Controller Placement with Respect to Controller Reachability

Xu, Ran ; Wang, Fenghua; Kooij, Robert E.

DOI
10.1109/ICSRS59833.2023.10381264

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 7th International Conference on System Reliability and Safety (ICSRS)

Controller placement with respect to controller reachability

Ran Xu
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands
R.Xu-5@student.tudelft.nl

Fenghua Wang
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands
F.Wang-8@tudelft.nl

Robert E. Kooij
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands
Unit ICT, Strategy and Policy
Netherlands Organisation for Applied Scientific Research (TNO)
Den Haag, The Netherlands
R.E.Kooij@tudelft.nl

2023 7th International Conference on System Reliability and Safety (ICSRS) | 979-8-3503-0605-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSRS59833.2023.10381264

***Abstract*—In this paper we investigate the controller placement problem on networks using controller reachability as the network performance metric. This metric is defined as the probability that each node can reach at least one controller, given that each link is operational with a fixed probability. By exploring placements for more than 100 real-world networks and by varying the number of controllers from two to five, we find that controller reachability varies greatly with different placements. Obviously, increasing the number of controllers increases the controller reachability. However, the extent of this increase depends on the strategy with which the controllers are placed. The findings indicate that efficient controller placement strategies should be developed to ensure good network performance. In this research, we propose four controller placement strategies. One strategy is based on topological network metrics: node degree and path length between controllers and nodes. The other three heuristic strategies are the greedy algorithm, the classic genetic algorithm and the heuristic genetic algorithm. By validating strategies on real-world networks, we find that all four strategies work well to solve the controller placement problem with respect to controller reachability.**

***Index Terms*—Controller reachability, controller placement, and heuristic algorithms**

I. Introduction

The Controller Placement Problem (CPP) on networks is one of the facility location problems, which looks to place facilities in potential locations, such that certain performance metrics are optimized, often under constraints of cost. The study of controller placement problems has been widely investigated in the setting of Software-Defined Networking (SDN). The SDN network can be divided into the data plane, the control plane, and the application plane, where the data plane corresponds to the networking devices that are responsible for forwarding data like switches. The control plane decides how to handle the network traffic using controllers, and the application plane remotely monitors and configures the control functionality through the interface (northbound API) [1]. SDN has a logically centralized control architecture achieved by decoupling the network control plane and the data plane [2].

Within the Controller Placement Problem (CPP) in Software-Defined Networking (SDN), two fundamental research questions arise, each crucial to optimizing network performance [3]. The first question centers on determining the appropriate number of controllers to be deployed in the network. Although a single controller is typically sufficient for smaller networks, larger networks require the addition of multiple controllers to maintain optimal performance levels [4]. The use of a multi-controller design has shown promising results in improving the performance of the SDN network [5], albeit at the expense of increased deployment costs. Hence, determining the optimal number of controllers becomes imperative for striking a balance between performance and cost considerations.

The second research question relates to the strategic placement of controllers within the network. The performance of the network is inherently influenced by the specific locations chosen for the controllers, given a fixed number of controllers. As the number of controllers increases, the number of potential placement combinations escalates exponentially, intensifying the challenge of identifying optimal configurations. Consequently, the question of where to place the controllers emerges as a critical consideration. Addressing this question is vital to achieving superior network performance by minimizing latency, optimizing resource utilization, and ensuring efficient communication between the control plane and the data plane. The solutions to the research questions heavily rely on the formulation approaches employed for the CPP in SDN.

The formulation of the CPP in SDN requires the consideration of various performance metrics that directly impact the design of effective controller placement strategies. Three fundamental metrics, namely latency, reliability, and cost, play critical roles in the evaluation and optimization of the placement of controllers [2]. The first performance metric, latency, captures the time delays experienced during data transmission and processing. Transmission latency encompasses delays that occur between switches and controllers, as well as inter-controller communication. It is often quantified using the aver-

age or maximum distances between the network components. Processing latency arises when controllers become overloaded, exceeding their processing capacity. Minimizing latency is crucial to ensure efficient communication between the control plane and the data plane, resulting in improved network responsiveness. Reliability constitutes the second performance metric and is concerned with the probability of successfully reaching controllers in the event of network component failures. Low reliability levels can lead to packet losses due to the absence of viable transmission paths. Evaluating the probabilities associated with multiple control paths and determining the availability of such paths are vital in ensuring network robustness and fault tolerance. The third performance metric encompasses cost considerations. Economic costs encompass expenses related to construction, maintenance, operation, and other financial factors. Environmental costs, such as energy consumption and CO_2 emissions, are also taken into account. Balancing costs while optimizing controller placements is essential for achieving economically and environmentally sustainable SDN deployments. The performance metrics can be used in isolation or in combination to formulate the CPP as either a single-objective optimization problem, aiming to optimize a specific metric, or a multi-objective optimization problem, considering multiple metrics simultaneously [6].

Determining the number of controllers required is a fundamental challenge, and it is closely intertwined with the problem of where to place the controllers. The NP-hard nature of the controller placement problem necessitates the exploration of diverse algorithms and methodologies to identify optimal placements. One commonly employed method is the brute force algorithm, which aims to find the optimal solution, simply by considering the full state space of all possible placements. However, this algorithm is inefficient and suitable primarily for small-scale networks. To address larger-scale networks, heuristic algorithms, including the greedy algorithm, simulated annealing, genetic algorithm, and others, have been widely adopted. Additionally, linear programming and graph-based algorithms have been proposed as alternative approaches to tackle the complexity of the controller placement problem [7].

