ABSTRACT  
Software Defined Networking (SDN) makes it easy to control and manage the network due to a centralized control plane. However, this centralization also limits the scalability of SDN. To address this scalability issue, multiple controllers are deployed in the SDN. Placing multiple controllers poses several challenges, one of which is to find optimum locations for the placement of multiple controllers. The existing approaches for controller placement in a multiple-controller SDN environment overlook many aspects, like the path reliability for switch-to-controller and controller-to-controller communication, and the use of an efficient machine-learning algorithm. To address these issues, this paper proposes a novel approach, named as CMOPHA, that uses NSGA-II (a multi-objective optimization MOO algorithm) to compute the optimum placement of controllers based on the parameters of maximum switch-to-controller path reliability, and minimum value of switch-to-controller hop count, maximum controller-to-controller reliability, load-balancing among the controllers and a minimum number of controllers. We conduct simulations using real network traces. Based on the results, we show that CMOPHA improves the network performance in terms of end-to-end delay, hop count, computation time, and network availability compared to the existing state-of-the-art approach.

INDEX TERMS  
Software-defined networks (SDN), controller placement problem (CPP), NSGA-II, optimization, multi-objective (MOO) algorithm, evolutionary algorithm.

I. INTRODUCTION  
Software-Defined Networking (SDN) is an emerging paradigm in the modern networking system, in which the control plane is implemented at a logically centralized entity, called SDN controller, and the forwarding devices have the data plane functionality. The centralized control plane in the SDN is responsible for implementing the networking protocols and policies at the data plane (i.e. the forwarding devices) and has a network-wide view. The forwarding devices on the other hand have a simpler role, i.e., to forward network traffic according to the decisions/directions of the control plane. This centralized control plane of SDN makes it easy to control and manage the network. Moreover, it provides a higher level of abstraction to the network administrators and users to configure, control and manage the network from the centralized controller. It also provides improved security by allowing the network administrators to define and declare the security policies at the centralized controller. SDN also enables network evolution and allows innovation with ease, since the centralized control allows for incorporating new algorithms and protocols by implementing them at only the controller to satisfy a variety of network objectives [1]. Despite these advantages, SDN has several challenges, including scalability and reliability due to centralized controller [2], [3], [4], [5]. In literature, two major types of approaches have been introduced to address the SDN scalability issue; one at the data-plane level [6] and the other at the control plane or controller level [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26].
a. Data plane level SDN scalability: The SDN scalability can be improved by moving some functionalities of the control plane to the data plane (i.e., switches) which can be termed as data-plane level scalability. Such approaches offload the controller from the functions that are local to the data plane. For example, Difane discussed in [6] introduces intermediate switches known as the authority switches between the controller and the forwarding switches. The controller proactively computes the flow rules for the whole network and then stores them at the authority switches. A data plane switch consults an authority switch for flow rules instead of controller. Thus, Difane reduces the traffic over the controller and allows the controller to do other tasks and hence improving the controller scalability. Data plane scalability has also been addressed in [27] and [28].

b. Control plane level SDN scalability: In this approach, the SDN scalability can be improved by introducing multiple controllers [9]. This is done because the ever-increasing network size generates large data traffic and a single controller cannot handle huge number of flow requests made by the network switches, in the current era.

Our focus is on the control plane level SDN scalability issues. There are two types of control paths, switch-to-controller (S-C) and controller-to-controller (C-C) paths. In this regard, the existing approaches, e.g. [8], [9], [10], [16], [17], [18], [23], [29], compute the optimum placements for the multiple controllers. The optimum placements can be found based on several parameters or even a combination of different parameters such as controllers load-balancing, control-plane latency, switch-to-controller (S-C) latency, controller-to-controller (C-C) latency, switch-controller reliability, energy efficiency, link utilization between switches, network deployment cost, etc. This problem is generally referred to as Controller Placement Problem (CPP) [9]. It has been observed that among the various other parameters, link reliability appears as the most significant one for placing the controllers since it affects the network’s availability and ability to tolerate faults [3]. However, the link reliability as defined in [30] is overlooked by the existing approaches for CPP.

Though, link reliability as a parameter has been taken up in several research works. For example, the authors in [10] computed the expected percentage of control path loss based on the link reliability and solved the CPP such that the control path loss is minimized. Similarly, in [14], [16], [17], [18], and [19], the control plane reliability has been addressed in various ways. However, the existing approaches overlook the solution of CPP in a way that enhances the computation time, overall control path reliability, and control path hop count.

Based on the above observations, we suggest a constrained multi-objective evolutionary approach i.e., constrained NSGA-II [31] for CPP which improves not only the computation time but also the accuracy of the global network reliability and availability including the global S-C path reliability as well as the inter-controllers path reliability. Among the two types of control paths S-C and C-C paths, we focus on the reliability of both types of control paths via an in-band mode of communication in this research. We also introduce another parameter of hop count, measured by the number of hops between the switch and its assigned controller. We consider a small-sized to large-sized topology to evaluate the performance of our proposed approach using real network traces in [32]. To the best of our knowledge, the CPP problem using a constrained Multi-Objective evolutionary approach, i.e. constrained NSGA-II, in this case, has not been utilized in research.

We have termed our proposed approach as Constrained Multi-Objective Heuristic Placement Approach (CMOHPA). We will refer to it as CMOHPA for the rest of the paper.

We highlight the major contributions of this paper as follows:

1. In this paper, we have proposed a new parameter, the overall network reliability in terms of link reliability, which has not been utilized for CPP in the existing literature, to the best of our knowledge. The proposed metric measures the minimum reliability of all paths between a Switch-to-Controller (S-C) and tends to find a path that maximizes the reliability of that path. This parameter is applied to both control paths, i.e. Switch-to-Controller (S-C) and Controller-to-Controller (C-C) paths. This represents the network reliability more accurately. This is described in detail in Section III.

2. A constrained MultiObjective algorithm CMOHPA is proposed in this paper, which considers the constraints such as (C-C path reliability, and load-balancing among controllers) as distinct measures without merging them with the two objectives since they are contrasting. To the best of our knowledge, this is the first work to utilize the constrained MOO techniques to address CPP. This is explained in detail in Section IV.

3. Our proposed approach not only successfully achieves a converged set of best solutions but also converges it further to a single best solution using a reference point approach. To the best of our knowledge, we are the first to use this for CPP. The best solution refers to the most appropriate multiple controller placement. This is done in the second phase of the proposed CMOHPA and is described in detail in Section IV.

4. To evaluate the performance and significance of our proposed approach compared to the existing state-of-the-art approaches, we utilized real network traces, Panopticon (based on Panopticon [32]). This includes not only real topology but also real network traffic. This tool, to the best of our knowledge, has not been utilized for any CPP problem. The evaluation process is discussed in Section V.

5. We have compared the proposed CMOHPA with an existing state-of-the-art technique termed as Greedy Placement Algorithm (GPA) and found out that the proposed solution provides faster and comparable results in terms of both objectives. The comparison results are depicted in Section V.
The remaining sections of the paper are organized in the following manner: Section II reviews the related work. Section III is concerned with mathematical modeling and formulation of the problem. Section IV describes the proposed algorithm CMOHPA in detail which also entails the convergence technique for the final results. Section V describes the simulation evaluation and analysis of the results of our proposed approach and finally, in Section VI the conclusion of the paper have been presented.

II. RELATED WORK

In this section, we describe some important existing approaches for CPP in SDN.

There are several research works that based their placement approach using the parameters under consideration such as reliability, hop count, and load-balancing among controllers. We will identify some of the relevant approaches which focus on network reliability, hop count, and/or multi-objective optimization (MOO) algorithms.

As categorized in [4], the relevant link reliability has been addressed in various research works through different methods. We have identified four ways in which reliability has been addressed in the existing literature, some of which have been discussed in [4]. The first reliability parameter is to minimize the control path which has been addressed in some of the research works such as in [10]. The second type of reliability parameter is to assign multiple control paths for each S-C communication by allocating at least two control paths to each switch to its respective controller such as in [33]. The third reliability parameter is to assign multiple controllers to a switch as a backup in case of a S-C link failure such as in [23]. The fourth is to minimize controllers for k-link failures depending upon the control path originating from each switch having a minimum failure distribution. This is discussed in [12] and [34].

Some of the relevant approaches have been discussed in a brief account in the literature review below.

Muller et.al in [33] have proposed an algorithm for multiple control paths which makes the switches to be able to shift to another alternate disjoint path with their controller if S-C path fails. They have designed a problem that aims to maximize the number of disjoint paths between switches and their respective controllers with enough capacity. This enhances the connectivity among the S-C paths to up to 66%. They have modeled the problem using Integer Linear Programming (ILP) [35] which is exhaustive and slower. However, they ignored other parameters such as C-C reliability, load-balancing. Also, they did not utilize any Multi-Objective Optimization (MOO) algorithm for solving this problem.

