$$f_{\text{swap}}(j_1,j_2,i,i') = \frac{\omega(j_1,j_2,i,i') + \theta_{j_1}^{ii'} + \theta_{j_2}^{i'i}}{|\ell_i - \ell_{i'}|}, \quad \omega(j_1,j_2,i,i') < 1 \tag{15}$$

C. SEARCH SCHEME

The proposed search scheme is detailed in Algorithm 1. The input parameters to this algorithm are: the set of controllers, $\mathcal{I}$, the set of switches, $\mathcal{J}$, the switch-controller master assignment matrix, $\boldsymbol{S}$, the maximum allowed load, L, and the maximum number of allowed swap moves, $\text{swaps}_{\max}$. The output is the updated $\boldsymbol{S}$.

The algorithm iteratively improves the load balancing by performing the following two steps:

1) Search for the best shift (Line 10) or swap (Line 15) moves while incorporating a filtering step using MCI and L. The filtering step is described as:

Algorithm 1 - MCBLB

1: *// input:* $\mathcal{I}$, $\mathcal{J}$, $\boldsymbol{S}$, L, $\text{swaps}_{\max}$, $\text{iter}_{\max}$.
2: *// output:* $\boldsymbol{S}$.
3: $\ell_i = \frac{1}{\gamma_i} \sum_{j \in \mathcal{J}_i} r_j, \quad \forall i \in \mathcal{I}$
4: $\text{swaps} = 0$
5: $\text{CoV} = \infty$
6: $\text{iter} = 1$
7: **while** $\text{CoV} > \frac{\sigma}{\bar{\ell}} \wedge \text{iter} \leq \text{iter}_{\max} \wedge \text{swaps} < \text{swaps}_{\max}$ **do**
8: $\quad \text{CoV} = \frac{\sigma}{\bar{\ell}}$
9: $\quad \text{iter} = \text{iter} + 1$
10: $\quad \{j, i, i'\} = \arg\min f_{\text{shift}}(j, i, i'), \forall i, i' \in \mathcal{I} | i \neq i', \forall j \in \mathcal{J}_i | \omega(j, i, i') < 1 \wedge \ell_{i'} + \frac{r_j}{\gamma_{i'}} \leq L$
11: $\quad$ **if** $j \neq \emptyset$ **then**
$\quad\quad$ *//Migration*
12: $\quad\quad s_{ij} = 0; s_{i'j} = 1$
13: $\quad$ **else**
14: $\quad\quad \text{swaps} = \text{swaps} + 1$
15: $\quad\quad \{j_1, j_2, i, i'\} = \arg\min f_{\text{swap}}(j_1, j_2, i, i'), \forall i, i' \in \mathcal{I} | i \neq i', \forall j_1 \in \mathcal{J}_i, \forall j_2 \in \mathcal{J}_{i'} | \omega(j_1, j_2, i, i') < 1 \wedge \ell_i + \frac{r_{j_2} - r_{j_1}}{\gamma_i} \leq L \wedge \ell_{i'} + \frac{r_{j_1} - r_{j_2}}{\gamma_{i'}} \leq L$
16: $\quad\quad$ **if** $j_1 \neq \emptyset$ **then**
$\quad\quad\quad$ *//Migration*
17: $\quad\quad\quad s_{ij_1} = 0; s_{i'j_1} = 1; s_{ij_2} = 1; s_{i'j_2} = 0$
18: $\quad\quad$ **end if**
19: $\quad$ **end if**
20: $\quad \ell_i = \frac{1}{\gamma_i} \sum_{j \in \mathcal{J}_i} r_j, \quad \forall i \in \mathcal{I}$
21: $\quad \sigma = \sqrt{\frac{1}{m} \sum_{i \in \mathcal{I}} (\ell_i - \bar{\ell})^2}$
22: **end while**
23: **return** $\boldsymbol{S}$

- a shift move (Line 10) is included in the search if $\omega(j, i, i') < 1 \wedge \ell_{i'} + \frac{r_j}{\gamma_{i'}} \leq L$;
- a swap move (Line 15) is included in the search if $\omega(j_1, j_2, i, i') < 1 \wedge \ell_i + \frac{r_{j_2} - r_{j_1}}{\gamma_i} \leq L \wedge \ell_{i'} + \frac{r_{j_1} - r_{j_2}}{\gamma_{i'}} \leq L$.

This filtering step will always ensure load balance improvement, whenever a move is selected, while obeying the capacity constraint of controllers.

2) Upon success of a search, the selected move is applied to the solution $\boldsymbol{S}$ in Line 12 for shift moves, and Line 17 for swap moves. Thereafter, the new loads and standard deviation are calculated in Lines 20 and 21, respectively.

The above two steps are repeated until either no load balancing improvement is recorded, measured by the decrement in the coefficient of variation (CoV), or $\text{swaps}_{\max}$ is reached (Line 7).

In this algorithm, in order to allow the user to choose a tradeoff between accuracy and computational cost, the search for shift moves is always allowed, while the search for swap moves is restricted by applying these two concepts:

- Searching for swap moves is allowed only when necessary, i.e., when the search for a shift move fails;
- The number of allowed swap searches is restricted to $\text{swaps}_{\max}$.

The reason behind restricting the search for swap moves is the complexity associated with its exploration.