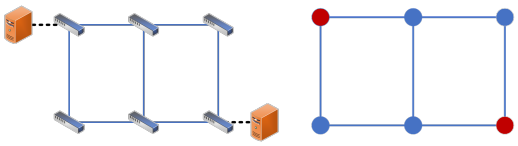
Heller *et al.* [3] propose the controller placement problem with a focus on minimizing propagation delays in wide-area networks. They utilize average latency (k-median) and worst-case latency (k-center) as performance metrics to optimize controller placements. By exhaustively enumerating every combination of controllers, they find that random placement falls significantly short of the optimal placement in nearly all network topologies, thereby underscoring the substantial impact of placement on network performance. Furthermore, in most topologies, it is challenging to optimize both average latency and worst-case latency simultaneously. Hu *et al.* [8] formulate the reliability-aware controller placement (RCP) problem. The expected percentage of control path loss is considered as a metric to reflect the reliability of networks. In the RCP problem, they look for the number of controllers and the placement in a network with given failure probability of each component such that the reliability is optimized. They find that reliability and latency cannot be optimized at the same time for most networks. However, the placement with optimal reliability exhibits quite a good average latency as well. Ros *et al.* [9] introduce the Fault Tolerant Controller problem, wherein a heuristic algorithm is developed to determine the optimal controller placement, considering the lower bound reliability as a performance metric. Zhong *et al.* [10] define a reliability metric in the control network as the average number of disconnected switches when a single edge fails. They propose a min-cover-based method to minimize the number of controllers and maximize reliability. The concept of "cover" is employed, representing a set of nodes with controllers to ensure that every switch in the network belongs to at least one controller's neighborhood, guaranteeing low propagation delays.

In this research, we aim to investigate efficient controller placement strategies using a reliability metric. We assume that nodes, including controllers, are always operational. At the same time, links between nodes and controllers or between controllers are also assumed to be always operational. However, links among nodes are subject to operational probabilities. In addition to traditional reliability metrics based on paths, we introduce a novel reliability metric called controller reachability, which quantifies the probability that nodes can reach at least one controller. To calculate the exact controller reachability value, we employ a path composition algorithm. By analyzing more than one hundred real-world networks and considering various controller placement scenarios, we aim to highlight the differences among placements and emphasize the importance of efficient placement techniques. Subsequently, we propose different controller placement strategies, including a strategy based on graph metrics, a greedy algorithm, and two strategies utilizing genetic algorithms. To validate the effectiveness of the proposed placement strategies, we conduct experiments on five representative medium-sized real-world networks.

The paper is organized as follows: Section 2 provides an introduction to the concept of controller reachability and outlines the algorithms used for its calculation. In Section 3, we describe the dataset utilized in our research. Section 4 presents the proposed controller placement strategies. The subsequent Section 5 presents the empirical results obtained from various network placements and different strategies. Finally, Section 6 offers a comprehensive conclusion and discussion.

II. Controller reachability

The data plane in SDN can be mathematically represented as an undirected graph, denoted as $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents the set of nodes corresponding to network devices, and $\mathcal{L}$ represents the set of links representing connections between network devices. The graph consists of N nodes and L links. Each link l_{ij} indicates the presence of a connection between node i and node j. Notably, in the data plane, controllers are co-located with nodes, assuming that connections between controllers and nodes are always operational. Fig. 1 provides



(a) SDN network (b) Graph representation

Fig. 1: Graph representation of SDN network.

a graphical representation of an SDN network using a graph model. Each node in the graph represents a network device, and the presence of a red node signifies the co-location of a controller. We assume that nodes within the network always remain operational, and links have an operational probability p.

We define controller reachability $C_r(K, p, s)$ as the probability that each node can reach at least one controller, assuming each link is operational with probability p, when placing K controllers in the graph G on the node set s, where the cardinality of set s is K and $s \subseteq \mathcal{N}$. To calculate controller reachability, we enumerate the cases of $m, 1 \leq m \leq L$ operational links and count the number of case I_m in which each node can reach at least one controller. The probability that each node can reach at least one controller given m operational links and $L-m$ nonoperational links is $I_m p^m (1-p)^{L-m}$. By considering all possible cases, the controller reachability with K controllers placed at node set s can be obtained as follows,

$$C_r(K, p, s) = \sum_{m=1}^{L} I_m p^m (1-p)^{L-m}. \tag{1}$$

The aforementioned enumeration method is suitable for small-scale networks, but it becomes computationally infeasible for larger networks due to the exponential growth in running time with an increasing number of links. To address this issue, we can estimate the controller reachability $\hat{C}_r(K, p, s)$ for a given operational probability value p using Monte Carlo simulation [11]. This estimation is obtained by performing a large number (n) of independent sampling trials as follows:

$$\hat{C}_r(K, p, s) = \frac{1}{n} U_n \tag{2}$$

where U_n represents the number of samples in which every node successfully reaches at least one controller. However, it is worth noting that Monte Carlo Simulation may not be computationally efficient when estimating rare event probabilities [12]. Achieving a good approximation often requires running the simulation for a significant number of iterations.

Given the limitations of the enumeration and Monte Carlo simulation methods in terms of network size and computation time, we propose a modified approach to calculate controller reliability by utilizing a path decomposition algorithm. The path decomposition algorithm [13], offers an efficient means to compute the exact value of all-terminal reliability. In the context of our problem, we leverage this algorithm to

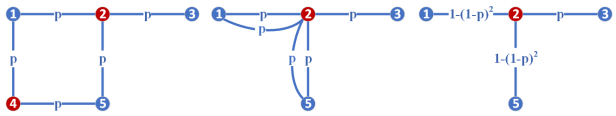


Fig. 2: Example of controllers merging.

calculate controller reachability, which is equivalent to all-terminal reliability when a single controller is present in the graph. To apply the path decomposition algorithm for cases where there are multiple controllers in the network, we introduce a modification of the problem at hand. Specifically, we consider two approaches: one approach is to add links between controllers and set the operational probability of links between controllers to one in the network; the other is to merge all controllers together as a super-controller. The links connecting controllers and nodes are modified to connect the super-controller with nodes. To eliminate duplicate links, we adjust the link operational probabilities accordingly. By employing the merging approach, we reduce the number of nodes and edges in the graph compared to the approach that uses the addition of links, which has the potential to increase the path width. The resulting network comprises a single controller and does not contain any duplicate links, allowing us to apply the path decomposition algorithm to calculate the all-terminal reliability efficiently. To illustrate the process of controllers merging, we provide an example in Fig. 2.

