Front. Comput. Sci., 2020, 14(3): 143402
https://doi.org/10.1007/s11704-018-7277-8

RESEARCH ARTICLE

Varna-based optimization: a novel method for capacitated controller placement problem in SDN

Ashutosh Kumar SINGH (✉), Saurabh MAURYA, Shashank SRIVASTAVA

Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology Allahabad, Allahabad 211004, U.P., India

Abstract Recently, software defined networking (SDN) is a promising paradigm shift that decouples the control plane from the data plane. It can centrally monitor and control the network through softwarization, i.e., *controller*. Multiple controllers are a necessity of current SDN based WAN. Placing multiple controllers in an optimum way is known as controller placement problem (CPP). Earlier, solutions of CPP only concentrated on propagation latency but overlooked the capacity of controllers and the dynamic load on switches, which is a significant factor in real networks. In this paper, we develop a novel optimization algorithm named varna-based optimization (VBO) and use it to solve CPP. To the best of our knowledge, this is the first attempt to minimize the total average latency of SDN along with the implementation of TLBO and Jaya algorithms to solve CPP for all twelve possible scenarios. Our experimental results show that TLBO outperforms PSO, and VBO outperforms TLBO and Jaya algorithms in all scenarios for all topologies.

Keywords SDN, controller placement, CPP, latency, VBO

1 Introduction

Software defined networking is a new edge network paradigm which aims to provide separation between the control plane and data plane [1–7]. The data plane is sometimes known as the forwarding plane (or user plane) which is responsible for forwarding traffic as per the decisions made by the controller(s). The control plane is responsible for handling the network traffic, generating rules and policies for the forwarding devices (switches and routers) with the help of controller(s). Whenever a switch receives a new flow, it sends a *Packet_In* message to its respective controller for setting up flow rules along with the best flow path. The controller is responsible for managing the routing of flows by interacting with the switches through a secure channel. It guides to the switches that how packets should be forwarded by installing new flow rules and policies. In a large-sized network, the single controller is not sufficient to handle a large number of switches (or routers) that are geographically distributed as it cannot ensure acceptable latencies between switches and controllers. Controller installed on a particular server has a limited resource capacity for handling a large number of *Packet_In* generated by switches. As a result, SDN based WAN uses multiple controllers to increase the performance of the network. Placing multiple controllers is a good choice if the decision of controller placement is based on placement metrics such as average switch-to-controller latency, maximum switch-to-controller latency, inter-controller latency, and so on. The controller placement problem was introduced by Heller et al. [8] in 2012. During the placement of controllers in the WAN [9, 10] following questions arise.

1) What should be the minimum number of controllers required for the WAN network?
2) Where should the controller(s) be placed in the networks?
3) How many networking devices should be attached to a

Received August 4, 2017; accepted March 30, 2018

E-mail: ashuit89@gmail.com

given controller?

During the spans of last six years, various researchers tried to solve the CPP by taking the different determinant of problems. According to previous research works, CPP can be broadly classified into two classes. One is capacitated controller placement problem (CCPP), and the other one is uncapacitated controller placement problem (UCPP). The performance of the controller [11] in SDN is a major factor when scalability of SDN is taken into account. Heller et al. [8] discussed CPP as facility location problem which is known to be an NP-Hard class of problem. To solve the CPP, researchers proposed their solutions leveraging optimization algorithms. The other school of thought is to take CPP as a clustering problem in which a large-sized network is partitioned into several small-sized network domains and only one controller controls each domain [12]. Network partitioning could reduce the overall complexity of the large-sized network. Gaining the understanding of CPP, we came to a conclusion that following determinants should be taken into attention while designing a new placement algorithm and approach.

- **Latency in CPP** Network latency is the time interval of transfer of packets between any two nodes of the network and these nodes may be either switch or controller. Latency is measured in terms of distance, and it may be either unit weight distance between any two nodes (it is called one hop distance) or weighted distance between any two nodes. In CPP, latency either may be switch-to-switch or switch-to-controller or inter-controller or total latency of the network. Most of the previous research works tried to solve the problem of CPP by minimizing switch-to-controller latency but ignoring the inter-controller latency and total latency of the network. However, Gao et al. [13] introduced a PSO based solution to minimize the total average latency of the network.
- **Inter-controller communication** If switch "A" attached to a controller wants to send the message to a switch "B" connected to another controller, then there is a need for inter-controller communication [14]. Controllers communicate with each other using border gateway protocol (BGP) [15]. If a large number of controllers are required for handling the network, then inter-controller communication overhead is increased, but the overall performance of the network may be improved.
- **Required number of controllers** Here, the objective is to place a minimum number of controllers that could increase the performance of SDN based infrastructure. However, assigning one controller per switch minimizes the switch-to-controller latency, but it leads to increase inter-controller communication overhead and reduce the controller utilization.
- **Controller capacity limitation** Papers [11, 16, 17] discuss the performance and capacity of SDN controllers. The *c-bench* [18] is used to measure the number of flows that are processed by the controller per second. SDN controller can supervise a certain number of switches because of limited resources like memory, processor, etc. For example, Nox Controller can handle 30k flows setups per second. Therefore, load balancing between controllers is required to increase the performance of SDN.
- **Load on the switches** Switches also can handle a certain amount of incoming and outgoing request (sending and receiving packets). The overloaded switch may drop the packets. Therefore, performance of SDN may be degraded due to packet loss.
- **Global latency minimization** Global latency is the sum of switch-to-controller latency and inter-controller latency. Researcher's objective is to maximize the performance of SDN by minimizing the total latency of the network [13].

To the best of our knowledge, up till now, there is no approach to consider all the above determinants for given solutions to the controller placement problem. In this paper, we discuss all possible scenarios as shown in Fig. 1. These scenarios are considering different parameters such as weight (distance between two nodes) on edges, the load on switches and capacity of controllers. The weight of the link between two switches defines the latency (in milliseconds) between them. We consider the weight of links either unit or different. The unit weight means that all links have same (*one hop*) distance latency and different weight means that all links have different latencies. For example, if a switch is in San Francisco (USA), then its distance from a switch in New York (USA) is 4,129 km. And if the switch is in Los Angeles (USA), the distance is 559 km. In both the cases, the distance is considered as a unit but considering them as same is inappropriate. The load on the switch represents the packets sent to controller per second. The load on the switches either same or different. The capacity of the controller means the number of *Packet_In* messages processed by the controller per second. Controllers can be either uncapacitated or capacitated.

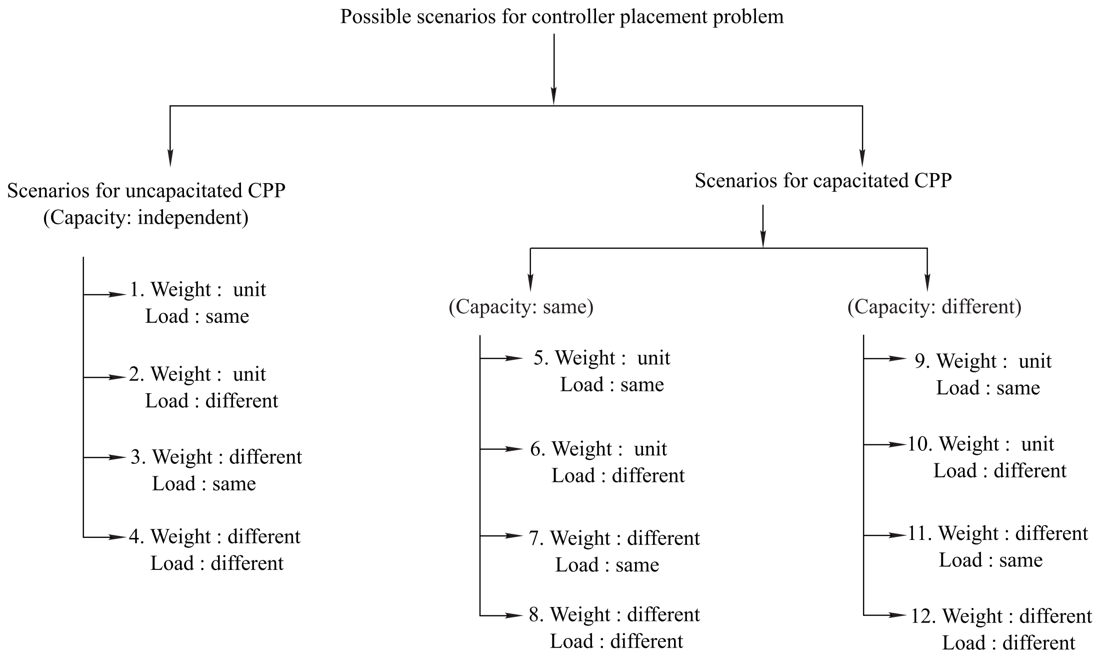

Possible scenarios for controller placement problem
Scenarios for uncapacitated CPP (Capacity: independent)
Scenarios for capacitated CPP
(Capacity: same)
(Capacity: different)
1. Weight : unit Load : same
2. Weight : unit Load : different
3. Weight : different Load : same
4. Weight : different Load : different
5. Weight : unit Load : same
6. Weight : unit Load : different
7. Weight : different Load : same
8. Weight : different Load : different
9. Weight : unit Load : same
10. Weight : unit Load : different
11. Weight : different Load : same
12. Weight : different Load : different

Fig. 1 Scenarios for controller placement problem

In this paper, we see the controller placement problem in two separate domains; 1) as a clustering problem and 2) as an optimization problem. We consider all possible scenarios of CPP including capacitated, un-capacitated, load aware capacitated and load aware un-capacitated controller placement problem. First, we compare clustering based solution of CPP with the optimization based solution of CPP for all possible scenarios. After experiments, our experimental results tell that optimization based solutions are better than clustering based solutions. Our proposed optimization algorithm provides better results as compared to previously used optimization algorithms. Second, we show the experimental results of all possible cases which are important for network administrator for performance analysis point of view. So, we give more attention to optimization based solutions over the clustering based solutions. The main contributions of this paper are summarized as:

1) We analyze the clustering and optimization based solutions for CPP. We also compare these two solutions and check which one is better for CPP.
2) We map PSO algorithm in CPP with some modifications in order to find minimum total average latency and simultaneously find an optimum number of controllers.
3) To the best of our knowledge, we apply TLBO (Teacher Learning Based Optimization) and Jaya algorithm in CPP for the first time.
4) We compare clustering and optimization based solutions with our proposed novel algorithm (VBO) where the performance of VBO is found better.

The rest of the paper is structured as follows. In this paper, we discuss the mathematical model for CPP in Section 2. In Section 3, we compare solutions of clustering based and optimization based approaches. Section 4 shows the experimental results and discussions. In Section 5, we test our proposed VBO algorithm on constrained and unconstrained benchmark functions. Section 6 surveys the CPP related work and finally, the paper is concluded in Section 7.

2 Mathematical model for controller placement problem

The network topology can be represented as a graph G(S; E), where S is a set of switches and E is set of edges between switches. First, we analyze the network topology and then partition it into different smaller sub-networks. According to the problem of controllers placement, we deploy one controller per sub-network.

Symbols and definitions are used in CPP as given in Table 1. Average switch-to-controller latency ($\pi^{avgS2Clatency}(P)$), maximum switch-to-controller latency ($\pi^{maxS2Clatency}(P)$), average inter-controller latency ($\pi^{avgC2Clatency}(P)$) and maximum inter-controller latency ($\pi^{maxC2Clatency}(P)$) are calculated by Eqs. (1–4) respectively.

$$\pi^{avgS2Clatency}(P) = \frac{1}{n}\sum_{s\epsilon S}\min_{c\epsilon P} d(s,c), \tag{1}$$

$$\pi^{maxS2Clatency}(P) = \min_{P\subseteq S}\max_{s\epsilon S}\min_{c\epsilon P} d(s,c), \tag{2}$$

$$\pi^{avgC2Clatency}(P) = \frac{1}{p_c}\sum_{c_i,c_j\epsilon P} d(c_i,c_j), \tag{3}$$

$$\pi^{maxC2Clatency}(P) = \max_{c_i,c_j\epsilon P} d(c_i,c_j). \tag{4}$$

Table 1 Symbols and Definitions used in CPP

Symbols	Definitions
$G(S,E)$	Graph G, where S is a set of switches and E is set of edges between switches
$d(s,c)$	Shortest path between switch $s\epsilon S$ and controller $c\epsilon C$
$d(c_i,c_j)$	Shortest path between controllers $c_i, c_j\epsilon C$
n	Total number of switches, i.e., $\lvert S\rvert$
k	Total number of controllers to be placed in the SDN
$P\subseteq S$	Set of all possible placements for controllers
S'	Set of placed controllers in SDN
p_c	Total number of inter-controllers paths
$S(c)$	Set of switches controlled by controller c
$L(c)$	Capacity of the controller c
$l(s)$	Load on the switch s
$\pi^{avgS2Clatency}(P)$	Average switch-to-controller latency
$\pi^{maxS2Clatency}(P)$	Maximum switch-to-controller latency
$\pi^{avgC2Clatency}(P)$	Average inter-controller latency
$\pi^{maxC2Clatency}(P)$	Maximum inter-controller latency
$T^{avg-latency}(P)$	Total average latency of the SDN
$T^{max-latency}(P)$	Total maximum latency of the SDN
X	Position vector of the particle
D	Shortest distance matrix of the network graph $G(S,E)$

With the above definitions, we can calculate the total average latency ($T^{avg-latency}(P)$) and total maximum latency ($T^{max-latency}(P)$) of the SDN by Eqs. (5) and (6) respectively.

$$T^{avg-latency}(P) = \frac{1}{n}\sum_{s\epsilon S}\min_{c\epsilon P} d(s,c) + \frac{1}{p_c}\sum_{c_i,c_j\epsilon P} d(c_i,c_j), \tag{5}$$

$$T^{max-latency}(P) = \min_{P\subseteq S}\max_{s\epsilon S}\min_{c\epsilon P} d(s,c) + \max_{c_i,c_j\epsilon P} d(c_i,c_j). \tag{6}$$

In this paper, our goal is to minimize the total average latency of the network $\left(T^{avg-latency}(P)\right)$ for all possible scenarios. In today's real world, traffic is increasing day by day, so one controller cannot handle such a large traffic. In this type of scenario, the capacity of controllers and load on the switches are required. When we are considering capacity of controllers and load on the switches, then our proposed algorithm (VBO) gives better results as compared to other optimization algorithms. Minimum total average latency based CPP is defined by Eq. (7). Equation (8) guarantees that the total loads on switches do not exceed the capacity of respective controller.

$$\min T^{avg-latency}(P). \tag{7}$$

Subject to:

$$\sum_{s\epsilon S(c)} l(s) \leqslant L(c), \quad \forall c\epsilon P. \tag{8}$$

3 Is clustering based solutions better or heuristic based optimization solutions for CPP?

In this section, we critically analyze, discuss and compare the clustering based solution and optimization based solution for controller placement problem. In the literature, [19–24] discussed the clustering based solution and [25–28] discussed the optimization based solution for controller placement problem.

