Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-179399
IOS Press

An optimal and dynamic elephant flow scheduling for SDN-based data center networks

Honghui Li*, Hailiang Lu and Xueliang Fu
College of Computer & Information Engineering, Inner Mongolia Agricultural University, Hohhot, China

Abstract. With the rapid development of data center network, the traditional traffic scheduling method can easily cause problems such as link congestion and load imbalance. Therefore, this paper proposes a novel dynamic flow scheduling algorithm GA-ACO (Genetic Algorithm and Ant COlony algorithms). GA-ACO algorithm obtains the global perspective of the network under the SDN (Software defined network) architecture. It then calculates the global optimal path for the elephant flow on the congestion link, and reroutes it. Extensive experiments have been executed to evaluate the performance of the proposed GA-ACO algorithm. The simulation results show that, in comparison with ECMP and ACO-SDN algorithm, GA-ACO can not only reduce the maximum link utilization, but also improve the bandwidth effectively.

Keywords: Data center network, SDN, elephant flow scheduling, genetic algorithm, ant colony algorithm

1. Introduction

In the traditional data center networks, most of traffic flow between external clients and internal servers. Nowadays, with the continuous development of cloud computing and big data technology, the scale of the data center network become more and more enormous. Moreover, the amount of the east-west traffic flow [1] in the network is far larger than that of the access traffic from the outside, and keeps growing exponentially [2].

Most of traditional data center networks adopt the tree topology, which may cause some issues, i.e., root node limiting network bandwidth, difficult expansion [3] etc. Also, the growing internal traffic overburdens the traditional data center network structure, which may cause link congestion frequently, and results in less efficiency of traffic transmission. Thereby, researchers proposed many network architectures for today's data centers, such as Fat-Tree [4], OSA [5], c-Through [6], BCube [7], VL2 [8], etc. Among these architectures, the fat-tree topology is relatively simple and provides higher network bisection bandwidth than the traditional tree topology. Moreover, there exist multiple redundant paths with the same cost between two nodes [9], which can effectively prevent from network congestion and facilitate load balancing.

Currently, ECMP (Equal-Cost Multipath Routing) algorithm [10] has been widely deployed for traffic scheduling in data center network. ECMP can evenly distribute different data streams between two endpoints to multiple equivalent paths based on the hash algorithm in the fat-tree networks, and thereby load balancing can be approached. Studies [11, 12] have shown that the traffic flow in the data center network includes elephant flow and mouse flow. The elephant flow refers to the flow which occupies 10% or more

*Corresponding author. Honghui Li, College of Computer & Information Engineering, Inner Mongolia Agricultural University, Hohhot, China. E-mail: lihh@imau.edu.cn.

of the link bandwidth. The elephant flows account for only 10% of the flows in the data center network. However, the elephant flows carry 90% of the traffic data. ECMP algorithm can well distribute the mouse flow evenly to the equivalent paths. Since the duration of the elephant flow usually lasts longer, upon a hash collision occurrence, multiple elephant flows might be allocated to the same path. The consequence is that the network congestion occurs, and the redundant paths are idle. Furthermore, network load is uneven distributed, and the utilization of network resources is lower down.

As a new type of the network architectures, SDN [13] (Software Define Network) overcomes the localized weakness of traditional networks, and can unified control the traffic flows from the global network point of view. And thus, SDN can manage network traffic and resources accurately and effectively. Thereby, SDN may provide a good solution to the bottleneck issue of the data center network development.

In order to address the issue of ECMP scheduling elephant flow, this paper proposes the new GA-ACO algorithm, which well combines the genetic algorithm (GA) and the improved ant colony algorithm (ACO). GA-ACO can schedule the elephant flow dynamically and efficiently in the SDN data center network with the fat-tree topology. From the global perspective, GA-ACO calculates the global optimal path according to the current state of the network, and re-routes the elephant flow on the congestion link. Simulation results show that, in comparison with the traditional ECMP algorithm and ACO-SDN [21], GA-ACO reduces the ratio of maximum link utilization, and boosts bisection bandwidth. Thus, network load balance is approached, and network resource utilization is improved.

2. Related work

SDN technology provides a feasible solution for the development and innovation of the today's data center networks. Network traffic scheduling under the SDN architecture has become a hot topic of many domestic and foreign scholars.

Mohammad Al-Fares et al. [14] presents the Hedera dynamic flow scheduling mechanism for the SDN data center network. In order to improve the bisection bandwidth, Global First Fit (GFF) algorithm is proposed. According to the state of current link bandwidth, GFF selects the first path to route the elephant flow from all possible paths satisfying the flow bandwidth.

Hedera may increase the network extra overload since it monitors the elephant flow in the forwarding layer. Aiming at this issue, Andrew R.Curtis et al. proposed the Mahout [3] algorithm. This algorithm detects the elephant flow by monitoring the connection cache of the host, and communicates with the controller through an in-band signal mechanism. The controller only calculates the path of the detected elephant flow, and other flows are forwarded by ECMP.

