On Using Genetic Algorithm for Optimal Controller Placement in Software-Defined Networks

Emmanuel Asamoah, Isaac Ampratwum, Amiya Nayak
School of Electrical Engineering and Computer Science
University of Ottawa, Canada

***Abstract*—Through centralized management, the Software-Defined Networking (SDN) paradigm separates the control and data layers of conventional networks and offers greater flexibility and scalability. However, the Controller Placement Problem (CPP) is a significant issue in SDN since it directly affects the network's effectiveness and performance. The CPP makes an effort to establish the optimal number of controllers for each network as well as their relative placement. This normally reduces communication lags between switches and controllers and upholds the robustness and dependability of the network. In this research, we provide a heuristic that effectively solves the CPP using the Genetic Algorithm (GA) technique. With the help of the GA, we are able to find the optimal controller placement correlation based on critical parameters like network delay, dependability, and availability.**

***Index Terms*—Controller Placement Problem (CPP), Software-Defined Networking (SDN), Genetic Algorithm (GA)**

I. Introduction

To obtain almost optimal controller placement in relation to particular specified or adaptive criteria, the CPP is very important in the SDN paradigm. Several of these requirements act as limitations required for optimal network performance. Previous studies have looked at individual criteria in an effort to alleviate the CPP in SDN. Our research strategy, in contrast, takes into account some of the most important aspects of any network, such as network delay, availability, and resilience. Considering the trade-offs between the numerous factors at play, we approach this already complex topic as a multi-objective optimization problem. By approaching the CPP as a multi-objective issue, we seek solutions that simultaneously optimize several goals, leading to a more thorough and effective network design.

In this work, we employ the Genetic Algorithm which is a heuristic method that simulates natural selection and genetics. The researchers in [1] define the GA as a stochastic search method that operates on populations of potential solutions. With strict adherence to the switch to controller and controller to controller latency requirements, our approach is designed to essentially forecast the number and location of controllers that will be most practical. Additionally, it incorporates computations of set threshold availability within the control plane and between traffic forwarding nodes and their controllers. We have successfully applied the genetic algorithm to handle such multi-objective situations and find the Pareto front of the issue [2].

In this paper, we use a GA-based aproach to predict controller placement according to a number of requirements. The contributions of this paper are summarized as follows:

- Our controller placement strategy takes network latency limits into account. While guaranteeing that each switch has access to at least two controllers for robustness, it seeks to reduce the network's overall latency.
- In order to further improve network availability in the event of controller failure, we establish highly available alternate routing path between controllers.
- With the addition of node-disjoint recovery routes to restore paths, we present a cost-based availability augmentation strategy for controller-to-controller communications. If a connection or controller fails, this strategy makes sure the network can quickly recover.
- We test our proposed strategy using an experiment on a benchmark network from the Survivable Network Design library [3], comparing the results to those of some existing methods.

II. Related Work

Using practical guidelines to swiftly arrive to approximations of answers is part of the heuristic approach, which is particularly useful for tackling challenging or significant problems. Shortcuts are used in this problem-solving method to produce virtually perfect solutions in a constrained amount of time [4]. Particle Swarm Optimization (PSO) and the Firefly algorithm were employed by the authors in [5] to put the optimal controller. These population-based meta-heuristic algorithms accept a set of goal functions and return the position that meets those objectives the best. The researchers in [6] offered a multi-critical method comparison of solution techniques using different swarm optimization algorithms. The methods utilized identify where the controllers should be located and how switches should be distributed, as well as the approximate optimal number of controllers required to service an SDN. In order to address the CPP and associated load balancing issue in the SDN control plane, the authors in [7] offer a novel strategy that reduces the load on the central element while keeping the maximum distance constraint between controllers.

