symmetry

Article

Genetic Algorithms with Variant Particle Swarm Optimization Based Mutation for Generic Controller Placement in Software-Defined Networks

Lingxia Liao [1,2,*], Victor C. M. Leung [3], Zhi Li [1] and Han-Chieh Chao [2]

1 School of Electronic Information and Automation, Guilin University of Aerospace Technology, Guilin 541001, China; cczhili@guat.edu.cn
2 Department of Electric Engineering, National Dong Hwa University, Hualien 974301, Taiwan; hcc@ndhu.edu.tw
3 Department of Electric and Computer Engineering, University of British Columbia, Vancouver, BC V6T1Z4, Canada; vleung@ece.ubc.ca
* Correspondence: liaolx@guat.edu.cn; Tel.: +86-18176368266

Abstract: To enable learning-based network management and optimization, the 5th Generation Mobile Communication Technology and Internet of Things systems usually involve software-defined networking (SDN) architecture and multiple SDN controllers to efficiently collect the big volume of runtime statistics, define network-wide policies, and enforce the policies over the whole network. To better plan the placement of controllers over SDN systems, this article proposes a generic controller placement problem (GCP) that considers the organization and placement of controllers as well as the switch attachment to optimize the delay between controllers and switches, the delay among controllers, and the load imbalance among controllers. To solve this problem without losing generality, a novel multi-objective genetic algorithm (MOGA) with a mutation based on a variant Particle Swarm Optimization (PSO) is proposed. This PSO chooses a global best position for a particle according to a pre-computed global best position set to lead the mutation of the particle. It successfully handles multiple conflicting objectives, fits the scenario of mutation, and can apply in many other flavors of MOGAs. Evaluations over 12 real Internet service provider networks show the effectiveness of our MOGA in reducing convergence time and improving the diversity and accuracy of the Pareto frontiers. The proposed approaches in formulating and solving the GCP in this article are general and can be applied in many other optimization problems with minor modifications.

Keywords: SDN; network optimization; controller placement optimization; GA; PSO; hybrid EA

check for updates

Citation: Liao, L.; Leung, V.C.M.; Li, Z.; Chao, H.-C. Genetic Algorithms with Variant Particle Swarm Optimization Based Mutation for Generic Controller Placement in Software-Defined Networks. *Symmetry* **2021**, *13*, 1133. https://doi.org/10.3390/sym13071133

Academic Editor: Antonio Palacios

Received: 29 April 2021
Accepted: 11 June 2021
Published: 24 June 2021

1. Introduction

Complex networks often exhibit the features of dynamics, connections, coupling, and symmetry [1]. The 5th Generation of Mobile Communication Technology (5G) and Internet of Things (IoT) systems are complex networks that adopt such features and make network resource management and optimization difficult [2]. Particularly, they generate a wide variety of network constraints and a huge amount of runtime statistics, which significantly increase the complexity of a network and are extremely hard to cope with using current computer and communication network systems [3,4]. Software-defined networking (SDN) [5] decouples network control from dedicated network devices to form the control plane that uses standardized interfaces to collect the runtime statistics, assemble a global network view from the statistics, and based on this view enable various learning-based network management and optimization applications. As shown in Figure 1, SDN provides a visible, adaptable, and programmable layered network architecture that is capable of dealing with the continuously increasing network complexity. This makes the control plane the perfect place to manage the unprecedented increased devices, data, and applications in widely various networking scenarios.

The control plane of SDN can consist of one controller (the centralized control plane) or multiple cooperating controllers (the distributed control plane). A centralized control plane usually cannot efficiently provide flow forwarding for a large-scaled network because each new flow in the network has to involve the controller to set up forwarding rules at the involved switches. Furthermore, the extra workload added by collecting runtime statistics, the long propagation delay between a controller and its switches in a Wide Area SDN (WA-SDN), and the longer flow forwarding procedure triggered by some security applications that force each flow in the network to be forwarded back to the control plane for security analysis [6] can further increase the delay of the control plane in forwarding new flows of a network. Furthermore, the centralized control plane is a single point of failure that decreases the scalability and availability of a network. By having multiple cooperating controllers in the control plane and carefully locating them over the whole network, a WA-SDN system can provide efficient flow management as well as enable various learning-based network applications over the whole network.

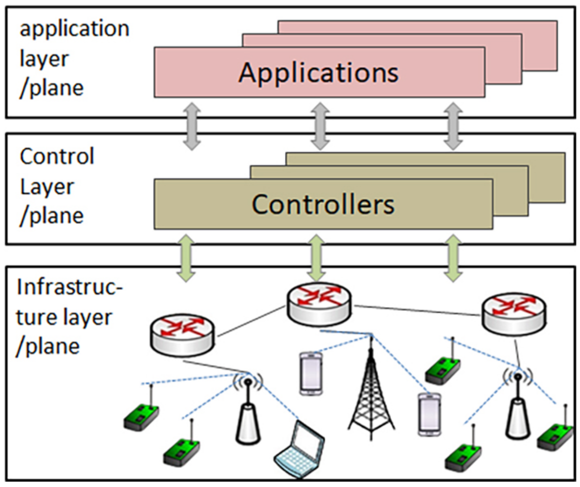

Figure 1. Software-defined networking architecture.

This article, therefore, considers a WA-SDN system with a distributed control plane containing multiple cooperating controllers, each of which directly supervises a subset of switches and collects the runtime statistics over these switches. It is a great challenge to design a WA-SDN system to reduce (a) the delay of a switch in forwarding flows to provide efficient flow forwarding, (b) the delay of a controller to synchronize runtime statistics for global network view construction in order to facilitate learning-based network management and optimization, and (c) the load imbalance among controllers to avoid controller overloading. In this article, we address this challenge by introducing a generic controller placement problem (GCP) that jointly optimizes the controller organizations and placements to simultaneously minimize the delay between controllers and switches (controller-to-switch), the delay among controllers (controller-to-controller), and the load imbalance among controllers for a WA-SDN system with learning-based applications enabled to optimize its performance, scalability, and availability.

Although existing controller placement problems (CPPs) in the literature may optimize network latency, reliability, cost, and multi-objective depending on their objective in different application scenarios [7], the classical CPPs have mainly considered the optimization of network latency that minimizes the controller-to-switch delay [8] or both controller-to-switch and controller-to-controller delays given a specific controller organization [9–12]. In contrast, the GCP is generic for controller organizations and formulates a controller placement optimization problem with three conflicting objectives, which have not been fully explored in the literature. To the best of our knowledge, our work is the first to consider a general scenario, in which controller organizations, controller placements, and switch assignments are simultaneously taken into account. The GCP generalizes the current CPPs to be a placement optimization problem with multiple conflicting objectives that simultaneously minimizes the delays between devices located in the control plane and devices located in the data plane, the delays between devices located in the same data plane, and the load imbalance among devices located in the control plane. Our approach can be readily applied to optimize the placements of many other devices such as security appliances, the Network Function Virtualization (NFV) chain, or base stations in various SDN enabled 5G/IoT systems with minimal modifications [13].

Although the optimal solution can always be found by an exhaustive search within the large solution space of the joint controller placement and switch assignment problem, such an algorithm would take too much time to be acceptable in practice. Given a WA-SDN system with m controllers and n switches, the computation complexity of an exhaustive algorithm that solves the GCP is $O(m^n n^m)$. A greedy mechanism that always attaches a switch to a controller with the shortest geographic distance can reduce the complexity of an exhaustive algorithm to $O(m^n)$, which is still too big to be acceptable when solving the GCP for a WA-SDN with a large number of controllers and switches. Current research often applies heuristics algorithms or approximate algorithms to solve CPPs [9–11]. However, most of such algorithms use similar greedy mechanisms to simplify the switch attachment and hence reduce the solution space, making these algorithms not overly suitable to solve the GCP.

Multi-Objective Genetic Algorithms (MOGAs) have generality in solving optimization problems. However, they typically have difficulty in generating a good approximation Pareto frontier that is close to the real Pareto frontier when the solution space of the problem is large [12]. A Pareto frontier is a set of Pareto optimal solutions in the objective space, each of which is at least as good as the others in at least one but not all dimensions. A multi-objective optimization problem often generates a Pareto frontier so that a decision maker can choose the suitable solution from the frontier to realize his particular preference [12]. Maintaining the diversity of population is the key for a MOGA to generate a good approximation Pareto frontier [14]. Current research has proposed many flavors of MOGAs. The Non-dominated Sorting Genetic Algorithm (NSGA-II) [15] uses an elite-preserving mechanism in its selection process to ensure the elite members stay a certain distance apart for diversity improvement. It is the best and most widely applied MOGA in current research [12].

Particle Swarm Optimization (PSO) is another general approach that can be used to solve many optimization problems [16]. It is motivated by the behaviors of birds that adjust their routes by identifying the bird in the best position during migration [17]. When applying PSO to an optimization problem, a bird is defined as a particle that represents a solution of the problem, and the movements of particles are guided by the global best position of the whole swarm of particles and the local best position of a particular particle. Because of this guidance, PSO can find the optimal solutions at a low time cost [18]. However, classical PSO is designed for optimization problems with a single objective and lacks mechanisms to maintain the diversity of solutions in a large population, while the GCP in this work has three objectives and the diversity of solutions in the population needs to be maintained to produce a Pareto frontier that is close to the real Pareto frontier. Therefore, the classical PSO cannot be directly used to solve the GCP.

To fill the gap, we propose an MOGA that uses a variant PSO to improve its mutation. A classical PSO algorithm usually keeps updating the global best position of the whole swarm and the local best positions of all the particles and uses both global and local best positions to update the velocity and position of each particle in each iteration of the algorithm. Our proposed variant PSO algorithm pre-calculates a global best position set that consists of three individual global best positions relative to the three particular objectives for the whole swarm, which remain unchanged in each iteration of the algorithm. The algorithm only uses the global best position relative to a particular objective to update a particle's velocity and position in each iteration. This approach successfully handles the multiple objectives in GCP, fits the situation of mutation functions, and can be applied in many MOGAs without affecting their existing mechanisms that optimize the population initialization and selection process. We have implemented our MOGA as NSGA-II by extending NSGA-II in Matlab [19] to have a mutation function based on the variant PSO. We compare the performance of NSGA-II with a variant PSO-based mutation with the original NSGA-II in Matlab. Our evaluation is conducted over 12 real internet service provider (ISP) networks chosen from the Rocketfuel repository [20]. Results show the effectiveness of our extended NSGA-II in reducing convergence time and improving Pareto frontier accuracy and diversity over WA-SDN systems with various topologies covering different areas.

The major contributions of this article are three-fold: (1) we propose and formulate the GCP that simultaneously considers controller organizations, controller placements, and switch assignments; (2) we extend the NSGA-II to have a variant PSO-based mutation; and (3) we provide an extensive evaluation over 12 real ISP backbone networks with various topologies covering different areas. Though the basic idea of the GCP and the variant PSO have been composed in a conference paper accepted by NFV-SDN'17 [21] (this paper is extended from a six-page conference paper, which has been listed in our reference), this article generalizes the GCP to accommodate various distributed controller organizations and provides an extensive evaluation to show our NSGA-II with variant PSO-based mutation can improve the accuracy, diversity, and convergence time of a general NSGA-II over various WA-SDN systems, and a further discussion on the parameters, potentials, and limitations of our NSGA-II is also included.

