Computer Communications 191 (2022) 133–144

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

# A cross entropy based approach to minimum propagation latency for controller placement in Software Defined Network

Jue Chen, Yu-Jie Xiong *, Xihe Qiu, Dun He, Hanmin Yin, Changwei Xiao

*School of Electronic and Electric Engineering, Shanghai University of Engineering Science, China*

ARTICLE INFO

*Keywords:*
SDN
Controller Placement Problem
Propagation latency
Cross entropy

ABSTRACT

In recent years, the Software Defined Network (SDN) has emerged as a pivotal element not only in data-centers and wide-area networks, but also in next generation networking architectures. SDN is characterized by decoupled data and control planes with logically centralized architecture. In order to span across the networks and avoid single point of failure, one major challenge in the SDN is to select appropriate locations for controllers to shorten the latency between controllers and switches, especially in wide area networks. For this purpose, we formulate the Controller Placement Problem (CPP) as an integer programming problem, which takes both the communication cost and synchronization cost into account. Due to its high computational complexity, the cross entropy belonging to the field of Stochastic Optimization is proposed and can sample the problem space and approximate the distribution of good solutions. As a result, we propose a cross entropy based approach to solve CPP, and we conduct experiments on 6 real topologies from the Internet Topology Zoo and Internet2 OS3E. The results verify that the proposed approach can realize the minimum propagation latency for different network scales with different number of controllers, with a less than 5.30% margin from the optimal solution. Moreover, the cross entropy can promise the calculation result be stable with a less than 2% margin, and can apply to all the network scales including large network topologies.

## 1. Introduction

Recently, Software Defined Network (SDN) has attracted a lot of attention in the field of networking, especially for data center networks and the Wide Area Network (WAN) [1]. Moreover, SDN has gained increasing importance in the context of several next generation networking technologies such as 5G [2], edge computing [3], Internet of Things (IoT) [4], mobile/wireless networks [5], Network Function Virtualization (NFV) [6], optical networks [7], sensor networks [8] and Vehicular Ad-hoc NETworks (VANET) [9].

The prevalence of SDN mainly owes to its centralized architecture, which separates the control plane from the data plane. With the expansion of the SDN, it is hard for a single controller to meet the extensive management needs [10]. In order to improve the scalability and reliability of the network and avoid single point of failure [11], logically centralized and physically distributed multi-controller network architecture emerges. The Controller Placement Problem (CPP) is proposed to quantify the performance concerns of the control plane, which primarily considers three issues: (1) the number of controllers, (2) the position of controllers, and (3) the allocation between switches and controllers.

In this paper, the subset of CPP is narrowed down to decrease network latency in WAN. Numerical metrics have been proposed in related literatures, and the latency is especially critical among those metrics because the control logic of the network is decoupled from simplified switches and all functions of the network are carried out through message exchanging between controllers and switches in SDN [12]. Note that the overall network latency includes queuing latency, transmission latency, propagation latency, and processing latency [13]. In WAN, we mainly discuss the propagation latency because of the following considerations.

- When the network is unobstructed, the queuing latency is negligible [1].
- The transmission latency is related to the data packet and port rate, and is usually a fixed value in the case of the same network device [14].
- The processing latency is primarily affected by the processing capability and load of the controller. In WAN, as most controllers will not be exhausted, the processing latency may be insignificant [14].

The propagation latency is determined by the distance between two nodes, and can be analyzed from different levels of the SDN architecture. In intra-domain, there are two types of latency, the worst-case latency between controller and switch should be accepted (which

* Corresponding author.
*E-mail address:* xiong@sues.edu.cn (Y.-J. Xiong).

https://doi.org/10.1016/j.comcom.2022.04.030
Received 10 August 2021; Received in revised form 23 March 2022; Accepted 27 April 2022
Available online 6 May 2022

means that this metric should not exceed a threshold value) and the average latency between controller and switch reflecting overall performance should be as small as possible. In inter-domain, the average latency between controllers should be considered for view consistency. As a result, the latency can be categorized into three aspects: (1) the average latency between switches and controllers (SC-avg latency), (2) the worst-case latency between switches and controllers (SC-worst latency) and (3) the average latency between controllers (CC-avg latency) [14].

Note that SC-avg latency and SC-worst latency are both affected by the switch-controller path length. On one hand, the SC-avg latency is derived from the quotient of the total latency and number of switches, while the total latency equals to the sum of distances from each switch to its nearest controller divided by propagation velocity, which is set to be $2 \times 10^8$ m/s [15] in this paper. The SC-worst latency is related to the maximum node-to-controller distance and can be calculated in the same way. On the other hand, the CC-avg latency can be captured as the sum of distances between each pair of controllers [16], which leads to a waste of network resources as the same message needs to be exchanged between each two controllers. As a result, we calculate the communication paths [17] connecting all controllers through the Prim algorithm as the abstraction of the inter-controller state synchronization cost through adapting the algorithm. Moreover, we define the global latency as the combination of SC-avg latency and CC-avg latency. In a conclusion, we propose a solution over CPP in terms of minimizing the SC-avg latency, SC-worst latency and global latency, respectively.

Given the structure of the topology and the number of controllers, the brute force algorithm can be used to find the optimal solution for placing controllers in terms of minimizing propagation latency, while the cost of computation time must be paid. As a result, the brute force is not suitable for WAN. The Cross-Entropy (CE) algorithm which belongs to the field of Stochastic Optimization [18,19], however, is considered as a probabilistic optimization algorithm. The cross entropy algorithm is to sample the problem space and approximate the distribution of good solutions. This is achieved by assuming a distribution of the problem space, sampling the problem domain by generating candidate solutions using the distribution, and updating the distribution based on the better candidate solutions discovered. As the algorithm progresses, the distribution becomes more refined until it focuses on the area or scope of optimal solutions in the domain [20].

In this paper, we propose a cross entropy based approach to solve the CPP in WAN to minimize the propagation latency. The main contributions and novelties of our proposed approach are summarized as follows.

- High Precision: our proposed approach can always promise a better performance when compared with other heuristic approaches. More specifically, our proposed approach can decrease the SC-worst latency by up to 80.3% and 75.8% when compared with the Pareto Simulated Annealing and Adaptive Bacterial Foraging Optimization, respectively, and can decrease the SC-avg latency by up to 12.18% and 36.66% when compared with K-means and K-means++, respectively. Moreover, our proposed approach can find out the optimal or near optimal solution with a less than 5.30% margin, for placing controllers for different network scales with different number of controllers.
- High stability: our proposed approach can promise that the calculation result is stable under the same experimental condition when compared with other heuristic approaches. More specifically, the maximum value and minimum value are almost coincident with a less than 2.27% margin when our proposed approach is applied.
- High scalability: our proposed approach can apply to all the network scales, including large network topologies composed of 200 nodes, with a less than 5.30% margin from the optimal solution.

The rest of this paper is organized as follows. Section 2 concludes related works on CPP of SDN, especially on optimizing the propagation latency and Section 3 analyzes the CPP mathematically. Our main contributions are described in the following three sections. The brute force, K-means based approaches and the Prim algorithm to calculate CC-avg latency are illustrated in Section 4, and the our proposed approach to optimize the communication cost is illustrated in Section 5. The corresponding experimental results which demonstrate the performance of our proposed approach are described in Section 6. Finally, concluding remarks are given in Section 7.

## 2. Related works

### 2.1. Controller placement problem

The CPP is firstly introduced and formulated by Heller et al. [21], and has seen a rich body of work since then, which highlights its significance, breadth and scope. The reason why it attracts a lot of attention is because the locations of controllers directly affect the control latency experienced by the SDN switches, which in turn affects a wide range of network issues such as routing policy updates, load balancing, scalability, energy efficiency, resiliency, fault management, quality-of-service (QoS), etc [1]. In general, the CPP optimizes the count, location of controllers to be placed, and the set of switches assigned to each controller.

A general formulation of the CPP is illustrated as follows. The SDN network is often modeled as a graph $G = (V, E, C)$, where $V$ represents the set of switches, $E$ is the set of physical links among switches or controllers, and $C$ denotes the set of controllers. Specifically, $n = |V|$ represents the number of nodes and $k = |C|$ refers to the number of controllers. The studies on the controller placement problem generally exploit approaches to solve two questions: (1) the value of $k$; (2) $C \rightarrow V$ mapping relation, so that a predefined objective function is optimized with multiple constraints [14].

Various strategies have been investigated to obtain optimal or heuristic CPP solutions, and different literatures focus on optimizing different objectives of CPP. In general, the state-of-the-art of CPP can be divided into four sections from the view of optimized objective: (1) latency, (2) reliability, (3) cost and (4) multi-objective. The latency can be classified into 4 subsections: (1) SC-avg latency, (2) SC-worst latency, (3) CC-avg latency and (4) processing latency. The reliability is divided into 3 subsections: (1) multiple control-paths, (2) multiple controllers, and (3) shortest control path. The cost is split into 2 subsections: (1) deployment cost, and (2) energy consumption [14].

