Multi Criteria Analysis of Controller Placement Problem in Software Defined Networks

Ahmad Jalili , Manijeh Keshtgari , Reza Akbari , Reza Javidan

PII: S0140-3664(18)30004-5
DOI: https://doi.org/10.1016/j.comcom.2018.08.003
Reference: COMCOM 5759

To appear in: *Computer Communications*

Received date: 2 January 2018
Revised date: 27 July 2018
Accepted date: 5 August 2018

Multi Criteria Analysis of Controller Placement Problem in Software Defined Networks

Ahmad Jalili , Manijeh Keshtgari, Reza Akbari, and Reza Javidan*

Department of Computer Engineering and IT, Shiraz University of Technology, Shiraz, Iran
Corresponding Author Email: javidan@sutech.ac.ir

***Abstract*— Controller Placement Problem (CPP) is an important task in Software Defined Networks (SDNs). Meanwhile, the assignment of switches to the controllers plays a key role in the Quality of Service (QoS). Most of previous works assumed that each switch is assigned to the nearest controller based on propagation latency. In this paper, propagation delay, hop count and link utilization are considered for such assignment. First, the influence of these metrics on the QoS is analyzed. Then, using these criteria, a new Analytic Hierarchy Process (AHP) technique is presented to perform the multi criteria assignment process. Several evaluations are conducted to analyze this approach and justify the importance of trade-off among all the assignment criteria. Furthermore, an algorithm called Controller Placement Genetic Algorithm (CPGA) that is hybridized by the proposed AHP technique is introduced to solve this special kind of CPP. Results show the efficiency of our new assignment approach regarding link load balancing problem.**

***Index Terms*—Software Defined Network, Controller Placement Problem, Genetic algorithm, Wide Area Networks, Analytical models, Analytic Hierarchy Process.**

I. Introduction

Software Defined Networking (SDN) provides communication networks with flexible operation and programmability in which the management as well as the operation of a network is performed through a controller having a global view of the network [1, 2]. Nonetheless, the control plane not necessarily represents a single and centralized device; but it may comprise of multiple controllers responsible for controlling various administrative domains of the network or different parts of the flow space. Many reasons for using a distributed control plane can be summarized as [3, 4]: (1) administration; (2) SDN providers with geographically local footprints; (3) scalability; (4) decreasing the latency between a switch and its closest controller; (5) fault-tolerance; and (6) load-balancing. In addition to administration and fault tolerance, efficiency also provides an important motivation.

It is necessary to overcome a network of distributed controllers; each of the controllers may have a partial view of the network, and they have to cooperate and exchange information together [5]. However, partitioning network into multiple control domains opens many challenges such as latency, reliability, and load balancing [6]. Given the real assumptions, there are many critical questions to answer including how to identify the minimum number of required controllers, how to partition the network into several small SDN domains, and where the controller location in each SDN domain is placed [7]. These issues, in turn, make the problem of deploying multiple controllers to manage a network and enforce policies in such a distributed control plane as a hot research issue in SDN [8]. This problem has been investigated in many papers as the Controller Placement Problem (CPP) [3, 6, 7].

Earlier studies showed different types of contributions to cope with SDN challenges. However, there are still challenges in solving the controller placement problem. Some articles in this area have not been very much involved in the assignment of switches to controllers. Indeed, they usually implicitly assumed that during the location finding of controllers, each switch is assigned to the closest controller, based on propagation delay [3, 10, 12]. Furthermore, none of the papers which considered load and capacity of the controllers has paid attention to load of paths and the amount of loads imposed on each link [16, 17]. It is clear that the more amount of load, the more delay and failure probability of links and connected nodes will be generated. Generally, an in-depth quantitative investigation of the assignment has received limited attention in the literature; i.e. how the controllers are assigned to switches and what paths should be selected to connect them. There are still many unanswered questions regarding assignment of switches in related papers.

This paper investigates the control placement problem so that an appropriate location of controllers is found. During the search in the search space of placements, two important aspects of this problem are also considered: Given a particular placement in the space which the algorithm faces, two mechanisms are developed to find the best possible assignment of switches to the considered controllers as well as to select the best paths for establishing these connections. Regarding the importance of the allocation of controllers to switches as well as the paths through which they are connected (assignment process), we aim at looking the controller placement problem in SDN not only as a location but also as a *location-allocation problem*.

For this purpose, fundamental parameters that are very effective in this regard are examined. These factors include switch to controller propagation latency, the number of hop count between switches and controllers and the amount of link utilization for control traffic in assigned path. Since many papers in the literature considered the propagation delay and this parameter affects the availability and convergence time in the large-scale networks, we also consider it as one of the most important parameters. The second important metric is hop count which is defined as the number of hops between the switches and the controllers. Since any hop incurs store and forward and other latencies, this metric may be a good estimation of distance between the switch and the controller. The final metric which is considered in our work is link utilization. This metric is related to the quality of assigned paths. It is simply the amount that links are being used by only control traffic.

In order to apply these criteria in the allocation of switches to controllers and also control paths to each pair switch-controller, a novel multi-criteria assignment approach which works based on Analytic Hierarchy Process (AHP) technique is designed. AHP technique is a smart choice to apply these criteria since it is flexible to impose the preferences of the decision maker to find the solution [20]. Using this technique, the impact of the important criteria is efficiently imposed on the placement of controllers. This method is adapted well to the considered controller placement problem for assigning not only switches to controllers but also control paths for these connections. By this multi-criteria technique and selecting the weight preferences, our proposed AHP-based assignment algorithm generalizes the classic assignment techniques of papers in the literature. Some techniques which introduced in [20], and [21] are adapted for this assignment.

To solve the controller placement problem with the new assignment (location-allocation model), a very efficient algorithm which is inherited from Genetic Algorithm is developed. The good design and adaptation of this strong algorithm to our underlying problem was the main reason for selecting it as the base of our algorithm. The results confirm this algorithmic choice. The Genetic Algorithm works as the master algorithm and other procedures and algorithms such as the AHP-based assignment algorithm are called by the master algorithm whenever they are needed.

In order to evaluate the performance of our proposed method, various evaluation experiments are carried out. First, the performance of the AHP-based algorithm with respect to different weights has been analyzed. The goal of this evaluation is to analyze the importance of different influential factors on the assignment of switches to controllers. Then, to show that the proposed model has a great deal of efficiency in locating a problem and provides better quality solutions, we have used two important metrics to evaluate our model, named "link load indicator" and "maximum link utilization of control paths", or briefly "$LL(l)$" and "$MLUCP$", respectively. The $MLUCP$ and $LL(l)$ metrics are important because they consider the load balancing on the links [22, 23]. The lower the value of these metrics, the better the load on the links is achieved. Several papers on the subject of networks and data centers seek to minimize these metrics [22, 24]. We have used this metric to evaluate the quality of our solution and we have shown that the proposed method has a better load balancing than other methods.

To the best of our knowledge, this is the first work to consider the effect of three parameters (propagation latency, hop count, link utilization) on optimizing location of controllers and offer optimal paths for assigning switches to the controllers in solving CPP. For this, a new AHP based algorithm is introduced to perform a new multi-criteria assignment. Here, we discuss how other criteria, in addition to propagation latency, can be effective on CPP problem and start assignment process as a new important problem in CPP. Furthermore, we propose Controller Placement Genetic Algorithm (CPGA) to solve the controller location-allocation problem.

The rest of the paper is outlined as following. Section II presents the literature review of the discussion. Section III describes the roles of important different criteria in assignment process. A multi-criteria AHP based algorithm with the proposed algorithm to solve a special kind of controller placement problem is proposed in Section IV. Several results of simulations are presented in Section V. The final section is dedicated to the conclusion.

II. Related works

The controller placement problem has been considered by many researchers in recent years. Usually, in controller placement literature, CPP is considered as K-median or K-center problems aiming at minimizing the average case or the worst-case latencies, respectively [9, 10, 11]. Several alternatives which go beyond the simple k-median or k-center alike problems have been studied in recent years. Given a topology, the CPP needs to find the number and the locations of required controllers while minimizing the cost associated with the placement. This cost can be expressed in terms of the number of controllers [12], the switch-controller communication latency [10], the synchronization time of controllers [13], or a combination of more than one of these metrics [3, 14, 15] (as a multi-objective optimization problem). In addition, most of the controller placement papers take somehow load issue into consideration as the most important parameter [16, 17]. To do this, a specific capacity is considered for each controller with load balancing among different controllers. If the load of each controller exceeds a threshold, the reassignment of switches to the other controllers will be occurred [7, 18, 19].

Heller et al. [9] established the first study in this area. Their model developed based on the mathematical k-median problem. The authors examined the effects of placements on average-latency and worst-case latencies on real topologies.

This paper focused on the number of controllers are needed as well as their location. However, the assignment and the importance of it has not been considered.

Yao et al. [25] extended the problem by considering restricted capacity of network elements. They discussed a capacitated controller placement problem, with the objective of minimizing the worst delay of the control paths subject to satisfying the load constraint of controllers. While their effort was to balance the load on the controllers, link load balancing and the assignment paths were not considered in this paper.

In [12], regarding the capacity of the controllers and delay of links, the authors provided a mathematical model for constructing a SDN with the lowest cost. Their model allowed one to simultaneously determine the number, the type, and the location of the controllers. In this paper, they simply considered the assignment procedure. The initial assignment is only based on the propagation delay, and they did not address hop count and link utilization.

Xiao et al. [26] provided a challenge by focusing on two specific questions: how to partition a wide-area network topology into several small SDN domains and the way that the controller should go in each SDN domain. They aimed at maximizing the reliability of controller as well as minimizing the latency of Wide Area Network (WAN) using spectral clustering placement algorithm. However, in their clustering approaches, the importance of assignment process and the path between control plane and data plane have not been considered.

Some papers formulate the controller placement problem as a Multi-Objective Combinatorial Optimization (MOCO) problem and some important objectives have been proposed [3, 14, 15]. In [15], other important objectives in network which play key roles in deciding the location of controllers are proposed. These objectives comprise of latency between each switch and its assigned controller, latency between each pair of controllers, and load balancing among the controllers. Jalili et al. provided a multi-objectives genetic algorithm–based solution for CPP [14]. They have proposed the heuristics based NSGA-II to solve the controller placement problem. In [27], authors developed a specialized heuristic to optimize same objectives called Pareto Capacitated k-Medoids (PCKM). They investigated PCKM, by considering a particular set of optimization objectives and returning solutions representing the possible trade-offs between them. New multi-objective algorithms and location of controllers have been evaluated based on these objectives. The main challenge of these multi-objective models is the lack of analysis for the assignment paths and its effective factors.

Some researchers have considered the amount of load imposed from switches to controllers while solving the controller placement problem [17, 18, 28]. They defined capacity for each controller and in the case of controller overload, i.e. when the load to some controllers exceed some threshold, they will perform reassignment.

