Computers, Materials & Continua
DOI:10.32604/cmc.2022.017787
Article

Tech Science Press

Dynamic Routing Optimization Algorithm for Software Defined Networking

Nancy Abbas El-Hefnawy[1,*], Osama Abdel Raouf[2] and Heba Askr[3]

[1]Department of Information Systems, Tanta University, Tanta, 31511, Egypt
[2]Department of Operations Research and Decision Support, Menoufia University, Shepen Alkom, Egypt
[3]Department of Information Systems, University of Sadat City, AlSadat City, 048, Egypt
[*]Corresponding Author: Nancy Abbas El-Hefnawy. Email: nancyabbas_1@ics.tanta.edu.eg
Received: 11 February 2021; Accepted: 23 March 2021

Abstract: Time and space complexity is the most critical problem of the current routing optimization algorithms for Software Defined Networking (SDN). To overcome this complexity, researchers use meta-heuristic techniques inside the routing optimization algorithms in the OpenFlow (OF) based large scale SDNs. This paper proposes a hybrid meta-heuristic algorithm to optimize the dynamic routing problem for the large scale SDNs. Due to the dynamic nature of SDNs, the proposed algorithm uses a mutation operator to overcome the memory-based problem of the ant colony algorithm. Besides, it uses the box-covering method and the k-means clustering method to divide the SDN network to overcome the problem of time and space complexity. The results of the proposed algorithm compared with the results of other similar algorithms and it shows that the proposed algorithm can handle the dynamic network changing, reduce the network congestion, the delay and running times and the packet loss rates.

Keywords: Dynamic routing optimization; Openflow; software defined networking

1 Introduction

Distributed routing algorithms are used in traditional networks and this cause problems in controlling and management of the network. SDN outperforms the traditional network architecture management in terms of cost. SDN separates the network control plane layer from the forwarding/data plane layer. SDN controllers have a full image of the network topology and make forwarding decisions based on flow tables using the OF protocol. SDNs controller have full image and control of the network topology and which improves the performance of routing processes [1].

Time and space complexity is the most critical problem of the current SDN routing optimization algorithms. These algorithms use Dijkstra algorithm in exploring the shortest path. The complexity of the Dijkstra algorithm is that the number of nodes and edges of the network affect the efficiency of the algorithm [2]. To overcome this complexity, researchers use meta-heuristic techniques inside the routing optimization algorithms in OF-based large scale SDNs. [3].

Ant Colony Optimization (ACO) is the most famous meta-heuristic technique that outperforms other traditional routing techniques beside the ACO methodologies have the potential to conduct the flow-based routing strategy as same as SDNs [4].

This paper proposes a hybrid meta-heuristic algorithm to optimize the dynamic routing problem for the large scale SDNs. it is called Hybrid Ant Colony Optimization (HACO) algorithm HACO uses two different methods for dividing the network into small subnets to overcome the problem of time and space complexity. These methods are box-covering and k-means clustering methods. Also, HACO uses a mutation operator to discover new areas in the search space and improve the solution.

The structure of this paper as follows. Section 2 presents the goal of the research. Section 3 covers the related work efforts. Section 4 gives an overview of the SDN. Section 5 presents an overview of the network routing problem. Section 6 presents an overview of Ant Colony Optimization. Section 7 presents the proposed algorithm. Section 8 presents the performance evaluation of the proposed algorithm. Finally, the conclusion of the paper is presented in Section 9.

2 Research Objective

The main goal of this paper is to overcome the problem of time and space complexity of the dynamic routing problem inside SDNs using the proposed HACO algorithm.

3 Related Works

Dijkstra algorithm is one of the most famous shortest path algorithms applied in network routing. But the complexity of the Dijkstra algorithm affects the efficiency of the routing process. Literature [2] proposed a box-covering-based routing (BCR) algorithm for large scale SDNs trying to minimize the time and space complexity of the Dijkstra algorithm by reducing the number of nodes and edges in the network. In the BCR algorithm, Firstly, the whole network is divided into several subnets and each subnet is covered by a box of size (l). Secondly, each subnet is treated as a new node, and the shortest path between these new nodes is calculated by the Dijkstra algorithm. Thirdly, the shortest path between nodes inside each subnet is calculated also by the Dijkstra algorithm. Finally, the shortest path between subnets and the shortest paths inside those corresponding subnets are linked together and the path from the source node to the target node is found.

Although the BCR algorithm in [2] reduces the network size and it still uses the Dijkstra algorithm in the routing process. This encourages researchers to use meta-heuristic techniques inside the routing optimization algorithms in the OpenFlow (OF) based large scale SDNs.