The rest of this article is organized as follows. Section 2 presents some background and related work. Section 3 proposes and formulates the GCP. The NSGA-II with variant PSO-based mutation is introduced in Section 4. An extensive evaluation that compares the quality of the Pareto frontiers and the time costs of our NSGA-II to a general NSGA-II over 12 real ISP networks together with the further discussion is provided in Section 5. Conclusions are drawn in Section 6.

2. Background and Related Work

2.1. Distributed Control Plane

The control plane of an SDN can be centralized or distributed. A centralized control plane consists of one controller [22], while a distributed one consists of multiple cooperating controllers [23–26]. Since a single controller can be a single point of failure of an SDN and only provides a limited flow setup rate to quickly set up new flows for a certain number of switches, an SDN system that has a single controller cannot provide the scalability and reliability expected by diverse applications. A distributed control plane can use the cooperation among controllers to maintain the global network view so that various learning-based network management and optimization can be enabled while improving the scalability and availability of a network by allowing each controller to only manage a subset of switches in the network.

Distributed control planes may have layered or flat organized controllers. The organization of controllers determines the cooperation among controllers. Layered organized controllers have a root controller in the upper layer and multiple local controllers in the lower layer. Switches only connect to local controllers, and local controllers connect to the root. No cooperation among local controllers and between root controllers and switches is enabled. Flat organizations allow all controllers in the control plane to be equal peers and enable interactions among any two controllers. Many proposed CPPs consider multiple controllers, but no cooperation among controllers is supported. The proposed GCP considers all such types of cooperation among controllers.

2.2. Placement Optimization Problems

The first CPP over SDN systems was proposed by Heller et al. [8]. The CPP allows the control plane to have multiple isolated controllers for the management of WA-SDN systems. This CPP is formulated to find the optimal controller placement that minimizes the controller-to-switch delay. Since the nodes in a WA-SDN are geographically spread out, and the propagation delay plays a major role in the network delay, placing controllers in a WA-SDN does not need to consider how the real data flows affect the placement of controllers. Current research has extended the CPPs to consider the capacity of controllers [11,27] to maximize the reliability of a network [28,29] or to optimize multiple objectives [10,30]. Zhang et al. in [9] introduce a CPP that is very similar to the GCP in optimizing the controller placement for a distributed control plane with multiple cooperating controllers. However, it only minimizes the controller-to-switch delay and controller-to-controller delay. In contrast, the GCP considers a more general scenario, where controller placements and organizations and switch attachments are formulated and the controller-to-switch delay, controller-to-controller delay, and the controller load imbalance are simultaneously minimized.

A CPP for the networks of data centers often has to consider the flow request processing delay, because the nodes in the data center networks are placed closely regarding the geographic distance. In Reference [31], both data and control traffic is modeled as regenerative traffic, and regenerative service processes are used to represent both link transmissions and service capabilities of controllers. Wang et al. in [32] and Zheng et al. in [33] consider a discrete time model where the length of the time slot matches the timescale at which switch requests can be precisely recorded, and assume that the request arrivals of a switch in a time slot follow a Poisson process. Yao et al. in [34] further extend this data traffic model to consider the weight of each network forwarder in the network using the degree of each network forwarder to simulate the hot spots in the network.

Unlike a CPP, to plan a placement of controllers so that the objective function is optimized, a controller migration problem investigates the adjustment of the current controller placement for the optimization of the network performance or resource usage in a dynamic scenario [35,36]. Similar to the GCP that optimizes the placement of distributed controllers, many other placement optimization problems, such as placing virtual services over the network edge [37], energy-efficient placement problems [38], and optimizing NFV chain placements [39], widely exist in SDN enabled networks. Since the GCP is generalized to consider both controller placements and switch assignments, it is suitable to formulate many of such existing placement problems with minimal modifications.

2.3. Solving Placement Optimization Problems

An exhaustive algorithm can always find the optimal solution for a CPP [8,35] but cannot generate the global-optimal solution in an acceptable time for a large-scaled network. Heuristic algorithms can be used to find approximately optimal solutions in shorter computation time. For instance, Lange et al. in [10] develops a K-Medoids based heuristic algorithm to optimize a multi-objective combinational function, and further extends it to be a Capacated K-Medoids heuristic algorithm considering the volume of each network partition no greater than the capacity of a controller; Ros et al. in [29] also develop a K-Medoids based heuristic algorithm, but the algorithm is used to study the reliability of the control plane rather than the performance or load balancing of controllers.

Since K-Medians-based algorithms often have to run a certain number of times and select the best result as the optimal solution due to the high dependency between the computation time and accuracy of an optimal solution and the initial solution chosen, to further reduce the time in solving CPPs, Zhang et al. in [9] propose an approximation algorithm that prunes the control plane placement frontier; Lin et al. in [31] develops an approximation algorithm that applies the primal-dual up-date rules of an alternating direction method of multipliers to solve the control traffic balancing in control plane placement in polynomial time for SDNs with large scales; Lange et al. in [10] introduced a simulated annealing algorithm to solve CPPs with two objectives. However, all mentioned algorithms are designed for specific predefined sets of objectives, and a general CPP as our proposed GCP cannot be solved by them over a large-scaled network.

Although many new mechanisms such as fuzzy technology [40] and stochastic systems [41] have been proposed to solve the optimization problems in different scenarios, Genetic Algorithms (GAs) and PSOs are evolution algorithms and can solve any general single or multi-objective CPP theoretically [12]. Sanner et al. in [42] applies a GA to search for the best controller placement but only optimizes the delay between controllers and switches. Jalili et al. in [43] apply an NSGA-II to solve a CPP that minimizes the delay among controllers and the load imbalance among controllers. In Reference [30], a PSO algorithm is applied to solve a CPP that minimizes the weight of the controller-to-switch delay and controller-to-controller delay. In Reference [44], a network clustering PSO algorithm is proposed to solve an optimization problem with multiple objectives over SDN systems. Our proposed algorithm is an NSGA-II with a variant PSO-based mutation. It successfully handles the multiple conflicting objectives, fits the scenario of mutation function, and can be applied in many other multiple-objective problem solving. It significantly improves the accuracy and member diversity of the Pareto frontier in a much shorter convergence time compared to a general NSGA-II. It maintains the generality of MOGAs and can be used to solve any other optimization problems theoretically.

3. Formulating GCP

We consider a WA-SDN system with a distributed in-band control plane. As shown in Figure 2, while an out-band control plane constructs dedicated network infrastructure to connect switches to controllers, an in-band control plane deploys its controllers to the places where the switches are placed as the same data network infrastructure is used for the supervision of switches. It saves money and fits in some networking environments where extra network infrastructure dedicated for network management cannot be built. Consider I and J as the sets of controllers and switches in a WA-SDN system, respectively. Given a controller placement pl in the placement space PL, J_i is the set of switches managed by controller $i \in I$ under placement pl. Let p_{ij} and p_{ik} be the propagation delays along the shortest path between controller i and switch j, and between controllers i and k, respectively. We define a binary parameter a_{ik}, and let it be 1 if controllers i and k are cooperated or 0 otherwise. This binary parameter allows the formulation to accommodate various controller organizations, which has not been achieved by our previous work [21]. The load of controller i is c_i. We formulate the mean controller-to-switch delay, mean controller-to-controller delay, and the maximal controller load imbalance as (1), (2), and (3), respectively.

$$D_{con2swi}(pl) = \frac{1}{|I|}\sum_{i\in I}\sum_{j\in J_i}(p_{ij}) \tag{1}$$

$$D_{con2con}(pl) = \frac{2}{|I||I-1|}\sum_{i\in I}\sum_{k\in I,k\neq i}(a_{ik}p_{ik}) \tag{2}$$

$$L_{imbalance}(pl) = Max_{i\in I}(c_i) - Min_{i\in I}(c_i) \tag{3}$$

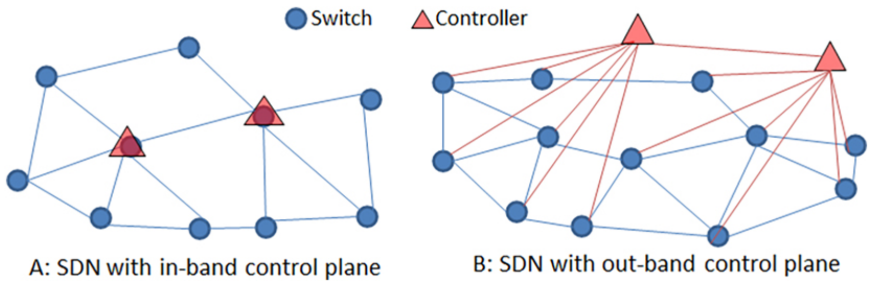
● Switch
▲ Controller
A: SDN with in-band control plane
B: SDN with out-band control plane

Figure 2. A WA-SDN system with in-band/out-band distributed control plane.

Then, for each $pl \in PL$, minimizing (1) is to minimize the mean controller-to-switch delay (OBJ1), minimizing (2) is to minimize the mean controller-to-controller delay (OBJ2), and minimizing (3) is to minimize the maximal load imbalance among controllers (OBJ3). It should be noticed that (2) fits flat and isolated organized controllers when for each i and k in I, let a_{ik} be 1 and 0, respectively. Equation (2) also fits the layered organized distributed controller when for local controllers i and k, a_{ik} is 0, and 1 otherwise. We also formulate the constraint (C1) as (4), where each switch only connects to one controller and only contributes one unit of load of this controller ($c_i = |J_i|$); and the constraint (C2) is formulated as (5), implying the same location cannot place two controllers.

$$\text{C1: } \sum_{i \in I} c_i = |I| \tag{4}$$

$$\text{C2: } p_{ik} \neq 0, i \in I, k \in I, i \neq k \tag{5}$$

Therefore, the GCP is an optimization problem that finds the best controller placements and switch attachments to minimize OBJ1, OBJ2, and OBJ3 subject to the constraints C1 and C2, as shown in (6).

$$\begin{aligned} f(pl^*) = Min(D_{con2swi}(pl), D_{con2con}(pl), L_{inbalance}(pl)) \\ s.t. \quad C1 \wedge C2 \quad \forall pl \in PL \end{aligned} \tag{6}$$

The proposed GCP has three conflicting objectives because minimizing controller-to-switch delay often spreads out the controllers over the entire network, leading to an increased geographic distance between two controllers and an increased controller-to-controller delay. Fixing the controller placement, the location of controllers is given, and the delay among controllers is specified. However, the delay between controllers and switches and the load imbalance among controllers rely on the assignment of switches to the controllers. Greedy mechanisms are likely to generate a high load imbalance among controllers when they attach switches to their geographically nearest controllers. Therefore, greedy-based heuristics algorithms cannot be used to solve the GCP.