A new load balancing scheme, called Load Balancing problem for Devolved Controllers (LBDC), for devolved controllers in data centers is presented by [17]. In this approach, each controller monitors the traffics of a part of the switches locally. When traffic load imbalance occurs, some of them will migrate a portion of their monitored works to other controllers so that the workload can be kept balanced dynamically. A Switch Migration-based Decision-Making (SMDM) scheme is proposed in [28]. The proposed approach could be made aware of the load imbalance by a switch migration trigger metric. They introduced the migration efficiency model to make a tradeoff between the migration cost and the load balance rate. They measured the real-time controller load collected by the monitoring module and then if it exceeded a threshold, a decision is made whether to perform switch migration. The work in [29] performed the switch-controller assignment update with respect to dynamic aggregate traffic to make better the load balancing among the controllers. Authors formulated this problem as a stable matching problem, and presented a hierarchically two-phase algorithm to solve it. These papers only focused on reassignment without considering the cost and difficulties of tuning of required parameters in reassignment, the importance of path assignment, and also packet lost which may occur in this process. They only paid attention to the controller's capacity. How and based on which parameters is the allocation path done are not addressed.

In [30], a new method to minimize the end-to-end latency between controllers and their associated switches has been proposed. Different from the existing work, they investigated more possible contributors to the overall latency between controllers and switches instead of propagation latency. The overall latency consists of the end-to-end latency (packet transmission latency, packet propagation latency and switch processing latency) and the controller’s queuing latency. A Clustering-based Network Partition Algorithm (CNPA) is developed to solve the model. However, in this paper, the importance of the assignment, the path assignment, and the importance of link utilization in the assignment process have not been addressed.

Unlike previously mentioned papers, in this paper a new issue in CPP called *switch to controller assignment process* is proposed. This issue makes better quality of solution for CPP and pays attention to how controllers assign to switches and which paths are chosen.

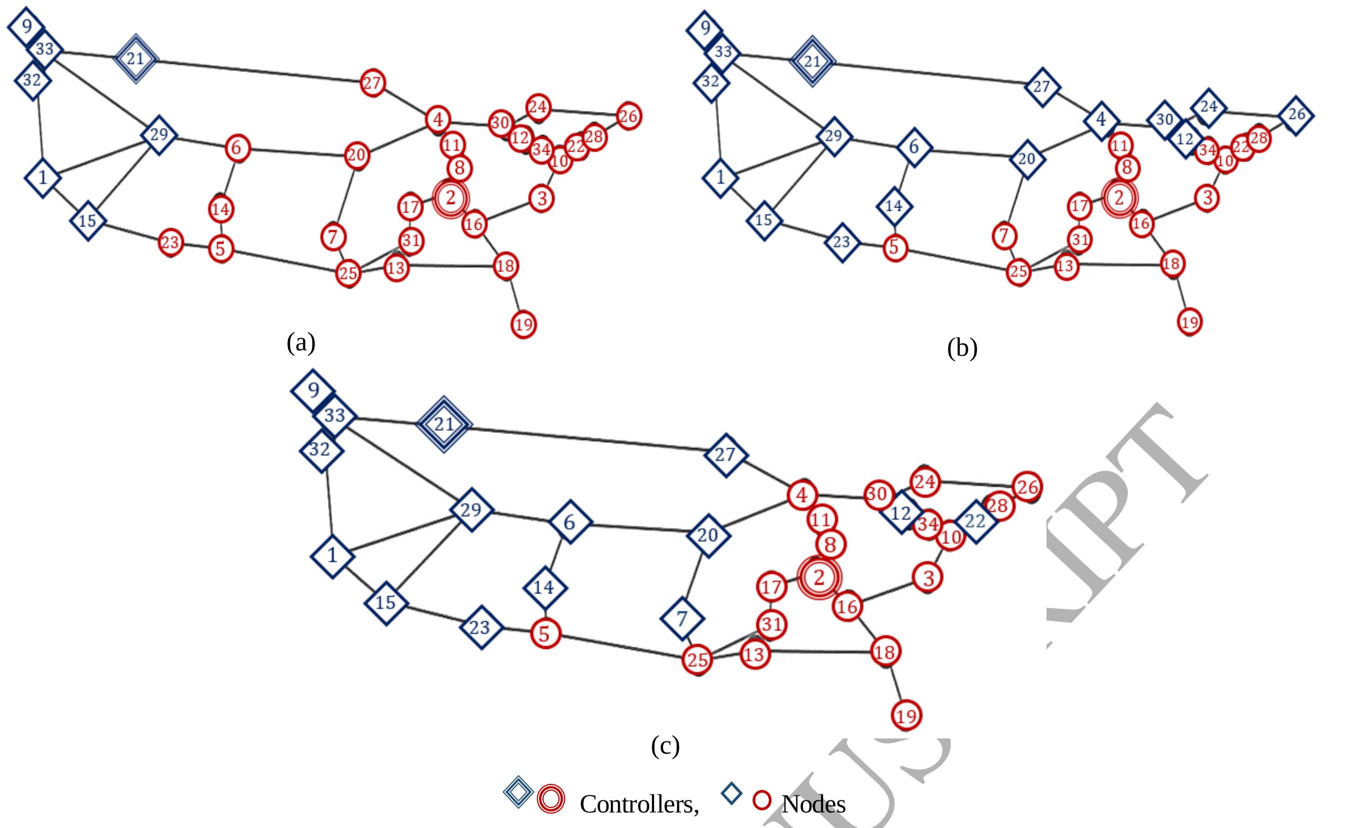

Figure 1. Assignment based on propagation delay

III. Importance of Criteria

In this section, the assignment problems for different criteria and their important roles in the assignment of switches to controllers are evaluated through examples.

A. *Latency Criterion*

Switch to controller latency is the first and the most significant factor for controller placement [10, 11, 12]. Long delay between a switch and its assigned controller can adversely affect the capability of the controller to respond to network events in a timely manner. Let investigate considering only propagation delay for assignment of switches to controllers through an example. Suppose that there should be two controllers in the network. Using the location algorithms existed in the literature [14, 15], and based on two objectives include switch to controller latency and inter-controller latency it will be performed. These controllers are located in the two positions 2 and 21 as shown in Figure 1(a). The figure illustrates the allocation of switches to controllers in the case that only propagation delay is used for allocation process.

According to the graph, an imbalanced distribution of assigned switches to the two controllers 2 and 21 can be seen. Almost 80 percent of the switches are assigned to the controller 2. Either the placement problem is regarded as a capacitated problem, i.e. with capacitated controllers, or in the case that the load balancing is important, this assignment cannot be considered properly. In the former case, this assignment may make the infeasible placement. For instance, assume the load of each switch and the capacity of each controller are 1 and 20 units, respectively. Since, 27 switches (higher than the capacity) are assigned to controller 2, this placement is infeasible and not regarded for further process, although it may be a good placement if other metrics such as processing and queuing delay in nodes, link utilization, and link failures are used in the assignment pr1ocess. For example, node 27 is assigned to the controller 2, passing through three nodes 4, 11, and 8 and along four links. Also, the links $4 \rightarrow 11$, $11 \rightarrow 8$, and $8 \rightarrow 2$ are used a lot to connect controller 2 with its many corresponding switches. Switches 26, 24, 30, 4, and 12 use these links while link between 21 and 27 is not used and can simply assign switches 4 and 27 to controller 21. Hence, these intermediate nodes and links may be so busy during the transferring data packets and as a result, the mentioned delays as well as the probability of failure and packet lost may rise considerably.

B. *Hop Count Criterion*

The hop count is defined as the number of intermediate nodes through which data must pass between source and destination. Since any hop incurs store and forward and other latencies, a large number of hops between source and destination results in lower real-time performance [31]. Then, considering the number of hops can provide the better solution for CPP and assignment process, because this metric somehow include some other latencies. Let investigate considering only the hop count for assignment of switches to controllers through an example.

Figure 1(b) shows the assignment based on hop count. In comparison to the assignment only based on propagation delay, there is a more balanced distribution of assigned switches to the controllers. Each of the controllers is assigned nearly 50 percent of the whole network switches which is a fare distribution. Moreover, some assignments have been getting better. For example, switch 27 is allocated to controller 21 in contrast with the controller 2 in the latency scenario. This assignment makes sense since they are connected without intermediate nodes and through a direct link. Therefore, queuing delay, processing delay in the nodes, and also the probability of the failure of intermediate nodes is reduced. Furthermore, since fewer links are used to connect 27 to 21, the new assignment consists of lower link utilization.

However, some problems may occur if only hop count is used in the assignment process. On the one hand, like delay parameter, the assignment only based on hop count does not consider the path assignment and link utilization. For instance, if there are two paths with similar number of intermediate nodes between a controller and its assigned switch, it chooses one of them without paying attention to the link utilization of each path and control traffic on the links. This matter will be described in the next section. On the other hand, the hop count based assignment does not consider the propagation delay on links and hence, makes several problems in large-scale networks where propagation delay is one of the key parameters in their assignments' computations. For example, if there are two paths with similar number of hops between switches and controllers, it assigns without regarding the length of the links. As a result, none of the above metrics alone can be sufficient for optimizing assignment problem.

C. *Link Utilization Criterion*

This criterion is simply defined as the amount that the link is being used. Generally, if utilization is getting very high, business users may experience reduced performance due to congestion and long delay [31]. Since the link utilization is mostly used as a QoS parameter in the network [32, 33, 34], considering it in the controller placement problem causes several aspects of QoS between switches and controllers.

In this study, we consider link utilization as the percentage of a link's bandwidth that is currently being consumed by control traffic. Although data traffic influences on the performance of the links, and also influences on control traffic, discussing about data traffic is not in the scope of this paper. Furthermore, when we say the link utilization of a typical link is *n* units, it means that this link is used *n* times in the assignment process of the switches and controllers. In the following, the link utilization is considered between the controllers and switches as the only metric in this allocation.

Figure 1(c) demonstrates the link utilization based assignment. This also shows a better distribution and load balancing as opposed to Figure 1(a). In the assignment only based on propagation delay (see Figure 1(a)) switch 12 is assigned to the controller 2 through the path $12 \xrightarrow{1} 30 \xrightarrow{3} 4 \xrightarrow{7} 11 \xrightarrow{8} 8 \xrightarrow{9} 2$. The number written on the above of each link represents how many times this link is used in the assignment process. For example, $12 \xrightarrow{1} 30$ denotes that the link $12 \rightarrow 30$ is used once in all the paths through which the switches are assigned to controllers. In Figure 1 (b), i.e. the assignment only based on hop count, the path $12 \xrightarrow{1} 30 \xrightarrow{4} 4 \xrightarrow{6} 27 \xrightarrow{7} 21$ is used to connect switch 12 to controller 21. From Figure 1(c), it can be seen that switch 12 is allocated to controller 21 through the path $12 \xrightarrow{2} 30 \xrightarrow{4} 4 \xrightarrow{4} 27 \xrightarrow{5} 21$. Clearly, in this case, compared to the delay based assignment, the corresponding controller of switch 12 has changed from 2 to 21. It is due to the high load on the links $11 \rightarrow 8$, and $8 \rightarrow 2$.