4 SDN Architecture

The SDN architecture is divided into three planes. At the very bottom is the data plane which comprises of hardware such as network switches. Above the data plane is the network control plane. The centralized controller could be as simple as a server machine attached to the network running on controller software [5]. Residing above the control plane is the application plane. This plane comprises of individual applications which could be network monitoring utilities, voice over IP applications which has a particular set of requirements such as delay. Communication between the application and the control plane is by means of northbound application programming interface (API) such as the restful protocols. While the controller communicates with the data

plane devices such as network switches using the southbound application programming interface commonly using the open flow protocol. This architecture is presented in Fig. 1.

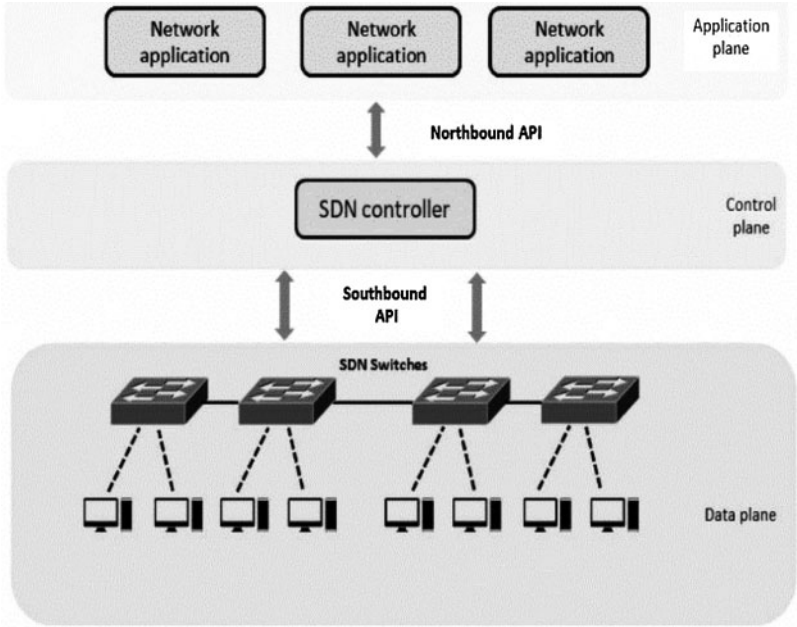

Figure 1: SDN architecture

OpenFlow is the core of the forwarding plane of network devices in SDNs [6]. An OpenFlow Switch consists of one or more flow tables and a group table, which perform packet lookups and forwarding, and an OpenFlow channel to an external controller. The switch communicates with the controller and the controller manages the switch via the OpenFlow protocol [7]. Using the OpenFlow protocol, the controller can add, update, and delete flow entries in flow tables. Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets [8]. The processing of packets always starts at the first flow table. Then proceed with the highest-priority matching flow table and of the instructions of that flow entry is executed. Otherwise, the packet is dropped. The OF switch is illustrated in Fig. 2.

5 Network Routing

Network routing is the process of selecting a path across one or more networks. Metrics are cost values used by routers to determine the best path to a destination network. The most common metric values are hop, bandwidth, delay, reliability, load, and cost [9]. SDN routing example is shown in Fig. 3. The SDN controller has full image and management over the SDN network [10].

Routing algorithms are responsible for selecting the best path for the communication [11]. Open Shortest Path First (OSPF) allows routers to dynamically update information about network topology. Dijkstra's algorithm uses Shortest Path First (SPF) algorithm [12]. It finds the path the shortest path between that node and every other node [13].

