Received: 17 November 2017 | Revised: 1 October 2018 | Accepted: 4 November 2018

DOI: 10.1002/dac.3875

RESEARCH ARTICLE

WILEY

Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network

Ali Akbar Neghabi[1] | Nima Jafari Navimipour[2] | Mehdi Hosseinzadeh[1] | Ali Rezaee[1]

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

[2] Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran

Correspondence
Mehdi Hosseinzadeh, Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
Email: hosseinzadeh@srbiau.ac.ir

Summary

The growth of the networks has difficult network management. Recently, a concept called software-defined network (SDN) has been proposed to address this issue, which makes network management more adaptable. Control and forwarding planes are separated in SDN. The control plane is a centralized logical controller that controls the network. The forwarding plane that consists of transfer devices is responsible for transmitting packets. Because the network resources are limited, optimizing the use of resources in the networks is an important issue. Load balancing improves the balanced distribution of loads across multiple resources in order to maximize the reliability and network resources efficiency. SDN controllers can create an optimal load balancing compared to traditional networks because they have a network global view. The load-balancing problem can be solved using many different nature-inspired meta-heuristic techniques because it has the NP-complete nature. Hence, for solving load balancing problem in SDN, nature-inspired meta-heuristic techniques are important methods. However, to the best of our knowledge, there is not a survey or systematic review on studying these matters. Accordingly, in the area of the load balancing in the SDN, this paper reviews systematically the nature-inspired meta-heuristic techniques. Also, this study demonstrates advantages and disadvantages regarded of the chosen nature-inspired meta-heuristic techniques and considers their algorithms metrics. Moreover, to apply better load balancing techniques in the future, the important challenges of these techniques have been investigated.

KEYWORDS

load balancing, meta-heuristic, nature-inspired, review, SDN, software-defined networks

1 | INTRODUCTION

At first, the concept of software-defined network) SDN (is proposed by Stanford University.[1] The primary purpose of SDN is to make network management more flexible and easier by separating network infrastructure into two logical layers: control and data plane.[2,3] In this manner, the network controller is moved from the transfer devices to a logically centralized controller with the goal that software can implement the network functions. It also provides a comprehensive overview of the network resources that, unlike traditional networks, supports making changes globally in a centralized manner.[4-7] With the use of some open standards such as OpenFlow, this new network technique is implemented.

OpenFlow is one of the fundamental protocols that are equipped for managing, configuring, and interoperating between different network devices.[8,9] Traditional SDN architecture comprises three principal layers.[10] The bottom layer consists of network transmission devices, such as SDN switches, routers, and so on, which represents the data plane and provides basic packet forwarding functionality.[11,12] The middle layer consists of a centralized SDN controller that provides the functionality of a Network Operating System (NOS).[13] The NOS deals with and hides the distributed nature of the physical network and provides the abstraction of a network graph to higher layer (application layer) services.[14] The SDN controller configures SDN switches by installing forwarding rules via the so-called southbound interface.[11] In an SDN architecture, control layer (middle layer) and infrastructure layer (bottom layer) are communicated by OpenFlow protocol. OpenFlow enables direct availability to the forwarding plane of network devices such as virtual and physical switches and routers.[10,15] The top layer of the SDN architecture is application layer. Network services and applications, such as traffic engineering, routing, load balancing, etc., are developed in application layer.[11,12]

However, with the rapid growth of the SDN's resources, load balancing directly affects the availability of applications and system services.[16] One of the goals of load balancing is to enhance resource utilization, and also load balancing causes the throughput of the system and the user experience will be improved and the response time of the network will be shortened by avoiding congestion.[16] In SDN associated research, load balancing issue is one of the most important issues because of industry concerns.[17] Many data centers spread network traffics by load balancer devices.[17] Nevertheless, the excessive use of these devices can often be too costly.[18]

In spite of the fact that nature-inspired meta-heuristic load balancing techniques are significant in the SDN, to the best of knowledge of the authors, there is not any complete and comprehensive systematic review of this issue. There are several methods that have used nature-inspired meta-heuristic algorithms for solving the load balancing problem in the SDN. The purpose of this article is to review the current methods and compares the features of them. Another purpose is to provide an outline of the types of important load balancing issues that could address in the SDN. Briefly, this article can contribute in the following ways:

- Providing SDN architecture review and defining load balancing problem in the SDN.
- Offering a systematic review of the current load balancing techniques based on different nature-inspired meta-heuristic algorithms in the SDN and classifying them.
- Investigating the reviewed techniques and determining their pros and cons.
- Identifying the key areas where works in the future can improve the function of load balancing techniques in the SDN.