Tang et al. [15] propose the DLBS (dynamic load balancing scheduling) mechanism to address load balancing and traffic scheduling in the SDN-based cloud data center network. And an efficient heuristic scheduling algorithm is proposed for or two typical OpenFlow network models. The experimental results show that DLBS is superior to other load balancing algorithms.

Chakraborty and Chen [16] propose a simple and effective multipath routing scheme for the elephant flow scheduling in the data center network. The objective is to minimize the flow completion time and delay. The main idea is to divide the elephant flow into the multiple mouse flow, and then schedule them based on the VLAN-based routing scheme. Thus, the completion time of the flow is effectively cut down, and the resource consumption of flow table items is also reduced on the aggregation switches.

Zhang et al. [17] formulate the problem of the elephant flow scheduling in the SDN data center network based on the stability matching theory. And the Fincher scheme is proposed to dispatch the elephant flow, which can effectively avoid data center network congestion and reduce latency. Simulation results show that the Fincher performance surpasses ECMP and Hedera in terms of the average network bisection bandwidth and the flow completion time.

Most related studies only focus on the elephant flow scheduling, which may disturb the mouse flow transmission. In order to deal with this issue, Zhang et al. [18] propose a differentiated dynamic scheduling algorithm. This algorithm consists of the weighted multipath scheduling algorithm and blockade Island path setting algorithm to schedule mouse flow and elephant flow respectively. Also, it integrates the traffic flow reroute mechanism of crowded link. Thereby, this algorithm achieves the efficiency of network traffic management and load balancing.

In order to approach the issue of load unbalancing and low throughput caused by the traditional routing methods in data center networks, Zang et al. [19]

propose the SDN-based multipath flow scheduling (SMFS) mechanism. Based on SDN, SMFS improves the load balancing and increases throughput with usage of periodic polling and dynamic flow scheduling. Also, the particle swarm optimization algorithm together with the segment routing is utilized to reroute selectively the elephant flow, which enhances the flow transmission rate efficiently.

Li et al. [20] propose the ACO-SND algorithm to address the problem of network congestion and poor load balancing caused by the traditional flow scheduling method in data center networks. ACO-SDN employs the improved ant colony algorithm to calculate the optimal path for the elephant flow on the congestion link, and then rerouting it. This method can improve the bisection bandwidth effectively and achieve good load balance. However, the ant colony algorithm is prone to fall into the problem of local optimization. This issue is not well solved, which may lead to the secondary collision of elephant flows and decrease the network performance.

3. Mathematical model

Let us represent a data center network as a graph $M(S, L)$, where $S = \{s_1, s_2, \ldots, s_n\}$ is node set, and L is the link set of the network. L_i denotes the link set of all traffic flows coming into node S_i, $L_i{}'$ is the link set of all data flows going out of node S_i, the capacity of link $l \in L$ is B_l, and the link utilization rate is u_l. Let E be the set of elephant flow to be scheduled in the current network, indexed by e. b_e denotes the bandwidth of elephant flow e. S_o^e and S_d^e represents respectively source and destination node for flow e. Variable $x_l^e \in \{0, 1\}$ indicates whether the routing of flow e passes through link l.

Link utilization ratio is defined as the ratio between the overall bandwidth usage of all data flow passing through the link and the link capacity, as shown in formula (1).

$$u_l = \frac{\sum\limits_{e \in E} x_l^e b_e}{B_l} \tag{1}$$

Maximum link utilization ratio refers to the maximum of all link utilization ratios in the network, that is,

$$u^{\max} = \max\{u_l\} \qquad l \in L \tag{2}$$

The mathematical model of the traffic scheduling problem is formulated as follows.

$$\min \quad u^{\max} \tag{3}$$

$$s.t. \quad \sum_{e \in E} x_l^e b_e \le B_l \qquad l \in L \tag{4}$$

$$\sum_{l \in L_i} x_l^e b_e - \sum_{l \in L_i{}'} x_l^e b_e = \begin{cases} -b_e & s_i = s_o^e, e \in E \\ b_e & s_i = s_d^e, e \in E \\ 0 & s_i \in s - \{s_o^e, s_d^e\} \end{cases} \tag{5}$$

$$x_l^e \in \{0, 1\} \qquad l \in L, e \in E \tag{6}$$

Formula (3) is the objective function of the flow scheduling problem. It means that the objective is to minimize the maximum link utilization ratio of the network. Formula (4∼6) are constraints that should be met in traffic scheduling. Formula (4) ensures that the overall bandwidth usage of data flows carried by the link will not exceed the capacity of the link. Formula (5) is the flow conservation constraints. It ensures that, for each intermediate node, the number of the incoming flows is equal to that of outgoing flows; and for source (destination) node, there is only outgoing (incoming) flows. Formula (6) defines the value range of variables.

