HPSOSA: A Hybrid Approach in Resilient Controller Placement in SDN

Khushboo Kanodia
Dept. of CSE
NIT Rourkela
Odisha, India
kkanodia2307@gmail.com

Sagarika Mohanty
Dept. of CSE
NIT Rourkela
Odisha, India
sagarikam_23@yahoo.com

Bibhudatta Sahoo
Dept. of CSE
NIT Rourkela
Odisha, India
bdsahu@nitrkl.ac.in

Kuldeep Kurroliya
Dept. of CSE
NIT Rourkela
Odisha, India
kuldeep.nitrkl2304@gmail.com

***Abstract*—Software-defined networking (SDN) is an emanate model for network management and design. The significant difference is the decoupling of the data plane from a control plane, which yields flexibility, programmability to configure the network and simplify the network management. SDN gives network administrators the capability to maintain and configure network services from a centralized location. The problem of controller placement concern with the number of controllers required and their placement in the network to preserve the network configuration. The resilient controller placement deals with the issue of either a node failure or link failure. In this paper, we proposed a meta-heuristic hybrid particle swarm optimization and simulated annealing(HPSOSA) approach for resilient controller placement to minimize the average latency. We evaluate our results on different network topologies present in Topology Zoo. Evaluation results verify that our proposed algorithm executes better than the other techniques.**

***Index Terms*—Software-defined networking, Controller placement, Network resilience, Particle Swarm Optimization, Simulated Annealing, Latency**

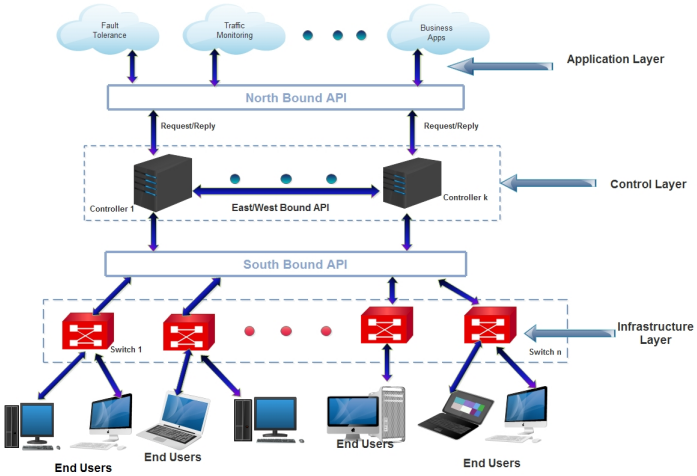

Fig. 1. SDN Architecture

I. Introduction

In the last few years, Software-Defined Network(SDN) has become the new network paradigm that provides a lot of advantages to the network service providers in comparison to the traditional network. The central concept of the Software-Defined Network is the detachment of the control plane from the data plane [1]. One controller in a system is beneficial as it yields centralized management, and the controller does all the routing judgment to establish the perspective of a global network. Yet, this undoubtedly upsurges the latency of switches that are afar from the controller.Also single failure controller is a bottleneck. To avoid this bottleneck, multiple controllers are required.They are physically distributed in the network, but acts as a logically centralized system.

There are three layers in underlying SDN architecture: infrastructure layer, control layer, and application layer as shown in Fig 1.The infrastructure layer consists of various switches and routers, which forward the packet to the controller to take the routing decisions through the southbound API or control layer interface. The control layer is composed of a definite number of controllers that control the network elements in the infrastructure layer according to the need of the application layer. The applications in the application layer communicate to the controller through the northbound API, such as REST. The SDN applications include security, quality of service(QoS), traffic engineering and monitoring etc. The applications are in a programmable format which eliminates the use of middleboxes such as firewall, load balancers because it implements their functionality in software.

The main contributions of the paper are summarized as follows: we proposed a meta-heuristic hybrid particle swarm optimization and simulated annealing(HPSOSA) approach to minimize the average latency in case of a controller node failure while considering the capacity of the controller. To the best of our knowledge, we apply this hybrid approach for the first time.We compared the HPSOSA approach with Particle Swarm Optimization(PSO) and Simulated Annealing(SA) and found that HPSOSA performs better.Using a hybrid method of PSO and SA, search efficiency increases, and it eliminates the weakness of both the algorithm.HPSOSA takes advantage of both PSO and SA as PSO has robust global search ability while SA has robust local search.