The rest of the article is organized as follows. In Section 2, SDN backgrounds and load balancing problem are described. In Section 3, some significant related work is focused. The research methodology and mechanism for selecting articles are presented in Section 4. Section 5 argues nature-inspired meta-heuristic algorithms for load balancing in the SDN and categorizing them, and also the selected mechanisms are compared. In Section 6, results and comparison are presented. Many open issues are discussed in Section 7. Eventually, this article conclusion and limitation are discussed in Section 8.

2 | BACKGROUND

The SDN structure has been perused, and also the essential profits of using SDN have been described in this section. Load balancing methods have been divided into two major categories which include static and dynamic methods. In the end, the metrics that affect load balancing efficiency are been explained.

2.1 | SDN architecture and OpenFlow protocol

In a traditional network, flows are particularly processed according to either one or some combinations of packet header attributes, including destination Media Access Control addresses, an integration of Internet Protocol (IP) addresses and Transmission Control Protocol/User Datagram Protocol port numbers, the longest destination IP prefixes, and so on.[19,20] The SDN handles the flows according to several characteristics of packet headers using a communication protocol such as the OpenFlow protocol.[20-24] As illustrated in Figure 1, Open Networking Foundation divides SDN architecture into three major planes:

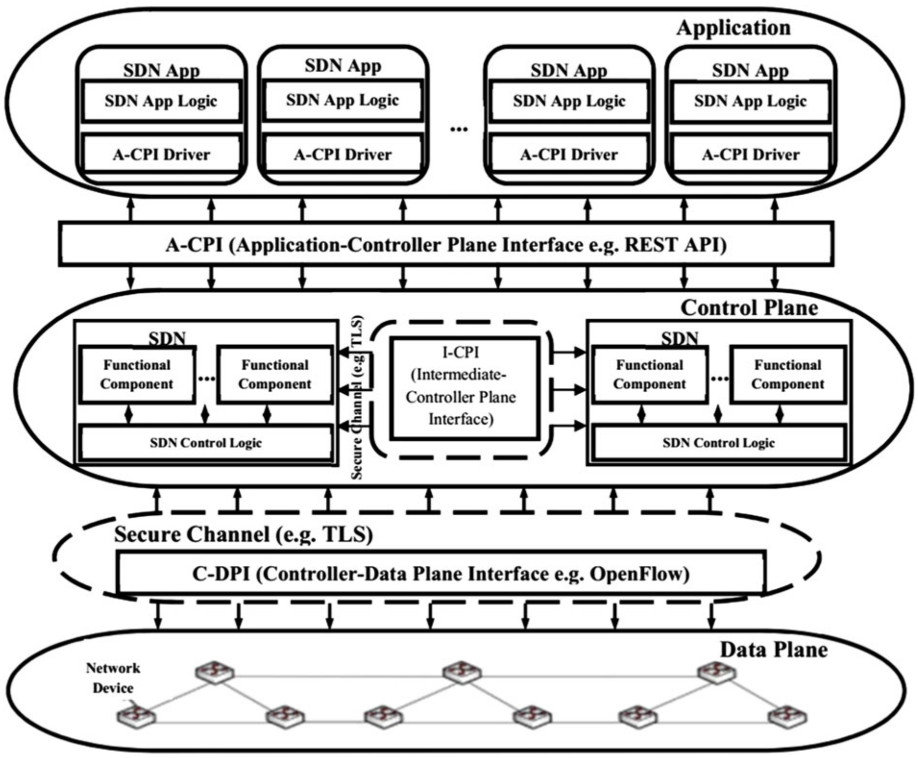
Application
SDN App
SDN App Logic
A-CPI Driver
...
A-CPI (Application-Controller Plane Interface e.g. REST API)
Control Plane
SDN
Functional Component
SDN Control Logic
Secure Channel (e.g TLS)
I-CPI (Intermediate-Controller Plane Interface)
Secure Channel (e.g. TLS)
C-DPI (Controller-Data Plane Interface e.g. OpenFlow)
Data Plane
Network Device