The concept of **R**eliable **C**ontroller **P**lacement (RCP) in SDN was proposed by researchers in [8]. The authors described this as a method for increasing control plane availability and safeguarding it against single link and node failures by continuously offering fail-over backup control pathways

using resilient routing concepts. The authors provided two methodologies and used Mixed Integer Linear Programming (MILP). The initial approach focused on the delay from switch to controller (SC) and made the assumption that switches and controllers must be linked by two distinct control paths. RCP-DCP, or RCP-disjoint control path, was the name given to this. Switches were to be connected to two Different Controller Replicas (RCP-DCR) via two different or discontinuous paths according to the second technique. The RCP-DCP and RCP-DCR techniques were created with the goal of enabling quick and effective failover with little disruption. Both RCP-DCP and RCP-DCR provided the same performance in the event of link failures with an applicable strategy based on topological features, characteristics, and controller count. RCP-DCP fared better for node failures, protecting controllers from failures.

III. Preliminaries

Unquestionably, the CPP is a crucial component in SDN network design. Delays between nodes (switches) and the appropriate controllers as well as delays within the controllers themselves are the main factors limiting the best controller placement. Numerous other factors are present that are unique to particular networks or predetermined criteria.

The CPP is acknowledged as being NP-hard [9]. As a result, the issue is quite complicated, and we suggest a heuristic method to identify rather precise predictions of controller placement. We approach this optimization problem by dealing with and solving the following issues head-on:

1) controller placement designed to meet delay constraints;
2) to increase availability, shortest links are extracted from a sub-graph using the Steiner tree idea, connecting every node in the set of controllers;
3) selecting a highly reliable alternate routing method between controllers;
4) providing redundancy by giving each node two controllers, and
5) cost of node-disjoint recovery pathways (restore paths) for controller-to-controller connections that improve availability.

This issue has several objectives by nature. To overcome the aforementioned difficulty, we base our methodology on the genetic algorithm approach.

A. CPP initial problem formulation

We model the Software-Defined Networking (SDN) data plane as a graph $G = (N, E)$, where N represents the set of nodes (switches), and E represents the set of links. Each link is denoted by its end nodes $\{i, j\}$. As in [10], we also assume that the delay between two nodes, represented by d_{ij}, is assumed to be proportional to the shortest path between the two nodes.

Here, we discuss two delay constraints in such scenario:

1) **Switch-to-controller delay (D_{sc})**: Each switch's delay from the controller that controls it cannot be longer than a specified maximum value D_{sc}. To ensure effective communication and control between the switches and their accompanying controllers, this constraint is required.
2) **Controller-to-controller delay (D_{cc})**: Any two controllers' delay cannot be greater than a specified maximum value D_{cc}. This constraint guarantees that the controllers can successfully communicate with one another for synchronization and coordination needs.

It is assumed that communication among controllers happens less frequently than communication among the switches they control. Therefore, D_{sc} is expected to be smaller than D_{cc} ($D_{\text{sc}} < D_{\text{cc}}$) [11]. This suggests that decreasing the switch-to-controller latency while yet keeping a respectable controller-to-controller delay should be given top priority when placing controllers.

B. Inter-controller primary path sub-graph - Steiner Tree

A possible sub-graph is inferred from a controller placement solution, resulting in a real-time connection of the shortest routes between all controllers. As in [12], we use a Steiner graph to represent our sub-network and assume that its terminal nodes are the controllers. When connecting many SDN controllers, the Steiner tree is used to make sure that their major paths are on the least expensive links in order to achieve the greatest inter-controller delay value D_{cc}. Using the Steiner tree reduces the overall cost of communication between network controllers and enables effective routing.

Our suggested approach establishes workable additional protection paths among the controllers and enables highly available controller backup paths to be developed, both of which further boost inter-controller communication availability. The existence of backup paths ensures that communication between controllers continues in the event that the primary paths encounter problems, cuts, or failures. For every two controllers, a node-disjoint backup route is computed. As a result, the backup path does not have to use the Steiner tree and shares no nodes with the primary path other than the source and destination controllers.

The network is better prepared to manage breakdowns or performance issues by incorporating both primary and node-disjoint backup pathways. This strategy increases the overall fault tolerance and reliability of the SDN control plane, ensuring that controllers can communicate continuously under any conditions. It is considered that the only factor influencing each network link availability value is the link distance. The possibility that a link will be active at any given time can be determined by looking at its availability, which also serves as a reliable indicator. According to the concept of distance-based availability, a link's dependability increases linearly with its physical length.