This mathematical model belongs to the integer linear programming (ILP) model. In order to get the flow scheduling scheme to route elephant flow, this paper proposes the GA-ACO algorithm to solve this ILP model.

4. GA-ACO algorithm

In this section, the overview of the GA-ACO algorithm is first present, and then GA-ACO is explained in detail how to solve the ILP model of flow scheduling problem in section 2.

4.1. GA-ACO overview

GA-ACO algorithm flow chart is shown as Fig. 1. The main steps of GA-ACO algorithm are described as follows.

1) Based on the sflow-rt collector, the link state information of the data centre network is continuously collected.
2) As soon as the data flow is arriving at the data center network, GA-ACO first calls the ECMP algorithm to route it.

3) If a link utilization ratio is greater than the threshold (for example, 60%), go to step 4, otherwise go to step 1.
4) Employ genetic algorithm to calculate several available candidate paths according to the current network link utilization ratio.
5) Initialize ant colony algorithm pheromone of each link. For the link on those paths obtained in step 4, its pheromone is initialized as a positive number, and for the other links, is set to 0.
6) Call ant colony algorithm to calculate the global optimal path, reroute the elephant flow.

The following presents first the genetic algorithm in detail regarding how to achieving multiple candidate paths; and then the ant colony algorithm for approaching the optimal path to reroute the elephant flows.

4.2. Genetic algorithm

Genetic algorithm is a process search and swarm intelligence optimization algorithm, which was proposed by John Holland et al in the 1970 s [21]. This algorithm is based on the Darwin's biological evolution theory, Weizmann's species selection theory and Mendel's genetic variation theory. The basic idea is to simulate the natural biological evolution and genetic mechanism to solve optimization problems. The algorithm takes the fitness function as the evaluation criterion for selection. Also, the operations are employed of coding, selection, crossover and mutation. After several iterations of these operations, the optimal solution was finally obtained.

Here, genetic algorithm is used to calculate multiple available paths that meet formula (4∼6) based on the current network topology and link utilization. The solution space R is composed of several available paths as alternative paths for the elephant flow. Genetic algorithm is specified as follows.

1) Encoding

Chromosomes in the genetic algorithm are defined as candidate paths in order to solve the ILP model above. One path corresponds to one chromosome, which consists of a chain of consecutive links. The following example illustrates how a path (a link list) is encoded.

As shown in Fig. 2, a path r_i exists between host h_1 and host h_5. Let represent switch i(i is a hexadecimal number), and the digital numbers next to the rectangle represent the indexes of the incoming and outgoing ports of the switch.

The encoding of path r_i is

$\{(s_d, s_f), [(s_d, 3), (s_d, 1), (s_5, 3), (s_5, 1), (s_1, 1), (s_1, 2), (s_7, 1), (s_7, 3), (s_f, 1), (s_f, 3)]\}$

2) Fitness value calculation

The fitness function is adapted to solve the ILP model above, which is a minimization problem. For path r_i, the formula is shown below as formula (7).

$$F(r_i) = 1/(\alpha * COST(r_i) + \beta * U(r_i)) \quad (7)$$

Where, path $r_i \in R$, R is the solution space. $COST(r_i)$ is the length of path r_i, and $U(r_i)$ is the maximum link utilization rate of path r_i. α and β are the impact factors. Their values affect the influence of the path length and the maximum link utilization on fitness value.

Formula (7) is used to calculate the fitness function value of each link list in solution space R. The higher the fitness value of a link list, the easier it is to be preserved in genetic manipulation.

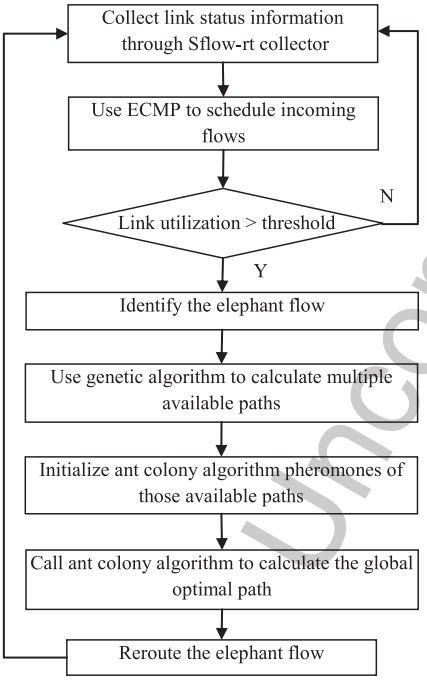

Fig. 1. GA-ACO flow chart.