D. COMPLEXITY

Let $\boldsymbol{S}$ be the current mapping solution. The *shift* neighborhood of $\boldsymbol{S}$ is the set of all solutions such that, any neighbor solution in this set is obtained from $\boldsymbol{S}$ by migrating a switch. The complexity to search all neighbor solutions starting from $\boldsymbol{S}$ is $\mathcal{O}(mn)$. Likewise, the complexity of searching the neighbor solutions of $\boldsymbol{S}$ when swapping two switches is $\mathcal{O}(n^2)$. Therefore, in the worst case the overall algorithm complexity will be $\mathcal{O}(\text{iter}_{\max} \times mn + \text{swap}_{\max} \times n^2)$.

E. IMPLEMENTATION

When implementing the proposed algorithm and in order to speedup the search, the components of the evaluation functions (13) and (15), i.e., θ, MCI, and the denominator, are calculated and stored in lists when the algorithm is called for the first time. In the next calls and upon the success of a switch migration, the entries in these lists corresponding to the controllers and switches affected by the switch migration are recalculated, while the other entries are kept intact.

F. IMPROVEMENT

To improve the solution obtained by Algorithm 1, a post-processing step is applied by shaking the solution. Shaking is a noising technique [37] used to avoid immature solutions. The objective of this procedure is to introduce a random migration in the solution obtained by the MCBLB, and then restart the search again. This improved version of MCBLB will be denoted by MCBLB-shaking.

1) SOLUTION SHAKING

To shake the solution, the following steps are repeated for a given number of iterations:

- Given a probability ρ, select a random switch j and shift it to the controller with the lowest load among all possible slave controllers that j can migrate to;
- Call MCBLB and track the best solution.

Additionally, and in order to control the shaking strength, ρ is updated using [38]:

$$\rho = \frac{\rho_0}{\log k} \tag{16}$$

where ρ_0 is the initial probability and k is the iteration number.

VI. RESULTS

In order to evaluate the proposed work, different versions of the proposed algorithm (MCBLB) are obtained. These are: **MCBLB**, with shift and swap moves; **MCBLB-shift-only**, with shift moves only; and **MCBLB-shaking**, with shift

and swap moves and solution shaking. These versions of the MCBLB algorithm are then compared against two of the most recent state-of-art approaches in the literature (**Elastic** [28] and **OCLB** [29]). In addition, the static mapping and optimal model are also included in the comparison to serve as baselines.

A. EXPERIMENTAL SETUP

A module was developed to create random topologies with predefined node connectivity similar to the ones in [26], [28]. The placement of controllers is determined by solving the classical k-median problem. In each topology, each switch is assured to have one master controller and at least one slave controller. The ratio between the number of controllers and the number of switches was set to $1:4$.

TABLE 2. Simulation parameters.

Parameter	Value
`packet-in` per second	$[5\ \ 500]$
switch-controller latency	$[0.1\ \ 4.8]$ milliseconds [39]
controller-controller latency	$[0.05\ \ 0.26]$ milliseconds [39]
controller capacity	2000 `packet-in` per second
simulation time	100 seconds
maximum allowed load L	95 %
ρ_0	0.15
shaking iterations	3
$\text{iter}_{\max}$	$\frac{n}{2}$
$\text{swaps}_{\max}$	1

The parameter values being adopted in this work are listed in Table 2, and all algorithms and programmings presented in this article were implemented in MATLAB in a computer running Linux with Intel(R) Xeon E3-1200 i7-6700 CPU of 3.40GHz and 15GB RAM.

To create traffic matrices that capture both the spatial and temporal dynamics of network traffics, the following procedure is used:

1) A spatial variation range of $[5\ \ 500]$ `packet-in` messages per second is used to generate a number of `packet-in` messages in each switch. This number of messages can be regarded as a baseline (or mean) of the number of messages generated in a given switch over a time period;
2) To incorporate the temporal variation in the number of generated messages. In each time slot, the baseline number of messages in each switch is varied randomly in the range $[\pm 10\% \ \ \pm 80\%]$ to create load imbalance in the control plane.

To ensure an imbalance state in each time slot, at least one of the controllers must be overloaded (by exceeding the maximum allowed load L) to trigger the switch migration. If no controller is overloaded in the current time slot, step 2 is repeated. Therefore, in each time slot an instance of the SMP is solved.

The evaluation measurements being used are:

- min-max ratio, which is the ratio between the minimum and maximum load of controllers;
- CoV of the load of controllers;
- the running time (in seconds);
- the number of solved instances of the SMP problem.

While the min-max ratio measures the similarity between the lowest and highest loads, the CoV measures the overall variation of the load at controllers.

The experiments are designed to target different aspects:

- Number of slave controllers, to study how the evaluated methods perform under different connectivity levels;
- Number of nodes, to study the scalability of the proposed work;
- Controller failure, to study how the evaluated methods perform under network undesirable conditions like partial failure in the control plane;
- Parameter tuning, to study how the proposed work can be tuned to achieve performance-speed tradeoff.

Each aspect is studied in a separate section, and the main findings and discussion are then presented in Section VI-F.

B. NUMBER OF SLAVE CONTROLLERS

The objective of this test is to analyze the effectiveness of the proposed work under the variation of the number of slave controllers available for each switch. To that end, the number of switches and controllers was set to 20 and 5, respectively, and four scenarios were created. These scenarios consider a specific maximum number of available slave controllers per switch, and for each scenario 10 topologies were created. The distributions of the numbers of available slave controllers in each scenario are shown in Figure 3. The simulation results of this test are shown in Figure 4, and the averages of the evaluation criteria are shown in Table 3.