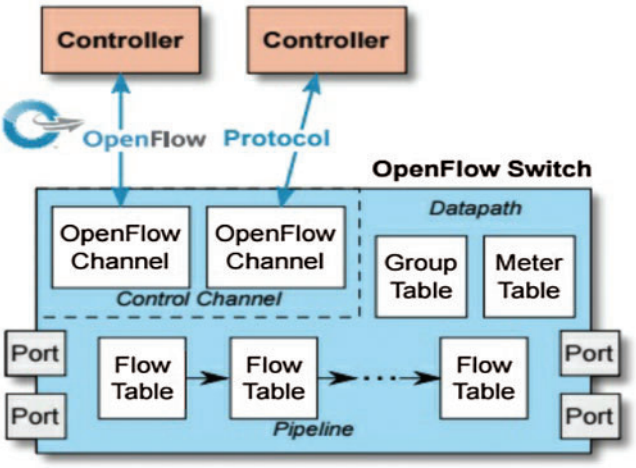

Figure 2: OpenFlow switch components

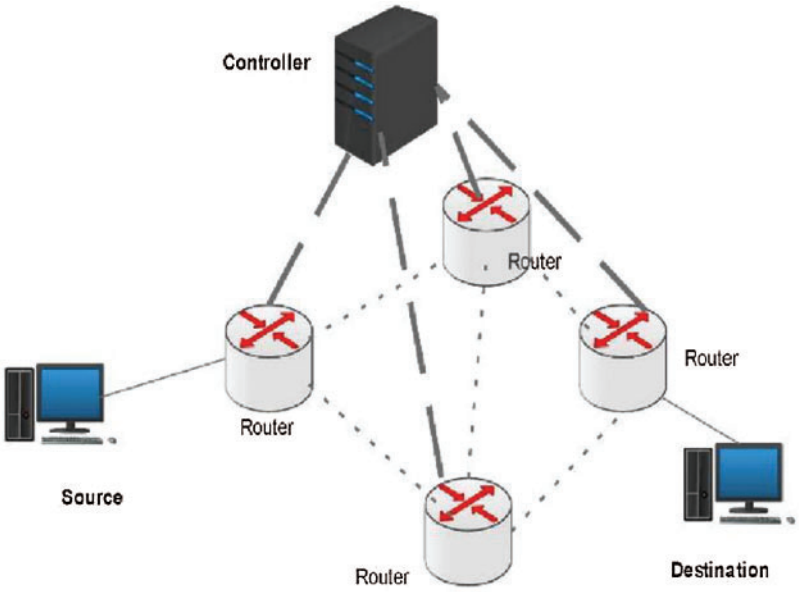

Figure 3: OpenFlow routing structure of SDN

6 Ant Colony Optimization (ACO)

ACO is a meta-heuristic algorithm where ants searching for food and depositing pheromone on the route. The quantity of pheromone on the route affects the behavior of ants; the path with the largest quantity of pheromone represents the shortest path [3].

ACO starts with generating m random ants and evaluates the fitness of each ant according to an objective function then updates the pheromone concentration of every possible trail using Eq. (1).

$$\tau_{ij}(t) = \rho\tau_{ij}(t-1) + \Delta\tau_{ij} \tag{1}$$

where i and j are nodes, t is a particular iteration; $\tau_{ij}(t)$ is the revised pheromone concentration related to the link ℓ_{ij} at iteration t, $\tau_{ij}(t-1)$ is the pheromone concentration at the previous iteration $(t-1)$; $\Delta\tau_{ij}$ is the pheromone concentration change; and ρ is the pheromone evaporation(decay) coefficient with value ranging from 0 to 1 to avoid too strong influence of the old pheromone so that premature solution stagnation is incurred. The decay value equals the average of the windows' size of the network. $\Delta\tau_{ij}$ is the sum of the contributions of all ants related to ℓ_{ij} at iteration t and can be calculated using Eq. (2).

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{2}$$

where m is the number of ants and $\Delta\tau_{ij}^{k}$ is the pheromone concentrate laid on link ℓ_{ij} by ant k. $\Delta\tau_{ij}^{k}$ can be calculated by Eq. (3) with R being the pheromone reward factor and $fitness^{k}$ being the value of the objective function for ant k.

$$\Delta\tau_{ij}^{k} = \sum_{k=1}^{m} \frac{R}{fitness^{k}} \quad if\ \ell_{ij} \text{ is chosen by ant k} \tag{3}$$

Once the pheromone is updated, each ant must update its route respecting the pheromone concentration and also some heuristic preference consistent with the subsequent probability by Eq. (4).

$$pij(k,t) = \frac{\tau_{ij}(t)^{\alpha} \times \eta_{ij}^{\beta}}{\sum_{\ell_{ij}}^{m} \tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}, \tag{4}$$

where $p_{ij}(k,t)$ is the probability that link ℓ_{ij} is chosen by ant k at iteration t; $\tau_{ij}(t)$ is the pheromone concentration related to link ℓ_{ij} at iteration t; η_{ij} is the heuristic factor for preferring among available links and is an indicator of how good it's for ant k to pick link ℓ_{ij}; α and β are exponent parameters that specify the impact of trail and attractiveness, respectively, and take values greater than 0.

7 Optimization of Dynamic Routing in SDN Using ACO

The deployment phases of the SDN environment are presented in this section followed by presenting the proposed algorithm.

SDN Deployment Phases:

Fig. 4 is illustrated SDN deployment phases as follow:

Phase (1) SDN Simulation: SDN is simulated by Mininet with VMware Workstation in the Ryu controller.

Phase (2) Network discovery and network dividing: SDN controller features a full image of the topology and it dynamically updates the topology after every data flow (Packet In). HACO divides the network using either the BCR or the k-means clustering algorithm. Both algorithms are introduced as follow:

Box-Covering algorithm (BCR) in [2] divides the network into boxes or subnets. A box size is given in terms of the network distance, which corresponds to the number of edges on the shortest

path between two nodes. The idea of the BCR algorithm is illustrated in Fig. 5. To find the shortest path from node 1 to node 25, the network is split into six boxes. Each box is considered as one node and the dimension of the network is prominently decreased. If there is an edge between two nodes in two different boxes, then these two boxes are connected. The shortest path between node 1(box 1) and node 5 is found using the proposed HACO rather than using the Dijkstra. Then, the shortest path in each box is calculated then the shortest paths are linked together to get the globally shortest path (the red lines) from node 1 to node 25.