The reliability of SDN is directly affected by the reliability of control paths since the management and control SDN messages are transmitted through control paths. A complete control path consists of nodes, links, and controllers. For controller failure, switches have to be connected to multiple different controllers to avoid single point of failure. And for node failures and link failures, two ways can be adopted to solve the problem, either by connecting switches to a controller over two disjoint paths, or by reducing the length of the control path which consists of fewer network elements [14].

The cost of SDN mainly includes the previous network construction and the later operation and maintenance costs. On one hand, deployment cost refers to the cost of network devices (controllers and switches) and their operational expenses, including installing the controller into the network, assigning the controller to the switch, and linking these controllers, and different literatures focus on different aspects. On the other hand, the energy consumption can be saved through putting links and controllers into a sleep mode when the network is idle [14].

The multi-objective problem focuses on optimizing multiple performance metrics at the same time. However, multiple metrics cannot achieve the best solution synchronously, such as conflicts between energy savings and network performance, the difficulty is how to make

a reasonable trade-off between multiple performance metrics. Multi-criteria decision algorithms [22], multi-start hybrid non-dominated sorting genetic algorithm [23] and adaptive bacterial foraging optimization [24] are proposed to solve the CPP with multi-objective [14].

### 2.2. Controller placement problem on optimizing propagation latency

As mentioned before, the propagation latency can be categorized into three aspects: (1) SC-avg latency, (2) SC-worst latency, (3) CC-avg latency. SC-avg latency represents the average value of the packet transmission latency between the switch and the controller, reflecting the basic performance of propagation latency in SDN [14]. In WAN, as the large number of switches, it cannot be solved by exhaustive way or even by optimization solvers in a limited time, so heuristic algorithms [25,26] have been proposed to reduce the search space and ensure rapid convergence to a near optimal placement.

SC-worst latency denotes the maximum value of the packet transmission latency between the switch and the controller, which is usually an optimal objective in a high-performance environment or a strict constraint [14]. In WAN, heuristic algorithms, such as particle swarm optimization algorithm [27] and firefly algorithm [28], are proposed to speed up in limited resource and time constraints; Sahoo et al. [29] divides network into several subdomains to reduce the search space and adopt meta heuristic technique to solve.

CC-avg latency is crucial to achieve a consistent view of the network state, which is required for proper operation of the network application. The observation and investigation confirm that the communication overhead caused by maintaining the shared state among the controllers is very significant, especially for WAN [14]. Ishigaki et al. [30] considers CPP as a weighted minimum set coverage problem and proposes greedy algorithm to solve it, but ignores the limitation of bandwidth, which is solved in [31].

In a conclusion, heuristic algorithms are proposed to reduce the search space and speed up the convergence for WAN, while may fall into a local optimal solution [14]. In order to promise a high precision, stability and scalability simultaneously, this paper proposes a cross entropy based approach to solve CPP.

## 3. Problem formulation

### 3.1. Problem formulation of decreasing SC-avg latency and SC-worst latency

The placement of controllers in the network improves network management, and the latency distribution changes with respect to the controller positions for a fixed number of controllers. Hence, it is necessary to deploy controllers in proper positions [32]. Among all of the metrics, the SC-avg latency is critical for SDN, because all of the functions in SDN are achieved by frequent message exchanges between controllers and switches [12]. In this subsection, we investigate the latency between controllers and switches and formulate the corresponding CPP firstly.

As mentioned before, in the graph $G = (V, E, C)$, let $d_{ij}$ denote the cost of serving switch $v_i$ by controller $c_j$, which means the shortest distance from $v_i$ to $c_j$. The binary variable $x_{ij}$ indicates whether controller $c_j$ is in charge of switch $v_i$ or not.

$$x_{ij} = \begin{cases} 1 & \text{if controller } c_j \text{ is in charge of switch } v_i, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The binary variable $z_i^j$ indicate whether a controller $j$ is placed in node $i$.

$$z_i^j = \begin{cases} 1 & \text{if controller } j \text{ is placed in node } i, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The CPP formulation of decreasing SC-avg latency can be formulated as:

$$\min \sum_{i \in V} \sum_{j \in C} d_{ij} x_{ij}. \tag{3}$$

subject to:

$$x_{ij} \in \{0, 1\}, \forall i \in V, \forall j \in C. \tag{4}$$

$$z_i^j \in \{0, 1\}, \forall i \in V, \forall j \in C. \tag{5}$$

$$\sum_{j \in C} x_{ij} = 1, \forall i \in V. \tag{6}$$

$$\sum_{j \in C} z_i^j <= 1, \forall i \in V. \tag{7}$$

$$\sum_{i \in V} z_i^j = k, \forall j \in C. \tag{8}$$

$$x_{ij} <= z_i^j, \forall i \in V, \forall j \in C. \tag{9}$$

The formulation is an Integer Linear Programming (ILP) problem. As the number of nodes is fixed, then the formulation (3) is equivalent to the following formulation.

$$\frac{1}{n-k} \sum_{i \in V} \sum_{j \in C} d_{ij} x_{ij}. \tag{10}$$

Constraint (4) and (5) specify $x_{ij}$ and $z_i^j$ are binary variables [16], constraint (6) makes sure that each switch is assigned to exactly one controller, constraint (7) states that at most one controller could be installed in a location, constraint (8) guarantees that $k$ controllers are placed in the network, and constraint (9) means that there is no control path of the node without controller. The formulations (3)–(9) correspond to a variant of facility location problem [33], which has been proven to be NP-hard. Due to its high computational complexity, we should develop algorithms to approximate the solution in WAN.

Similarly, the CPP formulation of decreasing SC-worst latency can be formulated as:

$$\min_{i \in V} \max_{j \in C} d_{ij} x_{ij}. \tag{11}$$

Subject to constraints (4)–(9).

### 3.2. Problem formulation of decreasing CC-avg latency and global latency

Since the network is composed of multiple controllers, and each switch is managed by exactly one controller, each controller has only a partial view of the network. To synthesize all flow statistics, a synchronization should be performed between each pair of controllers to exchange flow information, as the controllers are scaled in terms of horizontal expansion. In [16], the synchronization cost is defined as the sum of hops between each pair of controllers, which can lead to duplicate messages. An ideal abstraction of the inter-controller network synchronization cost is a tree connecting all controllers [17]. On account of lower computational complexity, the Prim algorithm is used to calculate the control paths for the control plane.

The binary variable $u_e$ indicate whether a link is part of the control paths.

$$u_e = \begin{cases} 1 & \text{if edge } e \text{ is part of the control paths,} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Let $r_j$ represent the initial point of the control paths, and $d_e$ represent the weight of link $e$, in other words, the length of the link. As a result, the inter-controller synchronization latency is expressed as the weight of the control paths connecting all controllers.

$$\delta_{sync} = \sum_{e \in E} d_e u_e \tag{13}$$

**Algorithm 1** Prim Algorithm to Calculate CC-Avg Latency

**Require:**
$P$
**Ensure:**
Corresponding CC-avg latency $L_{CC}$
$node\_primary\_list = \emptyset, node\_new\_list = \emptyset$;
$c\_primary\_shortest = -1$;
$controller\_distance = 0$;
add controllers of $P$ into $node\_primary\_list$;
add head of $node\_primary\_list$ into $node\_new\_list$;
remove head from $node\_primary\_list$;
**while** $node\_primary\_list! = \emptyset$ **do**
  $S = +\infty$;
  **for** each controller $c\_new$ of $node\_new\_list$ **do**
    **for** each controller $c\_primary$ of $node\_primary\_list$ **do**
      calculate shortest distance from $node\_new\_list$ to $node\_primary\_list$, and the corresponding node of $node\_primary\_list$ is $c\_primary\_shortest$, and assign it to $S$;
    **end for**
  **end for**
  add $c\_primary\_shortest$ into $node\_new\_list$;
  remove $c\_primary\_shortest$ from $node\_primary\_list$;
  $controller\_distance+ = S$;
**end while**
$L_{CC} = controller\_distance \div propagationvelocity \div k$;
return $L_{CC}$

Similarly, if the objective is to minimize the CC-avg latency, the formulation (13) can be used directly, as the number of controllers is fixed.

$$\min \sum_{e\in E} d_e u_e \tag{14}$$

It can be inferred that the shortest path traversed by each controller to the root controller must be composed of link(s) of the control paths.

$$\delta_{sync} = \sum_{j\in C} d_{jr_j} \tag{15}$$

$$\sum_{j\in C} r_j = 1 \tag{16}$$

Constraint (16) makes sure that only a single root controller exists.

As mentioned before, the global latency is a combination of SC-avg latency and CC-avg latency, and the objective can be formulated as:

$$\min \sum_{i\in V}\sum_{j\in C} d_{ij}x_{ij} + \sum_{e\in E} d_e u_e. \tag{17}$$

## 4. Brute force, K-means based approaches and prim algorithm to calculate CC-avg latency

### 4.1. Brute force and K-means based approaches to decrease propagation latency

The optimal solution for the CPP of decreasing propagation latency can be obtained by exhaustive searching for all possible combinations of the controller placements through the brute force. For each placement, we firstly calculate the shortest distance from each switch to its nearest controller, secondly figure out the sum and the maximum of all the shortest distances, and thirdly deduce the SC-avg latency and SC-worst latency. Repeat the above process to compare all the SC-avg latencies and SC-worst latencies for all the placements, and finally return the minimum and the corresponding controller placement scheme. Note that the integer programming problem can be solved by optimization solvers, we only compare the margin between the optimal solution and the latencies derived by the cross entropy to prove the high precision of our proposed approach.