FIGURE 1 An overview of SDN architecture with its main planes: Data, control, and application plane[19]

- Data plane: In the SDN architecture, data plane is the lowest level and includes network devices like physical/virtual switches, routers, access point, etc..[19,20] These network devices are available and manageable via a controller data plane interface from SDN controllers. The network devices and controllers can communicate via secure connections, for example, the Transport Layer Security connection. OpenFlow protocol is a popular standard controller data plane interface for communication between the controller and data devices.[19,20]
- Control plane: A set of software-based SDN controller exists in the SDN control plane to provide control capabilities to monitor network transfer behavior via the controller data plane interface.[19] It has interfaces to allow communication between controllers and network devices, as well as between controllers and applications (Application to Controller Plane Interface (A-CPI)). An A-CPI can provide communication facility between network applications/services and a controller for management, network security, and etc. Functional components and control logic are two primary components of a controller. Controllers include at least one functional component such as virtualizer, coordinator, and so on; thus, they can manage the behaviors of controllers.[19,20,25] The load balancer is an application that its goal is to balance the network load. The SDN controller can run various applications include programs for network monitoring, intrusion detection, network virtualization, and load balancing. Hence, SDN controller has a component that called load balancer.
- Application plane: An SDN application plane has one or more end-user applications that using with controllers capture an abstract view of the network so that they can demonstrate their internal decision-making process. These applications communicate with controllers through an open A-CPI. In fact, the SDN application contains both an SDN application logic and an A-CPI driver.[19]

2.2 | SDN load balancing

Mostly, in distributed systems, load balancing technology is utilized to ameliorate overall cluster performance.[26-28] Load balancing is distributing the amount of work between two or more computing resources. Static and dynamic algorithms are two major categories of load balancing mechanisms. In static algorithms, it is essential to have prior information of the system, such as communication time, job resource requirements, system nodes processing power, storage media, and memory capacity, etc. Static load balancing algorithms do not depend on the current system state, and they also use the voting approach (the round robin approach). The essential advantage of static algorithms is its simple implementation, but the feasibility of inappropriate balancing is high.[29,30] Therefore, the major disadvantage of static load balancing algorithms is that the current system status is not taken into account in decision making, so it is not a proper approach in systems that load status cannot be foretold in advance. Dynamic algorithms actually rely on the present status of the system and are mainly employed to manage the changeable processing loads.[29,30] Tasks can be transferred from an overloaded node to an under-loaded one using dynamic algorithms. Continuously changing of load balancing as the main advantages of dynamic load balancing algorithms depend on the current system status. Dynamic mechanisms enable us to achieve better performance and obtain more efficient and accurate solutions.[30,31]

Load balancing is also used to equally allocate the workload throughout all the nodes.[30] It accomplishes by better resource utilization along with guaranteeing an effective and reasonable resource allocation.[30,32] An appropriate load balancing aids in avoiding overload of any single resource, maximizing throughput, minimizing resource consumption, minimizing response time, maximizing scalability, and so on.[33]

2.3 | Parameters of load balancing

Researchers have used some parameters to evaluate a load balancing method. These parameters are used to compare load balancing methods. Papers apply different parameters such as execution time, response time, and utilization. The most important qualitative metrics for load balancing in the SDN are defined as follows:

Delay: The time needed to forward a packet across a network is named delay. Many kinds of delay exist such as communication delay[34] and traffic delivery latency.[35]

Energy consumption: In the network, the amount of consumed energy is being determined by this metric. It should be minimized. Also, energy consumption can be reduced by efficient load balancing technique.[36,37]

Execution time: For a given task, execution time is determined as the time spent by the system executing that task. In a load balancing program, it can include re-association time,[38] migration time,[39] computation time,[40] and flow setup time.[41] In Wi-Fi networks, the re-association process happens when a wireless client temporarily moves outside the reach of an access point to another access point. Re-association time is the time taken to perform the re-association process. The time it takes for a number of switches from one controller to migrate to other controllers is called the migration time. A switch asks the controller to install suitable forwarding rules when a switch receives a new flow. The time necessary for this process is called the flow setup time.