Another method for improving the link reliability is to minimize the control path between S-C. There are several research works that have taken into account this parameter. Such as in [15] Jimenez et al. argued that finding more than the required optimal number of multiple controllers to solve the CPP is not economical and efficient. Therefore, they focus on selecting an optimum number of controllers as an objective such that it improves the connectivity, and reliability and achieves neither a minimum nor a maximum delay but a suitable or required delay between S-C. For achieving these objectives, a K-Critical algorithm has been proposed. The K-Critical algorithm initially proposed in [29] is based on the idea that K number of controllers are to be placed in the network in such a way that the network faces minimum loss of management data by considering the network as a tree with each branch managed by a controller having a balanced number of nodes. To ensure that the controllers are equally distributed in the network, a critical node is identified at the farthest distance from each node and the delay from the critical node is computed, which should be within the limits of a pre-defined delay threshold for the network. They argue that this improves that joint control plane reliability and delay. Their proposed approach, however, compromises over the minimum S-C delay for efficient communication within a branch and overlooks the effect of network performance in case of a link failure. MATLAB was utilized for the evaluation of the proposed problem while working on a tree topology. This approach does not consider link reliability between S-C or C-C control path. Also, they have not proposed any MOO-based algorithm, although they have proposed multiple metrics to place the multiple controllers in the network. The authors did not utilize any real network traces for the evaluation of their proposed approach.

Also, Yanan et al. in [10] introduced a parameter termed as the expected percentage of control path loss where the control path is represented by S-C path only. The authors have measured the expected percentage of S-C path loss parameter based on the physical link failure probability of the individual links along the path and the number of control paths. Although the placement approach presented in this work is reliability aware, it focuses on a model that bases its formulation on measuring the failure probability of physical links and minimizing an overall proposed expected percentage of control path loss metric value which is dependent upon the number of control paths owned by each node. This metric only relates to the probability of link failures in the set of control paths and the impact of the failure of the link and switch. It overlooks the individual control path reliability of the multiple controllers such as what happens if the control path loss of one or more controllers is higher. It also overlooks other parameters (like load balancing, switch-to-controller delay, etc.) for placing the controllers. Moreover, their proposed approach utilizes Integer Linear Programming (ILP) [35] which is exhaustive and slower and does not use any MOO-based algorithm.

Another major category of considering path/link reliability for CPP involves the utilization of MOO algorithms. However, we have found very limited literature that focuses on utilizing MOO algorithms to address link reliability, hop count or delay, load-balancing, etc. But since our prime objective is to utilize a MOO algorithm, that is why we have presented an explicit review of all the MOO based algorithms to address CPP.
Borcoeli et al. in [36] have discussed how multi-criterion solutions can be opted to solve CPP. Although they have proposed the idea that the various parameters can be utilized in multi-criterion optimization algorithms such as average latency, worst-case latency, node reliability, path reliability of S-C, energy efficiency etc. but they did not work on any experimental analysis.

Ahmadi et al. in [37] have utilized multiple objectives one of which is the sum of average of S-C path delay and the other is worst-case S-C path delay. The authors have presented a modified version of NSGA II to solve the given problem. This modification focuses on improving the time and space complexity of the original algorithm. The authors have argued that the proposed algorithm provides enough diversity among the solution set and reaches the desired Pareto optimal solutions set with relative accuracy. The algorithm runs up to 104 iterations/generations. The results, however, were not shown for intermediate iteration sizes. Also, the entire research is more focused on the reduced time and space complexity and lesser on the quality of the solution set. The authors have not utilized elitism for the computation of the next generation, the outcomes of which were not discussed. Internet2 OS3E topology has been utilized to test the working of the algorithm using MATLAB. The authors in this work have overlooked the link or path reliability between S-C and C-C control paths. They also did not propose any load-balancing scheme. Also, no real network traffic-based traces were not utilized for the evaluation of the proposed scheme.

Bannour et al. in [38] have subdivided the problem into two main parts each of which is specified by individual objective functions. These have been proposed to find optimal locations for placing the multiple controllers. One of the metrics involves a set of parameters that improves the performance of the control plane. These include latency-related performance metrics such as average latency between S-C and maximum latency between S-C along with a load-balancing scheme. The other metric they have utilized is reliability and the authors consider the response time of the network after a failure as an indicator of its reliability. To solve this problem, the authors have utilized PAM-B algorithm (Partitioning Around Medoids) [39] for the first criterion and NSGA-II for the second criterion. Java-based simulator Sinalgo is utilized to solve the problem. However, this paper completely ignores the reliability of the network. Also, no comparison is included with any existing work. The authors did not utilize any real network-based traces for the evaluation of the proposed approach. Also, the parameters utilized in this approach completely overlook the link reliability between S-C or C-C paths.

Ahmadi et al. in [7] have utilized multiple objectives for the computation of optimal placements for multiple controllers. These objectives include the maximum and average-case S-C latency, and they also included maximum and average-case C-C latency. So, the aim is to improve the overall network delay in the entire control path in this work. The authors have proposed the utilization of an adaptation of the Non-dominated Sorting Genetic algorithm (NSGA) [40] Multi-Start Hybrid Non-dominated Sorting Genetic algorithm (MHNSGA) to solve the problem. This algorithm modifies the input which is specified as GreedyStart i.e., it does not get completely random initial solutions and in fact tries to initialize with a set of partial placements which is checked for its fitness using a function. The results are evaluated using the topologies from Internet topology zoo in MATLAB. However, they have overlooked the reliability of the network's control path. Also, no load-balancing scheme was proposed for the placement of multiple controllers. They did not utilize any real network traffic-based traces for evaluation.

Jalili et al. in [41] have further improved the model of [37]. They have utilized the S-C average delay and worst-case delay as their two objectives. They have proposed a Multi-Start Hybrid Non-dominated Sorting Genetic-II Algorithm (MHNSGA-II) to solve the CPP. They put forward a model that intends to minimize the parameters of the average delay for S-C control paths and the worst-case delay of the S-C control paths. They have developed an adaptation of NSGA-II [42] which is capable of finding global optimal placements by reducing the chances of getting stuck in local minima. This algorithm operates on modifying the input which is specified as GreedyStart which is based on the concept presented in [7]. MATLAB was utilized for evaluating the results. The authors have, however, ignored the control plane reliability and focused solely on control plane delay and no load-balancing approach was proposed. Also, they utilized MATLAB which does not support real network traffic-based evaluations.

Zhang et al. in [22] have worked on three objectives which include overall network reliability, load-balancing among the controllers with respect to the flow requests each controller receives, and control path latency which corresponds to worst-case latency between switch-to-controller in this paper. The individual reliable path is dependent upon the measure of the probability of a valid path between a switch-to-controller control path. The overall network reliability involves a parallel combination of all valid path reliabilities. They have utilized an Adaptive Bacterial Foraging Optimization (ABFO) algorithm to solve the defined Multi-Objective Controller Placement MOCP problem. The ABFO is inspired by [43] and the authors have suggested modifications to improve the 'chemotaxis' operation in the algorithm. This is utilized to enhance the global search operation to find a suitable placement solution. The proposed model is evaluated using MATLAB using the topologies from Internet topology Zoo which does not support real network traffic. The authors, however, only included S-C paths in the control path and ignored the C-C paths. They also did not propose any method for load-balancing among the controllers.

Singh et al. in [25] have utilized a parameter, the overall average latency between S-C and C-C, and combined the capacity of the controller to control traffic with these objectives. They aimed at minimizing this average latency of the control path. They have developed a heuristic model
to describe the problem termed as Varna-based optimization (VBO) inspired from [44]. For computation purposes, they have compared three algorithms namely, particle swarm optimization (PSO) [45], teacher learning-based optimization (TLBO) [46], Jaya [46] with their proposed approach VBO. The authors, after evaluation, suggested that VBO algorithm performs better compared to the other three algorithms. Evaluations were performed using Internet2 OS3E topology in MATLAB which does not include real network topology and traffic. The authors did not consider the link failure while assigning switches to controllers, and neither did they propose any load-balancing method for S-C paths.

Jalili et al. in [14] have computed the CPP based on two main objectives of C-C latency and end-to-end flow setup time [21], along with the constraints of limiting the hop count between S-C and link utilization such that the control flow does not exceed a threshold bandwidth. Though, they utilized a multi-objective optimization algorithm, i.e. a variation of NSGA II referred to as Non-dominated Sorting Moth Flame Controller Placement Optimizer (NS-CPMFO) [47]. MATLAB was utilized to evaluate the results on the topologies from Internet topology zoo. This does not include real network traffic. Also, they did not consider other parameters like the reliability of S-C path and C-C path or any load-balancing scheme.

Tao et.al in [12] have proposed a multi-objective scheme to address the CPP. They have proposed two objectives based on which the optimal placements for multiple controllers have been found. One of these objectives is minimizing the number of controllers to be placed under a finite number (k) of link failures and the other is minimizing the worst S-C delay. The authors have presented a metric termed link failure rate which is measured by tracing the real-network link failure distribution. They have also proposed other constraints such as the load of a controller cannot exceed its capacity and a threshold has been defined beyond which the C-C delay cannot exceed. However, both of these constraints are not treated as independent and they are added to the objective functions to compute the fitness of both of them. The authors have utilized an enhanced NSGA-II algorithm and they have termed it a heuristic controller placement algorithm (HCPA) which includes the modifications in classical NSGA-II such as the computation of Non-Dominated Sorting (NDS) stage, a dynamic competition is introduced among solutions which finds out appropriate NDS solutions with reduced time and space complexity. Also, they have reduced the time and space complexity of the computation of crowding distance (CD) in the algorithm. They did not propose any load-balancing mechanism in this work. They have evaluated the results using Internet topology zoo in Python environment which does not include real network traffic. We have tried to overcome the following technical challenges in this work:

- One of the key technical challenges involved in order to propose the model of our problem is the accurate representation of the network reliability. Since the idea is to place the controllers in a way that specifically improves the network reliability, we need a better representation of the reliability parameter which has not been addressed in the existing literature as described in Section II. Although there are a few methods proposed in the literature specified in Section II to improve the network reliability but none of them gives a measure of the overall network reliability. Hence, we design our problem in a way that we can measure its reliability parameter and maximize it to improve the network's availability. Our approach treats the two metrics (i.e. reliability of S-C path) and hop count of the S-C path as independent parameters along with other constraints (like the load balancing among the controllers, maximum reliability of C-C path, and a finite number of K controllers. This is an improvement over the existing work [22].