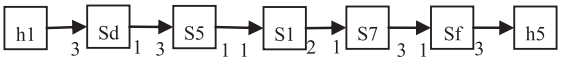

Fig. 2. Full path instance.

3) Selection operation

The purpose of the selection operation is to identify links that are passed on to the next generation. Thus, the genetic probability and the cumulative probability of each link list in the solution space need to be calculated. Based on these, the roulette algorithm is utilized to determine the inherited link list.

For the solution space R, its size is expressed as N. Let the probability of link list r_i be $p(r_i)$, and the cumulative probability be $q(r_i)$. They are calculated as formula (8) and (9) below respectively.

$$p(r_i) = \frac{F(r_i)}{\sum_{j=1}^{N} F(r_j)} \tag{8}$$

$$q(r_i) = \sum_{j=1}^{i} p(r_j) \tag{9}$$

According to Formula (8) and (9), the genetic probability and the cumulative probability are calculated for each link list in solution space R. Then, N iterations are carried out. Each iteration produces a random number t between [0,1]. If $t < q(r_i)$, the first link list is selected to join the next generation population; otherwise, select r_k to join the next generation population according to formula $q(r_{k-1}) < t \leq q(r_k)$.

4) Crossover operation

The traditional crossover operation takes random locations to cross two chromosomes, and then creates new individuals. This paper redefines the crossover operation according to the characteristics of paths. First, pair in order. For a pair of link lists with common nodes, all links after the first common node are exchanged to complete the single-point crossover. Then, it is checked whether a loop exists in the link list. If there is a loop, the crossover operation retains the link list with removal of the loop.

5) Mutation operation

The traditional mutation operation is also improved to satisfy the path integrity rules. The main idea is to mutate the relatively busy or disconnected links, let the path bypass the mutated links.

6) If the iteration is not completed, go to step 2). Otherwise, the link lists in the solution space R are sorted in descending order by fitness value. If the number of link lists is greater than NUM, take the first NUM link lists; otherwise, take all link lists and output them to the ant colony algorithm below.

4.3. Ant colony algorithm

Ant colony algorithm is a heuristic algorithm to solve the optimization problem. It is proposed by Italy M. Dorigo et al. The algorithm utilizes the positive feedback mechanism that pheromones are released among ants in nature [22]. The basic idea is as follows. As each ant searches for food, it continually releases pheromones in its path. All the ants go along the direction of the higher pheromone concentration. After a while, all ants concentrate on the shortest path to food.

In this paper, ant colony algorithm is called to achieve the global optimal path for elephant flow rerouting. The input is the current link utilization ratio and the candidate paths achieved using the above generic algorithm. The steps of the algorithm are specified as follows.

1) Initialization

Initialize the global pheromones. For each link on the multiple available paths, its pheromone is a positive number, other link is zero. The iteration number of the algorithm is M. And the number of ants is N, and each ant has its own tabu table T, which records the nodes that the ant walks through to prevent the generation of loop. Meanwhile, the T length is used to limit the maximum distance that the ant can move. All ants are placed on the source node and T is set as empty.

2) Ant move

According to the link pheromone quantity and the current link state, the probability $p_{ij}^{ant}(t)$ of ants moving to all accessible nodes (except nodes in T) is calculated according to formula (10). According to the calculation results, the roulette method is adopted to select the next node and add this node to tabu table T.

$$p_{ij}^{ant}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{n \in V_{ant}(i)} [\tau_{in}(t)]^{\alpha}[\eta_{in}(t)]^{\beta}}, & j \in V_{ant}(i) \\ 0, & \text{otherwises} \end{cases} \tag{10}$$

Where, $p_{ij}^{ant}(t)$ represents the probability of ant from node i to node j in t iterations. $\tau_{ij}(t)$ denotes the link pheromone quantity between nodes i and j. $\eta_{ij}(t)$ tells the heuristic information between nodes i and j. The heuristic information here is defined as the reciprocal of link utilization. $V_{ant}(i)$ represents the ant accessible node set at node j, which is not in tabu T.

3) Pheromone renewal

Pheromone renewal consists of two steps. First, the global pheromone is volatilized according to the volatile factor, which simulates the volatile phenomenon of ant in nature. And then, the pheromone is updated. When all the ants arrive at the destination node or the tabu list is full, the current iteration is over. The global pheromones are next updated according to formula (11) on the basis of the tabu of ant reaching the destination node. After that, if the number of iterations is less than, jump to step 2) to the following execution. Otherwise, go to step 4).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \qquad (11)$$

Formula (11) represents the pheromone update rule of the link between node i and j in iteration $t+1$, and $\Delta\tau_{ij}(t)$ represents the pheromone increment of iteration t. The pheromones are only added to those of the links traveled by the ants at the destination. The increment amount of the link pheromone is calculated as the total pheromone quantity G divided by the path length. ρ is the pheromone volatility factor.

