CCPGWO:A Meta-Heuristic Strategy for Link Failure Aware Placement of Controller in SDN

Khushboo Kanodia
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
kkanodia2307@gmail.com

Sagarika Mohanty
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
sagarikam_23@yahoo.com

Kuldeep Kurroliya
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
kuldeep.nitrkl2304@gmail.com

Bibhudatta Sahoo
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
bdsahu@nitrkl.ac.in

***Abstract*—The software-defined network is an advance network model that separates the data plane from the control plane. The control plane consists of controllers that act as a brain of the system, which takes an entire routing decision. For a wide-area network single controller is the bottleneck. Therefore, multiple controllers are required. To find the optimal positions of the controller in a network is a challenging task. As latency attains by the switches, rely upon the controller number and their position in a system. Worst-case latency increases rapidly for a link failure, in order to avoid this, planning for failure is required. We proposed a Capacited Controller Placement Grey Wolf Optimization(CCPGWO), a meta-heuristic strategy to search the best position of controllers in a network so that worst-case latency does not increase drastically for link failure.**

***Index Terms*—Software-defined network, Controller Placement, Worst-case Latency, Grey Wolf Optimization**

I. Introduction

The emergence of IoT, cloud, and fog computing and big data gives rise to colossal network traffic. We need a new network architecture which efficiently manages this traffic. Therefore, software-defined network(SDN) comes into existence as it poses the feature of detachment a data plane from the control plane [1]. It is the next-generation architecture of the network.It has three-layers: Application layer, Control layer, and Infrastructure layer, as shown in Fig.1. Switches and routers reside in the infrastructure layer, controllers are present in the control layer. The controller controls all the elements of the infrastructure layer according to the need of the application layer. Switches communicate to the controller through southbound API. The controller communicates with each other through east/westbound API. The applications in the application layer communicate to the controller through the northbound API such as REST. The control layer acts as a logically centralized system that is mange by the controller. The controller takes all the routing judgment to establish the perspective of a global network. Also, a single failure of controller is the bottleneck. Therefore more than one controller is required.Sometimes the latency will also increase due to link failure which needs a proper planning.

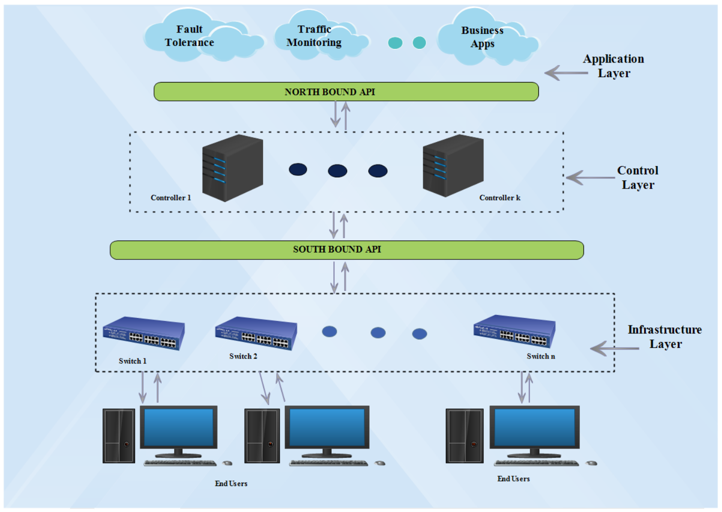

Fig. 1. SDN Architecture

Our contributions are as follows:

- We randomly placed the controllers in a network and assign switches to them based on the shortest distance considering the switches demand is less than or equal to the controller capacity.
- The main goal is to minimize the worst latency of network for single link failure.
- We also consider the weighted worst latency as a metric that consider both the worst latency without failure of a link and with a link failure.
- We proposed a CCPGWO algorithm for the placement of controllers in a network and compared it with the random and greedy algorithm.

