Controller Placement in Software-Defined WAN Using Multi Objective Genetic Algorithm

Ahmad Jalili1, Manijeh Keshtgari2 and Vahid Ahmadi3

1,2 Shiraz University of Technology, Department of Computer Engineering & IT, Shiraz, Iran.
3 Shiraz University of Technology, Department of Mathematics, Shiraz, Iran.

Corresponding Author’s E-mail: a.jalili@sutech.ac.ir

Abstract

Software defined network (SDN) is a promising architecture that can overcome the challenges facing traditional networks. In software-defined WAN deployments, multiple controllers are often needed, and the location of these controllers affect various metrics. Since these metrics conflict each other, this problem can be regarded as a multi-objective combinatorial optimization problem (MOCO). A particular efficient method to solve a typical MOCO is to find the actual Pareto frontier first and give it to the decision maker to select the most appropriate solution(s). In small and medium sized combinatorial problems, evaluating the whole search space and find the exact Pareto frontier may be possible in a reasonable time. However, for large scale networks whose search spaces involves thousands of millions of solutions, the exhaustive evaluation needs a considerable amount of computational efforts and memory used. For this, a heuristic approach, called NSGA-II that has been carried out quite well on different discrete and continuous optimization problems is adapted for a new presented multi-objective model of the control placement problem. The results carried out in Matlab 2013b on the internet2 OS3E topology show that this algorithm yields faster computation times and needs much less memory to perform.

Keywords: Multi-objective Combinatorial Optimization, Pareto frontier, SDN, Controller Placement Problem, NSGA-II.

1. Introduction

SDN is a promising architecture that can overcome the challenges facing traditional networks. Unlike traditional networks that both control and data planes are tightly coupled on the same boxes, it decouples control and data planes [10]. Such separation architecture enables administrator/operator to build a simpler, customizable, programmable, and manageable network. In SDN, network owners can dynamically and efficiently configure their network by the external intelligent elements called controllers [14]. Generally speaking, the SDN architecture consists of three distinct layers: data, control and application [1]. In the rest, these layers are introduced respectively. The data layer consists of the network elements and devices that provide packet switching and forwarding, this could also provide functions to collect and report network status. The control layer is responsible for centralized control functionality in logical way which manage the network forwarding trend through an open interface. The application layer consist of the end-user business application that control the switching devices by invoking the services in the control layer. Now, this architecture based on the split plane is facing various challenges that in future architectures of it should be addressed such as reliability, resiliency, scalability, and availability. These issues will become more important when large scale networks are considered.
and have been addressed in the diverse papers. Tootoonchian et al. [17] introduce a distributed control plane naming hyperflow that partition network into several domains, each with a dedicated controller. They strived to investigate the scalability problem while keeping the benefits of network control centralization.

One of the most important issues to address these challenges is the problem of controller placement, i.e., the deployment of a desired number of controllers within a network so that some requirements are satisfied. Determining the number of controllers and their location which also is referred as controller placement problem (CPP) in the literature, is a prerequisite to find answers for performance and fault tolerance questions of SDNs [6] [18].

Heller et al [6] have addressed this problem by focusing on reaction time requirements in wide area networks (WAN), so that the latency between each switch and its assigned controller should be minimized. Although this objective constitutes a crucial aspect in this issue, other objectives shall be considered as well. Stanislav et.al [11] introduce other important objectives in network that play key role in determining the number of required controllers in large scale networks and where they must be fitted. These objectives consist of latency between each switch and its assigned controller, latency between each pair of controllers, load balancing and failure in nodes, links and controllers. Therefore, the controller placement problem is formulated as a multi-objective combinatorial optimization (MOCO) problem in the paper. Using multi-objective approach allows for a clearer illustration of the trade-offs between competing criteria.