III. Data Set

In this study, we utilized real-world communication networks from the Topology Zoo dataset [14]. We specifically selected a subset of networks consisting of 100 small-sized networks with the number of nodes ranging from 11 to 30, as well as five medium-sized networks with over 50 nodes. To provide an overview of the networks used, we present the average node degree and network sizes in Fig. 3. The average node degree spans from 1.875 to 4.48, and among the networks, the smallest in size is Abilene with 11 nodes and 14 links, while the largest is Cogentco with 197 nodes and 243 links. Additionally, we present the number of nodes (N), the number of links (L), and the average degree d_{av}, the number of nodes with degrees equal to one and two ($d = 1$ and $d = 2$) for the five medium-sized networks in TABLE I.

	N	L	d_{av}	$d = 1$	$d = 2$
HinerniaGlobal	55	81	2.945	1	20
Syringa	74	74	2.000	23	34
Interoute	110	146	2.655	8	53
Cogentco	197	243	2.467	22	95
GtsCe	149	193	2.591	12	80

TABLE I: Properties of five medium-sized real-world networks from the Topology Zoo.

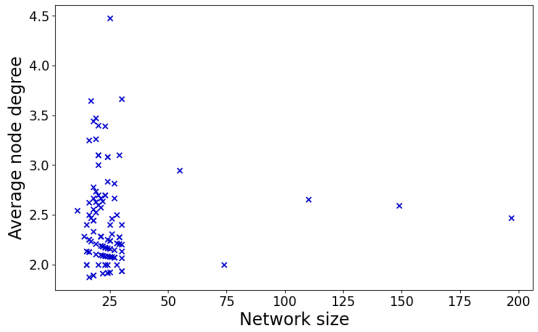


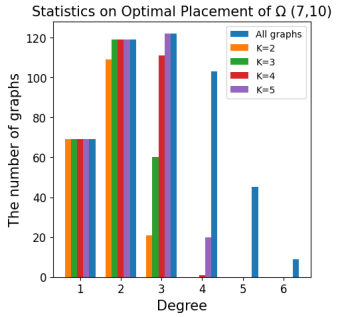
Fig. 3: Average node degree with respect to network size for 100 small real-world networks and five medium real-world networks from the Topology Zoo.

IV. Placement Strategies

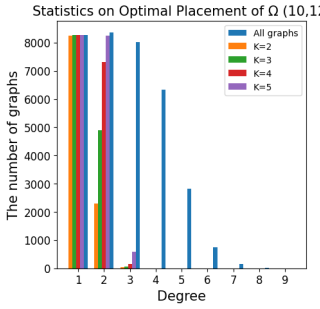
A. Controller placement strategy based on degree and distance

Nodes with low degrees, specifically degrees equal to one and two, exhibit higher vulnerability to disconnection within a network. Testing on three distinct sets of graphs, denoted as $\Omega(N, L)$, where each set encompasses all non-isomorphic connected graphs with the same numbers of nodes and links, we find that the optimal placements prefer to include nodes with low degrees. Specifically, we investigated the sets $\Omega(7, 10)$, $\Omega(10, 12)$, and $\Omega(9, 18)$, which contain 132, 8548, and 33366 graphs, respectively. The optimal placements with varying numbers of controllers (2, 3, 4, and 5) are determined for every graph within these selected sets. During this process, the degrees of the nodes corresponding to the optimal controller placements are recorded. The acquired information produces statistical findings, which are shown in Fig. 4. The results reveal a significant trend suggesting a preference for nodes with lower degrees in the optimal placement of controllers.

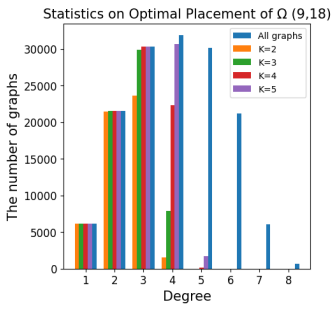
To enhance controller reachability, it is effective to assign higher priorities to nodes with lower degrees. Moreover, considering the distance between nodes and controllers is also important. Fig. 5 presents a comprehensive analysis of the distances between two controllers across our set of 100 small real-world graphs. Each column within the figure represents a specific graph, while each data point signifies a possible distance between two controllers. Notably, the red data points correspond to the distances observed when the optimal placements are implemented. By examining the positioning of these red data points within their respective columns, valuable insights can be gained regarding the relationship between optimal controller placements and the distance. Specifically, if a red point is at the top of its column, it indicates that the optimal placement of this graph is the node pair with the maximum distance. When the link operational probability $p = 0.99$, the optimal placements of 67 graphs are observed to be the node pairs with the maximum distance, while the optimal placements of 14 graphs correspond to the node pairs with the second-largest distance. None of the analyzed graphs



(a) $\Omega(7, 10)$



(b) $\Omega(10, 12)$



(c) $\Omega(9, 18)$

Fig. 4: Statistics on optimal placement of three sets of graphs. The blue bars represent the number of graphs containing nodes with a specific degree, while the remaining bars indicate the number of graphs where the optimal placement (at $p = 0.99$) of K controllers includes at least one node with that degree.

exhibit an optimal placement where the selected node pair consists of two neighboring nodes, i.e. with a distance of one. It can be concluded that distance is a significant graph metric that influences controller reachability. When placing two controllers, the optimal placement tends to occur at the node pair with the maximum distance. This preference for maximizing distance helps ensure that no node within the network is located too far away from the controllers.