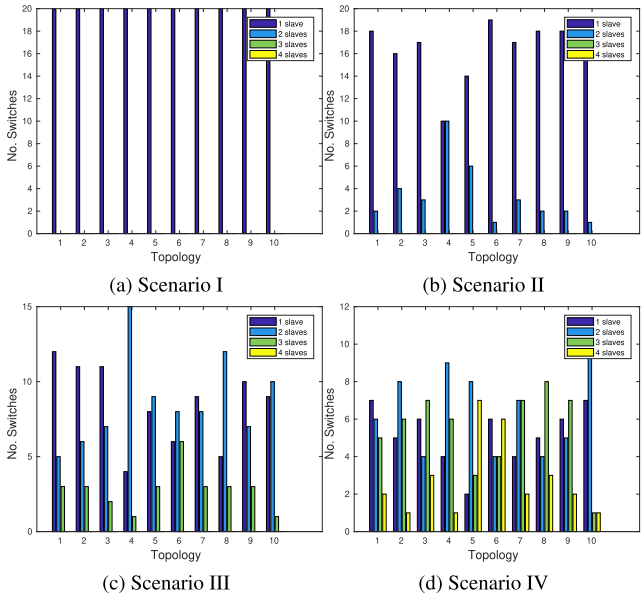


FIGURE 3. The distributions of the available number of slave controllers per switch for four scenarios. The number increases from left to right and top to bottom. Each scenario has 10 topologies and each topology contains 20 switches and 5 controllers. Each bar represents the number of switches that are connected to the color-coded number of slave controllers.

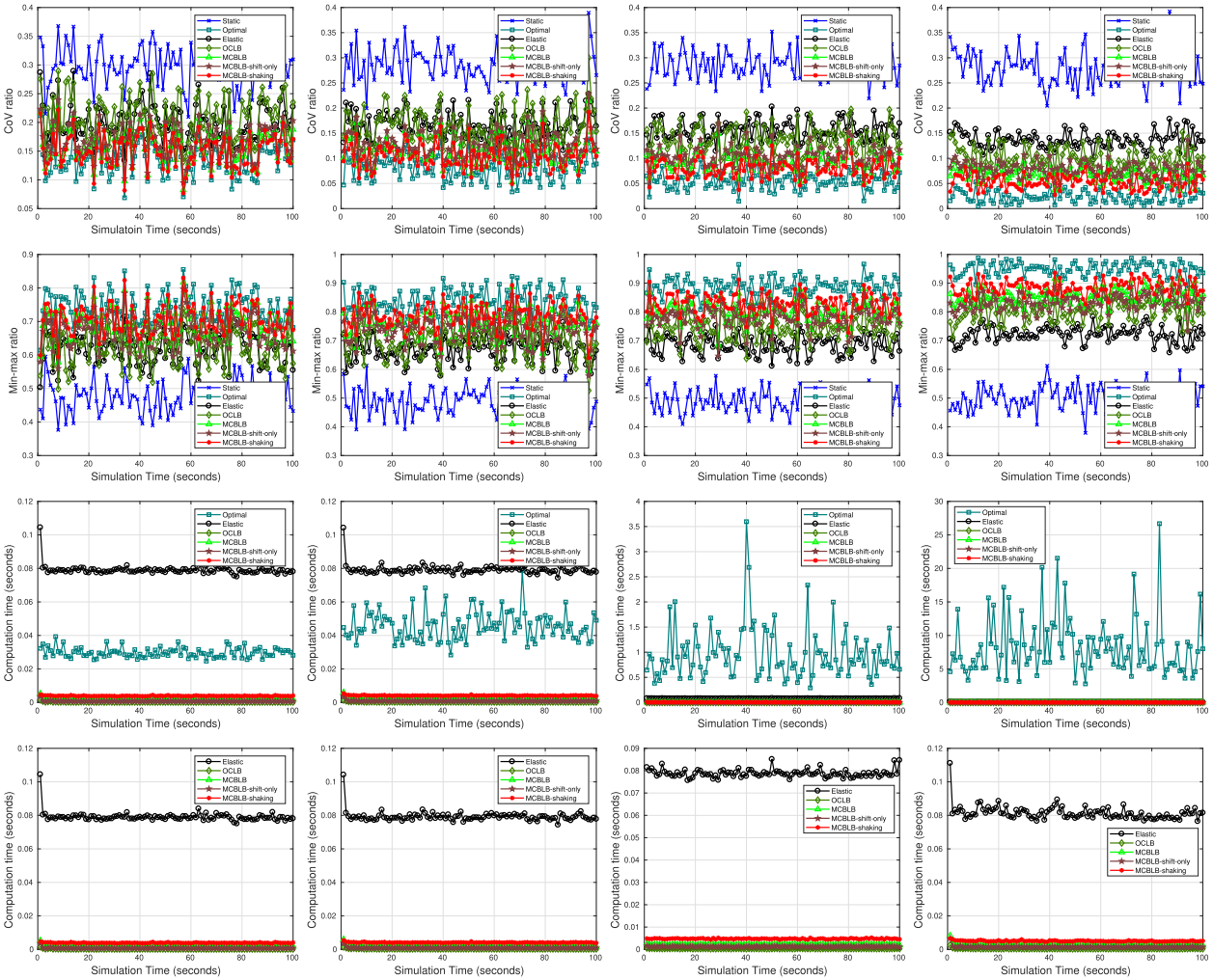


FIGURE 4. Switch migration results for 5 controllers and 20 switches. Each column represents the results of a scenario with the same order as in Figure 3.

TABLE 3. Averaged measurements for the results of 20 × 5 nodes and different numbers of available slave controllers.