The remaining part of the paper is arranged as follows. Section II contains the related works of controller placement and resilient controller placement. Section III discuss the system model. Section IV contains de- tail description of the HPSOSA algorithm. Section V describes the performance evaluation of

the algorithm. In the end, we conclude the paper in Section VI

II. Related Works

In this part, we survey the existing works done on the controller placement in SDN while giving the main focus on the resilient controller placement.

A. Controller Placement

Latency is the primary factor in placing the controller in SDN, which is calculated using the shortest distance between the switches and its assigned controller. [2], [3].Therefore, most of the authors mainly focus on latency while placing the controllers in SDN. In [3], the capacity of the controller is fixed. Their focus is to place the optimal number of the capacitated controller in a network to minimize the latency, which satisfy the demands of the switches in a system.The authors of [4] suggest various evolutionary algorithms to solve a complex optimization problem such as the placement of controller in SDN. Meta-heuristic technique solves the problem in less time as compared to other methods. In [5] authors reviewed various papers of controller placement problem and found that numerous mathematical models and meta-heuristic techniques were used to minimize the latency, cost in a network.The authors of [6] recommended a Bargaining Game theory approach to place the optimal number of controllers in a network. It consider three objective functions to minimize the latency between the switch to a controller, controller to controller, load balancing between the controllers and perform a trade-off between them.

B. Resilient Controller Placement

While placing the optimal number of controllers in a resilient network, it needs a trade off while considering more than one performance metrics such as latency, cost etc. The authors in [7] introduced the resilient Pareto-based Optimal Controller-placement to place the controllers in a network and evaluate their work on 140 topologies and claimed that a minimum of twenty percent nodes should be controller to achieve resilience in a network.The authors of [8] found that the latency of the network highly increases in case of controller failure. To avoid this, they have planned for failure and suggest a mixed-integer linear program (MILP) to solve this problem.The authors of [9] suggest a greedy-placed algorithm to solve a single link failure of a network in polynomial time.The authors of [10] propose a multi-controller mapping approach in which each switch is mapped to various controllers to achieve resilience in case of a controller failure. Minimum fifty percent resilience is maintained in this approach.
In paper [11], the authors proposed a mathematical model to minimize the cost in case of a controller failure and solved it using a GUROBI optimizer in MATLAB.The authors of [12] propose a clique based approach to solve a resilient controller placement in a wide area network by finding the maximal cliques. It is limiting the search space by providing the fact that if an optimal solution exists, it is a subset of either of the maximal cliques. The authors of [13] found that the placement of controllers in a network is NP-hard. Its time complexity increases with the size of the network.To solve this kind of problem meta-heuristic approaches are preferred. The authors proposed a genetic algorithm to place an optimal number of a controller in a network to increase its reliability.The authors of [14] suggest a new framework, sorting moth flame optimization that has taken into account the link utilization and propagation latency while placing the controllers. It also compared its model with another existing approach and found that its proposed one perform better results.

III. Problem Statement and Formulation

A. Problem Description

The controller failures in an SDN can evoke by natural calamity or deliberate attacks. We have to place the optimal number of the controller in a network such that it can handle failures, without degrading its performance too much extend. Connecting switches to more than one controller is one of the methods to achieve resilience. In this, if one of the controllers fails, then switches that are connected to it, can assign to its backup controller. To achieve this, we have to maintain a distinct replication protocol that requires inter-controller synchronization and communication. In this paper, we consider that the network can handle only one controller failure. While assigning switches to the controller, we have considered the load of the switches should be less than the capacity of the controller.

B. Problem Formulation

The SDN network graph is represented as $G(N, E)$, where N represents the sets of nodes, including switches S and controllers C. E represent the sets of edges. Here we assume that all nodes are OpenFlow-enable, so that we can place the controller in any of the nodes. Let P_b be the probable position for deploying the controllers. All switches acquire a load ld_i on their designated controller. Individual controller j is assigned with a capacity of K_j. The preeminent objective of the paper is to deploy m controller in their optimal position to minimize the average latency, so that it will not drastically increase in case of controller failure [15]. In case of failure of the primary controller, the switches will be assigned to its nearest working controller. The objective is defined as follows:
minimize