Today’s large scale WAN encounter many problems that dynamically change the condition of network [2] [9]. These rapid changes affect the network’s overall performance and administrators need to respond to them on the instant. In order to adapt to these dynamics changes, placements which take into account the current network situation need to be calculated. Based on the fact that controller placement problem is a NP-hard problem, exhaustive evaluation to finding the optimal location of controllers take a magnitude of tens of minutes in large scale networks [12]. Therefore, an exhaustive evaluation not sufficient for these networks to cope with the described dynamics. However, this work introduces heuristic approaches called NSGA-II, to the multi-objective controller placement problem. This algorithm finds a diverse and accurate enough approximation of the Pareto optimal. Comparing with the exhaustive evaluation, this approach may be less precise, but results in faster computation times and needs much less memory to perform. Furthermore, it works efficiently on dealing with conflict objectives in large scale networks whose search spaces include massive number of solutions.

The rest of this paper is organized as follows. In Sect. 2, we will describe the related works. We describe graph theoretic formulations of the replica placement problem in Section 3, and present our proposed adopted NSGA-II algorithm in Section 4. In section 5, evaluation of the algorithms is described. Conclusion is presented in section 6.

2. Related Works

There have been several ongoing research efforts addressed problem of controller placement in Software Defined Networks. Heller et al [6] motivate the controller placement problem and advocates its relevance. It examines the impacts of the controller placement on average and the worst-case propagation latencies for real world topologies. Thus, there is a guarantee for finding optima with respect to the latency. These optima are used to derive guidelines for dimensioning of the control plane. For example, most of the investigated topologies require only one controller to comply with realistic latency constraints.

In [20], Guang Yao et al. enhance the approach of Heller’s work. They present a capacitated controller placement problem (CCPP) that taking into consideration the load of controllers in addition to latency. Authors argue load is critical factor that have to respect in controller placement problem. In this work, it’s assumed that the capacity of each controller is limited base on bandwidth of links, and at the moment, a limited number of switch can be assigned to it. But, there are other critical factors is not
investigated in this paper. Also they claim that their capacitated controller placement (CCP) reduces number of required controllers.

Because of rapid changes in the condition of network a dynamic controller provisioning Problem (DCPP) is introduced by [3]. In this strategy location of controllers change over time depending on the current number of flows in the network. The researchers offer a framework which dynamically aligns the number of controllers activated in the network and assigns each controller to a subset of switches based on network dynamics while guaranteeing minimal flow setup time and communication budget.

The research in [15] investigates optimal controller placement for Software Defined Networks (SDN) and propose a non-zero-sum game based distributed technique. Using the proposed algorithms a controller can validate its value in the design and decide after comparing with the other controllers in the network whether controllers should be added, moved, or deleted. Nevertheless, they do not consider inter controller latency and switch-controller latency.

A Fault Tolerant Controller Placement problem is introduced in Ros et al. [16]. They develop a heuristic algorithm that computes placements with at least reliability. They heuristically search for the minimum number of controllers assigned to each node and the controllers’ placement to reach a certain reliability threshold as, e.g., “five nines”.

Hock et al. [7] proposed a novel controller placement in software defined networks to improve resiliency and failure tolerance. They studied several performance and resilience metrics. They inferred, there is a trade-off between these metrics, and usually no single best controller placement solution. Also a framework based on Pareto-optimal controller placement is presented. In this framework, an exhaustive evaluation of all possible placements has been performed for some topologies. But, they not utilize heuristic approaches.

In [8] authors announce that multiple controllers are often required for wide-area SDN deployments. The conclusion was finding an efficient controller placement algorithm is of great importance. They introduce and compare different heuristic approaches to improve the resilience of software defined networks against connection failures between nodes and controllers.

All the mentioned studies concentrate only either on resilience as opposed to network failures or node to controller latencies and do not consider any additional metrics such as load balancing or controller to controller latencies. In particular, the trade-off between their metrics and other objectives is not addressed. Furthermore, in comparison with the exhaustive evaluation, a heuristic method called PSA is offered that may be less precise, but results in faster computation times and requires much less memory to execute.