3.1 Clustering based solutions

In the year 2009, FlowVisor [29] first laid the foundation of multiple controllers in SDN that is based on the partitioning of the large-scale network into several domains and each domain is managed by a single controller. Liao et al. [30] used density based clustering to partition the large-scale SDN network into smaller clusters, and only one controller controls each partitioned cluster. Density based controller placement (DBCP) provides better results as compared to k-center for the CPP [31]. Here, we introduce weight on the edges after modifying the DBCP algorithm. And we also implement weighted algorithm for the CPP for all the publicly available internet topology zoo [32] and generate the results.

3.2 Heuristic based optimization solutions

Controller placement problem is an NP-hard problem. The papers [13, 26, 33–35] discuss controller placement problem and provide heuristic based optimization solutions for it. Gao et al. [13] introduced a PSO-based algorithm to solve this problem. Here, authors found optimal placements but a number of controllers should be known in advance. They did not address the first condition of CPP [8]. We eliminate this limitation such that our algorithm decides how many numbers of controllers are needed for optimal placement.

Most of the works in literature have not considered the capacity of controllers and load on switches. [25, 26, 30, 33, 36] have considered it but taken only unit weight among the switches of the network. In this paper, we consider both unit and weighted edges among switches and all possible scenarios of load on switches and capacity of controllers. Table 2 shows key points that are considered by different authors in the literature.

We consider all 12 different possible scenarios. The weight of the link between two switches defines the latency (in milliseconds) between them. We consider the weight of links either unit (one) or different. The unit weight means that all links have same *one hop* distance latency and different weight means that all links have different latencies.

The load on the switch represents the packets sent to controller per second. We consider the load on the switches either same or different. Same load on switches means all switches are sending equal number of *Packet_In* messages per second to their respective controller. Same load on switch is the generalization of unit load on switch. Different load on switches means switches are sending the different number of *Packet_In* messages per second to their respective controller.

The capacity of the controller means the number of *Packet_In* messages processed by the controller per second. Controllers can be either uncapacitated or capacitated. Literally, uncapacitated controllers considered as having unlimited packet processing power, i.e., it is independent of the capacity. Capacitated controllers may have the same and different capacity. The same capacity of the controller means all controllers have equal packet processing power. The different capacity of the controller means all controllers differ in the packet processing power.

Based on the capacity of controller, these scenarios are classified into two different sub-classes, namely 1) uncapacitated CPP and 2) capacitated CPP. There are four different scenarios possible for uncapacitated CPP. These scenarios are: (i) weight of each edge is one and load on each switch is same, (ii) weight of each edge is one and load on each switch is different, (iii) weight of each edge is different, and load on each switch is same, and (iv) weight of each edge is different and load on each switch is different. Capaciatated CPP means the controller is having either same capacity or different capacity. Also, controllers with same (or different) capacity are categorized into four different scenarios (same as uncapacitated CPP) which are already discussed above.

In most of the research works, a known number of controllers are assumed for large-sized networks, because it is difficult to determine the minimum number of controllers. Without traversing all possible locations, we cannot find this number, which is not feasible for large-sized networks. The clustering based approach may resolve this difficulty. In this paper, our objective is to minimize the overall latency of the SDN and at the same time find the required number of controllers.

3.2.1 Mapping of VBO and other optimization methods for CPP

In our mapping of the particle for all optimization methods, a particle refers to the particular placement of controllers and switches attached to respective controllers. Here, the particle is taken as a *d*-dimensional vector, where *d* is the total number of switches in the network, i.e., *n*. And value in each dimension indicates controller number. For example, let a network consists of seven switches and three controllers, these controllers are placed at switch number 2, 4 and 7. Where, controller at switch 2 controls switches 2, 3, and controller at switch 4 controls switches 1, 4 and 6. And controller at switch 7 controls switches 5 and 7. For this case, the mapping is shown in Fig. 2.

Table 2 The Key points considered from literature for controller placement problem (*S2C* = switch-to-controller and *C2C* = controler-to-controller)

References	WAN network	Latencies			Solutions based on	
		S2C	*C2C*	Total	Clustering	Optimization
Bari et al. [19]	✓	✗	✗	✗	✓	✗
Hu et al. [28]	✓	✗	✗	✗	✗	✓
Perrot and Reynaud [27]	✓	✓	✗	✗	✗	✓
Hock et al. [25]	✗	✓	✓	✗	✗	✓
Cheng et al. [20]	✓	✗	✗	✗	✓	✗
Cheng et al. [24]	✓	✓	✗	✗	✓	✗
Yao et al. [22]	✓	✗	✗	✗	✓	✗
Liu et al. [21]	✓	✗	✗	✗	✓	✗
Lange et al. [26]	✓	✓	✓	✗	✗	✓
Gao et al. [13]	✓	✓	✓	✓	✗	✓
Sallahi and St-Hilaire [37, 38]	✗	✗	✗	✗	✗	✓

Controller 4	Controller 2	Controller 2	Controller 4	Controller 7	Controller 4	Controller 7
Switch 1	Switch 2	Switch 3	Switch 4	Switch 5	Switch 6	Switch 7

Fig. 2 Mapping of particle

Let f be the fitness (objective) function according to Eqs. (7) and (8), d is the number of design variables (or dimensions), and N be the population size. We use these parameters as an inputs for optimization methods (PSO, TLBO, Jaya, and VBO) to calculate optimal placement of controllers (*optmPlacement*) and corresponding optimal total average latency $\left(optmLatency \text{ or } T^{avg-latency}(P)\right)$ in the network.

In Algorithm 1, lines 4–9 and lines 10–16 calculate the $\pi^{avgS2Clatency}(P)$ and $\pi^{avgC2Clatency}(P)$ of a particle respectively. Lines 17–22 are considering load on the switches and capacity of the controller to calculates the $T^{avg-latency}(P)$ of a particle.

Algorithm 1 Calculating the latency of the particle

```
   Input: $X, D, l(s), L(c)$
   Output: $T^{avg-latency}$
1  $X = round(X)$
2  $controllerList$ = Get unique controllers from $X$
3  $k$ = Number of controllers
4  $S2Clatency = 0$
5  for $i \leftarrow 1$ to $n$ do
6      $c$ = Get controller assigned to switch $i$ from $X$
       // Get shortest distance between switch $i$
          and controller $c$
7      $dist = D(i, c)$
8      $S2Clatency = S2Clatency + dist$
9  $\pi^{avgS2Clatency} = S2Clatency/n$
10 $C2Clatency = 0$
11 for $i \leftarrow 1$ to $k$ do
12     for $j \leftarrow i + 1$ to $k$ do
           // Get shortest distance between
              controllers $c_i$ and $c_j$
13         $dist = D(c_i, c_j)$
14         $C2Clatency = C2Clatency + dist$
15 $p_c = \binom{k}{2}$ // number of inter-controllers path
16 $\pi^{avgC2Clatency} = C2Clatency/p_c$
17 $penalty = 0$
18 for $i \leftarrow 1$ to $k$ do
19     $S$ = Get all switches assigned to controller $c_i$
20     $totalLoad$ = Sum up load on switches $\left(l(s)\right)$ in $S$
21     $penalty = penalty + \left(\max(0, totalLoad - L(c_i))\right)^2$
22 $T^{avg-latency} = \left(\pi^{avgS2Clatency} + \pi^{avgC2Clatency}\right) \times \left(1 + penalty\right)$
23 return $T^{avg-latency}$
```

3.2.2 Particle swarm optimization (PSO) for controller placement problem

Eberhart and Kennedy [39] introduced particle swarm optimization (PSO) algorithm in 1995. It is a nature inspired population based optimization algorithm. Cao et al. [13] provides a PSO based solution for CPP in which global latency is minimized for a given controllers. We use PSO algorithm with a different mapping (other than [13]) of the particle, where it finds the required number of controllers needed to be placed in the network, thus fulfilling the first requirement of [8] for CPP. The inertia (w) of the previous update linearly decreased as per Eq. (9).

$$w = w_{initial} + (w_{final} - w_{initial}) \times \frac{I}{I_{\max}}, \tag{9}$$

where I is the number of iterations. X_i is regarded as the current position, V_i is the current velocity vector, r_1 and $r_2 \epsilon (0, 1)$ be the random numbers, c_1 and c_2 be the controlling parameter, *pbest* be the local and *gbest* global best position of the particle. The velocity is resultant of the influence of the personal best position of a particle and global best position. Here, $w_{initial}$ and w_{final} is taken as 0.9 and 0.1 respectively. Constant c_1 and c_2 are set to 2. It is calculated by Eq. (10).

$$V_i = w \times V_i + c_1 \times r_1 \times (pbest_i - X_i) + c_2 \times r_2 \times (gbest - X_i). \tag{10}$$

The new position is updated by adding velocity to current position as given in Eq. (11).

$$X_i = X_i + V_i. \tag{11}$$

In Algorithm 2, lines 9–12 calculate the velocity of each particles and lines 13–14 calculate the position of each particle. Lines 16–18 check whether the latency of each particle is lower or greater than that of local best. Line 19 finds the minimum latency particles from all available local best results.

3.2.3 Teaching-learning-based optimization (TLBO) for controller placement problem

Rao et al. [40] introduced a teaching learning-based optimization (TLBO) in 2011. It works on the influence of a teacher on learners. Like other nature-inspired algorithms, it is also a population-based algorithm and uses a population of solutions to proceed to the global solution. The mapping of TLBO with controller placement problem is done for the first time. But for a fixed number of iterations, TLBO does twice number of function evaluations than PSO. In teacher phase, all students moved to a better mean by the teacher as expressed in Eq. (12).

$$X_i' = X_i + r_t \times (X_{best} - TF \times X_{mean}). \tag{12}$$

In learner phase, we select randomly two students X_i and

X_{peer} and if $f(X_i) < f(X_{peer})$ then modifies the previous solution by Eq. (13) otherwise by Eq. (14).

$$X'_i = X_i + r_s \times (X_i - X_{peer}), \tag{13}$$

$$X'_i = X_i + r_s \times (X_{peer} - X_i). \tag{14}$$

Algorithm 2 Particle swarm optimization (PSO) for CPP

```
   Input: f, d, N, Termination Criteria ($I_{max}$)
   Output: optmPlacement, optmLatency
1  Randomly initialize position of each of N particles as $X_1, X_2, X_3, \dots, X_N$
2  Randomly initialize velocity of each of N particles as $V_1, V_2, V_3, \dots, V_N$
3  Calculate latency of each particle using f
4  Set each particle's pbest equal to their respective positions
5  Set gbest equal to a particle's position having minimum latency
6  Initialize $w_{initial}$ and $w_{final}$
7  while Termination criteria not met do
8      $w = w_{initial} + (w_{final} - w_{initial}) \times \frac{I}{I_{max}}$
9      for i ← 1 to N do
10         $r_1$ = d-dimensional vector having random[0, 1]
11         $r_2$ = d-dimensional vector having random[0, 1]
12         $V_i = w \times V_i + c_1 \times r_1 \times (pbest_i - X_i) + c_2 \times r_2 \times (gbest - X_i)$
13     for i ← 1 to N do
14         $X_i = X_i + V_i$
15     Calculate latency of each particle using f
16     for i ← 1 to N do
17         if latency of $X_i$ is lower than that of $pbest_i$ then
18             $pbest_i = X_i$
19     gbest = Get particle's position having minimum latency
20 optmPlacement = gbest
21 optmLatency = latency corresponding to gbest
22 return optmPlacement, optmLatency
```

Algorithm 3 Teaching-learning-based optimization (TLBO) for CPP

```
   Input: f, d, N, Termination Criteria
   Output: optmPlacement, optmLatency
1  Randomly initialize each of N particles as $X_1, X_2, X_3, \dots, X_N$
2  Calculate latency of each particle using f
3  while Termination criteria not met do
       // Teacher Phase
4      $X_{mean}$ = Calculate mean of each design variable
5      $X_{best}$ = Get particle having minimum latency
6      $r_t$ = d-dimensional vector having random[0, 1]
7      $TF = random\{1, 2\}$
8      for i ← 1 to N do
9          $X'_i = X_i + r_t \times (X_{best} - TF \times X_{mean})$
10     Calculate latency of each new particle using f
11     for i ← 1 to N do
12         if latency of $X'_i$ is lower than that of $X_i$ then
13             $X_i = X'_i$
       // Student Phase
14     $r_s$ = d-dimensional vector having random[0, 1]
15     for i ← 1 to N do
16         $X_{peer}$ = Randomly choose a particle (other than $X_i$) from whole population
17         if latency of $X_i$ is lower than that of $X_{peer}$ then
18             $X'_i = X_i + r_s \times (X_i - X_{peer})$
19         else
20             $X'_i = X_i + r_s \times (X_{peer} - X_i)$
21     Calculate latency of each new particle using f
22     for i ← 1 to N do
23         if latency of $X'_i$ is lower than that of $X_i$ then
24             $X_i = X'_i$
25     optmPlacement = Get particle having minimum latency
26 optmLatency = latency corresponding to optmPlacement
27 return optmPlacement, optmLatency
```

In Algorithm 3, lines 4–13 show the teacher phase, this phase calculates the latency of each new particles using objective function f and compare the latency of new particle with the latency of old particle. Lines 14–20 show the learner phase, this phase calculates the latency of each new particles using objective function f. Lines 21–25 compare the latency of new particle with the latency of old particle and also find the optimum placements.

3.2.4 Jaya algorithm for controller placement problem

Rao [41] introduced Jaya algorithm in 2015 which is also a population based optimization method. The mapping of Jaya with CPP is also done for the first time. It gives better results in some scenarios and some other scenarios its results not good. Overall this algorithm is not suitable for our problem. Here, the motive of this algorithm is to move towards best solution and away from worst solution as in Eq. (15).

$$X'_i = X_i + r_1 \times (X_{best} - |X_i|) - r_2 \times (X_{worst} - |X_i|). \tag{15}$$

In Algorithm 4, lines 8–9 calculate new particles by using particle with minimum and maximum latency. Line 10 calculates the latency of each new particles using objective function f. Lines 11–13 compare the latency of new particle with the latency of old particle and line 14 finds the optimum placements.

3.2.5 Proposed varna-based optimization (VBO) for controller placement problem

Previously, most of the heuristics approach consider the same formulation for all the particles in the population. To work a system well, all components of the system need not do the same task, a variation in their task may improve the performance. We utilized this concept in our algorithm. We get inspired by the working of human society, where people are assigned specific tasks for development of the society.

In our algorithm, particles in the population are classified into two *Varna* (a Sanskrit word, which means Class), namely, *class A* and *class B*. This classification is based on the superiority of particles. Particles have better fitness value belongs to *class A* (like elite group), and rest particles belong to *class B*. The particles in a particular class follow rules of that class and work accordingly. Also, it is not necessary that particles present in a particular class in one generation will always remain in it. In a next generation, it may go to other class as well. So, *Varna* is decided by particle's *Karma* (fitness value), not by their birth.