$$\pi^{avglat} = \min_{Q \subseteq P_b, |Q|=m} \frac{1}{|S|} \sum_{i \in S} \sum_{j \in P_b} \sum_{k \in P_b} \{d(s_i, c_j) + d(c_j, c_k)\} z_{ijk} \tag{1}$$

subject to:

$$\sum_{j \in P_b} t_j = m \tag{2}$$

$$\sum_{j \in P_b} \sum_{\substack{k \in P_b \\ k \neq i,j \\ d(s_i,c_j) \leq d(s_i,c_k)}} z_{ijk} = 1 \quad \forall i \in S \tag{3}$$

$$z_{ijk} \in \{0,1\} \quad \forall i \in S, j,k \in P_b \tag{4}$$

$$\sum_{i\in S}\sum_{k\in P_b} ld_i z_{ijk} + \sum_{i\in S}\sum_{k\in S} ld_i z_{ikj} \leq K_j t_j \quad \forall j \in P_b \tag{5}$$

In the above equation 1, $d(s_i, c_j)$ is the shortest distance between switch s_i to controller c_j in a network. z_{ijk} and t_j are the decision variables takes value 0 or 1. If controller j is connected with switch i and controller j is connected to backup controller k than z_{ijk} is equal to 1 else 0. Likewise if controller is deployed at jth location than t_j is equal to 1 else 0. Equation 2 states that the total of controllers should be equal to m.The load of switches shouldn't exceed the capacity of controller as describe in equation 5.

IV. Algorithms Description

A. Particle Swarm Optimization

Particle Swarm Optimization is referring to a group of birds moving in the sky in search of food. They try to upgrade their position based on other associates and follow the better ones. This way, they optimize their search space. In PSO, we randomly generate initial solutions or particles.We upgrade the velocity and position of the particles at every iteration, based on the local and global best of the particles. Local best is the current best position among the particles, and the global best is the best position obtained so far. Each particle represents the solution space of the n-dimension.The mathematical study illustrated below [13]
Let m particles are there, and each particle has n dimension. Let the current position of the particle is $A_i = [a_{i1}, a_{i2}, a_{i3}, ...a_{in}]$, where $i \in 1, 2, ...m$. The current velocity is $B_i = [b_{i1}, b_{i2}, ...b_{in}]$. The best position of the particle $P_i\, is\, p_{i1}, p_{i2}, ..., p_{in}$. The P_{gb} is best global position.At the time $t+1$, it updates the position $(A)_{in}^{t+1}$ and velocity $(B)_{in}^{t+1}$ of the particle is defined as:

$$B_{in}^{t+1} = wB_{in}^{t} + c1r1(P_{in} - (A)_{in}^{t}) + c2r2(P_{gb} - (A)_{in}^{t}) \tag{6}$$

and

$$A_{in}^{t+1} = (A)_{in}^{t} + B_{in}^{t+1} \tag{7}$$

Where w represents initial weight, r1 and r2 denote the random sequence $[r1, r2 \in 0, 1]$. The constants c1 and c2 adjust the cognitive part and social part, respectively.The representation of the particle according to our problem is shown in Fig 2. Here we consider a network of 10 nodes and assume number of controllers(n) placed in a network is 4 $(n = 4)$.Let population $size = 3$.

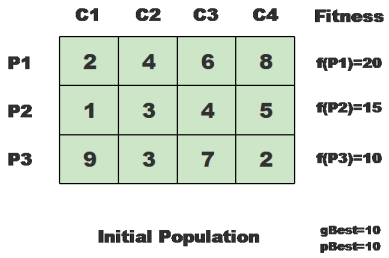

Fig. 2. Representation of Initial Population of Particles

At every iteration, we update the velocity and position of the particles using Equation 6 and 7 and calculate the global best P_{gb} and local best P_{in}.