Load balancing degree: This metric determines the monotony of the load distribution among entities. It should be maximized in load balancing. This metric is be measured by multiple indexes such as standard deviation of the load,[42] the arithmetic average for the coefficient of variation,[20] and Jain's fairness index and[38,43] entropy-based index.[44]

Overhead: It is any composition of excessive or indirect bandwidth, computation time, memory, or other resources that are required to perform a specific task. It should be minimized in load balancing. Communication,[45] synchronization,[46] and messages exchange and data stored overhead[47] are different types of overhead.

Response time: The interval that begins with accepting a job or request to responding to a task or request for the server is called response time.[48,49] It should be minimized in load balancing.

The rate of packet loss: When one or more packets do not reach to their target, packet loss happens. Network congestion usually caused packet loss. The rate of packet loss is the percentage of packets lost regarding packets sent.[6,50]

Throughput: During a given period of time, the amount of data that has been correctly transmitted from one place to another is called throughput.[51-53]

Utilization: A degree to which network resources are used such as bandwidth, link, memory, and utilization of CPU is called utilization. Load balancing can help to provide maximum resource utilization.[36,49,54,55]

3 | RELATED WORK

In this section, we describe the number of existing survey articles in SDN and determine some defects of them.

Karakus and Durresi[19] have reviewed scalability issues of the control plane in the SDN architectures. In the SDN architecture, the main reasons why the control plane suffers from scalability problems have been pointed out and the scalability conception has also been discussed. To measure the scalability of the systems, they have suggested some parameters for scalability. Their studies are organized according to the control plane scalability in the SDN, and the control plane scalability discussion is categorized into two extensive approaches consisting topology-related and mechanisms-related approaches. In addition, the relationship between the topology of architecture and scalability issues has been discussed in topology-related approaches. Also, the relationship between various mechanisms which are employed to optimize the performance of the controller and scalability issues have been reviewed in mechanisms-related approaches. However, mechanism of papers selection is unclear, and load balancing has not been investigated.

In many aspects, the SDN networks have shown great advantages compared with the conventional networks. To ameliorate the performance of the SDN networks, many load balancing methods have been suggested in the SDN. Li and Xu[56] have classified load balancing methods and analyzed the disadvantages and advantages of them. They have divided load balancing techniques into two main categories including load balancing in the centralized and load balancing in the distributed SDN architecture where centralized SDN architecture has a single controller whereas distributed SDN architecture has multiple controllers. However, there is a slit for discussion in mechanism of article selection. In addition, future works have not been satisfactorily explained.

Also, Benzekki, El Fergougui[57] have carried out one of the important studies of the SDN. An overview of SDN technology and the currently deployed network architecture have been explained by them. This survey presented different existing solutions and mitigation techniques that deal with SDN scalability, elasticity, dependability, reliability, high availability, resiliency, security, and performance issues. However, there is a slit for discussing the mechanism of article selection and open issues. Also, they have not discussed most articles published in 2016, and load balancing has not been studied.

To avoid overloading a single server, server load balancing is applied to balance requests between servers in large data centers. There are many techniques that perform load balancing. These techniques can be classified into two main categories including static and dynamic approaches. Static approaches do not provide the same balance of data between servers. Therefore, to improve the load balance of the server, dynamic methods are needed. With comparison among dynamic and static load balancing approaches in the SDN, Raghul, Subashri[58] have offered different experimental results and have viewed that dynamic methods present better performance in throughput and response time metrics. But, there is a slit in the mechanism for selecting articles and open issues.

Furthermore, Hu, Hao[22] have carried out an extensive survey of the essential topics in SDN/OpenFlow implementation, such as basic concept, language abstraction, applications, controller, security, QoS, and its integration with wireless networks. It not only describes the differences among some typical SDN applications but also shows OpenFlow applications for its flexibility. They have compared the differences in SDN security schemes based on OpenFlow/NOX; besides, they have introduced the new architecture parameters and also have compared network virtualization systems with flexibility, isolation, and management parameters. However, there is a slit in the mechanism for selecting articles and lately published papers. Moreover, load balancing problem has not been discussed.