The rest of the paper is arranged as follows. Section-II describes the related works of controller placement. Section-III discuss the system model. Section-IV contains a detail description of the algorithms. Section-V describes the results and analysis of the algorithm. Conclusion is done in Section-VI

II. Related Works

In this part, we surveyed existing works done on the placement of the controller in SDN while giving the preeminent focus on the link failure aware controller placement.

Heller et al. [2] first introduced the placement of controller problem by taking into consideration average and worst latency. They have not considered the capacity of controller. Latency is the dominant factor in placing the controller in SDN, [2], [3]. Therefore, most of the authors mainly focus on latency while placing the controllers in SDN. Guo et al. [4] proposed placement of controller strategy using selection and partition scheme. Using the interdependence network, cascading failure are analyzed by them due to switch and link failure.

The authors of [5] recommended a Bargaining Game theory approach to place the optimal controllers in a system. It consider three objective functions: latency minimization between the switch and controller, controller to controller, balancing of load between the controllers and perform a trade-off between them.The authors of [6] suggest a greedy-placed algorithm to solve a single link failure of a network in polynomial time.The authors of [7] suggest a new framework, sorting moth flame optimization that has taken into account the link utilization and propagation latency while placing the controllers.

In [8], a clique based approach is used to solve a resilient placement of controller in a network by finding out the maximal cliques. Killi et al. [9] proposed a multi-controller mapping approach in which switch is assigned to various controllers to achieve resilience in case of a controller failure. Sallahi et al. [10] proposed a controller placement strategy to minimize the cost required in connecting and installing the controllers and connecting controllers and switches.In [11] [12] discover that the placement of a controller in a network is NP-hard. As the size of the network increases, its time complexity increases. To solve such kind of problem meta-heuristic approaches are preferred.

Sampa Sahoo et al. in [13] proposed an learning automata-based and a game theory based scheduling in [14] technique for task scheduling in cloud. Those approach can be applied for controller placement in Software Defined Networks.

III. System Model

A. *Description of Problem*

The link failure in SDN can arouse due to number of reasons such as a wire being cut or unplugged, change of configuration in network, etc. We have to place the optimal number of the controller in a network such that it can handle failures, without degrading its performance. We assume that nodes are attached with multiple links.If one link fails, then it will able to take another path. In this paper, we consider the network can handle a single link failure. While assigning switches to the controller, we have considered the load of the switches should be less than controller capacity.

B. *Formulation*

The graph of SDN is express as $G(V, E)$, where V represents the sets of nodes, along with switches S and controllers C. E represent the sets of links. Assuming all nodes are OpenFlow-enable, so that we can place the controller in any of the nodes.Let P_r be the probable position for deploying the controllers.Let switch i incur a load Ld_i on their designated controller.Individual controller j is assigned with a capacity of U_j. The predominant objective of the paper is to deploy k controllers in their optimal position to minimize the worst-case latency, so that it will not drastically increase for link failure [15].The minimum propagation latency from node i to j is denoted as $\Delta_{i,j}$ and without using the link l with $\Delta^l_{i,j}$. The objective is defined as follows:
minimize

$$\pi^{worstLat} = \underset{\substack{Q \subseteq P_r \\ |Q|=k}}{min} \underset{i \in S}{max} \{ \underset{j \in P_r}{min} \Delta^l_{ij} \} \tag{1}$$

subject to:

$$\sum_{j \in P_r} a_j = k \tag{2}$$

$$b_{ij} \leq a_j \quad \forall i \in S, \quad \forall j \in P_r \tag{3}$$

$$\sum_{i \in S} b_{ij} Ld_i \leq U_j a_j \quad \forall j \in P_r \tag{4}$$

$$a_j \in \{0, 1\} \quad \forall j \in P_r \tag{5}$$

$$b_{ij} \in \{0, 1\} \quad \forall i \in S \quad \forall j \in P_r \tag{6}$$

Equation(2) delimited the number of controllers to k. Equation(3) prevent the switch i to assign to controller j if controller is not deployed at location j. Equation(4) ensures that the summation of the request, which is sent by the switches to controller j is equal to or less than the controller j capacity. Equation(5) and (6) are the decision variables and take the binary value 0 or 1.