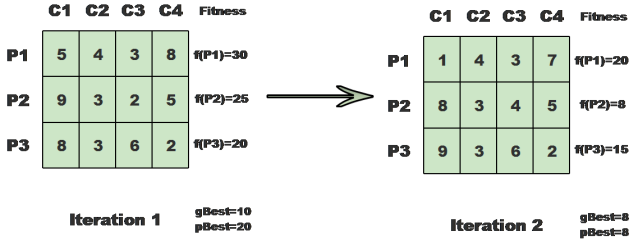

Fig. 3. Representation of Updated Position of Particles

B. Simulated Annealing

Simulated Annealing is a probabilistic method for the optimization problem. It is inspired by the metropolis algorithm that slowly lowers the temperature to cool down the material. The distinctive characteristic of this algorithm is to avoid being caught in local optimum. It accepts the worst solution with some probability. As the temperature decreases, the likelihood of taking the worst solution decreases. We have to be very cautious while selecting the starting and ending temperature, not too high or low. We take the initial temperature $T_s$to be 1 and ending T_e be 0.001. Initially, we choose a random solution and calculate its fitness value.At every iteration, we generate the neighborhood state of the random solution and calculate its fitness. The algorithm store the best value obtained until now. The later solution is accepted as the present solution with the probability, $P(S', \delta, Temp)$. Where δ is the change in fitness value, $i.e.$, the asymmetry between objectives of current and new solutions, and r is a random number lies between 0 and 1.

$$P(S', \delta, Temp) = e(-\frac{\delta}{Temp}) = e(-\frac{v' - v}{Temp}) \geq r \tag{8}$$

C. Hybrid PSO & SA

In spite of having many advantages and fast convergence, particle swarm optimization sometimes trapped in local optima. We use a simulated annealing approach to avoid being trapped in local optima. Using a hybrid method of PSO and SA, search efficiency increases, and it eliminates the weakness

of both the algorithm. Computational cost increases if we apply SA at each iteration, therefore we use SA to PSO at every kth iteration if we don't see any change in the global solution. SA has robust local search ability, while PSO has robust global search ability. By combining both, we take advantage of both the algorithm.

Algorithm 1 HPSOSA algorithm

Input : stMat,imax,nCont
Output : optCost(gBest), final pos of controllers (finalPos)

c1,c2,w,nPop,k
$cpt \leftarrow 0$
$pBest \leftarrow \infty$
for $i \leftarrow 1$ to nPop **do**
  $Ppos \leftarrow RandomPosition(nCont, nNode)$
  $pVel \leftarrow 0$
  $pCost = Fitness(stMat(S,E), Ppos)$
  if pCost<pBest **then**
    $pBest \leftarrow pCost$
    $Lpos \leftarrow Ppos$
  end if
end for
$gBest \leftarrow pBest$
$constgBest \leftarrow gBest$
$finalPos \leftarrow Lpos$
$pBest \leftarrow \infty$
for $ite \leftarrow 1$ to imax **do**
  for $i \leftarrow$ 1to nPop **do**
    $pVel \leftarrow Calculate\, velocity\, using\, Equation\, 6$
    $Ppos \leftarrow Calculate\, position\, using\, Equation\, 7$
    $pCost \leftarrow Fitness(stMat(S,E)), Ppos)$
    if pCost<pBest **then**
      $pBest \leftarrow pCost$
      $Lpos \leftarrow Ppos$
      if pBest<gBest **then**
        $gBest \leftarrow pBest$
        $finalPos \leftarrow Ppos$
      end if
    end if
  end for
  if gBest<constgBest **then**
    $constgBest \leftarrow gBest$
    $cpt \leftarrow 0$
  else
    $cpt \leftarrow cpt + 1$
  end if
  if cpt=k **then**
    $cpt \leftarrow 0$
    $gBest, finalPos = SA(stMat(S,E)), finalPos)$
  end if
end for
return gBest,finalPos

Algorithm 2 Fitness

Input : stMat(S,E),pCtr(Position of controllers)
Output : avglatency(minimum average case latency)

n(number of nodes), m(number of controllers)
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to m **do**
    $lat[i] \leftarrow Calculate\, the\, value\, using\, Equation\, 1$
  end for
end for
$avglatency \leftarrow min(lat)$
return avglatency

Algorithm 3 SA

Input : stMat,Ts,Te,maxite,α, Ppos
Output : sstar(final pos of controller),vstar(best value)

