IET Communications

Research Article

IET Journals
The Institution of Engineering and Technology

Energy-aware dynamic-link load balancing method for a software-defined network using a multi-objective artificial bee colony algorithm and genetic operators

ISSN 1751-8628
Received on 8th December 2019
Revised 19th August 2020
Accepted on 27th August 2020
E-First on 13th October 2020
doi: 10.1049/iet-com.2019.1300
www.ietdl.org

Ali Akbar Neghabi[1], Nima Jafari Navimipour[2], Mehdi Hosseinzadeh[3,4] ✉, Ali Rezaee[1]

[1]*Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*

[2]*Future Technology Research Center, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan*

[3]*Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam*

[4]*Mental Health Research Center, Psychosocial Health Research Institue, Iran University of Medical Sciences, Tehran, Iran*

✉ *E-mail: Hosseinzadeh.m@lums.ac.ir*

Abstract: Information and communication technology (ICT) is one of the sectors that have the highest energy consumption worldwide. It implies that the use of energy in the ICT must be controlled. A software-defined network (SDN) is a new technology in computer networking. It separates the control and data planes to make networks more programmable and flexible. To obtain maximum scalability and robustness, load balancing is essential. The SDN controller has full knowledge of the network. It can perform load balancing efficiently. Link congestion causes some problems such as long transmission delay and increased queueing time. To overcome this obstacle, the link load balancing strategy is useful. The link load-balancing problem has the nature of NP-complete; therefore, it can be solved using a meta-heuristic approach. In this study, a novel energy-aware dynamic routing method is proposed to solve the link load-balancing problem while reducing power consumption using the multi-objective artificial bee colony algorithm and genetic operators. The simulation results have shown that the proposed scheme has improved packet loss rate, round trip time and jitter metrics compared with the basic ant colony, genetic-ant colony optimisation, and round-robin methods. Moreover, it has reduced energy consumption.

1 Introduction

Network management is complicated and costly in traditional networks [1, 2]. Software-defined network (SDN) [3] technology has emerged from Clean Slate's Project at Stanford University. It separates the control plane from the data plane to make network management more convenient. The SDN structure has three levels: application, control, and infrastructure [4]. The third level has forwarding components, SDN switches, which supply packet forwarding. Different from the traditional networks, SDN has the feature of centralised control. In the control layer, the logically centralised and programmable SDN controller takes the global network's data, leading to more convenient network management. The network administrators can control the network and decompose the underlying network infrastructure from the applications via a central controller [3]. SDN switches are configured by the SDN controller and are installing forwarding rules via interactive interfaces, such as OpenFlow [3]. These switches simply process the task requests, according to the appropriate rules. Network software and services like load balancing, scheduling, and routing can be performed by the application layer [3]. In the infrastructure layer, the SDN switches fulfil only simple packets forwarding functions so that the entire network can easily handle the increasing traffic volume [1].

The distribution of flows over the entire network can be one of the definitions of load balancing [3]. The growth of the network has increased traffic volumes and subsequently reduced network performance and efficiency [1]. Thus, an efficient technique for load management is obligatory. Load balancing is used to decrease the response time, optimise resource utilisation, maximise throughput, and solve the problem of load imbalance between resources [1]. One can do load balancing via a particular application or procedure. One can also do this by installing a specific gateway or load balancer. The traditional load balancing techniques have applied costly tools. They were not desirably accurate in controlling the traffic load [1]. The mentioned restrictions make them inappropriate for large networks. Since the SDN controller has full knowledge about the network, it can perform load balancing efficiently. One can categorise the SDN load balancing into two groups: link load balancing and server load balancing. In the server load balancing, requests are distributed to different servers based on the load balancing strategy to serve more clients with a maximum of throughput and a minimum of latency. Usually, in the conventional route protocols, traffic is forwarded between source and destination based on the shortest path, which may lead to congestion links. It caused some problems such as long transmission delay and increased queueing time. To overcome this obstacle, the link load balancing strategy is useful. Accordingly, in the network, it's very significant to apply link load balancing.

The demand for people to have Internet connectivity everywhere is increasing. To satisfy this demand, more networks are needed. It implies a significant increase in energy consumption. The development of new technologies and communication networks to support all these new applications has increased the consumption of electric energy and carbon emission. Moreover, today, energy consumption in Information and Communication Technology (ICT) is estimated to be 4.7% of the electricity consumption in the world [5]. It could ascend to >10% of the world's electricity consumption by 2025 [5]. It implies that the energy saving is becoming a significant concern. The energy-saving approaches can be categorised into two groups: rate adaptation and ON/OFF methods. In rate adaptation approaches, the network device's electricity consumption depends on the traffic amount that it transports [6]. These methods try to reduce network energy consumption by trading off between a network device's power consumption and the quantity of traffic it carries. ON/OFF approaches save energy by switching off unexploited network devices.

In recent years, SDN has shown that it can improve load balancing and power-saving techniques. Thus, the scheme and

execution of load balancing and power saving methods in the SDN are practicable and effective. In this paper, we have untangled the imbalance links based on the SDN architecture through a link load balancing strategy. Also, the policy tries to minimise network energy consumption by reducing the power consumption of links. Given the NP-complete nature of the load-balancing problem in the SDN [1], we can solve it by a meta-heuristic method. The artificial bee colony (ABC) algorithm [7] and the genetic algorithm (GA) are two meta-heuristic algorithms with distributed computing. ABC is encouraged by the foraging conduct of a honey bee swarm. The standard ABC algorithm is suitable for single-objective optimisation problems. It has good solution quality and a high convergence speed [8]. The multi-objective ABC (MOABC) has been introduced by Martín-Moreno and Vega-Rodríguez [8] to handle the multi-objective (MO) issues. GA is a particular type of evolution algorithm that uses biology techniques such as inheritance and mutation. This algorithm is accorded to the process of natural selection, where the fittest individuals are selected for reproduction to generate offspring of the following generation. The goal of this paper is to present a novel dynamic routing method to solve the link load balancing problem, jointly considering energy consumption, according to the MOABC and GA operators (GAOs) on networks using the SDN controller to exploit the pros of both algorithms (MOABC and GA). The main contributions of this paper are as follows:

- Using SDN technology to perform link load balancing in networks to increase efficiency.
- Developing a MO linear programming (MOLP) for energy-aware link load balancing to optimise a load of links and network energy consumption in SDN with a centralised controller.
- Tackling the problem of energy saving by selecting the forwarding paths between source and destination nodes that minimise the links' power consumption required to route a specific traffic demand.
- Applying the MOABC algorithm and GA operators to provide a new SDN-based energy-aware link load-balancing algorithm.
- Juxtaposing the performance of the introduced method with some link load-balancing algorithms in SDNs.

The remaining part of this paper is organised as follows. Section 2 gives a summary of the recent load-balancing approaches. Section 3 describes the system model and problem definition. Section 4 demonstrates the ABC, MOABC algorithms, and GA operators. Section 5 explains our proposed algorithm for the link load-balancing problem using the MOABC algorithm and GAOs named the MOABC-GAO. Section 6 provides trial and numerical outcomes. Finally, Section 7 concludes the paper and gives some suggestions for the upcoming works.

2 Related work

Moreover, one can improve network and system functionality by controlling the utilisation of resources. We can manage resources by load-balancing techniques. To perform load balancing, a dynamic load balancing method has been presented by Sathyanarayana and Moh [9]. In this method, the best server has been chosen using the minimum loaded server load-balancing procedure. Ant colony optimisation (ACO) has been applied to select the ideal route to get to the nominated server. Therefore, a load-balancing module is created in the SDN controller by a standard dynamic server load-balancing method and ACO routing. Network data and the load of the server that are collected by the controller can be used by the algorithm to acquire the ideal path and the best server for flows of the network. The simulation outcomes have shown that the suggested algorithm has decreased the delay and increased the network throughput as compared to the shortest route round-robin (RR) and shortest route random methods. However, they have not evaluated the overhead and utilisation of the technique. Too, as the technique is based on ACO, having a vast memory is vital. Local optimal can be another obstacle.