Since SDN is a layered architecture, as shown in Figure 1, the GCP can be further generalized to a multi-objective placement problem that optimizes the delay between two devices located in two different layers, the delay between two devices located in the same layer, and the load imbalance among devices in the same layer. The generalized GCP can formulate many other placement optimization problems in a wide range of SDN enabled networking systems. For instance, consider a switching device placement problem of an industrial IoT system, where each switching device needs to be carefully located to reduce the delay between a switching device and a thing, the delay between two switching devices, and the load imbalance among switching devices. The GCP can be simply applied without any modification, as long as we move the GCP into the infrastructure layer and view the controllers and switches of our current GCP as the switching devices and things of the switching device placement problem, respectively. When we consider a distributed application placement problem of an SDN system, each application node needs to be carefully located to reduce the delay between the application and switches, the delay between two application nodes, and the load imbalance among application nodes. Since each application node runs on a controller, the distributed application placement problem is really a distributed controller placement formulated as our current GCP without any modifications.

4. Solving GCP

GAs are algorithms inspired by the process of natural selection. Given an initial population, GAs use the selection function to find the parents with higher quality from the current population set and apply crossover and mutation functions to generate offspring. Therefore, a GA typically consists of three processes: population initialization, fitness evaluation, and new population generation, as shown in Figure 3A. MOGAs are GAs that can generate the approximation Pareto optimal solutions for any optimization problems theoretically. NSGA-II is the best and the most widely used MOGA in current research.

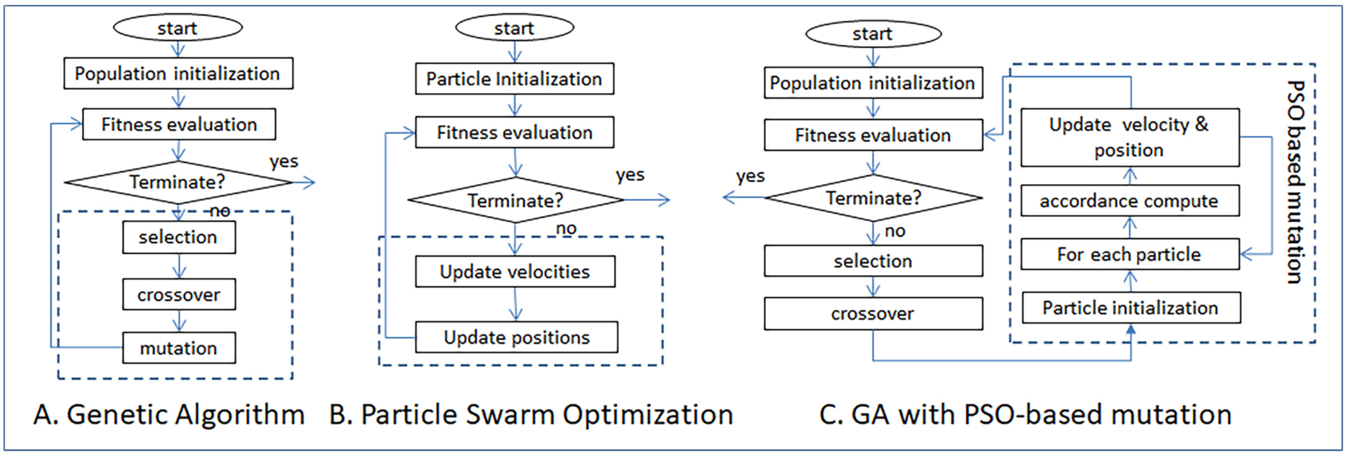

Figure 3. Flow chart of general GAs, classical PSO, and our MOGA.

4.1. Population Initialization and Fitness Evaluation

In general, the GCP encodes two chromosomes: chromosome1 represents the locations of controllers, and chromosome2 indicates the attachments of switches. Consider a network having $|I|$ controllers and $|J|$ switches, chromosome1 includes $|I|$ genes, each indicating a controller's location; chromosome2 is composed of $|J|$ genes, each representing a switch attachment. Since controllers should be located at the places where switches are located, the value of each gene of chromosome1 is an integer suggesting that place. Such value should be less than $|J| + 1$ but greater than 0. Each gene of chromosome2 indicates the index of a controller in chromosome1, and its value is also an integer that is less than $|I| + 1$ but greater than 0. In particular, as shown in Figure 4, let x_k be the value of the k gene in the chromosomes, and i be a controller in I and j be a switch in J. Then, $i = k$, if $0 < k <= |I|$, and $j = k - 5$, if $|I| < k <= |J|$. x_i presents the location of controller i, and it also equals an integer between 1 and |J|. x_j represents the assignment of switch j, and it is also equal to an integer between 1 and |I|. We then generate the fitness functions as (7), (8), and (9). Our MOGA starts from randomly generating a population set with the value of each gene of a chromosome bounded by the given value, then evaluates the quality of each population so that the population generation process can use selection, crossover, and mutation operators to generate offspring.

$$\text{fitness function for OBJ1: } \frac{1}{|I|}\sum_{i\in I}\sum_{j\in J_i}(p_{ij}) \tag{7}$$

$$\text{fitness function for OBJ2: } \frac{2}{|I||I-1|}\sum_{i\in I}\sum_{k\in I,k\neq i}(a_{ik}p_{ik}) \tag{8}$$

$$\text{fitness function for OBJ3: } Max_{i\in I}(c_i) - Min_{i\in I}(c_i) \tag{9}$$

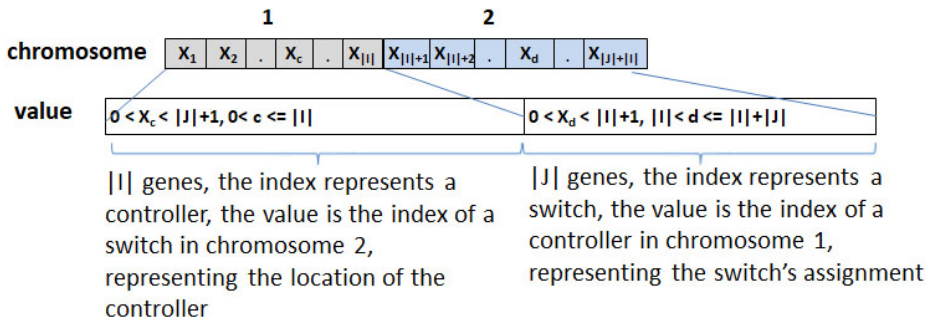
1
2
chromosome
X_1 X_2 . X_c . $X_{|I|}$ $X_{|I|+1}$ $X_{|I|+2}$. X_d . $X_{|J|+|I|}$
value
$0 < X_c < |J|+1, 0 < c <= |I|$
$0 < X_d < |I|+1, |I| < d <= |I|+|J|$
|I| genes, the index represents a controller, the value is the index of a switch in chromosome 2, representing the location of the controller
|J| genes, the index represents a switch, the value is the index of a controller in chromosome 1, representing the switch's assignment

Figure 4. Chromosome and gene value.

4.2. Selection

To generate a new generation, a selection function is applied by MOGAs to create a parent pool from the current population for the offspring generation. In addition to selecting the population with a high fitness, a MOGA also has to maintain the diversity of population to produce an approximation Pareto frontier with high quality. Our MOGA adopts the selection function of the NSGA-II provided by Matlab, where the members with high fitness values and relatively far away on the front are favored to maintain the diversity of the members on the frontier.

4.3. Crossover

After a parent pool is generated by selecting members from a population set, our MOGA starts the crossover operation to produce offspring. Our MOGA adopts the simple crossover function of the NSGA-II implemented in Matlab. The crossover function starts from randomly selecting two members in the parent pool as *parent*1 and *parent*2, then a random variable (r) ranging from 0 to 1 is generated. The weight of this random variable is specified as R and set as 0.8 in our case. Equation (10) is used to produce the genes of a child, as shown in Figure 5. The bound of genes needs to be ensured.

$$child = parent1 + r * R * (parent2 - parent1) \tag{10}$$

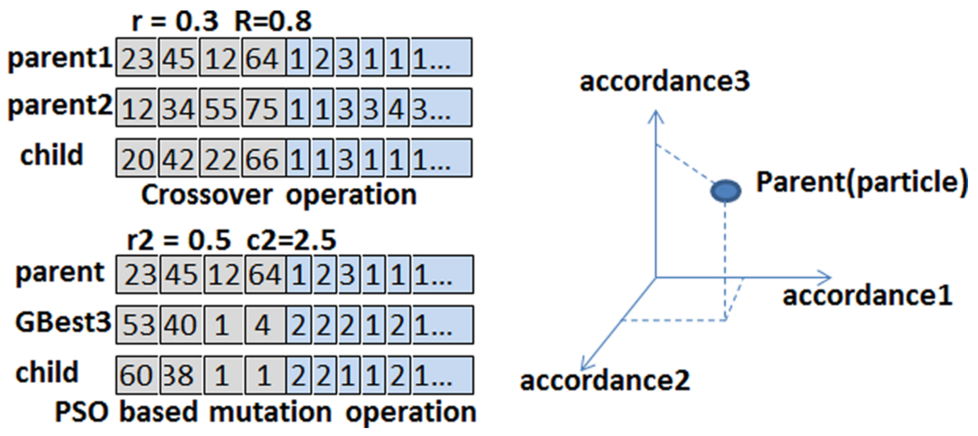
r = 0.3 R=0.8
parent1 23 45 12 64 1 2 3 1 1 1...
parent2 12 34 55 75 1 1 3 3 4 3...
child 20 42 22 66 1 1 3 1 1 1...
Crossover operation
r2 = 0.5 c2=2.5
parent 23 45 12 64 1 2 3 1 1 1...
GBest3 53 40 1 4 2 2 2 1 2 1...
child 60 38 1 1 2 2 1 1 2 1...
PSO based mutation operation
accordance3
Parent(particle)
accordance1
accordance2

Figure 5. Crossover and mutation operations.

4.4. Mutation

The Matlab NSGA-II has an adaptive feasible mutation operator to solve optimization problems with multiple objectives and bounded constraints. In such a mutation operator, a random direction and a random step length with linear constraints and bounds satisfied are generated to slightly change the values of some genes of a parent. This mutation operator does not guarantee that the produced offspring will have better fitness values due to the randomly generated direction and step length. Applying such mutation operator to solve the GCP may lead to an approximation Pareto frontier with poor diversity and long converging time because the GCP has a large search space (refer to the evaluation section for details).