Measure	Static	Optimal	Elastic [28]	OCLB [29]	MCBLB-shift-only	MCBLB	MCBLB-shaking
Scenario I							
Min-max	0.4747	0.7439	0.6150	0.6199	0.6745	0.6922	0.7094
CoV	0.2991	0.1356	0.2023	0.2119	0.1668	0.1579	0.1498
Time(s)	N/A	0.0298	0.0790	0.0003	0.0008	0.0017	0.0039
Solved instances	N/A	997	897	964	983	992	994
Scenario II							
Min-max	0.4878	0.8381	0.6659	0.6896	0.7429	0.7619	0.7854
CoV	0.2887	0.0805	0.1695	0.1681	0.1256	0.1159	0.1048
Time(s)	N/A	0.0459	0.0791	0.0004	0.0010	0.0018	0.0041
Solved instances	N/A	1000	912	990	992	996	997
Scenario III							
Min-max	0.4882	0.8894	0.6828	0.7396	0.7830	0.8010	0.8297
CoV	0.2890	0.0538	0.1574	0.1378	0.1028	0.0934	0.0799
Time(s)	N/A	0.9655	0.0787	0.0004	0.0014	0.0022	0.0048
Solved instances	N/A	1000	934	995	997	1000	1000
Scenario IV							
Min-max	0.4978	0.9508	0.7224	0.8013	0.8343	0.8552	0.8877
CoV	0.2814	0.0238	0.1370	0.1007	0.0779	0.0671	0.0516
Time(s)	N/A	8.3210	0.0813	0.0005	0.0015	0.0024	0.0050
Solved instances	N/A	1000	933	988	1000	1000	1000

It is evident from these results that the proposed work outweighs the other approaches in all scenarios in terms of CoV, min-max ratios, and the number of solved instances, and ranked second in terms of computation time. Considering the min-max ratio measurement, MCBLB-shift-only, MCBLB and MCBLB-shaking were able to improve the load

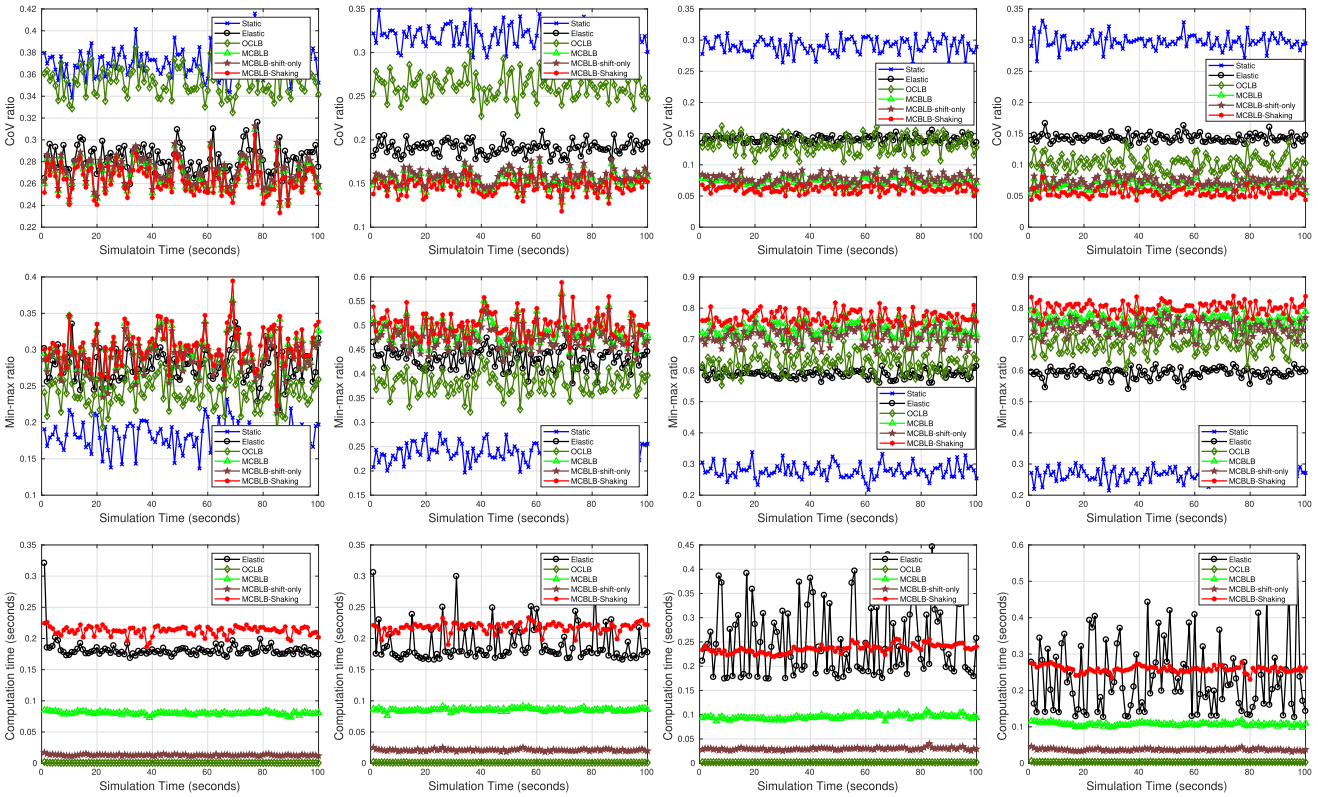


FIGURE 5. Switch migration results for 120 switches and 30 controllers.

balancing, compared to the best second method OCLB, up to $\approx$ 5%, 7% and 9%, respectively. Similar improvements are obtained for the CoV results. Considering the computational time, all heuristics scaled normally with OCLB being the fastest and MCBLB-shift-only ranked the second. Moreover, the proposed search scheme outweighs the other approaches in terms of number of solved instance, which indicates more robustness. We will further analyze this aspect later under controller failure.