Our objective is to ensure that the end-to-end availability between any two controllers is at least a given value K, which is five nines **(0.99999)**. If the availability of the primary and backup paths is less than five nines, compute and assess additional restore paths for that particular controller-to-controller link [13]. Path redundancy is a crucial strategy for improving end-to-end availability.

This ensures that the overall availability between any two controllers meets or surpasses the desired criterion of five nines by identifying and incorporating additional redundant paths that are node-disjoint.

C. Switch to Controller Connection

Our approach primarily focuses on locating and assigning to the closest controller for every given switch, which requires satisfying the minimal delay D_{sc} condition. In addition, our secondary goal is to ensure an average of four nines (0.9999) for all switch-controller connections.

In order to achieve this, we identify a second controller for each switch in addition to its primary controller that is located within a range that will result in an average switch-to-controller availability of four nines throughout the network. This approach ensures that even if the primary controller fails, the switch can maintain its connection with a secondary controller, maintaining the desired level of availability in the network. Our approach attempts to increase overall network performance and resilience by maximizing switch-controller connections, prioritizing the closest controller, and guaranteeing a secondary controller within the specified range.

IV. Methodology

As an adaptive technique for resolving challenging issues and problems, genetic algorithms (GA) are increasingly frequently employed in engineering teaching and learning. To resolve the optimization issue that corresponds to the CPP in SDN, we make use of the GA. Our GA starts off in the context of the CPP by initializing a population of potential controller placement solutions, which are symbolized as chromosomes. The number and locations of the controllers in the network topology are encoded on each chromosome. Based on specified parameters, the fitness function assesses the quality of each solution. By using selection, crossover, and mutation operations, the GA refines the population iteratively. The crossover and mutation operators create new offspring solutions by mixing and perturbing the genetic material of the parent chromosomes, while the selection process favors solutions that are more fit. The GA moves closer to an ideal or nearly ideal solution for the CPP by utilizing and exploring the search space. The final solution provides the best trade-off between the objectives specified. The deployment of controllers in actual SDN networks can be guided by our Pareto frontier, which essentially focuses around minimizing latency or delay restrictions and maximizing controller and node availability. We outline below the steps in our algorithm.

1) The first step of our approach is to consider every conceivable controller count for the network. For instance, we explore the option of 1 to 5 controllers in a network of 5 nodes. The objective, however, is still to determine the bare minimum of controllers required to satisfy the specified restrictions. We initialize a list of solutions accordingly.
2) We create the initial population for the specified number of controllers.
3) We determine whether each potential solution candidate complies with the limitations. Switch-controller delay cannot exceed D_{sc} while inter-controller delay limit is D_{cc} We postulate that the delay between a switch and its managing controller must not surpass a specified maximum threshold $D_{\text{sc}} < D_{\text{cc}}$, given that communication between the controllers and the switches they manage occurs more frequently than inter-controller communication. The constraints are as follows. There should be two controllers for each node, each at a maximum distance from it of D_{sc}. This is the primary controller that a node is connected to. The secondary node is introduced for node-controller reliability. The second controller needs to be placed so that there is at least a four-nine availability between the node and any one of the two controllers. Any two controllers cannot be positioned further than D_{cc} apart.
4) Calculate the fitness of each individual.
5) Through selection, crossover, and mutation, update the GA population.
6) In order to meet the GA halting criterion, repeat Steps 2 through 4 as necessary.
7) Add the resulting solution to the list of solutions for the controllers under consideration.
8) Repeat Step 2 through to 7 for all possible numbers of controllers.
9) Select the smallest group of controllers that satisfies each of the taken into account constraints.

A. Initialization and Population Generation

The development of a population to serve as the controllers for the specified network is considered in the first step of our suggested methodology. Choosing the best network node locations to host controllers is the objective. In the initial population, each chromosome corresponds to a functional controller placement scheme. A genetic technique with binary coding is used to achieve this. One technique of controller selection is represented by each chromosome, which is represented as a binary array with a length equal to the number of network nodes (each bit represents a network node). As opposed to zeros, which represent ordinary nodes, bits in the array with a value of 1 show that the corresponding node has been assigned as a controller. The fitness of each chromosome is calculated once a random population is produced. Our fitness function is as follows:

$$\text{Fitness} = (\alpha \times N_{\text{restore paths}} + \beta \times \bar{D})$$

where α and β are weighting factors that determine the relative importance of the number of restore paths and the average delay in the optimization. Since the objective is to minimize the consumption of network link resources, we provide α a greater value. Restore paths consume network link resources. According to $N_{\text{restore paths}}$, the quantity of restore paths necessary to guarantee a specific level of connection availability for controller to controller connections, and $\bar{D}$,

the typical communication delay between network nodes (S-C and C-C), these numbers are used.