The worst-case latency will increase if the link will not fail, as it has previously plan for failure. Therefore, we modified the objective, which deals with both worst-case latencies with and without failure. The revised goal is defined as follows:

$$\pi^{weightedWorstLat} = \underset{\substack{Q \subseteq P_r \\ |Q|=k}}{min} \underset{i \in S}{max} \underset{j \in P_r}{min} \{ \alpha \Delta^l_{ij} + (1-\alpha) \Delta_{ij} \} \tag{7}$$

The alpha(α) is assumed to be the probability of link failure, and its value is 0.1.The description of the notations used in the equations is tabulated in the table I.

TABLE I
DESCRIPTION OF SYMBOLS

Symbol	Description
$G(V,E)$	Graph Representation
V	Nodes including switches and controllers
E	Links
P_r	Probable position for deploying the controllers
Ld_i	Load of switch i
k	No of controllers
U_j	Capacity of controller j
Δ_{ij}	Propagation latency between i^{th} and j^{th} node
Δ_{ij}^{l}	Propagation latency between i^{th} and j^{th} node without using the link l
a_j	=1, if controller placed at position j =0, else
b_{ij}	=1, if s switch i serverd be controller j =0, else

IV. Algorithms Description

Algorithm 1 Fitness

```
Input : $topologyMat(V,E)$, $agentPos$
Output : $worstlat$

1: $n(number\ of\ switches)$, $k(number\ of\ controllers)$
2: for $ii \leftarrow 1$ to n do
3:     for $jj \leftarrow 1$ to k do
4:         $worstlat[ii] \leftarrow Calculate\,the\,value\,using\,Equation\,1$
5:     end for
6: end for
7: $worstlat \leftarrow min(worstlat)$
8: return $worstlat$
```

A. Random Controller Placement Algorithm

In random placement, we randomly select the initial solution and calculate its fitness function. We repeat this for a certain number of iterations and return the best among them. Assuming there are k controllers, which is randomly generated from the the values range from 1 to $numNodes$(total nodes). It is the initial solution. We calculate the fitness value. This is done for the $imax$ number of iterations and returned the optimal value and its position.

Algorithm 2 Random Controller Placement Algorithm

```
Input : $topologyMat(V,E)$, $k$, $imax$
Output : $worstLat$, $bestPos$

1: $worstLat \leftarrow \infty$
2: for $i \leftarrow 1$ to $imax$ do
3:     $pos \leftarrow RandomPosition(k, numNodes)$
4:     $lat \leftarrow Fitness(topologyMat, pos)$
5:     if $lat \leq worstLat$ then
6:         $worstLat \leftarrow lat$
7:         $bestPos \leftarrow pos$
8:     end if
9: end for
10: return $worstLat$, $bestPos$
```

B. Greedy Algorithm

The algorithm pursue a greedy approach to search the best position for controllers. It considers the degree of the node as a metric. It places the controller on a node that has the highest node degree. Assuming we have to set k controllers in a network, then following the greedy approach, the first k nodes which have the highest node degree are set as controllers.After that the objective function is calculated.

Algorithm 3 Greedy Algorithm

```
Input : $topologyMat(V,E)$, $k$
Output : $worstlat$, $bestPos$

1: $bestPos \leftarrow \phi$
2: for $i \leftarrow 1$ to numNodes do
3:     $d[i] \leftarrow nodeDegree(i)$
4: end for
5: $d[numNodes] \leftarrow Sort(d[numNodes])$
6: for $j \leftarrow 1$ to k do
7:     $bestPos \leftarrow bestPos \cup d[j]$
8: end for
9: $worstLat \leftarrow Fitness(topologyMat, bestPos)$
10: return $worstLat$, $bestPos$
```

C. CCPGWO

Grey Wolf Optimization(GWO) is developed by Mirjalili et al. [16] is a swarm-based intelligent technique. It impersonates the leadership hierarchy of wolves, which is famous for their troop hunting. It divides the population of search agents into four types α, β, δ, and ω according to their fitness value.Alpha is the troop leader who is responsible for making decision. Beta helps alpha for making decision. Deltas are subordinate, which dominates omega and reports to alpha and beta. Rest of the wolves are treated as omegas.The preeminent steps of GWO are:

1) Social Hierarchy: To mathematically model the social hierarchy of the grey wolf optimization, we are assuming that their are pSize solutions among them α is the best solution, β is second best, δ is third-best, and ω are the rest solutions. The omegas(ω) has been guided by α, β, δ for hunting(optimization).