Algorithm 4 Jaya algorithm for CPP

```
    Input: f, d, N, Termination Criteria
    Output: optmPlacement, optmLatency
1   Randomly initialize each of N particles as $X_1, X_2, X_3, \dots, X_N$
2   Calculate latency of each particle using f
3   while Termination criteria not met do
4       $X_{best}$ = Get particle having minimum latency
5       $X_{worst}$ = Get particle having maximum latency
6       $r_1$ = d-dimensional vector having random[0, 1]
7       $r_2$ = d-dimensional vector having random[0, 1]
8       for i ← 1 to N do
9           $X'_i = X_i + r_1 \times (X_{best} - |X_i|) - r_2 \times (X_{worst} - |X_i|)$
10      Calculate latency of each new particle using f
11      for i ← 1 to N do
12          if latency of $X'_i$ is lower than that of $X_i$ then
13              $X_i = X'_i$
14      optmPlacement = Get particle having minimum latency
15  optmLatency = latency corresponding to optmPlacement
16  return optmPlacement, optmLatency
```

Here, the task for *class A* is exploitation and for *class B* is exploration. The particles in *class A* have the property to move towards the *best* solution and away from the *worst* solution. On the other hand, particles in *class B* interact whole population peer to peer, and their movement is decided by respective peer particles. For deciding the sizes of classes, we take a fixed fraction *(α)* of the population in *class A* and rest in *class B*. We recommend the value of α to be from 0.05 to 0.20, in this paper we set $\alpha = 0.10$ for the experiment. And we set peer constants as $c_1 = 1.50$ and $c_2 = 1.25$. The values for c_1 and c_2 are kept higher than 1 to cover the search regions around the better counterpart. The value of c_1 is still kept higher than c_2 as there is more chance of promising solution around particle having the best solution. Figure 3 shows the flowchart of VBO algorithm.

Particles in *class A* move towards the best solution and simultaneously moving away from the worst solution is given by Eq. (16).

$$X'_i = X_i + r_A \times (X_{best} - X_{worst}). \tag{16}$$

For each particles X_i in *class B*, we randomly choose a particle from whole population as X_{peer}. If fitness of X_i is better than that of X_{peer} we move that particle towards best solution and away from peer solution (see Eq. (17)).

$$X'_i = X_i + c_1 \times r_B \times (X_{best} - X_{peer}). \tag{17}$$

If fitness value of X_i is worse than that of X_{peer} we move particle towards X_{peer} (see Eq. (18)).

$$X'_i = X_i + c_2 \times r_B \times (X_{peer} - X_i). \tag{18}$$

If both particle have same fitness value then new position is updated as in range zero to twice of current position (see Eq. (19)).

$$X'_i = 2 \times r_B \times X_i. \tag{19}$$

In Algorithm 5, lines 9–11 calculate positions of new particles for *class A* by using particle with minimum and maximum latency. Lines 12–20 calculate positions of new particles for *class B* by using latencies of peer particles. Line 21 calculates the latency of each new particles using objective

Algorithm 5 Proposed varna-based optimization (VBO) for CPP

```
    Input: f, d, N, Termination Criteria
    Output: optmPlacement, optmLatency
1   $\alpha$ = Initialize fraction of population for class A
2   $n_A = \alpha \times N$
3   Randomly initialize each of N particles as $X_1, X_2, X_3, \dots, X_N$
4   Calculate latency of each particle using f
5   while Termination criteria not met do
6       Sort particles according to their latency in ascending order
7       $X_{best}$ = Get particle having minimum latency
8       $X_{worst}$ = Get particle having maximum latency
9       for i ← 1 to $n_A$ do
10          $r_A$ = random[0, 1] vector of d dimensions
11          $X'_i = X_i + r_A \times (X_{best} - X_{worst})$
12      for i ← $n_A + 1$ to N do
13          $X_{peer}$ = Randomly choose a particle (other than $X_i$) from whole
            population
14          $r_B$ = random[0, 1] vector of d dimensions
15          if latency of $X_i$ is lower than that of $X_{peer}$ then
16              $X'_i = X_i + c_1 \times r_B \times (X_{best} - X_{peer})$
17          else if latency of $X_i$ is higher than that of $X_{peer}$ then
18              $X'_i = X_i + c_2 \times r_B \times (X_{peer} - X_i)$
19          else
                // when both $X_i$ and $X_{peer}$ have same latency
20              $X'_i = 2 \times r_B \times X_i$
21      Calculate latency of each new particle using f
22      for i ← 1 to N do
23          if latency of $X'_i$ is lower than that of $X_i$ then
24              $X_i = X'_i$
25      optmPlacement = Get particle having minimum latency
26  optmLatency = latency corresponding to optmPlacement
27  return optmPlacement, optmLatency
```

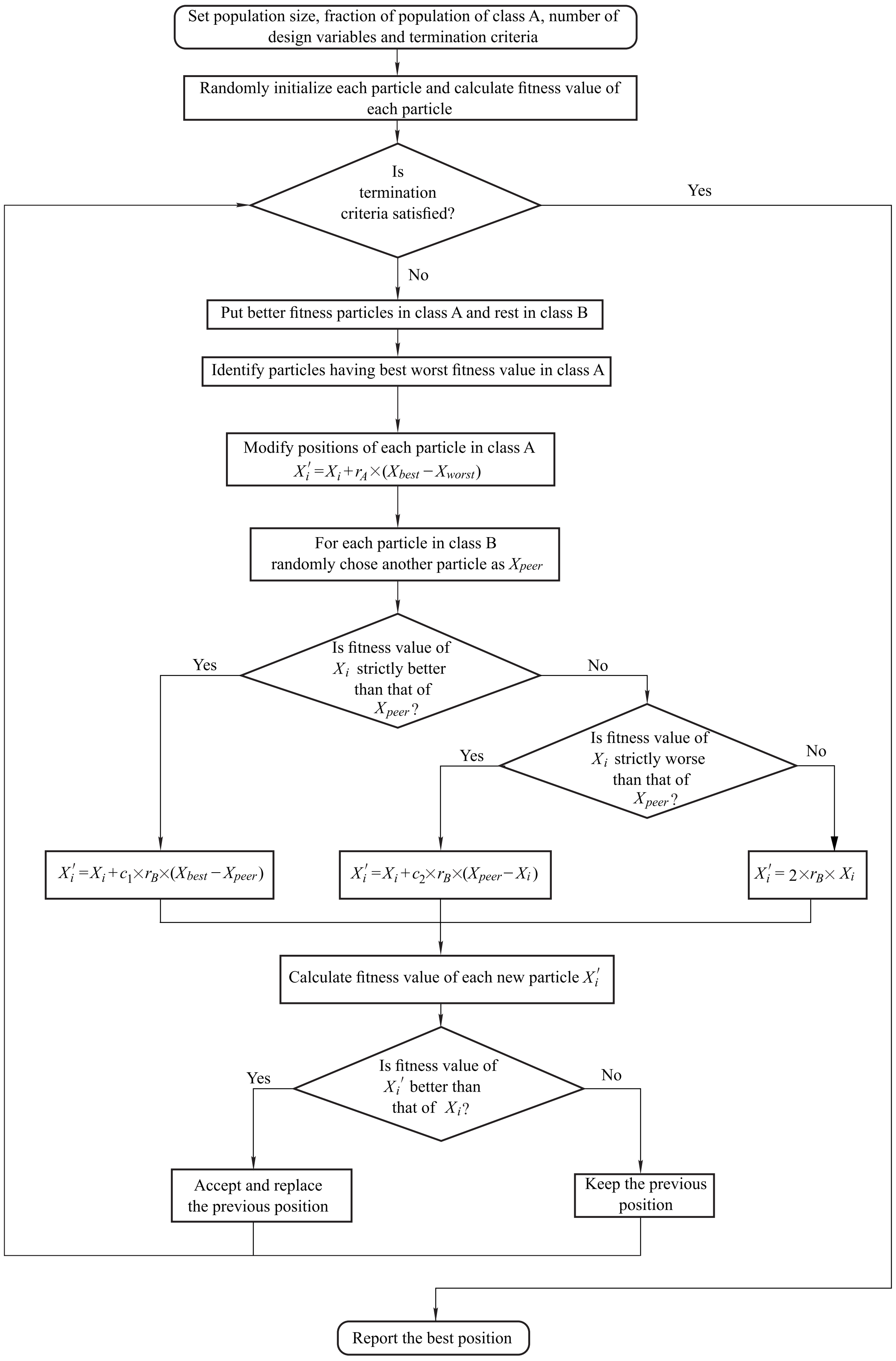

Set population size, fraction of population of class A, number of design variables and termination criteria
Randomly initialize each particle and calculate fitness value of each particle
Is termination criteria satisfied?
Yes
No
Put better fitness particles in class A and rest in class B
Identify particles having best worst fitness value in class A
Modify positions of each particle in class A $X_i' = X_i + r_A \times (X_{best} - X_{worst})$
For each particle in class B randomly chose another particle as X_{peer}
Is fitness value of X_i strictly better than that of X_{peer}?
Yes
No
Is fitness value of X_i strictly worse than that of X_{peer}?
Yes
No
$X_i' = X_i + c_1 \times r_B \times (X_{best} - X_{peer})$
$X_i' = X_i + c_2 \times r_B \times (X_{peer} - X_i)$
$X_i' = 2 \times r_B \times X_i$
Calculate fitness value of each new particle X_i'
Is fitness value of X_i' better than that of X_i?
Yes
No
Accept and replace the previous position
Keep the previous position
Report the best position

Fig. 3 Flowchart of VBO algorithm

function *f*. Lines 22–24 compare the latency of new particle with the latency of old particle, and line 25 finds the optimum placements.

4 Results and discussions

As an experimental platform, we have used MATLAB 2017a for the simulation of the optimization based solution of CPP and also used python language for the simulation of the clustering based solution of CPP. The system consists of Windows 8.1 (64-bit) with Intel Core i7-4770 CPU @ 3.40 GHz and 16 GB RAM. For all experiments of optimization methods, we have set population size equal to 100. In this section, we provide the comparative results of both clustering based solutions and heuristic based optimization solutions for CPP. Our experimental results tell that optimization based solutions are better than clustering based solutions. We also show that our proposed VBO algorithm gives better convergence rate as compared to other optimization methods.

4.1 Comparative results analysis of both clustering and heuristic based optimization solutions for CPP

In our experiment, we consider four topologies (Abilene, Savvis, Biznet, and Internet2 OS3E) from Internet Zoo topologies [32]. As shown in Fig. 4(a), it can say that optimization based solution is better than the clustering based solution for un-capacitated controller placement problem. VBO gives better results for less number of switches. If the number of switches is increased, TLBO gives better results. As shown in Fig. 4(b), it is clear that optimization based solutions provide better results over the clustering based solutions. In Capacitated CPP, our proposed VBO algorithm gives better results as compared to other optimizations algorithms like PSO, TLBO, and Jaya.

4.2 Result analysis of our proposed VBO algorithm over other heuristic based optimization solutions for CPP

We take a set of 262 publicly available network topologies in our experiments and use PSO and TLBO algorithms for finding optimal placement in a given topology. Experimental results show that TLBO outperforms PSO and VBO outperforms TLBO in all scenarios for all topologies. Here, we give the results for two most popular topologies (Internet2 OS3E and Savvis) for all 12 possible scenarios.

4.2.1 Scenarios for uncapacitated controller placement problem

In this section, we consider 4 different scenarios (1 to 4) which are already discussed in Section 3.2. These scenarios use controllers with unlimited capacity, i.e., controller capacity is not a constraint. We run our proposed VBO algorithm as well as PSO, TBLO, and Jaya algorithms for all four scenarios. As shown in Figs. 5–8, and from Tables 3–6, it can be observed that TLBO gives better results as compared to PSO, Jaya, and VBO for uncapacitated CPP. We have also seen that VBO provides better convergence rate as compared to PSO and Jaya. But, it is not clear that either PSO gives better convergence rate or Jaya algorithm. Sometimes PSO gives better results, and other times Jaya algorithm gives better convergence rate in some cases. Moreover, after analysis of all results, we can say that TLBO gives better convergence rate compared to PSO, Jaya, and VBO algorithm.

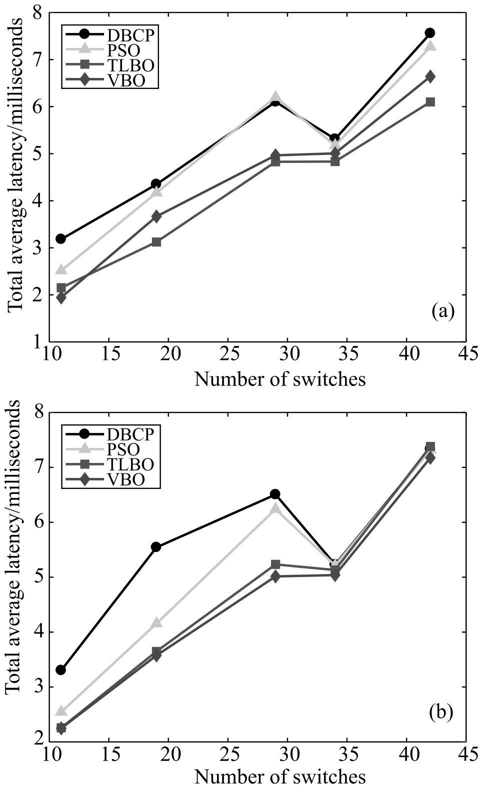


Fig. 4 Comparison of clustering and optimization algorithms on controller placement problem. (a) Un-capacitated CPP; (b) capacitated CPP

Table 3 Comparative results of Scenario 1 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	4.93	5.58	5.18	0.15
	TLBO	**4.65**	5.41	**4.89**	0.26
	JAYA	5.68	6.20	5.89	0.13
	VBO	4.73	**5.18**	4.95	**0.13**
Savvis	PSO	3.84	4.73	4.18	0.18
	TLBO	2.63	**3.93**	**3.23**	0.53
	JAYA	3.68	4.29	3.91	**0.15**
	VBO	**2.63**	4.05	3.61	0.23

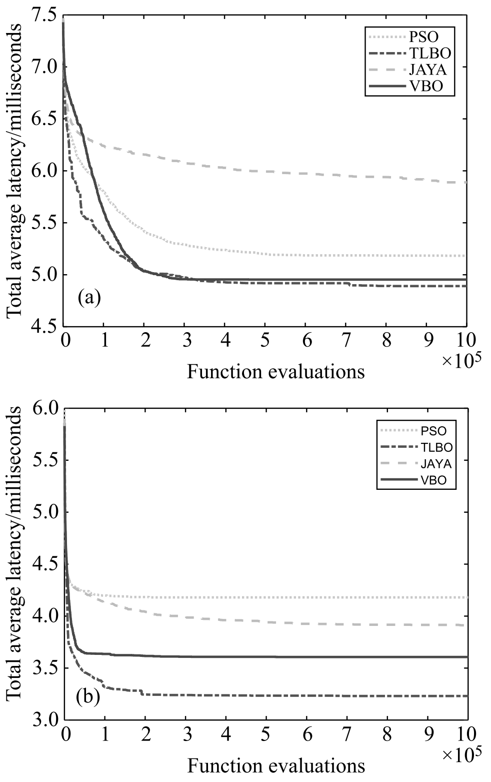


Fig. 5 Convergence plots for uncapacitated controller placement problem by different optimization methods in Scenario 1. (a) Internet2 OS3E topology; (b) Savvis topology