PSO is another type of evolutionary algorithm motivated by the movement of birds in migration. By identifying the bird in the best position, each of the other birds in a swarm adjusts its flying velocity and direction to achieve the best migration route. PSO has a similar flow chart as GAs, but different mechanisms are adopted in each process, as shown in Figure 3B. When applying PSO to solve an optimization problem, each solution of the problem is a "bird" or referred to as a "particle". The population of solutions is a swarm of particles. Each particle is initialized by a random position and a random velocity for the movement in the solution space. PSO also needs to define a fitness function to evaluate the quality of particles and record the best position of the whole swarm of particles, so far as the global best position, and the best position of a particular particle so far as the local best position of the particle. Once a particle reaches a new position, the best position of this particle (the local best position) and the best position of the whole swarm of particles (the global best position) are updated. The new velocity of a particle depends on the global best position and the local best position and can be calculated using (11), where the last and current velocity of a particle are denoted by V^l and V^c, respectively; the last position of a particle is denoted by p^l; and the local and global best positions of a particle are denoted by $Lbest$ and $Gbest$, respectively. Let w, c_1, c_2 be parameters, and r_1 and r_2 be random variables with the range of (0,1).

$$V^c = wV^l + c_1r_1(Lbest - P^l) + c_2r_2(Gbest - P^l) \tag{11}$$

Since PSO does not need to sort the fitness of particles in any process, and the movement of each particle in a PSO is guided by the local best position and the global best position, PSO algorithms often have a short convergence time. However, as classical PSO algorithms have no mechanisms to maintain the diversity of particles, we propose a mutation based on variant PSO for a NSGA-II by taking advantage of the short convergence time of PSO and the diversity maintenance of a general NSGA-II.

For each objective ii, our variant PSO firstly pre-computes a global best position ($Gbest_{ii}$) using any existing exhaustive algorithms, general GAs, or PSO algorithms. Let $FGbest_{ii}$ be the fitness value of $Gbest_{ii}$. Then, in the particle initialization process, each child in the child set output by the crossover function becomes a particle, and a $Gbest$ set and a $FGbest$ set with the number of objectives members in each set are generated. We pick the fitness value of each particle relative to a single objective (F_{ii}) from the fitness set output from the crossover function and compute the accordance of a particle against that single objective ($Accordance_{ii}$) using (12). For each particle, the global best position of an objective with the largest accordance ($Accordance_b$) is chosen, and the movement of all particles in the swarm is conducted by that global best position, as shown in Figure 5.

$$Accordance_{ii} = FGbest_{ii}/F_{ii} \tag{12}$$

Since we consider the scenario in which each particle is allowed to mutate only once in the mutation function, the last velocity and the local best position have nothing to do with the new velocity. The first two parts of (11) are therefore dropped, and a new formula is provided to compute the new velocity, as shown in (13), where the $Gbest_b$ is the global best position of the objective with which the current particle has the highest accordance. Let P^l be the current placement of a particle, where Equation (14) is used to calculate the new position of a particle. This way, our variant PSO creates a guided mutation for an MOGA. It fits the special situation of a mutation function, reduces the computation time in executing a mutation function, and is able to generate an approximation Pareto frontier with higher accuracy and lower computation time cost. Algorithm 1 illustrates the detail of the mutation. The flow chart of our MOGA with a mutation based on the proposed variant PSO is shown in Figure 3C.

$$V^c = c_2r_2(Gbest_b - P^l) \tag{13}$$

$$P^c = P^l + V^c \tag{14}$$

Algorithm 1 Pesudo-code for PSO-based mutation function.

```
1: INPUT: Current child set and its fitness set output by crossover function; a global best
   position set and its corresponding global best fitness set.
2: Initial new child set
3: for each child in current child set output by crosser function do
4:     Let a particle be the current child
5:     Initial $P^l$ as the placement of this particle
6:     Initial $Accordance_b$ and $Gbest_b$ as 0
7:     Initial $V^c$ and $P^c$ as 0
8:     for each objective ii do
9:         Get $Gbest_{ii}$ from global best position set
10:        Get $FGbest_{ii}$ from global best fitness set
11:        Calculate $Accordance_{ii}$ using (12)
12:        update $Accordance_b$
13:        update $Gbest_b$
14:    end for
15:    Compute $V^c$ of current particle using (13)
16:    Compute $P^c$ of current particle using (14)
17:    Add $P^c$ to the new child set
18: end for
19: OUTPUT: The new child set
```

5. Evaluations

To investigate the accuracy and convergence time of our NSGA-II with a variant PSO-based mutation operator, we consider a GCP with four flat distributed controllers to optimize one objective, two objectives, and all the three objectives discussed in Section 3. Without loss of generality, for each optimization scenario, we run our NSGA-II and a general NSGA-II implemented in Matlab over 12 ISP backbone networks, selected from the Rocketfuel repository, with the number of nodes ranging from 17 to 92 covering various geographical areas, as listed in Table 1. We select real ISP networks from the Rocketfuel repository rather than the Internet Topology Zoo [45] because (1) the networks in the former repository have larger network scales; (2) the Rocketfuel repository has been used to evaluate many other CPPs in SDN systems; and (3) the Rocketfuel repository records delays between any two adjacent nodes calculated by trace data, and thus we can use it to generate the shortest delay between any two nodes of a network. The convergence time is the overall running time of an algorithm computed by the "tip" and "top" functions of Matlab. To investigate the factors affecting the performance of a GA, we choose the ASN 3561 network, the largest ISP network in the Rocketfuel repository, and vary the population size and global best position set for PSO-based mutation when running our NSGA-II. All the experiments conducted in this section run on a laptop with an Intel I5 CPU.

Table 1. ISP backbone networks from the Rocketfuel repository.

ASN	Nodes	Carrier	Coverage
15,290	17	Allstream	Canada
9942	23	Soul conv. Australia	Australia
8220	25	Tal-de Germany	Europe
577	29	Bell Canada	Canada
3300	41	British Telecom	Europe, Asia, US
6543	41	Ecospar British	North America, Europe
1221	44	Telstra Australia	Australia, US
1239	52	Sprintlink US	US
3356	62	Level3 US	US
3320	70	Deutsche telekom AG	German
701	82	MCI communication	US, Canada
3561	92	CenturyLink US	Nor. America, Europe, Aisa

5.1. Optimizing OBJ1

5.1.1. Accuracy

This subsection evaluates the accuracy of the proposed NSGA-II when solving the GCP with only OBJ1 minimized. The proposed PSO-based mutation is implemented as a customized mutation function of the Matlab GA solver. Since the GCP only minimizes OBJ1, switches can be simply attached to their nearest controllers, and chromosome2 can be dropped to reduce the search space. In the variant PSO-based mutation function, each parent in the parent pool is a particle, and the movement of particles is guided by the global best solution, which is the parent with the highest fitness in the current parent pool. Since the accuracy and convergence time of a GA highly rely on the initial population, while GAs randomly generate their initial population, we set the population size to 200, and run our NSGA-II and the Matlab GA solver 10 times. We choose the smallest solution for each selected network. We compare the results to the real global optima that can be generated by any existing exhaustive approach, as shown in Figure 6A. It is apparent that for each selected network, our NSGA-II always generates a controller placement with shorter controller-to-switch delay than the Matlab GA solver. Our NSGA-II reaches the real global optima at ASNs 6543, 1239, and 3561, while the Matlab GA does not reach the global optima at any selected ASN.

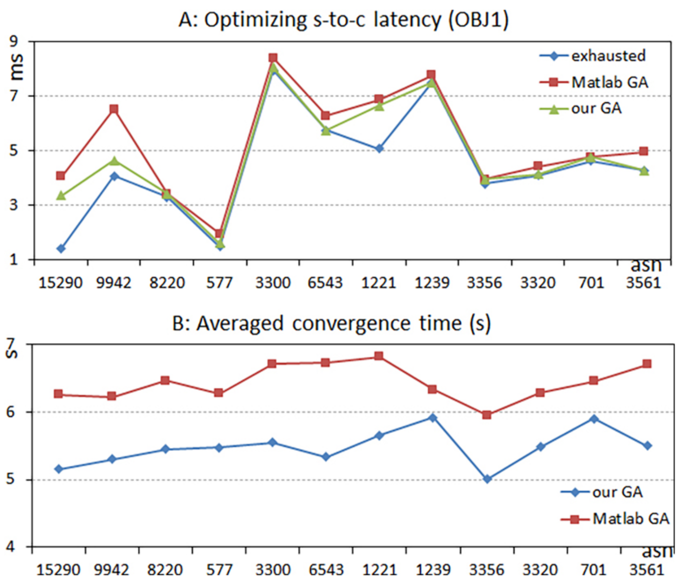

Figure 6. Minimized controller-to-switch delay (ms) and convergence time (s) when minimizing OBJ1 when population size (the number of placements) is set to 200.

5.1.2. Convergence Time

The mean convergence time of our NSGA-II and the Matlab GA when only optimizing OBJ1 for each selected network is collected and shown in Figure 6B. The mean convergence time of the proposed NSGA-II and the Matlab GA solver over all selected networks ranges from 5 to 6 s and from 6 to 7 s, respectively. The convergence time of these two algorithms has not shown a big difference over the 12 networks, which implies that the network scale does not significantly affect the convergence time of the two algorithms. However, an exhaustive approach can cost several hours or even more to generate the global optima, and the more switches in a network, the longer convergence time the exhaustive approach needs.

5.1.3. Population Size

Since the solution space of the GCP that finds the best controller placement for four controllers to optimize OBJ1 is $n * (n-1) * (n-2) * (n-3)$ (n is the number of switches in a network), a population set consisting of 200 members is only a very small part of the whole solution space. For instance, the population size of 200 is only 0.35% of the whole solution set of ASN 15,290 that has 17 switches and is only 0.000298% of the whole solution set of ASN 3561 that has 92 switches. Therefore, no real global optimized solution is guaranteed for GAs. Increasing the population size can allow the initial population set to cover a larger area of the solution set and, hence, has the chance to increase the algorithm's accuracy. Therefore, we vary the population size for our NSGA-II and the Matlab GA and find the optima of the GCP only optimizing OBJ1. We let each GA run 10 times over the ASN 3561, the largest ISP network in Rocketfuel, and choose the solutions with the smallest delays, as shown in Figure 7A. The average convergence time for each population size is collected and shown in Figure 7B. It is apparent that the real global optima have been found by our NSGA-II when the population set consists of 200 or 600 members, whereas no real global optima have been found by the Matlab GA solver with any given population size. Our NSGA-II demonstrates a higher accuracy than the Matlab GA. However, a larger population size does lead to a longer convergence time without a better quality result guaranteed for both GAs. As shown in Figure 7A, increasing population size allowed Matlab GA to generate a better solution that is closer to the real global optima, but it did not work on our proposed NSGA-II.

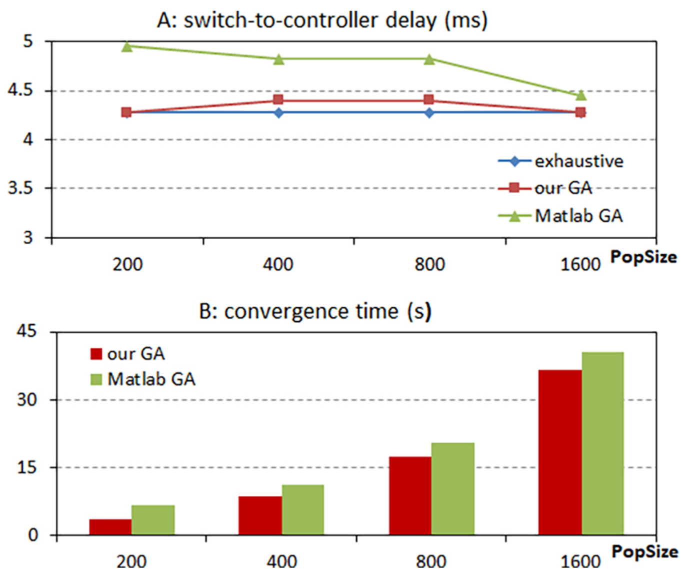

Figure 7. Controller-to-switch delay (ms) and convergence time (s) with various population sizes (the number of placements) when minimizing OBJ1 for ASN 3561.