Although Elastic took more time than the optimal model in the first two scenarios, the former has shown stable computation times for the other scenarios. It can be noticed that min-max, CoV, and number of solved instances measurements have enhanced as the number of available slave controllers increased. This is due to the fact when a switch has more options, i.e., more slave controllers to migrate to, the evaluated algorithms can perform better since they have larger degree of freedom. However, handling scenarios with high number of available slave controllers per switch requires a higher computational cost because the number of alternative slave controllers to be evaluated per switch increases. We can clearly see this relationship in the results of the optimal method where, in one hand, the min-max ratio improved from 0.7439 in scenario I to 0.9508 in scenario IV but, on the other hand, the computational time jumped from 0.0298 seconds in scenario I to 8.3210 seconds in scenario IV.

C. NUMBER OF NODES

The objective of this test is to extend and validate the previous test by increasing the network size. Since solving the optimal model for large number of nodes is very time consuming, in this test, only the heuristics are evaluated. Therefore, the number of nodes is increased from 20×5 to 120×30 (switches $\times$ controllers) and four scenarios similar to the previous ones (see Figure 3) are created but with this new scale.

The results of this test are shown in Figure 5 and summarized in Table 4. From these results, it is clear that the proposed work produces the best load balancing results, with the MCBLB-shaking being superior to all other approaches. MCBLB-shift-only, MCBLB, and MCBLB-shaking are able to improve the min-max ratio up to $\approx$ 6%, 9% and 14%, respectively. Similar improvements can be noticed in the CoV results. However, this test reveled new results. Regardless of the scenario, our approach improved both min-max and CoV at the same time. Which indicates an efficient shrinking of the small gap between the highest and lowest loads in the network and, at the same time, reducing the load variation among all controllers. On the other hand, Elastic outweighed

TABLE 4. Averaged measurements for the results of 120 × 30 nodes and different scenarios regarding the available number of slave controllers per switch.

Measure	Static	Elastic [28]	OCLB [29]	MCBLB-shift-only	MCBLB	MCBLB-shaking
Scenario I						
Min-max	0.1818	0.2782	0.2407	0.2911	0.2947	0.3036
CoV	0.3694	0.2815	0.3537	0.2710	0.2683	0.2631
Time(s)	N/A	0.1815	0.0015	0.0130	0.0802	0.2123
Solved instances	0	797	852	993	997	998
Scenario II						
Min-max	0.2383	0.4327	0.3829	0.4746	0.4844	0.5016
CoV	0.3188	0.1908	0.2613	0.1573	0.1537	0.1462
Time(s)	N/A	0.1902	0.0017	0.0210	0.0857	0.2179
Solved instances	N/A	948	977	987	1000	1000
Scenario III						
Min-max	0.2782	0.5876	0.6260	0.7089	0.7288	0.7679
CoV	0.2910	0.1411	0.1325	0.0781	0.0729	0.0610
Time(s)	N/A	0.2103	0.0020	0.0296	0.0954	0.2362
Solved instances	N/A	976	994	994	1000	1000
Scenario IV						
Min-max	0.2673	0.5908	0.6829	0.7385	0.7592	0.8012
CoV	0.2979	0.1432	0.1042	0.0735	0.0682	0.0549
Time(s)	N/A	0.2374	0.0034	0.0372	0.1074	0.2587
Solved instances	N/A	984	994	996	1000	1000

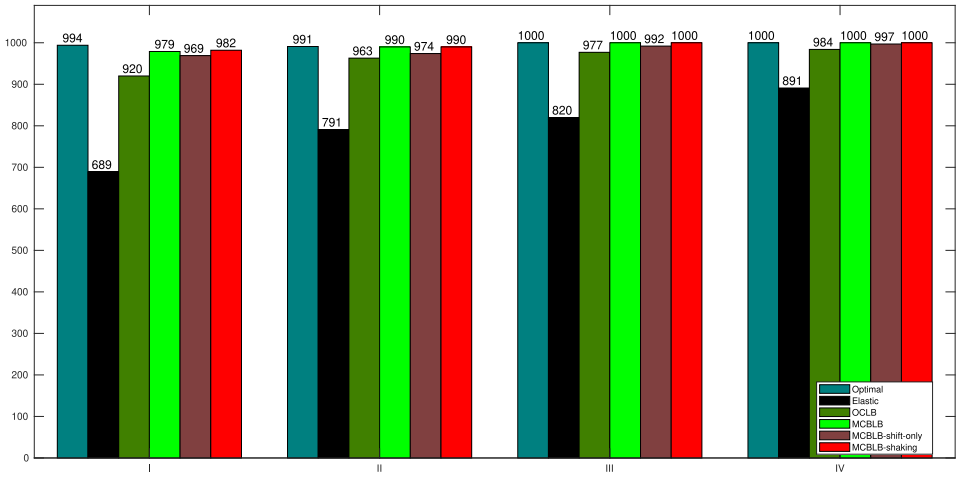


FIGURE 6. The number of solved instances under a controller failure in a network of 20 switches and 5 controllers.

OCLB in the first two scenarios while in the last two scenarios OCLB performed better. Considering the number of solved instances, like the small scale test (Table 3), our approach showed a stable performance. Regarding the computational time, all approaches produced scalable results and, in some scenarios, MCBLB-shaking took longer time than other approaches since it deals with a larger search space and repeats the MCBLB procedure multiple times (see Table 2). In this test, similar to the previous test, the min-max, CoV, and number of solved problems have shown improvement in the results when increasing the number of slave controllers.