$Temp \leftarrow Ts$, $vstar \leftarrow \infty$ $ite \leftarrow 1$
$s \leftarrow Ppos$
$v \leftarrow Fitness(stMat, s)$
while Temp $\geq$ Te **do**
  if v $<$ vstar **then**
    $sstar \leftarrow s$
    $vstar \leftarrow v$
  end if
  $s' \leftarrow GenerateRandomNeighbour(s)$
  $v' \leftarrow Fitness(stMat, s')$
  $\Delta \leftarrow v' - v$
  $r \leftarrow Random\, number\, lies\, within\, 0\, and\, 1$
  if $P(s', \delta, Temp) \geq r$ **then**
    $s \leftarrow sstar$
    $v \leftarrow vstar$
  end if
  $ite \leftarrow ite + 1$
  if ite$\geq$maxite **then**
    $Temp \leftarrow Temp \times \alpha$
    $ite \leftarrow 1$
  end if
end while
return sstar, vstar

V. Result and Analysis

For execution, the program is written in PYTHON and runs it on a system having an i5 processor with 8 GB RAM. We took the real network topologies for our experiment from Internet Topology Zoo [16]. It is the repertory of real networks, where the information is stored in graphical format.We calculate the distance between two nodes using haversine formula.We consider three networks for our experiment Surfnet(50 nodes), Forthnet(60 nodes), TataNId(143 nodes). We fix the controller capacity to $7.8*10^6$ packets/second [15]. And the demand for each switch is set to 100-kilo req/s. Therefore one controller can serve 78 switches at a time. Assuming that one controller fails at a time, a minimum of two

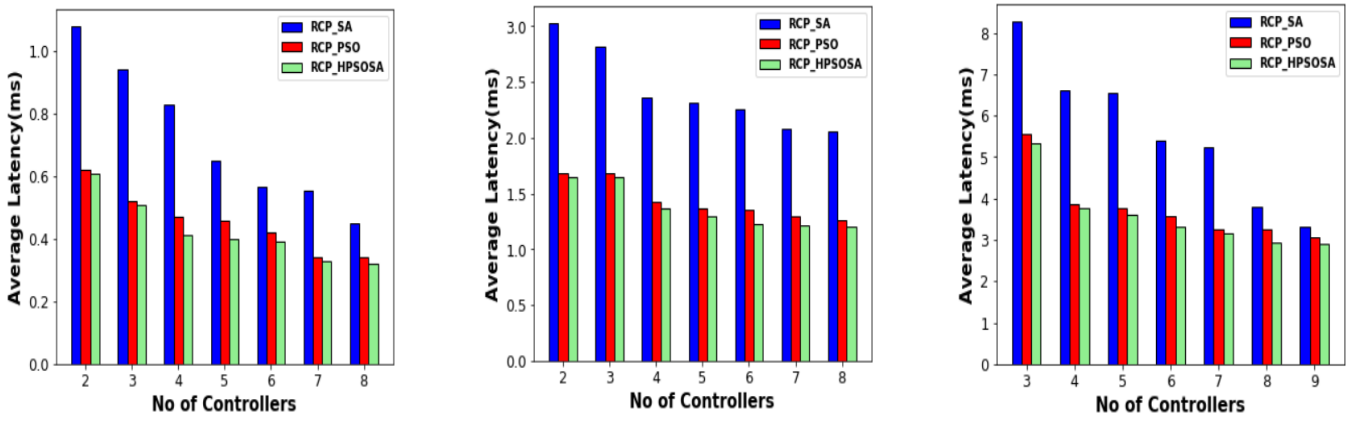

Fig. 4. Impact of average latency on the no of controllers for Surfnet, Forthnet and TataNId network