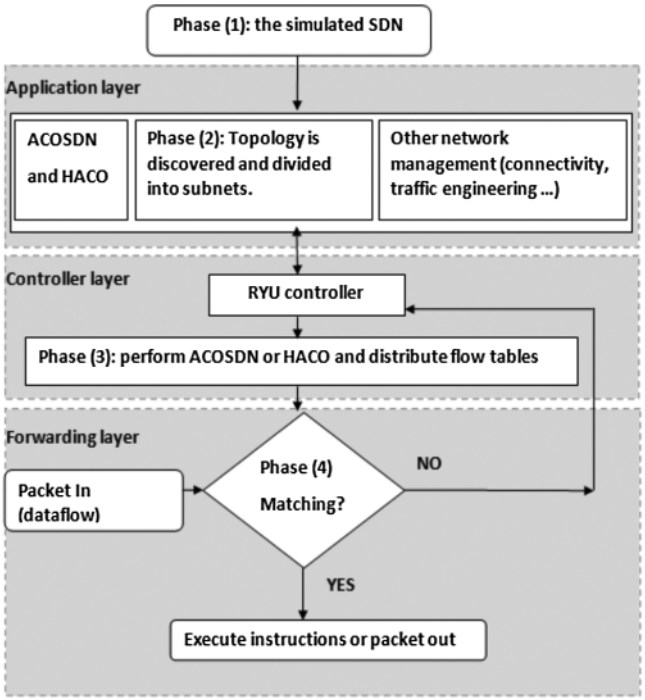

Figure 4: SDN deployment phases

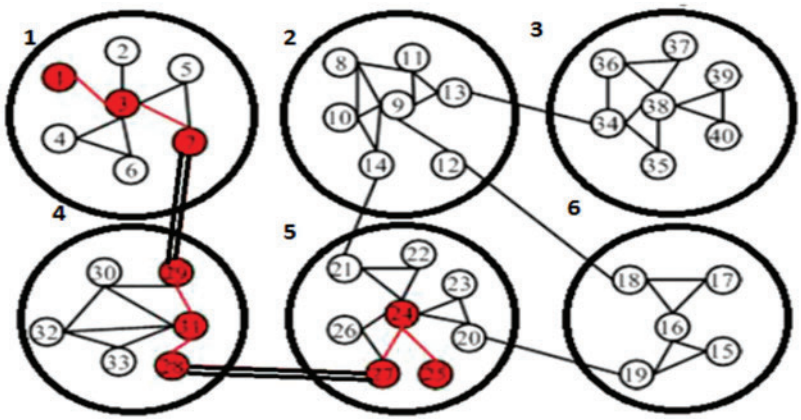

Figure 5: An example to illustrate the Box-Covering algorithm

K-means clustering is a type of unsupervised learning. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. For large scale networking, K-Means is computationally faster than hierarchical clustering [14] and considers the best partitioning clustering algorithm according to the time complexity [15]. The goal of the algorithm is to partition the n nodes into k sets (clusters) Si where, i = 1, 2… k so that the within-cluster sum of squares is minimized, defined as Eq. (5) [16].

$$\sum_{j=1}^{k}\sum_{i=1}^{n}\left(\left|x_i^j - c_j\right|\right)^2, \tag{5}$$

where, term $\left(\left|x_i^j - c_j\right|\right)$ provides the distance between a node and the cluster's centroid. Traditional k-Means algorithm selects initial centroids randomly and the result of clustering highly depends on selection of initial centroids and the algorithm may find a suboptimal solution when the centres are chosen badly [17]. The Pseudo code of k-means algorithm is shown in Fig. 6. Some methods for selecting the initial centroids includes Forgy's Approach, MacQueen Method, Simple Cluster Seeking method, Kaufman Approach, and k-means++ method. This research used the k-means++ as an algorithm for choosing the initial values for the k-means clustering algorithm because it is successfully overcome some of the problems associated with the other methods [18].

BEGIN

1. Select k points as initial centroids.
2. Assign all points to the closest centroid.
3. Re-compute the centroids of each cluster as an average of the sum of squares for all nodes within the cluster.
4. Repeat steps 2 and 3 until the centroids do not change (become stable).

END

Figure 6: Pseudo code of k-means algorithm

Phase (3) The Suggested Algorithm Implementation: Here the routing process is executed by the proposed algorithm.

Phase (4) Forwarding: This phase responsible for forwarding the data through the path given form phase (3). If no matching happens, the controller is informed to take new action (drop the packet or install it in the pipeline tables).

The Proposed Algorithm:

HACO optimizes the routing in large scale SDN using four parallel optimization steps.