4) Find the optimal path

From all tabu lists of ants that reach the target node, the optimal link list is selected according to the optimization objective, and then forms an optimal path back to the controller along with the source and destination switch identifier.

5. Experimental results and analysis

Extensive experiments have been conducted to verify the performance of the proposed GA-ACO algorithm in comparisons with ECMP and ACO-SDN [20]. Two criteria, bisection bandwidth and maximum link utilization rate are employed to evaluate GA-ACO effectiveness. Bisection bandwidth is defined as the transmission bandwidth per unit time from one subnet to another when a network is divided into two subnets. The larger the bisection bandwidth is, the larger the throughput of the network will be. Maximum link utilization refers to the maximum link utilization in the network under network load. Under the same load, the lower the maximum link utilization is, the better the load balancing performance of the network will be.

5.1. Experimental environment

(1) Topology

The experimental data center network adopts 4-yuan fat-tree topology structure, as shown in Fig. 3.

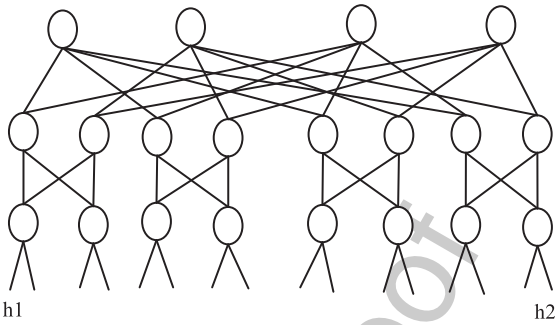

Fig. 3. Network topology.

The ellipse represents the OpenFlow switch, the line segment represents the link, and the upper, middle and lower layers represent the core layer, aggregation layer and access layer respectively. The experimental topology was simulated by the Mininet [23] simulation platform, and the bandwidth of each link was set at 100M/s. Floodlight is adopted as SDN controller, and Sflow [24] is called to monitor the network state.

(2) Communication mode

Based on literature [3], the flow with a 10Mb/s bandwidth or above is determined as elephant flow here. The communication model of data center network is designed according to literature [25–29].

Network traffic is generated by Iperf, which is Mininet's built-in traffic generation tool. The flow size follows the exponential distribution [30–32], where the exponential function parameter r is set to 0.23 to generate the flow. The time interval of each flow is subject to the Poisson distribution (time scale is second), which is limited within the (0,1) interval. The duration of each flow is 30 seconds. There communication patterns are described as follows.

1) *Stride*(i)

The host number indexed by x transmits data to the host indexed by $(x+1) \bmod n$, where n is the number of hosts in the network and i is a parameter.

2) *Staggered*($p1$, $p2$)

Each host sends data to hosts belonging to the same access layer switches with probability $p1$, sends data to hosts belonging to the same Pod with probability $p2$, and sends data to other hosts with probability $1-p1-p2$.

3) Random

Each host sends data to the other hosts with the same random probability in the network.

(3) GA-ACO algorithm parameter setting

Table 1
Main parameters of the algorithm

Parameters of genetic algorithm	Value	Parameters of ant colony algorithm	Value
Initial chromosome mumber N	50	Number of ants N	10
Impact factor α	1	Pheromone heuristic factor α	2
Impact factor β	10	Expectancy heuristic factor β	3
Crossover probability ρ	0.6	Pheromone volatility ρ	0.6
Output path number NUM	8	Total pheromone G	100

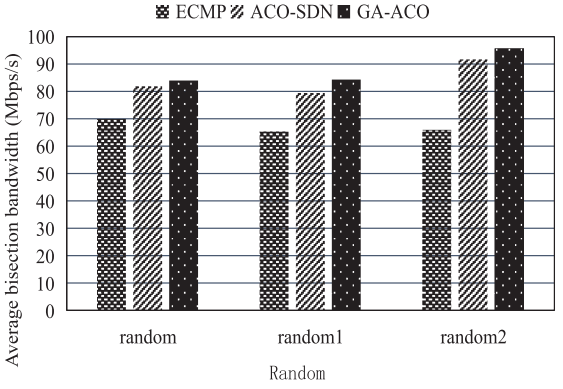

Fig. 4. Average bisection bandwidth comparisons under random mode.

The parameter setting would directly affect the performance of GA-ACO algorithm. The main parameters of the GA-ACO algorithm are set as shown in Table 1 according to the addressed traffic scheduling problem and the extensive simulation results.

During the simulation processes, it is found that the path with more than 11 hops is very unlikely to be the optimal solution in the 4-pod fat-tree network topology. Thereby, it is specified that the path greater than 11 hops is unreachable, and the length of the ant taboo table T is set as 10.