Wireless body area networks (WBANs) form a new technology that enables data on critical parameters of a patient's body to be collected by small portable or implantable sensors that communicate using short-range wireless techniques. WBAN has a wide range of health surveillance applications, including the emergency medical response. However, WBANs control and management are difficult due to its complex and heterogeneous structure. Based on the SDN technology, Cicioğlu and Çalhan [10] have proposed WBAN architecture to create more dynamic and flexible network structures in WBANs. Also, for healthcare architecture, they have suggested a new energy-aware routing algorithm. This algorithm has been called SDNRouting [10]. It determines the best route using the Dijkstra algorithm based on the battery levels and signal-to-noise ratio parameters. The results of the experiment have shown that the SDN-based structure has decreased the delay and energy consumption parameters. Also, it has increased the successful transmission rate and network throughput parameters compared to the traditional routing approach. However, low availability, scalability, and system bottleneck are weaknesses of the proposed WBAN architecture because it has a single controller. Also, packets can be lost. It can be due to congestion wireless links because the SDNRouting algorithm does not consider load-balancing parameters.

Furthermore, the rapid rise of Internet operators and software has caused a single server to deal with the needs of the client often ineffectively. Therefore, load balancing is often one solution to increase the performance of the network, service availability, and network scalability. To find the least loaded path, Patil [11] has proposed a load-balancing method. The method has distributed the flow to the minimum loaded route to adjust the load of the network. The method has been applied to several parameters such as latency, packet loss, packet overhead, bandwidth ratio, hop count, and trust in real-time to detect the least loaded path. Simulation results have proved that applying this load-balancing method increases the bandwidth utilisation ratio and decreases the packet loss and transmission latency compared to a random path algorithm. However, the functionality of the proposed method has just been compared to that of the random path one and not with those of the other benchmark algorithms. Moreover, the algorithm overhead has not been evaluated (Table 1).

Also, a link load-balancing method accorded to the ACO has been proposed by Wang *et al.* [12]. In their algorithm, the search rule of the ACO has been used. Ants choose a next node based on pack-loss, link load, and delay as impact factors. For decreasing end-to-end transferring delay and retaining the link load-balancing, ants can choose the shortest and widest route of all. The simulation outcomes have illustrated that using the method has adjusted the link load of the network, reduced the overhead of the network, and ameliorated the quality of service (QoS). However, the algorithm throughput has not been discussed. Also, this method may become trapped in regional optimal. Furthermore, the convergence degree can be reduced because this approach uses the ACO algorithm.

Finally, Xue *et al.* [13] have proposed a genetic-ACO (G-ACO) scheme for load-balancing in the SDN. The ACO algorithm with the crossover, selection, and mutation operation of GA has been combined. The ability of rapid universal exploration of GA and practical exploration of an ideal answer for ACO has been used to enhance the capability of searching an optimal path and calculating the search speed of path. The simulation results have shown that the round-trip time, searching optimal rate, and packet loss rate have been improved compared to those of other algorithms like ACO and RR algorithms. However, they have applied the length of path criterion for finding a path. Other criteria, such as delay and bandwidth, have been ignored. Also, they have used single path routing. Furthermore, throughput, latency, and overhead of the algorithm have not been considered.

3 System model and problem definition

In this paper, an SDN system model (consists of several SDN switches (nodes)) and an SDN controller have been considered. Fig. 1 shows the system model. It is assumed that one flow with the determined amount of traffic is necessary to transfer from one

Table 1 Properties of the selected load-balancing mechanisms

Reference	Technique	Advantages	Disadvantages
Sathyanarayana and Moh [9]	ant colony routing and dynamic server load balancing algorithm	• Increasing network throughput • Decreasing delay	• Not evaluating the overhead and utilisation • Local optimal • Requiring a lot of memory
Cicioğlu and Çalhan [10]	SDNRouting algorithm based on Dijkstra algorithm	• Decreasing delay and energy consumption • Increasing the successful transmission rate and network throughput	• Low availability, scalability, and system bottleneck • Packet loss
Patil [11]	finding the least loaded path	• Increasing the bandwidth utilisation ratio • Decreasing the packet loss rate • Decreasing transmission latency	• Algorithm overhead has not been evaluated • Not comparing the functionality of the method with that of the other benchmark algorithms
Wang *et al.* [12]	ant colony algorithm	• Adjusting the link load of the network • Improving the QoS • Decreasing the overhead of the network	• Local optimal • Not evaluating the throughput • Not evaluating the energy consumption • Rate of convergence may be slow down
Xue *et al.* [13]	G-ACO algorithm	• Improving the packet loss degree and round-trip time • Decreasing searching optimal rate	• Not evaluating the throughput • Not assessing the latency and overhead of the approach • Not estimating the delay and the bandwidth

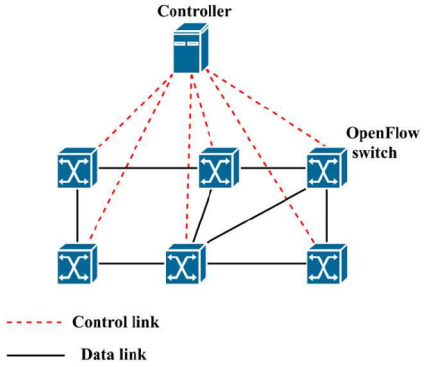

Fig. 1 *System model*

source node to its destination node. An optimal path should be obtained by a routing algorithm where no direct link exists between the source nodes and the destination ones. We have assumed that the flow split is not acceptable.

The purpose of this paper is routing the network traffic so that the minimum network link congestion becomes reachable with the least delay of network and reduces the energy consumption of the network. Minimum network link congestion causes link load-balancing on the network. Here, we have formulated the issue of the minimum network links congestion, a minimum delay of network and minimum network energy consumption.

We have applied a non-directed graph $G=(V, E)$ to describe SDN topology. E is a group of links. V is the group of nodes so that $|V|=N$ is the count of switches. To simplify the problem, it is assumed that a link represents the data connection link connecting two switches. Therefore, we have ignored the control links between switches and controllers. For each link $(i, j)\ \varepsilon\ E$, c_{ij} represents the capacity of the link. In a moment, the traffic demands can be indicated by an ST pair where it is shown by the triple (s_i, t_i, d_i), where s_i and t_i show the source node and the destination node of traffic demand i, respectively. d_i represents the amount of traffic between s_i and t_i. Usually, there are K traffic demands $STs=\{(s_i, t_i, d_i),\ i=1,\ 2,\ \ldots,\ K\}$ in the network. The symbols used in the equations are shown in Table 2. Based on the explanation above, the following MOLP model can be obtained:

Objective function: Minimise $\Theta\ (\Phi, \Omega, \Psi)$

Where

$$\Phi = \sum_{\forall (i,j)} \varphi_{ij} x_{ij} \tag{1}$$

$$\Omega = \frac{1}{K} \sum_{i=1}^{K} \text{delay}_p \tag{2}$$

$$\Psi = \sum_{\forall (i,j)} E_{ij} \tag{3}$$

Subject to

$$\varphi_{ij} = \begin{cases} l_{ij} & 0 \le l_{ij}/c_{ij} \le 1/3 \\ 3l_{ij} - \frac{2}{3}c_{ij} & 1/3 \le l_{ij}/c_{ij} \le 2/3 \\ 10l_{ij} - \frac{16}{3}c_{ij} & 2/3 \le l_{ij}/c_{ij} \le 9/10 \\ 70l_{ij} - \frac{178}{3}c_{ij} & 9/10 \le l_{ij}/c_{ij} \le 1 \end{cases} \quad \forall i, j \tag{4}$$

$$\text{delay}_p = \sum_{i=1}^{N} \sum_{j=1}^{N} \text{delay}_{ij} x_{ij}^{st} \quad \forall i, j, \forall s, t \tag{5}$$

$$l_{ij} = \sum_{s=1}^{N} \sum_{t=1}^{N} f_{ij}^{st} \quad \forall i, j, \forall s, t \tag{6}$$

$$l_{ij} \le c_{ij} x_{ij} \quad \forall i, j \tag{7}$$

$$E_{ij} = E_{ij}^{\text{Idle}} + E_{ij}^{\text{Load}} \times \frac{l_{ij}}{c_{ij}} \tag{8}$$

$$\sum_{j=1}^{N} f_{ij}^{st} - \sum_{j=1}^{N} f_{ji}^{st} = \begin{cases} d^{st} & \forall s, t, i = s \\ -d^{st} & \forall s, t, i = t \\ 0 & \forall s, t, i \ne s, t \end{cases} \tag{9}$$

$$x_{ij}, x_{ij}^{st} \in \{0, 1\}$$

$$c_{ij} > 0, d^{st} > 0, f_{ij}^{st} > 0, E_{ij}^{\text{Idle}} > 0, E_{ij}^{\text{load}} > 0$$

Minimising network congestion is our first optimisation objective that causes link load-balancing on the network. The second optimisation objective is minimising network delay. The third optimisation objective is reducing network energy consumption. We can obtain network congestion by (1). Network delay is calculated by (2) where is the average of paths delay. Network energy consumption is obtained by (3). It is the sum of energy consumption of links. We have obtained congestion of a link using

a model proposed by Fortz and Thorup [14]. It is a function of link utilisation [15]. Links with the heavy load are penalised by this model, as shown by φ_{ij} in (4). The overall delay of a routing route consecrated to request from source node s to the destination node t has been calculated by (5). On each link, the total routed flow has been obtained by formula (6). Equation (7) is for calculating the link load constraint. The link load should not be higher than the link capacity. The energy consumption of networking elements is constructed by dynamic and static (idle) components [16]. The static component is due to the energy consumed by line-cards, fans, chassis, etc. It demonstrates the power needed by a not used element. The dynamic one is depended on the amount of traffic flowing through their port interfaces [16]. If the traffic load is increasing, the energy consumption must treat proportionally. With the traffic raise, it linearly generates [16]. We have attained energy consumption of a link using a model proposed by Huin [17]. Equation (8) is for estimating the link energy consumption. The energy consumption of a relationship has two sections. The first is a baseline energy consumption (static/idle component), i.e. the power consumed when a link is switched on. The other is additional energy proportional to link load (dynamic part), i.e. the extra power used by a link when it is wholly capacitated. Equation (9) illustrates the flow conservation constraints. It guarantees that the routed demands are starting with their source nodes and ending in their destination nodes.

This model is a MOLP issue. The MOLP will remain unanswered in the polynomial period because it has an NP-complete nature. It means that, in large-scale networks, a lot of time is needed to find a feasible routing solution. Therefore, network efficiency will be affected. As a result, a heuristic algorithm shall be proposed to tackle this problem.

4 Overview of the ABC and MOABC algorithms and GA operators

The MOABC-GA has been proposed to obtain an ideal path for routing traffic between the source and the destination nodes, to deal with the routing problem mentioned before. The introduced method benefits from both the MOABC method and GA operators. Section 4.1 describes ABC and MOABC algorithms. GA operators are explained in Section 4.2.

Table 2 Applied symbols in this paper

Name	Description	Name	Description
G	SDN topology	f_{ij}^{st}	amount of traffic among node s and node t that passed via a link (i, j)
V	set of the switches	l_{ij}	load of a link from i to j
E	set of the links	φ_{ij}	amount of congestion a link (i, j)
s_i	source node of traffic demand i	Φ	network congestion
t_i	destination node of traffic demand i	Ω	network delay
d_i	amount of traffic between source and destination nodes of traffic demand i	delay_{ij}	delay of a link (i, j)
c_{ij}	capacity of a link (i, j)	delay_p	delay of a path p
x_{ij}	1 if the link (i, j) is applied, else 0	E_{ij}	energy consumption of a link (i, j)
x_{ij}^{st}	1 if there's a link (i, j) using traffic request from source node s and destination node t, else 0	E_{ij}^{Idle}	baseline electricity consumption of a link (i, j)
d^{st}	traffic request among source node s and destination node t	E_{ij}^{load}	linear energy coefficient of a link (i, j)
Ψ	network energy consumption		

4.1 ABC and MOABC algorithms

Karaboga [18] has proposed the ABC, which is a swarm intelligence method. It is according to the conduct of bees' swarm during two natural procedures: the employment of bees for the exploitation of food sources and the neglect of exhausted ones.

In the ABC algorithm, we can classify the bees in a colony into three sets: employees, onlookers, and scouts [19].

Employed bee: The count of employed bees is half the individuals of the colony (colony size). Their initial positions in the search space have been randomly defined. The employed bees' number is equivalent to the food sources (solutions) number, and each employee has been assigned to one of them. Each employee detects a new food source from the present one and maintains the best one [19]. The novel food source is better than the current one if the novel one has a better nectar amount than the current one. When the search process has been terminated, the employees give their data to the onlookers using the waggle dance. Each employed bee uses the following equation to produce novel food source v_i from current one x_i:

$$V_{ij} = X_{ij} + \varnothing_{ij} \times (X_{ij} - X_{kj}) \tag{10}$$

where X_{ij} is the jth parameter of the ith food source, NS is the count of food sources, k and i but are two different values in the range of values [1, 2 ,..., NS]. In addition, D indicates the dimension of the optimised problem. The $j = \{1, 2, ..., D\}$, so j and k have arbitrarily been produced. The symbol $\varnothing_{ij}$ represents an arbitrary amount between [−1, 1].

Onlooker bee: The count of onlookers is equal to employees. They select a food source, according to the data shared by employees via waggle dance, where the food source having the ideal nectar amount (amount of the objective function) has been chosen [19]. An onlooker is responsible for finding new food sources about his assigned food source with a high nectar value. Each onlooker chooses a food source randomly. A possibility value of p_i related to the solution x_i has been calculated by (11) to calculate the selection probabilities of each food source

$$p_i = \frac{f_i}{\sum_{i=1}^{NS} fi} \tag{11}$$

where f_i is the nectar amount (fitness of solution) of solution x_i.

After choosing a food source by each onlooker, a novel food source will be generated by (10). If this novel one would be more useful than the present food source, the present one has been substituted by it.

Scout bee: In the colony, the number of scout bees is not described. They produce a new food source in randomly to replace the current one that has not been enhanced. When a food source has not gotten better after a specific count of cycles (limit), it is neglected and substituted by another one source found by a scout bee [19]. Therefore, the employee-related to this food source turns a scout bee to explore for a novel one using the following equation:

$$V_{ij} = X_{\min j} + rand[0, 1] \times (X_{\max j} - X_{\min j}) \tag{12}$$

where $x_{\min j}$ is the lower bound in the jth aspect of the issue, space and $x_{\max j}$ is its upper bound.