Moreover, in this paper, we analyze and evaluate our proposed approach with two K-means based approaches in terms of propagation latencies, including the standard K-means and K-means++. K-means is the most well-known partition-based clustering algorithm, which aims to minimize the deviations of points in the same cluster [34]. The standard K-means clustering algorithm includes four main steps: (1) initialize $k$ clusters and allocate one center for each cluster using random sampling; (2) allocate nodes into one of the clusters based on distance; (3) recalculate centroid for each cluster; (4) repeat steps 2 and 3 until there is no change in each cluster.

As all the $k$ centers are generated randomly in the initialization stage, then the results are not stable and the accuracy cannot be promised as influenced by the chosen $k$ initial points deeply. Considering the drawback of the standard K-means, K-means++ [35] proposes a specific way of choosing centers with the following steps: (1) take the first center randomly; (2) choose a new center with a higher probability if it is far away from the first center; (3) repeat the second step until $k$ centers are located; (4) all the remaining steps are the same as the standard K-means. To simplify the algorithm, each time we choose a new center which is the farthest from the previous center(s).

### 4.2. Prim algorithm to calculate CC-avg latency

Since the set of controllers does not involve all the nodes of the topology, it is hard to motivate a spanning tree between controllers, and the problem to be solved is the more difficult Steiner tree problem. In order to simplify the problem, we adopt the main thought of the Prim algorithm to establish communication paths between controllers which is shown in Algorithm 1. More specifically, we greedily and iteratively add the shortest path between controllers for now into the set of control paths, until each pair of controller is connected through the control paths mentioned above. Even though the controller connections may not compose a tree, the duplicate synchronization of controllers can be avoided, and the total length of communication paths can be reduced considerably. On this basis, the brute force and K-means based approaches can import Algorithm 1 to calculate CC-avg latency.

## 5. Cross entropy based approach

### 5.1. Background of cross entropy approach

In order to estimate the probability of rare events, a straightforward approach is to use the uniform sampling, and the result becomes more accurate as the number of samples increases. However, as the probability is small, then a large number of samples is needed to achieve a satisfactory accuracy [36]. unlike the uniform sampling, Rubinstein et al. apply the concept of cross entropy in information entropy to rare event simulation [18]. In general, the cross entropy takes samples from a more important zone with a greater probability, which generates a new probability density distribution. Therefore, the target now becomes to minimize the cross entropy of the importance sampling density and the primary probability density. As the occurrence of the optimal solution to the objective function can be seen as a small probability event, the concept of cross entropy can be applied to the optimization problem [37].

The cross entropy does not require the new generated solution is better than the sample solution of the previous iteration. According to the distribution function, it is possible that a poor solution is generated. However, in the process of iteration, the probability of generating a good solution is increasing, and the probability of generating a bad solution is decreasing. In this sense, the optimization process is not easy to fall into the local optimal solution, and the cross entropy can be seen as a global optimization approach [38].

The traditional approaches to solve optimization problems include the simulated annealing, ant colony and genetic approaches. In [38], the authors solve the 0/1 knapsack problem of different scales by using these approaches. The experimental results show that the convergence time and the obtained solution by using the cross entropy are better than other three approaches under the same experimental conditions. Considering the similarity between 0/1 knapsack problem and CPP, we decide to apply the cross entropy to this paper.

### 5.2. Cross entropy based approach to decrease propagation latency

In general, the algorithm of the cross entropy to decrease propagation latency is introduced in Algorithm 2. In the initialization stage, the parameters containing $\mathbf{p}_0$, $N$, $\rho$ and $tol$ should be selected carefully. $\mathbf{p}_0$ is the initial distribution parameter vector. $N$ and $\rho$ represent the sample number and quantile. In general, the top $(1-\rho)\times N$ samples are used to update the distributive function. $tol$ denotes the terminal condition which decides whether to end iterations or not.

**Algorithm 2** Cross Entropy based Approach to Decrease Propagation Latency

**Require:**
$G=(V,E,C)$
$\mathbf{p}_0$, $N$, $\rho$, $tol$
**Ensure:**
A placement P with corresponding minimum propagation latency
$L_{SC}=+\infty, L_{global}=+\infty$;
$whether\_to\_end\_while = False$;
$probability\_matrix = \{\}$;
$updated\_probability\_list = []$;
**while** $whether\_to\_end\_while == False$ **do**
  execute Algorithm 3 to generate samples;
  execute Algorithm 4 to calculate distances;
  execute Algorithm 5 to update probabilities and judge whether termination condition is satisfied;
**end while**
execute Algorithm 6 to find out minimum latency;

After the initialization stage and before the terminal condition is satisfied, the following three steps are required:

- Execute Algorithm 3 to generate samples. In each eligible sample, the number of controllers equals to the given $k$.
- Execute Algorithm 4 to calculate the physical distance for each feasible placement according to different optimization targets.
- Execute Algorithm 5 to rank all the placements according to distances from small to large, choose better solutions to update probabilities and judge whether the terminal condition is satisfied.

When the terminal condition is satisfied (which is related to $tol$), the boolean variable $whether\_to\_end\_while$ is set to be "true" to end iterations, and the cross entropy converges to only one solution which is the returned controller placement scheme. With different optimization targets, the corresponding physical distance is calculated, and the propagation latency can be further derived.

As shown in Algorithm 3, $N$ samples are generated firstly. Based on whether this is the first iteration of the cross entropy or not, the probability of setting a node to be a controller is assigned according to random distribution or Bernoulli distribution. Secondly, $N$ samples are filtrated, and only the samples in which the number of controllers equal to $k$ are reserved, with the corresponding probability density functions recorded in $location\_matrix$.

As shown in Algorithm 4, the physical distance for each placement is calculated, and the calculation formulas differ as optimization targets differ. When the optimization target is SC-avg latency, for each

**Algorithm 3** Sample Generation

**if** first iteration **then**
  **for** each sample $x$ **do**
    **for** each node $y$ **do**
      $probability\_matrix[x][y] = 0$;
    **end for**
    generate a sample, choose $k$ (of $n$) nodes to be controllers according to Random Distribution;
    **for** each node $y$ which is the controller **do**
      $probability\_matrix[x][y] = 1$;
    **end for**
  **end for**
**else**
  **for** each sample $x$ **do**
    **for** each node $y$ **do**
      generate a sample, assign $probability\_matrix[x][y]$ according to Bernoulli Distribution with $p = updated\_probability\_list[y]$;
    **end for**
  **end for**
**end if**
$location\_matrix = \{\}$;
$matrix\_index = 0$;
**for** each sample $x$ **do**
  count the number $count$ of items that $probability\_matrix[x] == 1$;
  **if** $count == k$ **then**
    $location\_matrix[matrix\_index] = probability\_matrix[x]$;
    $matrix\_index++$;
  **end if**
**end for**
$location\_distance\_list = []$;
$location\_matrix\_index = 0$;

placement, the total distance from switches to controllers needs to be calculated, which is composed of the distance from each switch to its nearest controller. When the optimization target is SC-worst latency, the maximum distance from switches to controllers needs to be calculated. More specifically, we firstly calculate the shortest distance from each switch to its nearest controller, and then find out the maximum value among all the calculated distances. When the optimization target is global latency, the total distance for each placement is composed of two parts, one part is the total distance from switches to controllers, and another part is the total distance between controllers which can be calculated according to Algorithm 1.

As shown in Algorithm 5, firstly all the samples are sorted in order of physical distances from small to large, and only the top $(1-\rho)\times N$ samples are used to update the distributive function. Secondly, the following equation is solved and $updated\_probability\_list$ is calculated.

$$P_{t,j}=\frac{\sum_{k=1}^{N} I_{\{S(X_k)\geq\gamma_t\}}x_j}{\sum_{k=1}^{N} I_{\{S(X_k)\geq\gamma_t\}}}(j=1,2,\ldots,n). \tag{18}$$

If the $k$th sample belongs to the selected samples, then $I_{\{S(X_k)\geq\gamma_t\}}$ (i.e. $denominator$) is set to be 1. If the $k$th sample belongs to the selected samples, and the $j$th node is set to be a controller, then $x_j$ (i.e. $numerator$) is set to be 1. Before the end of this iteration, we need to judge whether the termination condition is satisfied. If $\max(p_i^t - p_i^{t-1}) < tol(i=1,2,\ldots,n)$, then the iterations end, otherwise, return to the next iteration.