In the first step, the SDN network is divided into boxes using BCR or k-means. This optimizes the search space and the packet time of exploring the best path.

In the second step, assuming a zero-memory system within the network initiated for the first time. A broadcast is a way to explore all network nodes; this is often like ants' first time randomly distributed on all the available paths. An ant within the HACO algorithm decides the path to follow based on the pheromone trails on the path but, instead of covering the path where the pheromone trail is stronger just like the natural ant would do, it explores the path where

the pheromone intensity does not exceed a predefined threshold. This avoids the congestion and maximizes the network throughput.

In the third step, the packet matching time spent in each router is optimized by creating a new matching table within the OF pipeline with entries of the discovered best paths and giving the matching table the priority so that decreasing the time spent in the packet matching process and minimize both the total delay time and the packet loss rate. The probability of choosing a node is consistent with the roulette wheel statistical distribution [19] as given by Eq. (6):

$$p_{ij}(t) = \frac{\tau_{ij}(t)\,\eta_j(t)}{\sum_{k=1}^{successors\_of\_i} \tau_{ik}(t)\,\eta_k(t)}, \tag{6}$$

where $\tau_{ij}(t)$ is the concentration of pheromone between node i and node j for the (t)th iteration, $\eta_j(t)$ is the value of the heuristic information in node j and supposed to equal 0.01, $\tau_{ik}(t)$ is the concentration of pheromone between node i and node k where k is a value increasing from 1 to the number of successors of node i, $\eta_k(t)$ is its current value of the heuristic function.

The local pheromone level on all the paths discovered is decreased by an amount called the pheromone decay or the evaporation rate ρ and therefore the global pheromone level on the best path is updated and increased by α using Eq. (7):

$$\tau_{ij}(t) = \tau_{ij}(t-1) + \alpha, \tag{7}$$

In the fourth step, HACO uses a mutation operation to discover new paths. Mutation operation is mainly derived from the Genetic Algorithm (GA) but it can be applied to other meta-heuristic algorithms to increase the probability of exploring a better solution in the search space and improve the routing optimization process [20]. The mutation operation randomly selects a path from the paths that have generated in step (3) and mutates this path by a mutation probability in Eq. (8):

$$p_m = \frac{mutaion\_parameter}{number\_of\_generated\_paths}, \tag{8}$$

where p_m is the probability of mutation. For example, if the $number\_of\_generated\_paths = 20$ paths and the $mutaion\_parameter = 2$, this means that 2 paths from 20 will be mutated. Pseudo code of HACO is described in Fig. 7.

8 Performance Evaluation

The platform for implementing the proposed HACO algorithm on large-scale SDNs involves the following software tools and programming language: Ubuntu16.04, Mininet 2.2, Ryu 3.6, VMware Workstation Pro, the size l of each box is 1, clustering parameter k is 3, Iperf software was used in the SDN network flow, flow rate 2 Mb/s, bandwidth 5 Mb/s and Python 2.7.9. The hardware environment includes a PC that has an Intel i7 as a CPU, 8 GB memory, and 1 GB hard disk.

This platform is used to create SDN, and then the performance of the HACO algorithm is measured as follows:

- Measuring the performance of HACO under dynamic changing of the topology.
- Testing the HACO using k-means network delay and packet loss at different centroids.
- Testing the proposed HACO total network delay and packet loss rates at different network sizes.

```
BEGIN
Initialize
Divide SDN network into subnets using Box-Covering or k-means
While stopping criteria is not satisfied do
    Position all ants in a starting node
    Repeat
    For each ant do
    Choose the next node using Eq. 6
    Apply local pheromone update using Eq. 1
    End for
    Add discovered paths to the new table in the OF-pipeline
    Until every ant has built a path
    Implement mutation using Eq. 8
    Update the new table with the discovered paths by the mutation process
    Choose the best path from the new table
    Apply global pheromone update using Eq. 7
End While
END
```

Figure 7: Pseudo code of the proposed HACO

- Comparing the performance of HACO against other routing algorithms in SDN and literature relevant algorithms consistent with the running time.
- Comparing the performance of HACO against other routing algorithms in SDN and literature relevant algorithms consistent with the delay time.

Measuring the performance of the HACO under dynamic changing of the network topology:

At a predefined time-instance, a network device is added, and the network is reconfigured, and therefore the best paths are updated consistent with the least hop count and congestion [15].

Testing the HACO using k-means network delay and packet loss at different centroids:

For different network sizes, the k-means++ method is generating the initial centroids, then the HACO using k-means is implemented at different centroids to choose the best centroids which achieve the minimum network delay and packet loss rates. Figs. 8 and 9 presents plots of the centroid's movement against the network delay and packet loss in case of network size is 100 nodes. It is observed that the best centroid value which achieves the minimum network delay and packet loss is 3.