D. CONTROLLER FAILURE

Although our main objective in this work is to improve the load balancing results via switch migration, in this test we evaluate the efficiency of the proposed work under controller failure. In this test, the most important measurement is the number of solved instances because it will tell us how efficient a method is in keeping the switches served under a controller failure.

To create failure scenarios, the settings used in Section VI-B are assumed and, in each time slot, a controller is randomly selected and removed from the control plane pool (i.e., deliberately failed). Thereafter, all methods are used to solve SMP and the results are collected. The results of these methods are shown in Figure 6.

This test shows three things. First, our local search scheme is more robust to controller failure compared to the other approaches because it solved more problem instances than the others. Second, the proposed work has shown a stable performance regardless of the available number of slave controllers per switch, where the MCBLB algorithm is able to produce near optimal results. Third, including swap moves is beneficial to solve more problem instances.

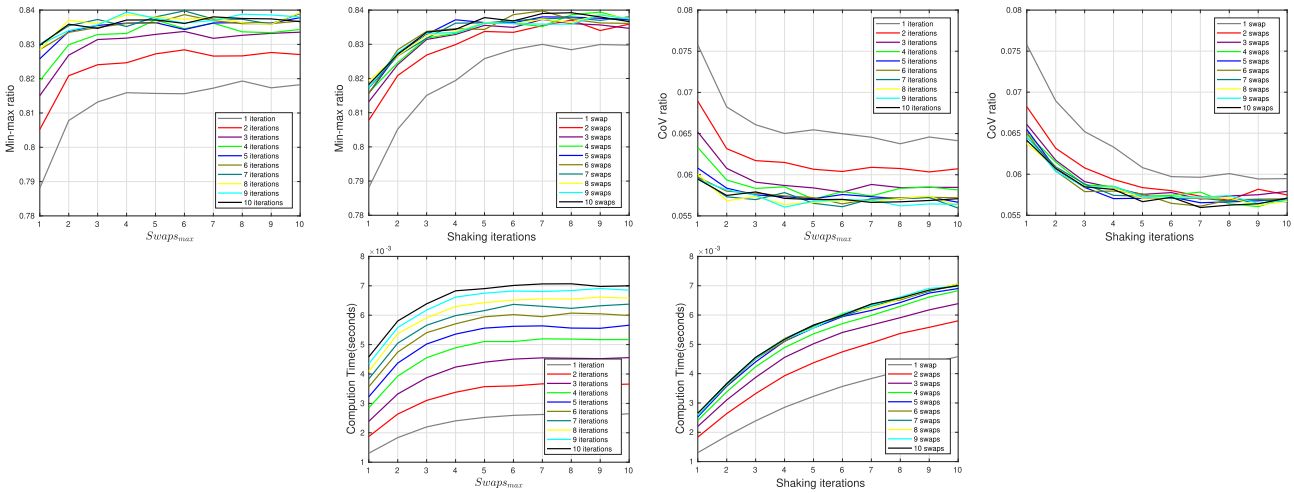


FIGURE 7. Variable number of swaps and shakings and their effect on the accuracy and time on scenario III for a network with 20 switches and 5 controllers.

E. NUMBER OF SWAPS AND SHAKINGS

This test is oriented toward studying the effect of the parameters on the speed-performance tradeoff of the proposed work. This test will help us understand how to tune the shaking iterations and $\text{swaps}_{\text{max}}$ parameters, and determine which parameter is more influential. Moreover, this test will allow us understand to what extent the proposed work can improve the load balancing when more freedom is given to the search scheme.

To design this test, the settings of the third scenario III are used for a network of 20 switches and 5 controllers. The number of $\text{swaps}_{\text{max}}$ and the number of shaking iterations are permuted from 1 to 10.

The results of this test are shown in Figure 7. With respect to the $\text{swaps}_{\text{max}}$, the algorithm performance and computational time saturated after a specific $\text{swaps}_{\text{max}}$. With regard to the number of shakings, the performance of the algorithm enhanced progressively, even slowly as we approach high number of shakings, and the computational time increased linearly with the number of shakings.

F. DISCUSSION

The results of the evaluated methods have shown that the proposed work is able to improve the state-of-art load balancing results up to $\approx$ 14% in some scenarios when using MCBLB-shaking and by $\approx$ 9% when using MCBLB only. When comparing the MCBLB and MCBLB-shift-only results, we can see that including swap moves, besides shift moves, increases the time complexity due to the increase of the search space in one hand but, on the other hand, it allows us to solve more problem instances and improve the load balancing results. Considering the other two approaches, Elastic and OCLB, we can notice that Elastic outperforms OCLB in sparse scenarios (I and II), while OCLB is better in the dense scenarios (III and IV). However, our approach produces stable results in all scenarios.

The results tell us that increasing the number of available slave controllers per switch can improve the load balancing results. However, there is a penalty for computational time. In fact, it is difficult to rely on the current results to judge which network design is the optimal regarding the number of available slave controllers per switch. Since there are many variables and considerations out of the scope of this work, like infrastructure cost, connectivity constraints, bandwidth, and so on. However, these results encourage us to research, in the future, the problem of finding an optimal design and deployment while considering stable network load under dynamic traffic and possible switch migrations.