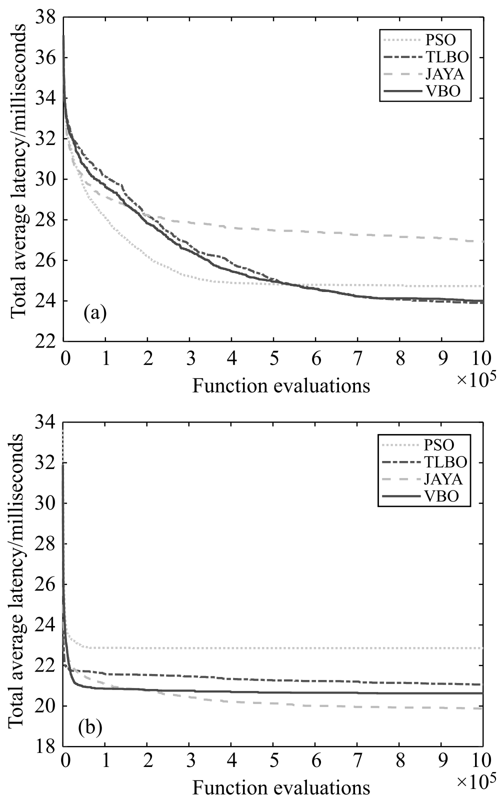


Fig. 7 Convergence plots for uncapacitated controller placement problem by different optimization methods in Scenario 3. (a) Internet2 OS3E topology; (b) Savvis topology

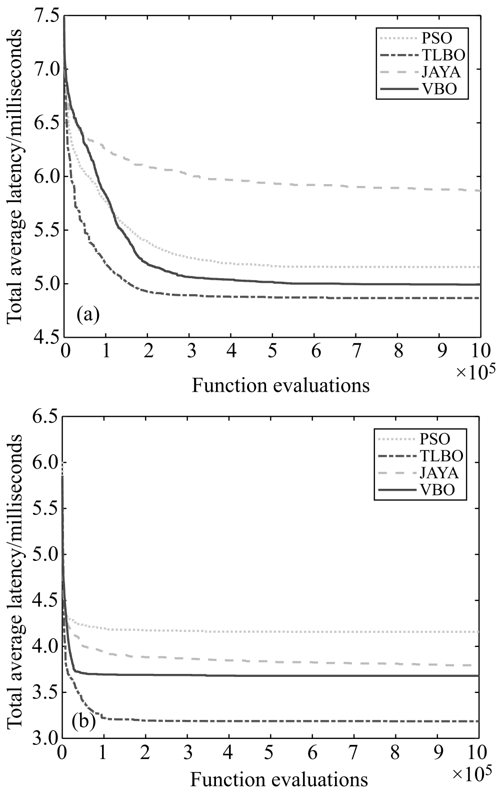


Fig. 6 Convergence plots for uncapacitated controller placement problem by different optimization methods in Scenario 2. (a) Internet2 OS3E topology; (b) Savvis topology

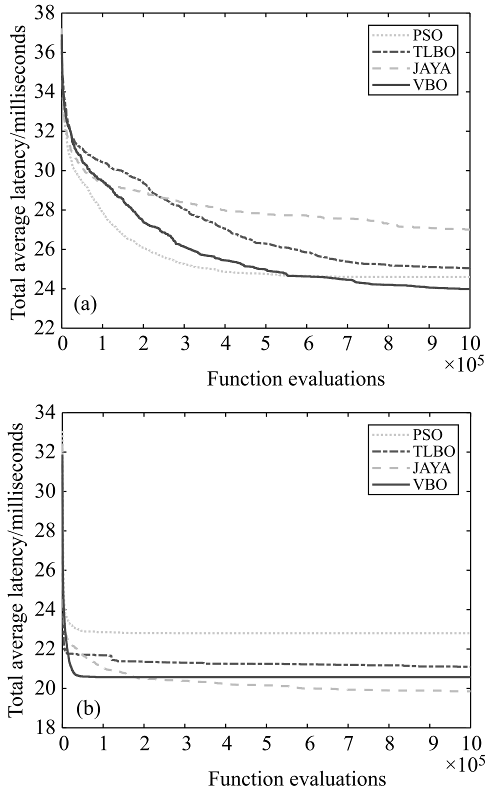


Fig. 8 Convergence plots for uncapacitated controller placement problem by different optimization methods in Scenario 4. (a) Internet2 OS3E topology; (b) Savvis topology

Table 4 Comparative results of Scenario 2 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	4.81	5.44	5.16	0.17
	TLBO	**4.65**	5.37	**4.87**	0.25
	JAYA	5.59	6.41	5.87	0.20
	VBO	4.69	**5.33**	4.99	**0.15**
Savvis	PSO	3.80	4.50	4.16	0.18
	TLBO	**2.63**	**3.86**	**3.19**	0.49
	JAYA	2.63	4.32	3.79	0.42
	VBO	3.34	3.96	3.68	**0.17**

Table 5 Comparative results of Scenario 3 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	23.28	**26.63**	24.73	**0.88**
	TLBO	22.13	27.53	**23.90**	1.09
	JAYA	23.96	30.15	26.90	1.41
	VBO	**20.91**	26.93	24.01	1.49
Savvis	PSO	19.30	27.06	22.86	1.78
	TLBO	19.35	21.71	21.06	0.91
	JAYA	**18.92**	**21.13**	**19.87**	**0.65**
	VBO	19.35	21.94	20.63	0.70

Table 6 Comparative results of Scenario 4 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	22.58	27.13	24.60	**1.00**
	TLBO	22.49	28.42	25.03	1.50
	JAYA	24.57	30.77	27.01	1.42
	VBO	**22.35**	**26.47**	**23.99**	1.04
Savvis	PSO	18.99	26.45	22.80	1.71
	TLBO	**15.05**	21.71	21.10	1.37
	JAYA	18.92	**21.07**	**19.85**	**0.65**
	VBO	18.92	21.96	20.57	0.95

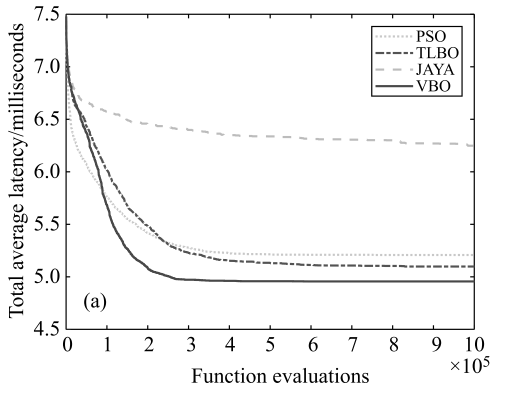
4.2.2 Scenarios for capacitated controller placement problem

We consider eight different Scenarios (5–12) for capacitated CPP. The capacitated controller may have the same and different capacity. We assume that capacity of each controller is same in Scenarios 5–8 and either same or different in Scenarios 9–12. For example, we take random values of capacity of controller, i.e., $L(c)$ and load on switch, i.e., $l(s)$ as given in Tables 7 and 8 for Internet2 OS3E and Savvis, respectively.

We run VBO algorithm as well as PSO, TBLO, and Jaya algorithms for capacitated controller placement problem (all controllers have equal packet processing power). We consider all four possible Scenarios (5–8) and find out the results and convergence rate for all these optimization methods. As shown in Figs. 9–12, and from Tables 9–12, it is clear that VBO gives better convergence rate as compared to PSO, TLBO, and Jaya algorithms. In most of the scenarios, start-up of PSO gives better convergence rate as compare to other, but after some function evaluations, it goes uniform, and its results are not better than VBO and TLBO. We have also seen that TLBO provides better convergence rate as compare to PSO and Jaya. Moreover, after analysis of results of Scenar-

Table 7 The values of $l(s)$ and $L(c)$ used in Internet2 OS3E

Switch No.	$l(s)$	$L(c)$	Switch No.	$l(s)$	$L(c)$
1	4	25	18	10	129
2	5	108	19	3	48
3	6	12	20	2	163
4	5	102	21	3	109
5	8	136	22	3	106
6	4	48	23	4	38
7	4	82	24	5	24
8	2	101	25	5	51
9	7	20	26	3	147
10	6	89	27	9	156
11	1	113	28	6	122
12	6	45	29	8	126
13	5	144	30	7	47
14	10	165	31	3	102
15	5	145	32	1	104
16	6	91	33	7	75
17	5	55	34	2	168



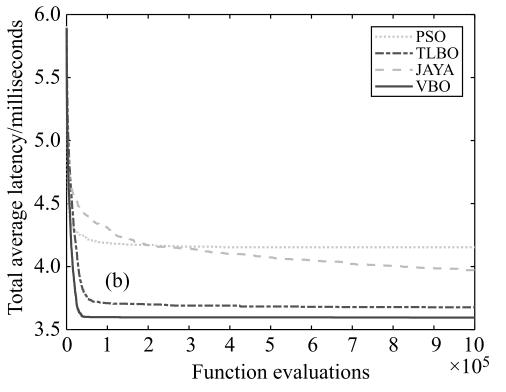


Fig. 9 Convergence plots for weighted capacitated controller placement problem by different optimization methods in Scenario 5. (a) Internet2 OS3E topology; (b) Savvis topology

ios (5–8), it is clear that VBO dominates PSO, TLBO, and Jaya algorithms. TLBO gives much better results as compared to PSO and Jaya algorithms.

Table 8 The values of $l(s)$ and $L(c)$ used in Savvis

Switch No.	$l(s)$	$L(c)$	Switch No.	$l(s)$	$L(c)$
1	5	16	11	10	12
2	4	19	12	1	10
3	8	19	13	4	11
4	5	23	14	6	10
5	4	10	15	3	25
6	6	12	16	9	14
7	7	13	17	6	11
8	7	21	18	8	11
9	2	21	19	5	24
10	5	10			

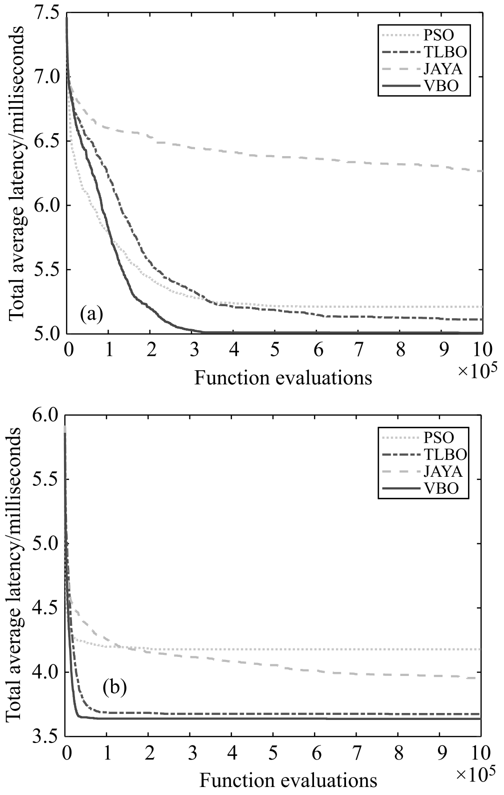


Fig. 10 Convergence plots for unit weight capacitated controller placement problem by different optimization methods in Scenario 6. (a) Internet2 OS3E topology; (b) Savvis topology

We also run VBO algorithm as well as PSO, TBLO, and Jaya algorithms for capacitated controller placement problem (all controllers differ in packet processing power). We consider all four possible Scenarios (9 to 12) and find out the results and convergence rate for all these optimization methods. As shown in Figs. 13(a), 14, 15(a), and 16, and Tables 13–16, it is observed that VBO gives better convergence rate as compared to PSO, TLBO, and Jaya algorithms.

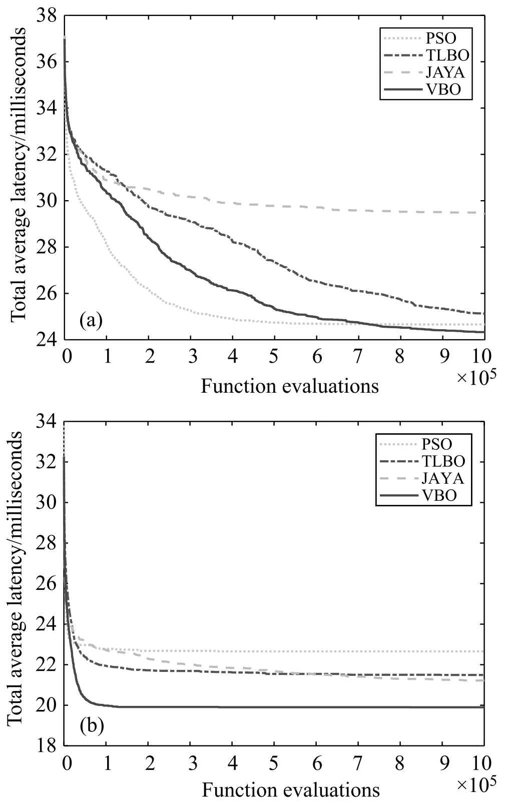


Fig. 11 Convergence plots for weighted capacitated controller placement problem by different optimization methods in Scenario 7. (a) Internet2 OS3E topology; (b) Savvis topology

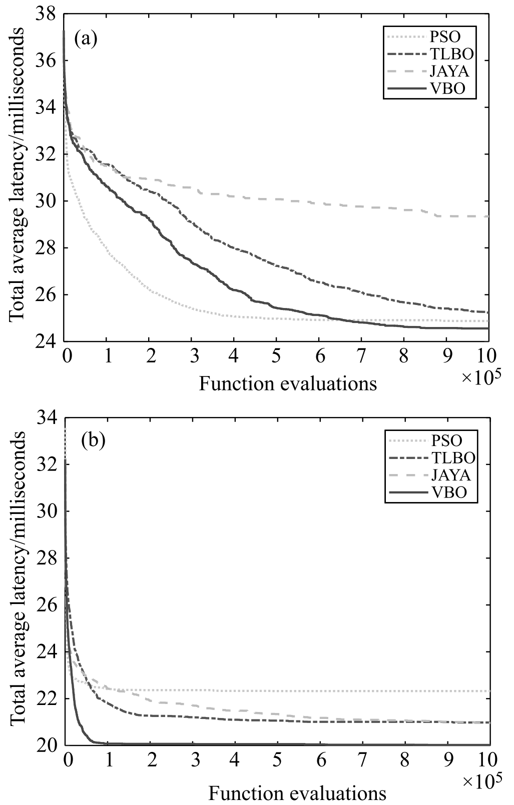


Fig. 12 Convergence plots for weighted capacitated controller placement problem by different optimization methods in Scenario 8. (a) Internet2 OS3E topology; (b) Savvis topology

Table 9 Comparative results of Scenario 5 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	4.79	5.56	5.21	0.16
	TLBO	**4.62**	5.41	5.10	0.18
	JAYA	5.91	6.52	6.25	**0.14**
	VBO	4.64	**5.31**	**4.96**	0.17
Savvis	PSO	3.83	4.61	4.15	0.20
	TLBO	3.41	4.02	3.68	0.15
	JAYA	3.74	4.25	3.97	0.14
	VBO	**3.35**	**3.89**	**3.60**	**0.12**

Table 10 Comparative results of Scenario 6 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	4.79	5.60	5.21	0.19
	TLBO	**4.64**	5.40	5.11	0.15
	JAYA	6.01	6.54	6.27	0.15
	VBO	4.69	**5.28**	**5.01**	**0.14**
Savvis	PSO	3.65	4.80	4.18	0.26
	TLBO	**3.39**	**3.92**	3.67	0.13
	JAYA	3.76	4.26	3.95	0.14
	VBO	3.45	3.98	**3.64**	**0.11**

Table 11 Comparative results of Scenario 7 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	23.17	**26.51**	24.66	**0.81**
	TLBO	22.10	28.77	25.13	1.64
	JAYA	27.45	31.09	29.44	0.98
	VBO	**21.59**	27.54	**24.33**	1.33
Savvis	PSO	19.13	25.22	22.66	1.61
	TLBO	19.96	23.10	21.49	0.94
	JAYA	20.04	22.36	21.22	**0.57**
	VBO	**18.62**	**21.66**	**19.90**	0.83

Table 12 Comparative results of Scenario 8 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	22.52	27.85	24.88	**1.04**
	TLBO	22.62	28.17	25.23	1.40
	JAYA	27.10	31.25	29.34	1.23
	VBO	**22.00**	**26.69**	**24.56**	1.25
Savvis	PSO	19.93	25.86	22.32	1.45
	TLBO	19.24	22.97	20.98	0.89
	JAYA	19.79	24.05	20.96	0.97
	VBO	**18.15**	**22.18**	**20.02**	**0.76**

As shown in Fig. 13(b), we see that TLBO gives better convergence rate as compared to VBO and other methods. As shown in Fig. 15(b), we see that Jaya gives better convergence rate as compared to VBO and other methods. We have also seen that TLBO provides better convergence rate as compare to PSO and Jaya. Moreover, after analysis of results of Scenarios (9–12), it is clear that VBO dominates PSO, TLBO, and Jaya algorithms. TLBO gives much better results as compared to PSO and Jaya algorithms.