As shown in Algorithm 6, the list variable $updated\_probability\_list$ is used to update the probability of setting a node to be a controller, in the range of 0 to 1. When iterations end, all the best controller placements have been recorded in $updated\_probability\_list$, hence we only need to

**Algorithm 4** Distance Calculation

```
if optimization target is SC-avg latency then
  for each x in location_matrix do
    for each switch s do
      calculate the shortest distance
      shortest_distance_this_switch from s to the nearest controller;
      location_distance_list[x]
      += shortest_distance_this_switch;
      x++;
    end for
  end for
else if optimization target is SC-worst latency then
  for each x in location_matrix do
    location_distance_list[x] = 0;
    for each switch s do
      calculate the shortest distance
      shortest_distance_this_switch from s to the nearest controller;
      if location_distance_list[x] <
      shortest_distance_this_switch then
        location_distance_list[x]=
        shortest_distance_this_switch;
      end if
      x++;
    end for
  end for
else if optimization target is global latency then
  for each x in location_matrix do
    execute Algorithm 1 to calculate distances among all the
    controllers D_CC;
    location_distance_list[x] += D_cc;
    for each switch s do
      calculate the shortest distance
      shortest_distance_this_switch from s to the nearest controller;
      location_distance_list[x]
      += shortest_distance_this_switch;
      x++;
    end for
  end for
end if
```

**Algorithm 5** Probability Update and Termination Judgement

```
sorted_location_total_distance_list
= sorted(location_distance_list);
calculate fractile_distance for
sorted_location_total_distance_list;
for each switch x do
  lasttime_probability_list[x]
  = updated_probability_list[x];
end for
for each x in sorted_location_total_distance_list do
  if x <= fractile_distance then
    denominator+ = 1;
  end if
end for
for each switch x do
  for each y in location_distance_list do
    if location_distance_list[y]
    <= fractile_distance
    AND location_matrix[y][x] == 1 then
      numerator+ = 1;
    end if
  end for
  updated_probability_list[x] = numerator/denominator;
end for
for each switch x do
  if |updated_probability_list[x]
  −lasttime_probability_list[x]|
  > max_probability_distance then
    max_probability_distance = difference_value;
  end if
end for
if max_probability_distance < tol then
  whether_to_end_while = True;
end if
```

return this variable and corresponding propagation latency. According to the principle of the cross entropy, for each iteration, all the samples need to be traversed; while for each sample, the shortest distances from each node to all the controllers require to be calculated and compared. We use $i$ and $N$ to represent the numbers of iterations and samples, respectively, then the time complexity is $O(i \cdot N \cdot n \cdot k)$.

## 6. Performance evaluations

### 6.1. Experimental setup

In order to verify the performance of proposed approaches, we realize Python codes to evaluate them in terms of the solution and solving time. Experiments are conducted on a Windows PC with an Intel i5-6500 3.20 GHz processor and 20$G$ RAM. We use the real network topologies to test our algorithms, and most of the real network topologies are obtained from a public repository (the Internet Topology Zoo [39]), which consists of 261 topologies in total. We choose five different topologies with different network scales from 9 to 197. Moreover, the Internet2 OS3E topology [40] is also used to test our algorithms. The information of these six topologies are shown in Table 1.

The simulation includes the following steps. First, the topologies of the Internet Topology Zoo and Internet2 OS3E are referred from *.gml and *.py files, respectively, where we can get the number of nodes, connections between them, and the longitude and latitude data of nodes. Second, the longitude and latitude data are transformed into 2-dimensional coordinate. As a result, the Euclidean distance between any two connected nodes can be derived easily. On that basis, the shortest path distance can be calculated through Dijkstras algorithm [41]. Third, the SC-avg latency, SC-worst latency, CC-avg latency and global latency are calculated according to objectives (3), (11), (15) and (17), respectively. Fourth, the controller placement demonstrations are plotted in MATLAB.

In order to evaluate the performance of the proposed approaches, we compare the cross entropy with four representative solutions in the literature: K-means, K-means++, Adaptive Bacterial Foraging Optimization (ABFO) [24] and Pareto Simulated Annealing (PSA) [25]. Moreover, the experimental results of the cross entropy are further compared with the optimal solution. In order to prove the effectiveness of the cross entropy, we compare the SC-avg latency, SC-worst latency and global latency between the cross entropy and other four approaches and the optimal solution in different network scales with different number of controllers. Before the comparison of these approaches, the discussion on the parameter selection of cross entropy is given, as the distributive function is updated by the sample number and quantile, hence the assignment of these two parameters may influence the performance of the whole approach.

**Algorithm 6** Finding of Minimum Latency

$D_l = 0$;
**if** optimization target is SC-avg latency **then**
  **for** each node $x$ **do**
    **for** each $y$ in $updated\_probability\_list$ **do**
      **if** $updated\_probability\_list[y] == 1$ **then**
        calculate the shortest distance $shortest\_distance\_this\_switch$ from $x$ to the nearest controller;
        $P.append(updated\_probability\_list[y])$;
      **end if**
      $D_l += shortest\_distance\_this\_switch$;
    **end for**
  **end for**
  $L_{SC} = D_l \div propagationvelocity \div (n-k)$;
  return $P, L_{SC}$
**else if** optimization target is SC-worst latency **then**
  **for** each node $x$ **do**
    **for** each $y$ in $updated\_probability\_list$ **do**
      **if** $updated\_probability\_list[y] == 1$ **then**
        calculate the shortest distance $shortest\_distance\_this\_switch$ from $x$ to the nearest controller;
        $P.append(updated\_probability\_list[y])$;
      **end if**
      **if** $D_l < shortest\_distance\_this\_switch$ **then**
        $D_l = shortest\_distance\_this\_switch$;
      **end if**
    **end for**
  **end for**
  $L_{SC} = D_l \div propagationvelocity$;
  return $P, L_{SC}$
**else if** optimization target is global latency **then**
  **for** each node $x$ **do**
    **for** each $y$ in $updated\_probability\_list$ **do**
      **if** $updated\_probability\_list[y] == 1$ **then**
        calculate the shortest distance $shortest\_distance\_this\_switch$ from $x$ to the nearest controller;
        $P.append(updated\_probability\_list[y])$;
      **end if**
      $D_l += shortest\_distance\_this\_switch$;
    **end for**
  **end for**
  execute Algorithm 1 to calculate distances between controllers $D_{CC}$;
  $D_l += D_{CC}$;
  $L_{global} = D_l \div propagationvelocity \div n$;
  return $P, L_{global}$
**end if**

**Table 1**
Topology information.

| Name of topology | Gridnet | Bellcanada | Columbus | GtsCe | Cogentco | OS3E |
|---|---|---|---|---|---|---|
| Number of nodes | 9 | 48 | 70 | 149 | 197 | 34 |

### 6.2. Parameter selection of quantile

As mentioned before, only the selected $(1-\rho)\times N$ samples are used to update the distributive function. Keep the sample number constant, the influence of the quantile on the solution and solving time can be derived. Note $\rho$ is in the range of $[0, 1]$, and should be set to be close to 1 to restrict the number of samples used for update. In this subsection, we choose the Cogentco topology, and the number of required controllers is 3. Besides, the sample number $N$ is set to be 1000, and the values of $\rho$ include 0.80, 0.85, 0.90, 0.95, 0.98, 0.99 and 0.999. For each $\rho$, the cross entropy is executed by 10 times to derive the average value in order to make the simulation results more convinced.

**Table 2**
Effect of quantile on solution and solving time.

| Quantile $\rho$ | 0.80 | 0.85 | 0.90 | 0.95 | 0.98 | 0.99 | 0.999 |
|---|---|---|---|---|---|---|---|
| SC-avg latency (ms) | 8.46 | 7.77 | 7.56 | 7.20 | 7.15 | 7.17 | 7.38 |
| Average solving time (s) | 82.4 | 53.3 | 41.7 | 33 | 20.8 | 18.2 | 8 |