Due to the fact that the proposed work deals with larger search spaces, especially MCBLB-shaking, the proposed work can take longer time than the other approaches. MCBLB-shaking iteratively repeats the MCBLB procedure after shaking the solution many times and, therefore, requires more time. However, our approach can easily be tuned to find a tradeoff between speed and accuracy. In addition, switch migration for load balancing is in general a second-level task [45], especially when using the four-phase migration protocol [46].

Under the failure of a single controller, the proposed work has shown a stable performance by being able to produce near-optimal results while the other two methods have shown less stable performance in scenarios I and II, i.e., when the number of slave controllers per switch is not high. This is because the proposed work does not stop the search if single move is not possible (i.e., shifting a switch), but it further tries to incorporate more complex moves (i.e., swapping two switches), thus increasing the possibility of finding solutions even under partial failure of the control plane.

To summarize, from these results it is possible to concluded the following. The Elastic method produces the most scalable results regarding computation time because it divides the problem into small problems that are solved via

integer programmings. Therefore it scales nicely. OCLB is the fastest but its performance degrades noticeably when small number of available slave controllers per switch is used, as less available controllers means less options to reassign the switches to controllers. The proposed work is the best among all in terms of efficiency in load balancing and more robust in solving more instances of the SMP problem, especially when considering the controllers failure. In addition, the proposed work can easily be tuned to obtain a speed-performance tradeoff by adjusting its parameters.

VII. CONCLUSIONS AND FUTURE WORK

SMP is a key-issue in dynamic switch-controller assignments in SDN. However, offloading controllers by migrating switches from overloaded to underloaded controllers is not always possible. In this work, a local search algorithm is presented that considers shift and swap moves and incorporates a controlled solution shaking scheme. The results were conducted in terms of load balancing, robustness and computation time. They have demonstrated that the proposed work is able to increase the load balancing by up to $\approx 14\%$ compared to the most recent work. However, this work has also raised some questions which will be handled in the future like: finding the optimal network design considering node placement, connectivity constraints, infrastructure cost, bandwidth, and possible switch migrations. In addition, we will focus on developing distributed and parallel algorithms for solving SMP.

REFERENCES

[1] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *Proc. IEEE GLOBECOM*, St. Louis, MO, USA, Nov./Dec. 2005.

[2] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Commun. Mag.*, vol. 35, no. 1, pp. 80–86, Jan. 1997.

[3] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The softrouter architecture," in *Proc. ACM SIGCOMM Workshop Hot Topics Netw.*, 2004, pp. 1–6.

[4] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Comput. Commu. Rev.*, vol. 35, no. 5, pp. 41–54, Oct. 2005.

[5] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *Proc. SIGCOMM Comput. Commun. Rev.*, New York, NY, USA, 2007, pp. 1–12.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[7] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Comput. Netw.*, vol. 71, pp. 1–30, Oct. 2014.

[8] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[9] *OpenFlow Switch Specification*, ON Foundation, 2011. [Online]. Available: https://www.opennetworking.org/

[10] *OpenFlow Switch Specification*, ON Foundation, Jun. 2012. [Online]. Available: https://www.opennetworking.org/

[11] P. Goransson, C. Black, and T. Culver, *Software Defined Networks: A Comprehensive Approach*. Amsterdam, The Netherlands: Elsevier, 2016.

[12] M. P. Fernandez, "Comparing OpenFlow controller paradigms scalability: Reactive and proactive," in *Proc. IEEE 27th Conf. Adv. Inf. Netw. Appl. (AINA)*, Barcelona, Spain, Mar. 2013, pp. 1009–1016.

[13] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Comput. Commun.*, vol. 67, pp. 1–10, Aug. 2015.

[14] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[15] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Comput. Commu. Rev.*, vol. 43, no. 4, pp. 7–17, Oct. 2013.

[16] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, "A switch migration-based decision-making scheme for balancing load in SDN," *IEEE Access*, vol. 5, pp. 4537–4544, 2017.

[17] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller based software-defined networking: A survey," *IEEE Access*, vol. 6, pp. 15980–15996, 2018.

[18] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *J. Netw. Comput. Appl.*, vol. 103, pp. 101–118, Feb. 2018.

[19] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug. 2014.

[20] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. 2nd USENIX Conf. Hot Topics Manage. Internet, Cloud, Enterprise Netw. Services*. Berkeley, CA, USA: USENIX, 2012, p. 10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228283.2228297

[21] S.-C. Lin, P. Wang, and M. Luo, "Control traffic balancing in software defined networks," *Comput. Net.*, vol. 106, pp. 260–271, Sep. 2016.

[22] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!" in *Proc. 10th ACM Workshop Hot Topics Netw. (HotNets-X)*. New York, NY, USA, 2011, pp. 7:1–7:6. doi: 10.1145/2070562.2070569.

[23] X. Gao, L. Kong, W. Li, W. Liang, Y. Chen, and G. Chen, "Traffic load balancing schemes for devolved controllers in mega data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 572–585, Feb. 2017.

[24] X. Ye, G. Cheng, and X. Luo, "Maximizing SDN control resource utilization via switch migration," *Comput. Net.*, vol. 126, pp. 69–80, Oct. 2017.

[25] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, "Controller placement and flow based dynamic management problem towards SDN," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, London, U.K., Jun. 2015, pp. 363–368.

[26] Y. Chen, Y. Yang, X. Zou, Q. Li, and Y. Jiang, "Adaptive distributed software defined networking," *Comput. Commu.*, vol. 102, pp. 120–129, Apr. 2017.

[27] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *Proc. IEEE 21st Int. Conf. Netw. Protocols (ICNP)*, Gottingen, Germany, Oct. 2013, pp. 1–2.