5.2. Average bisection bandwidth comparison

The average bisection bandwidth comparison results are shown in Figs. 4–6 of GA-ACO, ECMP and ACO-SDN under the three communication modes respectively. In these three figures, the horizontal axis represents three communication modes respectively, and the vertical axis represents average bisectional bandwidth.

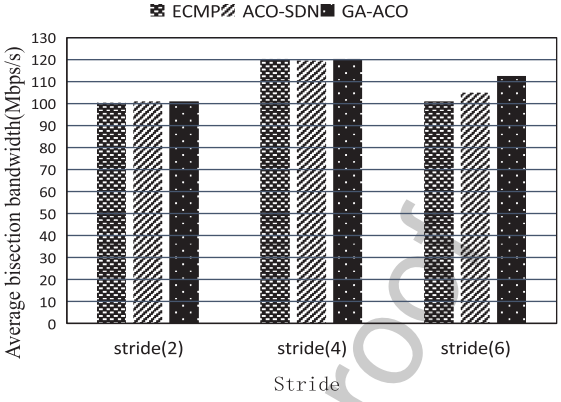

Fig. 5. Average bisection bandwidth comparisons under stride mode.

As can be seen from Fig. 4, the proposed GA-ACO overweighs ECMP and ACO-SDN in terms of average bisection bandwidth in three random modes with different random probability. GA-ACO improves by 20% 45% compared with ECMP, and improves 2.4% 6.3% compared with ACO-SDN.

Figure 5 shows that, in the stride mode, there is almost no difference between the three algorithms when the parameter i is 2 and 4. The reason is due to that the traffic pattern cause less network load. However, upon the parameter i set to 6, GA-ACO is superior to ECMP and ACO-SDN. GA-ACO improves by 11.4% compared with ECMP, and 7.2% compared with ACO-SDN. It results in the network load increasing.

It can be seen from the Fig. 6, in staggered mode, the GA-ACO average bisection bandwidth is higher than the other two algorithms. Among them, GA-ACO increases by 21.4% and 11.1% relative to ACO-SDN, and ACO-SDN increased by 5.2% and 5.9% relative to ECMP.

5.3. Maximum link utilization comparisons

It is shown that the maximum link utilization comparison of the three algorithms in three traffic modes in Figs. 7–9 respectively. In these figures, the horizontal axis is the maximum link utilization and the vertical axis is its cumulative distribution function (CDF).

As shown in Fig. 7, the maximum link utilization rates are all in the 50% to 70% range using ACO-SDN and GA-ACO in stride (6) mode. However, the ECMP algorithm is in the range of 70% to 85%. Among them, GA-ACO algorithm reduces

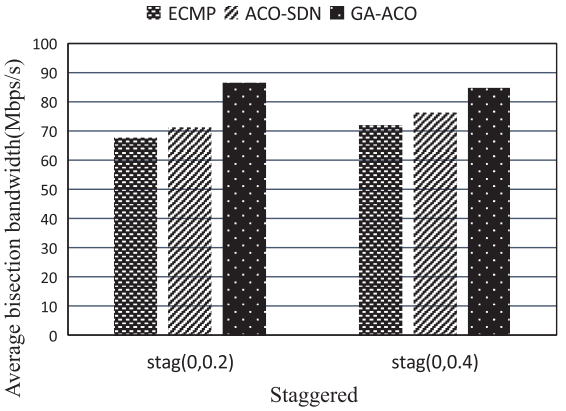

Fig. 6. Average bisection bandwidth comparisons under staggered mode.

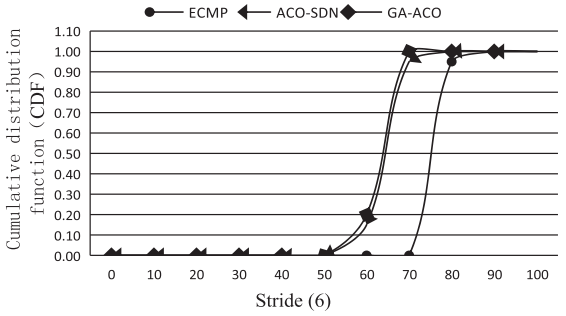

Fig. 7. Maximum link utilization under stride mode.

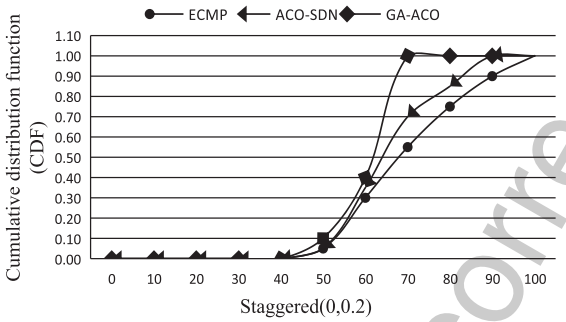

Fig. 8. Maximum link utilization under staggered mode.

by 18.4% in comparison with ECMP, and 1.5% with ACO-SDN.