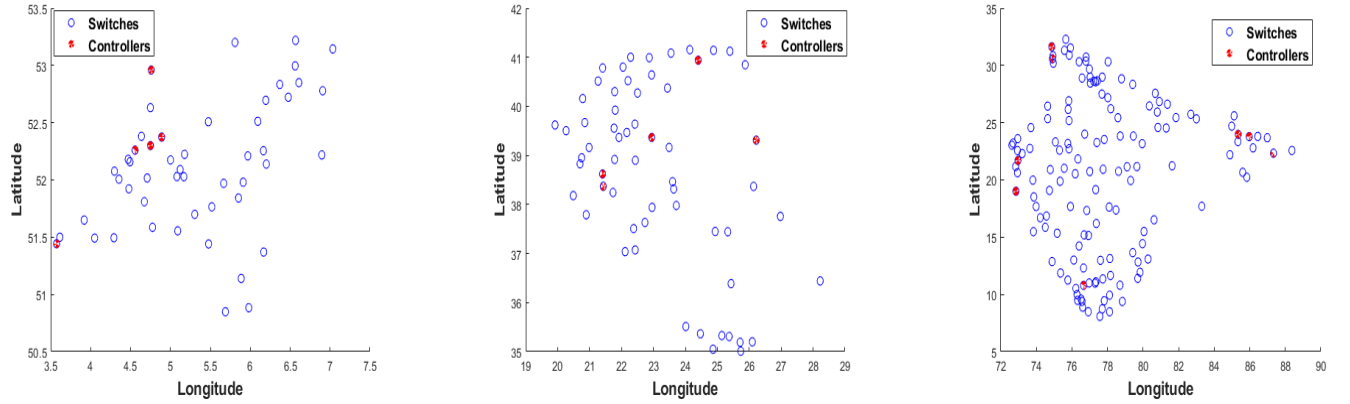

Fig. 5. Controller Locations using HPSOSA for Surfnet, Forthnet and TataNId network

controllers are required for Surfnet and Forthnet network as one controller act as a backup in resilient controller placement. But for TataNId, more than two controllers are needed. We conduct the experiment several times to set the initial parameters of the HPSOSA algorithm. The values are tabulated in Table I.

TABLE I
PARAMETRS USED IN HPSOSA ALGORITHM

Parameters	Description	Value
c1	constant	1.7
c2	constant	2
w	initial weight	0.75
nPop	No of particles	50
imax	No of iteration	100
k	No of iteration	20
T_s	initial temperature	1
T_e	ending temperature	0.001
maxite	No of iteration	200
α	cooling factor	0.9

A. Analysis of the performance of the algorithms

The main goal is to minimize the average latency in case of controller failure. The objective is if a controller fails due to any reason, the latency will not drastically increase.So planning is required for this. From Fig 4, we have seen that the average latency decreases with increasing number of controllers. But after a certain number of controllers, the change is not very significant. We take the controller range from 2 to 9. We evaluated our HPSOSA algorithm on three topologies Surfnet, Forthnet, and TataNId. From the results it is found that HPSOSA performs better than PSO and SA.

B. Optimal location of the controllers

Fig 5. shows the optimal location of the controllers for the three networks. The blue circles represent the switch positions, and the red circles represent the controller positions. After doing experimental analysis, we observed that 3 to 5 controllers are enough for Surfnet and Forthnet. And for TataNId, 6 to 9 controllers. If we add more controllers than either very few switches are assigned to it or the controller will remain idle.

VI. CONCLUSION

In this paper, we proposed a hybrid approach of both particle swarm optimization and simulated annealing(HPSOSA) for the resilient placement of the controller in the SDN. The main

goal is to minimize the average latency of the network in case of controller failure. For evaluation, we took the various network from the Internet Topology Zoo and found that our proposed algorithm performs better than PSO and SA. Our proposed HPSOSA algorithm takes advantage of both PSO and SA methods to find the optimal location of controllers while minimizing the latency. For future work, we plan to consider multiple controller failures at a time and will also consider other performance metric such as load balancing.

REFERENCES

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[2] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[3] 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.

[4] V. C. M. L. LIAO Lingxia and L. ChinFeng, "Evolutionary algorithms in software defined networks: Techniques, applications, and issues," 2017.

[5] B. P. R. Killi and S. V. Rao, "Controller placement in software defined networks: A comprehensive survey," *Computer Networks*, vol. 163, p. 106883, 2019.

[6] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of sdn controllers," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[7] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in sdn-based core networks," in *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*. IEEE, 2013, pp. 1–9.

[8] B. P. R. Killi and S. V. Rao, "Controller placement with planning for failures in software defined networks," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2016, pp. 1–6.

[9] S. Guo, S. Yang, Q. Li, and Y. Jiang, "Towards controller placement for robust software-defined networks," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.

[10] B. P. R. Killi and S. V. Rao, "Towards improving resilience of controller placement with minimum backup capacity in software defined networks," *Computer Networks*, vol. 149, pp. 102–114, 2019.

[11] M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[12] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined wans," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.

[13] J.-M. Sanner, Y. Hadjadj-Aoul, M. Ouzzif, and G. Rubino, "An evolutionary controllers' placement algorithm for reliable sdn networks," in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–6.

[14] A. Jalili, M. Keshtgari, and R. Akbari, "A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach," *Soft Computing*, pp. 1–20.

[15] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, 2017.

[16] 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.