2) Searching(exploration): Omegas(ω) update their position according to the position of alpha(α), beta(β), and delta(δ).In search for prey they diverge and converge for attacking.If $|B| > 1$ wolves diverge and find the better solution.

3) Encircling(exploration): The mathematical formulation for encircling behavior are defined as follow:

$$\vec{E} = \mid \vec{D}.\vec{Y_p(i)} - \vec{Y(i)} \mid \tag{8}$$

$$\vec{Y}(i+1) = \vec{Y_p(i)} - \vec{B}.\vec{E} \tag{9}$$

where i is the current iteration, $\vec{B}$ and $\vec{D}$ are coefficient vectors, $\vec{Y_p}$ prey(optimal value) position, and $\vec{Y}$ indicates wolf position.The value of $\vec{B}$ and $\vec{D}$ are as follow:

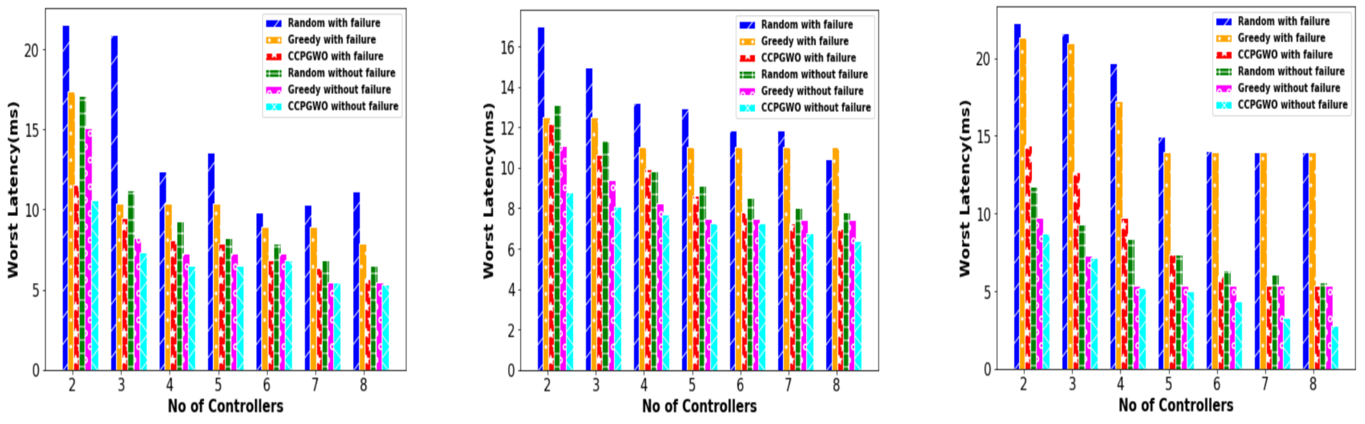

Fig. 2. Impact of worst latency with and without link failure on AT&T, BTNA and Sprint network