Moreover, comparing the hop count based assignment with link utilization assignment will clarify an interesting aspect of the changed allocation. Although, in both cases, switch 12 is assigned to controller 21 through similar connecting path, the usage of the links of the path has decreased. It means that, the load on these links has been reduced and distributed on other less used links of network via this kind of assignment, load balancing among links.

Despite that, this parameter takes the useful aspects of the assignment problem into account, considering only this criterion may create some problems. For instance, since this metric only focuses on the link usage, it aims at assigning the paths so that almost all the links are used more fairly balanced. Therefore, this kind of assignment does not consider other parameters like delays, hop count, closeness and farness of switches to controllers. For example, from Figure 1(c), switch 22 is assigned to controller 21 through the path $22 \xrightarrow{3} 10 \xrightarrow{2} 34 \xrightarrow{1} 12 \xrightarrow{2} 30 \xrightarrow{4} 4 \xrightarrow{4} 27 \xrightarrow{5} 21$ to reduce the usage percentage of busy links $10 \rightarrow 3$, $3 \rightarrow 16$, and $16 \rightarrow 2$. However, if other parameters like hop count and delay are also taken into account, better allocation will be obtained. For example, if hop count is also considered, the switches 4 and 30 will be connected to the controller 21 through the path $4 \rightarrow 27 \rightarrow 21$. If so, in one hand, the control traffic on the links $4 \rightarrow 11$, $11 \rightarrow 8$, and $8 \rightarrow 2$ will be reduced and on the other hand, the closeness of switches to the controllers will be considered.

Generally, considering only one of these parameters may not obtain a good result and hence, an appropriate trade off among them should be followed. Therefore, it must be considered which parameters are critical for network administrators to make a correct decision. Therefore, in the next section, we introduce a new algorithm that enables network administrators to determine desired location-allocation results according to their requirements and preferences. This strategy provides the administrators with considering different preferences in their placement decisions.

IV. PROPOSED METHOD

In this section, regarding the importance of propagation delay, hop count and link utilization in the assignment of switches to controllers, a new efficient algorithm called Analytic Hierarchy Process (AHP) method is proposed to consider these criteria with different preferences. Furthermore, a Controller Placement Genetic Algorithm (CPGA) is introduced to solve this special kind of CPP. In the following, the assignment process and related issues are presented.

A. Assignment Procedure

Here, required information and some preliminaries for AHP assignment process are described. Then, our proposed assignment algorithm is presented.

A.1. Multiple-Shortest Paths

Due to the importance of the path in the process of assignment, it needs to find all shortest paths with the same cost between two arbitrary nodes. Algorithm 1 depicts the procedure *Dijikstra_s_d* in order to compute the shortest path *sp(s,d)* between the source node *s* and destination node *d* together with its cost (shortest distance) *spcost*. Here, *DisMat* denotes the distance adjacency matrix for network nodes defined as follows (Equation (1)):

$$DisMat(i,j)=\begin{Bmatrix} Distance(i,j),\ if\ there is\ a\ direct edge between\ i\ and\ j \\ \infty \qquad\qquad otherwise \end{Bmatrix} \quad (1)$$

In Algorithm 1, *S*, *dist*, and *prev* are n-vectors which represent the set of visited vectors, the set of shortest distance between the source node *s* and other nods, and the set of previous nodes, informs about the best previous node known so far to reach each network node in the path from *s*, respectively. The notation S(1:n) ← 0 is used to indicate that all elements of array S are initialized with 0, i.e. S(1) = S(2) = … = S(n) = 0. Furthermore, in our notation, X ← [X y] (X ← [y X]) indicates that the element *y* is added to the last (first) of array X.

DijikstraMultiPaths_s_d process is extended from *Dijikstra* and is shown in Algorithm 2. This procedure calculates all of the shortest paths from the source node *s* to the destination *d*.

In Algorithm 2, *PathMat* is a matrix so that each of its entry has a specific structure as shown in Figure 2. *PathMat(i,j)* as a typical entry of *PathMat*, when $1 \le i, j \le n$ are two nodes, has two components, where *n* is the number of nodes. Cost represents the shortest distance between these nodes in the networks, which is calculated based on a distance matrix. *AssignPathMat* denotes all the shortest paths which connect *i* to *j* with a length equal to the Cost.

Cost	AssignPathMat

Figure 2. A PathMat entry

Starting from node *s*, in the current phase, there may be more than one candidate node to be selected for proceeding the *Dijkstra*'s algorithm. For example, in Figure 3, two nodes 4 and 6 whose distances from source node 1 are equal to 5, have the minimum distance 4 from node 3. Hence, from node 3, we can have two choices for further processing. In this strategy, instead of just recording any of the nodes 4 or 6 as *Dijkstra*'s algorithm suggests, both of these candidates are maintained and added as a row to *IndMatCan* matrix. Then, the first element 4 is selected, and the standard *Dijkstra* is performed. By doing this, the path $1 \to 3 \to 4 \to 5$ is resulted. It is called the shortest path *sp*. Then array *prev* will be equal to [7 1 1 3 4 3].

Algorithm 1/ Dijikstra_s_d

```
Input: DisMat, s, d , n
  S(1:n)←0, dist(1:n) ← ∞, prev(1:n) ← n+1;
  dist(s)←0;
  while (sum(S) ≠ n ) do
     Candidate ← [];
     for i=1 to n do
        if S(i)==0 then
           Candidate(i) ← dist(i);
        else
           Candidate(i) ← ∞;
        end if
     end for
     u←index of minimum value in Candidate array;
     S(u) ← 1;
     for i=1 to n do
        if dist(u)+ DisMat(u,i) < dist(i) then
           dist(i) ← dist(u)+ DisMat(u,i);
           prev(i) ← u;
        end if
     end for
  end while
  sp← [d];
  while sp(1) ≠ s do
     sp← [prev(sp(1))   sp];
  end while
     spcost ←dist(d);
Output: sp, spcost
```

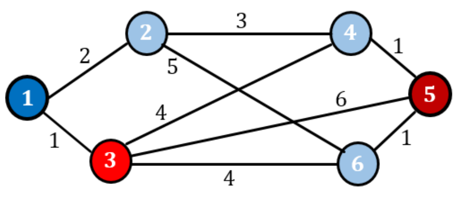

Figure 3. The graph with delays between nodes; source node 1 and destination node 5

In order to find other shortest paths, two procedures are carried out. During computation of *sp*, some useful information is obtained. This information is kept in a matrix called *PrevMat*. Stepping back from the destination 5, the two nodes 4 and 6 have equal distance from the source node, i.e. $dist(4) = dist(6) = 5$. Hence, $PrevMat(5) = [4\ \ 6]$. Also, by doing the same for node 4, can be seen that two nodes 2 and 3 make the same distance from source to 4. Therefore, $PrevMat(4) = [2\ \ 3]$. Next, the two partial path $1 \to 2 \to 4$ and $1 \to 3 \to 4$ are recognized from the source till the node 4. By combining these with path from 4 to destination, two shortest paths $1 \to 2 \to 4 \to 5$ and $1 \to 3 \to 4 \to 5$ of cost (distance) 6 are constructed. This procedure is the same for

$PrevMat(i), 1 \le i \le n$.

```
Algorithm 2// DijikstraMultiPaths_s_d
1.  Input: DisMat, PathMat, s, d , n
2.     S(1:n)←0, dist(1:n) ← ∞, prev(1:n) ← n+1;
3.     dist(s)←0, IndMatCan← [], tt← 1, PrevMat=[];
4.     while (sum(S) ≠ n ) do
5.       Candidate ← [];
6.       for i=1 to n do
7.          if S(i)==0 then
8.             Candidate(i) ← dist(i);
9.          else
10.             Candidate(i) ← ∞;
11.         end if
12.      end for
13.      Minind←indices of minimum value in Candidate array;
14.      if   |Minind| > 1 then
15.         for r=1 to |Minind| do
16.            IndMatCan(tt,r) ← Minind(r);
17.         end for
18.         tt ← tt +1;
19.      end if
17.      S(u) ← 1;
18.      for i=1 to n do
19.         if dist(u)+ DisMat(u, i) < dist(i) then
20.            dist(i) ← dist(u)+ DisMat(u, i);
21.            prev(i) ← u; PrevMat(i) ←[u];
22.         else if dist(u)+ DisMat(u, i) == dist(i) & dist(i) ≠∞ then
23.            Add u to PrevMat(i)
24.         end if
25.      end for
26.    end while
27.    PathMat (s,d).AssignPathMat← [], sp← [d], spt← [];
28.    while sp(1) ≠ s do
29.      if |PrevMat(sp(1))|==1 then
30.         sp← [prev(sp(1))   sp];
31.      else
32.         for j=2: |PrevMat(sp(1))| do
33.            spt← Recursive_Path_s_d(s,PrevMat(sp(1),j),n,prev);
34.            spt← [spt   sp]; sptcost←evaluate(spt);
35.            if sptcost ≤ dist(d) then
36.               Add spt to PathMat (s,d).AssignPathMat
37.            end if
38.         end for
39.         sp=[PrevMat(sp(1),1) sp];
40.      end if
41.    end while
42.    spcost ←dist(d);
43.    PathMat (s,d).Cost ←spcost, Add sp to PathMat (s,d).AssignPathMat;
44.    m ← the number of rows in IndMatCan;
45.    for i= 1 to m do
46.      v← the number of non-zero elements in ith row of IndMatCan;
47.    for j=2 to v do
48.      spt ← [], [spt, sptcost]←dijkstra_s_d(DisMat,IndMatCan(i,j),d);
49.      sptcost← sptcost+dist(IndMatCandidate(i,j));
50.      if sptcost ≤ spcost then
51.         while spt(1) ≠ s & prev(spt(1)) does not belong to spt do
52.            spt← [prev(spt(1)) spt];
53.         end while
54.         Add optimal path spt to PathMat (s,d).AssignPathMat;
55.      end if
56.    end for (j)
57.    end for (i)
58.    PathMat (s,d).Cost ← spcost;
59. Output: PathMat (s,d)
```

The next procedure calls back the matrix *IndMatCan*. Recall that [4 6] was a row in this matrix. The first element (4) is used to construct *sp*. Then the second element (6) is used for further processing. The steps are a bit different with finding the first optimal path. First, the standard **Dijkstra** is executed to find the shortest path from 6 to the destination, i.e. with $s = 6$ and $d = 5$, that is 6→5 with distance 1 as the first partial path. Then the shortest path between the source node 1 and node 6, is considered from array prev. Since $prev(6) = 3$ and $prev(3) = 1$, i.e. the source is reached, the partial path 1→3→6 with distance 5 is achieved. By combining these two partial paths, one other shortest path 1→3→6→5 with distance 6 is created. Therefore, the three optimal paths are recorded as *PathMat(1,5)* and

$$AssignPathMat = \begin{bmatrix} 1\,3\,4\,5 \\ 1\,3\,6\,5 \\ 1\,2\,4\,5 \end{bmatrix}$$

All the process can be seen in Algorithm 2. Here, |X| denotes the number of elements in X. The function evaluates (path) calculates the (distance) cost of the path argument.