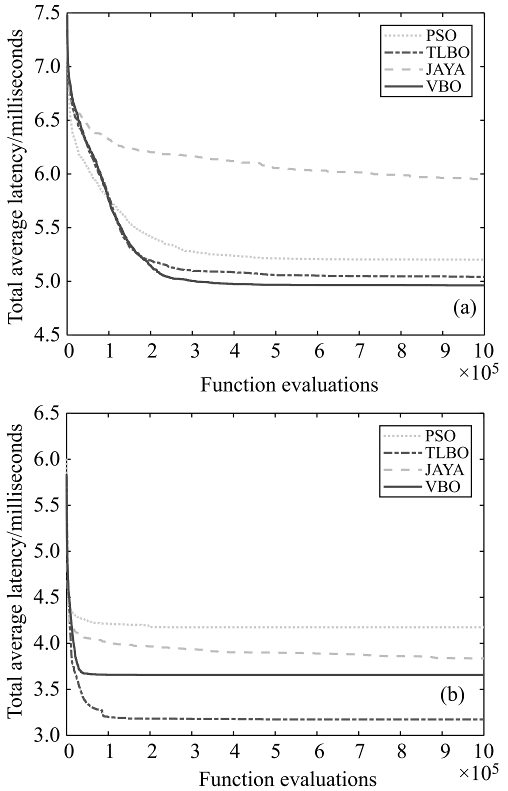


Fig. 13 Convergence plots for unit weight capacitated controller placement problem by different optimization methods in Scenario 9. (a) Internet2 OS3E topology; (b) Savvis topology

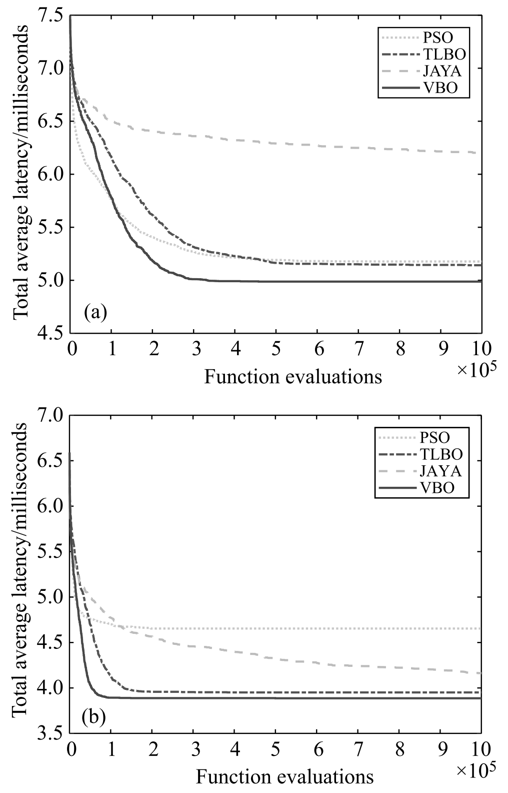


Fig. 14 Convergence plots for unit weight capacitated controller placement problem by different optimization methods in Scenario 10. (a) Internet2 OS3E topology; (b) Savvis topology

Table 13 Comparative results of Scenario 9 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	4.84	5.51	5.20	**0.14**
	TLBO	**4.55**	5.29	5.04	0.17
	JAYA	5.68	6.42	5.95	0.20
	VBO	4.64	**5.21**	**4.96**	0.15
Savvis	PSO	3.75	4.56	4.17	0.20
	TLBO	2.63	**3.84**	**3.17**	0.51
	JAYA	**2.63**	4.11	3.84	0.26
	VBO	3.34	3.97	3.66	**0.16**

Table 14 Comparative results of Scenario 10 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	**4.69**	5.64	5.18	0.20
	TLBO	4.80	5.42	5.14	0.15
	JAYA	5.91	6.64	6.20	**0.13**
	VBO	4.70	**5.29**	**4.99**	0.16
Savvis	PSO	4.20	5.35	4.65	0.25
	TLBO	**3.66**	4.21	3.95	0.13
	JAYA	3.77	4.84	4.16	0.29
	VBO	3.71	**4.08**	**3.89**	**0.10**

Table 15 Comparative results of Scenario 11 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	23.01	27.26	24.98	**1.04**
	TLBO	21.97	28.60	24.17	1.76
	JAYA	24.47	30.82	27.30	1.35
	VBO	**21.18**	**26.75**	**23.97**	1.22
Savvis	PSO	21.01	26.44	22.98	1.31
	TLBO	**15.05**	22.73	21.02	2.15
	JAYA	18.91	**22.15**	**20.05**	**0.88**
	VBO	18.92	22.60	20.79	0.94

Table 16 Comparative results of Scenario 12 (in milliseconds)

Topology	Algo	Best	Worst	Mean	StdDev
Internet2	PSO	23.27	**26.91**	24.89	**0.85**
	TLBO	**22.75**	29.61	26.14	1.91
	JAYA	26.87	30.84	28.87	1.07
	VBO	23.01	26.95	**24.53**	1.15
Savvis	PSO	23.02	31.04	25.87	2.04
	TLBO	**19.78**	24.05	21.59	1.01
	JAYA	20.70	25.50	23.27	1.24
	VBO	19.94	**23.66**	**21.03**	**0.92**

After analysis of results of all twelve scenarios, it is clear that VBO gives better results as compared to PSO, TLBO, and Jaya methods for eight Scenarios (5–12). These eight scenarios are considered the capacity of the controller as a constraint. However, VBO does not give much better results as compared to these optimization methods for remaining four Scenarios (1–4) and these scenarios do not consider the capacity of the controller as a constraint. In Scenarios 1–4, TLBO gives better results as compared to PSO, Jaya,

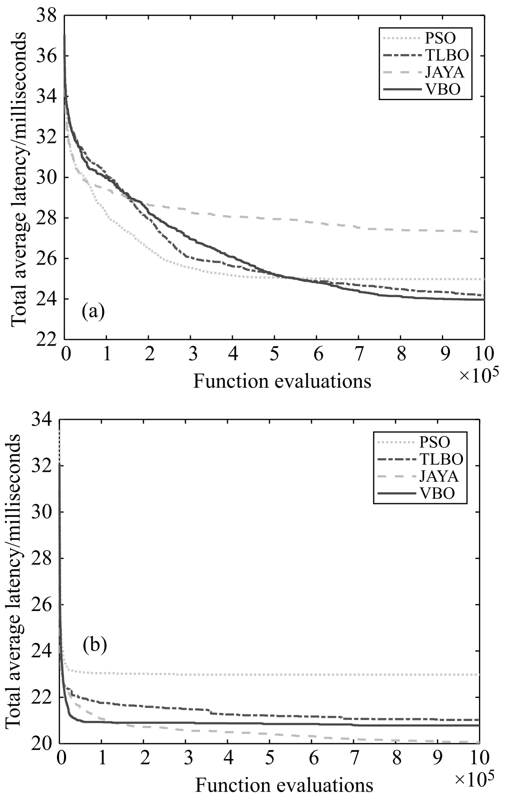


Fig. 15 Convergence plots for weighted capacitated controller placement problem by different optimization methods in Scenario 11. (a) Internet2 OS3E topology; (b) Savvis topology

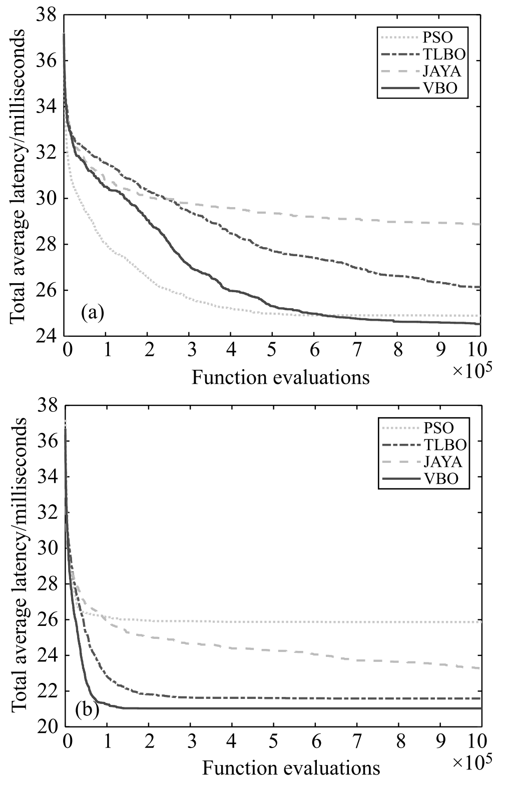


Fig. 16 Convergence plots for weighted capacitated controller placement problem by different optimization methods in Scenario 12. (a) Internet2 OS3E topology; (b) Savvis topology

and VBO but still VBO gives better results as compared to PSO and Jaya.

5 Testing of VBO on benchmark functions

In this section, we have performed the different experiment on unconstrained and constrained benchmark functions to check the effectiveness of VBO with other optimization algorithms. These benchmark functions have different characteristics and functionality such as uni-model, multi-model, regular, non-regular, separable, non-separable, etc. A function is having only one local optimum (minima or maxima), and more than one local optima are called uni-model, and more than one local optima are called multi-model. A function is regular if it is differentiable at every point of its search space otherwise non-regular.

5.1 Experiments on unconstrained benchmark functions

We used six different unconstrained benchmark functions with different functionality and characteristics. Details of these functions are considered by Akay and Karaboga [42] as given in Table 17. In this experiment, we test our proposed VBO algorithm on these six benchmark functions and compare its results with PSO, TLBO, and Jaya. An optimization algorithm is tested for 100 independent runs with the population size of 100, different dimension size (D= 2, 3, 5, 10, 20, and 30) and the number of generations is taken as 1000. The comparison of mean and standard deviation (SD) of PSO, TLBO, Jaya, and VBO for these six benchmark functions is given in Table 18. Figs. 17–22 show the convergence plots of VBO with PSO, TLBO, and Jaya algorithms with different dimension size such as D = 2, 3, 5, 10, 20, and 30. From all these six Figs. 17–22, it is clear that VBO is an effective approach for finding the optimal solutions.

Table 17 Details of benchmark functions considered by Akay and Karaboga [42]

Name	Formulation	Search space	Multimodal?	Separable?	Regular?
Sphere	$\sum_{i=1}^{D} x_i^2$	[–100, 100]	No	Yes	Yes
Rosenbrock	$\sum_{i=1}^{D-1}\left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	[–30, 30]	No	No	Yes
Schwefel	$418.9829D - \sum_{i=1}^{D}\left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$	[–500, 500]	Yes	Yes	No
Rastrigin	$\sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	[–5.12, 5.12]	Yes	Yes	Yes
Griewank	$1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	[–600, 600]	Yes	No	Yes
Ackley	$20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$	[–32, 32]	Yes	No	Yes

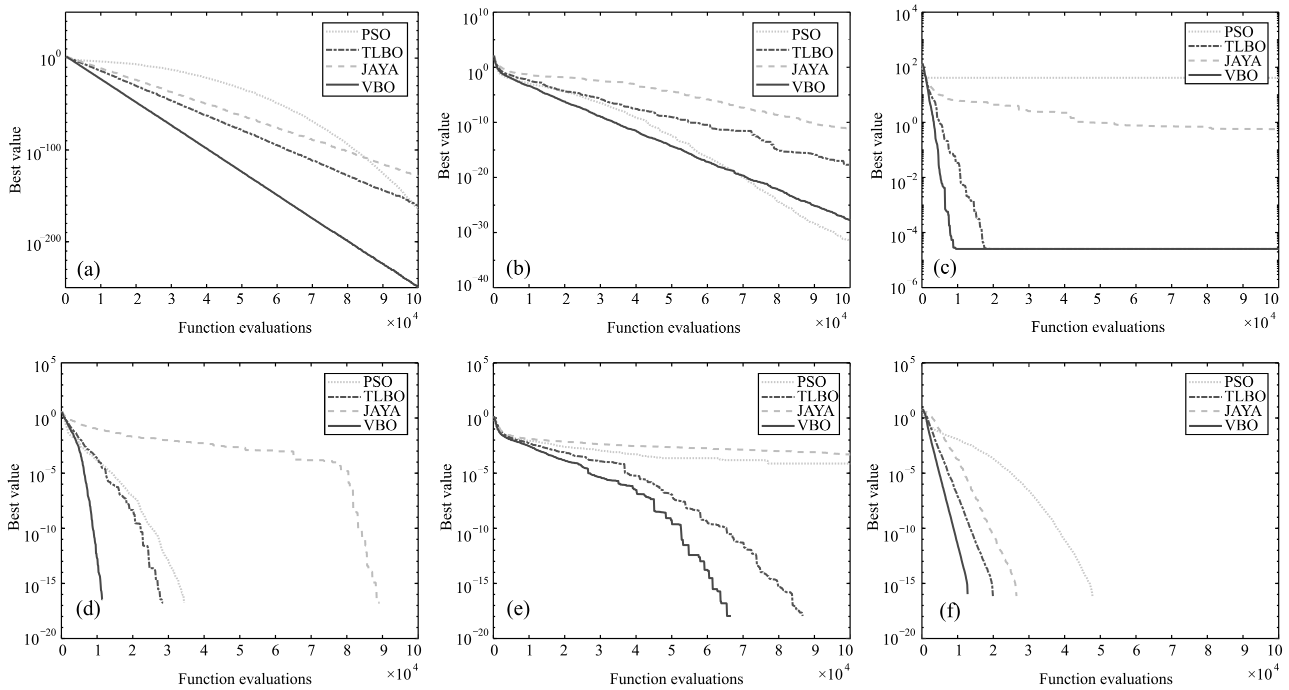


Fig. 17 Convergence plots of VBO algorithm with other optimization algorithms on six benchmark functions (D = 2). (a) Sphere; (b) Rosenbrock; (c) Schwefel; (d) Rastrigin; (e) Griewank; (f) Ackley

Table 18 Comparison of mean and standard deviation (SD) of PSO, TLBO, Jaya, and VBO for unconstrained benchmark six functions