Considering these two graph metrics, we propose an effective strategy for placing K controllers. The core concept of this strategy involves partitioning nodes into distinct sets based on their degrees, with a preference for selecting nodes from sets with smaller degrees. In cases where nodes have the same degree, the algorithm aims to maximize the distance between controllers, thereby minimizing the overall distance between controllers and nodes.

Specifically, in the graph G, we denote the lowest degree as d_1, the second lowest degree as d_2 (where $d_1 \neq d_2$), the i-th lowest degree as d_i and the highest degree as d_M ($d_1 < d_2 < \ldots < d_M$). We consider n_1 nodes with degree d_1 as set $S(d_1)$, n_2 nodes with degree d_2 as set $S(d_2)$, n_i nodes with degree d_i as set $S(d_i)$. Besides, we use $d_0 = 0$, $n_0 = 0$, and $S(d_0) = \emptyset$ to indicate that there are no nodes with a degree of 0 in the network. Consequently, we obtain M non-empty sets of nodes that do not overlap and collectively cover every node in graph G. We aim to find the node set $S(d_k)$ such

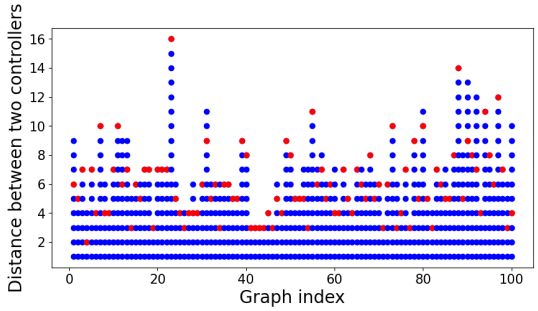


Fig. 5: Statistics on optimal placement of 100 real-world graphs ($K = 2, p = 0.99$). The x-axis denotes the index of graphs, and the y-axis denotes the distance between two controllers. In this figure, each point represents a possible distance between two controllers. The red points represent the distance between two controllers when the optimal placements are employed.

that $\sum_{i=0}^{k-1} n_i < K \leq \sum_{i=0}^{k} n_i$. The node sets with degree lower than d_k are defined as the initial existing controllers set $S_C = \bigcup_{i=0}^{k-1} S(d_i)$, the controllers are placed on every node in this set due to their low degree. The number of remaining controllers is defined as $K' = K - \sum_{i=0}^{k-1} n_i$. The nodes in $S(d_k)$ are considered as potential locations for placing K' controllers according to the distance.

If $K \leq n_1$, $S_C = \emptyset$, the nodes in $S(d_1)$ are considered as potential locations to place $K' = K$ controllers. For each node in set $S(d_1)$, we compute the sum of its distance to other nodes in G. The node with the highest sum of distances is selected as the location for the first controller. From the placement of the second controller onwards, we select the node that has the longest distance to the existing controllers as the location for the next controller.

If $K > n_1$, the node set that we are going to place K' controller is determined as follows: If $n_1 < K \leq n_1 + n_2$, $S_C = S(d_1)$, the nodes in $S(d_2)$ are considered as potential locations for placing $K' = K - n_1$ controllers. If $n_1 + n_2 < K \leq n_1 + n_2 + n_3$, $S_C = S(d_1) \cup S(d_2)$, the nodes in $S(d_3)$ are considered as potential locations for placing $K' = K - n_1 - n_2$ controllers, etc. Using this method, we can identify the locations of $K - K'$ controllers based on the node degree, and then place the remaining K' controllers based on the distance within a smaller set of nodes. Within this set, the distance between each node and the existing controllers is computed. The node with the longest distance to the existing controllers is selected as the location for the next controller. This process is repeated until K nodes are identified. The pseudo-code of the algorithm is presented in Algorithm 1.

Algorithm 1 Controller Placement Algorithm based on degree and distance

Input: network G, number of controllers K
Output: Set S_C

Define set S_C as the set of nodes with controllers
Define d_i as the i-th lowest degree
Define n_i as the number of nodes with degree d_i
Define $S(d_i)$ as the set of nodes with degree d_i
Define $n_0 = 0$, $S(d_0) = \emptyset$
Define $distance(u, v)$ as the shortest path length between u and v
Find the set $S(d_k)$ such that $\sum_{i=0}^{k-1} n_i < K \leq \sum_{i=0}^{k} n_i$
$S_C = \bigcup_{i=0}^{k-1} S(d_i)$
$K' = K - \sum_{i=0}^{k-1} n_i$
if $K < n_1$ **then**
  for $v \in S(d_1)$ **do**
    $SumDistance(v) = \sum_{u \in G, u \neq v} (distance(u, v))$
  end for
  Add the node v with the highest $SumDistance(v)$ into S_C
  $K' = K' - 1$
end if
while $K' > 0$ **do**
  for $v \in S(d_k)$ **do**
    $D(v) = \min(distance(u, v))$ where $u \in S_C$
  end for
  Add the node v with the highest $D(v)$ into S_C
  $K' = K' - 1$
end while
return Set S_C

B. Greedy algorithm

The greedy algorithm is employed to determine controller placements in a step-by-step manner. At each iteration, the algorithm selects a location that maximizes the improvement in the performance metric, which in our case is the controller reachability. Since each decision is made based solely on the available information at that step, the greedy algorithm may yield a solution that is locally optimal but not necessarily globally optimal.