A.2. Preprocess

This process is performed to compute the basic element of the algorithm called *InfoMat*. Each entry of this matrix, or *InfoMat(i, j)* for $1 \le i, j \le n$, is a structure shown in Figure 4. Delay denotes the shortest distance, with respect to delay, between two nodes *i* and *j* and Delay Paths indicates the set of these shortest paths, at least on path exists, between the two nodes calculated by the function *DijikstraMultiPaths_s_d*, regarding the delay adjacency matrix *DelayAdjMat*.

Delay	Delay Paths
Hop Count	Hop Paths

Figure 4. The structure of InfoMat(i, j)

See Equation (1), as its input argument. Hop Count represents the shortest distance, considering the number of hops, between the nodes *i* and *j*, i.e. the minimum number of hops through which the nodes *i* and *j* can be connected. Hop Paths are all the paths with this length computed again by the function *DijikstraMultiPaths_s_d*, considering hop adjacency matrix *HobAdjMat*, see Equation (2), as the input.

$$HopAdjMat(i,j) = \begin{cases} Distance(i,j), & if\ there\ is\ a\ direct\ edge\ between\ i\ and\ j \\ \infty & otherwise \end{cases} \quad (2)$$

Hence, all the information required to take different steps is included in this matrix. Link utilization matrix *LUMat* is also initiated. Each entry *LUMat(i, j)* denotes how many times the link *(i, j)* are used in the current assignment of switches to controllers, based on pre-mentioned *AssignMat*. This matrix is initialized by *n* by *n* zero matrix, i.e. none of the links has been used so far.

A.3. Evaluation process

Each time a placement is created, either in the initialization process or searching in a neighborhood, it should be evaluated by the objective functions. The evaluation method depends on two factors; how the controllers of the placement are distributed in the network and which switches are assigned to each controller. Hence, the method by which a controller is selected to be assigned to a switch is a crucial factor. In the following, the assignment process will be explained. Suppose that a placement $P = [p_1, p_2, ..., p_k]$ is considered at this step. Given some objective functions in the controller placement problem, evaluation of this placement depends on how switches are assigned to controllers in *P*.

Table 1. The comparison method of the criteria

Value	The preference situation of criterion *i* with respect to criterion *j*	Explanation
1	Equally Preferred	Criterion *i* has no priority to criterion *j*.
3	Moderately Preferred	Criterion *i* is slightly more important than criterion *j*
5	Strongly Preferred	Criterion *i* is more important than criterion *j*
7	Very strongly Preferred	Criterion *i* is much more important than criterion *j*
9	Extremely Preferred	Criterion *i* is extremely more important than criterion *j*
2-4-6-8	Intermediate	Intermediate values; for example 8 indicates the more importance than 7 and less importance than 9 for criterion i.

Table 2 Mutual comparison of the three criteria

	Propagation Delay	Hop Count	Link Utilization	Weights
Propagation Delay	1	B	C	w_1
Hop Count	1/b	1	D	w_2
Link Utilization	1/c	1/d	1	w_3

As discussed, the three criteria propagation delay (PD), hop count (HC), and link utilization (LU) are considered for assigning switches to controllers in this paper. Therefore, the assignment of a controller to a switch is a multi-criteria problem. We use the AHP technique to solve this problem as an efficient approach to this kind of problems. It is flexible to impose the preferences of the decision maker to find the solution. For a typical switch *s*, the goal is assigning a controller among the candidates $p_1, p_2, ..., p_k$ to *s*. If *s* and one of the controllers in *P* are the same location, this controller is assigned to *s*. Otherwise, the next steps should be executed.

A.3.1 Mutual Comparison of Criteria

In this step, a criterion weight w_i is assigned to each criterion *i*, *i=1,2,3*. Now, the construction procedures of these weights are explained. The method [20, 21] showed in Table 1 is applied for mutual comparison of the criteria.

The initial weights for criteria are determined by information provided in Table 1. A 3-by-4 Table 2 is generated by the decision maker to capture the mutual comparison results of the criteria. Afterwards, the geometric mean of each row is calculated. Finally, the three obtained weights are normalized and written in the Weights column. The final phase is to yield a set of weights whose sum equals to 1 as $w = (w_1, w_2, w_3)$.

A.3.2. Mutual Comparison of Candidates

After the weight determination of the criteria, in the next step, a pairwise comparisons of the candidates, i.e. $p_1, p_2, ..., p_k$, should be performed based on each criterion. Therefore, it is required to do three kinds of comparison between candidates, with respect to propagation delay, hop count and link utilization, respectively.

A.3.2.1. Comparison method

Algorithm 3 shows the evaluation method which is common for the three comparisons. The inputs are two values *a* and *b* which are going to be compared together with two values e_{min} and e_{max} as their lower and upper bounds, respectively, i.e. $a,b \in [e_{min}, e_{max}]$. If *a* is equal to *b*, the output will be 1. It means they have the same preferences. Otherwise, $a < b$ or $a > b$, the interval $[e_{min}, e_{max}]$ consisting of *a* and *b* are divided into 9 separate intervals $[(i-1)*p, i*p], i = 1,...,9$, where $p = {(e_{min} - e_{max})}/{9}$.

The number of intervals is according to the comparison method of Table 1. If $a < b$, the results, i.e. *w*, belongs to $\{1,2,..,9\}$. In this case, the smaller *a* in comparison with b is, the higher the result will be. On the other hand, if $a > b$, the result belongs to $\{1, 1/2, 1/3, ..., 1/9\}$. In this case, the more *a* in comparison with b is, the smaller the result will be. Therefore, the output of the comparison depends on the order of the input arguments, i.e. a and b. For example, if the result is 9, it means a is extremely preferred to b and if the results is 1/5, that means, b is strongly preferred to a, see Table 1. In the following, the method is described by the three kinds of comparisons among the candidates.

Algorithm 3/ Criterion Comparison Procedure(CriComparison)

```
1.  Input: a, b, $e_{max}$, $e_{min}$
2.  if a==b then w= 1; return  end if
3.      p ← ($e_{max}$- $e_{min}$)/9;
4.      if a<b then
5.        for i=1 to 9 do
6.            if (i-1)*p < b-a ≤ i*p then
7.               w ← i; return
8.            end if
9.        end for
10.     end if
11.     if a>b then
12.       for i=1 to 9 do
13.           if (i-1)*p < a-b ≤ i*p then
14.              w ← 1/i; return
15.           end if
16.       end for
```

```
17.     end if
18. end if
19. Output: w
```

A.3.2.2. Comparison based on Propagation Delay

Table 3 demonstrates such *a* comparison for the first criterion, i.e. propagation delay. The first *k* columns create a *k*-by-*k* matrix. The last column is weights obtained by normalization of the geometric means of the rows similar to Table 2. Algorithm 4 shows how this comparison is made. Here, d_{min} and d_{max} denote the minimum distance between node *s* and all the candidates, i.e. $p_i, i = 1,..,k$, in *P*, respectively.

For each pair of candidates (p_i, p_j), *a* and *b* represent the shortest distances (based on propagation delay) from *s* to p_i and p_j, respectively. $W_D(i,j)$ is calculated as the comparison value for this pair of candidates through Algorithm 3. For this, the values *a*, *b*, d_{min} and d_{max} are sent as the arguments of Algorithm 3. These values are the numbers in the i^{th} row of the *k*-by-*k* matrix in Table 3. Afterwards, the geometric mean of each row is calculated and after normalization, they are written as the "Weights" column in Table 3. The output of this mutual comparison is the weight vector $(w_{1,pd}, w_{2,pd},.., w_{k,pd})$.

Table 3. Mutual comparison of the controller candidates based on propagation delay

	p_1	p_2	…….	p_k	Weights
p_1	1	$a_{1,2}$	…….	$a_{1,k}$	$w_{1,pd}$
p_2	$1/a_{1,2}$	1	……..	$a_{2,k}$	$w_{2,pd}$
⋮	⋮	⋮	⋮	⋮	⋮
p_k	$1/a_{1,k}$	$1/a_{2,,k}$		1	$w_{k,pd}$

Algorithm 4/ Delay Prefer Determination Procedure

```
1.  Input: InfoMat, P, LUMat, s , n, k
2.  $W_D \leftarrow$ [];
3.  $d_{min} \leftarrow$ minimum delay between s and P;
4.  $d_{max} \leftarrow$ maximum delay between s and P;
5.  for i=1 to k do
6.      for j=i to k do
7.          a ← InfoMat(s ,P(i)).Delay;
8.          b ← InfoMat(s ,P(j)).Delay;
9.          $W_D(i,j) \leftarrow$ CriComparison( a,b, $d_{min}$, $d_{max}$); //Algorithm3
10.         $W_D(j,i) \leftarrow 1/(W_D(i,j))$;
11.     end for
12. end for
13. for i=1 to k do
14.     $w_d(i) \leftarrow$ the geometric mean of row i;
15. end for
16. Normalization of the weight vector $w_d$;
17. Output: weight vector $w_d$
```

Algorithm 5/ Hop Count Prefer Determination Procedure

```
1.  Input: InfoMat, P, LUMat, s , n, k
2.      $W_H \leftarrow$ [];
3.      $h_{min} \leftarrow$ minimum hop count between s and P;
4.      $h_{max} \leftarrow$ maximum hop count between s and P;
5.      for i=1 to k do
6.          for j=i to k do
7.              a ← InfoMat(s ,P(i)).Hop Count;
8.              b ← InfoMat(s ,P(j)).Hop Count;
9.              $W_H(i,j) \leftarrow$ CriComparison( a,b, $h_{min}$, $h_{max}$); //Algorithm3
10.             $W_H(j,i) \leftarrow 1/(W_H(i,j))$;
11.         end for
12.     end for
13.     for i=1 to k do
14.         $w_h(i) \leftarrow$ the geometric mean of row i;
15.     end for
16.     Normalization of the weight vector $w_h$;
17. Output: weight vector $w_h$
```

A.3.2.3. Comparison based on Hop Count

The process at this stage is similar to Algorithm 4. However, here, hmin and hmax denote the minimum hop count between node s and all the candidates, i.e. $p_i, i = 1,..,k$, in P, respectively. For each pair of candidates (p_i, p_j), *a* and *b* represent the shortest distances (based on hop count) from s to pi and pj, respectively. Algorithm 5 shows this procedure. The result of this comparison is $(w_{1,hc}, w_{2,hc},.., w_{k,hc})$.