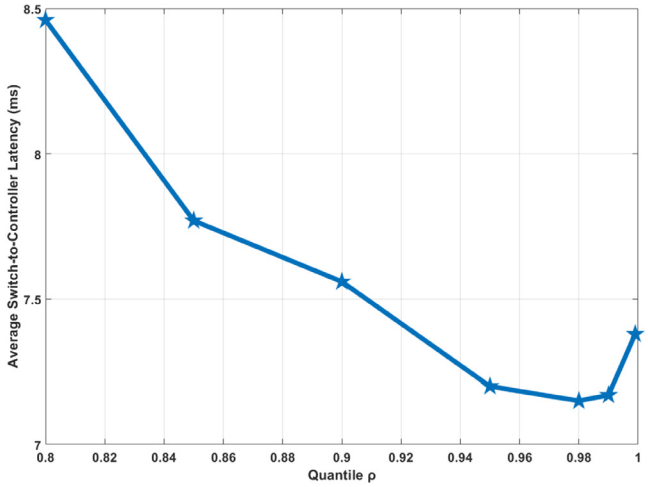


**Fig. 1.** Effect of quantile on solution.

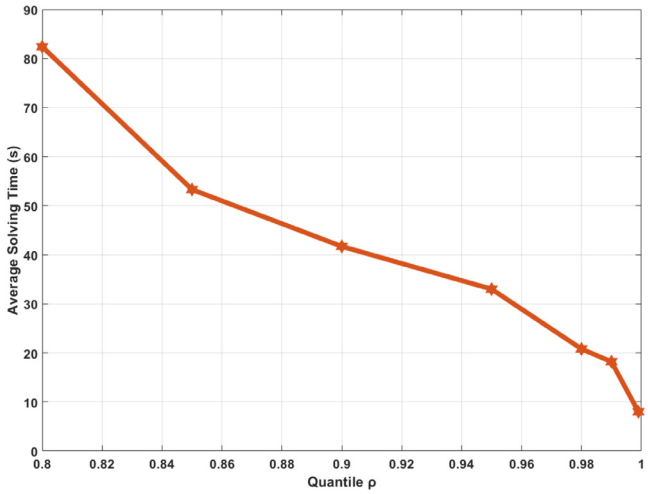


**Fig. 2.** Effect of quantile on solving time.

The simulation results are shown in Table 2, and Figs. 1–2 are derived from the statistics of the table. As shown in Fig. 2, when $\rho$ increases, the average solving time decreases. This is because when the number of samples used for update decreases in each iteration, the convergence is accelerated. However, it should be noticed that when $\rho$ equals to 0.98 or 0.99, the SC-avg latency reaches the minimum (with a less than 1% margin). However, when $\rho = 0.999$, in each iteration only 1 sample is used to update the distributive function, which may result in premature convergence. According to the simulation results, $\rho$ is set to be 0.99, and the number of samples used for update needs to be greater than 10, in order to promise a fast convergence rate and improve the precision of simulation results.

**Table 3**
Effect of sample number on solution and solving time.

| Sample number $N$ | 5000 | 4000 | 3000 | 2000 | 1000 | 500 |
|---|---|---|---|---|---|---|
| SC-avg latency (ms) | 7.07 | 7.08 | 7.10 | 7.11 | 7.16 | 7.22 |
| Average solving time (s) | 97 | 78.5 | 60.3 | 36.3 | 18 | 7.7 |

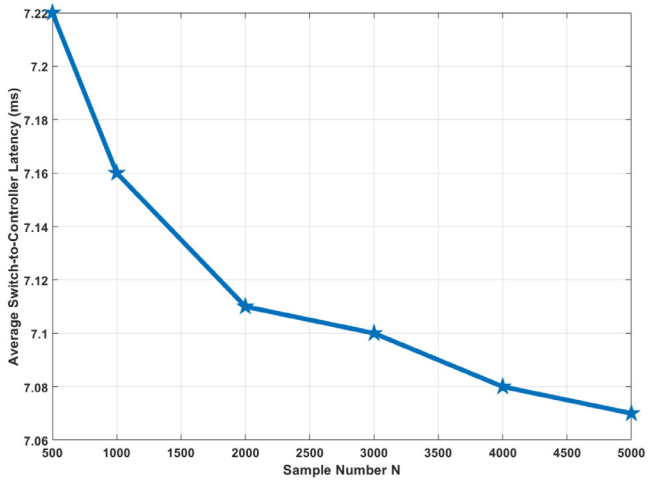


**Fig. 3.** Effect of sample number on solution.

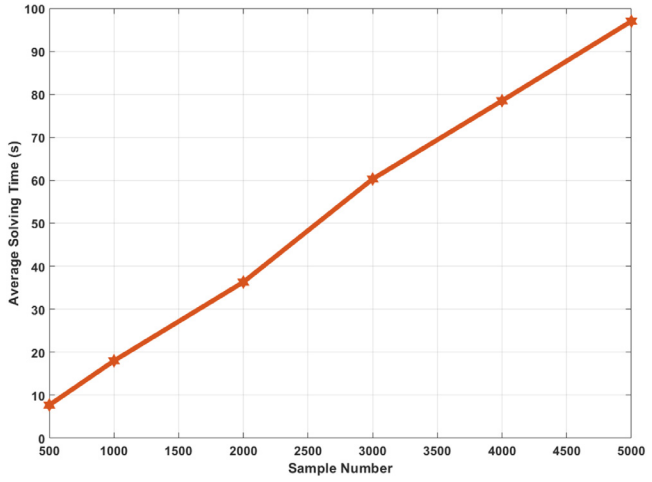


**Fig. 4.** Effect of sample number on solving time.

### 6.3. Parameter selection of sample number

In this subsection, we keep $\rho$ constant, the influence of the sample number on the solution and solving time can be derived. We choose the Cogentco topology and the number of required controllers is 3. Besides, the quantile $\rho$ is set to be 0.99, and the values of $N$ include 500, 1000, 2000, 3000, 4000, and 5000. For each $N$, the cross entropy is executed by 10 times to derive the average value in order to make the simulation results more convinced.

The simulation results are shown in Table 3, and Figs. 3–4 are derived from the statistics of the table. When $N$ increases, the average solving time increases. This is because when the number of samples used for update increases in each iteration, the convergence is decelerated. However, it should be noticed that when the value of $N$ is 3000, 4000 or 5000, the SC-avg latency is almost equal (with a less than 1% margin). As a result, in the following experiments, $\rho$ and $N$ are set to be 0.99 and 3000, respectively, in order to promise a fast convergence rate and improve the precision of simulation results.

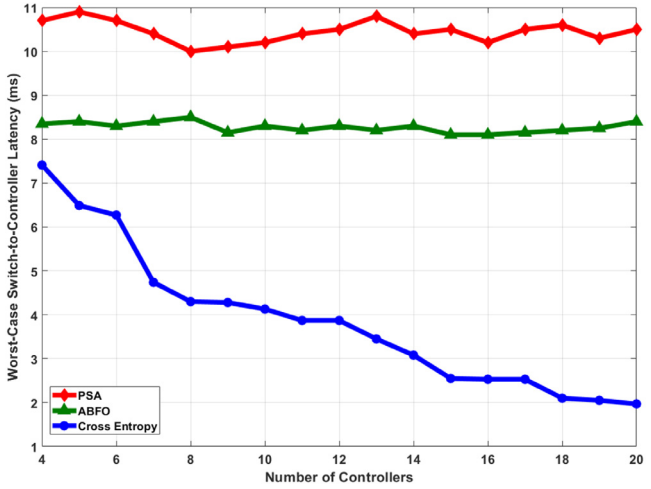


**Fig. 5.** Comparison of cross entropy, abfo and psa on different number of controllers.

### 6.4. Comparison of cross entropy, ABFO and PSA on decreasing SC-worst latency

The Internet2 OS3E topology composed of 34 nodes and 42 edges is usually used to evaluate the network performance of different approaches in CPP. In this subsection, we design a set of experiments to compare the SC-worst latencies of the cross entropy, ABFO and PSA with different number of controllers ranging from 4 to 20.

On one hand, in order to o find the Pareto optimal placements with respect to more than two metrics, a straightforward method is to perform an exhaustive evaluation of all possible placements for a given network topology and desired number of controllers. However, the time and memory requirements increase with the size of the search space, and hence it is not suitable for large scale networks, especially for WAN. On the other hand, simulated annealing [42] is a popular heuristic approach which can decrease the search space and avoid getting stuck in a local optimum.

The ABFO [24] employs adaptive foraging strategies to improve the original Bacterial Foraging Optimization (BFO), which consists of three principal mechanisms, namely, chemotaxis, reproduction or dispersal and elimination. Bacterial chemotaxis keeps bacteria in the places of higher concentrations of nutrients. In the CPP, the heuristic rules corresponding to nutrients contain controllers should prefer to have more connections to other nodes, etc. As a result, ABFO can apply to multi-objective optimization CPP.

The simulation results are shown in Fig. 5. It can be observed from the plot that when the number of controllers is given, the SC-worst latency of the cross entropy is always smaller than other two approaches. More specifically, the SC-worst latencies of PSA and ABFO are in the range of $[10, 11]$ ms and $[8, 9]$ ms, respectively, while the same metric of the cross entropy decreases from 7.41 ms to 1.97 ms. As a result, the differential ratio between the cross entropy and PSA changes from 25.9% to 80.3%, and the differential ratio between the cross entropy and ABFO changes from 7.38% to 75.8%.

Moreover, when the number of controllers increases, the SC-worst latencies of PSA and ABFO almost keep stable. Note that the phenomenon where the latency increases as the number of controllers increases even exists. On the contrary, when the number of controllers increases, the SC-worst latency of the cross entropy decreases or stays unchanged.