- Another significant challenge involved is, the modeling of our target problem so that we can compute it correctly and efficiently. In this regard, the existing literature mostly utilized exhaustive searching techniques such as Integer linear programming [35], Mixed integer programming [8], Simulated annealing [17] etc. These techniques focus on single-objective optimization problems which can provide a single best solution focusing on a single criterion. Sometimes multiple criteria can also be combined in a single objective but it does not effectively compute the objectives which are contrasting. Also, these techniques are mostly exhaustive causing them

### TABLE 1.

| Ref. No. | Large-Scale Network | Link Reliability | IHC | Link Rel. & IHC | Constrained MO | Evaluation (based on) |
|----------|---------------------|------------------|-----|----------------|-----------------|---------------------|
| [36]     | ✓                   | ✓                | ✓   | ✓              |                 | None               |
| [37]     | ✓                   | ✓                | ✓   | ✓              |     ✗           | INSGA-II (NSGA-II)  |
| [38]     | ✓                   | ✓                | ✓   | ✓              |     ✗           | NSGA-II            |
| [22]     | ✓                   | ✓                | ✓   | ✓              |     ✗           | ABFO               |
| [7]      | ✓                   | ✓                | ✓   | ✓              | ✗              | MNSGA (NSGA)       |
| [41]     | ✓                   | ✓                | ✓   | ✓              | ✗              | MNSGA-II (NSGA-II) |
| [23]     | ✓                   | ✓                | ✓   | ✓              | ✗              | VBO                |
| [14]     | ✓                   | ✓                | ✓   | ✓              | ✗              | NS-CPMFO           |
| [12]     | ✓                   | ✓                | ✓   | ✓              | ✗              | (HCPA)NSGA-II      |
| Our Approach | ✓                   | ✓                | ✓   | ✓              | ✓              | (CMHPA) Constrained NSGA-II |

### Notes:

- **HC**: Hop count.
- **HC**: Controller to Controller, S-C: Switch/Node to controller.
to have higher time-complexity [48]. Apart from these, several heuristic approaches have also been utilized such as in [24] and [25] etc. Heuristic approaches though faster are problem-dependent and may or may not generate approximate results. Evolutionary approaches generally considered meta-heuristic techniques have also been utilized such as in [49] and many more. They are applicable to a wide-range of problems since they are problem-independent. Since we are also proposing other parameters along with network reliability to be addressed, so a machine learning-based Multi-Objective Optimization MOO algorithm which is a special type of evolutionary technique is more appropriate for modeling our problem. Our target is to apply similar priority to our proposed contrasting objectives which makes the MOO algorithm such as NSGA-II suitable for our problem. Unlike the approaches e.g. in [7], [12], [14], and [22] our proposed model finds optimal placement locations for multiple controllers using a constrained Multi-Objective MOO algorithm [50] and [51].

- Another challenge involved the most appropriate choice of machine learning algorithm which in our case is NSGA-II. However, traditional NSGA-II algorithm generates a set of best possible solutions after a suitable number of iterations, which is not desirable for the CPP. We are interested in only one best placement scheme for the multiple controllers. So, we devise a way to converge the set of best solutions to a single best placement scheme using a reference point approach [50] described in detail in Section IV. Also, we have utilized a method to handle constraints separately without merging into the objective functions as described in [42] and [52] specialized for NSGA-II.

The comparison of proposed technique with the existing ones is summarized in Table 1.

### III. MULTI-CONTROLLER PLACEMENT MODEL AND PROBLEM FORMULATION

The CPP in SDN is an optimization problem falling under the NP-Hard problems [9].

In this section, we formulate our problem in the form of a mathematical model using CMOHPA based on NSGA-II [50] and [51]. Here, we also introduce the novel reliability parameter we have utilized to place the multiple controllers in the network. The working of our proposed model and our proposed reliability parameter can be understood using a scenario presented in the following subsection III-A.

#### A. EXAMPLE TOPOLOGY

We first present a scenario using an example topology to explain the computation of the parameters, and based on these computations we generate a reliable network.

Suppose we have a network scenario given in Fig. 1a which consists of a finite set of 5 nodes (possible controller and switches locations) and 6 links connecting these nodes. For this example, we try to place at least 2 controllers in the network at locations $l$ and $j$ Fig. 1b. This means that the remaining locations $i$, $k$ and $f$ will be allocated to one of these controllers according to our model. Each link is described by different weights representing the reliability of each path, $[R_{i,j}, R_{j,k}, R_{k,i}, R_{i,f}, R_{f,j}$ and $R_{j,l}]$ such that $R_{i,j} > R_{j,f} > R_{k,i} > R_{i,f} > R_{j,l} > R_{l,k}$. This corresponds to a single placement of multiple controllers in this simple network topology represented in Fig. 1. Our goal is to allocate the most suitable controller depending on the most reliable paths to each available switch location. For the switch, $i$, the possible path to controller location $j$ is $\phi_{i,j}$. Similarly, from switch $i$ to controller location $l$, the possible paths are $\phi_{i,l}$ as represented in Fig. 1b. We compute the two functions, maximum of minimum reliability and hop count of all of these paths and allocate the switch $i$ to any controller via a path having the maximum reliability and minimum hop count. We shall define these functions in the following subsection III-C. In the same manner, we find the suitable controller to allocate to the remaining switches $k$ and $f$. These computations are tabulated in Table 2. The control paths existing between the two controllers $l$ and $j$ are also described in Fig. 1b and since it is an in-band mode of communication, so the control traffic and data traffic may utilize the same medium or network paths [53].

There are two types of modes for switch to controller communication.

- **In-band** mode refers to the one in which the control plane and data-plane traffic routes within the same medium or network paths [53].
- **Out-of-band** mode refers to the one in which the control traffic between the switch and controller utilizes a separate/dedicated path to communicate to the network [53].

For this research work, our focus will be on the in-band mode, since we are considering a large network. In-band mode is suitable for large networks [54]. Therefore, a controller can take the same location as that of the switches by attaching the controller to that switch. There are two types of control paths, switch-to-controller and controller-to-controller paths. In this research, we focus on reliability of the both types of control paths via the in-band mode of communication. Second, we assume that every link has a reliability value
K. Saeed, M. O. Ullah: Toward Reliable Controller Placements in Software-Defined Network

FIGURE 1. Example topology with 5 nodes and 4 different possible controller locations having 2-to-3 possible paths $\phi$ computed for each controller location from available switch nodes $i, k$ and $f$ are termed as $\phi_1, \phi_2, \phi_3$ etc.

TABLE 2. Reliability and Hop Count (HC) of each path between a switch to its respective controller where, $R_{i,j}$, $R_{j,f}$, $R_{k,l}$, $R_{l,f}$ in relation to Fig. 1 and Best location $=$ $\{\max \phi_k [R], \min \phi_k [HC]\}$, where, $k = 1, 2, \cdots, M$.

| Switch Loc. | Cont. Loc. | $\phi_1$ Reliability | $\phi_2$ Reliability | $\phi_3$ Reliability | $\phi_3$ HC | Best Path | Best Cont. Loc. |
|-------------|------------|----------------------|----------------------|----------------------|------------|-----------|----------------|
| $i$         |            | $1$                  | $2$                  | $3$                  |            | $\phi_3 (k - f_2)$ | j               |
| $j$         |            | $1$                  | $2$                  | $3$                  |            | $\phi_3 (k - f_1)$ | j               |
| $k$         |            | $1$                  | $2$                  | $3$                  |            | $\phi_3 (k - f_2)$ | j               |
| $l$         |            | $1$                  | $2$                  | $3$                  |            | $\phi_3 (l - f_3)$ | j               |

FIGURE 2. Single placement 'P' representing 'z' nodes in the form of binary values. '0s' represent switches and '1s' represent controllers.

and the switches share it as a link state information with the controller. The computation of link reliability is out-of-scope of this work. We assume that a node (like switch/controller) itself does not fail. The link failures are the most frequently occurring phenomenon, in contrast to the node failure [10].

C. MATHEMATICAL MODELING

We represent the network as an undirected and connected graph $G = (V, E)$. The set $V$ represents the set of nodes where there are three types of nodes; the set of switches $S$, set of hosts $H$ and the set of controllers $C$. Thus $V = H \cup C \cup S$. $E$ represents the set of edges/links that connect the nodes in $V$. The other symbols used in our proposed approach are defined in Table 3.

For the rest of the paper, we represent the variables 'h', 's' and 'c' such that, $h \in H$, $s \in S$ and $c \in C$. To represent their respective location in the network graph $G$, we describe a subscript alphabet to every 'h', 's' and 'c' where, a different alphabet shows a different location such as $i \neq j \neq f \neq l$ etc. Also, we assume that a different variable with same subscript represent different nodes but directly connected to each other, if the path exists between them.