A.3.2.4. Comparison based on Link Utilization

Algorithm 6 is used for comparison based on link utilization between candidates. Here, l_{min} *and* l_{max} denote the minimum and maximum values of average link utilization of all paths with shortest propagation delay and shortest hop counts between node *s* and all the candidates, i.e. $p_i, i = 1,..,k$, in *P*, respectively. First, for each pair *(s,p_i)*, all the shortest paths based on propagation delay and hop count are determined from the *InfoMat*. Then, the link utilization of each path is calculated as the summation of link utilization of all the links of the path (from *LUMat*). Finally, the average value of these link utilizations is calculated and regarded as the average link utilization of paths from *s* to p_i. For each pair of candidates (p_i, p_j), *a* and *b* represent the average link utilization from *s* to p_i and p_j, respectively. $W_{LU}(i,j)$ is calculated as the comparison value for this pair of candidates through Algorithm 3. Afterwards, the geometric mean of each row is calculated and these weights are normalized. The output of this process is the weight vector $(w_{1,lu}, w_{2,lu},.., w_{k,lu})$.

Algorithm 6/ Link Prefer Determination Procedure

```
1.  Input: InfoMat, P, LUMat, s , n, k
2.      $W_{LU}$ ← [];
3.      $l_{min}$←minimum average link utilization between s and P;
4.      $l_{max}$←maximum average link utilization between s and P;
5.      for i=1 to k do
6.          for j=i to k do
7.              a ← average link utilization between s and P(i);
8.              b ← average link utilization between s and P(j);
9.              $W_{LU}(i,j)$ ← CriComparison( a,b, $l_{min}$, $l_{max}$); //Algorithm3
10.             $W_{LU}(j,i)$ ← 1/($W_{LU}(i,j)$);
11.         end for
12.     end for
13.     for i=1 to k do
14.         $w_{lu}(i)$ ← the geometric mean of row i;
15.     end for
16.     Normalization of the vector weights $w_{lu}$;
17. Output: weight vector $w_{lu}$
```

Therefore, for a typical node *s*, all the candidates $p_i, i=1,..,k$ are mutually compared and three weight vectors $(w_{1,pd}, w_{2,pd}, ..., w_{k,pd})$, $(w_{1,hc}, w_{2,hc}, ..., w_{k,hc})$ and $(w_{1,lu}, w_{2,lu}, ..., w_{k,lu})$ are made as the weights of the *k* candidates based on propagation delay, hop count, and link utilization, respectively. This result is depicted in Table 4 and the following matrix:

$$WC = \begin{pmatrix} w_{1,pd} & w_{1,hc} & w_{1,lu} \\ w_{2,pd} & w_{2,hc} & w_{2,lu} \\ & .. & \\ w_{k,pd} & w_{k,hc} & w_{k,lu} \end{pmatrix}$$

Table 4. The result table of mutual comparison between candidates

	Propagation Delay	Hop Count	Link Utilization
p1	$w_{1,pd}$	$w_{1,hc}$	$w_{1,lu}$
p2	$w_{2,pd}$	$w_{2,hc}$	$w_{2,lu}$
:	:	:	:
pk	$w_{k,pd}$	$w_{k,hc}$	$w_{k,lu}$

A.4. Main Algorithm

Finally, the Algorithm 7 is used to do the assignment of switches to controllers based on the three mentioned criteria. Algorithm 7 demonstrates how the controllers in a placement $P=[p_1, p_2, ..., p_k]$ are assigned to all switches $s \in V$. According to line 3, the *LUMat* is initialized as follows:

$$LUMat(i,j) = \begin{cases} 0, & if\ (i,j) \in E \\ 1, & otherwise \end{cases} \quad (3)$$

Where $i,j=1,..,n$, and *E* denotes the set of edges in the network graph. Line 4 indicates that the three criteria are compared mutually to determine the normalized weights $w=(w_1, w_2, w_3)$. If a typical switch *s* is a controller in the placement *P*, i.e. $\exists 1 \le i \le k; s=P(i)$, *s*, as a controller, is assigned to itself, as a switch. In this case, the assigned path would be *s* itself. Otherwise, i.e. *s* is not a controller number in *P*, the following process is performed. First, the weights $w=(w_d, w_h, w_{lu})$, as described in Section IV.A.3.1, are obtained, as illustrated in matrix *WC*, see Table 4. In line 11, the score of each candidate $p_i, i=1,..,k$, is obtained as the sum of the products of the priority of each criterion by the priority of the candidates based on the criterion. Therefore:

$$sc(p(i)) = w_1 * w_{i,pd} + w_2 * w_{i,hc} + w_3 * w_{i,lu}, i=1,..,k \quad (4)$$

After obtaining the score of all candidates based on Equation (4), the candidate which is getting the maximum score is selected to be assigned to the switch *s*. The selected candidate is called *d*.

Algorithm 7/ AHP-Based Assignment

```
1.  Input: InfoMat, P, LUMat, DelayAdjMat,
2.      HobAdjMat, n, k
3.      LUMat ← Initialize(LUMat);
4.      w=($w_1$,$w_2$,$w_3$) ← Mutual comparison of the criteria //Table 1&2
5.      AssignVector ← [],  AssignPathMat ← [];
6.      for each node s in V do
7.          if $s \in P$ then
8.              d=AssignVector(s) ← s;  path← [s];
9.              Add path to AssignPathMat;
10.         else
11.             WC=($w_d$, $w_h$,$w_{lu}$) ← Mutual comparison of candidates // Algorithms 4,5, 6 & Table 4
12.             d=AssignVector(s) ← $P(Argmax_{i=1,..,k}(WC * w))$;
13.             PathMat ← []; EvalMat ← [];
14.             Add all shortest paths from s to d into PathMat
15.             EvalMat ←Evaluation of PathMat
16.             WA=($w_{Ad}$, $w_{Ah}$,$w_{Alu}$) ← Mutual comparison of candidates // Algorithms 4,5, 6 & Table 4
17.             path ← $Path(Argmax(WA * w))$
18.         end if
19.         Add path to AssignPathMat;
20.         LUMat ← UpdateULMat( LUMat, path );
21.     end for
22. Output: AssignVector, AssignPathMat, LUMat
```

As there may be more than one shortest path regarding the three criteria, propagation delay, hop count, and link utilization, selecting the best assigned path for connection between *s* and *d* would be our next priority. First, some the distinct shortest paths with respect to propagation delay and hop count existing in the *InfoMat* matrix are added to a matrix called *PathMat*. This matrix then would be the matrix of candidate paths in order to connect the current switch *s* to its assigned controller *d*. Here, the word candidate is used to each of the paths. In parallel, the delay, hop count, and link utilization of such paths are added to *EvalMat* matrix.

Table 5 shows this matrix and the candidate's paths. Like what is described for mutual comparison of controller candidates, see line 11, the path candidates: Path 1 to Path X are mutually compared with respect to the three criteria. For example, for comparing Path 1 with Path 2 with respect to propagation delay, Algorithm 3 is used with $a=D_1$ and $b=D_2$, and d_{min} and d_{max} as the minimum and maximum values in the first column of the *EvalMat* matrix, respectively. Hence, regarding to Table 5:

$$d_{min} = min\{D_1, D_2, .., D_X\},\ d_{max} = max\{D_1, D_2, .., D_X\} \quad (5)$$

Table 5. The X-by-3 matrix EvalMat

	Delay	Hop Count	Link Utilization
Path 1	D_1	H_1	L_1
Path 2	D_2	H_2	L_2
:	:	:	:
Path X	D_X	H_X	L_X

Line 17 shows the weight matrix $w_A = (w_{Ad}, w_{Ah}, w_{Alu})$ as the output of these comparisons based on algorithms related to criteria comparison (Algorithms 4 to 6). Hence, a table and matrix W_A similar to Table 4 and matrix *WC* is resulted, respectively. By the same discussion as the controller candidates, the formulation (4) is used to determine the best candidate path among $\{Path1, ..., PathX\}$. Therefore, for a typical switch *s*, the process describing which controller and path from *s* to the controller must be assigned, has been already presented. The assigned path is added to the matrix of assigned paths, i.e. *AssignPathMat* and the matrix of link utilization or *LUMat* is updated with increasing the utilization of the links along this path by 1, see lines 19 and 20. For a typical placement *P*, Algorithm 7 is performed to assign switches in *V* to the controllers in *P*.

B. Location Procedure

In order to find location of controllers in a network, an algorithm is proposed to solve CPP based on AHP method. For this, in the following, an adapted Genetic Algorithm is presented to solve our special kind of controller placement problem.

B.1. Controller Placement Genetic Algorithm (CPGA)

The controller placement problem deals with location of controllers with respect to the available network topology and the required number of controllers. The user may define various metrics to control the placement of the controller in a network. Since minimizing latencies between each node and its assigned controller constitutes a crucial aspect of the controller placement problem, we choose deliberately maximum node to controller latency as the only objective function in order to evaluate the optimal placement of CPGA with respect to the propagation latency.

The network is represented a graph ??=(??,??), where *V* is the set of *n* nodes, or switches which are connected by the links, or edges, from the set *E*. Furthermore, a distance matrix *D* contains the shortest path latencies between each pair of nodes where ?????? indicates the latency between node *i* and node *j*. The desired number of controllers, denoted by *k*, is supposed to be known before starting to solve the problem.

In this section, the Controller Placement Genetic Algorithm (CPGA) (Algorithm 8) is presented to solve the following problem:

$$min\big(f(P) = max_{v \in V}\ Delay(v, c_v)\big), \\ |P| = k, P \in 2^V \tag{6}$$

Where c_v is the controller on the placement *P* which is assigned to switch *v*, and $Delay(v, c_v)$ denotes the shortest distance (based on the propagation delay) between switch *v* and controller c_v. The objective function, maximum node to controller latency, decide on whether a solution is accepted or not. Hence, it indirectly influence on how location are made.

Algorithm 8 shows our *CPGA*. Inputs are *G=(V,E)*, *k*, *maxIt*, *wicmax*, *DelayAdjMat*, *HobAdjMat* which denote the topology graph, the number of controllers, the maximum number of iteration as the termination criteria, permutation counter, delay adjacency matrix, and hop adjacency matrix, respectively. An external set *BestP* is used to record the solutions found so far and hence, initialized with first a random placement set *P* and will be the output.