3. Problem Definition

The controller placement problem includes location of controllers with respect to the available network topology and the number of controllers needed. The user may be defined various metrics that control the placement of the controller in a network. While minimizing latencies between each node and its assigned controller constitutes a crucial aspect of the controller placement problem, there are numerous other, possibly competing, objectives that require consideration such as node or link failure, controller to controller latency and load balancing.

As discussed in the section 1, we consider the objectives that first introduced in [11]. In this paper, these metrics have been used to constitute a multi-objective controller placement problem. In the following, an outline of these metrics are presented. For a better understanding of the reader, we recommend to refer to [7] and [11]. Afterwards, the problem has been solved by our effective proposed heuristic algorithm.

We indicate the network by \( G = (V, E) \), where \( V \) and \( E \) show the set of switches (or routers) and the set of physical links among the switches, respectively. A distance matrix \( D \) involves shortest path latencies among each pair of switches, where its entries, \( d_{ij} \), indicates the latency between node \( i \) to node \( j \). For the purpose of normalization, latencies recorded in matrix \( D \) are divided in the graph’s
diameter, results in \( d_{ij} \in [0, 1] \). Given the favorite number of controllers, \( k \), a finite set of probable placements is achieved, and concludes the statement combinatorial optimization. MOCO has a main goal to obtain placements involve controllers with size \( k \) from the \( k \)-element placement set \( P_k = \{ P \in 2^V \mid |P| = k \} \) which are Pareto optimal in terms of several, may be conflict, objective functions \( f(i \in \{ 1, \ldots, J \}) \). We regard a placement \( x \) as Pareto optimal, if an only if there not exists any better placement \( y \), i.e. \( \forall i \ f(i(y)) \leq f(i(x)) \) and \( f(i(y)) < f(i(x)) \) for at least a simple index \( i \). All Pareto optimal placements constitute a set called Pareto frontier.

The node to controller latency, as the first objective, gives special information about the relationship between each node and its assigned controller. Latency metrics can be considered as average and the maximum. For each solution, or placement, \( P \in 2^V \) and the predefined distance matrix \( D \), the equations (1), (2) represent the maximum node to controller latency \( \pi_{\text{L}_{\text{max}}N_{2}C}(P) \) and the average node to controller latency \( \pi_{\text{L}_{\text{avg}}N_{2}C}(P) \), respectively.

\[
\pi_{\text{L}_{\text{max}}N_{2}C}(P) = \max_{v \in V} \min_{p \in P} d_{v,p}
\]

(1)

\[
\pi_{\text{L}_{\text{avg}}N_{2}C}(P) = \frac{1}{|V|} \sum_{v \in V} \min_{p \in P} d_{v,p}
\]

(2)

Discussing in a similar way, the equations (3), (4) can be used to calculate the controller to controller latencies either in terms of the maximum or average latency, respectively.

\[
\pi_{\text{L}_{\text{max}}C_{2}C}(P) = \max_{p_1,p_2 \in P} d_{p_1,p_2}
\]

(3)

\[
\pi_{\text{L}_{\text{avg}}C_{2}C}(P) = \frac{1}{|P|} \sum_{p_1,p_2 \in P} d_{p_1,p_2}
\]

(4)

When the reliability property of a typical network is important, load balancing needs also to be taken into account. For the compatibility with the described problem definition and metrics, a metric called imbalance is offered instead of a balance metric such that the objective is to minimize this metric’s value. For any placement \( P \) and controller \( p \) belonged to \( P \), the overall number of switches dedicated to \( p \) when each node links to its nearest controller is identified as \( n_p \). The mentioned imbalance metric, \( \pi_{\text{imbalance}} \), records the difference in \( n_p \) for the two controllers with the smallest and greatest number of dedicated switches, respectively. The imbalance metric is defined in equation (5). Moreover, for normalization purpose, the imbalance metric can be divided by \( |V| \) and this is regarded as the maximum number of nodes which can be assigned to a controller in the worst case.