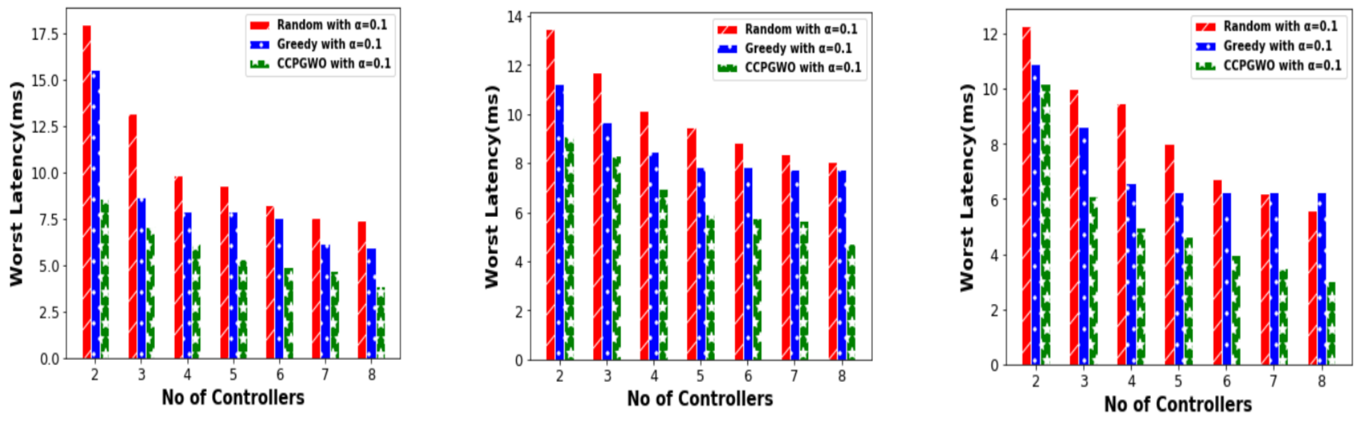

Fig. 3. Weighted worst latency on AT&T, BTNA and Sprint network

$$\vec{B} = 2.\vec{b}.\vec{r_1} - \vec{b} \tag{10}$$

$$\vec{D} = 2.\vec{r_2} \tag{11}$$

where value of $\vec{b}$ decremented from 2 to 0 over the course of iterations(imax) and value of $\vec{r_1}$ and $\vec{r_2}$ lies between 0 and 1.

4) Attacking(exploitation): When prey stops moving, wolves attack him and finish their hunt.In GWO flucuation range of $\vec{B}$ is also decreases when $\vec{b}$ is decremented from 2 to 0 and when $\mid \vec{B} \mid < 1$ the prey is attacked by the wolves or algorithm return the optimal value.

The main advantage of CCPGWO is that it has exceptional exploration and exploitation characteristics,than other meta-heuristic techniques. As CCPGWO follows the hierarchy of leaders, therefore it does not trap in local optima and give the best performance.

Limitations of CCPGWO is that it convergence rate is poor. Its local search ability is not too good.

Algorithm 4 CCPGWO Algorithm

```
Input : $topologyMat,\ imax,\ k,\ pSize$
Output :$bestPos,\ bestVal$

1: Initialize the $b$, $B$ and $D$ using equation (10) and (11)
2: for $i \leftarrow 1$ to $pSize$ do
3:     $agentPos[i] \leftarrow RandomPosition(k, numNodes)$
4:     $agentFit[i] = Fitness(topologyMat, agentPos[i])$
5: end for
6: $Y_{\alpha pos}, Y_{\alpha score} \leftarrow$ the best position and score
7: $Y_{\beta pos}, Y_{\beta score} \leftarrow$ the second best position and score
8: $Y_{\delta pos}, Y_{\delta score} \leftarrow$ the third best position and score
9: while $ite \leq imax$ do
10:    for $i \leftarrow 1$ to pSize do
11:        $agentPos[i] \leftarrow$ Update position using equation (9)
12:        $agentFit[i] = Fitness(topologyMat, agentPos[i])$
13:    end for
14:    Update $b$, $B$ and $D$ using equation (10) and (11)
15:    Update $Y_{\alpha pos}, Y_{\alpha score}$
16:    Update $Y_{\beta pos}, Y_{\beta score}$
17:    Update $Y_{\delta pos}, Y_{\delta score}$
18:    $ite \leftarrow ite + 1$
19: end while
20: return $Y_{\alpha pos}, Y_{\alpha score}$
```