The ABC is a single objective optimisation algorithm, but our problem is a MO optimisation problem that has three objective functions. Therefore, we have to use a version of the ABC that can solve the MO optimisation problem. A traditional one-dimensional comparison that indicates whether one solution is better than another is not useful. Consequently, we can use the Pareto dominance concept and crowding distance to select a better solution. In the Pareto dominance concept, objective vector $\boldsymbol{y}_1$ dominates another vector $\boldsymbol{y}_2$ ($\boldsymbol{y}_1 \succ \boldsymbol{y}_2$) if and only if $\boldsymbol{y}_1$ is not worse than $\boldsymbol{y}_2$ in all goals, and if it is quite better than $\boldsymbol{y}_2$ in minimum in one.

MOABC has the following differences compared with ABC:

Calculate the rank of current population solutions using rank operators.
If |solutions with the least rank|<= |repository|
  Store solutions with the least rank into the repository.
Else
  Calculate crowding distance for solutions with the least rank.
  Depending on the size of the repository, store some of the solutions with the least rank and the highest crowding distance into the repository.

Fig. 2 *Quasi-code of storing the best solutions into the repository*

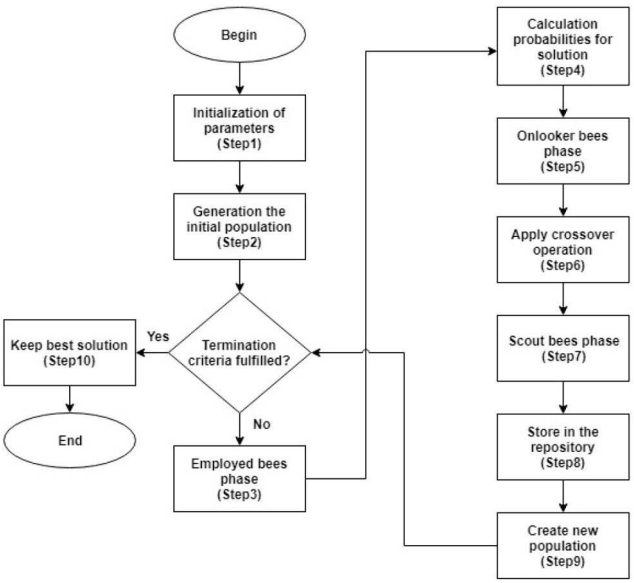

Fig. 3 *Flowchart of the MOABC-GAO algorithm*

i. The MO operators crowding and rank [20] have been used to perform fitness evaluations because multiple objectives exist. According to the Pareto dominance concept, the solutions have been ranked in different Pareto fronts using the rank operator. To have more diverse solutions, the crowding distance concept has been used by the crowding operator. These two MO operators have been explained in [20]. By using these operators, (13) [8] calculates the fitness function of each solution $\boldsymbol{x}_i$.

$$MO-\text{fitness}(x_i)=\frac{1}{2^{\text{rank}(x_i)+(1/(1+\text{crowding}(x_i)))}} \quad (13)$$

ii. To calculate the selection probabilities of each solution, a probability value of the solution $\boldsymbol{x}_i$ has been calculated by (14) [8]

$$\text{MO}-p(x_i)=\frac{\text{MO}-\text{fitness}(x_i)}{\sum_{m=1}^{NS} MO-\text{fitness}(x_m)} \quad (14)$$

iii. In employed and onlooker bees' phases, after generating a new solution, it has been juxtaposed with the present one by the dominance concept. If the new solution dominates the current one, the new one is maintained.

iv. To preserve the best solutions ever found, a fixed size repository has been used. On each iteration, the obtained non-dominated solutions have been stored into a repository to retain them while new solutions are produced. In this process, first, all solutions have been classified into dominated or non-dominated. After this, all non-dominated solutions have been stored in the first frontier. First, frontier solutions have been separated from all solutions. The remaining solutions have been classified into dominated or non-dominated. Then, all non-dominated solutions have been stored in the next frontier. The ranking process has been continued, so all solutions are classified. Now, the best solutions have been stored in the first frontier. If the first frontier would not be empty and its size would be smaller or equal to the repository size, the first frontier solutions have been stored in the repository. However, if the first frontier would not be empty and its size would be larger than the size of the repository, the first frontier solutions have been ordered based on crowding distance and some of them equal to the size of the repository, are stored in the repository. Otherwise, if the first frontier would be empty, the other frontier solutions have been selected, and the process has continued. Fig. 2 shows the quasi-code of this process.

v. Selecting a new population for the next iteration should be done. On each iteration, first, all solutions have been ordered by rank and crowding operators. Then, the best *NS* (size of the population) solutions, that have the least level and highest crowding distance, have been retained as the new population for the subsequent iteration.

4.2 Genetic operators

To search for optimisation problems, according to Darwin's natural selection theory, an algorithm has been suggested by Koza [21] that is named genetic algorithm. GA is an artificial intelligence technique [19]. The GA uses statistical methods to generate and choose the best solutions during the iterative optimisation process. The essential idea is to diversify a population of individuals who, at the end of the process, should be more adapted to the objective function constraints [19]. An individual is a possible solution to the issue characterised by a chromosome. In our problem, a path among the source nodes and destination ones has been introduced by each chromosome. Each chromosome, includes some genes that show a network node (switch) in the chromosome in our problem. The chromosome length is the length of the path. At each iteration (or generation) of the algorithm, genetic operations are used to generate individuals with greater aptitude. These operations are selection, crossover, mutation, and elitism. In this paper, among these operators, we will use selection, crossover, and mutation operators.

Selection: It is used for choosing solutions (individuals) for regeneration grounded on their fitness.

Crossover: This operator recombines two selected individuals to produce off-springs by sharing the genes of both of them. The main idea of the crossover operator is based on the fact that if two individuals are taken correctly, the obtained off-springs have higher fitness than each of the parents, separately [19]. The GA crossover operator has retained the population diversity. The efficiency of regional exploration can be ameliorated by MOABC multi-dimensional regional exploration tactics. There are many crossover algorithms like a 1-point crossover, 2-point crossover, and uniform crossover. In the 1-point crossover, one crossover point is selected randomly and the tails of its two parents are exchanged to make new off-springs. In our proposed method, a 1-point crossover has been used.

Mutation: In the mutation process, one or more gene values of an existent solution have been chosen randomly and modified to generate a new solution.

5 Proposed hybrid method

In this section, the MOABC algorithm based on the GAO has been used to find an optimal path to route traffic between the source nodes and the destination ones. To prevail problems such as local optimal and slow convergence, the suggested combined technique has to reinforce the MOABC using GA operators. The steps of the suggested method are shown in Fig. 3. The following steps construct the suggested method.

Step 1 (Initialisation of parameters): Initial factors like colony size, crossover rate (P_c), maximum execution time (*MET*), repository size, and limit have been set. (population size (*NS*) = count of employees = count of onlookers = colony size/2).