B. Selection, Crossover, Mutation and Termination

The Roulette-Wheel method [14] is used to choose individuals, with the likelihood of selection rising as the fitness value of each chromosome falls. We produced new offspring from the chosen parents using the single-point crossover. Afterwards, mutation is carried out to augment offspring. We gave each gene on a chromosome a mutation chance in order to perform the mutation over a person. The mutation rate is then contrasted with each gene probability. We swap the gene if the likelihood of a mutation is lower than the rate. A chromosome's controller placements are always regulated to correspond to the controller numbers being taken into account for the iteration. We use a mutation probability of 0.1. A genetic algorithm's termination condition is either after it completes a predetermined number of iterations or when the population does not continue to improve. Our termination condition is set at 100 iterations.

V. Experimentation

A. Dataset & Test Network

We conduct tests and simulations on a few networks from the Survivable Network Design library in order to evaluate and validate the functionality and performance of our heuristic GA-based model. A well-established library for designing resilient fixed telecommunications networks is SNDlib [3].

The $Germany50$ network shown in Figure 1 is one of many networks offered by genuine network service providers and original equipment manufacturers (OEMs), which have been built as benchmark networks in comprehensive research initiatives.

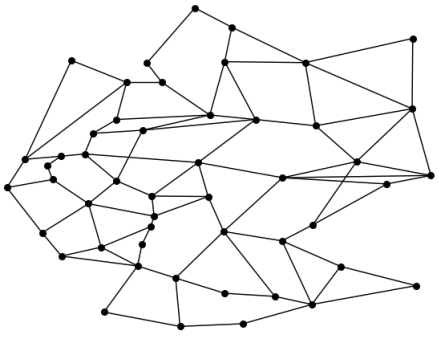

Fig. 1. Topology of Germany50

B. Evaluation Measures

We assume that the high threshold delay constraints D_{sc} and D_{cc} are expressed as a percentage of the graph diameter D_g (longest shortest path between two nodes) for each network [11]. This foundation led us to conclude that the maximum switch-to-controller (SC) delay value for the German50 network is $D_{\text{sc}} = 35\%$ for the initial iteration, followed by 40%, 45% and 50%. For each round of experiments, the maximum controller-to-controller (CC) delay was estimated to be $D_{\text{cc}} = 70\%$, 75% and then 80%. By starting with the smallest value that meets the specified delay conditions, our technique for determining the ideal number of controllers (C) is able to accomplish the desired results. Then, in order to keep our network availability costs as low as possible, we gradually increase C. However, it is important to think about any potential disadvantages of having more controllers. Increased inter-controller communication overhead can have a detrimental effect on control plane performance and negate the entire point of our CPP. This can happen when there are a significant number more controllers. Our model gave extremely good results, and we only sometimes had to pay to build a controller-to-controller restore path or raise C to reach our availability requirement. To help network operators analyze the advantages and disadvantages of every potential solution, we show how the trade-off between the number of controllers and the related availability costs might be achieved. Therefore, maintaining optimal control plane operations necessitates finding the right balance between controller count and network performance.

Both inter-controller communication and communication between the data plane and control plane have precise needed effective average availability values. The goal was to maintain the delay constraints while achieving an effective average availability value of $\lambda_{cc} = \mathbf{0.99999}$ and $\lambda_{sc} = \mathbf{0.9999}$ across the entire network. By following these guidelines, we want to keep the network highly reliable and resilient even in the event of potential link or node outages.