First, preprocess stage (see Section IV.A.2) is implemented to gather information used in evaluation of solutions (see IV.A.2). Next, using a randomization process, an initial population of placements is created. These solutions should be evaluated in the next step. For this purpose, for each placement, Algorithm 7 is used to assign switches to the controllers in the corresponding placement. After assignment process, objective function Equation (6) is used to evaluate these solutions.

Algorithm 8 uses main genetic algorithm operators: crossover and mutation. Here, a placement is regarded as a *k*-vector. In Algorithm 8, first a random placement *P* is created and considered as the best placement found so far as *BestP*. Note that, as soon as the placement of controllers is found through algorithm's operators, the described AHP process is implemented for assignment of all switches to this set of controllers and control path assignment. Afterwards, it is evaluated based on objective function, Equation (6).

All this process is called evaluation. Using a randomization process to create initial population, the evaluation process is performed. The main loop consists of these operations: A random placement Q is generated to execute the crossover with P. Here, a random crossover is considered in which the obtained child inherits its bits from two parent placements P and Q randomly. Mutation is also performed on the child P\` by replacing one of its controllers with a random one. Then a Local Search approach is used to search in the neighborhood of the mutated placement (P\`\`).

Algorithm 8/ Controller Placement Genetic Algorithm

```
1:   Input: G=(V,E), k, MaxIt, wicmax, DelayAdjMat, HobAdjMat
2.     n← |V|
3.     Preprocess stage
3.     P← Generate a random placement
4.     Evaluate all search agents (AHP+Objective) //Call Algorithm7
5:     BestP ← P, it←1, wic←0;
6:     while it ≤ MaxIt do
7:        Q ← Generate a random placement
8.        Evaluate all search agents (AHP+Objective) //Call Algorithm7
9:        P` ← Crossover (P,Q);
10:       P`` ← Mutation (P`);
11:       P``` ← Local Search(P``);
12:       if f(P```) ≤ f(P) then
13:          P← P```;
14:          if f(P) ≤ f(BestP) then
15:             BestP← P;
16:          end if (line 14)
17:       else
18:          wic ← wic+1;
19:       end if (line 12)
20:       if wic==wicmax then
```

```
21.             P← Mutation(BestP);
22.             wic←0;
23.         end if (line 20)
24.         it← it+1;
25.         Evaluate all search agents (AHP+Objective) //Call Algorithm7
26.     end while
27:  Output: BestP
```

This procedure is an extension for the mutation procedure. In the local search process, one controller (from $p_i, i = 1,..,k$) is replaced with some random numbers and the best neighbor, in terms of the objective function, is selected as P'''. If the resulted placement has a better objective function than the original placement *P*, it is substituted with *P*. If not, *P* is passed to the next phase of the algorithm and *wic* as the "without improvement counter" is increased. When this counter reaches a threshold *wicmax*, a mutation is performed on the best placement found so far and the algorithm restarts again.

The output of this algorithm is the best found placement *BestP*. Now, an iteration of the algorithm is described. For termination condition, a threshold is set as the maximum iteration number.

Algorithm 8 is used in Section V.B, in order to analysis how the assignment process can affect the optimal solutions of the controller placement problem.

Table 6. Different scenarios for analyzing the AHP-based assignment

	b=9, c=9, d=1	**b=1/9, c=1, d=9**	**b=1, c=1/9, d=1/9**	**b=1, c=1, d=1**	**b=1, c=5, d=5**
k=2	CS1	CS2	CS3	CS4	CS5
k=3	CS6	CS7	CS8	CS8	CS10
k=4	CS11	CS12	CS13	CS14	CS15
k=5	CS16	CS17	CS18	CS19	CS20

V. RESULT AND EVALUATIONS

In this section, the evaluation method through which our experiments are performed is described in some planned scenarios. In order to evaluate the performance of our proposed method, various evaluation experiments are carried out. This is done through two subsections. The first Section (V.A) aims to analyze the performance of the AHP-based algorithm with respect to different weights. For this, the performance of AHP-based assignment is examined on some case studies. The goal of this evaluation is to analyze the importance of different influential factors on the assignment of switches to controllers. As the second scenario, Section (V.B), to show that the proposed model has a great deal of efficiency in locating a problem and provides better quality solutions, we have used two important metrics to evaluate our model. Each set of solutions are evaluated based on the two indicators called "link load indicator" as well as "maximum link utilization of control paths", or briefly "$LL(l)$" and "*MLUCP*", respectively (defined in Section V.B) to compare the obtained placements. For this, the CPGA that is used to solve the Equation (7) is run in two ways with different weights. It shows that considering the AHP-assignment during the CPGA, leads to placements which are reasonable in terms of not only propagation latency, but also link load balancing.

All experiments are run on a 4-GH Intel Core i7 machine with 8 GB RAM. Solution approaches have been implemented using Matlab 2016a. The internet topology Zoo [35] which has been considered in several papers as the basic topology set for controller placement problem is used in our evaluation method. The AHP weight vectors used in our evaluation are inserted in Table 6. Furthermore, we assume the control-flow arrival rate of every node is 100kpps. This assumption is made based on reference [25].

A. Assignment analysis

In this section, the performance of AHP-based assignment is examined on some case studies. The goal of this evaluation is to analyze the importance of different influential factors such as propagation delay, hop count, and link utilization on the assignment of switches to controllers. Since the Internet2 topology has been mostly considered in recent papers as the basic topology for controller placement problem [3, 14, 25]. We also use this topology with 34 nodes, different number of controllers (*k*) and five parameter settings for AHP-based assignment algorithm. For each number of controllers k, 50 sets of controllers (placement), which are the optimal placements of our NSGA-II algorithm [14] are investigated and the average of values are recorded. The scenarios are inserted in Table 6. The first column shows different values for *k*, while the first row demonstrates some values for vector (b, c, d), according to Table 2, each one corresponds to an input weight vector for Algorithm 7. We call each case study which is written in Table 6 as a CS. For example, CS1 corresponds to the internet2 topology with 2 controllers and the following matrix related to mutual comparison of the criteria, see Table 2:

$$\begin{pmatrix} 1 & b & c \\ 1/b & 1 & d \\ 1/c & 1/d & 1 \end{pmatrix} = \begin{pmatrix} 1 & 9 & 9 \\ 1/9 & 1 & 1 \\ 1/9 & 1 & 1 \end{pmatrix} \tag{7}$$

Therefore, for each controller set, five case studies corresponding to different weights are defined. After doing several evaluations and tests on AHP-based method with different weights on various topologies from Internet

Topology Zoo, five weights were selected with better performance. The first three emphasize strictly on propagation delay, hop count, and link utilization, respectively. The forth consider the same priority for the criteria and the last imposes the same preference for propagation delay and hop count but much less emphasis on link utilization. Hence, the total number of case studies is 20. The results are used to analyze how different weights can influence on the obtained assignment. Finally, we want to conclude the most appropriate trade-off among the three criteria.

Figure 5 consists of five sub-figures a-e depicting the results for CS1-CS5, respectively. Kiviat charts have been recommended as richly suited for making trade-off decisions among some candidates. Kiviat chart here is used to illustrate the influence of each weight set of the criteria on the obtained assignment. Each scaled axis shows an integrated value of one of the criteria. Indeed, after implementing Algorithm 7 on a case study, total propagation delay and total hop count (as the sum of propagation delays and hop counts of each node to its assigned controller, respectively) and maximum link utilization (as the maximum of link utilizations of each node to its assigned controller) are calculated for the obtained assignment.

This procedure is repeated 50 times for the case study and the average of values are recorded. Finally, the numbers are normalized by dividing on the maximum corresponding values obtained in all runs and written on each axis. Hence, all the values for the three criteria have the same interval range [0, 1].

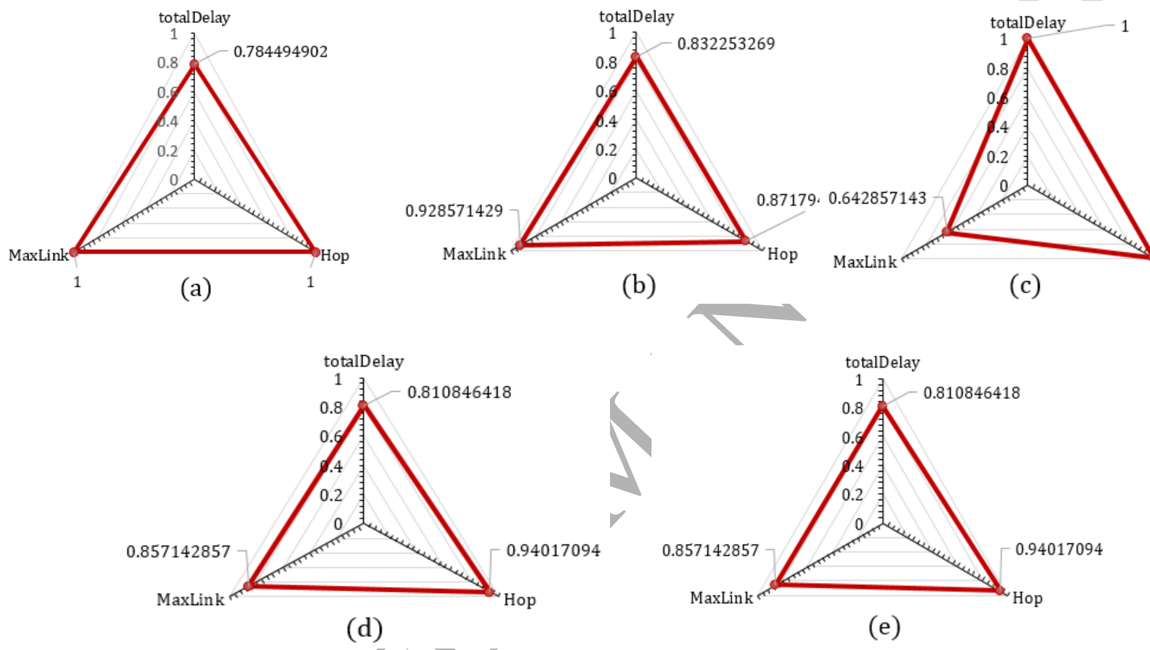

Figure 5. Kiviat graph a) Case Study 1, b) Case Study 2, c) Case Study 3, d) Case Study 4, e) Case Study 5 based on Table 6.

We can look at the first plot (Figure 5 (a)) of the assignment which is only based on propagation delay and see that it received the highest value of hop count and link utilization. That means, considering only the propagation delays is resulted in the maximum values for the two other items. Therefore, although this assignment considers the shortest path between the switches and controllers with respect to the propagation delay, it does not take the other possible delays like queuing delay, processing delays, link utilization, and the failure probability in nodes and links into account. Emphasizing only on hop count (Figure 5(b)) has led to a better assignment than the assignment achieved when only link utilization (Figure 5(c)) is taken into account. Also, Figure 5(c) shows that the strict emphasis on link utilization may not consider shortest paths between switches and controllers and hence, is not an enough criterion. Comparison between Figures 5(d) and 5(e) clarify that the numbers are close and more reasonable as opposed to their counterparts in Figures 5(a), 5(b), and 5(c), with lower values for 5(e).