5.2. Optimizing OBJ1 and OBJ2

5.2.1. Accuracy

Since the delay among controllers is not affected by the switch attachments when solving the GCP with OBJ1 and OBJ2 simultaneously optimized, chromosome2 can be dropped, and switches are always attached to their nearest controllers for the reduction of the search space. We generate a global best position set that consists of the global best positions for both OBJ1 and OBJ2. The global best positions for OBJ1 and OBJ2 are determined by the controller placement with the shortest controller-to-switch delay and the controller placement with the shortest controller-to-controller delay, respectively, in a Pareto frontier produced by an exhaustive algorithm with each switch attaching to its nearest controller. Since the Pareto frontier generated by a GA is highly dependent on the initial population, we run our NSGA-II and a Matlab NSGA-II solver 10 times and choose a Pareto frontier with the most members non-dominated by other Pareto frontiers for each network listed in Table 1. Since the results over all networks have a similar trend, we present the results of ASNs 8220, 1239, 3320, and 3561, as shown in Figure 8A. The Pareto frontiers generated by our NSGA-II are exactly the same as the ones generated by the

Matlab NSGA-II over ASNs 1239 and 3320 but slightly better than the ones generated by the Matlab NSGA-II over ASNs 8220 and 3561. It should be noticed that real Pareto frontiers should consist of many non-dominated solutions. Since the two NSGA-IIs used can only generate approximation Pareto frontiers due to the limited solution space searched, some approximation Pareto frontiers in Figure 8A only consist of one solution.

5.2.2. Convergence Time

The average convergence time of our NSGA-II and the Matlab NSGA-II for each network listed in Table 1 when optimizing OBJ1 and OBJ2 is collected and shown in Figure 8B. It is noticed that the proposed NSGA-II takes a lower time cost to converge an approximation Pareto frontier than the Matlab NSGA-II, but the time cost for both algorithms over a network with more switches is not always larger than the one over a network with fewer switches.

5.2.3. Population Size

We also choose the ASN 3561 and vary the population size for our NSGA-II and Matlab NSGA-II to investigate how the population size affects the algorithm's accuracy and convergence time. Our NSGA-II has not significantly shown any accuracy improvement compared to the Matlab NSGA-II at any tested population size, but an increased population size does increase the convergence time of two MOGAs, though our NSGA-II's convergence time is always shorter than the Matlab NSGA-II's at each tested population size, as shown in Figure 8C.

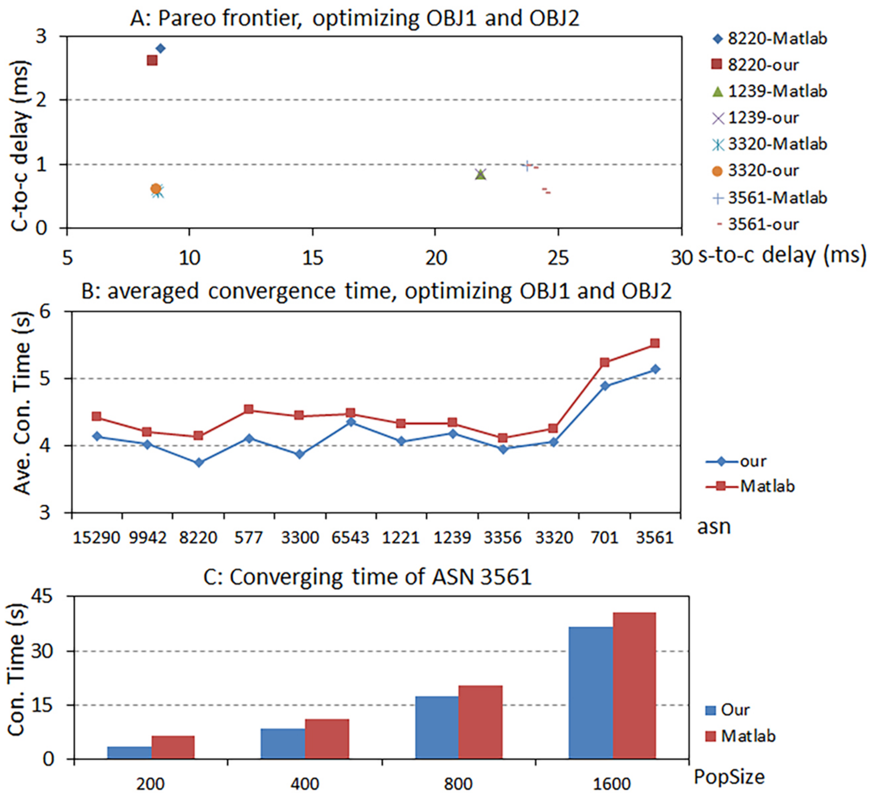

Figure 8. Pareto frontier and convergence time (s) when minimizing OBJ1 (ms) and OBJ2 (ms).

5.3. Optimizing OBJ1 and OBJ3

5.3.1. Accuracy

This subsection fixes the controller placement as the optimal controller placement computed by an exhaustive algorithm and only minimizes OBJ1 and OBJ3 by optimizing the switch attachments. The chromosomes can be simplified to only include chromosome2 consisting of n genes (n is the number of switches in a network). The global best position set consists of two assignments, one is for OBJ1 and produced by assigning each switch to its nearest controller, and the other is for OBJ3 that minimizes the load imbalance among controllers. Without loss of generality, we calculate the approximation Pareto frontier produced by our NSGA-II and the Matlab NSGA-II over each network listed in Table 1. Since the Pareto frontier of each tested network shows a similar trend, we choose the Pareto frontiers produced by our NSGA-II and the Matlab NSGA-II for ASNs 8820, 1239, 3320, and 3561 as the typical representatives, as shown in Figure 9. It is noticed that our NSGA-II outputs a Pareto frontier with larger diversity than the frontier generated by the Matlab NSGA-II or has more members located in a solution area with shorter controller-to-switch delay.

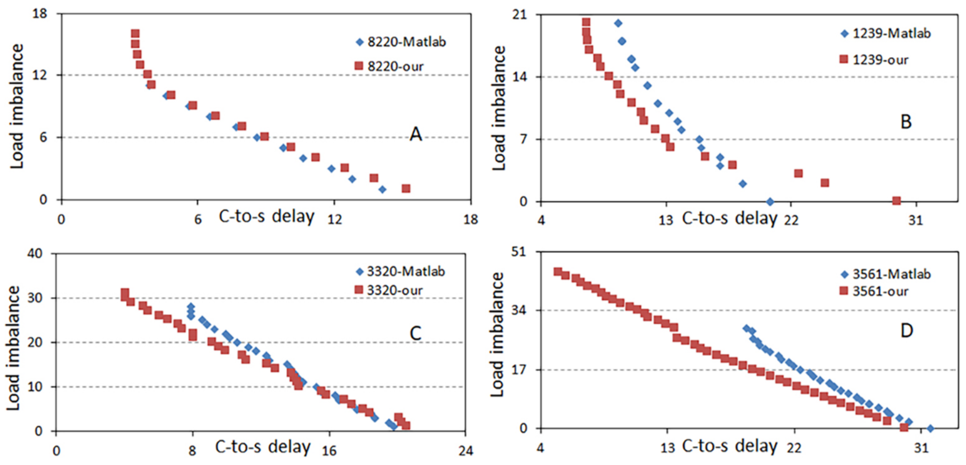

Figure 9. Pareto frontier when simultaneously minimizing OBJ1 and OBJ3 in ASNs (**A**) 8220, (**B**) 1239, (**C**) 3320 and (**D**) 3561 with 4 controllers. Population size is configured to 200, and c-to-s and c-to-c delay is the controller-to-switch and controller-to-controller delay (ms) in the GCP, respectively.

5.3.2. Convergence Time

The average convergence time of these two algorithms when optimizing OBJ1 and OBJ3 is collected and shown in Figure 10, where our NSGA-II presents a shorter convergence time than the Matlab NSGA-II over each selected network, though the number of switches in a network (the number of genes in the chromosome) does not really affect the convergence time of an MOGA.

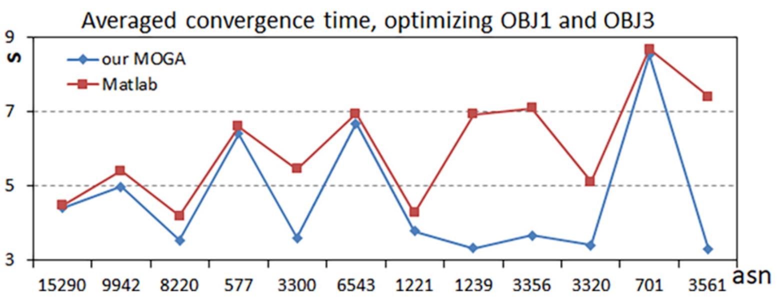

Figure 10. Convergence time (s) when minimizing OBJ1 (ms) and OBJ3.

5.3.3. Population Size

We also choose the ASN 3561 network and calculate the Pareto frontier. We configure the population size to 200 and 800. As shown in Figure 11A, the frontier produced by two NSGA-IIs with a population size set to 800 is lower than the one with a population size configured to 200, which indicates that increasing population size can improve the quality of Pareto frontiers. The mean convergence time of our NSGA-II and Matlab NSGA-II with population size configured to 200, 400, 800, and 1600 for network 3561 is collected in Figure 11B, where the convergence time of our NSGA-II is shorter than the one of the

Matlab NSGA-II, and the convergence time of both solvers is increased as the population size grows.

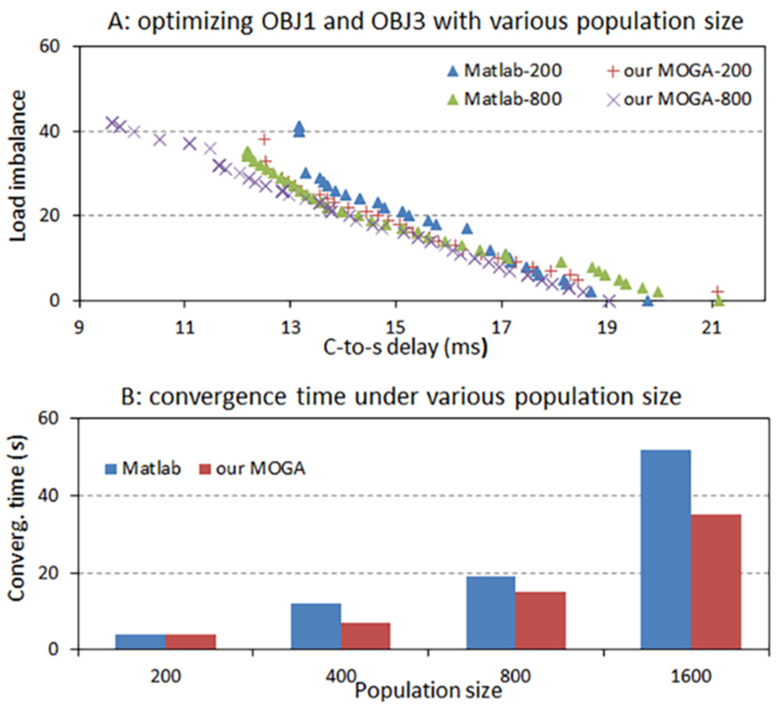

Figure 11. Pareto frontier and convergence time when optimizing OBJ1 (ms) and OBJ3 with various population size (the number of placements) for network 3561.