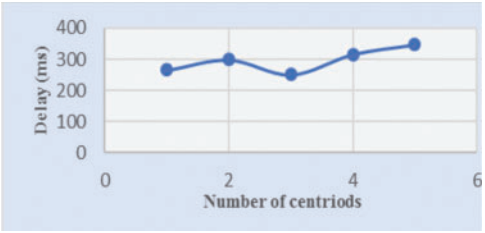

Figure 8: Centroid movement against the network delay

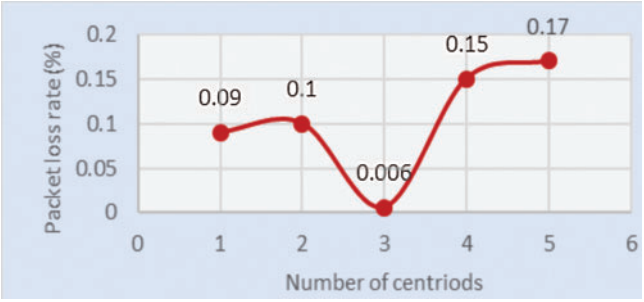

Figure 9: Centroid movement against packet loss rate (%)

Testing the proposed HACO total network delay and packet loss rates at different network sizes:

HACO is executed at different network sizes as shown in Tab. 1.

Table 1: Network size against total delay and packet loss

Network size	Total delay using box-covering (ms)	Total delay using k-means (ms)	Packet loss rate using box-covering (%)	Packet loss rate using k-means (%)
10	19.2	27.2	0.002	0.003
50	90.2	105.3	0.003	0.004
100	220.1	251.2	0.005	0.006
500	403.4	469.4	0.008	0.009
750	489.4	567.1	0.010	0.201
1000	667.3	813.6	0.100	0.365
2000	876.1	1165.8	0.231	0.582
5000	1004.2	1398.4	0.398	0.895

Fig. 10 shows that the entire delay by box-covering or k-means is proportional to the network size but not in a linear behavior. This because of the stochastic nature of meta-heuristic algorithms. the entire delay by box-covering and k-means is approximately equal until 100 nodes. With the rapid growth of the numbers of nodes from 500 to 5000, the entire delay by k-means is worse than the entire delay using box-covering.

Fig. 11 shows acceptable packet loss rates by either box-covering or k-means which is smaller than the benchmark of 1% at 10 Mb/s dedicated for voice and video streaming. The packet loss rates by box-covering and k-means are approximately equal to 500 nodes. With the rapid growth of the numbers of nodes from 750 to 5000, the packet loss rate using k-means is worse than the packet loss rate using box-covering.

Comparing the performance of HACO against other routing algorithms in SDN and literature relevant algorithms according to the running time:

HACO is implemented at different network sizes against the running time and compared with both Dijkstra and BCR algorithm in [2] as indicated in Tab. 2.

Fig. 12 indicates that the running time of Dijkstra, BCR, HACO using box-covering and HACO using k-means algorithms is approximately equal to 500 nodes. With the rapid growth

of the numbers of nodes from 750 to 5000, the advantage of HACO using box-covering and mutation algorithm becomes increasingly obvious.