For $E$, we assume that the edges between the nodes $h_i$ and $h_j$, $c_i$ and $h_i$, $c_i$ and $c_j$ do not exist as given in Equation 1. However, the edges between $h_i$ and $s_i$, $s_i$ and $c_i$, $s_i$ and $s_j$ do exist as described in the Equation 2.

$$\langle h_i, h_j \rangle = \emptyset, \langle c_i, h_j \rangle \emptyset, \langle c_i, c_j \rangle = \emptyset \quad (1)$$

$$\langle h_i, s_i \rangle \in E, \langle s_i, c_i \rangle \in E \text{ and } \langle s_i, s_j \rangle \in E \quad (2)$$

The path (if there exists) between switch $s_i$ to a controller $c_j$ can be defined as the set of all nodes (switches) that exist within that path which is expressed as follows,

$$\phi_{si-cj} = \{(s_i, s_f, \ldots, s_j) \exists s_i, s_f, s_j \in V \quad (3)$$

where, $s_j$ is the last switch of the path directly connected with $c_j$. The path defined in Equation 3 consists of set of edges given by,

$$e_{si-cj} = \{(s_i, s_f, \ldots, (s_k, s_j)) \exists s_i, s_f, s_k, s_j \in V \quad (4)$$
In Equation 7, we find out the minimum reliability values of \( s \) link reliability. For each \( s \) link via Prepass-Flow technique presented in [30], but the relevant switch location for which the best reliability to a controller \( c \) is to be computed.

Among all paths originating from each switch to a controller \( c \), we can compute reliability of each path \( \phi_{s_i \rightarrow c_j} \) and set this with \( r_{s_i \rightarrow c_j} \). The local objective of computing the maximum of the minimum reliability of the set of paths originating from each switch to a controller \( c_j \) is by finding the maximum of all the paths’ minimum reliabilities as defined in Equation 8.

\[
\Gamma_{s_i \rightarrow c_j} = \max_{\beta=1,2,...,n} \{ r_{s_i \rightarrow c_j} \} \tag{8}
\]

where, \( \Gamma_{s_i \rightarrow c_j} \) describes, the overall reliability of the best path from switch \( s_i \) to controller \( c_j \). We have utilized a subscript \( \beta \) to represent the switches in Equation 8 which takes up values from 1 to \( n \) since this equation is operable on the set of all switches \( S \). Similarly, we define another parameter of hop count, \( h_{c_j} \), of a path which represents a finite number of intermediate switches existing in the shortest paths between switch \( s_i \) and \( c_j \). It is represented in Equation 9.

\[
h_{c_j} = |\phi_{s_i \rightarrow c_j}| \forall(s_i, c_j \in V) \tag{9}
\]

We also present here the global representation of the parameter Hop Count \( HC \) of all paths originating from \( n/K \) switches to \( K \) controller. This is defined in Equation 10.

\[
HC_{s_i \rightarrow c_j} = \sum_{\alpha=1}^{K} \sum_{\beta=1}^{n/K} (h_{c_j}) \tag{10}
\]

For Equations 10, 11 and 13 we introduce two new subscripts \( \alpha \) and \( \beta \) which do not necessarily represent the location of the relevant switch or controller (which have been represented by small-case subscripts previously) but they are utilized to represent the range of possible locations for switches and controllers respectively.

### D. OBJECTIVE FUNCTIONS

The goal of our proposed model is to find optimal locations for a finite number of \( K \) controllers in such a way that the reliability of the S-C path is maximum, it has minimum number of hops, the C-C path has maximum reliability and also the load is balanced over the S-C paths. So our problem can be considered multi-objective optimization (MOO) in nature. To solve this problem, we propose a Constrained Multi-Objective Heuristic Placement Approach (CMOHPA).

It is formulated as follows.

First, we redefine the entire search space which consists of \( z \) nodes including \( n \) possible switches and \( K \) possible controllers, in the form of a binary search space. In this, the
'0s' represent the switches and '1s' represent the controller locations. A single alternative placement $P$ has been shown in Fig. 2. For a placement scheme, $P$ in which we have to place $K$ number of controllers in $V$ possible locations, the objective functions $\Delta$ and $\Psi$ are described as in Equations 11 and 12 respectively. As shown in Fig. 1, we can have multiple alternatives for placing the controllers in the network depending upon the number of switches and controllers. We compute the reliability, hop count, and load balancing of S-C path for all possible alternatives of controller placement in the network. A placement scheme $P$ represents one alternative for placing the controllers in the network; an example of which has been described in detail in Fig. 1.

$$\Delta_{s_i-c_0}(P) = \max_{\alpha=1,2,..,K} \min_{b=1,2,..,n/K} \{\Gamma_{s_i-c_0}^\alpha\} \cdot Y_{c_0} \cdot Z_{s_i-c_0}$$

and,

$$\Psi_{s_i-c_j}(P) = \min_{i\in S, j\in C} \{HC_{s_i-c_j}\} \cdot Y_{c_j} \cdot Z_{s_i-c_j}$$

with constraints:

$$\Delta_{c_0-c_0}(P) = \max_{\alpha=1,2,..,K} \min_{b=1,2,..,n/K} \{\Gamma_{c_0-c_0}^\alpha\}$$

$$\sum_{c_j\in V} Y_{c_j} = K$$

$$\frac{1}{K} \sum_{j=0}^{n} \sum_{i=0}^{K} X_{s_i,c_j} \leq \text{ceil}\left[\frac{n}{K}\right]$$

where, $\Delta_{s_i-c_0}$ in Equation 11 represents the maximum of minimum reliability of all S-C paths for one alternative of controllers placement which is based on Equation 8 (which is defined for a switch to be allocated to a controller). In this the minimization is applied to a single controller location and $n/K$ switches (to achieve load-balancing) represented by $\alpha$. The maximization is applied to $K$ controllers represented by $\alpha$. The decision variables are switch $s_i$, controller $c_0$, the binary variable $Y_{c_0}$ and mapping matrix $Z_{s_i-c_0}$.

$\Psi_{s_i-c_j}$ in Equation 12 represents the minimum hop counts of each S-C path of one alternative of controllers placement between switch $s_i$ to controller $c_j$.

And Equation 13 ensures a controller-to-controller reliable path between controllers $c_i$ and $c_d$ which can acquire any location depending on the proposed approach.

In Equation 14, $Y$ is a binary variable such that, $Y_{c_j} = 1$, if a controller is to be placed at $c_j$, otherwise $Y_{c_j} = 0$. Equation 14 ensures finite controllers $K$ to be placed in the network.

$Z_{s_i-c_j}$ in Equations 11 and 12 respectively, represents a matrix of mapping relationship between a switch $s_i$ and controller $c_j$ such that $Z_{s_i-c_j} = 1$ only if $s_i$ is controlled by $c_j$. It can also be represented as, $Z = [Z_{s_i-c_j}]_{K \times n}$.

Equation 15, ensures a load balancing among the controllers such that equal number of switches $n$ are attached to finite controllers $K$. Where, $X$ represents another binary variable such that $X_{s_i,c_j} = 1$, if $s_i$ is controlled by $c_j$. Otherwise, $X_{s_i,c_j} = 0$. The equation shows that the value of $X$ will be computed separately for each controller $c_j$.

If the reliability of S-C path is maximum, its hop count is minimum and the load is balanced over S-C paths, and C-C path has maximum reliability, then the corresponding switch $i$ becomes a part of the domain controlled by $j_{th}$ controller, i.e. the switch $i$ is assigned/connected to $j_{th}$ controller.

Before we try to solve the problem, we apply normalization to the objective functions for consistent scaling in a way that both objectives fall in the same range. For this process, we observe that we can apply normalization to the objective function $\Psi_{s_i-c_j}$ so that it can have the range $[0,1]$ as described in Equation 16.

$$\Psi_{s_i-c_j} = \frac{\Psi_{s_i-c_j} - \Psi_{s_i-c_j_{min}}}{\Psi_{s_i-c_j_{max}} - \Psi_{s_i-c_j_{min}}}$$

To solve the concerned MultiObjective CPP Problem using a Constrained Multi-Objective Heuristic Placement Approach (CMOHPA), we have modified the classical approach of the NSGA-II algorithm. In the following Section IV, the working of the proposed Constrained Multi-Objective Heuristic Placement Approach (CMOHPA) has been described.

**IV. PROPOSED CONSTRAINED MULTIOBJECTIVE HEURISTIC PLACEMENT ALGORITHM, CMOHPA FOR CPP**

The two objective functions described in Section III-D have range of values as shown below.

$\Delta_{s_i-c_j}$ is comprised of probability values which range from $\{0,1\}$, $\Psi_{s_i-c_j}$ is comprised of integer values that range from $\{1,w\}$ ($w$ is a finite number that varies from topology to topology). We have normalized $\Psi_{s_i-c_j}$ as described in Equation 16.

According to the definition in Section III-D both of these objectives are independent of each other. However, both objectives present a tradeoff with each other, i.e., if $\Psi_{s_i-c_j}$ is high, then there is a higher probability that $\Delta_{s_i-c_j}$ decreases. We need to maximize $\Delta_{s_i-c_j}$ and we need to minimize $\Psi_{s_i-c_j}$.

NSGA II first introduced in [42] is a non-dominated sorting based evolutionary technique with a fast elitist optimization method and is capable of handling multiple objectives simultaneously. The algorithm is specialized in providing optimal results which are not dominated by any other possible results [56] and [57]. NSGA II tends to achieve close to optimum results with polynomial-time complexity. Several researchers such as in [56] prefer this genetic algorithm to support multi-objectives optimization due to its improved time-complexity and reliable results. We have modified this algorithm according to our requirements as follows. First, we subdivide the working of our algorithm CMOHPA into two phases as follows.