### 6.5. Comparison of cross entropy and K-means based approaches on decreasing SC-avg latency

To investigate the precision and stability of the cross entropy on decreasing SC-avg latency, we design two sets of experiments to compare

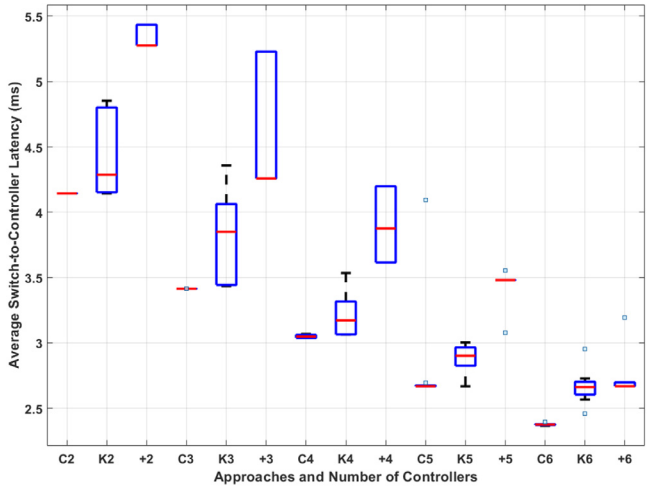


**Fig. 6.** Comparison of SC-avg latency between cross entropy and K-means based approaches on different number of controllers. On the $X$-axis, "C", "K" and "+" represent cross entropy, K-means and K-means++, respectively; and the number means the number of controllers.

**Table 4**
Comparison of cross entropy and K-means based approaches on different number of controllers.

| Number of controllers | 6 | 5 | 4 | 3 | 2 |
|---|---|---|---|---|---|
| Mean value by cross entropy (ms) | 2.38 | 2.81 | 3.05 | 3.41 | 4.14 |
| Mean value by K-means (ms) | 2.67 | 2.88 | 3.21 | 3.81 | 4.44 |
| Mean value by K-means++ (ms) | 2.73 | 3.45 | 3.90 | 4.55 | 5.32 |

the proposed approach, K-means and K-means++, where the influence of the controllers' number and network scales on the solution can be derived, respectively. In the first set of experiments, we choose the Interoute topology and the number of required controllers changes from 2 to 6 incrementally. Given the number of controllers, all the approaches are executed by 10 times to record the SC-avg latency for each time in order to make the simulation results more convinced.

The simulation results are shown in Fig. 6. The figure contains 15 box plots, and each box plot corresponds to one approach with a given number of controllers. In each box plot, 5 numerical points are displayed, which are the maximum value (black whisker above the box), 75th percentile (the top of the box in blue color), 50th percentile (the band inside the box in red color), 25th percentile (the bottom of the box in blue color) and minimum value (black whisker under the box) from top to bottom, respectively. For instance, the box plot named "C6" means that the cross entropy is applied and the number of controllers is 6; and "+5" means that the K-means++ is applied and the number of controller is 5.

It can be observed from three adjacent box plots with the same number of controllers that the cross entropy achieves a smaller SC-avg latency than K-means and K-means++. The calculated mean values of SC-avg latencies by three approaches are shown in Table 4. When the number of controllers changes from 2 to 6, the error rate between cross entropy and K-means is in the range of [2.49%, 12.18%], and the error rate between cross entropy and K-means++ is in the range of [14.71%, 33.43%]. Moreover, it can be inferred that the cross entropy is more stable than K-means and K-means++. More specifically, the maximum value and minimum value are almost coincident with a less than 2.27% margin when the cross entropy is applied.

In the second set of experiments, we set the number of controllers to be 3 to reduce the calculation time, and choose totally five topologies from the Internet Topology Zoo with different network scales (which is shown in Table 1). Other experimental setups are in accordance with the first set of experiments in this subsection.

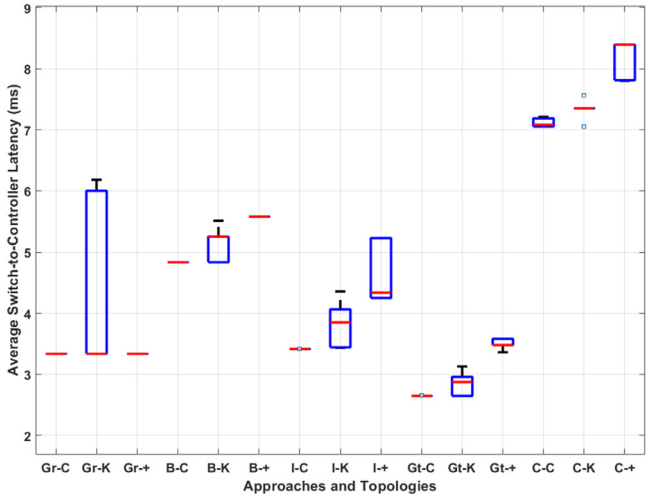


**Fig. 7.** Comparison of SC-avg latency between Cross Entropy and K-means based approaches on different network scales. On the $X$-axis, "C", "K" and "+" represent cross entropy, K-means and K-means++, respectively; and "Gr", "B", "I", "Gt" and "C" represent the topology of "Gridnet", "Bellcanada", "Interoute", "GtsCe" and "Cogentco", respectively.

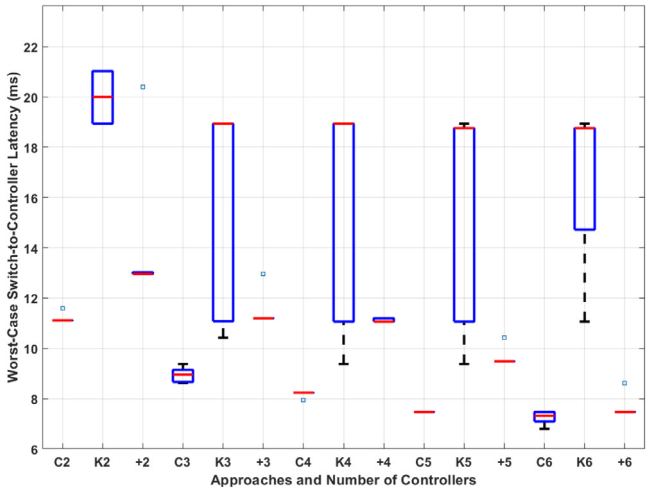


**Fig. 8.** Comparison of SC-worst latency between Cross Entropy and K-means based approaches on different number of controllers. On the $X$-axis, "C", "K" and "+" represent cross entropy, K-means and K-means++, respectively; and the number means the number of controllers.

The simulation results are shown in Fig. 7. It can be observed from three adjacent box plots with the same topology that the cross entropy achieves a smaller SC-avg latency than K-means and K-means++ in the medium-size and large-size networks (i.e. in the topology which is composed of dozens of nodes or more). The calculated mean values of SC-avg latencies by three approaches are shown in Table 5. When the number of nodes increases from 48 to 197, the error rate between cross entropy and K-means is in the range of [2.96%, 11.73%], and the error rate between cross entropy and K-means++ is in the range of [14.93%, 36.66%].

### 6.6. Comparison of cross entropy and K-means based approaches on decreasing SC-worst latency

To further compare the performance of cross entropy, K-means and K-means++, we evaluate the SC-worst latencies of three approaches. we design two sets of experiments, where the influence of the controllers' number and network scales on the solution can be derived, respectively,

**Table 5**
Comparison of cross entropy and K-means based approaches on different network scales.

| Name of topology | Gridnet | Bellcanada | Interoute | GtsCe | Cogentco |
|---|---|---|---|---|---|
| Mean value by cross entropy (ms) | 3.34 | 4.83 | 3.41 | 2.65 | 7.10 |
| Mean value by K-means (ms) | 4.17 | 5.18 | 3.81 | 2.87 | 7.31 |
| Mean value by K-means++ (ms) | 3.34 | 5.58 | 4.66 | 3.49 | 8.16 |

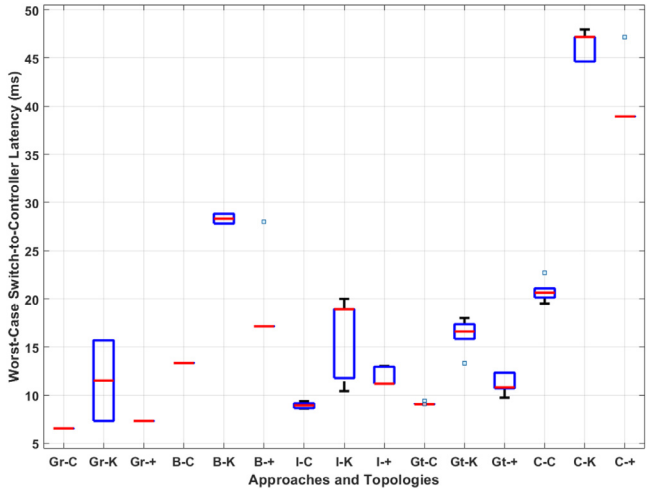


**Fig. 9.** Comparison of SC-worst latency between cross entropy and K-means based approaches on different network scales. On the $X$-axis, "C", "K" and "+" represent cross entropy, K-means and K-means++, respectively; and "Gr", "B", "I", "Gt" and "C" represent the topology of "Gridnet", "Bellcanada", "Interoute", "GtsCe" and "Cogentco", respectively.