\[
\pi_{\text{imbalance}}(P) = \max_{p \in P} n_p - \min_{p \in P} n_p
\]

(5)

As our example, The Internet2 OS3E topology is used that is described below. Fig. 1 demonstrates this topology with \( k = 5 \) controllers. In Fig. 1(a), the imbalance aspect of the latency-optimal placement is shown. As can be seen, each node is colored and shaped based on the controller to which is assigned. Although the blue controller is in charge of eleven network switches, the green and red controllers require to manage only five switches. Furthermore, in [4], it is stated that for some controller implementations, the order and number of linked nodes may cause unfairness in terms of facets like the switches’ flow setup times. Hence, when it comes to selecting a controller placement for several types of SDN, load balancing should be considered as an important part of the decision criteria.

Furthermore, when several controllers are needed for topology to meet scalability and fault tolerance requirements, the architecture also needs various forms of state synchronization between the individual controllers.
This assures proper functionality when it comes to outages and permits making decisions which are not restricted to just a local perspective of a part of the network. Hence, another objective of the controller placement problem is to maintain a small controller to controller latency in order to minimize synchronization times. A visualization of this metric is illustrated in Fig. 1(b), where each controller is colored based on the distance to the controller from which it is farthest away. Similar to Fig. 2(a), the distances are normalized in terms of the graph’s diameter. For most controllers, the depicted placement leads to high maximum latencies to the other controllers that may be not admissible for certain use cases.

![Figure 1](image.png)

**Figure 1:** POCO GUI displaying the current placement with the latency to the controller at the top and the whole solution space at the bottom. Extracted from [11].

The Pareto frontier of all probable placements, which indeed is the search space illustrated in the figure by circles, in terms of the imbalance (x-axis) and inter-controller latencies (y-axis) is demonstrated. In order to make a better distinction, the real Pareto frontier is linked via line segments. Fig. 1(c) depicts the search space in terms of the two mentioned metrics with each placement is illustrated via circles.

### 4. The Adapted NSGA-II

Author Evaluation of all possible placements for a particular network topology and predefined number of controllers to find the Pareto optimal placements with respect to some user defined metrics is very exhaustive, time-consuming, and in many cases, exceeds the RAM of the used machine. The search space of this problem consists of all $k$-combinations of $n$ potential nodes, i.e. to seek for the best placement for $k$ controllers of $n$-node graphs, with the total number of \( \binom{n}{k} \) placements. Even for relatively small $n$, this number increases dramatically with a rise in the number of controllers, $k$; for example moving from \( \binom{34}{3} = 5984 \) to \( \binom{34}{10} = 131,128,140 \) placements. Hence, by rising either the size of the network or the number of desired controllers, the time and memory required for the exhaustive evolution will surge dramatically.

When it comes to performing just a single network planning task before deployment, an exhaustive evaluation also for bigger instances is justified even if it requires a high computational effort and a large amount of time. However, in the context of a dynamic and flexible network which requires to adapt to changes in the environment and usage patterns, time is a critical limiting factor.
Heuristic algorithms are very common in coupling with these difficulties of exploring the whole search space in combinatorial optimization. Such strategies often explore only a small subset of the search space and return the Pareto frontier. Finding optimal solutions in single objective optimization problems is a straightforward task for the decision maker since in each part of the mechanism used, they can evaluate the obtained solutions with respect to just one metric. These mechanisms try to improve the objective in each iteration of the algorithm.