**A. PHASE 1: ACHIEVING PARETO-OPTIMAL SOLUTIONS**

In this phase, we apply the constrained NSGA II algorithm, which considers all possible controller placement schemes in the network for $K$ number of controllers and produces as output, a set of the best controller placement schemes
which are not dominated by any other controller placement schemes. This set of controller placement schemes is termed as Pareto-Optimal Solutions. At the end of a single simulation, we arrange all the individuals (placement schemes) included in a generation in such a way that the Pareto-Optimal Solutions appear at the first front of the 2-dimensional plot (corresponding to our 2 main objectives) and the second best appear in the next front and so on. We have shown this in Fig. 3. To acquire the Pareto-Optimal solutions, we define the following eleven steps:
1) STEP 1: INPUT
We take the entire network graph \( G \), which is defined above, as input for the processing of our proposed algorithm with its \( Z \) matrix representing the switches from \( S \) with which the controllers can be attached. For \( K \) number of controllers to be placed in the network with \( N \) number of possible switches in a graph consisting of \( Z \) nodes can have the number of rows representing \( \binom{N}{K} \) possible placements. A single \( I_{th} \) placement, can be represented by \( P_i \) as a vector as shown in Fig. 2.

2) STEP 2: RANDOM INITIALIZATION OF POPULATION
Our search space consists of up to the number of rows in \( Z \) representing the possible controllers’ locations identified with an integer label. For correlating the constrained NSGA II algorithm with our problem, we transform our search space in the form of binary labels i.e., either ‘0’ or ‘1’ with ‘0s’ representing the switch without controller location and ‘1s’ representing the possible the switch location attached to a controller as depicted in Fig. 2. We initialize the population by randomly selecting \( r \) number of placement schemes among the total \( \binom{N}{K} \) possible placements called the first generation, such that \( C_X = \{P_1, P_2, \ldots, P_s\} \).

3) STEP 3: SET COUNTER FOR GENERATION
At this stage, we initialize a counter for the generation. Initially, the counter is set to 0. We run the algorithm for a finite number of times and at end of each simulation, the generation counter is incremented by 1. The algorithm is designed in such a way that each time at the end of a generation the set of solutions or best individuals achieved are better than the previous generation and trending towards the required objective values. However, there can be a suitable limit to when the algorithm should stop, which varies according to the requirements.

4) STEP 4: EVALUATE OBJECTIVES AND CONSTRAINTS
For any placement \( P_i \in C_s \), we compute the objective functions described in Section III-D. We compute \( \Delta_{x_i-c_j}(P_i), \Psi_{x_i-c_j}(P_i) \) described in Equations 11 and 12 also termed as the fitness values of each placement or individual and we also evaluate constraints \( \Delta_{s_i-c_j}(P_i), Y_{c_j} \) and \( X_{s_i-c_j} \) for each individual placement scheme as defined in Equations 13, 14 and 15. We apply normalization to the fitness function \( \Psi_{x_i-c_j}(P_i) \) as described in Equation 16.

**a: CONSTRAINT HANDLING TECHNIQUE**
For the constraint handling, we evaluate the constraints along with the objectives in Step 4 of Fig. 3. We check the solutions which fulfill the objectives and constraints and discard the others. There are multiple techniques which have been discussed in the literature for constraint handling in MOO problems. Such as in [58] several techniques have been discussed which are suitable for use in Evolutionary Algorithms. The constraint handling in MOO problems are complex tasks and increases the time-complexity of the relevant algorithms, but these trade-offs can be compared with the usefulness of the constraints. The authors in [58] have mostly focused on the Constraint Dominance. According to this, the most popular method to evaluate Constraint Dominance is through the use of Penalty Functions depending upon the level of constraint violation of any solution. In [52] the authors have stated that the constraints can be treated as an additional objective in Bi-Objective Optimization problems. They have worked on a method in which the set of solutions can be divided into feasible and infeasible ones. The feasible solutions are the ones for which \( w \)-objective values in the non-dominated sorting \( NDS \) are equivalent to \((w + 1)-\) objectives (the constraint acting as additional objective) since the constraint violation value is 0 for the feasible solutions. The rest of the solutions are all infeasible. Both sets are ranked according to \( NDS \), however, the solutions infeasible set are ranked according to their constraint violation level. For the next generation, the solutions are picked from both sets depending on their feasibility and rank.

| Algorithm 1: Modified Infeasibility Driven Evolutionary Algorithm |
|---------------------------------------------------------------|
| **input**: Population \( G_i \) with \( C_s \) individuals,  |
| \( CV = \{cv_1, cv_2, cv_3, \ldots, cv_{s-1}\} \) for each individual |
| **output**: Set of Feasible and Infeasible solutions \( S_f, S_{inf} \) |
| 1: for count=2 to max number of \( G \) do |
| 2: Create child population from parent population \( (pop_{i-1}), childpop_{i-1} \) |
| 3: Evaluate child population \( childpop_{i-1} \) based on both objectives, \( \Delta_{x_i-c_j}, \Psi_{x_i-c_j} \) and third objective or constraint violation measure, |
| 4: \( (S_f, S_{inf}) = Split(pop_{i-1} + childpop_{i-1}) \) after recombination and mutation) |
| 5: Rank \( S_f \) |
| 6: Rank \( S_{inf} \) |
| 7: Keep 100% of the \( S_{inf} \) in the new population |
| 8: Create new population based on ranks of \( S_f \) and \( S_{inf} \) \( pop_i = S_{inf}(1, G_{inf}) + S_f(1, G_f) \) |
| 9: end for |

In another technique presented in [31], the authors have utilized a constrained optimization method using a Penalty Function approach. The authors base their approach on the understanding that Evolutionary Algorithms or Evolutionary MultiObjective Algorithms do not use any gradient or traditional mathematical optimality principle, which makes the constraint handling unusual for such algorithms. They utilize a Penalty approach which follows a positive penalty for the solutions if the constraint violation for them is \( g(x) > 0 \) and there shall be no penalty on solutions with \( g(x) = 0 \). The authors have defined \( g(x) \) as the \( j_{th} \) inequality constraint. In this paper, we modify the technique presented in [52], in which the authors have presented the Infeasibility
Driven Evolutionary Algorithm IDEA which applies to all the solutions which violate the constraints within a population. Our modified technique is named as Modified Infeasibility Driven Evolutionary Approach. According to this, we first identify the solutions which violate the constraints during the formation of the next generation. For this, we compute the Constraint Violation Measure as described in Equation 17. This CV is basically the distance of each solution’s C – C reliability, with the max min Rc–cj if the C – C reliability is less than the required max min Rc–cj. The CV of each solution is utilized to sort the solutions of the next generation. We tend to accommodate 100% of the Sinf in the new generation since their ranks are assumed high as they have CV > 0 and it acts as the third objective costing them to move away from the Pareto-Optimal Front F1. The phase 2 of our proposed CMOHPA takes only F1 into consideration, so the presence of all Sinf in the next generation does not affect the reliability of the results.

Algorithm 2: Non Dominated Sorting

**Input:** Random population Gi with Ci individuals each having, fitness function = {ΔSij–cj, ΨSij–cj, Δci–cj}  
**Output:** Non-Dominated Set of individuals,  

1. Fn = F1, F2, ···, Fn  
2. for each individual Pt in Cx do  
3. for each individual Pj+1 in Cx do  
4. if Pt+1 > Pt then  
5. Dp = Dp + 1  
6. Sp = Sp ∪ Pj+1  
7. else  
8. Dp = Dp + 1  
9. Sp = Sp ∪ Pt  
10. end if  
11. if Dp == 0 then  
12. Fc = Fc ∪ Pt  
13. end if  
14. end for  
15. end for  
16. for each individual Ic in Fc do  
17. for each individual Id in Sp do  
18. Dp = Dp + 1  
19. if current individual in Fc i.e., Dp(Id) == 0 then  
20. Fj = Fj ∪ {Id ∈ Cx }  
21. Fp ∪ {Id ∈ Cx }  
22. Fc = Fj  
23. Fj = ∅  
24. end if  
25. end for  
26. end for  

Eventually, all the solutions are stored in the set F such that, F = {F1, F2, F3, ···, Fn}, where F1 represents the set of solutions which are the best ones according to their fitness values. The second-best solutions appear in set F2 and so on. To apply this process, we keep track of the current individual Pt and the next individual Pj+1, the number of individuals that dominate the current individual, n and the list of individuals the current individual dominates Sp. We also rank each front Fi according to the front it belongs to where the subscript i represents the rank. We utilize a < comparison operator defined in the following subsection to compare the fitness of the current individual Pt and the next individual placement scheme Pj+1. We suggest that between two feasible solutions Pt and Pj+1, the solution Pt is termed better than Pj+1 if the following conditions hold true:

5) STEP 5: NON-DOMINATED SET OF PARETO-OPTIMAL FRONT

After sorting the solutions into Feasible and Infeasible sets, we then incorporate them into the new generation formation as described in Algorithm 1. We apply Non-Dominated Sorting on both the Sf and Sinf and compute their ranks in the new generation.
1. \( \Delta s_{i-j, norm}(P_t) > \Delta s_{i-j, norm}(P_{t+1}) \) \( \forall \{ P_t \in C_k \} \)
2. \( (\Psi_{s_{i-j}, norm}(P_t) < (\Psi_{s_{i-j}, norm}(P_{t+1})) \) \( \forall \{ P_t \in C_k \} \)

and also, fulfilling the constraints such that,

- a- \( (\Delta s_{c_{i-j}}(P_t) > \Delta c_{i-j}(P_{t+1})) \) \( \forall \{ P_t \in C_k \} \)
- b- \( (Y_{c_i}(P_t) > Y_c(P_{t+1})) \) \( \forall \{ P_t \in C_k \} \)
- c- \( (X_{s_i-c_j}(P_t) > X_{s_i-c_j}(P_{t+1})) \) \( \forall \{ P_t \in C_k \} \)

The working of the Non-Dominated Sorting is described in Algorithm 2. In this, the sets \( D_p, F_c, F_n, F_t, S_p \) are initialized in line 1. Where, \( F_c \) holds the individuals to be sorted in the current front and \( F_t \) holds the sorted individuals after the NDS is applied to the current generation individuals. \( S_p \) holds the individuals which are dominating all other individuals and \( D_p \) holds the number of individuals which dominate the current individual. These are summarized in Table 4. \( P_t \) denotes the current individual whose dominance is being checked against \( P_{t+1} \) in terms of their fitness values in lines 2-10. If \( P_t \) is dominating to \( P_{t+1} \), then the set \( S_p \) is updated accordingly and vice versa. If there is no dominating individual as \( D_p == 0 \) then it means that the current individual is the most dominating and it is added to the current front \( F_c \) in lines 11-12. Similarly, after the \( F_c \) is updated with the dominated individuals then each of its individuals is ranked according to its fitness level and is placed in either the current front \( F_c \) or the next front \( F_j \) in lines 16-20. The set \( F_n \) is updated with the individuals in \( F_c \) with respect to their ranks in line 21. \( F_j \) becomes \( \emptyset \) before checking the next individual in the dominated set \( S_p \) in line 23.

6) STEP 6a: SELECTION OF ELITIST INDIVIDUALS

In evolutionary algorithms, we tend to seek diversity in the solutions generated after applying crossover and mutation, so that the search space can be explored efficiently. However, this may often cause ‘genetic drift’ [59] due to which it might not be possible to reach a set of optimal solutions. Elitism, as described in [59], ensures that the probability of exploring unnecessary search space is reduced. A finite number of ‘elitist’ or best solutions (usually 10-50%) from the current generation is injected into the next generation. We select 10% [60] elitist individuals or placements from the current generation before the development of the next generation individuals represented as set \( Q \) which is extracted from set \( F_1 \). In this case, 10% best solutions are utilized as elitist since we tend to apply Phase 2 of the proposed CMOHPA algorithm on the Pareto-Front \( F_1 \) of the set of solutions \( F_1, F_2, \ldots F_n \) which consists of the best of all optimal solutions, and we try to fit a smaller window for elitist individuals so that all or most of them become a part of the Pareto-Front. Also, a smaller number of elitist individuals tend to reduce the convergence time of the algorithm according to our problem simulations.

7) STEP 6b: COMPUTATION OF CROWDING DISTANCE

After the multiple layers and/or fronts of non-dominant individuals have been sorted which identifies each individual (node) as best, second best and so on ··· described in the previous subsection, this step introduces an operator known as crowding distance \( CD \) for the set \( F = F_1, F_2, \ldots F_n \). This operator estimates the density of solutions or individuals surrounding a solution under consideration. This computes the Euclidean distance between the current solution and the solutions present on its either side. The higher the crowding distance, the solution or the individual is said to be present in the least crowded region. The \( CD \) computation requires sorting the population in set \( F = F_1, F_2, \ldots F_n \) according to each objective function value in ascending order of magnitude. So, for each objective function, the boundary solutions (solutions with smallest and largest function values) of a front \( F_1 \) are assigned an infinite distance value. All other intermediate solutions of the same front are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions.

In this paper, we defined, \( f'_{t+1} \), which represents \( i_{th} \) objective function value of the \( i_{th} \) solution or individual. For the \( CD \) computation, we require keeping track of \( f'_{t}^\text{max} \) and \( f'_{t}^\text{min} \) which are the maximum and minimum values of the \( i_{th} \) objective. For \( x \) number of solutions, we compute the crowding distance for the \( i_{th} \) solution as in Equation 18:

\[
CD_t = \sum_{i=0}^{x} \frac{f'_{t+1} - f'_{t-1}}{f'_{t}^\text{max} - f'_{t}^\text{min}}
\]

The Algorithm 3 shows the computation process of the crowding distance of each individual or placement \( P_t \). In this first the crowding distance \( cd_f \) for each \( j^{th} \) individual in a front \( F_n \) is initialized to 0 in line 2. For each objective, sort the individuals in the front \( F_n \) and assign \( \infty \) to the individuals existing at the boundary of \( F_n \) (one is represented as 1\( ^{st} \) individual and other as \( l^{th} \) in lines 3-6. Then for the remaining individuals i.e. 2\( ^{nd} \) to \( l-1 \), \( cd_f \) is computed from equation 18 in line 8.

8) STEP 7: TOURNAMENT SELECTION

The rank of any individual \( x \) is assigned according to the Front they exist in such as the individual \( P_t \) in a Front, \( F_i \) will

\[ \]
be $p_\epsilon = i$. We define a Crowded Comparison Operator $<$ or $>$ for $x$ number of individuals such that an individual $P_i$ is said to be better than $P_{i+1}$ i.e., $P_i > P_{i+1}$, if:

i. $p_\epsilon(P_i) > p_\epsilon(P_{i+1})$, and

ii. $CD(P_i) > CD(P_{i+1})$

Using this operator, we select at least two best individuals from the population, as parents for crossover and mutation.

9) STEP 8a: CROSSOVER
In this algorithm, we have utilized the single-point crossover technique. The crossover technique is inspired by nature i.e., a genetic crossover between genes. We select a random number $r$, such that $r \leq i$ for choosing the crossover point between the parent individuals. The crossover technique has been described in Fig. 3.

10) STEP 8b: MUTATION
The mutation process mutates 1 bit of the resulting offspring while taking care that we require the same, $K$ number of controllers initially desired. There are two parameters controlling the mutation process. One is the mutation rate and the other is the number of controllers in the resulting offspring. We keep the mutation rate $\sigma$ as low i.e., $\sigma \leq 0.1$ so that the mutation is mostly dependent upon the controlling the count of total number of controllers in the offspring individual. The mutation of a single chromosome consisting of $i$ switches and controllers is given by equation 19.

$$c_i = [(p_i - 1)] \cdot \delta$$  

(19)

where, $\delta$ is a binary variable dependent upon the $\sigma$ value. $\delta = 1$ if $\sigma \leq 0.1$ and 0, otherwise.

$c_i$ is the offspring individual in an $i$th location of the individual and $p_i$ is the new parent individual after crossover.

11) STEP 9: RECOMBINATION AND NEXT GENERATION SELECTION
In this step, we define the pattern in which the next generation is to be formed. We, select the 10% elitist individuals from the current generation as described in step 6 of Fig. 3, (the flow of NSGA II), and the remaining 90% individuals are created by repeating steps 8a and 8b after step 7 has been executed for creation of parents generation. We select the size of population as $x$ which means that after the tournament selection, we select two parent individuals and with crossover and mutation applied to the parent individuals we create 90% offsprings. The 10% elitist or best individuals are selected from the current generation based on pareto-front, rank of the individual and the crowded distance value of the individual.

In Fig. 3, a concise working flow of the NSGA II algorithm has been described.

After the first phase of the CMOHPA algorithm has been performed as described in Section IV-A, we receive a set of finite pareto-optimal solutions. Each of these solutions shows a possible location for the controllers. We utilize the method of finding a reference point to converge the pareto-optimal solutions to a single best location. This has been described in phase 2 of the algorithm. The list of variables utilized in the above mentioned steps have been summarized in 4.

B. PHASE 2: CONVERGENCE OF CMOHPA USING REFERENCE POINT APPROACH
In this section, we describe Phase 2 i.e., the convergence of the finite number of solutions received in the Pareto front after simulating a specified number of generations in the proposed CMOHPA algorithm to a single solution. This has been described in Fig. 4. The method of finding reference point utilized in this paper is:

- Independent upon the shape of the Pareto-optimal front $F_1$.
- Both objectives are assumed to be equally significant, and no set of preferences or preconditions are made prior to setting of the reference point.
- A set of solutions are targeted in order to find the median point from the reference point. This set of solutions are supposed to be closer to each other in terms of the Euclidean distance from the reference point. The solution closest to the median point is selected as the best solution.
- This method is applicable to more than two objectives as well.

In this regard, we consider the feasible region for the reference point $z$ where both the objectives coincide at their best possible values according to the Pareto front solution using the approaches presented in [61] and in [50], as shown in Fig. 5.