V. Results and Analysis

We used a system having an i7 processor and 16 GB RAM for the execution of the program, which is written in PYTHON. For the experiment, topologies are taken from the Topology Zoo [17]. It is the repertory of real networks, where the information is stored in graphical format.Haversine formula is used to find the distance between two nodes.We consider three networks for our experiment AT&T(25 nodes and 55 links), BTNA(36 nodes and 69 links), Sprint(11 nodes and 17 links). AT&t and BTNA are medium size network and Sprint is small size network. In AT&T and BTNA network, all nodes haves node degree two or more than two. Only in the Sprint network, one node has node degree one except that all have degree two or more than that. We are assuming that each node in a network is connected to more than one node, so if one link fails, we can take another path. We discard the nodes whose node degree is 0 or 1. The controller capacity is fix to $7.8 * 10^6$ packets/second [15]. And the demand for each switch is set to 200-kilo req/s.

The main goal is to minimize the worst latency for single link failure. If we do not plan for link failure, then the worst case latency will drastically increases. Also, it is seen that if we will plan for failure and the link will not fail, then the worst latency is more compared to the worst latency if we do not plan for failure. So to avoid this, we take the weighted worst latency, which takes into account both the cases. The probability (α) of link failure is considered as 0.1.

From Fig.2 and Fig.3, we have seen the decrement of worst latency with the increasing number of controllers. But after a certain number of controllers, the change is not very significant. The controllers' range is taken from 2 to 8. In Fig.2 the effect of worst-case latency on number of controller with and without link failure is shown. It is found that the worst latency does not increase drastically if the link will fail, because we are planning ahead for failure. If the planning is not done previously than worst latency increases drastically. In Fig.3, we take the probability (α) of link failure and shown its effect on the worst latency. Weighted worst latency is taken because if we plan for link failure and placed controller according to that and the link will not fail than worst latency increases, compared to if planning is not done. Therefore both cases is consider by taking the probability(α). We compared our Capacitated Controller Placement Grey Wolf Optimization (CCPGWO) algorithm with the Greedy and Random algorithms and evaluate the results on three topologies AT&T, BTNA, and Sprint. From both the figures Fig.2 and Fig.3, it is found that CPPGWO gives better performance than the Greedy and Random algorithm.

VI. Conclusion

In this paper, we proposed a meta-heuristic strategy CCPGWO for the link failure aware placement of the controller in the SDN. The main goal is to minimize the worst-case latency of the network for link failure. For evaluation, we took various network from the Topology Zoo and found that the proposed CCPGWO algorithm gives better performance over the Random and Greedy algorithm.The random and greedy algorithm trap in local optima. Therefore, they do not give better result compare to CCPGWO. CCPGWO do not catch in local optima and give result nearer to global optima. For future work, we will plan to consider both link and controller failures at a time and will also consider other metrics 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] M. Guo and P. Bhattacharya, "Controller placement for improving resilience of software-defined networks," in *2013 Fourth International Conference on Networking and Distributed Computing*. IEEE, 2013, pp. 23–27.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE communications letters*, vol. 19, no. 1, pp. 30–33, 2014.

[11] 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.

[12] K. S. Sahoo, D. Puthal, M. S. Obaidat, A. Sarkar, S. K. Mishra, and B. Sahoo, "On the placement of controllers in software-defined-wan using meta-heuristic approach," *Journal of Systems and Software*, vol. 145, pp. 180–194, 2018.

[13] S. Sahoo, B. Sahoo, and A. K. Turuk, "A learning automata-based scheduling for deadline sensitive task in the cloud," *IEEE Transactions on Services Computing*, 2019.

[14] M. K. Patra, S. Sahoo, B. Sahoo, and A. K. Turuk, "Game theoretic approach for real-time task scheduling in cloud computing environment," in *2019 International Conference on Information Technology (ICIT)*. IEEE, 2019, pp. 454–459.

[15] B. P. R. Killi and S. V. Rao, "Link failure aware capacitated controller placement in software defined networks," in *2018 International Conference on Information Networking (ICOIN)*. IEEE, 2018, pp. 292–297.

[16] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[17] 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.