However, the minimum values for propagation delay, hop count, and link utilization are respectively achieved by Figures 5(a), 5(b), and 5(c). Hence, depending on which criterion is the most important for the administrator, the preference weight vectors can be chosen. However, for implanting in real networks and considering different possible delays and failures, Figure 5(e) may offer a better trade-off for these samples.

Figure 6 provides the administrator with a total description of the results obtained by the all preferences in one graph. The horizontal axis shows the case studies and vertical axis presents the normalized average values (NAV) for the three criteria. Hence, each case study is presented by three bars, each one represents one criterion.

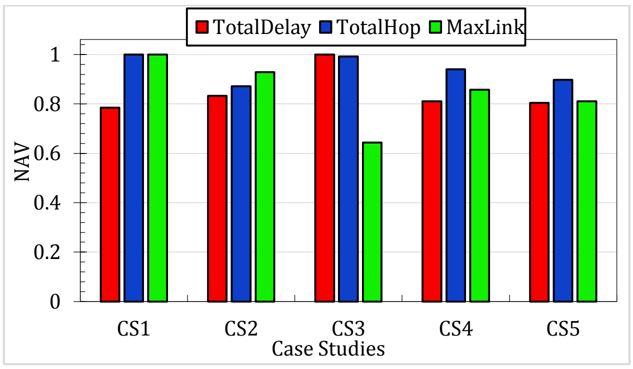

Figure 6. Three criteria values for CS1 to CS5

First triple reveals that focusing strictly on propagation delay will rise substantially the total hop count and maximum link utilization, particularly the latter. Hence, some other possible delays and failures may arise in the situation that switches are assigned to controllers based on the shortest paths with respect to propagation delay.

It is inferred that these shortest paths pass through the busiest links. Therefore, if link utilization is not considered, shortest paths may lead to considerably higher delay paths than expected. As well, based on the chart, the second triple, i.e. emphasizing strictly on hop count, gets more reasonable values compared to the first three cases. In this Figure, only considering link utilization caused a considerable rise in the total delay and hop count. However, for each of the last two triples, the heights of bars are close together. It shows that, setting up a trade-off between the criteria and not just considering one of them can result in the assignments whose three criteria's level are almost the same.

One can see that the last triple relatively dominates the other four ones. Table 7 shows the deviation of minimum values for each of the three criteria. We choose the deviation from the minimum because we are looking for a good trade-off among the criteria and the best value for each criterion will be the minimum one. Each entry of the table has been calculated as by Equation (8):

$$t_{CSi_c} = \sqrt{\frac{x_{CSi_c} - x_c^{min}}{5}}, \quad i = 1,..,5, c \in \{D,H,L\} \qquad (8)$$

Where c represents a criterion, propagation delay (D), hop count (H), and link utilization (L), x_{CSi_c} denotes the normalized average value for i^{th} case study with respect to criterion c, and x_c^{min} represents the minimum value achieved by all the case studies for criterion c.

Table 7. Deviation of minimum for CS1 to CS5

	deviation (Min)		
	TotalDelay	TotalHop	LinkUtilization
CS1	0	0.670820393	2.236067977
CS2	0.118179549	0	1.788854382
CS3	0.533273996	0.626099034	0
CS4	0.065207637	0.357770876	1.341640786
CS5	0.047095486	0.134164079	1.050951949

Table 8. Three criteria values for CS6 to CS20

N. Controllers	CS#	Total Propagation Delay	Total Hop Count	Maximum Link Utilization
K=3	**CS6**	0.82215891	0.96	1
	CS7	0.88552339	0.88	0.91667
	CS8	1	1	0.83333
	CS9	0.86473101	0.93333	1
	CS10	0.82883807	0.88	0.83333
K=4	**CS11**	0.696114	1	1
	CS12	0.751438	0.80233	0.45455
	CS13	1	1	0.45455
	CS14	0.759027	0.87209	0.63636
	CS15	0.715973	0.80233	0.45455
K=5	**CS16**	0.675426	0.91071	1
	CS17	0.73475	0.875	1
	CS18	1	1	0.75
	CS19	0.73475	0.875	1
	CS20	0.73475	0.875	0.875

Table 9. Deviation of minimum for CS6 to CS20

N. of Controllers	CS#	deviation (Min)		
		TotalDelay	TotalHop	LinkUtilization
K=3	**CS6**	0	0.2683157	0.44721395
	CS7	0.1106116	0	0.22360698
	CS8	0.3104289	0.4024926	0
	CS9	0.0743185	0.1788838	0.44721395
	CS10	0.01166895	0	0
K=4	**CS11**	0	0.7602112	2.68328173
	CS12	0.0997838	0	0
	CS13	0.5474708	0.7623112	0
	CS14	0.1133997	0.2683157	0.89442791
	CS15	0.0352079	0	0
K=5	**CS16**	0	0.0894719	0.447213595
	CS17	0.0675785	0	0.447213595
	CS18	0.368725	0.3130517	0
	CS19	0.0675785	0	0.447213595
	CS20	0.0675785	0	0.223606798

According to this table, the highest deviation corresponds to CS1, when considering link utilization criterion. It means that considering only propagation delay for assigning switches to controllers may not pay any attention to how much the links are busy. Furthermore, CS5 may be a far better choice than others if an appropriate trade-off among the criteria is desired by the administrator. Indeed, focusing equally on both propagation delay and hop count far more than link utilization, has resulted in the best assignment. Due to space limitation, results of other case studies are only brought in Table 8 and Table 9 similar to evaluation method in Figure 6 and Table 7 respectively.

In summary, one can see that strictly focusing on one metric will substantially sacrifice two other criteria. However, the last two set of parameter settings here work well. Again, focusing equally on both propagation delay and hop count far more than link utilization (the last weight vector), has resulted in the best assignment and offers a better trade-off for minimum of costs.

B. *Placement analysis*

In this section, the CPGA proposed in Section IV.B is used to solve the Equation (6). We show that considering the AHP-assignment during the CPGA, leads to placements which are reasonable in terms of not only propagation latency, but also link load balancing.

As mentioned before, load issue is an important aspect in controller placement problem and should be taken into account, especially load of the control paths. It is important to note that if all the switches are assigned to controllers through the paths with several overlaps (common links) or some links are used too much through the assignment, following results are highly expected:

1) The delay between switches and controllers increases. It is due to the rise in the queuing and processing delay within the related paths.

2) If the failure occurs for the highly shared links, several numbers of assignments between switches and controllers which include this link should be changed. Note that the reassignment is an expensive process. If the number of switches that has to be reassigned is high, both managing the error and maintaining the network in a stable state are expensive.

Therefore, link load balancing provides advanced failover and bandwidth management for SDN to assure continuous operation of the network in the event that one or more links between switches and controllers become unavailable or slow to respond.

The *MLUCP* and $LL(l)$ metrics are important because they consider the load balancing on the links [22]. The lower the value of these metrics, the better the load on the links is done. We have used these metrics to evaluate the quality of our solution and we have shown that the proposed method has a better load balancing than other methods. The following link load indicator is used to evaluate the quality of placements in terms of link load balancing. Given a typical placement *P*, the following quantity is used for load of each link *l* in the network topology:

$$LL(l)=\frac{(n-1)}{m}*2^{n-1} \qquad (9)$$

Where *n* denotes how many times this link, *l*, is used in all paths between controllers and their allocated switches in the assignment imposed by placement *P* and *m* shows the total number of links (including the repeated ones) used in the paths. Based on this formula, increasing the number of link usage will be punished exponentially. The quantity obtained by (9) is used as an indicator for the quality of the assignment based on link load balancing.

Here, for the first scenario, the algorithm CPGA is run on the Equation (6) in two ways. First, the problem considers the minimum propagation delay as the only criterion for the assignment process. Second, The AHP assignment with weight setting *b=1*, *c=5*, and *d=5* is taken into account. To this end, simulations are done using internet2, with 34 nodes, as our network topology and the results denote which assignment is better regarding link load balancing. Suppose that the number of controller is 6, *k*=6, and the algorithm is run 30 times. The parameter setting for implementing the algorithm on both scenarios were the same.

For both assignment methods, after each running of the CPGA, the maximum node to controller latency, $f(P)$, and the link load indicator, $LL(l)$, are captured as the performance metrics value. Finally, average, maximum and minimum of them are calculated. The results are presented in Table 10 for the propagation latency-based assignment and AHP-based assignment.

Table 10. Comparison of assignment methods

	Assignment method	MaxN_C Latency	Link load Indicator
Average	**Latency-based**	1.3152	8.137
	AHP-based	1.3913	2.519
Maximum	**Latency-based**	1.415	18.07
	AHP-based	1.4544	3.571
Minimum	**Latency-based**	1.141	2.73
	AHP-based	1.3351	1.043

As it can be seen from Table 10, although there is not much difference between the delays of the assignments induced by the resulted placements (compare the third column of the table), the link load indicator values (the forth column) show a highly different scenario. For instance, the maximum of the first metric, maximum node-to-controller latency, in latency-based assignment and AHP-based assignment is equal to 1.415 and 1.4544, respectively. While the maximum of the second metric, link load indicator, is equal to 8.137 and 2.519 for latency-based assignment and AHP-based assignment respectively. The proposed location-allocation method is going to minimize objective function, maximum node-to-controller latency, it improves link load balancing. Regarding only propagation latency in the assignment problem leads to a weak load balancing over the links along the path assigned between switch and controller.

MLUCP is another important metric to evaluate the quality of achieved placements, with respect to assigned control paths. The *MLUCP* denotes the highest link utilization among all control paths from switches to their assigned controllers. This metric is stated in Equation (10).

$$\mathrm{MLUCP(P)} = \max_{i,j\in V} DP_{ij} \qquad (10)$$

Where *P* represents a placement of controllers and DP_{ij} as the control load of switch *i* imposed by placement *P*. In other words, given a placement *P* as a set of controllers, the AHP-based approach determines the controller assigned to each switch *i* and also the control path. Each control path has a link utilization and $MLUCP$ denotes the highest link utilization among these paths. Therefore, for each AHP weight of Table 6, after each running of the CPGA on problem (6), the achieved solution is evaluated based on (10). For this weight, the algorithm is run for 50 times and the average values for the indicator (10) is captured as the performance metric value for this weight. Furthermore, we assume the control-flow arrival rate of every node is 100kpps. This assumption is made based on reference [25].