Step 2 (Generation the initial population): The primary population consists of *NS* solutions. Each solution is a path between the source nodes and the destination ones. This step generates the initial population with *NS* solutions. Fig. 4 indicates the quasi-code of the

```
Let G=(V, E) be a non-directed graph, and p be an array.
Let s be source node and d be the destination node.
p=s
SearchPath(p, d)
  x is the last node of p.
  For each edge xy for some y in E
    If y is not in p
      If y=d
        p=p+d
        Return p as a path
      Else
        SearchPath(p+y, d)
```

Fig. 4 *Quasi-code of the algorithm that generates a path between source and destination nodes randomly (Algorithm 1)*

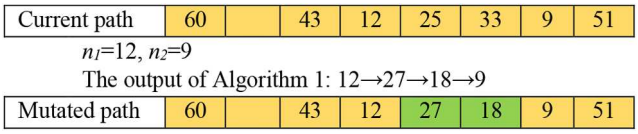

Fig. 5 *Example of the mutation process*

```
For i=1 to NS/2
  Generate a real number rc in the range [0, 1] randomly.
  Select the best path (pbest) of the population based on
the rank and crowding operators.
  If rc<= Pc
    Select a path pj from the population randomly.
    Perform crossover on the pbest and pj.
    If each of off-springs produced from crossover on
the pj and pbest dominates pj or pbest, replace the worst
parent with the best new offspring.
  End if
End for
```

Fig. 6 *Quasi-code of the crossover operation (Algorithm 2)*

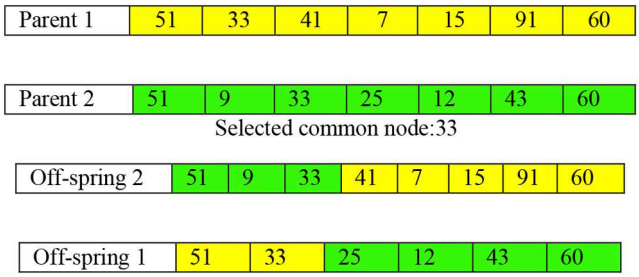

Fig. 7 *Example of a crossover process*

proposed algorithm (Algorithm 1). It creates a route among the source nodes and the destination ones randomly. The algorithm runs *NS* times to produce the initial population.

Step 3 (Employees stage): A regional exploration in the neighbourhood of the corresponding solution has been done by each employed bee. In the population, the employee j is associated with the path (solution) j. In the employed bees' phase, the mutation operator has been applied as a regional exploration. The mutation of the current route has been done as

i. Except for the first and the last nodes of the current path, two points n_1 and n_2 are selected randomly.
ii. q is a random path between n_1 and n_2 produced by Algorithm 1.
iii. Replace all points between n_1 and n_2 with q.

Fig. 5 shows an example of the mutation process.

Now, if the current path is dominated by the novel route outcomes in a mutation, the new path will be maintained. Otherwise, the new path will be ignored. To compare two paths by means of the dominance concept, we need to design fitness functions. We have designed different fitness evaluation functions of paths, for different optimisation objectives, shown in the following equations:

$$\text{Fitness}(p_i)_{\varphi} = \text{e}^{-\varphi_{pi}} \tag{15}$$

$$\text{Fitness}(p_i)_{\text{delay}} = \text{e}^{-\text{delay}_{pi}} \tag{16}$$

$$\text{Fitness}(p_i)_{E} = \text{e}^{-E_{pi}} \tag{17}$$

where $\text{Fitness}(p_i)_{\varphi}$ is the congestion fitness function of the path p_i. φ_{pi} represents the congestion of path p_i and it is the maximum congestion of links related to the path p_i. A path having lower congestion obtains better fitness. $\text{Fitness}(p_i)_{\text{delay}}$ is the fitness function regarding the delay of the path p_i. Delay_{pi} represents the delay of path p_i. It is the sum of the delay of links related to the path p_i. A path having lower delay obtains better fitness. $\text{Fitness}(p_i)_E$ is the energy consumption function of path p_i. E_{pi} shows the energy consumption of path p_i. It is calculated by the sum of the power consumption of links related to the path p_i. A path having lower energy consumption obtains better fitness. More quickly than linear function, an exponential function can reverberate a small variation of the variables. Therefore, we have used the weighted value of exponential functions. Since the OpenFlow protocol has not measured the link delay by itself, we need to use the mechanism proposed by Francois and Gelenbe [22] to specify link delay. For simplicity, we have supposed that the latency of controller links is zero. The mechanism has two stages to compute the delay of the link between N_s and N_d with source and destination port P_s and P_d, respectively:

- *Stage 1:* An OpenFlow packet-out message is sent to node N_s by the SDN controller. The source and destination media access control (MAC) address of the packet-out message is assigned to the MAC address of the port P_s and port P_d, respectively. The packet-out message also includes the single action of outgoing the packet on the port P_s. To recognise the packet as a probe packet via the SDN controller, an arbitrary value such as 0x08c5 is selected to set the packet Ethernet protocol type. The time T_t it takes for the SDN controller to receive the probe packet from N_d is measured. It is the delay between N_s and N_d.
- *Stage 2:* To calculate the delay of all links, the SDN controller periodically sends probe packets to all links as in Stage 1.

Step 4 (Calculating probabilities for solution): Calculating probabilities of each solution have been presented in Section 4.1. Here, the selection probabilities of each path (employed bee) are calculated by (14).

Step 5 (Onlooker bees' phase): By applying the roulette wheel method, depending on the probability associated with each path, each onlooker selects one route to do a regional exploration on this route. Remember, local exploration is performed by the mutation operator. It is explained in Step3. If the different route outcomes in a mutation dominate the current path, the new way would be memorised. Otherwise, the original path would be neglected.

Step 6 (Apply crossover operation): In this step, based on crossover probability P_c, crossover operation is performed on the best path of the population and randomly selected the way. It has retained the balance between intensification and diversification [19]. The population is diversified by crossover operation. Fig. 6 shows the quasi-code of the crossover algorithm (Algorithm 2).

We have used the 1-point crossover to perform a crossover operation between two paths. In the 1-point crossover process, first, one common point (node) has been chosen except for the starting nodes and ending ones among parents. After that, the off-springs have been generated by replacing all nodes after the common node in parents. Fig. 7 shows an example of a 1-point crossover process.

```
Begin
Input: colony size, limit, Pc, Maximum Execution Time (MET), repository size;
Output: Best solution;
NS=number of employed bees=number of onlooker bees=colony size/2;
Generate NS solutions by Algorithm 1;
For i=1 to MET
 ++ The Phase of Employed Bees ++
    FOR(each employed bee)
      Generate a new solution from the current solution by mutation operator;
      Evaluate the fitness functions of the new and present solutions by Equations 15,16,17;
      If the new solution ≻ current solution then
         Maintain the new solution and ignore the current solution;
    END FOR.
 Calculate the probability MO − p(x_i) for each solution using Equation 14;
 ++ The Phase of Onlooker Bees ++
   FOR(each onlooker bee)
     Choose one of the solutions depending on the value of MO − p(x_i) by using the roulette wheel;
     Generate a new solution from the selected solution by mutation operator;
     Evaluate the fitness functions of the new and selected solutions by Equations 15,16,17;
     If the new solution ≻ selected solution then
        Maintain the new solution and ignore the selected solution;
   END FOR
 Perform crossover operation with probability Pc by Algorithm 2;
 **Scout Bees' Phase**
   If trial counter of a solution>limit then
      Abandon the solution and replace it with a new solution obtained by Algorithm 1;
 Store best solutions into the repository using the rank and crowding operators;
 Create a new population;
END FOR
Obtain the best solution of the repository using rank and crowding operators;
End.
```

Fig. 8 *Quasi-code of the MOABC-GAO algorithm*

Table 3 Network simulation parameters

Name of parameter	Value
SDN protocol	OpenFlow 1.3.0 [25]
SDN controller	Ryu OpenFlow controller [24]
data plane	Mininet [23]
MAC protocol	IEEE 802.3 [26]
number of OpenFlow switches	Topology 1:10 switches Topology 2: 14 switches
link bandwidth	100 Mbps
link propagation delay	0 (default value) [23]
link type	TCLink [27]
switch type	OpenFlow Switch [23]
E^{idle} (Energy consumed by the link during idle state)	30 W [28]
data rate	5 Mbps
testing tool	iPerf [29]
simulation time	600 s

Step 7 (Scout bees' phase): When a path cannot be enhanced after certain repetitions (limit), it is neglected. The path is substituted by a new path obtained by Algorithm 1.

Step 8 (Store in the repository): To preserve the best paths ever found, a fixed size repository is used. On each repetition, the obtained non-dominated paths are stored in a repository. The process of saving the best paths in the repository has been explained in Fig. 2.