Function	D	PSO		TBLO		Jaya		VBO	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD
Sphere	2	1.69E-166	0.00E+00	3.70E-161	2.22E-160	3.16E-129	2.64E-128	3.13E-250	0.00E+00
	3	1.57E-148	8.95E-148	2.19E-127	1.94E-126	2.76E-85	1.78E-84	2.68E-186	0.00E+00
	5	4.47E-120	2.82E-119	1.73E-105	6.84E-105	6.94E-47	3.77E-46	7.97E-134	2.76E-133
	10	7.87E-69	3.55E-68	2.12E-85	4.85E-85	1.22E-17	1.49E-17	6.38E-89	3.14E-88
	20	7.74E-26	3.89E-25	3.37E-70	9.14E-70	6.85E-04	3.35E-04	4.03E-56	7.87E-56
	30	1.79E-13	1.37E-12	2.92E-64	3.60E-64	2.20E+00	8.00E-01	5.81E-41	9.69E-41
Rosenbrock	2	3.61E-32	2.16E-31	2.13E-18	1.20E-17	7.42E-12	3.00E-11	1.86E-28	7.08E-28
	3	1.38E-03	1.18E-02	3.20E-06	9.20E-06	3.16E-01	2.28E+00	8.91E-07	3.40E-06
	5	2.19E-01	5.50E-01	4.06E-02	3.91E-01	1.62E+00	4.31E+00	2.37E-02	1.16E-01
	10	3.37E+00	1.48E+00	5.62E-01	6.09E-01	7.04E+00	1.03E+01	4.40E-01	5.74E-01
	20	2.19E+01	2.04E+01	1.53E+01	5.62E-01	6.59E+01	5.24E+01	9.94E+00	6.71E+00
	30	3.90E+01	2.65E+01	2.60E+01	3.92E-01	6.45E+02	4.97E+02	2.27E+01	1.38E+01
Schwefel	2	4.14E+01	6.12E+01	2.54E-05	0.00E+00	5.67E-01	2.85E+00	2.54E-05	0.00E+00
	3	1.29E+02	8.83E+01	1.18E+00	1.17E+01	2.08E+01	4.32E+01	3.55E+00	2.02E+01
	5	4.17E+02	1.39E+02	7.49E+01	8.08E+01	1.29E+02	1.52E+02	5.71E+01	7.92E+01
	10	1.47E+03	3.00E+02	4.06E+02	1.83E+02	1.00E+03	3.801E+02	3.68E+02	1.90E+02
	20	3.69E+03	5.29E+02	2.30E+03	7.34E+02	3.79E+03	5.20E+02	1.18E+03	4.17E+02
	30	5.84E+03	8.05E+02	4.88E+03	1.26E+03	7.04E+03	5.82E+02	2.67E+03	9.12E+02
Rastrigin	2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	3	4.97E-02	2.16E-01	0.00E+00	0.00E+00	1.16E-01	1.64E-01	0.00E+00	0.00E+00
	5	1.28E+00	9.13E-01	9.94E-03	9.89E-02	4.03E+00	1.13E+00	0.00E+00	0.00E+00
	10	7.45E+00	3.63E+00	1.12E+00	1.07E+00	3.37E+01	5.28E+00	0.00E+00	0.00E+00
	20	1.90E+01	6.71E+00	5.15E+00	2.79E+00	1.31E+02	1.32E+01	2.57E+00	3.09E+00
	30	2.97E+01	7.80E+00	7.53E+00	4.74E+00	2.43E+02	1.56E+01	7.31E+00	2.98E+00
Griewank	2	7.39E-05	7.35E-04	0.00E+00	0.00E+00	4.84E-04	6.83E-04	0.00E+00	0.00E+00
	3	2.56E-03	3.92E-03	3.39E-04	1.01E-03	1.69E-02	6.73E-03	4.04E-05	2.23E-04
	5	1.45E-02	7.50E-03	1.38E-02	9.01E-03	1.21E-01	3.34E-02	3.70E-03	4.67E-03
	10	6.94E-02	2.58E-02	5.51E-03	9.59E-03	4.81E-01	8.52E-02	4.89E-07	2.98E-06
	20	3.42E-02	2.55E-02	3.45E-04	1.74E-03	5.86E-01	1.06E-01	8.08E-06	6.95E-05
	30	1.27E-02	1.45E-02	0.00E+00	0.00E+00	9.86E-01	4.10E-02	1.04E-05	4.78E-05
Ackley	2	–8.88E-16	0.00E+00	–8.88E-16	0.00E+00	–8.88E-16	0.00E+00	–8.88E-16	0.00E+00
	3	–8.88E-16	0.00E+00	–8.88E-16	0.00E+00	–8.88E-16	0.00E+00	–8.88E-16	0.00E+00
	5	–6.75E-16	8.43E-16	–8.88E-16	0.00E+00	–7.10E-16	7.74E-16	–8.88E-16	0.00E+00
	10	2.66E-15	0.00E+00	1.98E-15	1.39E-15	1.09E-06	6.67E-06	–8.88E-16	0.00E+00
	20	8.73E-15	6.65E-15	1.66E-06	1.65E-05	7.01E-01	3.14E+00	2.59E-15	4.97E-16
	30	1.10E-08	2.81E-08	2.25E-02	2.20E-01	4.91E+00	5.78E+00	2.66E-15	0.00E+00

5.2 Experiments on constrained benchmark functions

We used six different constrained benchmark functions with different functionality and characteristics. These functions are considered by Liang et al. [43]. In this experiment, we test our proposed VBO algorithm on these six benchmark functions and compare its results with PSO, TLBO, and Jaya. An optimization algorithm is tested for 100 independent runs with the population size of 100 and number of generations is taken as 2,000. The comparison of mean and standard deviation of PSO, TLBO, Jaya, and VBO for these six benchmarks functions are given in Table 19. It is clear that VBO is an effective approach for finding the optimal solutions.

5.2.1 Constrained benchmark function 1

This test function is minimization problem which is quadratic in nature. It has nine linear inequality constraints and thirteen design variables. The ratio between the feasible reason and search space is about 0.0111%.

$$\textbf{Minimize } f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{13} x_i. \tag{20}$$

subject to:

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leqslant 0,$$
$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leqslant 0,$$
$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{10} + x_{12} - 10 \leqslant 0,$$

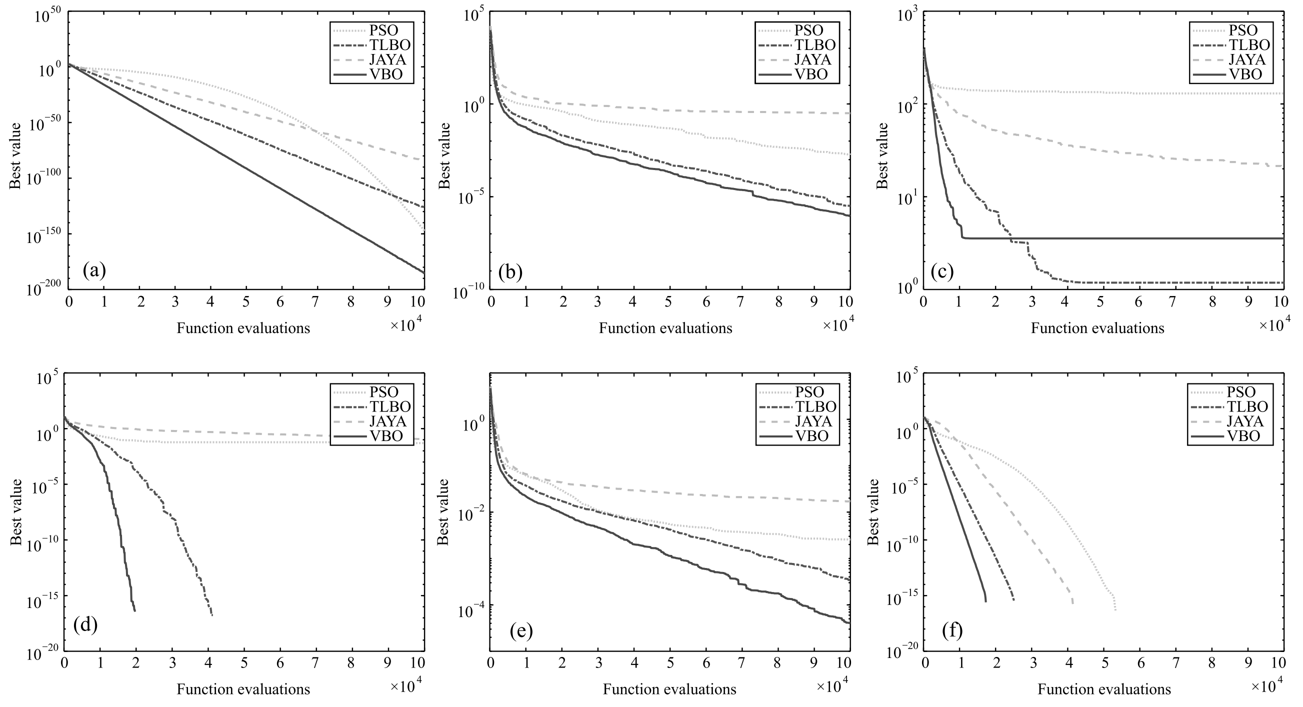


Fig. 18 Convergence plots of VBO algorithm with other optimization algorithms on six benchmark functions (D = 3). (a) Sphere; (b) Rosenbrock; (c) Schwefel; (d) Rastrigin; (e) Griewank; (f) Ackley

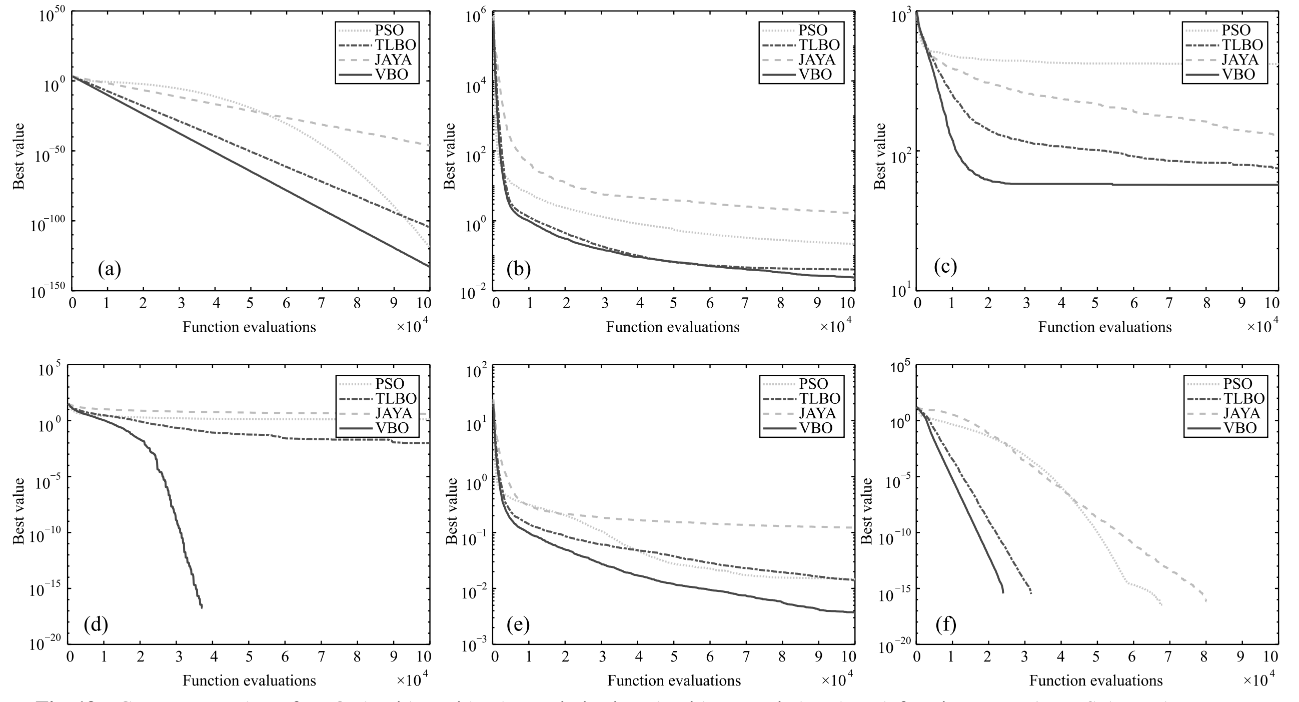


Fig. 19 Convergence plots of VBO algorithm with other optimization algorithms on six benchmark functions (D = 5). (a) Sphere; (b) Rosenbrock; (c) Schwefel; (d) Rastrigin; (e) Griewank; (f) Ackley

$$g_4(\vec{x}) = -8x_1 + x_{10} \leqslant 0,$$
$$g_5(\vec{x}) = -8x_2 + x_{11} \leqslant 0,$$
$$g_6(\vec{x}) = -8x_3 + x_{12} \leqslant 0,$$
$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leqslant 0,$$
$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leqslant 0,$$
$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leqslant 0. \tag{21}$$

The search space are $0 \leqslant x_i \leqslant 1$, where $i = 1, 2, 3, \ldots, 9$ and $0 \leqslant x_i \leqslant 100$, where $i = 11, 12, 13$. The global minima is at $\vec{x^*} = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 3\ 3\ 3\ 1)$ where six constraints are active (g_1, g_2, g_3, g_7, g_8, and g_9) and $f(\vec{x^*}) = -15$.