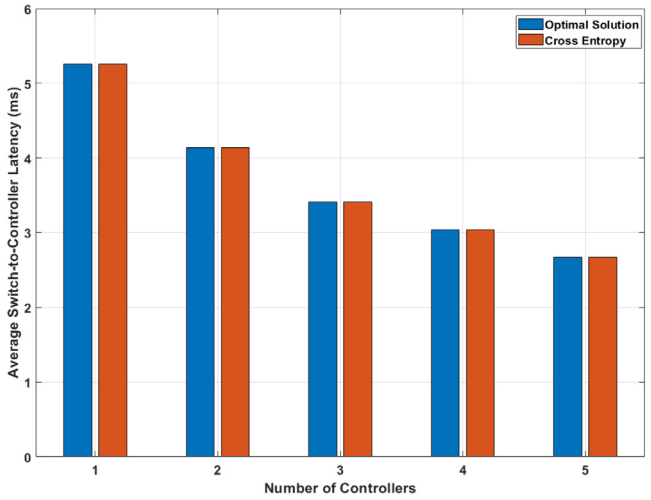


**Fig. 10.** Comparison of SC-Avg latencies between cross entropy and optimal solution on different number of controllers.

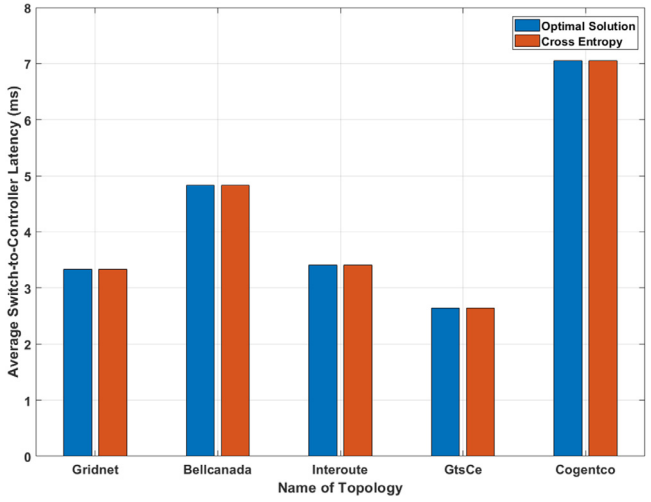


**Fig. 11.** Comparison of SC-Avg latencies between cross entropy and optimal solution on different network scales.

and all the experimental setups are in accordance with Section 6.5. The simulation results are shown in Figs. 8 and 9.

Finally, two conclusions can be derived from these two subsections.

(1) The cross entropy can always achieve a smaller SC-avg latency and SC-worst latency than K-means and K-means++ in the same experimental condition.
(2) The cross entropy can keep the simulation results stable when the experiments are repeated for multiple times, which demonstrates the high stability of our proposed approach.

Moreover, It also can be derived that when the optimization target is the SC-avg latency, K-means performs better than K-means++, while when the optimization target is the SC-worst latency, K-means++ performs better than K-means. It can be explained as K-means initializes the $k$ centers randomly, while K-means++ prefers to choose a node as a new center which is the farthest from the previous centers, and avoid the emergence of extreme points with long distances to reach all the controllers.

### *6.7. Comparison of SC-avg latency between cross entropy and optimal solution*

In this subsection, we evaluate and compare the SC-avg latency between the cross entropy and the optimal solution. To demonstrate the precision and scalability of the cross entropy, we also design two sets of experiments. In the first set of experiments, we choose the Interoute topology and the number of required controllers changes from 1 to 5 incrementally, and all the other procedures are similar with Section 6.5.

The simulation results are shown in Fig. 10, it can be observed from the histogram that when the number of controllers is given, the SC-avg latencies of the cross entropy are always equal to the corresponding optimal solution.

In the second set of experiments, we set the number of controllers to be 3, and choose totally five topologies from the Internet Topology Zoo with different network scales (which is shown in Table 1). All the other procedures are similar with Section 6.5. The simulation results are shown in Fig. 11. It can be observed that in different network topologies, the SC-avg latencies of the cross entropy are always equal to the corresponding optimal solution.

### *6.8. Comparison of SC-worst latency and global latency between cross entropy and optimal solution*

To further demonstrate the advantage of our proposed approach, we evaluate and compare the SC-worst latencies and global latencies between cross entropy and optimal solution, note that the calculation rule of CC-avg latency has been explained in Algorithm 1. we design two sets of experiments, where the influence of the controllers' number

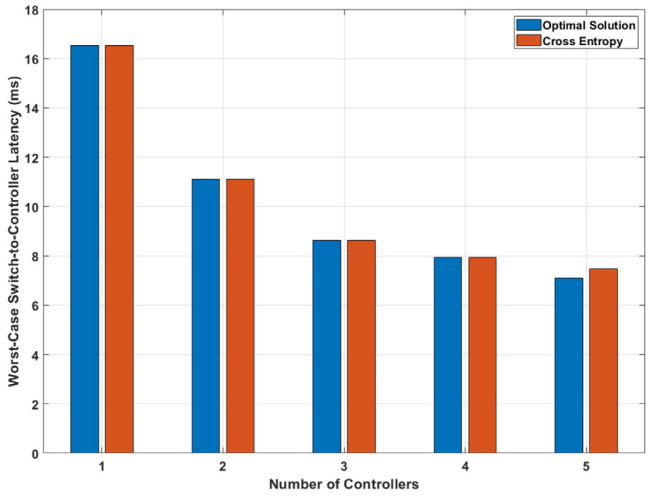


**Fig. 12.** Comparison of SC-worst latencies between cross entropy and optimal solution on different number of controllers.

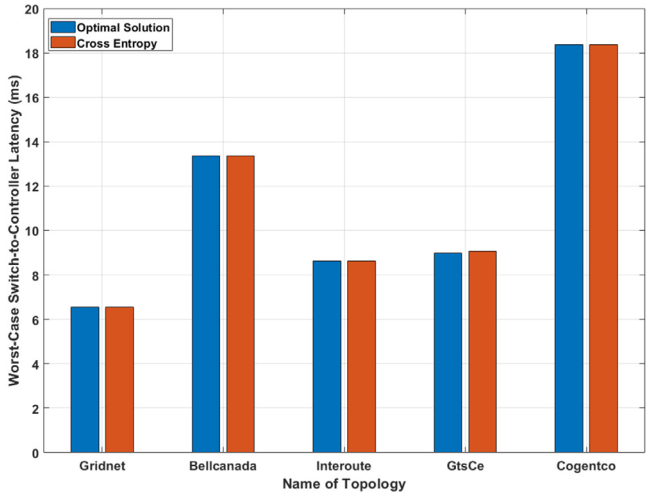


**Fig. 13.** Comparison of SC-worst latencies between cross entropy and optimal solution on different network scales.

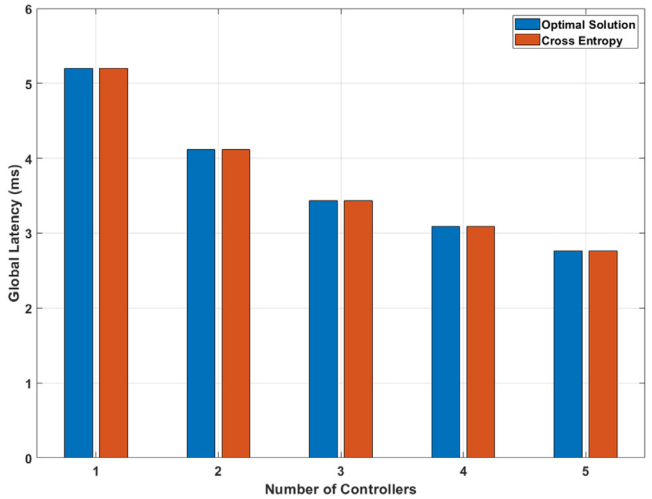


**Fig. 14.** Comparison of global latencies between cross entropy and optimal solution on different number of controllers.

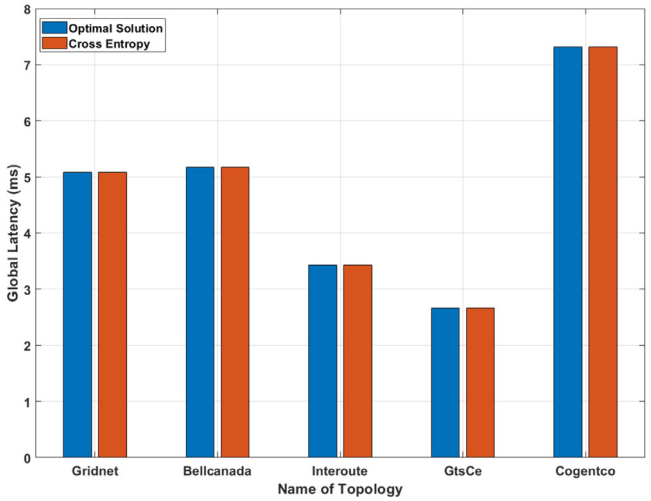


**Fig. 15.** Comparison of global latencies between cross entropy and optimal solution on different network scales.

and network scales on the solution can be derived, respectively, and all the experimental setups are in accordance with Section 6.5.