Step 9 (Create a new population): New population should be created for the next iteration. This process has been explained in Section 4.1.

Step 10 (Keep the best solution): The best path of the repository is obtained based on the rank and crowding operators.

Fig. 8 indicates the quasi-code of the MOABC-GAO algorithm.

6 Experimental results

In this section, a series of simulations have been done to evaluate the performance of MOABC-GAO. The MOABC-GAO has been compared with the G-ACO [13], basic ACO [13], SDNRouting [10], and RR to confirm the efficacy of our proposed design. Section 6.1 describes the simulation setting. The topology of simulation is shown in Section 6.2. The simulation parameters are described in Section 6.3. The performance metrics used in the article are explained in Section 6.4. Finally, the outcomes are illustrated in Section 6.5.

6.1 Simulation environment

Multiple controllers and emulators exist to use in the SDN to decouple control and data planes. We have applied the Mininet emulator [23] and the Ryu SDN controller [24] for setting up the switch and the controller, respectively. Using the OpenFlow protocol [25, 26], the Ryu SDN controller will interact with emulated OpenFlow switches in the Mininet. Network simulation parameters have been shown in Table 3. Link propagation delay and link bandwidth are set to the default value (0) and 100 Mbps, respectively. In this experiment, Ryu and Mininet have been installed on a virtual machine. The virtual machine has been installed on a laptop. The specifications of the laptop are Intel Core I7-7700 CPU, 2.80 GHz, 8 GB of RAM. The operating system of the laptop is Windows 10. MiniEdit is a Graphical User Interface (GUI) editor for Mininet. This tool can create different network simulations, configure the network elements, and save the topology. We have used MiniEdit to create test topology.

6.2 Topology of simulation

Recently, many data centre networks have applied the fat-tree topology to increase fault tolerance and bandwidth. Many advantages of the fat-tree have been described by Jo *et al.* [27]. Therefore, a fat-tree topology has been applied to confirm the performance of the MOABC-GAO. Here, the proposed fat-tree topology has ten switches and one controller. Also, each switch has

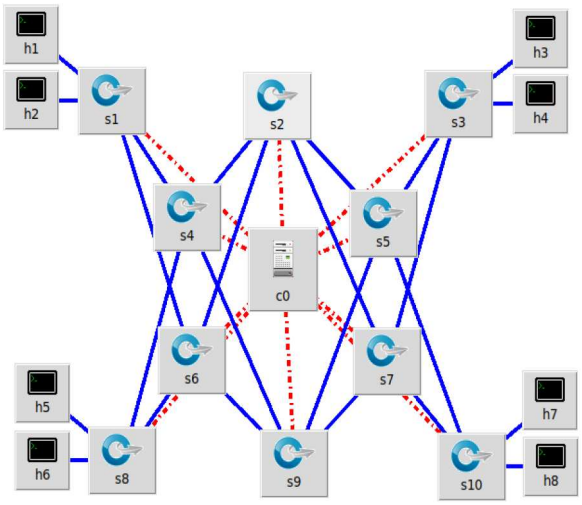

Fig. 9 *First used topology (Topology 1) in this paper in the MiniEdit environment*

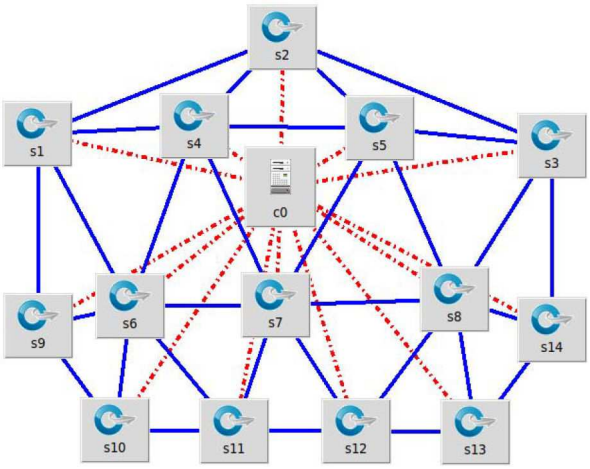

Fig. 10 *Second used topology (Topology 2) in this paper in the MiniEdit environment*

Table 4 Simulation parameters of the MOABC-GAO

Name of parameter	Value
colony size	40
limit	10
maximum execution time	150
repository size	10
P_c	0.7

TCP commands:
To begin the server (host h2): iperf -s 10.0.0.2 -i 60
To begin flow in client (host h1): iperf -c 10.0.0.2 -w 130k -i 60 -t 600 -e
UDP commands:
To begin the server (host h3): iperf -s –u 10.0.0.3 -i 60
To begin flow in client (host h1): iperf -c 10.0.0.3 -u -i 60 -t 600 -b 5

Fig. 11 *Some of the used iperf commands*

two hosts. It is shown in Fig. 9. Moreover, for a more extensive evaluation, the method is also evaluated by a topology with fourteen nodes. The topology is illustrated in Fig. 10 [30].

6.3 Simulation parameters

For parameters, preliminary tests have been done using a spectrum of amounts. After that, we have assigned MOABC-GAO configuration factors, as shown in Table 4.

6.4 Performance metrics

Five parameters have been applied to evaluate the functionality of the MOABC-GAO, explained in the following:

Packet loss rate (PLR): If one or more data packets fail to attain their target, packet loss will occur [1, 3]. The performance of load-balancing can be evaluated by the PLR. The smaller PLR shows that the network links have been applied optimally. It means that the congestion of the network is less. A link will be congested if the link load is unbalanced. It means that when congested links encounter burst traffic, packets will be lost. The PLR percentage of a particular node has been calculated by (18). In the equation, Ps is the count of packets directed by the source node. P_R is the count of packets gained by the destination node.

$$\text{PLR percentage} = \frac{P_S - P_R}{P_S} \times 100 \tag{18}$$

Round trip time (RTT): It is the duration from when the source node directs a packet to the destination node to when the source node receives a response from the destination node. Many factors exist affecting the RTT, such as the interval among the source nodes and destination ones, the count of network hops, queuing delay, etc. One of the main factors influencing the RTT is network congestion.

RTT increases when a network is congested. i.e. if the traffic is sent by congestion paths, RTT will be increased. Consequently, paths delay and congestion of links impress RTT.

Average jitter: A variation in the delay of the received packets is defined as jitter. Packets are transmitted in a continuous stream by the source node. It spaces them evenly apart. Usually, the arrival time of packets is varied. It is due to the congestion of the network. Therefore, jitter can be used as a metric for determining network congestion. Also, it is suitable for evaluating link load-balancing. Jitter is calculated by measuring the delay for all the packets from a source node to a destination one and the gain of the statistical variance of the same.

Energy consumption: It is the sum of the electricity consumption by links in the network. Network energy consumption needs to be minimised. We suppose the power consumption of a control link is zero. The electricity consumption of a data link can be calculated by (8). Based on Cisco characteristics [28], links are using 30 W as the baseline and become 40 W if they have full capacity, i.e. $E_{ij}^{\text{Idle}} = 30\,\text{W}$, $E_{ij}^{\text{load}} = 10\,\text{W}$.

Run time: Traditionally, it is the quantity of time it takes to run an algorithm. We calculate the run time of a method by measuring the amount of time it takes to specific traffic volume transfers from a host to another host.