5.2.2 Constrained benchmark function 2

This test function is also minimization problem which is polynomial in nature. It has one non-linear equality constraints and ten design variables. The ratio between the feasible reason and search space is about 0.0000%.

$$\textbf{Minimize } f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i, \tag{22}$$

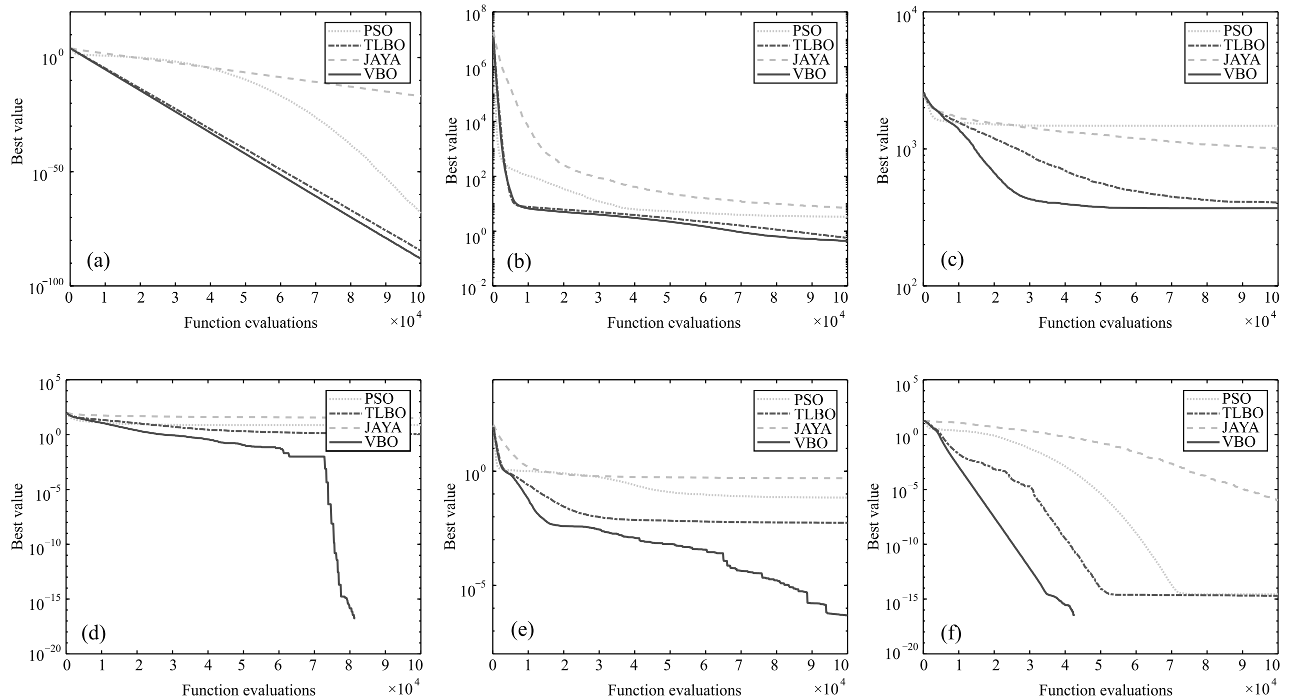


Fig. 20 Convergence plots of VBO algorithm with other optimization algorithms on six benchmark functions (D = 10). (a) Sphere; (b) Rosenbrock; (c) Schwefel; (d) Rastrigin; (e) Griewank; (f) Ackley

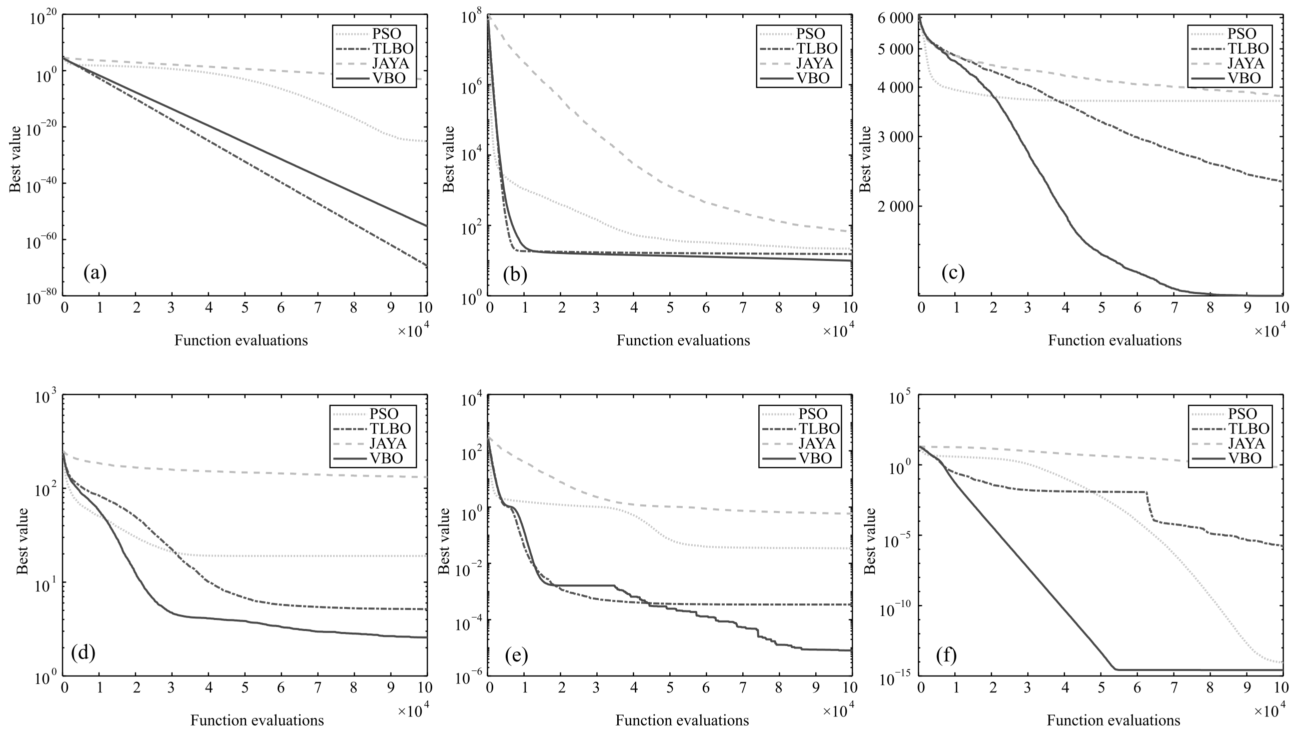


Fig. 21 Convergence plots of VBO algorithm with other optimization algorithms on six benchmark functions (D = 20). (a) Sphere; (b) Rosenbrock; (c) Schwefel; (d) Rastrigin; (e) Griewank; (f) Ackley

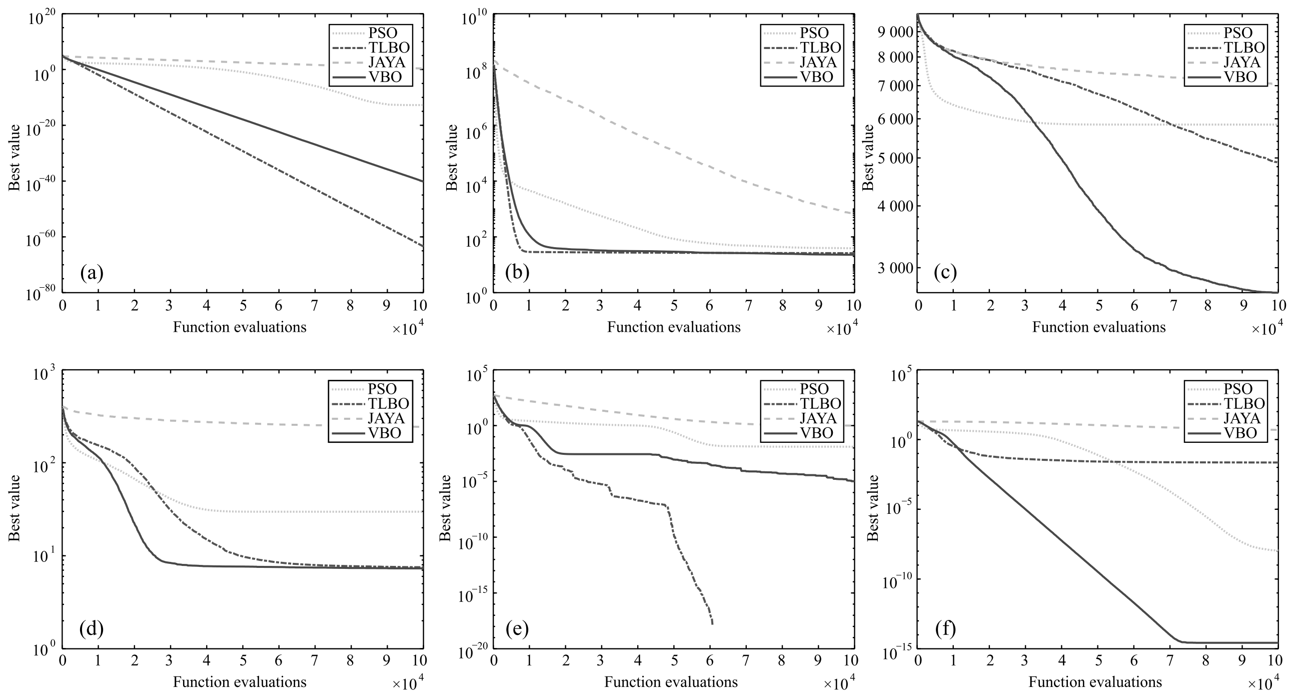


Fig. 22 Convergence plots of VBO algorithm with other optimization algorithms on six benchmark functions (D = 30). (a) Sphere; (b) Rosenbrock; (c) Schwefel; (d) Rastrigin; (e) Griewank; (f) Ackley

Table 19 Comparison of mean and standard deviation of PSO, TLBO, Jaya, and VBO for six constrained benchmark functions

Function		PSO	TLBO	Jaya	VBO
	Best	–1.500000E+01	–1.500000E+01	–1.500000E+01	–1.500000E+01
	Worst	–7.828127E+00	–9.453131E+00	–6.050375E+00	–9.453129E+00
function 1	Mean	–1.113469E+01	–1.366360E+01	–1.235749E+01	–1.390969E+01
	SD	1.713556E+00	1.348880E+00	2.188258E+00	1.216994E+00
	Median	–1.082813E+01	–1.382813E+01	–1.200001E+01	–1.500000E+01
	Best	–1.000024E+00	–9.567401E-01	–7.357541E-01	–9.909789E-01
	Worst	–9.954412E-01	3.730522E-10	3.949745E-05	0.000000E+00
function 2	Mean	–9.997986E-01	–4.142315E-01	–1.528574E-01	–7.334062E-01
	SD	5.025495E-04	3.339131E-01	2.298001E-01	2.682645E-01
	Median	–9.998931E-01	–3.678652E-01	–2.592824E-04	–8.386836E-01
	Best	–6.968587E+03	–6.968587E+03	–6.968587E+03	–6.968587E+03
	Worst	–6.968587E+03	2.390804E+05	3.436044E+04	–6.968587E+03
function 3	Mean	–6.968587E+03	–2.047607E+03	–3.062072E+03	–6.968587E+03
	SD	4.919685E-12	3.444686E+04	1.134683E+04	1.371257E-11
	Median	–6.968587E+03	–6.968587E+03	–6.968587E+03	–6.968587E+03
	Best	6.806339E+02	6.806314E+02	6.807234E+02	6.806311E+02
	Worst	6.807022E+02	6.806407E+02	6.811249E+02	6.806419E+02
function 4	Mean	6.806528E+02	6.806347E+02	6.808954E+02	6.806356E+02
	SD	1.388113E-02	1.976942E-03	9.008059E-02	2.652274E-03
	Median	6.806492E+02	6.806344E+02	6.808888E+02	6.806351E+02
	Best	7.499975E-01	7.499991E-01	7.500194E-01	7.500061E-01
	Worst	7.501193E-01	7.555725E-01	1.000000E+00	7.534555E-01
function 5	Mean	7.500017E-01	7.506142E-01	7.736391E-01	7.505710E-01
	SD	1.688186E-05	9.199588E-04	5.321350E-02	7.088048E-04
	Median	7.499975E-01	7.502538E-01	7.529423E-01	7.502686E-01
	Best	9.617258E+02	9.617173E+02	9.617730E+02	9.617174E+02
	Worst	5.999889E+06	9.691742E+02	1.658443E+03	9.655521E+02
function 6	Mean	1.209433E+05	9.627041E+02	9.832574E+02	9.621547E+02
	SD	8.398494E+05	1.598102E+00	7.273386E+01	7.298098E-01
	Median	9.642301E+02	9.619606E+02	9.676444E+02	9.618547E+02

subject to:

$$h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0, \tag{23}$$

where $n = 10$ and the search space are $0 \leqslant x_i \leqslant 1$, where $i = 1, 2, 3, \ldots, n$. The global minima is at $\vec{x^*}$ = (0.31624357647283069, 0.316243577414338339, ..., 0.316243576147374916) and $f(\vec{x^*}) = -1.00050010001000$.

5.2.3 Constrained benchmark function 3

This test function is also minimization problem which is also polynomial in nature. It has four non-linear inequality constraints and seven design variables. The ratio between the feasible reason and search space is about 0.5121%.

Minimize

$$f(x) = (x_1 - 10)^2 + (x_2 - 20)^3, \tag{24}$$

subject to:

$$\begin{aligned} g_1(x) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leqslant 0, \\ g_2(x) &= -(x_1 - 6)^2 - (x_2 - 5)^2 - 82.81 \leqslant 0. \end{aligned} \tag{25}$$

The search space is $13 \leqslant x_1 \leqslant 100$. The global minima is at $\vec{x^*}$ = (14.095000, 0.842960) and $f(\vec{x^*}) = -6961.813875$. Both constraints ($g_1$ and g_2) are active.

5.2.4 Constrained benchmark function 4

This test function is also minimization problem which is also polynomial in nature. It has four non-linear inequality constraints and seven design variables. The ratio between the feasible reason and search space is about 0.5121%.

Minimize

$$\begin{aligned} f(x) =& (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ &+ 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - x_7, \end{aligned} \tag{26}$$

subject to:

$$\begin{aligned} g_1(x) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0, \\ g_2(x) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0, \\ g_3(x) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0, \\ g_4(x) &= 4x_1^2 - x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 + 11x_7 \leqslant 0. \end{aligned} \tag{27}$$

The search space is $-10 \leqslant x_1 \leqslant 10$, where $i = 1, 2, \ldots, 7$. The global minima is at $\vec{x^*}$ = (2.330499, ..., 1.594226) and $f(\vec{x^*})$ = 680.630057. Two constraints (g_1 and g_4) are active.

5.2.5 Constrained benchmark function 5

This test function is minimization problem which is quadratic in nature. It has only one nonlinear equality constraints and two design variables. The ratio between the feasible reason and search space is about 0.0000%.

$$\textbf{Minimize } f(\vec{x}) = x_1^2 + (x_2 - 1)^2, \tag{28}$$

subject to:

$$h(\vec{x}) = x_2 - x_1^2 = 0. \tag{29}$$

The search space is $-1 \leqslant x_1, x_2 \leqslant 1$. The global minima is at $\vec{x^*}$ = (–0.707036070037170616, 0.500000004333606807) and $f(\vec{x^*}) = 0.7499$.

5.2.6 Constrained benchmark function 6

This test function is minimization problem which is quadratic in nature. It has one linear and one nonlinear equality constraints and three design variables. The ratio between the feasible reason and search space is about 0.0000%.

$$\textbf{Minimize } f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3, \tag{30}$$

subject to:

$$\begin{aligned} h_1(\vec{x}) &= x_1^2 + x_2^2 + x_3^2 - 25 = 0, \\ h_2(\vec{x}) &= 8x_1 + 14x_2 + 7x_3 - 56 = 0. \end{aligned} \tag{31}$$

The search space is $-1 \leqslant x_i \leqslant 10, i = 1, 2, 3$. The global minima is at $\vec{x^*}$ = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921) and $f(\vec{x^*}) = 61.715022289961$.

6 Related work

This section provides an overview of CPP. First, related work of controller placement problem is classified into two categories; un-capacitated and capacitated. Generally, un-capacitated means controllers having an infinite capacity whereas capacited means different controllers having the same or different capacity. Next, work related to clustering based solutions are discussed. Finally, different optimization algorithms are discussed that are used to solve this problem.

6.1 Un-capacitated and capacitated controller placement problem

Papers [8,44–47] have not considered the capacity of the controllers and load on switches. This kind of problem can be restricted to the static environment because the burden on the controllers cannot be distributed to other controllers which are less loaded. Static traffic means fixed traffic. Here, traffic is either propagation latency or the number of hops or a number of packets transferred between switches and controllers.