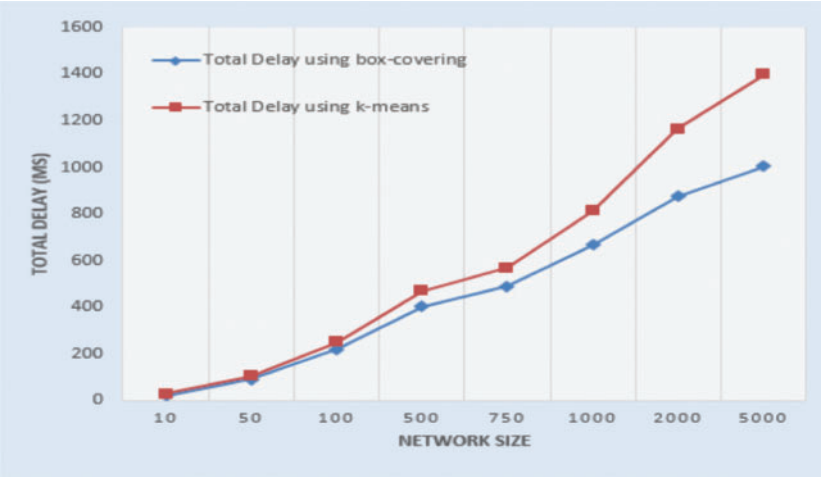

Figure 10: Network size against total delay

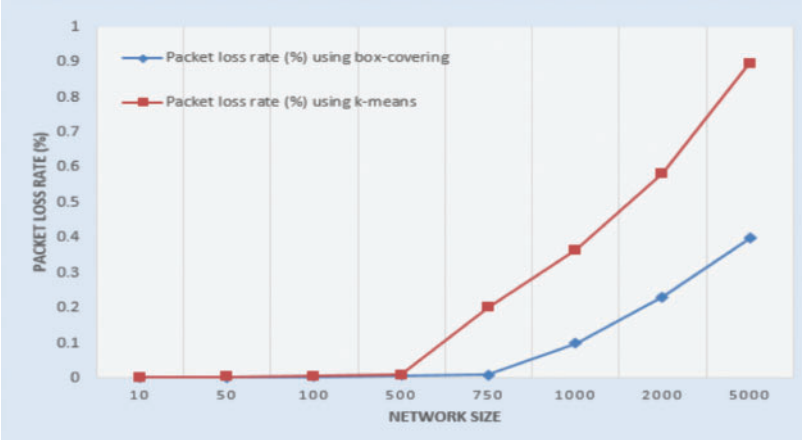

Figure 11: Network size against packet loss rate

Table 2: Performance of HACO against other routing algorithms (Network size against running time)

Network size	Running time (ms)			
	HACO using box-covering	HACO Using k-means	Dijkstra Alg.	BCR Alg. in [2]
10	0.00001	0.00098	0.00113	0.00998
50	0.00009	0.00812	0.03098	0.03899
100	0.00021	0.08001	0.09302	0.08042
500	0.04302	1.97674	3.36523	2.34667
750	1.13456	3.8534	8.72980	4.69111
1000	2.46721	4.6542	17.7252	7.32198
2000	17.5231	25.7634	92.1916	32.1823
5000	131.4875	171.872	815.167	204.111

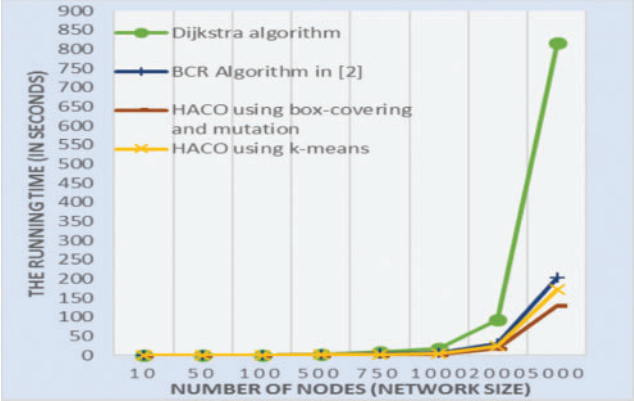

Figure 12: Performance of HACO against other routing algorithms (network size against running time)

Comparing the performance of HACO against other routing algorithms in SDN and literature relevant algorithms according to the delay time:

HACO is implemented at different network sizes against the total delay time and compared with Dijkstra and BCR algorithm in [2] as indicated in Tab. 3. The comparison is made for only 10, 50 and 100 nodes because these are the only network sizes used as the benchmark sizes for the literature relevant algorithms.

Table 3: Performance of HACO against other routing algorithms (network size against delay time)

	Delay time (ms)			
Network size	HACO using box-covering	HACO using k-means	Dijkstra Alg.	BCR Alg. in [2]
10	19.2	27.2	50.4	100.1
50	90.2	105.3	130.6	129.7
100	220.1	251.2	535.4	300.4

The results shown in Fig. 13 are analysed as follow:

When the number of nodes is 10, the delay time by BCR algorithm is the worst and the delay time by HACO using box-covering is the best one. When the number of nodes is 50, the two delay times by BCR and Dijkstra are approximately the same but the delay time by HACO using box-covering is still the best one. When the number of nodes reaches 100, the delay time by Dijkstra algorithm becomes the worst and the delay time by HACO using box-covering is still the best one, consequently the proposed HACO using box-covering outperforms the other algorithms.

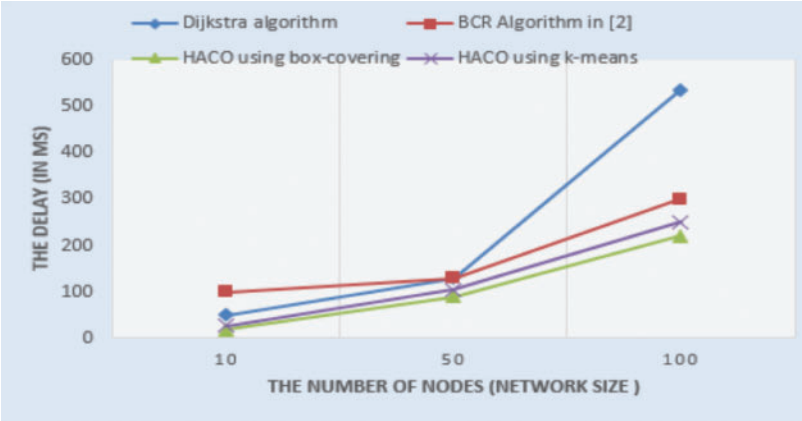

Figure 13: Performance of HACO against other routing algorithms (network size against delay time)

9 Conclusion and Points for Discussion

This paper suggested Hybrid Ant Colony Optimization (HACO) algorithm for optimizing the routing problem inside SDNs.