To distribute the request between resources and increase the overall performance of the system, different load balancing strategies can be used. Each load balancing method has some advantages and disadvantages. Kumari and Thakur[59] have surveyed different load balancing strategies used in the SDN and investigate qualitative parameters that each article has used. However, they have written this survey in a non-systematically way. Also, open issues have not been described.

Finally, Neghabi, Navimipour[60] systematically have reviewed the load balancing techniques in the SDN. The load balancing mechanisms have been categorized into two classes including non-deterministic and deterministic approaches. Also, weaknesses and benefits of the selected load balancing algorithms have been represented by them, and they have investigated algorithm metrics. Furthermore, the key challenges of these algorithms have been investigated. But, they have not investigated nature-inspired load balancing techniques in SDN.

It should be noted that a pure systematic literature-based review of the current nature-inspired load balancing techniques in SDN and discussing the important future challenges have not been provided by these surveys. Also, a comprehensive and systematic review of the discussion on their categorization is not presented to date, despite the impact of load balancing techniques in the field of nature-inspired meta-heuristic techniques. By answering to any of these questions, this article in the next section formalizes three questions to select significant studies for evaluation and then describes the importance of nature-inspired load balancing techniques, key challenges, and future directions in SDN.

4 | RESEARCH METHODOLOGY

This section supplies a Systematic Literature Review (SLR) methodology which is presented by Kitchenham[61] paying attention to studies relevant to nature-inspired load balancing techniques in the SDN to increase apprehending of them. The SLR is used to perform a systematic study and a wide range of the nature-inspired load balancing algorithms in the SDN. Three research questions have been proposed by researchers to deal with the key issues of load balancing in the SDN to emboss the incumbency of nature-inspired load balancing in the SDN. We will formalize these questions in the next subsection.

4.1 | Question formalization

This study attempts to respond the following three research questions:

RQ1:. What is the importance of load balancing in the SDN?

The number of SDN nature-inspired load balancing studies which have been published all the time is determined by this question to confirm the emphasis of it in SDN.

RQ2:. How much do the current methods meet the main metrics of load balancing?

This question targets at evaluating the existing load balancing approaches derived from the primary metrics in SDN.

RQ3:. What obstacles and solutions can be identified in relation to the load balancing in the future years?

This question aims to investigate the load balancing role in the SDN and recognize the challenges and the methods used to insure the QoS.

4.2 | Process of paper selection

Automated keyword-based search; selection of the article according to the title, abstract, and quality of the publication; and formulation of selection criteria are three steps of the paper selection process.[62]

Stage 1:. Automated search based on keywords

Electronic searching on some popular academic databases is used in the search process. Due to that, electronic databases have been identified to detect an article for which the following online databases were used: Google Scholar,[63] Science Direct,[64] IEEE explorer,[65] Sage,[66] ACM,[67] Springer,[68] Wiley,[69] Emerald,[70] and Inderscience.[71] Then, a search string has been specified by selecting the most appropriate keywords in terms of supplying our subject. By adding other spellings of the essential elements, the following search string was defined to detect pertinent papers.

- ("Software Defined Network" OR "SDN") AND ("Load" OR "Balancing") AND ("Nature inspired" OR "Meta-heuristic")

We automatically for search articles based on the keywords across four preselected databases in February 2018 and then found 178 articles from the journals, conference proceedings, thesis, books, and patent. Between 2013 and 2018, these articles were published.

Stage 2:. Article selection based on the quality of the publisher

This step starts with the selection of certain practical selection criteria to guarantee that only high-quality publications and articles are selected for the review.[72] By searching for conference papers and journal articles, the search string is constrained. Hence, thesis, books, and patents are removed. Furthermore, the papers which are not written in English are eliminated to provide more accurate results. As a result, 137 papers are chosen. An overview of the used process to identify the articles in this study has been shown in Figure 2.