When placing the first controller in the network, there is no distinction in terms of controller reachability among the available options. However, the choice made for the first controller will impact the subsequent decisions. To address this "black start" problem encountered in the greedy algorithm, we explore four different approaches to start the algorithm:

- Randomly choose a node,
- Randomly choose a node with the lowest degree,
- Use Algorithm1 to determine the first node,
- Enumerate all possible placements for $K = 2$ controllers and select the optimal solution as the first two controllers to be placed.

Not surprisingly, randomly choosing a node performs poorly when placing the first few controllers, but it gradually approaches the performance of other methods. Enumerating all possible placements with two controllers yields the best performance, but it is rather time-consuming. The remaining two methods both select a node with the lowest degree, but Algorithm 1 also takes into consideration the distance. Therefore, we have ultimately decided to use Algorithm 1

to determine the first node in the greedy algorithm. The computational time required for determining the first node is short, almost negligible compared to the running time of the greedy algorithm.

Starting from the second controller, the greedy algorithm iterates over all nodes without controllers and selects the node that offers the highest improvement in controller reachability. This process is repeated until K controllers have been placed. The pseudo-code for the greedy algorithm is provided in Algorithm 2.

Algorithm 2 Greedy algorithm

Input: network G, number of controllers K
Output: Set S_C

```
Define set S_C as the set of nodes with controllers
The first controller S_C1 is chosen based on algorithm 1
K = K − 1
while K > 0 do
    for node v ∉ S_C do
        Compute the controller reachability C_r(v) if controllers are placed at node v and nodes in S_C
    end for
    Add node v with the highest C_r(v) into set S_C
    K = K − 1
end while
return Set S_C
```

C. Genetic algorithms

The genetic algorithm is a well-known meta-heuristic algorithm that draws inspiration from biological evolution. In each generation, individuals from the current population are selected as parents, and through their reproduction, new children are generated. The algorithm aims to gradually progress towards the optimal solution by continually improving the fitness of the individuals in the population. The genetic algorithm encompasses several key elements, namely chromosome representation, selection, crossover, mutation, and fitness function computation [15]. Selection, mutation, and crossover are often referred to as biologically-inspired operators. In this research, we developed two genetic algorithms: the classic genetic algorithm and the heuristic genetic algorithm. Both algorithms employ the same method for generating the initial population, and the fitness value is determined based on the controller reachability.

In the context of the controller placement problem, an individual in the genetic algorithm population is represented as a sequence of K genes, where each gene corresponds to the node that hosts a controller. To generate the initial population, we employ the method proposed in [16], which ensures that each gene has a similar frequency of occurrence in the population. Specifically, the gene frequency f in the initial population is determined by the following equation,

$$f = \max\left\{2, \left\lceil \frac{N}{100} \cdot \frac{\ln(n_s)}{d} \right\rceil \right\} \tag{3}$$

where N is the number of nodes, K is the number of controllers, $n_s = \binom{N}{K}$ is the number of possible placements, $d = \lceil \frac{N}{K} \rceil$ is the rounded-up density of the problem. The gene frequency is at least two. The initial population size P can be calculated by $P = f \cdot d$.

After determining the frequency and the population size, the nodes are assigned to each solution. In the first set of d solutions, the nodes $1, 2, \ldots, K$ are assigned to the first solution, the nodes $K+1, K+2, \ldots, 2K$ are assigned to the second solution, etc. The process is repeated until all nodes are assigned to solutions, resulting in d solutions. If N/K is not an integer, random non-repeating genes are selected to fill the last solution. In the second set of d solutions, the nodes $1, 3, \cdots 2K-1$ are assigned to the first solution, the nodes $2K+1, 2K+3, \cdots 4K-1$ are assigned to the second solution, etc. The process continues until f sets of solutions are obtained. By adjusting the increment of nodes when generating different sets of solutions, we ensure that there are no repeated solutions in the initial population. Next, we will present how the two algorithms work in the subsequent steps.

1) Classic genetic algorithm:

- **Selection operator** Binary tournament selection, a selection method known for its fast convergence and ease of implementation [17], is employed in this study. Two individuals are randomly sampled from the population, and the tournament is conducted based on their fitness values. The individual with the highest fitness value is selected to proceed with the following operation.
- **Crossover operator** Partial Mapped Crossover (PMX) is employed as the crossover operator in this study. It is recognized as the most commonly used technique for permutation-encoded chromosomes. PMX ensures that the generated offspring do not contain any duplicate genes [15], making it particularly suitable for this context. Compared to other crossover operators, PMX has demonstrated superior performance. The crossover rate is set to $p_c = 0.8$.
- **Mutation operator** The mutation operator is employed to preserve the genetic diversity within the population, thereby preventing the algorithm from converging to a locally optimal solution. Randomly selecting a gene from an individual, the mutation operator replaces it with another gene that is not present in that individual. This process guarantees that the offspring do not contain any duplicate genes. In this research, the mutation rate is set to $p_m = 0.1$.

The classic genetic algorithm will terminate after the iteration times reaches the preset value. The pseudo-code of the classic genetic algorithm is presented in Algorithm 3.