5.4. Optimizing All Three Objectives

5.4.1. Accuracy

Finally, we evaluate the performance of our NSGA-II when all the three objectives are minimized. In this case, both chromosome1 and chromosome2 have to be used. We calculate a global best position set (SET1) that consists of three placements. The first placement is for OBJ1, which is generated by an exhaustive algorithm that attaches switches to their nearest controllers and has the shortest mean controller-to-switch delay; the second placement is for OBJ2, which is generated by an exhaustive algorithm that attaches switches to their nearest controllers and has the shortest mean controller-to-controller delay; the third placement is for OBJ3, which combines a controller placement that attaches switches to their nearest controllers and has the shortest controller-to-switch delay, and under this controller, placement generates a switch assignment that minimizes the maximal controller load imbalance for OBJ3. Because it is not straightforward to directly represent a Pareto frontier with more than two dimensions, the Pareto frontier is projected into two dimensions with x and y axes as two of the three objectives. The projected Pareto frontier for each ASN is listed in Table 1. Since all the Pareto frontiers follow the same trend, we present the Pareto frontiers over ASNs 8220, 1239, 3320, and 3561, as shown in Figure 12. The population size is 200.

It is apparent that the Pareto frontier of our NSGA-II presents larger member diversity than the Matlab NSGA-II for each tested ASN, especially in the solution area with shorter controller-to-switch delay, because two of the three global best positions in SET1 generate the shortest controller-to-switch delays. On the contrary, the Pareto frontier generated by Matlab NSGA-II is limited to a small area. The poor member diversity makes the Matlab NSGA-II not really suitable for solving the GCP when optimizing all three objectives.

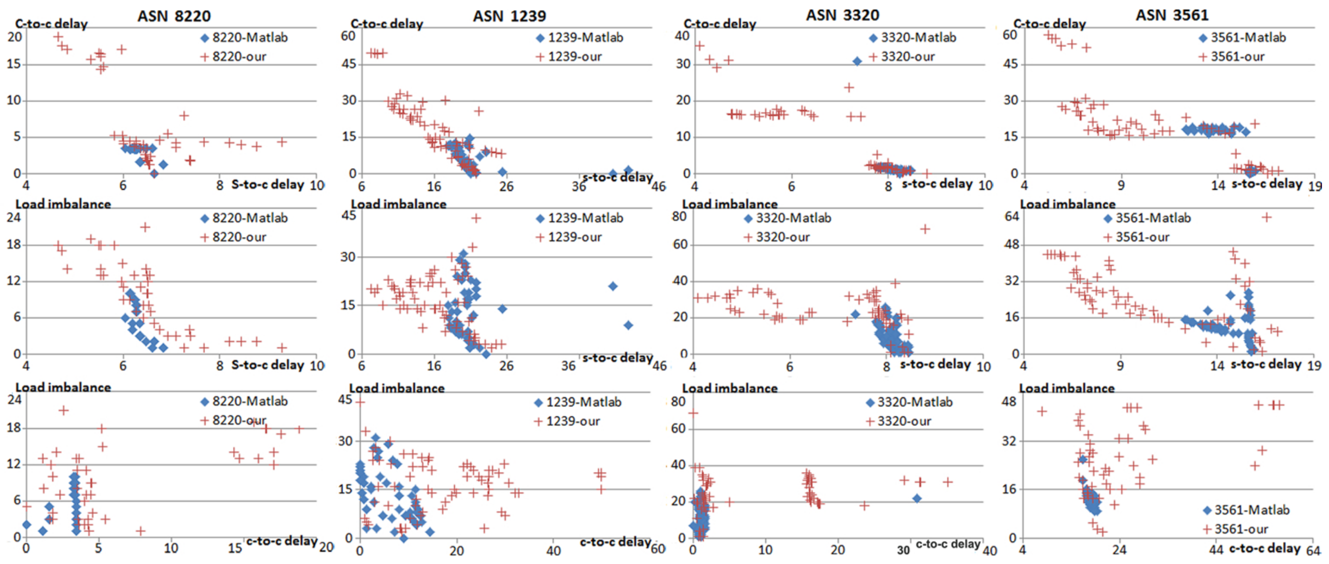

Figure 12. The projections of the Pareto frontier when simultaneously optimizing OBJ1, OBJ2, and OBJ3 for ASNs 8220, 1239, 3320, and 3561. The c-to-c and s-to-c delay is the controller-to-controller and controller-to-switch delay (ms) in the GCP, respectively.

5.4.2. Convergence Time

For each ASN, as listed in Table 1, we collect its average convergence time when optimizing all the three objectives with population size set to 200. As shown in Figure 13A, our NSGA-II can always converge a Pareto frontier in a shorter time than the Matlab NSGA-II.

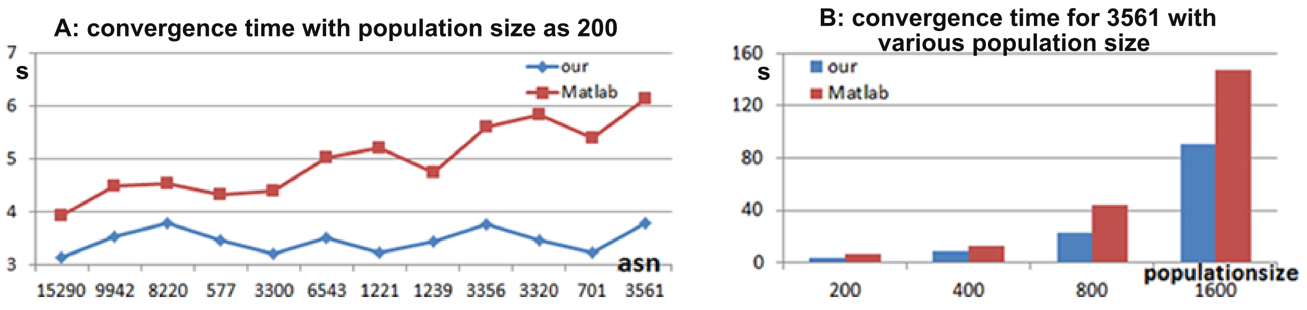

Figure 13. Convergence time (s) with population size set (the number of placements) to 200 or various when simultaneously minimizing OBJ1, OBJ2, and OBJ3.

5.4.3. Population Size

We also vary the population size of our NSGA-II and Matlab NSGA-II from 200 to 800 and present the Pareto frontiers generated by the two algorithms over ASN 3561, as shown in Figure 14. It is apparent that both MOGAs can generate Pareto frontiers with members dominating the ones when the population size used is increased. We also collect the average convergence time of two MOGAs with population size set to 200, 400, 800, and 1600, as shown in Figure 13B. It is apparent that increasing the population size increases the convergence time of both MOGAs, though our NSGA-II always has a shorter convergence time than the Matlab NSGA-II at each tested population size.

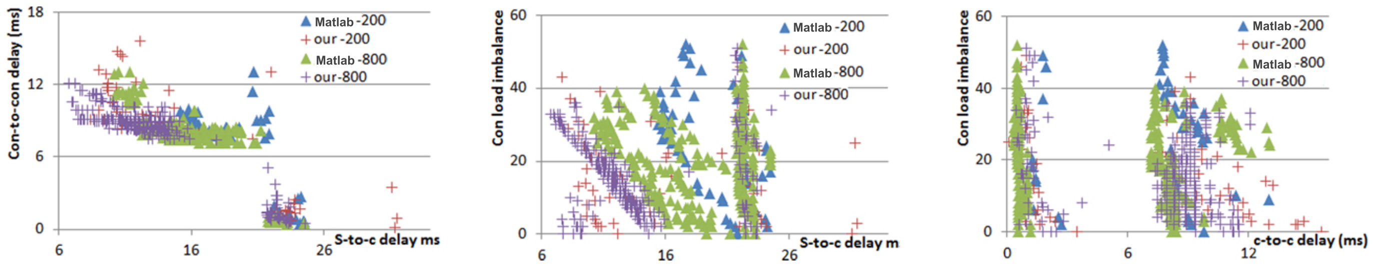

Figure 14. The projections of the Pareto frontier when simultaneously minimizing OBJ1, OBJ2, and OBJ3 with various population sizes for ASN 3561. C-to-c and s-to-c delay is the controller-to-controller and controller-to-switch delay (ms) in the GCP, respectively.

5.4.4. Varying Global Best Positions

In addition to using SET1, as used in Section 5.4.1, we generate another global best position set (SET2) that consists of the same global best positions for OBJ1 and OBJ2 as in SET1, but adjust the global best position of OBJ3 to combine a controller placement that attaches switches to their nearest controllers and has the shortest delay among controllers, with a switch assignment that maximized balances the controller load under this controller placement. We calculate the Pareto frontier of our NSGA-II and Matlab NSGA-II under SET1 and SET2 with population size set to 200, as shown in Figure 15, where our-1 presents the result using SET1, and our-2 presents the result using SET2.

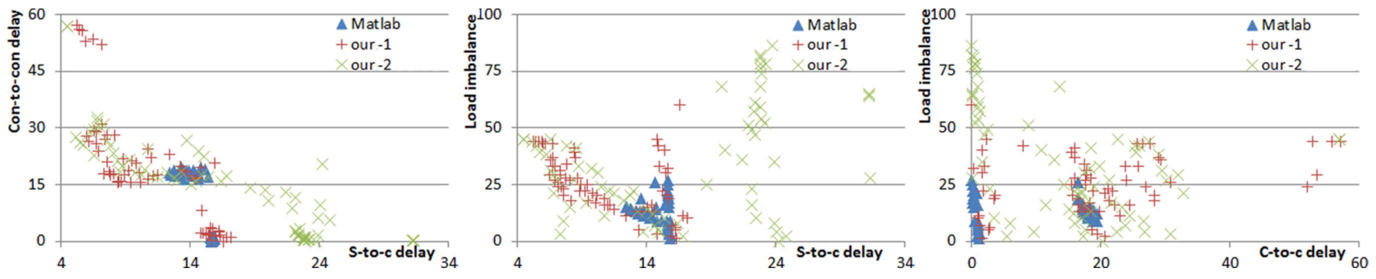

Figure 15. The projections of the Pareto frontier when simultaneously minimizing OBJ1, OBJ2, and OBJ3 with population size set to 200 and various global best positions for ASN 3561. The c-to-c and s-to-c delay is the controller-to-controller and controller-to-switch delay (ms), respectively.