Authors of [8] present the CPP as to find the set of k controllers and their respective placements so that the average latency between switches and controllers ($\pi^{avgS2Clatency}$(P)) is minimized. The average latency between switches and controllers is given in Eq. (1). This optimization problem is known as the minimum *k*-median problem. The *k*-median problem is an extended version of the *k*-mean problem. The *k*-median problem divides networks into *k*-clusters, and each cluster finds median instead of mean.

Authors of [8] also propose to solve the CPP by minimizing the maximum latency between switches and controllers ($\pi^{maxS2Clatency}$(P)). This maximum latency is given in Eq. (2) and the minimization consists of solving the minimum *k*-center problem, being *k* again the number of controllers. The *k*-center problem is a location analysis problem related to the optimization problem in the area of operation research. Heller et al. [8] concentrates only on the static environment and ignores the load on the controllers, inter-controller latency, and failure scenarios.

The capacity of controllers and load on switches are considered during the placement of controllers. There is a chance of controller failure if load and capacity are not taken into account. In other words, we can say that some controllers may be overloaded and some controllers have a negligible load. Thus, load balancing is required to distribute the load amongst the controllers. The capacity of the controller means the number of *Packet_In* messages processed by the controller per second. Most popular NOX controller can handle around 30k flow initiations per second [11]. Most of the researchers have considered capacity of controller and load on the switch while placing the controllers in SDN.

A mathematical model for optimal controller placement is proposed by Sallahi and St-Hilaire [37]. Controllers and links can be activated or deactivated to enhance the performance of the network. They find an optimal number of controllers, locations where controllers should be placed and the kind of controllers. This model tries to minimize the cost of the controllers. This is the first mathematical model for the SDN controller placement. They use CPLEX Optimizer 12.5 [48] to find all optimized results for controller placement in SDN. This model is valid only for small area networks (1km × 1km) in SDN. Authors extended their work by considering expansion problem [38]. Expansion problem can be defined as the expansion of current SDN infrastructure such that investment cost is minimized.

6.2 Network clustering based controller placement problem

In 2013, Bari et al. [19] introduced dynamic controller provisioning problem (DCPP) in SDN for the first time. They deployed multiple controllers that work simultaneously to manage and control the SDN infrastructure. The objective of [19] is to minimize the communication overhead as well as flow setup time, but the controller load imbalance, switch-to-controller propagation latency, and controller-to-controller latency metrics are ignored. They do not explain how to find which switches are reassigned to the controller(s).

LiDy (location independent dynamic flow management) [49] provides best algorithms to find the optimum controllers and locations as well as minimize the switch-to-controller latency. They discuss about controller utilization. Authors extends their work as *LiDy+* in [50] that gives a better result as compared to *LiDy* and runs in $O(n^2)$ where the previous solution takes $O(n^2 logn)$ time complexity. *LiDy+* does not only provide the minimum number controllers and maximum controller utilization but also takes less energy consumption and maintenance cost as compared to *LiDy*. In general, the researchers deployed the controllers only to switch locations. However, authors of [49, 50] provide their solution in which the deployment locations of controller are not limited to switch locations.

Spectral clustering based controller placement (SCCP) algorithm [51] divides the large-sized network into different small networks and then find the optimum number of controllers. Min-max cut function is used for partitioning the large-sized network into SDN domains, and each SDN domain has its controller. Here, the question arises as to where the controller should be placed in SDN domain, to improve the performance of the SDN. Reliability and efficiency of the controller can be enhanced by reducing the size of the network. Authors extended their previous work in [23] to ensuring security, reliability, management and so on. An objective is to minimize switch-to-controller latency. Additionally, authors focused on balanced partitions of the large-sized network.

Density based controller placement (DBCP) [30] partitions the large-scale network into the small sized network domains. The authors discuss CPP with and without capacity and provide solutions for each of them. In most of the research works, a known number of controllers are assumed for large-sized networks, because it is not easy to determine the minimum number of controllers. Without traversing all possible locations, we cannot find this number, which is not feasible for large-sized networks. In their experiments, authors considered unit weight for all the edges.

6.3 Optimization based controller placement problem

PSO (particle swarm optimization) based work [13] have focused on the capacity of the controller as well as latency among the controllers. The authors try to minimize the global latency of the SDN-based network as well inter-controller latency. But they ignored the failure scenario for the controllers in CPP. Lantz et al. [52] eliminated the limitations of [13,51]. In [52], average case switch-to-controller latency, worst case switch-to-controller latency, inter-controller latency, and load imbalance have been discussed.

7 Conclusion and future work

Placement of controller(s) is an important aspect in the large-sized SDN. Efficient controller placement tries to minimize the total average latency of SDN to maximizing the performance of SDN. This paper discusses and analyses the clustering based solutions and optimization based solution for controller placement. Our experimental results show that optimization based solutions give better results as compared to clustering based solutions.

Next, we propose an optimization based algorithm, named as a Varna-based optimization (VBO), which minimizes the total average latency of SDN. We discuss all 12 different possible scenarios for CPP. VBO has several advantages as compared to optimization methods. First, VBO does not consider the same formulation for all the particles in the population. Also, it is not necessary that particles present in a particular class in one generation will always remain in it. So, Varna (class) is decided by particle's *Karma* (fitness value), not by their birth. Second, VBO gives the best optimum result as compared to clustering based results as well as optimizations based results for capacitated controller placement problem. Third, its convergence rate is better as compared to TLBO, PSO, and Jaya algorithms.

VBO can further be improved by classifying particles in more than two classes, where each class has a specific task. This work can be improved by changing the sizes of classes dynamically across generations.

References

1. Fundation O N. Software-defined networking: the new norm for networks. ONF White Paper, 2012, 2: 2–6
2. Fundation O N. SDN Architecture Overview, version 1.0. ONF White Paper, 2013, 1–5
3. Jammal M, Singh T, Shami A, Asal R, Li Y. Software defined networking: state of the art and research challenges. Computer Networks, 2014, 72: 74–98
4. Farhady H, Lee H, Nakao A. Software-defined networking: a survey. Computer Networks, 2015, 81: 79–95
5. Singh A K, Srivastava S. A survey and classification of controller placement problem in SDN. International Journal of Network Management, 2018, 28(3): e2018
6. Hakiri A, Gokhale A, Berthou P, Schmidt D C, Gayraud T. Software defined networking: challenges and research opportunities for future internet. Computer Networks, 2014, 75: 453–471
7. Gong Y, Huang W, Wang W, Lei Y. A survey on software defined networking and its applications. Frontiers of Computer Science, 2015, 9(6): 827–845
8. Heller B, Sherwood R, McKeown N. The controller placement problem. In: Proceedings of the 1st Workshop on Hot Topics in Software Defined networks. 2012, 7–12
9. Ahmed R, Boutaba R. Design considerations for managing wide area software defined networks. IEEE Communications Magazine, 2014, 52(7): 116–123
10. Lange S, Gebert S, Spoerhase J, Rygielski P, Zinner T, Kounev S, Tran-Gia P. Specialized heuristics for the controller placement problem in large scale SDN networks. In: Proceedings of the 27th International Teletraffic Congress. 2015, 210–218
11. Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R. On controller performance in software-defined networks. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud and Enterprise Networks and Services. 2012
12. Schmid S, Suomela J. Exploiting locality in distributed SDN control. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. 2013, 121–126
13. Gao C, Wang H, Zhu F, Zhai L, Yi S. A particle swarm optimization algorithm for controller placement problem in software defined network. In: Proceedings of International Conference on Algorithms and Architectures for Parallel Processing. 2015, 44–54
14. Dixit A, Hao F, Mukherjee S, Lakshman T, Kompella R. Towards an elastic distributed SDN controller. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 7–12
15. Gupta D, Jahan R. Inter-SDN controller communication: using border gateway protocol. White Paper by Tata Consultancy Services (TCS), 2014, 1–16
16. Casado M, Freedman M J, Pettit J, Luo J, McKeown N, Shenker S. Ethane: taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 1–12
17. Ng E, Cai Z, Cox A. Maestro: a system for scalable openflow control. Rice University, Houston, TX, USA, TSEN Maestro-Techn Rep, TR10-08, 2010
18. Sherwood R, Kok-kiong Y. Cbench: an open-flow controller benchmarker, 2010
19. Bari M F, Roy A R, Chowdhury S R, Zhang Q, Zhani M F, Ahmed R, Boutaba R. Dynamic controller provisioning in software defined networks. In: Proceedings of the 9th International Conference on Network and Service Management. 2013, 18–25
20. Cheng T Y, Wang M, Jia X. QoS-guaranteed controller placement in SDN. In: Proceedings of the Global Communications Conference. 2015, 1–6

21. Liu J, Liu J, Xie R. Reliability-based controller placement algorithm in software defined networking. Computer Science and Information Systems, 2016, 13(2): 547–560

22. Yao L, Hong P, Zhang W, Li J, Ni D. Controller placement and flow-based dynamic management problem towards SDN. In: Proceedings of the IEEE International Conference on Communication Workshop. 2015, 363–368

23. Xiao P, Li Z Y, Guo S, Qi H, Qu W Y, Yu H S. A *K* self-adaptive SDN controller placement for wide area networks. Frontiers of Information Technology & Electronic Engineering, 2016, 17: 620–633

24. Cheng G, Chen H, Hu H, Lan J. Dynamic switch migration towards a scalable SDN control plane. International Journal of Communication Systems, 2016, 29(9): 1482–1499

25. Hock D, Hartmann M, Gebert S, Zinner T, Tran-Gia P. POCOPLC: enabling dynamic pareto-optimal resilient controller placement in SDN networks. In: Proceedings of IEEE Conference on Computer Communications Workshops. 2014, 115–116

26. Lange S, Gebert S, Zinner T, Tran-Gia P, Hock D, Jarschel M, Gebert S. Heuristic approaches to the controller placement problem in large scale SDN networks. IEEE Transactions on Network and Service Management, 2015, 12(1): 4–17

27. Perrot N, Reynaud T. Optimal placement of controllers in a resilient SDN architecture. In: Proceedings of the 12th IEEE International Conference on Design of Reliable Communication Networks. 2016, 145–151

28. Hu Y, Luo T, Beaulieu N C, Deng C. The energy-aware controller placement problem in software defined networks. IEEE Communications Letters, 2017, 21(4): 741–744

29. Sherwood R, Gibb G, Yap K K, Appenzeller G, Casado M, McKeown N, Parulkar G. FlowVisor: a network virtualization layer. OpenFlow Switch Consortium, Tech. Rep., 2009, 1–14

30. Liao J, Sun H, Wang J, Qi Q, Li K, Li T. Density cluster based approach for controller placement problem in large-scale software defined networkings. Computer Networks, 2017, 112: 24–35

31. Yao G, Bi J, Li Y, Guo L. On the capacitated controller placement problem in software defined networks. IEEE Communications Letters, 2014, 18(8): 1339–1342

32. Knight S, Nguyen H X, Falkner N, Bowden R, Roughan M. The internet topology zoo. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765–1775

33. Hock D, Hartmann M, Gebert S, Jarschel M, Zinner T, Tran-Gia P. Pareto-optimal resilient controller placement in SDN-based core networks. In: Proceedings of the 25th IEEE International Teletraffic Congress. 2013, 1–9

34. Jalili A, Ahmadi V, Keshtgari M, Kazemi M. Controller placement in software-defined WAN using multi objective genetic algorithm. In: Proceedings of the 2nd IEEE International Conference on Knowledge-Based Engineering and Innovation. 2015, 656–662

35. Liu S, Wang H, Yi S, Zhu F. NCPSO: a solution of the controller placement problem in software defined networks. In: Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing. 2015, 213–225

36. Hock D, Gebert S, Hartmann M, Zinner T, Tran-Gia P. POCO framework for Pareto-optimal resilient controller placement in SDN based core networks. In: Proceedings of 2014 IEEE Network Operations and Management Symposium. 2014, 1–2

37. Sallahi A, St-Hilaire M. Optimal model for the controller placement problem in software defined networks. IEEE Communications Letters, 2015, 19(1): 30–33

38. Sallahi A, St-Hilaire M. Expansion model for the controller placement problem in software defined networks. IEEE Communications Letters, 2017, 21(2): 274–277

39. Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the 6th IEEE International Symposium on Micro Machine and Human Science. 1995, 39–43

40. Rao R V, Savsani V J, Vakharia D. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-Aided Design, 2011, 43(3): 303–315

41. Rao R V. Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 2016, 7(1): 19–34

42. Akay B, Karaboga D. Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of Intelligent Manufacturing, 2012, 23(4): 1001–1014

43. Liang J, Runarsson T P, Mezura-Montes E, Clerc M, Suganthan P, Coello C C, Deb K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics, 2006, 41(8): 8–31

44. Hu Y, Wendong W, Gong X, Que X, Shiduan C. Reliability-aware controller placement for software-defined networks. In: Proceedings of IFIP/IEEE International Symposium on Integrated Network Management. 2013, 672–675

45. Hu Y, Wang W, Gong X, Que X, Cheng S. On reliability-optimized controller placement for software-defined networks. China Communications, 2014, 11(2): 38–54

46. Hu Y N, Wang W D, Gong X Y, Que X R, Cheng S D. On the placement of controllers in software-defined networks. The Journal of China Universities of Posts and Telecommunications, 2012, 19: 92–171

47. Guo M, Bhattacharya P. Controller placement for improving resilience of software-defined networks. In: Proceedings of the 4th IEEE International Conference on Networking and Distributed Computing. 2013, 23–27

48. ILOG I. CPLEX optimizer. 2012

49. Ul Huque M T I, Jourjon G, Gramoli V. Revisiting the controller placement problem. In: Proceedings of the 40th IEEE Conference on Local Computer Networks. 2015, 450–453

50. Ul Huque M T I, Si W, Jourjon G, Gramoli V. Large-scale dynamic controller placement. IEEE Transactions on Network and Service Management, 2017, 14(1): 63–76

51. Xiao P, Qu W, Qi H, Li Z, Xu Y. The SDN controller placement problem for WAN. In: Proceedings of 2014 IEEE/CIC International Conference on Communications in China. 2014, 220–224

52. Lantz B, Heller B, McKeown N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM-SIGCOMM Workshop on Hot Topics in Networks. 2010, 1–6

Ashutosh Kumar Singh obtained his BTech degree in Information Technology from Uttar Pradesh Technical University Lucknow, India in 2011 and MTech degree in Computer Science and Engineering from Indian Institute of Information Technology and Management Gwalior, India in 2014. Now he is currently a PhD student in the Department of CSE, Motilal Nehru National Institute of Technology Allahabad, India. He has a membership of IEEE and ACM. His research interest includes network optimization and software defined networking.

Saurabh Maurya is a software engineer in a private company at Gurgaon, India. He received his MTech degree in software engineering from Motilal Nehru National Institute of Technology Allahabad, India in 2017. His current research interests include machine learning, data analytics, optimization, and mathematical modeling.

Shashank Srivastava obtained his PhD degree at Indian Institute of Information Technology Allahabad, India in 2014. He is currently working as an assistant professor in the Department of CSE, Motilal Nehru National Institute of Technology Allahabad, India. He possesses an experience of more than six years in the field of teaching and research. He has published various research papers in the area of network and security. At present he is supervising five PhD students in the field of software defined networking (SDN), named data networking (NDN), network flow optimization and security. He is having the Membership of IEEE, ACM, CSI, and CRSI. His areas of expertise are SDN, NDN, information security, and future internet technologies.