2) Heuristic genetic algorithm: The heuristic genetic algorithm employed in this research is based on the algorithm proposed by O. Alp *et al.* for solving the facility location problem, where evolution is facilitated by a greedy heuristic [16]. In the heuristic genetic algorithm, only the crossover operator is utilized. Unlike the traditional crossover method that involves exchanging genes between two individuals to produce

Algorithm 3 Classic genetic algorithm

Input: network G, number of controllers K, maximum number of iterations MAX
Output: Set S_C

Define set S_C as the set of nodes with controllers
Determine the population size P
Initializing the population
Compute the fitness value of each individual
Set iteration counter $t = 0$
while $t < MAX$ **do**
    Select P individuals from the population using tournament selection.
    Apply crossover on $P/2$ pairs of individuals with crossover probability.
    Apply mutation on the offspring with mutation probability.
    New population with size P is generated
    $t = t + 1$
end while
Add nodes in the best solution in the last iteration to set S_C
return Set S_C

offspring, this novel genetic algorithm begins by combining the genes of the two parents. Subsequently, a greedy deletion heuristic is applied to reduce the number of genes until the solution contains exactly K genes. Furthermore, instead of generating a new population at each iteration, the algorithm continuously updates the initial population.

- **Crossover operator**
  Two individuals are randomly selected as parents to initiate the crossover process. The genes of the parents are combined to create a temporary offspring. However, this offspring cannot be directly utilized in the subsequent steps as it contains $2K$ genes, which exceeds the desired number of K genes. To refine the offspring and ensure it consists of only K genes, a removal procedure is applied. The gene removals follow two rules until K genes are kept:
  - The gene that is present twice is kept.
  - The gene that contributes least to improving controller reachability is removed.

  Although the crossover increases the time demands, the quality of the offspring is improved.

Whenever a new individual is produced through crossover, it undergoes a comparison process with the existing members of the population. If the generated individual is not identical to any of the current population members and exhibits a better fitness value (controller reachability) than the worst fitness value in the population, it will replace the worst individual. This replacement mechanism facilitates the gradual improvement of the average quality of the entire population. Furthermore, the population maintains good diversity as it consists of non-duplicate individuals. The termination condition for the heuristic algorithm is reached when the best fitness value remains unchanged in successive iterations. The pseudo-code of the heuristic genetic algorithm is presented in Algorithm 4.

Algorithm 4 Heuristic genetic algorithm

Input: network G, controllers' number K
Output: Set S_C

Determine the population size P
Initializing the population
Compute the fitness value of each individual
Store the best value and the worst value
Set iteration counter $t = 0$
while $t < n$ **do**
    Select two individuals from the population randomly
    Apply crossover and generate one offspring
    if offspring not in population **then**
        Compute the fitness value of offspring
        if fitness value $>$ worst value **then**
            Update population
        else
            $t = t + 1$
        end if
    else
        $t = t + 1$
    end if
end while
Add nodes of the best solution to set S_C
return Set S_C

V. Results

A. How does placement affect controller reachability

To investigate performance disparities among different controller placement strategies, we conducted a comprehensive analysis of controller reachability for the small-sized real-world networks with the number of nodes ranging from 11 to 30. Specifically, we considered the placement of 2, 3, 4, and 5 controllers with an operational probability p set to 0.99. The resulting controller reachability values were utilized to determine the maximum, minimum, and average reachability, represented by error bars in Fig. 7. Analysis of the figure reveals that only a few networks achieve near-optimal controller reachability when randomly placing controllers. For most networks, the average reachability values deviate significantly from the optimal levels. To further substantiate our observations, we employed max-min scaling on our set of 100 graphs to quantify the deviation from the average reachability value (random placements) to the optimal value (optimal placement). The scaling process involved dividing the range between the scaled maximum value (1) and the scaled minimum value (0) into four intervals: (0, 0.25), (0.25, 0.5), (0.5, 0.75), and (0.75, 1). When the scaled value equals 0, the average controller reachability is the farthest from the optimal reachability and when the scaled value equals 1, the average controller reachability is the optimal reachability. The

K	(0, 0.25)	(0.25, 0.5)	(0.5, 0.75)	(0.75, 1)
2	31	52	11	6
3	25	48	21	6
4	17	50	27	6
5	8	54	31	7

TABLE II: The number of graphs based on the scaled average reachability values in each interval with $K = 2, 3, 4, 5$.

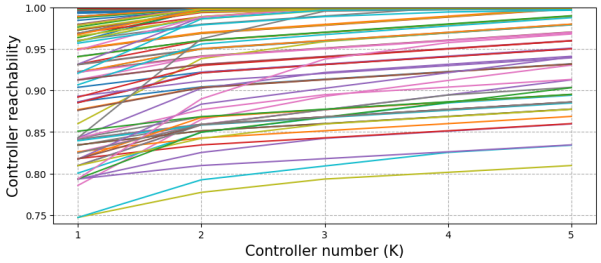


Fig. 6: The optimal controller reachability of 100 graphs. In this figure, the x-axis denotes the number of controllers, the y-axis denotes the controller reachability. Each curve represents the optimal controller reachability of a graph with $K = 1, 2, 3, 4, 5$ at $p = 0.99$.

number of graphs falling within each interval based on the scaled average reachability values is shown in TABLE II.

Our analysis reveals that an increasing number of networks fall within the intervals (0.5, 0.75) and (0.75, 1) as the number of controllers (K) increases, which suggests that the performance of random placement approaches closer to that of the optimal placement with a larger K. However, for most networks, the controller reachability of random placements is still far away from the controller reachability with the optimal placement, regardless of the value of K. Consequently, it is imperative to employ efficient strategies for controller placement to achieve enhanced performance for most networks.

B. How many controllers are needed

To investigate the optimal number of controllers required in the networks, we analyzed our 100 graphs to determine the optimal controller placements for varying values of K, ranging from one to five, as depicted in Fig. 6. Although the sizes of the networks are close, the controller reachability with a single controller exhibits a significant range, spanning from 0.75 to 0.9995.