It is apparent that our NSGA-II always has better member diversity in its Pareto frontier than the Matlab NSGA-II under both global best position sets. However, since the global best positions for OBJ1 and OBJ2 in SET1 and SET2 are the same and consist of a controller placement that minimizes the delay between controllers and switches, while the global best positions for OBJ3 in SET1 and SET2 consist of a controller placement minimizing controller-to-switch delay and the controller-to-controller delay, respectively, the Pareto frontier of our NSGA-II under SET1 has more member spreading in the solution area with shorter controller-to-switch delay and the Pareto frontier of our NSGA-II under SET2 has members spreading to a wider area of the whole solution space. This because the mutation under SET1 is always guided by a solution with short controller-to-switch delay. However, the mutation under SET2 can be guided by a solution with short controller-to-switch delay as well as a solution with large controller-to-switch delay because the global best position for OBJ3 in SET2 can guide a particle to a placement with short controller-to-controller delay. However, a placement with short controller-to-controller delay often has large controller-to-switch delay. The convergence time of our NSGA-II under SET2 is around 3.5 s and has not shown much difference in the convergence time of our NSGA-II under SET1.

5.5. Discussion

5.5.1. Performance Improvement

Although we only choose 12 networks from the Rockfuel repository for a better presentation, the selected 12 networks have nodes ranging from 17 to 92, representing small, medium, and large ISP backbone networks that are geographically located in reality. Therefore, we make a statistical analysis over the entire selected networks, as listed in Table 2, to further evaluate our proposed NSGA-II algorithm. We observed that our proposed NSGA-II always generated Pareto frontiers with higher quality in shorter convergence time than the NSGA-II provided by Matlab when the proposed GSP with OBJ1, both OBJ1 and OBJ3, and all three minimization objectives are solved. When the GSP minimizes OBJ1 and OBJ2, the Pareto frontiers produced by our proposed NSGA-II over 50% of the selected networks had better quality than those produced by Matlab NSGA-II, though our NSGA-II always generated Pareto frontiers with quality better than the Matlab NSGA-II over all the selected networks. Comparing our proposed NSGA-II to the exhaustive algorithms, our proposed NSGA-II always converged to Pareto frontiers in a shorter time, though our NSGA-II may not be able to find the global optima. In fact, we cannot really make a comparison over the quality of Pareto frontier between our proposed NSGA-II and the exhaustive algorithms, because applying exhaustive algorithms to solve the GCP minimization of two or three objectives is too time-consuming. However, we are able to compare the quality of Pareto frontiers generated by our NSGA-II and the exhaustive algorithms with GCP minimizing of OBJ1. We found that our proposed NSGA-II could generate the global optima over 25% of the selected networks.

Table 2. Improvement over selected networks.

Objective	Our NSGA-II and Exhaustive		Our and Matlab NSGA-IIs	
	Quality	**Conv.time**	**Quality**	**Conv.time**
OBJ1	25%	100%	100%	100%
OBJ1 and OBJ2	-	100%	50%	100%
OBJ1 and OBJ3	-	100%	100%	100%
OBJ1 and OBJ2&OBJ3	-	100%	100%	100%

It is apparent that our proposed NSGA-II can converge Pareto frontiers with higher quality in a shorter time period than the NSGA-II provided by Matlab. This is because our NSGA-II uses a variant PSO to guide the mutation of its population. This variant PSO only uses the pre-computed global best position set to calculate the new velocity and position of a particle, and the global best position set remains unchanged during the iterations. It reduces the computation time of a mutation function, fits the special scenario of a mutation that has only one iteration, and can be applied to many other flavors of MOGAs without affecting their existing mechanisms in diversity improvement.

Though both GAs and PSO algorithms cannot guarantee a global optima because the initial population of them is randomly generated and the population size is only a very small part of the solution space, our NSGA-II with variant PSO-based mutation achieves a significant accuracy improvement in generating solutions or Pareto frontiers compared to a general NSGA-II when optimizing OBJ1, both OBJ1 and OBJ3, or all OBJ1, OBJ2, and OBJ3 over all selected ISP backbone networks. However, the accuracy improvement in optimizing both OBJ1 and OBJ2 is slight. This is mainly because the chromosome of the GCP, when optimizing both OBJ1 and OBJ2, only consists of four genes (which is equal to the number of controllers in the network), and each gene has a value ranging from one to the number of switches in the network. This generates a big velocity value and incurs a big position change for a particle, leading to poor solution space exploration and resulting in poor accuracy improvement in optimizing OBJ1 and OBJ2 [18]. When optimizing both OBJ1 and OBJ3 or all OBJ1, OBJ2, and OBJ3, our NSGA-II has a larger number of genes and each gene (or most of the genes) has a smaller upper bound, leading to a small position change for a particle at each iteration and resulting in a larger accuracy improvement compared to a general NSGA-II. However, more research needs to be done in our future work.

Increasing population size does not guarantee a Pareto frontier with higher accuracy because the initial population set is randomly generated, and a population set with more members does not imply it can cover a larger area of solution space. However, as long as the population is widely spread, a larger population size will generate a Pareto frontier with higher accuracy. Increasing the population size of our NSGA-II and the Matlab NSGA-II always increases their convergence time due to the ranking and sorting process in the selection function of both MOGAs. Carefully choosing the population size can help an MOGA to balance the accuracy and convergence time. Hybridizing other algorithms such as Bayesian Optimization [46] may increase the population coverage and lead to a Pareto frontier with better quality.

The pre-computed global best position set works as the heuristics to conduct the mutation in our proposed NSGA-II. Carefully choosing this global best position set can lead to a Pareto frontier with higher accuracy, larger diversity, and shorter convergence time. In the experiment of Section 5.4, the Pareto frontiers generated by our MOGA using SET2 present higher accuracy and larger diversity than the ones using SET1, but the convergence time of our MOGA under both sets has not really shown a difference. We use a pre-computed global best position set and leave it unchanged during all iterations in our current NSGA-II implementation. In fact, the global best position set can be kept updated by choosing the population with the best fitness value for an objective from the current parent pool. However, this approach involves a process to sort the current parent pool for each objective, leading to an increased convergence time. Randomly choosing members to form the global best position set can reduce the convergence time increase, but the accuracy of the calculated Pareto frontier has to be further investigated. Our MOGA focused on optimizing the mutation process of a MOGA, and the variant PSO-based mutation can be applied to many other flavors of MOGAs without affecting their existing mechanisms of optimizing the population initialization, selection, and crossover processes of an MOGA.

5.5.2. Symmetry in the Proposed Approach

Multiple cooperated controllers are often symmetrically organized in an SDN control plane. However, controllers may present asymmetry regarding the number of switches attaching, causing controller load imbalance that reduces the network scalability and availability. Our proposed GCP aims to minimize such imbalance as well as minimizing the controller-to-switch and controller-to-controller delays to reduce such asymmetry.

When formulating the GCP, we do not consider a WA-SDN consisting of links with asymmetric bandwidth because we believe the propagation delay is much larger than the data processing and transmitting delay due to the long geographic distance between two network nodes in WA-SDNs. Such simplification has been widely used in formulating CPPs over WA-SDNs. However, such simplification does not work on data center networks, where data processing and transmitting delay play a major role.

Computer and communication networks may have symmetry in their topology. When considering such symmetry, networks can be partitioned. GCP can be firstly formulated and solved over a sub-network, then the final formulation and solutions can be assembled over the entire network using symmetry technology. Such approach reduces the computation complexity of a GCP and makes our proposed NSGA-II able to solve a GCP over larger-scaled WA-SDNs in an acceptable convergence time.

5.5.3. Solving Other Optimization Problems

Our proposed NSGA-II with variant PSO-based mutation function has shown its effectiveness in reducing convergence time and improving solution accuracy for the GCP with three conflicting objectives. Though the GCP is a static optimization problem, reducing convergence time does not seem critical for it, as network scales grow, the solution space of a CPP is increased, without increasing the population size and aided by extra mechanisms, an MOGA has a big chance to generate a Pareto frontier with lower accuracy and diversity within the same long convergence time. Our proposed NSGA-II, which generates Pareto frontiers with higher quality in shorter convergence time than a general NSGA-II, can adjust its parameters aided by some population injection mechanisms to generate a Pareto frontier with the same good quality in the same or even shorter convergence time compared to a general NSGA-II. This allows our NSGA-II to be able to solve many other static or dynamic optimization problems in a wide variety of networking scenarios.

Though MOGAs can be used to solve any optimization problem, they have a technical limitation in that the encoded chromosomes cannot be too long. In fact, an SDN enabled networking system can consist of a huge number of switches, leading to very long chromosomes, though carefully choosing the encoding method can reduce the length of chromosomes. Thanks to the existing hierarchical network administration strategy, a large-scaled SDN system can be and should be hierarchically partitioned so that each partition is managed by a distributed controller consisting of multiple control nodes. This way, each control node in the distributed controller only needs to manage a certain number of switches. A solution of a placement optimization problem over a network partition can be encoded to chromosomes with limited length and solved by our NSGA-II. More importantly, the computing capability and memory space of current computers are continuously and quickly increasing, which means that a single computer or a computer cluster has been able to use NSGA-II to solve an optimization problem with a large solution space.

In addition to controller placement problems, MOGAs have also been widely used by SDN systems to solve many other optimization problems, for instance, optimizing the signature for intrusion detection systems, the parameter of a support vector machine classification to detect distributed Denial-of-Service, the virtual network mapping, NFV chain allocation, and the usage of flow tables. MOGAs can also be used to solve the routing optimization problems of SDN systems. However, instead of encoding the network devices as genes in the GCP, applying MOGAs to solve routing optimization problems may have to encode network links or routes as genes. More complicated optimization problems may have to encode both links and nodes to chromosomes. However, as long as the solutions of an optimization problem can be encoded, our proposed NSGA-II can be applied.

Since our proposed NSGA-II that hybrids the classical NSGA-II and PSO achieves higher quality Pareto frontiers and shorter convergence time than the classical NSGA-II, we consider applying fuzzy or Bayesian technologies to NSGA-II to solve the combinatorial optimization problem with longer encoded genes and to be able to produce a higher quality Pareto frontier with shorter convergence time.

6. Conclusions

Though most of the controller placement problems in the current research use greedy mechanisms that attach switches to their geographically nearest controller to simplify their switch assignments, the controller placement problems of SDN consist of controller placements and switch assignments. This article formulates a GCP that considers such a general scenario and simultaneously minimizes the delay between controllers and switches, the delay among controllers, and the load imbalance among controllers to improve the performance, scalability, and availability of an SDN system.