More formally, Phase 2 of the CMOHPA consists of the following 6 steps:

- Step 1: Input the set $F = F_1, F_2, \ldots, F_n$ from the previous phase 1 of CMOHPA.
- Step 2: Extract the best front set which is $F_1$ also termed as Pareto-Optimal set from the set $F$.
- Step 3: For finding the reference point $z$, the boundary solutions need to be identified in $F_1$. The boundary solutions are the ones which have either maximum reliability or minimum hop count. There are 2 boundary solutions in each front $F$. We perturb both the boundary solutions such that, each boundary solution is represented as $P_i = (\Delta_i, \Psi_i)$, then after perturbation, we have $(P_i + \epsilon) = (\Delta_i + \epsilon), (\Psi_i + \epsilon)$. Where, $\epsilon$ is a small variation in the original value (approximately 0.001) to obtain exact solution. From each perturbed boundary solution, $(P_i + \epsilon)$, we extrapolate them within the feasible region. The point of intersection is termed as ‘reference point’, $z$. This is the classical approach to find the reference point from a known Pareto-set of optimal solutions, as described in [50].
- Step 4: We then compute the Euclidean distance of each solution in the Pareto-Optimal set of solutions $F_1$ from the reference point, $z$. The distance $d_i$ from each $i$th
solution from $z$ is given in Equation 20.

$$d_i = \sqrt{\sum_{i=1}^{n} (z - F_i(P_i))^2}$$  \hspace{1cm} (20)$$

where, $n$ is the number of solutions in the front $F_1$.

- Step 5: We then identify a solution which has a distance $d_i$ minimum from the reference point $z$ as described in [50]. However, we have observed in our simulations that there are always a multiple number of solutions in the pareto-optimal front which have a minimum distance from $z$ since several solutions are closely packed in the pareto-curve. We term the subset of solutions with same minimum distance $d$ from $z$ as $d_s$ given in equation 21.

$$d_s = \sum_{i=1}^{q} (d_i)$$  \hspace{1cm} (21)$$

where, $q$ is the number of minimum distance values from $z$ reference point.

- Step 6: Next we take the median of the subset $d_s$ and the solution existing closest to this median is selected as the optimal solution of this algorithm.

$$\text{Median}_{d_i} = (d_{s_{\text{med}}} + 1)/2$$  \hspace{1cm} (22)$$

V. SIMULATION AND ANALYSIS
We use the real network traces of Panopticon network [32] to evaluate the performance of our proposed approach. All experiments were performed on Intel(R) Core(TM) M CPU @ 1.40GHz and 16 GB RAM with Ubuntu-16 VM on Windows 10 (64-bit).

In Panopticon data set, the number of controllers is varied from 2 to 8, with 417 switches, 1092 hosts, and 1064 edges connecting these nodes. We run the simulation up to 15 times and the results presented in this section are the average of the multiple results for each parameter.

Phase 1 of CMOHPA, described in Section IV-A is iterated to up to 10 generations. So the value of $Count$ in Step 3 of Phase 1 is set to 10. We have also compared the results with
varying the Count value between $1 - 10$. The population size of each generation $G$ consists of 100 individuals i.e., 100 possible alternate placements for controllers which are represented by $C_i$ as described in Section IV-A. The tournament selection in Step 7 of Phase 1 consists of a set of solutions $T$ which is a multiple of 4. In the simulations, we select at least 2 parents from the elitist set of individuals from the current generation.

We compare the performance of our proposed approach, CMOHPA, with the state-of-the-art approach [34] named Greedy Placement Approach, GPA. An obvious limitation of the GPA is focusing on a single prime objective while keeping the other objectives as secondary. We have made reliability probability as the prime objective and hop count as the secondary objective with the same constraints in CMOHPA as described above. By increasing the number of generations in NSGA-II used in CMOHPA, the quality of best solutions in the NSGA-II increases which, subsequently, increases the correctness of the results of CMOHPA. However, increasing the number of generations in NSGA-II, also increases the computation time. We are only interested in the performance of our proposed algorithm, CMOHPA, in terms of our constrained bi-objective problem. If both objectives are close to optimum values then we do not need to increase the number of generations. In this regard, Fig. 8 shows that increasing the generation beyond 10 does not improve (i.e. reduce) the hop count of the path from the switch-to-controller communication by varying the number of controllers. Therefore, we took the generation value as 10 for the rest of the experiment.

In Fig. 6, a comparison of the computation (i.e. running) time of both CMOHPA and GPA is presented. The figure shows that increasing the number of controllers increases the computation time of both GPA and CMOHPA algorithms. However, CMOHPA is significantly better performing than GPA in terms of running time for all numbers of controllers. The reason for this is that GPA has a time-complexity of $O(VEK)$ where, $V$ is the number of nodes in the network, $E$ is the number of edges in the network and $K$ are the number of controllers to be placed. On the other hand, the time-complexity of CMOHPA is $O(G.3.C(F + log(C)))$ where $G$ is the number of generations/iterations, $C$ is the number of individuals in a population, $F$ is the number of individuals in the set of not dominated by any other individuals, and 3 are the number of objectives including the constraint. This shows that the runtime of CMOHPA depending upon the number of generations and objectives should be lower than that of GPA which linearly increases with the size of the network.

In Fig. 7, we have shown the results of reliability for varying numbers of controllers and also varying the number of generations from $G = 1, G = 5$ and $G = 10$ respectively. It can be observed that a lower number of generations produce better results in terms of $S - C$ overall reliability than GPA, except when the number of total controllers are $K = 8$. However, we can deduce that increasing the number of controllers also enhances chances of higher $S - C$ reliability probability which contradicts the argument made in terms of $C - C$ communication overhead, cost and execution time.
of the entire network for the increasing number of controllers. This further promotes the need to find an optimal number of controllers.

In Fig. 8 it can be observed that there is a remarkably improved performance of the proposed CMOHPA in terms of normalized hop count values compared to the GPA. There is a slightly better performance of CMOHPA, if we increase the number of controllers as well. However, in contrast to this, the performance of GPA significantly degrades after increasing the number of controllers.

Fig. 9 provides further insight into the above argument. In this figure, it can be observed that the quality of Pareto-Fronts $F_1$ in terms of their fitness values improves as the number of maximum generations increases. However, the overall $S-C$ reliability remains above 0.5, but the overall $S-C$ hop count improves only for higher generation iterations. The pattern remains somewhat similar for a higher number of controllers as well. For simplicity, we have only depicted the results for $K = 5$ controllers.

In Fig. 9 the individual values of both the objective function values i.e., maximum of minimum $S-C$ reliability and minimum $S-C$ normalized hop count values are given for the different number of controllers. It can be observed that the increasing number of controllers improve the best solutions in terms of both parameters i.e., $S-C$ reliability and $S-C$ normalized hop count. In this Fig. 9, we have also shown the performance of the CMOHPA if we increase the number of generations to up to $count = 100$.

### A. HOP COUNT RELEVANCY TO END-TO-END DELAY

The previous results show that increasing the number of generations significantly high does not improve the reliability to a large extent and the results in terms of overall $S-C$ reliability remain either unaffected or are slightly compromised. On the other hand, the normalized hop count tends to improve to some extent if the number of generations are increased to a higher count. We can safely conclude, that our algorithm works well enough if we keep the generations iteration lower, which improves both the reliability and execution time without compromising the hop count values. In Table 5, the impact of varying number of generations on both parameters has been described in detail. The parameter hop count corresponds to the delay between the respective communicating nodes. This has been described in detail in [62] and [63] according to which the end-to-end delay is dependent on the hop distance between the communicating nodes as well as the traffic size. Assuming uniform data traffic for all nodes and using the similar technique utilized in [63], we have found out the end-to-end delay between $S-C$ using the hop count information which has been depicted in Fig. 10. It can be observed that for higher number of generations, the $S-C$ delay is reduced significantly compared to the GPA. Also, increasing the number of controllers to up to 5, the end-to-end delay for higher number of generations reduces to a minimal level compared to GPA. If we further increase the number of controllers, the overall end-to-end delay does not reduce further for the increasing number of generations in CMOHPA.
TABLE 5. Impact of varying Generations with required no. of controllers to be placed, k=5.

| Generation | min R | max R | max(min R) | max HC | min HC | maximum no. of dominated solutions |
|------------|-------|-------|------------|--------|--------|------------------------------------|
| 1          | 0.0348 | 0.8172 | 0.8172     | 0.13158463 | 0.0753243577 | 1                                  |
| 5          | 0.1139 | 0.8172 | 0.8172     | 0.152112849 | 0.0803867871 | 6                                  |
| 10         | 0.1052 | 0.8908 | 0.8172     | 0.124741895 | 0.07245235  | 21                                 |
| 15         | 0.0489 | 0.9636 | 0.8172     | 0.124741895 | 0.11286553  | 31                                 |
| 20         | 0.1139 | 0.8525 | 0.8172     | 0.13158463 | 0.07532435  | 32                                 |
| 25         | 0.0523 | 0.9636 | 0.8172     | 0.13158463 | 0.0836854748 | 11                                 |
| 30         | 0.0396 | 0.9978 | 0.8172     | 0.124741895 | 0.07245235  | 2                                  |
| 35         | 0.1052 | 0.9636 | 0.8172     | 0.138427374 | 0.0803867871 | 3                                  |
| 40         | 0.0862 | 0.9087 | 0.8096     | 0.13158463 | 0.08124444  | 1                                  |
| 50         | 0.1052 | 0.9636 | 0.5723     | 0.138427374 | 0.0809113394 | 3                                  |
| 100        | 0.5671590 | 0.8172 | 0.5723     | 0.138427374 | 0.0790569415 | 1                                  |

VI. CONCLUSION
In this paper, we have proposed a novel solution, named CMOHPA, that computes the optimum placement of multiple controllers in SDN using NSGA-II algorithm (a constrained multi-objective optimization algorithm) based on the parameters of maximum switch-to-controller path reliability, and minimum value of switch-to-controller hop count, maximum controller-to-controller reliability, load-balancing among the controllers and minimum number of controllers. We further enhanced the working of the algorithm by converging the set of the best solutions to a single best solution. The simulation results showed that our proposed solution CMOHPA performs better than the state-of-the-art related approaches. In future work, we would like to enhance the proposed approach by considering the prediction of link reliability.