In the second part of this evaluation, the load distribution on the links is compared for different weights. Here, we are two types of topologies to compare; Internet2 (34 nodes) and Abilene (11 nodes) (the second column). The AHP weight vectors used in our evaluation are inserted in Table 11. The third column demonstrates values for vector (b, c, d), according to Table 2, each one corresponds to an input weight vector for Algorithm 7.

According to Table 11, AW1 and AW4 emphasize strictly on propagation delay, AW2 and AW5 impose the same preference for propagation delay and hop count but much less emphasis on link utilization, and AW3 and AW6 consider the same priority for the three criteria (propagation delay, hop count, and link utilization).

Table 11. Different weights for the AHP-based assignment

Notation	Topology	Weight Vectors
AW1		b=9, c=9, d=1
AW2	Abilene	b=1, c=5, d=5
AW3		b=1, c=1, d=1
AW4		b=9, c=9, d=1
AW5	Internet2	b=1, c=5, d=5
AW6		b=1, c=1, d=1

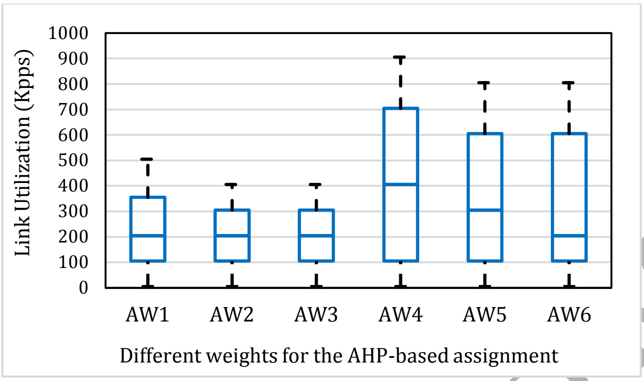

Figure 7. Link utilization distribution

Figure 7 shows the results. The below error bar shows the minimum usage of the links of a topology which are used by the control traffic. In a given topology, it is likely that there are some links which are not used at all by the control traffic, and in fact they are not involve in paths between switches and controllers, so their usage is zero. The middle line shows the mean and the high error bar differs between the third quartile and the maximum link utilization. The shorter the box is, the better the load balancing is done and the better traffic control is distributed. Moreover, the length of the box with its load is longer, the unbalanced load.

For example, for Abilene topology, the weight AW2 and AW3 have approximately the same and better performance than the weight AW1. For weight AW1, we can see that the maximum link utilization is 500kpps. Similarly, for the Internet2 topology, the weights AW5 and AW6 are approximately the same and have a better performance than the weight AW4. However, the weight AW6 outperforms AW5, because its mean is lower. As the number of nodes and links is higher, the weights show their properties better. We have looked into the control traffic distribution on links. It can be seen that that for both topologies, AW1 and AW4 constantly exhibit a wider spread in link utilization over two other preferences, implying that traffic on the links across the network is highly unbalanced, where in Internet2 network the variance reaches a peak of 23 percent.

In general, from the results, the AHP-based assignment shows a more reasonable performance compared to latency-based assignment. Because it creates somehow intermediate values for all considered metrics and can be regarded as a trade-off between minimizing delay based function values and load balancing purpose.

VI. CONCLUSION

In this paper, a new issue in Controller Placement Problem called *switch to controller assignment process* is presented. In order to assess the performance of the process, three important metrics including latency, hop count and link utilization are considered. These metrics are used to analyze the optimality of the controller's locations and also switch to controller assignment. Moreover, since a multi-criteria decision must be performed for the switch to controller assignment problem, we adapted Analytic Hierarchy Process (AHP) technique for the problem. In order to verify the proposed approach, several evaluations were investigated. Beside, a Controller Placement Genetic Algorithm that is improved by this new technique was proposed to solve a location-allocation problem in CPP. Running CPGA for different case studies revealed the efficiency of our proposed approach in determining the optimal location of controllers and optimal assignment. Results showed that the proposed multi criteria assignment approach performs better than latency based assignment in terms of link load balancing. To the best of our knowledge, this is the first study which investigated the influence of different metrics on switch assignment in CPP. This work may be motivated to navigate the pure controller placement problem towards the controller placement problem associated with assignment and routing.

REFERENCES

[1] R. Mohammadi, R. Javidan, M. Keshtgari, "OpenIPTV: a comprehensive SDN-based IPTV service framework." Multimedia Systems, pp.1-13, 2017.
[2] R. Mohammadi, R. Javidan, M. Keshtgari, R. Akbari, "A novel multicast traffic engineering technique in SDN using TLBO algorithm." Telecommunication Systems, pp. 1-10, 2017.
[3] A. Jalili, M. Keshtgari, R. Akbari, "Optimal controller placement in large scale software defined networks based on modified NSGA-II," Applied Intelligence, pp. 1-15, 2017.
[4] G. Wang, Y. Zhao, J. Huang, Y. Wu, "An Effective Approach to Controller Placement in Software Defined Wide Area Networks." IEEE Transactions on Network and Service Management, 15(1), pp. 344-355, 2018.
[5] M. Tanha, D. Sajjadi, R. Ruby, J. Pan, "Capacity-aware and Delay-guaranteed Resilient Controller Placement for Software-Defined WANs." IEEE Transactions on Network and Service Management, 2018.
[6] A. K. Singh, S. Srivastava, "A survey and classification of controller placement problem in SDN." International Journal of Network Management, Wiley Online Library, pp. 1-25, 2018.

[7] Y. W. Ma, J. L. Chen, Y. H. Tsai, K. H. Cheng, W. C. Hung, "Load-Balancing Multiple Controllers Mechanism for Software-Defined Networking," Wireless Personal Communications, vol. 94, no. 4, pp. 3549-3574, 2017.

[8] M. T. I. ul Huque, W. Si, G. Jourjon, V. Gramoli, "Large-Scale Dynamic Controller Placement," IEEE Transactions on Network and Service Management, vol. 14, no. 1, pp. 63-76, 2017.

[9] B. Heller, R. Sherwood, N. McKeown, "The controller placement problem," In Proceedings of the first workshop on hot topics in software defined networks, ACM, pp. 7-12, 2012.

[10] G. Wang, Y. Zhao, J. Huang, Q. Duan, J. Li, "A K-means-based network partition algorithm for controller placement in software defined network," In Communications (ICC), 2016 IEEE International Conference on IEEE, pp. 1-6, 2016.

[11] J. Zhao, H. Qu, J. Zhao, Z. Luan, Y. Guo, "Towards controller placement problem for software-defined network using affinity propagation," Electronics Letters, 2017.

[12] A. Sallahi, M. St-Hilaire, "Expansion Model for the Controller Placement Problem in Software Defined Networks," IEEE Communications Letters, vol. 21, no. 2, pp. 274-277, 2017.

[13] T. Zhang, P. Giaccone, A. Bianco, S. De Domenico, "The Role of the Inter-Controller Consensus in the Placement of Distributed SDN Controllers," Computer Communications, 2017.

[14] A. Jalili, V. Ahmadi, M. Keshtgari, M. Kazemi, "Controller placement in software-defined WAN using multi objective genetic algorithm," In Knowledge-Based Engineering and Innovation (KBEI), 2015 2nd International Conference on IEEE, pp. 656-662, 2015.

[15] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale SDN networks," IEEE Transactions on Network and Service Management, vol. 12, no,. 1, pp. 4-17, 2015.

[16] K. S. K. Liyanage, M. Ma, P. H. J. Chong, "Controller placement optimization in hierarchical distributed software defined vehicular networks." Computer Networks, 135, 226-239, 2018.

[17] X. Gao, L. Kong, W. Li, W. Liang, Y. Chen, G. Chen, "Traffic load balancing schemes for devolved controllers in mega data centers." IEEE Transactions on Parallel and Distributed Systems, 28(2), 572-585, 2017.

[18] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," Computer Networks, no. 112, pp. 24-35, 2017.

[19] Y. Hu, T. Luo, W. Wang, C. Deng, "On the load balanced controller placement problem in Software defined networks," In Computer and Communications (ICCC), 2016 2nd IEEE International Conference on, pp. 2430-2434, 2016.

[20] T. L. Saaty, L. G. Vargas, "Uncertainty and rank order in the analytic hierarchy process," European Journal of Operational Research, vol. 32, no. 1, pp. 107-117, 1987.

[21] T. L. Saaty, "Decision making with the analytic hierarchy process," International journal of services sciences, vol. 1, no. 1, pp. 83-98, 2008.

[22] F. P. Tso, D. P. Pezaros, "Improving data center network utilization using near-optimal traffic engineering." IEEE transactions on parallel and distributed systems, 24(6), 1139-1148, 2013.

[23] L. Liu, H. Mei, B. Xie, "Towards a multi-QoS human-centric cloud computing load balance resource allocation method." The Journal of Supercomputing, 72(7), 2488-2501, 2016.

[24] H. P. Jiang, D. Chuck, W. M. Chen, "Energy-aware data center networks." Journal of Network and Computer Applications, 68, 80-89, 2016.

[25] G. Yao, J. Bi, Y. Li, L. Guo, "On the capacitated controller placement problem in software defined networks," IEEE Communications Letters, vol. 18, no. 8, pp. 1339-1342, 2014.

[26] P. Xiao, Z. Y. Li, S. Guo, H. Qi, W. Y. Qu, H. S. Yu, "AK self-adaptive SDN controller placement for wide area networks," Frontiers of Information Technology & Electronic Engineering, no. 17, pp. 620-633, 2016.

[27] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, P. Tran-Gia, "Specialized heuristics for the controller placement problem in large scale SDN networks." In 27th International IEEE Teletraffic Congress (ITC 27), pp. 210-218, 2015.

[28] C. A. Wang, B. Hu, S. Chen, D. Li, B. Liu, "A Switch Migration-based Decision-making Scheme for Balancing Load in SDN," IEEE Access, 2017.

[29] T. Wang, F. Liu, J. Guo, H. Xu, "Dynamic SDN controller assignment in data center networks: stable matching with transfers," In Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on, pp. 1-9, 2016.

[30] G. Wang, Y. Zhao, J. Huang, Y. Wu, "An Effective Approach to Controller Placement in Software Defined Wide Area Networks." IEEE Transactions on Network and Service Management, 15(1), 344-355, 2018.

[31] J. F. Kurose, "Computer networking: A top-down approach featuring the internet," 3/E. Pearson Education India, pp. 50-190, 2005.

[32] M. Karakus, A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A Survey," Journal of Network and Computer Applications, 2016.

[33] A. V. Akella, K. Xiong, "Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN)," In Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on, pp. 7-13, 2014.

[34] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, E. Onem, "A qos-enabled openflow environment for scalable video streaming," In GLOBECOM Workshops IEEE (GC Wkshps), pp. 351-356, 2010.

[35] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, M. Roughan, "The internet topology zoo." IEEE Journal on Selected Areas in Communications, 29(9), 1765-1775, 2011.