Because the objectives of the GCP are conflicted, it is infeasible to apply greedy algorithms that simplify the switch assignments of the GCP. This leads to a large solution search space that cannot utilize many of the heuristics algorithms to converge a Pareto frontier in an acceptable time. To solve the GCP without loss of generality, this article proposes an MOGA that takes advantage of an MOGA in diversity maintenance and a classical PSO in global best-position guided solution searching. Our MOGA uses a variant PSO-based mutation to improve the convergence time and the quality of Pareto frontiers. The proposed variant PSO is not only capable of dealing with multiple objectives but also fits the situation of mutation and can be applied in many other MOGAs without affecting their existing optimization mechanisms in population initialization and selection. We implement the variant PSO-based mutation in a general NSGA-II and evaluate it over 12 real ISP networks selected from the Rocketfuel repository with various topologies covering different geographical areas. The results demonstrate that our NSGA-II can generate Pareto frontiers with better accuracy and diversity at a shorter time cost than a general NSGA-II.

Our NSGA-II has not shown a different convergence time when a particular population size is configured to solve the GCP over different WA-SDNs. Increasing population size cannot always generate a Pareto frontier with better quality; in contrast, it significantly increases the convergence time. The global best position set works as a heuristic to guide the mutation of a population in the proposed NSGA-II. Providing a suitable global best position set can allow our NSGA-II to generate a Pareto frontier with higher accuracy at a lower time cost.

The approaches in formulating and solving GCP are general and can be readily applied to many other static or dynamical optimization problems with minimal modifications. However, NSGA-II algorithms are often used to solve combinatorial optimizations. They cannot directly apply to optimizations with continuous variables or combinatorial optimizations with solutions that are difficult to encode to chromosomes. More research that combines newer technologies such as fuzzy and multi-objective Bayesian algorithms needs to be done in our future work.

Author Contributions: Conceptualization, L.L.; methodology, L.L.; software, L.L.; validation, L.L.; writing—original draft preparation, L.L.; writing—review and editing, V.C.M.L. and H.-C.C.; funding, L.L., V.C.M.L., Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Nature Science Foundation of China under Grant number 61962016.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: https://research.cs.washington.edu/networking/rocketfuel/ (accessed on 28 April 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. In, V.; Palacios, A. *Symmetry in Complex Network Systems*; Springer International Publishing: Berlin, Germany, 2018.
2. Wijethilaka, S.; Liyanage, M. Survey on network slicing for internet of things realization in 5g networks. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 957–994. [CrossRef]
3. Wang, B.; Su, J. FlexMonitor: A Flexible Monitoring Framework in SDN. *Symmetry* **2018**, *10*, 713. [CrossRef]
4. Hantouti, H.; Benamar, N.; Bagaa, M.; Taleb, T. Symmetry-Aware SFC Framework for 5G Networks. *IEEE Netw.* **2021**. [CrossRef]
5. Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. Available online: https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/ (accessed on 28 April 2021).
6. Chica, J.C.C.; Imbachi, J.C.; Vega, J.F.B. Security in SDN: A comprehensive survey. *J. Netw. Comput. Appl.* **2020**, *159*, 102595. [CrossRef]
7. Lu, J.; Zhang, Z.; Hu, T.; Yi, P.; Lan, J. A survey of controller placement problem in software-defined networking. *IEEE Access* **2019**, *7*, 24290–24307. [CrossRef]
8. Heller, B.; Sherwood R, R.; McKeown, N. The controller placement problem. In Proceedings of the ACM the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13–17 August 2012; pp. 7–12.
9. Zhang, T.; Giaccone, P.; Bianco, A.; De Domenico, S. The role of the inter-controller consensus in the placement of distributed SDN controllers. *Comput. Commun.* **2017**, *113*, 1–13. [CrossRef]
10. Lange, S.; Gebert, S.; Zinner, T.; Tran-Gia, P.; Hock, D. Heuristic approaches to the controller placement problem in large scale SDN networks. *IEEE Trans. Netw. Serv. Manag.* **2015**, *12*, 4–17. [CrossRef]
11. Yao, G.; Bi, J.; Li, Y.; Guo, L. On the Capacitated Controller Placement Problem in Software Defined Networks. *IEEE Commun. Lett.* **2014**, *18*, 1339–1342. [CrossRef]
12. Mukhopadhyay, A.; Maulik, U.; Bandyopadhyay, S.; Coello Coello, C.A. A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Trans. Evol. Comput.* **2014**, *18*, 4–19. [CrossRef]
13. Das, T.; Sridharan, V.; Gurusamy, M. A survey on controller placement in sdn. *IEEE Commun. Surv. Tutor.* **2019**, *22*, 472–503. [CrossRef]

14. Karafotias, G.; Hoogendoorn, M.; Eibenand, Á.E. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Trans. Evol. Comput.* **2015**, *19*, 167–187. [CrossRef]
15. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
16. Kukkarni, R.V.; Venayagamoorthy, G.K. Particle swarm optimization in wireless-sensor networks: A brief survey. *IEEE Trans. Syst. Man Cybern. Part C* **2011**, *41*, 262–267. [CrossRef]
17. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57. [CrossRef]
18. Chen, W.N.; Zhang, J.; Lin, Y.; Chen, N.; Zhan, Z.H.; Chung, H.S.H.; Shi, Y.H. Particle swarm optimization with an aging leader and challengers. *IEEE Trans. Evol. Comput.* **2012**, *17*, 241–258. [CrossRef]
19. MathWorks. Available online: https://www.mathworks.com/products/matlab.html (accessed on 28 April 2021).
20. Spring, N.; Mahajan, R.; Wetherall, D. Measuring ISP topologies with Rocketfuel. *ACM Sigcomm Comput. Commun. Rev.* **2002**, *32*, 133–145. [CrossRef]
21. Liao, L.; Leung, V.C. Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 1–6.
22. Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX: Towards an operating system for networks. *ACM Sigcomm Comput. Commun. Rev.* **2008**, *38*, 105–110. [CrossRef]
23. Tootoonchian, A.; Ganjali, Y. Hyperflow: A distributed control plane for openflow. In Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, Berkeley, CA, USA, 27 April 2010.
24. Koponen, T.; Casado, M.; Gude, N.; Stribling, J.; Poutievski, L.; Zhu, M.; Shenker, S. Onix: A distributed control platform for large-scale production networks. In Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, Vancouver, BC, Canada, 4–6 October 2010; Volume 10.
25. Berde, P.; Gerola, M.; Hart, J.; Higuchi, Y.; Kobayashi, M.; Koide, T.; Parulkar, G. ONOS: Towards an open, distributed SDN OS. In Proceedings of the ACM Third Workshop on Hot Topics in Software Defined Networking, Chicago, IL, USA, 22 August 2014; pp. 1–6.
26. Medved, J.; Varga, R.; Tkacik, A.; Gray, K. Opendaylight: Towards a model-driven sdn controller architecture. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, Sydney, Australia, 19 June 2014.
27. Qi, H.; Li, K. *Software Defined Networking Applications in Distributed Datacenters*; Springer International Publishing: Cham, Switzerland; New York, NY, USA, 2016.
28. Müller, L.F.; Oliveira, R.R.; Luizelli, M.C.; Gaspary, L.P.; Barcellos, M.P. Survivor: An enhanced controller placement strategy for improving SDN survivability. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; pp. 1909–1915.
29. Ros Francisco, J.; Ruiz Pedro, M. On reliable controller placements in software-defined networks. *Comput. Commun.* **2016**, *77*, 41–51.
30. Gao, C.; Wang, H.; Zhu, F.; Yi, S. A Particle Swarm Optimization Algorithm for Controller Placement Problem in Software Defined Network. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, Zhangjiajie, China, 18–20 November 2015; pp. 44–54.
31. Lin, S.C.; Wang, P.P.; Luo, M. Control traffic balancing in software defined networks. *Comput. Netw.* **2016**, *106*, 260–271. [CrossRef]
32. Wang, T.; Liu, F.; Guo, J.; Xu, H. Dynamic sdn controller assignment in data center networks: Stable matching with transfers. In Proceedings of the IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.
33. Zeng, D.; Teng, C.; Gu, L.; Yao, H.; Liang, Q. Flow setup time aware minimum cost switch-controller association in Software-Defined Networks. In Proceedings of the IEEE 2015 11th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), Taipei, Taiwan, 19–20 August 2015; pp. 259–264.
34. Yao, L.; Hong, P.; Zhang, W.; Li, J.; Ni, D. Controller placement and flow based dynamic management problem towards SDN. In Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW), London, UK, 8–12 June 2015; pp. 363–368.
35. Faizul Bari, M.; Roy, A.R.; Chowdhury, S.R.; Zhang, Q.; Zhani, M.F.; Ahmed, R.; Boutaba, R. Dynamic controller provisioning in software defined network. In Proceedings of the 9th IEEE International Conference on Network and Service Management, Zurich, Switzerland, 14–18 October 2013.
36. Cheng, G.; Chen, H.; Hu, H.; Lan, J. Dynamic switch migration towards a scalable SDN control plane. *Int. J. Commun. Syst.* **2016**, *29*, 1482–1499. [CrossRef]
37. He, S.; Lyu, X.; Ni, W.; Tian, H.; Liu, R.P.; Hossain, E. Virtual Service Placement for Edge Computing Under Finite Memory and Bandwidth. *IEEE Trans. Commun.* **2020**, *68*, 7702–7718. [CrossRef]
38. Demirci, S.; Sagiroglu, S.; Demirci, M. Energy-efficient virtual security function placement in NFV-enabled networks. *Sustain. Comput. Inform. Syst.* **2021**, *30*, 100494. [CrossRef]
39. Xie, Y.; Wang, S.; Dai, Y. Revenue-maximizing virtualized network function chain placement in dynamic environment. *Future Gener. Comput. Syst.* **2020**, *108*, 650–661. [CrossRef]

40. Roman, R.C.; Precup, R.E.; Petriu, E.M. Hybrid data-driven fuzzy active disturbance rejection control for tower crane systems. *Eur. J. Control* **2021**, *58*, 373–387. [CrossRef]
41. Zhu, Z.; Pan, Y.; Zhou, Q.; Lu, C. Event-triggered adaptive fuzzy control for stochastic nonlinear systems with unmeasured states and unknown backlash-like hysteresis. *IEEE Trans. Fuzzy Syst.* **2021**, *29*, 1273–1283. [CrossRef]
42. Sanner, J.l.; Ouzzif, M.; J-aoul, Y.H. Evolutionary algorithms for optimized SDN controllers & NVFs' placement in SDN networks. In *SDN Day 2016*; Hal-Inria: Paris, France, 2016.
43. Jalili, A.; Ahmadi, V.; Keshtgari, M.; Kazemi, M. Controller placement in software-defined WAN using multi objective genetic algorithm. In Proceedings of the 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran, 5–6 November 2015; pp. 656–662.
44. Liu, S.I.; Wang, H.; Yi, S.; Zhu, F. NCPSO: A solution of the controller placement problem in software defined networks. In Proceedings of the 2015 International Conference on Algorithms and Architectures for Parallel Processing, Zhangjiajie, China, 18–20 November 2015.
45. The Internet Topology Zoo. Available online: http://www.topology-zoo.org/dataset.html (accessed on 28 April 2021).
46. Galuzio, P.P.; de Vasconcelos Segundo, E.H.; dos Santos Coelho, L.; Mariani, V.C. MOBOpt—Multi-objective Bayesian optimization. *SoftwareX* **2020**, *12*, 100520. [CrossRef]