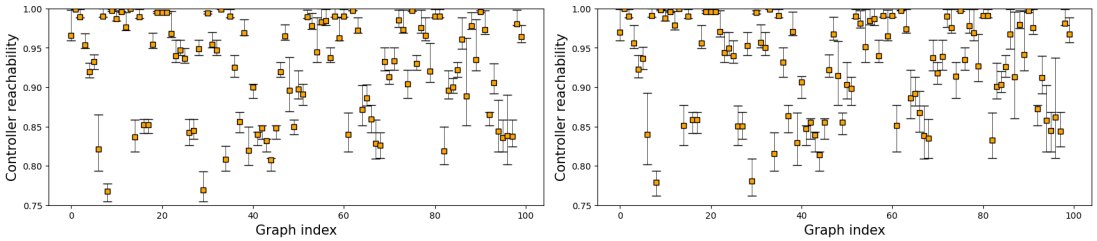
Upon setting a controller reachability threshold of 0.995, we observed that among the analyzed networks, six networks achieved this criterion with a single controller, while 26 networks necessitated two controllers. Furthermore, ten networks met the requirement with three controllers, one network required four controllers, and four networks necessitated five controllers. Notably, 53 networks could not attain a controller reachability of 0.995 even with five controllers placed.

Determining the optimal number of controllers should be based on specific requirements, link probability p, and network topology. It is impractical to establish a universal number that applies to all networks. Nevertheless, our observations indicate that deploying multiple controllers consistently enhances controller reachability across all networks. Notably, when $K = 2$, all the curves show a significant improvement in controller reachability with just two controllers.

C. Compare different placement strategies

To assess the effectiveness of the proposed four placement strategies, we conducted experiments using five medium-sized networks. Considering the large number of nodes in each graph and the observation that optimal placements often involve nodes with low degrees, we investigated the feasibility of utilizing nodes with degrees below the average degree as potential candidates instead of all nodes for controller placements. In addition to the four strategies, we compared the results with random placements among all nodes, which were generated through 10000 simulations, where the average value was computed over the 10000 realizations. For the HinerniaGlobal network and the Syringa network, we also obtained the optimal placements for different values of K within the potential nodes set, comprising nodes with degrees equal to one and two. The outcomes for the five medium-sized networks are presented in Fig. 8. Notably, all four placement strategies yielded satisfactory results for the HinerniaGlobal, Interoute, and GtsCe networks. However, the graph metric-based strategy exhibited relatively poor performance compared to the other methods for the Syringa and Cogentco networks. The discrepancy can be attributed to the influence of network topology.

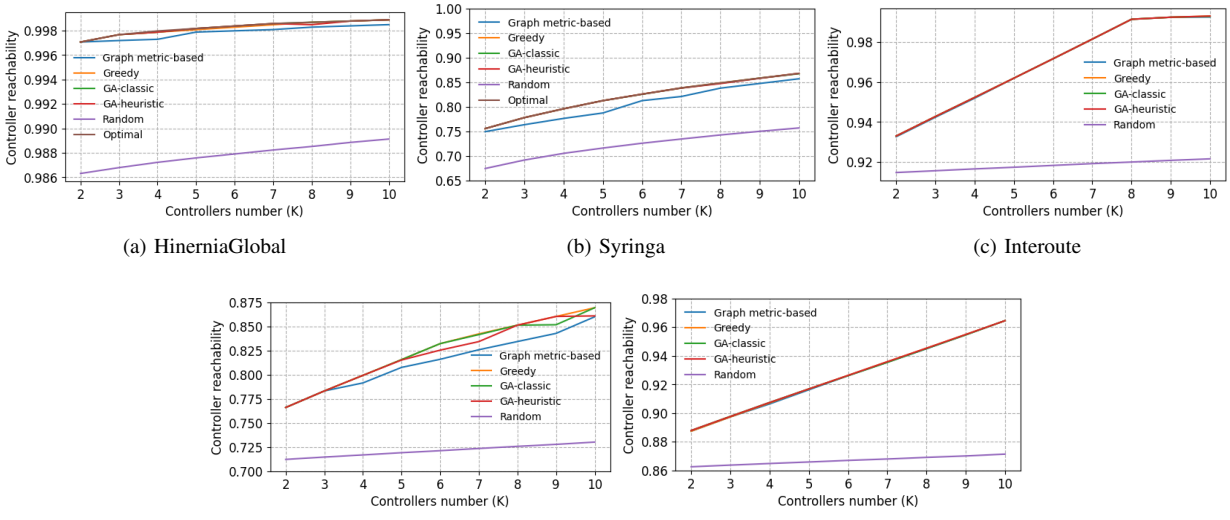
To elucidate the limitations of the graph metric-based strategy in certain scenarios, we compared its node selection approach with the greedy method, which also involves selecting nodes one by one. Here, we present an example using the Syringa network with ten controllers and a link operational probability of $p = 0.99$, as depicted in Fig. 9. In this example, both the greedy algorithm and the graph metric-based strategy initially select node 5 as the first controller. However, as $K = 2$, the subsequent controller selections differ between the two strategies. The greedy algorithm chooses node 18 as the location for the second controller, whereas the graph metric-based method selects node 21. Upon closer examination of the network topology, it becomes evident that the greedy algorithm makes a more advantageous choice. Although node 21 has the greatest distance from node 5, with a distance of 31, node 18 is more prone to disconnection due to its connection to the "ring" portion of the network, which requires a path length of three. The path length is higher than the path length between node 21 and the "ring" portion. In the Syringa network, the "ring" portion can be considered relatively more reliable. Consequently, in scenarios where similar situations arise, optimal decisions cannot be solely based on node degree and distance metrics. The provided example serves to emphasize the limitations of the graph metric-based strategy in certain network topologies.