6.5 Simulation results

For simplicity, we have assumed that all links capacities in the topology of the simulation are set to 100 Mbps. Also, it is assumed that the capacity of the nodes is sufficiently large to prevent traffic congestion. The simulation run takes 600 s using the iperf software tool [29]. For simulation, the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffics have been generated. For example, some of the used iperf commands have been illustrated in Fig. 11. The proposed MOABC-GAO, ACO, SDNRouting, RR, and G-ACO methods have been simulated. Based on Fig. 9, the node $h1$ has been selected as the source node of the packet and $h2$, $h3$, $h4$, $h5$, $h6$, $h7$, and $h8$ nodes have been selected as the destination nodes of the packets, respectively. PLR and RTT have been gathered for $h1$ node. The simulation outcomes are indicated in Figs. 12 and 13.

Fig. 12 illustrates that the RTT mainly has been better reduced using the suggested MOABC-GAO method compared to the other

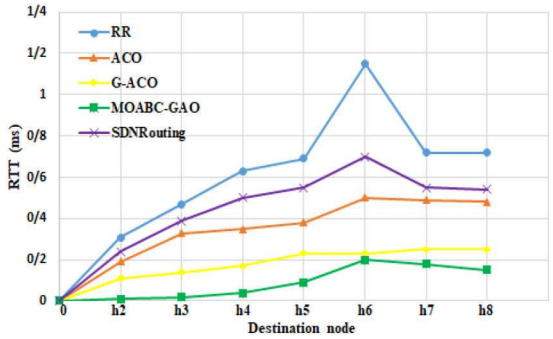

Fig. 12 *RTTs comparison*

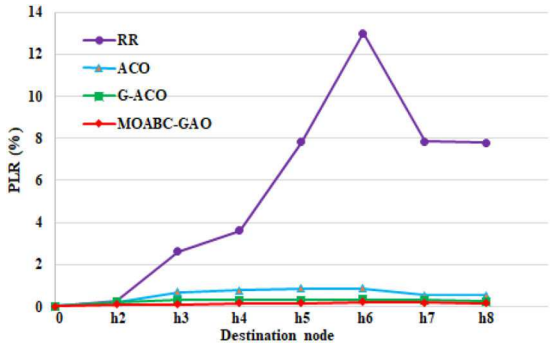

Fig. 13 *PLRs comparison*

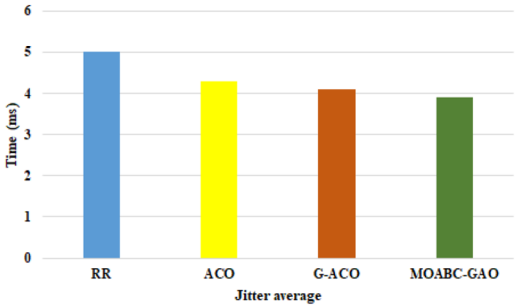

Fig. 14 *Jitter average comparison*

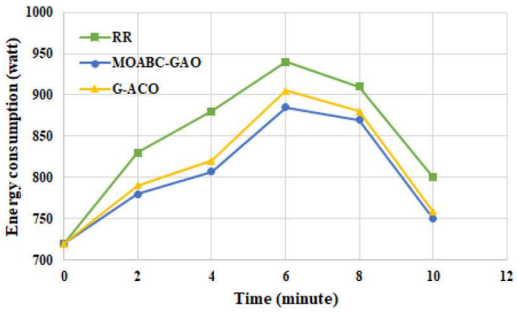

Fig. 15 *Network power consumption of three methods*

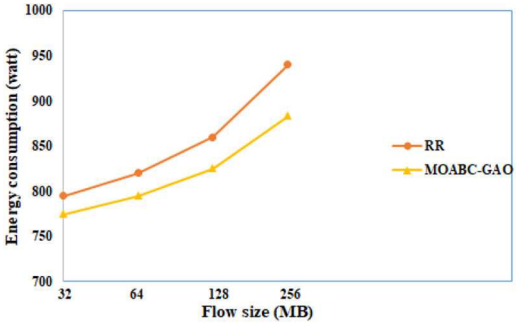

Fig. 16 *Network power consumption via the flow size*

three methods. It can be due to the fact that the routing strategy of our method considers congestion and delay of routing paths, whereas the only length of routing paths is applied to find ways by SDNRouting, ACO, and G-ACO algorithms. Also, RR has not used any of these parameters. Compared to RR, SDNRouting, ACO, and G-ACO have improved the RRT parameter. RR has the worst value of RTT between five compared methods. The RTT has the maximum value in node 6. It can be due to the load of node1 to node 6 that causes congestion on the path.

Fig. 13 shows that four methods are approximately equal in PLR at the beginning. Compared to the ACO and G-ACO methods, the suggested MOABC-GAO scheme has slightly reduced the PLR. However, this parameter has been significantly reduced compared to that of the RR scheme due to the fact that the routing tactic of the RR has not considered the congestion parameter. The packet loss of our load-balancing algorithm is lower than those of the ACO and G-ACO algorithms. It is because the traffic has been distributed more reasonably to the paths than in the other two algorithms.

Figs. 12 and 13 show that the RTT and PLR charts are rising and afterward diminishing. It can be due to the fact that $h1$ respectively sends traffic to $h2$, $h3$, …., and $h8$ using iperf commands. Therefore, with increasing traffic volume, the network congestion increases. As a result, the RTT and PLR increase. The completion of some iperf commands has decreased the traffic volume and network congestion. It can be the reason for reducing RTT and PLR.

To measure average jitter, after node $h1$ sends UDP traffic to all destination nodes ($h2$–$h8$), the jitter of all destination nodes has been collected and averaged. Fig. 14 illustrates the average jitter of the four methods. It shows that ACO, G-ACO, and RR schemes have a more significant jitter than the MOABC-GAO. It can because of the fact that these schemes have not considered the load of links to route traffic. Therefore, link congestion can occur and then jitter increasing.

By sending UDP traffic, the energy consumption of the network has been measured every 2 min in the total of the simulation run (10 min). Fig. 15 shows the network energy consumption of the three methods. It is shown that with increasing traffic volume, the energy consumption of three methods increases. Then, with a decreasing quantity of traffic, the power consumption of them drops. It is because of the power consumption of links with increasing traffic volume and vice versa. Fig. 15 illustrates that the MOABC-GAO algorithm has the lowest energy consumption, and RR has the biggest energy consumption. The result shows that the proposed method can decrease network power consumption. The reason is that the RR and G-ACO do not consider link energy consumption parameters when they find paths. However, the proposed method selects the route with the minimum energy consumption. Due to the high volume of traffic on links and nearing parameter l_{ij} to c_{ij}, the RR has the most significant power consumption.

To evaluate the impact of traffic load on network energy consumption, we consider four flow sizes: 32, 64, 128, and 256 MB. Fig. 16 shows the network energy consumption via the flow size in the MOABC-GAO and RR methods. MOABC-GAO can effectively reduce network power consumption at the peak time of traffic. When flow size is bigger, links use more energy electricity because they need to work with higher utilisation to support the heavy traffic.

For evaluating the effect of the network size in the performance of the suggested method, by using the topology that has been shown in Figs. 9 and 10, the run time of the three mentioned methods have been measured. Here, the source and destination nodes are 1 and 10 in the Topology 1. Also, in Topology 2, the source and destination nodes are 1 and 13. Moreover, file data are sent at a rate of 1 Mbps. Fig. 17 illustrates the run time of MOABC-GAO, basic ACO, and G-ACO methods. It shows the presented scheme diminishes the run time parameter compared to the G-ACO and basic ACO methods. Also, the impact of increasing the network size on the ACO run time is higher than G-ACO and MOABC-GAO methods.