As shown in Fig. 8, under the staggered(0,0.2) mode, GA-ACO maximum link utilization is concentrated between 50% and 65%, ACO-SDN is in the 50% to 88% range, and ECMP is in the 50% to 95% range. GA-ACO reduces by 14.5% compared with ECMP, and 8.1% compared with ACO-SDN.

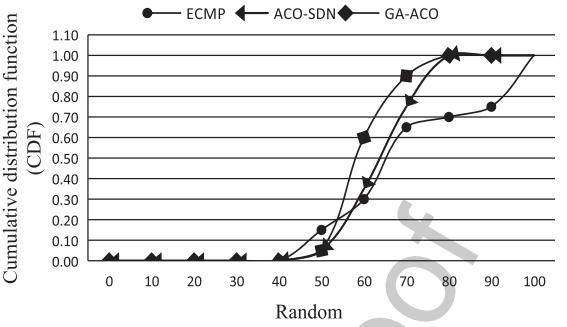

Fig. 9. Maximum link utilization under random mode.

As shown in Fig. 9, in random mode, the GA-ACO maximum link utilization rate is concentrated between 50% and 70%, ACO-SDN between 50% and 75%, and ECMP between 40% and 95%. Among them, ACO-SDN algorithm reduces by 9.8% compared with ECMP, and GA-ACO algorithm reduces by 5.9% for ACO-SDN.

ECMP is a static scheduling algorithm, which does not consider network link congestion. In the network communication mode of data center, it is easy to dispatch multiple elephant flows to the same path, resulting in network congestion. Therefore, it has the worst effect on bisection bandwidth and maximum link utilization. Although ACO-SDN algorithm makes use of the real-time state information of the network, the initial part of the pheromone of the algorithm is defective, which may make the algorithm fall into the local optimum. Therefore, the experimental effect was between GA-ACO and ECMP. The GA-ACO algorithm proposed in this paper finds out several available paths through the global search of the genetic algorithm, which serves as the basis for the pheromone initialization of the ant colony algorithm, so as to solve the problem that the ant colony algorithm is prone to fall into local optimization. It has a better effect in increasing the bisection bandwidth and reducing the maximum link utilization.

6. Conclusion

In this paper, the GA-ACO traffic scheduling algorithm is proposed to solve network congestion and load balancing problems in SDN data center network. The GA-ACO algorithm utilizes ECMP to schedule all incoming data flows in the beginning. When the link utilization ratio exceeds to some threshold, GA-ACO is called to calculate the optimal path for

rerouting elephant flow on the congestion link. GA-ACO is developed based on genetic algorithm and ant colony algorithm. Genetic algorithm is employed to search several available paths globally, which are taken as input to the ant colony algorithm. And the links on these paths are with the higher initial pheromone value that other links. Then, the (near) optimal path is found by ant algorithm. In this way, it is addressed that of pheromone lack in the early stage, and slow convergence issue, as well as being easy to fall into local optimum. Simulation results show that GA-ACO is superior to ECMP and ACO-SDN in terms of bisection bandwidth and the maximum link utilization rate.

Acknowledgments

This work was financial supported in part by Natural Science Foundation of China government, contract 61363016, and Natural science foundation of Inner Mongolia autonomous region, contract 2015MS0605 and 2015MS0626.

References

[1] D. Gang, G. Zhenghu and H. Wang, Characteristics research on modern data center network[J], *Journal of Computer Research and Development* **51**(2) (2014), 395–407.

[2] C. Yueping and W. Changping, Software defined data center network with hybrid routing[J], *Journal on Communications* **37**(04) (2016), 44–52.

[3] A.R. Curtis, W. Kim and P.M. Yalagandula, Low-overhead datacenter traffic management using end-host-based elephant detection[C]// *Proc of IEEE INFOCOM*, 2011, pp. 1629–1637.

[4] A. Greenberg, P. Lahiri, D.A. Maltz, et al., Towards a next generation data center architecture, scalability and commoditization[C]// *ACM Workshop on Programmable Routers for Extensible Services of Tomorrow ACM*, 2008, pp. 57–62.

[5] K. Chen, A. Singla, A. Singh, et al., OSA: An optical switching architecture for data center networks with unprecedented flexibility[J], *IEEE/ACM Transactions on Networking* **22**(2) (2014), 498–511.

[6] G. Wang, D.G. Andersen, M. Kaminsky, et al., C-Through, part-time optics in data centers[C]// *Acm Sigcomm Conference, ACM*, 2010.

[7] C. Guo, G. Lu, D. Li, et al., BCube, A High Performance, Server-centric Network Architecture for Modular Data Centers[J], 2009.

[8] A. Greenberg, J.R. Hamilton, N. Jain, et al., VL2: A scalable and flexible data center network[J], *Communications of the ACM* **54**(3) (2009), 95–104.