VI. Results and Analysis

A. Evaluation Results

We used Python and Networkx module [15] to implement our working model. Both the Google Colab platform and the Jupyter notebook were used for our simulation. Table I shows the evaluation result. The table showcases the outcomes obtained for various combinations of D_{sc} and D_{cc} values. The number of controllers is displayed in column 'C', starting at the lowest permissible value that strives to satisfy all required requirements and, in some cases, ending at a more ideal, least expensive higher value. The '$RestorePaths$' column merely lists the quantity of expensive extra restore C-C paths required to satisfy the specified threshold availability requirement. The last step is to further segregate the S-C and C-C columns into three separate columns, each of which describes the lowest, maximum, and average computed availability for both communications. When a '-' symbol appears in these columns, it means that the instance is either infeasible, which means that a suitable set of controllers cannot be found for the specified maximum delay values, or that the heuristic failed to find a workable solution that satisfies the constraints. Nevertheless, a solution might exist.

B. Analysis

As depicted in Figure 1, we tested the 50-node Germany50 network. We found the largest optimal number of controllers for the network when testing at a minimum S-C delay of 35%.

TABLE I
RESULTS FOR GERMANY50 NETWORK

D_{sc}	D_{cc}	C	$RestorePaths$	S-C			C-C		
				Min A	**Max A**	**Av A**	**Min A**	**Max A**	**Av A**
3*35%	70%	4	2	0.99991	0.99999	**0.9999**7	0.999993	0.999999	**0.999999**6
	75%	4	2	0.99991	0.99999	**0.9999**7	0.999993	0.999999	**0.999999**6
	80%	3	2	0.99989	0.99999	**0.9999**6	0.999993	0.999999	**0.999999**6
3* 40%	70%	3	1	0.99991	0.99999	**0.9999**6	0.999994	0.999997	**0.999995**
	75%	3	0	0.99991	0.99999	**0.9999**6	0.999994	0.999997	**0.999995**
	80%	3	0	0.99991	0.99999	**0.9999**6	0.999994	0.999997	**0.999995**
3* 45%	70%	2	0	0.99949	0.99995	**0.9997**8	0.999993	0.999993	**0.999997**
		3	0	0.99988	0.99999	**0.9999**4	0.999999	0.999999	**0.999999**
	75%	2	0	0.99949	0.99995	**0.9997**8	0.999993	0.999993	**0.999994**
		3	0	0.99988	0.99999	**0.9999**4	0.999999	0.999999	**0.999999**
	80%	2	0	0.99949	0.99995	**0.9997**8	0.999993	0.999993	**0.999996**
		3	0	0.99988	0.99999	**0.9999**4	0.999999	0.999999	**0.999999**
3* 50%	70%	2	0	0.99959	0.99999	**0.9998**8	0.999998	0.999999	**0.999999**
		3	0	0.99981	0.99999	**0.9999**1	0.999999	0.999999	**0.999999**
	75%	2	0	0.99959	0.99999	**0.9998**8	0.999998	0.999998	**0.999998**
		3	0	0.99981	0.99999	**0.9999**1	0.999999	0.999999	**0.999999**
	80%	2	0	0.99959	0.99999	**0.9998**8	0.999998	0.999998	**0.999998**
		3	0	0.99981	0.99999	**0.9999**1	0.999999	0.999999	**0.999999**

It cost the restoration of two paths in order to comply with all of our network limits. Figure 2 shows a visual of this. With D_{sc} = 40%, only the test case with corresponding D_{cc} = 70% incurred a cost of restoring 1 path, otherwise requirements were met at zero cost. The Germany50 network highlights the notion that too many controllers in any network might defeat the CPP objectives in SDN. In order to achieve the needed five nines availability, a densely connected control plane created by a high number of controllers may require restore path costs. To achieve our four nines goal for S-C connections, a slight increase in the number of controllers at D_{sc} = 45% and 50% was also required. All other constraints were satisfied with the initial computed minimum.