(a) $K = 2$ (b) $K = 3$

(c) $K = 4$ (d) $K = 5$

Fig. 7: Controller reachability error bars of 100 small real-world graphs when $K = 2, 3, 4, 5$ and link operational probability equals to 0.99. The x-axis denotes the index of graph, the y-axis denotes the controller reachability. Each error bar represents the maximim, minimum, and average controller reachability of all possible placements of K controllers.



(a) HinerniaGlobal (b) Syringa (c) Interoute

(d) Cogentco (e) GtsCe

Fig. 8: Comparison of placement strategies on five real-world networks by placing different numbers of controllers with link operational probability p equal to 0.99. The x-axis denotes the number of controllers, the y-axis denotes the controller reachability.

VI. Conclusion

In this study, we adopt controller reachability as the performance metric for addressing the controller placement problem. Controller reachability is defined as the probability that each node can reach at least one controller. To evaluate controller reachability, we employ the path decomposition algorithm, commonly used for computing all-terminal reliability. By applying this algorithm, we can calculate the controller reachability and quantify the impact of different placements on network performance. Our analysis reveals that the choice of placement strategy can significantly influence controller reachability, highlighting the importance of efficient placement strategies. Moreover, we investigate how many controllers are necessary to ensure satisfactory performance. For small-size networks, introducing a second controller yields substantial improvements in controller reachability for most networks.

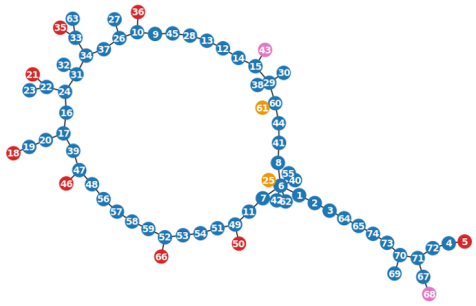


Fig. 9: Syringa: ten controllers chosen by the graph metric-based strategy and the greedy algorithm. The red nodes are found by both strategies, the pink nodes are only found by the greedy algorithm, and the yellow nodes are only found by the graph metric-based strategy.

However, it is important to note that the number of controllers required depends on the specific network topology and the performance requirements in real-world scenarios. Thus, determining the optimal number of controllers necessitates consideration of both network topology and the desired performance levels.

Upon analyzing the optimal placements for various graphs, we have determined that two topological graph metrics, namely degree and distance, heavily influence controller reachability. Building upon this insight, we propose a controller placement strategy that leverages degree and distance as key factors. Additionally, we develop three heuristic algorithms: the greedy algorithm, the classic genetic algorithm, and the heuristic genetic algorithm, to facilitate controller placement. To evaluate the effectiveness of these strategies, we conducted experiments using real-world graphs from the Topology Zoo dataset. The strategy based on graph metrics proves to be a time-efficient approach, delivering controller placements comparable to those obtained using other heuristic methods. Although the strategy based on graph metrics exhibits certain limitations for specific network topologies, its overall performance is satisfactory. Among the three heuristic algorithms, the greedy algorithm demonstrates superior performance across all networks. Both the classic genetic algorithm and the heuristic genetic algorithm are more computationally intensive, yet they do not surpass the performance of the greedy algorithm. Considering both performance and algorithm complexity, we conclude that the strategy based on graph metrics and the greedy algorithm are suitable approaches for addressing the controller placement problem.

For future work, we might look into the following aspects: 1) the impact of nodes being non-operational with a certain probability; 2) other performance metrics like the worst or the average node-controller reachability where node-controller reachability is defined as the probability of a node being able to establish communication with at least one controller; 3) the impact of links between network devices and controllers being non-operational with a certain probability; 4) improvement of the strategy based on graph metrics by incorporating additional topology properties, such as the existence of multiple paths between network devices and controllers; 5) the study of multi-objective optimization, such as considering both latency and controller reachability.

Acknowledgment

F. Wang thanks the financial support from the China Scholarship Council (No.201906040194).

References

[1] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, pp. 24 290–24 307, 2019.

[2] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101–118, 2018.

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 473–478, 2012.

[4] M. Dhar, A. Debnath, B. K. Bhattacharyya, M. K. Debbarma, and S. Debbarma, "A comprehensive study of different objectives and solutions of controller placement problem in software-defined networks," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 5, p. e4440, 2022.

[5] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller based software-defined networking: A survey," *IEEE Access*, vol. 6, pp. 15 980–15 996, 2018.

[6] M. Dhar, A. Debnath, B. K. Bhattacharyya, M. K. Debbarma, and S. Debbarma, "A comprehensive study of different objectives and solutions of controller placement problem in software-defined networks," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 5, p. e4440, 2022.

[7] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.

[8] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[9] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 31–36.

[10] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in sdn," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 481–487.

[11] V. Gaur, O. P. Yadav, G. Soni, and A. P. S. Rathore, "A literature review on network reliability analysis and its engineering applications," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 235, no. 2, pp. 167–181, 2021.

[12] J. L. Beck and K. M. Zuev, "Rare event simulation," *arXiv preprint arXiv:1508.05047*, 2015.

[13] J. Carlier and C. Lucet, "A decomposition algorithm for network reliability evaluation," *Discrete Applied Mathematics*, vol. 65, no. 1-3, pp. 141–156, 1996.

[14] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[15] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, pp. 8091–8126, 2021.

[16] O. Alp, E. Erkut, and Z. Drezner, "An efficient genetic algorithm for the p-median problem," *Annals of Operations research*, vol. 122, pp. 21–42, 2003.

[17] J. Zhong, X. Hu, J. Zhang, and M. Gu, "Comparison of performance between different selection strategies on simple genetic algorithms," in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, vol. 2. IEEE, 2005, pp. 1115–1121.