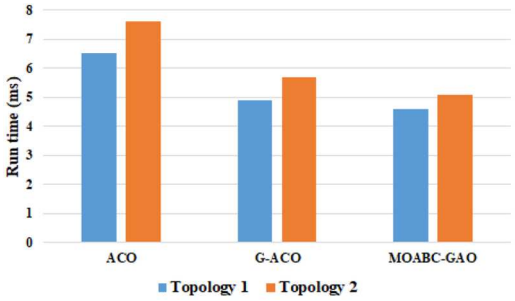

Fig. 17 *Comparison of three methods running time with the Topologies 1 and 2*

7 Conclusion and future work

In the last decade, the growth of the network has increased traffic volumes and power consumption. Due to this, the researchers' attention to energy saving and load balancing subjects. Burst traffic will bring packet loss, congestion, and delay on the link when the network is unbalanced. The purpose of this paper was finding an optimal path to route traffic in the SDN. The path with minimum congestion, delay, and the energy consumption is an optimal path. These three objectives have been optimised at the same time. A MOLP model for optimisation of load balancing and energy saving has been formulated, too. To solve this NP-complete problem, an energy-aware routing method has been presented. It has been called the MOABC-GAO algorithm. This method has been used the MOABC algorithm and GA operators to find an optimal path. Link congestion, link delay and power consumption of the link parameters have been used by the MOABC-GAO method to select the path with the least congestion, delay, and energy consumption for routing dynamic traffic. The simulation outcomes have indicated that the introduced MOABC-GAO has improved packet loss rate, RTT, and jitter metrics compared with ACO, G-ACO, and RR schemes. Also, the proposed method has consumed less energy in comparison to G-ACO and RR schemes.

The scheme is grounded on a single-controller SDN. For the upcoming works, proposing a scheme for link load-balancing in the multi-controller SDNs will be an interesting subject. Also, data centre networks have many network devices such as routers, switches, links, and so on. Therefore, in the SDN data centre networks, surveying the impact of diminishing the number of active network devices for power saving is an attractive line for future trends. Moreover, other meta-heuristic algorithms such as grey wolf, bat, and whale optimisation algorithms can be used to suggest a new load-balancing algorithm.

8 References

[1] Neghabi, A.A., Navimipour, N.J., Hosseinzadeh, M., *et al.*: 'Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature', *IEEE Access*, 2018, **6**, pp. 14159–14178

[2] Yaghmaee, M.H., Neghabi, A.A.: 'Extended gradient: a modified gradient algorithm for node placement in shufflenet network'. 11th Annual Conf. of Computer Society of Iran, Iran, 2005

[3] Akbar Neghabi, A., Jafari Navimipour, N., Hosseinzadeh, M., *et al.*: 'Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network', *Int. J. Commun. Syst.*, 2019, **32**, (4), p. e3875

[4] Al-Hubaishi, M., Çeken, C., Al-Shaikhli, A.: 'A novel energy-aware routing mechanism for SDN-enabled WSAN', *Int. J. Commun. Syst.*, 2019, **32**, (17), p. e3724

[5] Ding, Z., Xing, S., Yan, F., *et al.*: 'An interference-aware energy-efficient routing algorithm with quality of service requirements for software-defined WSNs', *IET Commun.*, 2019, **13**, (18), pp. 3105–3116

[6] Andrews, M., Anta, A.F., Zhang, L., *et al.*: 'Routing for energy minimization in the speed scaling model'. 2010 Proc. IEEE INFOCOM, San Diego, CA, USA, 2010, pp. 1–9

[7] Patil, S.D., Ragha, L.: 'Adaptive fuzzy-based message dissemination and micro-artificial bee colony algorithm optimised routing scheme for vehicular *ad hoc* network', *IET Commun.*, 2020, **14**, (6), pp. 994–1004

[8] Martín-Moreno, R., Vega-Rodríguez, M.A.: 'Multi-objective artificial bee colony algorithm applied to the bi-objective orienteering problem', *Knowl.-Based Syst.*, 2018, **154**, pp. 93–101

[9] Sathyanarayana, S., Moh, M.: 'Joint route-server load balancing in software defined networks using ant colony optimization'. 2016 Int. Conf. on High Performance Computing & Simulation (HPCS), Innsbruck, Austria, 2016, pp. 156–163

[10] Cicioğlu, M., Çalhan, A.: 'SDN-based wireless body area network routing algorithm for healthcare architecture', *ETRI J.*, 2019, **41**, (4), pp. 452–464

[11] Patil, S.: 'Load balancing approach for finding best path in SDN'. 2018 Int. Conf. on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2018, pp. 612–616

[12] Wang, C., Zhang, G., Xu, H., *et al.*: 'An ACO-based link load-balancing algorithm in SDN'. 2016 7th Int. Conf. on Cloud Computing and Big Data (CCBD), Macau, China, 2016, pp. 214–218

[13] Xue, H., Kim, K.T., Youn, H.Y.: 'Dynamic load balancing of software-defined networking based on genetic-ant colony optimization', *Sensors*, 2019, **19**, (2), p. 311

[14] Fortz, B., Thorup, M.: 'Internet traffic engineering by optimizing OSPF weights'. Proc. IEEE INFOCOM 2000. Conf. on Computer Communications. Nineteenth Annual Joint Conf. of the IEEE Computer and Communications Societies (Cat. No. 00CH37064), Tel Aviv, Israel, 2000, vol. 2, pp. 519–528

[15] Bharti, S., Pattanaik, K.K.: 'Dynamic distributed flow scheduling for effective link utilization in data center networks', *J. High Speed Netw.*, 2014, **20**, (1), pp. 1–10

[16] Huin, N., Rifai, M., Giroire, F., *et al.*: 'Bringing energy aware routing closer to reality with SDN hybrid networks', *IEEE Trans. Green Commun. Netw.*, 2018, **2**, (4), pp. 1128–1139

[17] Huin, N.: 'Energy efficient software defined networks (Réseaux pilotés par logiciels efficaces en énergie)'. University of Côte d'Azur, France, 2017

[18] Karaboga, D.: 'An idea based on honey bee swarm for numerical optimization'

[19] Panahi, V., Navimipour, N.J.: 'Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators', *Concurrency Comput., Practice Exp.*, 2019, **31**, (17), p. e5218

[20] Deb, K., Pratap, A., Agarwal, S., *et al.*: 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Trans. Evol. Comput.*, 2002, **6**, (2), pp. 182–197

[21] Koza, J.R.: 'Genetic programming', 1997

[22] Francois, F., Gelenbe, E.: 'Towards a cognitive routing engine for software defined networks'. 2016 IEEE Int. Conf. on Communications (ICC), Kuala Lumpur, Malaysia, 2016, pp. 1–6

[23] Mininet an Instant Virtual Network on your Laptop (or other PC). Available at http://mininet.org/

[24] R. team, RYU SDN Framework – English Edition. RYU project team, 2014

[25] 2014. OpenFlow Switch Specification 1.3.0. Available at https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[26] IEEE 802.3 ETHERNET WORKING GROUP. Available at http://www.ieee802.org/3/

[27] Jo, E., Pan, D., Liu, J., *et al.*: 'A simulation and emulation study of SDN-based multipath routing for fat-tree data center networks'. Proc. of the Winter Simulation Conf. 2014, Savanah, GA, USA, 2014, pp. 3072–3083

[28] Cisco 1941 Series Integrated Services Routers Data Sheet. Available at https://www.cisco.com/c/en/us/products/collateral/routers/1900-series-integrated-services-routers-isr/data_sheet_c78_556319.html

[29] Gueant, V.: 'Iperf-the TCP, UDP and SCTP network bandwidth measurement tool', Iperf. fr. Np, 2017

[30] Dobrijevic, O., Santl, M., Matijasevic, M.: 'Ant colony optimization for QoE-centric flow routing in software-defined networks'. 2015 11th Int. Conf. on Network and Service Management (CNSM), Barcelona, Spain, 2015, pp. 274–278