The simulation results of SC-worst latencies on different number of controllers and different network scales are shown in Figs. 12 and 13, respectively. The simulation results of global latencies on different number of controllers and different network scales are shown in Figs. 14 and 15, respectively. It can be observed from the histogram that in the same experimental condition, the global latencies of the cross entropy are always equal to the corresponding optimal solution, and SC-worst latencies of cross entropy are almost equal to the optimal solution, with the error rate less than 5.30%. In order to decrease the margin from the optimal solution, we can modify the parameters of cross entropy, such as increasing the sample number $N$.

Finally, two conclusions can be derived from these two subsections.

(1) The cross entropy can always find the minimum or near minimum SC-avg latency, SC-worst latency and global latency for different network scales with different number of controllers, with a less than 5.30% margin from the optimal solution, which demonstrates the high precision of our proposed approach.
(2) The cross entropy can apply to all the network scales, including large network topologies composed of 200 nodes, with a less than 5.30% margin from the optimal solution, which demonstrates the high scalability of our proposed approach.

## 7. Conclusion

In this paper, we have proposed new methods to shorten the propagation latency between controllers and their associated switches. First of all, the communication cost between controllers and switches and the synchronization cost between controllers are investigated and formulated qualitatively. Due to the high computational complexity of the integer programming problem, we develop a cross entropy base approach to efficiently obtain the placements. Through the process of iteration, the probability of generating a good solution is always increasing, which makes this approach not be easy to fall into the local optimal solution. In order to evaluate the performance of the proposed approach, extensive simulations are conducted under six real topologies. Simulation results verify that the cross entropy can always find out the optimal or near optimal solution with a less than 5.30% margin, for placing controllers for different network scales with different number of controllers. In a conclusion, the cross entropy can promise a high precision, stability and scalability simultaneously. Future work includes the balance and the Pareto optimality of multiple performance metrics,

such as the SC-avg latency, SC-worst latency and CC-avg latency. For instance, we can compare the two extreme Pareto points: minimize SC-avg latency followed by minimizing CC-avg latency, and vice-versa.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

This work was supported by Science and Technology Commission of Shanghai Municipality (STCSM), China under grant No. 19YF1418300.

## References

[1] T. Das, V. Sridharan, M. Gurusamy, A survey on controller placement in SDN, IEEE Commun. Surv. Tutor. PP (99) (2019) 1.
[2] F.Z. Yousaf, M. Bredel, S. Schaller, F. Schneider, NFV and SDN - Key technology enablers for 5G networks, IEEE J. Sel. Areas Commun. PP (11) (2018) 1.
[3] A.C. Baktir, A. Ozgovde, C. Ersoy, How can edge computing benefit from software-defined networking: A survey, use cases & future directions, IEEE Commun. Surv. Tutor. PP (4) (2017) 1.
[4] N. Bizanis, F. Kuipers, SDN and virtualization solutions for the Internet of Things: A survey, IEEE Access PP (99) (2016) 1.
[5] I.T. Haque, N. Abu-Ghazaleh, Wireless software defined networking: A survey and taxonomy, IEEE Commun. Surv. Tutor. 18 (4) (2016) 1.
[6] Y. Li, M. Chen, Software-defined network function virtualization: A survey, IEEE Access 3 (2017) 2542–2553.
[7] P. Bhaumik, S. Zhang, P. Chowdhury, S. Lee, J.H. Lee, B. Mukherjee, Software-defined optical networks (SDONs): A survey, Photonic Netw. Commun. 28 (1) (2014) 4–18.
[8] H.I. Kobo, A.M. Abu-Mahfouz, G.P. Hancke, A survey on software-defined wireless sensor networks: Challenges and design requirements, IEEE Access 5 (2017) 1872–1899.
[9] A. Mc, A. Sh, B. Kkm, C. Aks, D. Zz, A survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases - ScienceDirect, Sustainable Cities Soc. 35 (2017) 830–840.
[10] Z. Guo, S. Mu, X. Yang, Z. Duan, W. Luo, S. Hui, H.J. Chao, Improving the performance of load balancing in software-defined networks through load variance-based synchronization, Comput. Netw. 68 (aug.5) (2014), 95–109.
[11] Z. Guo, R. Liu, Y. Xu, A. Gushchin, A. Walid, H.J. Chao, STAR: Preventing flow-table overflow in software-defined networks, Comput. Netw. 125 (oct.9) (2017) 15–25.
[12] G. Wang, Y. Zhao, J. Huang, Y. Wu, An effective approach to controller placement in software Defined Wide Area networks, IEEE Trans. Netw. Serv. Manag. 15 (1) (2017) 1.
[13] J.F. Kurose, Computer Networking : A Top-Down Approach Featuring the Internet, 2008.
[14] J. Lu, Z. Zhang, T. Hu, P. Yi, J. Lan, A survey of controller placement problem in software-defined networking, IEEE Access 7 (2019) 1.
[15] G. Wang, Y. Zhao, J. Huang, D. Qiang, J. Li, A K-means-based network partition algorithm for controller placement in software defined network, in: ICC 2016 IEEE International Conference on Communications, 2016.
[16] Z. Su, M. Hamdi, MDCP: Measurement-Aware Distributed Controller Placement for Software Defined Networks, in: IEEE International Conference on Parallel & Distributed Systems, 2015.
[17] T. Das, M. Gurusamy, Multi-objective control plane dimensioning in hybrid SDN/Legacy networks, IEEE Trans. Netw. Serv. Manag. 18 (3) (2021) 2929–2942, http://dx.doi.org/10.1109/TNSM.2021.3066847.
[18] R.Y. Rubinstein, Optimization of computer simulation models with rare events, European J. Oper. Res. 99 (1) (1997) 89–112.
[19] R.Y. Rubinstein, D.P. Kroese, The Cross Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics), 2004.
[20] U. Mahlab, Y. Wolbrum, Z. Erlichson, Traffic load-balancing for software-defined elastic optical networks based cross-entropy technique, in: 2019 21st International Conference on Transparent Optical Networks, ICTON, 2019.
[21] B. Heller, R. Sherwood, N. Mckeown, The controller placement problem, ACM SIGCOMM Comput. Commun. Rev. 42 (4) (2012).
[22] A. Ksentini, M. BaGaa, T. Taleb, I. BaLasingham, On using bargaining game for Optimal Placement of SDN controllers, in: IEEE International Conference on Communications, 2016, pp. 1–6.
[23] V. Ahmadi, M. Khorramizadeh, An adaptive heuristic for multi-objective controller placement in software-defined networks, Comput. Electr. Eng. (2017) 204–228.
[24] B. Zhang, X. Wang, M. Huang, Multi-objective optimization controller placement problem in internet-oriented software defined network, Comput. Commun. 123 (JUN.) (2018) 24–35.
[25] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, M. Hoffmann, Heuristic approaches to the controller placement problem in large scale SDN networks, IEEE Trans. Netw. Serv. Manag. 12 (1) (2017) 4–17.
[26] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, T. Li, Density cluster based approach for controller placement problem in large-scale software defined networkings, Comput. Netw. (2017).
[27] L. Liao, C. Victor, M. Leung, Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs, in: Network Function Virtualization & Software Defined Networks, 2017.
[28] B. Killi, E.A. Reddy, S.V. Rao, Cooperative Game Theory Based Network Partitioning for Controller Placement in SDN, 2018, pp. 105–112.
[29] K.S. Sahoo, A. Sarkar, S. Sahoo, B. Sahoo, R. Dash, On the placement of controllers for designing a wide area software defined networks, in: Tencon IEEE Region 10 Conference, 2017.
[30] G. Ishigaki, R. Gour, A. Yousefpour, N. Shinomiya, J.P. Jue, Cluster leader election problem for distributed controller placement in SDN, in: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017.
[31] S. Liu, R. Steinert, D. Kostic, Flexible distributed control plane deployment, in: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018.
[32] P. Shrinivas, T. Jaisingh, Failure-based controller placement in software defined networks, IEEE Trans. Netw. Serv. Manag. 17 (1) (2019) 503–516.
[33] K. Jain, M. Mahdian, A. Saberi, A new greedy approach for facility location problems, in: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, 2002.
[34] T. Su, J.G. Dy, In search of deterministic methods for initializing K-means and Gaussian mixture clustering, Intell. Data Anal. 11 (4) (2007) 319–338.
[35] D. Arthur, S. Vassilvitskii, K-Means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007, 2007.
[36] J. Bucklew, Introduction to Rare Event Simulation, 2010.
[37] R. Rubinstein, The cross-entropy method for combinatorial and continuous optimization, Methodol. Comput. Appl. Probab. 1 (2) (1999) 127–190.
[38] L.U. Chang-Xian, L.U. Yi-Ping, C. Jian-Zhong, A cross-entropy method for solving 0-1 knapsack problem, Comput. Simul. (2007).
[39] The Internet Topology Zoo, http://www.topology-zoo.org/.
[40] The OS3E Topology, https://github.com/ParanoiaUPC/mininet-OS3E/.
[41] S. Skiena, Dijkstra's algorithm, Encycl. Oper. Res. Manag. Sci. (1990).
[42] C.D. Gelatt, M.P. Vecchi, S. Kirkpatrick, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.