However, in the context of multi-objective combinatorial optimization, this trend is not such straightforward. Each solution should be accessed with respect to more than one metric in the process. The situation becomes worsen when it comes to conflict objectives. For instance, considering just controller to controller latency imposes the closeness of positions of controllers in order to reduce the delays among them for better data transmission. However, this compactness of a placement rises the delay among each controller and its assigned switches and so, deteriorates the node to controller metrics. These kinds of metrics are looking for a high degree of dispersion in controller sites. Conflicting objectives in one hand and exploring only a small piece of the large search space on the other hand, makes is essential to design and implement a very effective heuristic to obtain a diverse set of optimal or near optimal solutions. Deb et al. [5] introduced a fast and elitist multi-objective algorithm, called NSGA-II. This procedure benefits from some concepts like fast non-dominated sorting, crowding distance, crowded-comparison operator, and elitism mechanism. NSGA-II was very efficient population-based procedure on the tests on which carried out [13] [19]. Elitism through non-dominated sorting the population and diversification by using the crowding distance mechanism made the algorithm very powerful to find the Pareto front of multi-objective continuous problems even with non-convex and non-connective Pareto optimal fronts. In this paper, inheriting the main concepts of this effective algorithm, we adapt it to solve the controller placement problem. This algorithm has been successfully applied to many optimization problems, like [4], [21].

The reason behind selecting this approach is twofold. First, we aim to find an accurate estimation of the Pareto optimal front. Second, a diverse approximation set to present the decision maker is desired. The NSGA-II presented in this work, adapts the original algorithm to perform efficiently on the controller placement problem in SDN architecture even for large instances. The pseudo code for this procedure is presented in the following. Firstly, we should note that the solution representation used in this algorithm is a k-vector of node numbers as a typical controller placement.

**Algorithm 1: NSGA-II**

1. **Input**: G=(V,E), NP, k, MaxIt, pc, pm
2. n = |V|, nc = pc * NP, nm = pm * NP;
3. pop = GenRandPlacements (n, NP, k);
4. [pop, F] = NonDomSorting (pop);
5. pop = CrowdingDistance (pop, F);
6. iteration = 1;
7. while (iteration ≤ MaxIt ) do
   8. popc = Cross (pop, n, k, nc);
   9. popm = Mutation (pop, n, k, nm);
10. pop = [pop, popc];
11. pop = [pop, popm];
12. [pop, F] = NonDomSorting (pop);
13. pop = CrowdingDistance (pop, F);
14. pop = SortPopulation (pop);
15. pop = pop (1 : NP);
16. [pop, F] = NonDomSorting (pop);
17. pop = CrowdingDistance (pop, F);
18. iteration = iteration + 1;
19. end while
20. **Output**: M;

Ahmad Jalili et al. / Vol. 5(18) Special Issue, Dec. 2015, pp. 2655-2663
The input consists of two parts. The problem specific data like the topology graph \( G \) with \( V \) and \( E \) represent the set of nodes and edges in the graph, respectively, and the desired number of controllers \( k \) is the first part (1). Some parameters of the algorithm include the number of placements in the population \( NP \), the probability of crossover and mutation is depicted by \( pc \) and \( pm \) respectively. \( MaxIt \) determines the maximum number of iterations (2). As can be seen in Algorithm1, the mechanism starts with generating an initial population of \( NP \) random placements, i.e. each one is a \( k \)-combination of \( n \) nodes (3).

It should be noted here that generating random numbers is done by using function `rand()` in Matlab 2013(b). Then, the population is evaluated with respect to the objectives used. Non-dominated sorting and calculating the crowding distance is done in the next phase, (4), (5), (5). Hence, the population is classified into different classes of solutions. \( F \) is the collection of sets involving the indices number of placements in each class. Crossover (8) gets two placements as its arguments, or parents, which are obtained after applying a Triple-Tournament-Selection strategy two times. Each time, three random solutions are selected from the population and the best one is returned by applying the crowded-comparison operator [5]. After the parents identified, the crossover looks for non-dominated solutions with respect to them in a way.

A Cross-Controllers-Operator is implemented on the parent placements to create two new children. The same nodes in both parent are kept and convey to the both children. By doing this, we hope that the created children inherit the same characteristics of their children, as can be seen in the nature. Hence, it is regarded as a respectful crossover. This trend is repeated \( k/2 \) times to create \( k \) children. The mixed set including the solutions produced by the Cross-Controllers-Operator is then purified by removing the dominated solutions. Mutation operator (9) works as follows. Firstly, the Triple-Tournament-Selection is used to select a parent placement. Then, a random number in the interval \([1, k/2]\), is generated, where, \([k/2]\) represents the integer part of \( k/2 \).This number is used as the number of positions for which the parent’s placement values should be changed [11]. This modification allows for more diversity in the created children.