ACKNOWLEDGMENT
The authors would like to show their gratitude and appreciation for the support and helpful insight of Dr. Nadir Shah into their entire research work.

REFERENCES
[1] A. Greenberg, G. Hjaltmysson, D. A. Malitz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A clean slate 4D approach to network control and management,” ACM SIGCOMM Comput. Commun. Rev., vol. 35, no. 5, pp. 41–54, 2005.
[2] G. Wang, Y. Zhao, J. Huang, and W. Wang, “The controller placement problem in software defined networking: A survey,” IEEE Netw., vol. 31, no. 5, pp. 21–27, Sep./Oct. 2017.
[3] S. Ahmad and A. H. Mir, “Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers,” J. Netw. Syst. Manage., vol. 29, no. 1, pp. 1–59, Jan. 2021.
[4] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, “A survey of controller placement problem in software-defined networking,” IEEE Access, vol. 7, pp. 24290–24307, 2019.
[5] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, “Multi-controller based software-defined networking: A survey,” IEEE Access, vol. 6, pp. 15980–15996, 2018.
[6] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, “Scalable flow-based networking with DIFANE,” SIGCOMM Comput. Commun. Rev., vol. 40, no. 4, pp. 351–362, Oct. 2010.
[7] V. Ahmad and M. Khorramizadeh, “An adaptive heuristic for multi-objective controller placement in software-defined networks,” Comput. Elect. Eng., vol. 66, pp. 204–228, Feb. 2018.
[8] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, “Control plane latency with SDN network hypervisors: The cost of virtualization,” IEEE Trans. Netw. Serv. Manage., vol. 13, no. 3, pp. 366–380, Sep. 2016.
[9] B. Heller, R. Sherwood, and N. McKeown, “The controller placement problem,” ACM SIGCOMM Comput. Commun. Rev., vol. 42, no. 4, pp. 473–478, Sep. 2012.
[10] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, “On reliability-optimized controller placement for software-defined networks,” China Commun., vol. 11, no. 2, pp. 38–54, Feb. 2014.
[11] Y. Hu, T. Luo, N. C. Beaulieu, and C. Deng, “The energy-aware controller placement problem in software defined networks,” IEEE Commun. Lett., vol. 21, no. 4, pp. 741–744, Apr. 2017.
[12] T. Hu, Q. Ren, P. Yi, Z. Li, J. Lan, Y. Hu, and Q. Li, “An efficient approach to robust controller placement for link failures in software-defined networks,” Future Gener. Comput. Syst., vol. 124, pp. 187–205, Nov. 2021.
[13] A. Jalili, V. Ahmadi, M. Keshtgari, and M. Kazemi, “Controller placement in software-defined WAN using multi objective genetic algorithm,” in Proc. 2nd Int. Conf. Knowl.-Based Eng. Innov. (KBEI), Nov. 2015, pp. 656–662.
[14] A. Jalili, M. Keshtgari, and R. Akbari, “A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach,” Soft. Comput., vol. 24, no. 4, pp. 2897–2916, Feb. 2020.
[15] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, “On the controller placement for designing a distributed SDN control layer,” in Proc. IFIP Netw. Conf., Jun. 2014, pp. 1–9.
[16] M. Tanha, D. Sajjadi, and J. Pan, “Enduring node failures through resilient controller placement for software defined networks,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2016, pp. 1–7.
[17] B. P. R. Killi and S. V. Rao, “Optimal model for failure foresight capacitated controller placement in software-defined networks,” IEEE Commun. Lett., vol. 20, no. 6, pp. 1108–1111, Jun. 2016.
[18] A. Jalili and M. Keshtgari, “A new reliable controller placement model for software-defined wans,” J. AI Data Mining, vol. 8, no. 2, pp. 269–277, 2020.
[19] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, “Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs,” IEEE Trans. Netw. Service Manage., vol. 15, no. 3, pp. 991–1005, Sep. 2018.
[20] Q. Qin, K. Poularakis, G. Iosifidis, and L. Tassiulas, “SDN controller placement at the edge: Optimizing delay and overheads,” in Proc. IEEE Trans. Netw. Service Manage., vol. 15, no. 3, pp. 991–1005, Sep. 2018.
[21] M. He, A. Basta, A. Blenk, and W. Kellerer, “Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2017, pp. 1–7.
[22] B. Zhang, X. Wang, and M. Huang, “Multi-objective optimization controller placement problem in internet-oriented software defined network,” Comput. Commun., vol. 123, pp. 24–35, Jun. 2018.
[23] B. P. R. Killi and S. V. Rao, “Capacitated next controller placement in software defined networks,” IEEE Trans. Netw. Service Manage., vol. 14, no. 3, pp. 514–527, Sep. 2017.
[24] S. Lange, S. Gebert, T. Zimmer, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, “Heuristic approaches to the controller placement problem in large scale SDN networks,” IEEE Trans. Netw. Service Manage., vol. 12, no. 1, pp. 4–17, Mar. 2015.
[25] A. K. Singh, S. Mauyra, N. Kumar, and S. Srivastava, “Heuristic approaches for the reliable SDN controller placement problem,” Trans. Emerg. Telecommun. Technol., vol. 31, no. 2, p. e3761, Feb. 2020.
[26] G. Yao, J. Bi, Y. Li, and L. Guo, “On the capacitated controller placement problem in software defined networks,” IEEE Commun. Lett., vol. 18, no. 8, pp. 1339–1342, Aug. 2014.
K. Deb and J. Sundar, “Reference point based multi-objective optimization using evolutionary algorithms,” in Proc. 8th Annu. Conf. Genetic Evol. Comput. (GECCO), 2006, pp. 635–642.

A. Sheshadri, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” MATLAB Central, vol. 182, pp. 182–197, Apr. 2006.

T. Ray, H. K. Singh, A. Isaacs, and W. Smith, “Feasibility driven evolutionary algorithm for constrained optimization,” in Constraint-Handling in Evolutionary Optimization. Cham, Switzerland: Springer, 2009, pp. 145–165.

A. Jalili, H. Nazari, S. Namvarasl, and M. Keshgati, “A comprehensive analysis on control plane deployment in SDN: In-band versus out-of-band solutions,” in Proc. IEEE 6th Int. Conf. Knowl.-Based Eng. Innov. (KBEI), Dec. 2017, pp. 1025–1031.

T. Zhang, A. Bianco, and P. Giaccone, “The role of inter-controller traffic in SDN controllers placement,” in Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN), Nov. 2016, pp. 87–92.

M. Ibrar, L. Wang, N. Shah, O. Rottenreith, G.-M. Muntean, and A. Akbar, “Reliability-aware flow distribution algorithm in SDN-enabled fog computing for smart cities,” IEEE Trans. Veh. Technol., early access, Aug. 29, 2022, doi: 10.1109/TVT.2022.3202195.

Y. Yusoff, M. S. Ngadiman, and A. M. Zain, “Overview of NSGA-II for optimizing machining process parameters,” Proc. Eng., vol. 15, pp. 3978–3983, Feb. 2011.

S. Verma, M. Pant, and V. Sasesel, “A comprehensive review of NSGA-II for multi-objective combinatorial optimization problems,” IEEE Access, vol. 9, pp. 57757–57791, 2021.

J. H. Gobbie, A. H. Gandomi, and I. Rahimi, “A comparison of constraint handling techniques on NSGA-II,” Arch. Comput. Methods Eng., vol. 28, no. 5, pp. 3475–3490, Aug. 2021.

H. Du, Z. Wang, W. Zhan, and J. Guo, “Elitism and distance strategy for selection of evolutionary algorithms,” IEEE Access, vol. 6, pp. 44531–44541, 2018.

S. Rani, B. Suri, and R. Goyal, “On the effectiveness of using elitist genetic algorithm in mutation testing,” Symmetry, vol. 11, no. 9, pp. 1145, Sep. 2019.

A. P. Wierzbicki, “A methodological guide to multiobjective optimization,” in Optimization Techniques. Cham, Switzerland: Springer, 1980, pp. 99–123.

D. Kim and Y. Kim, “Enhancing NDN feasibility via dedicated routing and caching,” Comput. Netw., vol. 126, pp. 218–228, Oct. 2017.

J. Badshah, M. M. Alhaisoni, N. Shah, and M. Kamran, “Cache servers placement based on important switches for SDN-based ICN,” Electronics, vol. 9, no. 1, p. 39, Dec. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/9/1/39

KANWAL SAEED received the B.S. degree in electronics engineering from the University of Engineering and Technology, Taxila, Taxila, Pakistan, in 2008, and the M.S. degree in telecommunication engineering, in 2014. She is currently pursuing the Ph.D. degree in electronics engineering with the University of Engineering and Technology, Taxila. Since 2014, she has been working as a Research Associate at COMSATS University Islamabad. Her research interests include software-defined networks, optimization methods, and machine learning algorithms.

MUHAMMAD OBAID ULLAH (Member, IEEE) received the Ph.D. degree in electrical engineering from The University of Manchester, Manchester, U.K., in 2012. He is currently working as a Professor with the University of Engineering and Technology, Taxila, Taxila, Pakistan. His current research interests include applied signal processing, machine learning, and ad hoc wireless networks.