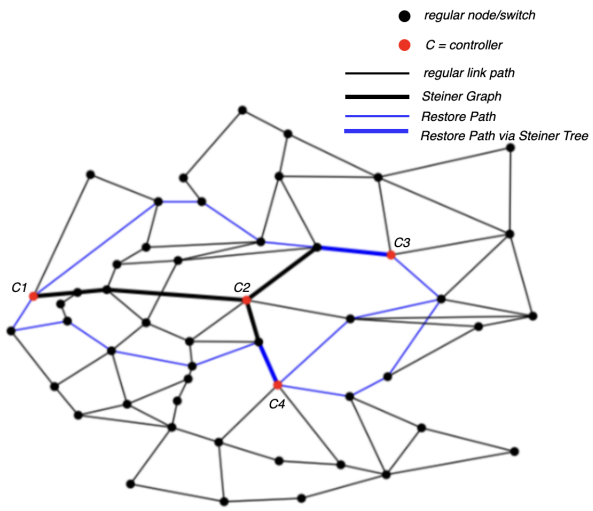

Fig. 2. Germany50 Network with Steiner Graph shown in solid lines for C=4, D_{sc} = 35%, D_{cc} = 75% and $Restorepaths$ = 2; shown in **BLUE**

C. Comparison

In order to determine the benefits of our methodology, we compare our work and results to those of related studies in this section. Santos et al. in both [16] and [12] and the work by [8] are such. The strategy in [16] treats the inter-controller availability and geodiversity guarantees of the controller placement problem as two separate bi-objective joint optimization problems. The geodiversity constraints boost resistance to disaster-related failures, while the availability assurances point to a subgraph with links that can be upgraded at a cost. In [16], the authors took into account the issue of controller placement within delay bounds while assuming improved availability subgraphs between the controllers through upgrades at predetermined costs. They used another approach and performed simulations on networks that were similar.

1) Runtime: Our research on the Germany50 network, which has 50 nodes, can be compared to the research cited in [8]. Because both of these papers place a strong emphasis on control path availability and reliability, we chose this one to be our comparative benchmark. We contrast our results with the RCP-DCP model. Despite taking into account other restrictions like the overall network delay, our execution time is still quite competitive. Our approach calculates an initial controller count of two for test cases where D_{sc} is set at 45 and 50 percent, and it also determines an incredibly quick average

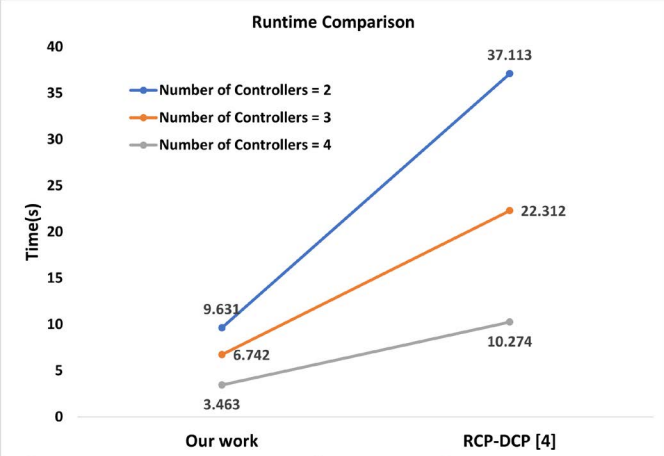

Fig. 3. Runtime comparison - Germany50

runtime of 9.631 seconds. This contrasts with the RCP-DCP model's runtime of 37.113 seconds. With a controller count of three, we further reduce the time to 6.742 seconds from 22.312 seconds to meet S-C availability goal. The controller count of four, determined for D_sc set at 35%, is also reached after 3.463 seconds, as opposed to the 10.274 seconds required by RCP-DCP. Longer durations are required for RCP-DCP because, despite the model's similarity to the other model in [8], there is an additional requirement that the identities of the primary and backup controllers be the same.

2) Availability vs Number of controllers: We contrast the findings from our technique with those from [16] and the two approaches, RCP-DCP and RCP-DCR, from [8].