Next, all the selected positions are replaced with random nodes and a child is generated. These children of selected parents during the crossover and mutation strategies are added to population (10), (11). Lines (12) and (13) again perform nod-dominated sorting and calculate the crowding distance. Sorting the population (14) is done in the next step to identify different fronts of the population [5]. The first best NP solutions are passed to make the next generation of the population (15). Non-dominated sorting and crowding distance mechanism is applied to the new population to make it ready for the next generation computations (16) and (17). The algorithm returns the first front in the population. By using equation (9), we can set an upper bound for the number of distinct placements which are evaluated during this method.

\[
MEP = (n\text{cross} \times k + n\text{mute}) \times MaxIt, \tag{6}
\]

Maximum Evaluated Placements (MEP) involves up to \( n\text{cross} \times k \) distinct placements created by crossover operator, i.e. \( n\text{cross} \) is the number of solutions attend the crossover procedure, \( k \) children is generated from Cross Controllers process. The number of placements participate in mutation is \( n\text{mute} \) and in each run of this operator, one child is created.

**5. Evaluation**

Figure 2 demonstrates the aforementioned mechanism by providing different perspectives on a single run of the NSGA-II on an Intel Core i5 CPU at 2.5 GH and 4 GB RAM running Windows 8.1 and Matlab version 2013b. The main purpose is to find appropriate placements of size \( k = 6 \) for the Internet2 OS3E network topology. For setting a reference data to assess our findings, the POCO framework [11] is used to do an exhaustive search. As described before, this topology includes \( n = 34 \) nodes. We set the parameters of the NSGA-II as follows:
\[ NP = 10, \quad Maxt = 200, \quad pc = 0.4, \quad pm = 0.5. \]

By using equation (6), at most 5,800 distinct placements are evaluated during the optimization mechanism which involves only \( \frac{5,800}{\binom{10}{6}} = 0.43\% \) of the whole search space that is a considerable tiny percentage. This number is calculated as follows:

\[ \text{MEP} = (4*6+5)*200 = 5,800. \]

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{The whole search space of placement in a two dimensional plot, along imbalance metric in the x-axis and maximum controller to controller latency in the y-axis.}
\end{figure}

The half top of the figure shows how the estimated Pareto frontier is developed by progressing the algorithm in a two-dimensional plot by taking a subset of two metrics, \( \pi^{im\text{balance}} \) on the x-axis and \( \pi^{L_{\text{max, C2C}}} \) on the y-axis together with all the placements of the search space. Plot captions show the relative progress of the NSGA-II. As can be observed from the plots, after the first iteration of the algorithm, the random generated solutions are located in the middle of the space. After passing only 10\% of the mechanism’s runtime, a remarkable progress is made towards the actual Pareto, down of the figure. After 50\% of the approach, it is clear that our algorithm imposes a good diversification on the obtained solutions while improving their qualities. At the end of the process, a good approximation of the Pareto frontier is achieved.

**Conclusion**

In an SDN-based network, finding appropriate locations for controllers are critically important to address many challenges faced by this state-of-the-art architecture. Although solving the controller placement problem can lead to address the challenges, the presence of competing objectives in the resulted multi-objective problem introduces several difficulties to decision makers. This work addresses the controller placement problem with respect to various important metrics. These involve latency between nodes and its assigned controllers, among controllers themselves, and load balancing. The decision maker should have a good view of the trade-off between these competing metrics to make a reasonable choice. The NSGA-II is adapted which proved to be efficient for obtaining a diverse approximation set of the Pareto Optimal front considering these objectives. In this paper the advantages
of this heuristic method as opposed to the exhaustive evaluation of the whole search space is investigated.

References