HACO using box-covering optimized the time and space complexity and the mutation gives a far better divergence and a far better chance for HACO for exploring less congested paths. A new table within the OF pipeline is created which contains all the explored paths. This optimizes the packet matching time and both the network delay and running times and maximizing the network throughput.

By comparing with other routing algorithms, the results show that HACO using box-covering outperforms all other algorithms and achieves a significant reduction of the network delay, packet loss rates, and running times.

It is recommended to use either HACO using box-covering or HACO using k-means when the network size is less than 50 nodes and to use HACO using box-covering when the network size exceeds 50 nodes.

As a future point for research, the proposed HACO may be improved by optimizing the initial centroids or the box-size values.

Funding Statement: The authors received specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Abdulaziz and S. Yahya, "Improved extended dijkstra's algorithm for software defined networks," *International Journal of Applied Information Systems*, vol. 1, no. 2, pp. 249–868, 2017.

[2] L. Zhang and Y. Hu, "A box-covering-based routing algorithm for large-scale SDNs," *Journal of IEEE Access*, vol. 5, no. 2, pp. 314–327, 2017.

[3] L. Lingxia and L. Feng, "Evolutionary algorithms in software defined networks," *Journal of ZTE Communications*, vol. 15, no. 3, pp. 20–36, 2017.

[4] B. Assefa and O. Ozkasap, "State-of-the-art energy efficiency approaches in software defined networking," in *Proc. SoftNetworking*, Barcelona, Spain, pp. 555–567, 2015.
[5] E. Duran and G. Caraus, "On software defined networks for particle accelerators," Ph.D. dissertation, Lund University, Scandinavia, Northen Europe, 2018.
[6] W. Braun and M. Menth, "Software-defined networking using openflow: Protocols, applications and architectural design choices," *Journal of Future Internet*, vol. 5, no. 3, pp. 302–336, 2014.
[7] M. Maugendre, "Development of a performance measurement tool for SDN," M.Sc. dissertation, UPC University, Barcelona, Spain, 2015.
[8] C. Black and T. Culver, *Openflow Switch Specification*. USA: Open Networking Foundation, pp. 89–136, 2015. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.
[9] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 7th ed., USA: Pearson, 2017. [Online]. Available: https://www.pearson.com/us/higher-education/program/Kurose-Computer-Networking-A-Top-Down-Approach-7th-Edition/PGM1101673.html?tab=features.
[10] M. Alnaser, "A method of multipath routing in SDN networks," in *Advances in Computer Science and Engineering*. vol. 17. Allahabad, India: Publishing House, pp. 11–17, 2018.
[11] P. Asher, "Comprehensive analysis of dynamic routing protocols in computer networks," *International Journal of Computer Science and Information Technologies*, vol. 6, pp. 4450–4455, 2015.
[12] A. Karim and M. Khan, "Behaviour of routing protocols for medium to large scale networks," *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 3, pp. 1605–1613, 2011.
[13] N. Gupta and K. Mangla, "Applying Dijkstra's algorithmin routing process," *International Journal of New Technology and Research*, vol. 2, no. 5, pp. 122–124, 2016.
[14] O. Abdel Raouf and H. Askr, "ACOSDN–Ant colony optimization algorithm for dynamic routing in software defined networking," in *Proc. ICCES*, Cairo, Egypt, pp. 141–148, 2019.
[15] D. Xu and Y. Tian, *A Comprehensive Survey of Clustering Algorithms*. vol. 2. Berlin Heidelberg: Springer-Verlag, pp. 165–193, 2015.
[16] A. Baswade and P. Nalwade, "Selection of initial centroids for k-Means algorithm," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 7, pp. 161–164, 2013.
[17] R. Alvida and I. Ikhwan, "Using k-means++ algorithm for researchers clustering," in *Proc. AIP*, USA, pp. 20052, 2017.
[18] D. Sonagara and S. Badheka, "Comparison of basic clustering algorithms," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 2, pp. 58–61, 2014.
[19] Z. Liang and Z. Zhu, "Orderly roulette selection based ant colony algorithm for hierarchical multi-label protein function prediction," *Journal of Mathematical Problems in Engineering*, vol. 2017, no. 1, pp. 1–15, 2017.
[20] H. Xu and F. Duan, "A Hybrid ant colony optimization for dynamic multi-depot vehicle routing problem," *Journal of Discrete Dynamics in Nature and Society*, vol. 2018, no. 1, pp. 1–10, 2018.