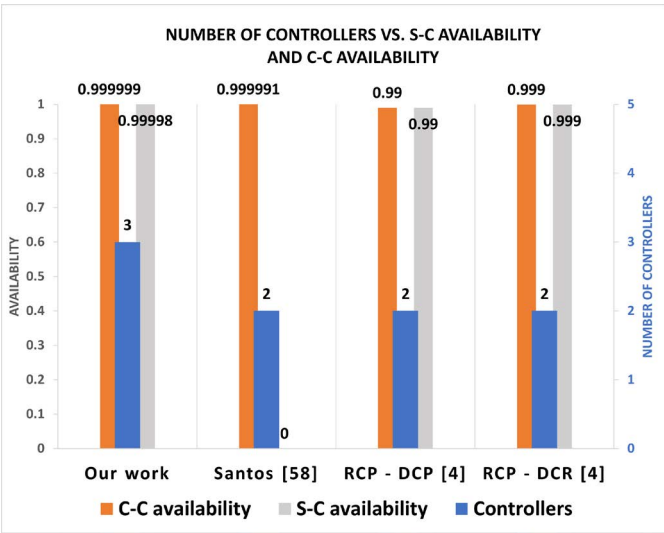

Fig. 4. Availability comparison

In the similar situation, where D_{sc} = 40% and D_{cc} = 70%, all compared articles agree that two controllers are the ideal number. Without taking into account the switch-to-controller (S-C) availability, the authors of [16] achieve a controller-to-controller (C-C) availability of five nines. Additionally, for both C-C and S-C availability, RCP-DCP and RCP-DCR techniques are only able to achieve two and three nines, respectively. By increasing the number of controllers to three and introducing the necessity of a trade-off, our strategy enables the accomplishment of five nines in C-C availability and four nines in S-C availability, thereby promoting long-term overall availability and cost effectiveness.

VII. Conclusion

The goal of this work was to significantly improve the switch to controller and controller to controller link availability while also optimizing the control plane design by placing the controller where there will be the least amount of delay. We outline an improved genetic algorithm methodology and show a precise way to maximize the availability limitations that already exist. To evaluate the trade-off between the deployment of controllers and the price of raising certain node link availabilities, we experimented on a 50-node benchmark topology.

References

[1] J. Shapiro, *Genetic Algorithms in Machine Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[2] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," Tech. Rep. [Online]. Available: http://www3.informatik.uni-

[3] S. Orlowski, M. Pioro, A. Tomaszewski, and R. Wessäly, "Sndlib 1.0 - survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2009.

[4] M. Patnaik and A. M. Adrian, "A perspective depiction of heuristics in virtual reality," in *Cognitive Big Data Intelligence with a Metaheuristic Approach*, 2022, pp. 101–116.

[5] K. S. Sahoo, A. Sarkar, S. K. Mishra, B. Sahoo, D. Puthal, M. S. Obaidat, and B. Sadun, "Metaheuristic solutions for solving controller placement problem in SDN-based WAN architecture," in *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, 2017, pp. 15–23.

[6] A. Vybornova, "A Survey on the Swarm Intelligence Approaches to Controller Placement Problem in the Software Defined Networks Design and Optimization," *Telecom IT*, vol. 8, no. 4, pp. 83–92, 12 2020.

[7] H. Sufiev, Y. Haddad, L. Barenboim, and J. Soler, "Dynamic SDN controller load balancing," *Future Internet*, vol. 11, no. 3, 2019.

[8] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient sdn control plane," in *2016 8th international workshop on resilient networks design and modeling (RNDM)*. IEEE, 2016, pp. 253–259.

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

[10] D. Santos, T. Gomes, and D. Tipper, "Software-Defined Network Design driven by Availability Requirements," in *16th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2020.

[11] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture," in *Proceedings of the 12th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2016, pp. 145–151.

[12] D. Santos, J. P. Vidal, T. Gomes, and L. Martins, "A Heuristic Method for Controller Placement and Enhanced Availability between SDN Controllers," in *Proceedings of the 11th International Conference on Network of the Future (NoF)*, 2020, pp. 82–90.

[13] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proceedings of the 3rd workshop on hot topics in software defined networking*, 2014, pp. 31–36.

[14] M. Melanie, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[15] "NetworkX — NetworkX documentation." [Online]. Available: https://networkx.org/

[16] D. Santos, T. Gomes, L. Martins, and J. P. Vidal, "Resilient sdn intercontroller network design under availability requirements and geodiversity constraints," in *2022 18th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2022, pp. 1–8.