[28] Y. Zhou, K. Zheng, W. Ni, and R. P. Liu, "Elastic switch migration for control plane load balancing in SDN," *IEEE Access*, vol. 6, pp. 3909–3919, 2018.

[29] S. Zhang, J. Lan, P. Sun, and Y. Jiang, "Online load balancing for distributed control plane in software-defined data center network," *IEEE Access*, vol. 6, pp. 18184–18191, 2018.

[30] W. Lan, F. Li, X. Liu, and Y. Qiu, "A dynamic load balancing mechanism for distributed controllers in software-defined networking," in *Proc. 10th Int. Conf. Measuring Technol. Mechatron. Automat. (ICMTMA)*, Feb. 2018, pp. 259–262.

[31] F. Al-Tam, M. Ashrafi, and N. Correia, "On controllers' utilization in software-defined networking by switch migration," in *Proc. Int. Conf. Broadband Commun., Netw. Syst.* Cham, Switzerland: Springer, 2018.

[32] T. Hu, P. Yi, Z. Guo, J. Lan, and J. Zhang, "Bidirectional matching strategy for multi-controller deployment in distributed software defined networking," *IEEE Access*, vol. 6, pp. 14946–14953, 2018.

[33] T. Wang, F. Liu, and H. Xu, "An efficient online algorithm for dynamic sdn controller assignment in data center networks," *IEEE/ACM Trans. Net.*, vol. 25, no. 5, pp. 2788–2801, Oct. 2017.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.

[35] M. Yagiura, T. Ibaraki, and F. Glover, "An ejection chain approach for the generalized assignment problem," *INFORMS J. Comput.*, vol. 16, no. 2, pp. 133–151, 2004.

[36] H. Gavranović, M. Buljubašić, and E. Demirović, "Variable neighborhood search for Google machine reassignment problem," *Electron. Notes Discrete Math.*, vol. 39, pp. 209–216, Dec. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571065312000297

[37] P. Hansen and N. Mladenović, *An Introduction to Variable Neighborhood Search*. Boston, MA, USA: Springer, 1999, pp. 433–458. doi: 10.1007/978-1-4615-5775-3_30.
[38] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 6, pp. 721–741, Nov. 1984.
[39] T. Mizrahi, E. Saat, and Y. Moses, "Timed consistent network updates in software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3412–3425, Dec. 2016.
[40] F. Al-Tam and N. Correia, "Fractional switch migration in multi-controller software-defined networking," *Comput. Netw.* Amsterdam, The Netherlands: Elsevier, vol. 157, pp. 1–10, Jul. 2019.
[41] W. Guowei, W. Jinlei, M. Obaidat, L. Yao, and K.-F. Hsiao, "Dynamic switch migration with noncooperative game towards control plane scalability in SDN," *Int. J. Commun. Syst.*, vol. 32, no. 7, p. e3927, 2019.
[42] G. Burak, T. Sinan, T. Murat, C. Seyhan, and L. Erhan, "Dynamic control plane for SDN at scale," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2688–2701, Dec. 2018.
[43] M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, and M. Marchese, "BalCon: A distributed elastic SDN control via efficient switch migration," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2017, pp. 40–50. doi: 10.1109/IC2E.2017.33.
[44] Z. Min, Q. Hua, and Z. Jihong, "Dynamic switch migration algorithm with Q-learning towards scalable SDN control plane," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Nanjing, China, Oct. 2017.
[45] K. Anand, P. Shoban, and A. Gember-Jacobson. "Pratyaastha: An efficient elastic distributed SDN control plane," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, New York, NY, USA, 2014, pp. 133–138.
[46] P. Song, Y. Liu, T. Liu, and D. Qian, "Flow stealer: Lightweight load balancing by stealing flows in distributed SDN controllers," *Sci. China Inf. Sci.*, vol. 60, Mar. 2017, Art. no. 032202. doi: 10.1007/s11432-016-0333-0.
[47] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *Proc. 9th Int. Conf. Netw. Service Manag. (CNSM)*, 2013, pp. 18–25. doi: 10.1109/CNSM.2013.6727805.
[48] G. Li, X. Wang, and Z. Zhang, "SDN-based load balancing scheme for multi-controller deployment," *IEEE Access*, vol. 7, pp. 39612–39622, 2019. doi: 10.1109/ACCESS.2019.2906683.

F. AL-TAM received the bachelor's degree in computer science (4 years) from the University of Thamar (THU) in 2004. He received the master's and Ph.D. degrees in computer science from the University of Algarve (UALG) - Portugal, in 2012 and 2016, respectively. He is currently a full-time Researcher with the Networks and Systems Group, Center for Electronics, Optoelectronics and Telecommunications (CEOT), a research center supported by the Portuguese Foundation for Science and Technology (FCT). His major research interests include modeling and optimization problems in image processing and computer networks.

N. CORREIA received the B.Sc. and M.Sc. degrees in computer science from the University of Algarve, in Faro, Portugal, in 1995 and 1998, respectively, and the Ph.D. degree in optical networks (computer science) from the University of Algarve in 2005, and done in collaboration with University College London, U.K. She is a Lecturer with the Science and Technology Faculty, University of Algarve, in Faro, Portugal. Her research interests include the application of optimization techniques to several network design problems, in the optical, wireless, and sensor networks fields, and development of algorithms. She is a Founding Member of the Center for Electronics, Optoelectronics and Telecommunications, University of Algarve, a research center supported by the Portuguese Foundation for Science and Technology. She is also the Networks and Systems Group Coordinator.