[9] J. Escudero-Sahuquillo, P.J. Garcia, F.J. Quiles, et al., A new proposal to deal with congestion in InfiniBand-based fat-trees[J], *Journal of Parallel & Distributed Computing* **74**(1) (2014), 1802–1819.

[10] C. Hopps, Analysis of an Equal-Cost Multi-Path Algorithm[M]. RFC 2992, IETF, RFC, 2000.

[11] S. Kandula, S. Sengupta, A. Greenberg, et al., The nature of data center traffic, measurements & analysis[C]// *Proc of ACM SIGCOMM Conference on Internet Measurement*, 2009, pp. 202–208.

[12] T. Benson, A. Akella and D.A. Maltz, Network traffic characteristics of data centers in the wild [C]// *Proc of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 267–280.

[13] D.P. Manoj Kumar and Dr.Y.P Gowramma, Development of sensitivity classification approach for personalized privacy preservation in data publishing (PPPDP), *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)* **5**(12) (2017).

[14] M. Al-Fares, S. Radhakrishnan, B. Raghavan, et al., Hedera, dynamic flow scheduling for data center networks[C]// *Usenix Symposium on Networked Systems Design and Implementation, NSDI* **2010** (2010), 281–296.

[15] R.V. Tali and S. Bandi, A comparative study on image isolation and classification techniques in microscopic blood smear images, *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)* **5**(11) (2017).

[16] S. Chakraborty and C. Chen, A low-latency multipath routing without elephant flow detection for data centers[C]// *IEEE, International Conference on High Performance Switching and Routing IEEE*, 2016, pp. 49–54.

[17] Y. Zhang, L. Cui and Y. Zhang, A stable matching based elephant flow scheduling algorithm in data center networks[J], *Computer Networks* **120** (2017), 186–197.

[18] H. Zhang, F. Tang and L. Barolli, Efficient flow detection and scheduling for SDN-based big data centers[J], *Journal of Ambient Intelligence & Humanized Computing* (2018), 1–12.

[19] Z. Weifei, L. Julong and H. Yuxiang, SDN based multipath flow scheduling mechanism[J], *Application Research of Computers* **35**(320(06)) (2018), 223–227.

[20] L. Honghui, Y. Guang, L. Hailiang, F. Xueliang and S. Zhijun, Flow scheduling of elephant flows in SDN data center network based on ant colony algorithm[J/OL], *Application Research of Computers*, 1–7 [2019-05-31].

[21] J.H. Holland, Adaptation in Natural and Artificial System[M], Ann Arbor University of Michigan Press, 1975.

[22] M. Chaturvedi and Prof.D. Agrawal, Time sliced and priority based load balancer, *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)* **6**(1) (2018).

[23] B. Lantz, B. Heller and N. Mckeown, A network in a laptop, rapid prototyping for software-defined networks[C]// ACM Workshop on Hot Topics in Networks HOTNETS 2010, Monterey, CA, USA – October. DBLP, 2010, pp. 1–6.

[24] K. Giotis, C. Argyropoulos, G. Androulidakis, et al., Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments [J], *Computer Networks* **62**(5) (2014), 122–136.

[25] M. Al-Fares, A. Loukissas and A. Vahdat, A scalable, commodity data center network architecture[C]// *Proc of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2008, pp. 63–74.

[26] M. Al-Fares, S. Radhakrishnan, B. Raghavan, et al., Hedera, dynamic flow scheduling for data center networks[C]// *Usenix Symposium on Networked Systems Design and Implementation, NSDI* **2010** (2010), 281–296.

[27] W. Wentao, Z. Fang, W. Lingxia, et al., Design and implementation of flow scheduling mechanism based on SDN for data center network[J], *Journal of South-Central University for Nationalities (Nat, Sci Edition)* **35**(03) (2016), 135–140.

[28] S. Din, A. Paul and A. Rehman, 5G-enabled Hierarchical architecture for software-defined intelligent transportation system, *Computer Networks* **150** (2019), 81–89.

[29] L. Zhihua, G. Wen, W. Chunming and L. Yongyan, Data center network flow scheduling based on DPSO algorithm[J], *Acta Electronica Sinica* **44**(09) (2016), 2197–2202.

[30] F. Tang, L.T. Yang, C. Tang, et al., A dynamical and load-balanced flow scheduling approach for big data centers in clouds[J], *IEEE Transactions on Cloud Computing* (2016), 1–1.

[31] N. Mckeown, T. Anderson, H. Balakrishnan, et al., OpenFlow: Enabling innovation in campus networks[J], *ACM SIGCOMM Computer Communication Review* **38**(2) (2008), 69–74.

[32] D. Haipin, Z. Xiangyin and X. Chunfang, Bio-inspired Computing[M]. Beijing, Science Press, 2011.