Stage 3:. Selection criteria

In addition to emerged articles from the initial search are refined, a Quality Assessment Checklist based on Kitchenham, Brereton[73] has been developed in this section. The checklist contains the following questions[73]: (1) Is the research methodology clearly put forward in this article? (2) Does the research methodology fulfill the requirements of the problem under study? (3) Is this study analyzed as it should be? If this study fulfills the evaluation criteria, then we can expect lots of "yes" replies. As described in Table 1, to pick up the right article, these criteria are introduced.

We eliminated the inappropriate abstracts, after reading searching keywords and abstracts as well as concepts that reflect the contribution of the paper. Afterward, the entire section of the residuary articles was investigated, and those which were not related to our respective area were also deleted. After screening the aforementioned studies based on the inclusion/exclusion criteria and Quality Assessment Checklist, 23 studies were identified which are shown in Table 2 where 13% of the articles are related to Springer, 4% are related to ACM, 9% are related to Science Direct, 61% are related to IEEE, 13% are related to other publications, and 0% are related to Emerald, Sage, Inderscience, and Wiley.

4.3 | Results

Selected articles are published between 2013 and 2017. Table 3 illustrates the distribution of nature-inspired meta-heuristic load balancing articles based on the year of publication. Most articles were published in 2016. Articles techniques can be classified into five categories including ant colony optimization (ACO), genetic algorithm (GA),

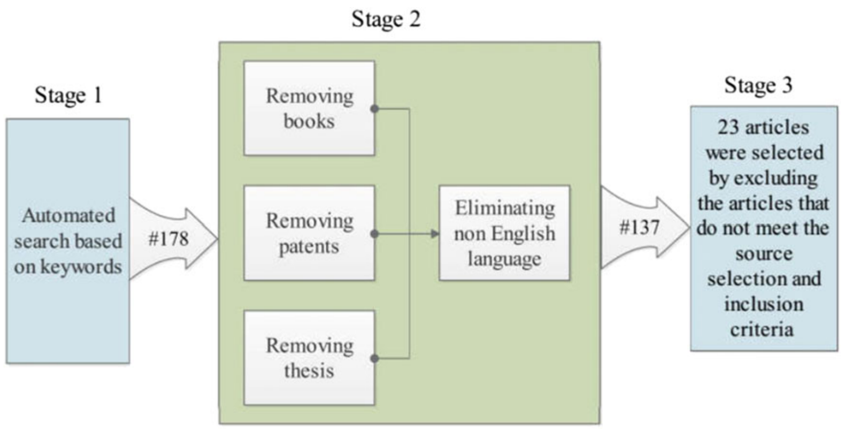
Stage 1
Automated search based on keywords
#178
Stage 2
Removing books
Removing patents
Removing thesis
Eliminating non English language
#137
Stage 3
23 articles were selected by excluding the articles that do not meet the source selection and inclusion criteria

FIGURE 2 Filtering method for found articles

TABLE 1 Summary of the inclusion-exclusion criteria for articles selection

Criterion	Rational
Inclusion1. A study that clearly mentioned the problem of load balancing in the SDN.	We need papers that straightly proposed the load balancing technique in the SDN.
Inclusion2. A study that is developed by either academics or practitioners.	Both academic and engineering solutions are relevant to this study.
Inclusion3. A study that is published in the SDN field.	SDN is our reference field.
Exclusion1. A study that does not focus on the load balancing techniques in the field of SDN environments.	The focus of this paper is only on studying the presented load balancing techniques in the SDN.
Exclusion2. A study that does not use a nature-inspired meta-heuristic approach.	We want to investigate nature-inspired meta-heuristic approaches.
Exclusion3. A study that does present a good way for evaluation and analysis.	Papers that do not provide the appropriate evaluation and analysis of the study are removed.

TABLE 2 Percentage of nature-inspired meta-heuristic load balancing articles in any publication

Publication	Percentage
ACM	4%
IEEE	61%
Science Direct	9%
Springer	13%
Other publications	13%

TABLE 3 Distribution of nature-inspired meta-heuristic load balancing in the SDN articles by year

Years	Numbers
2013	1
2014	3
2015	3
2016	9
2017	7

particle swarm optimization (PSO), greedy, and simulated annealing (SA). Table 4 illustrates the distribution of the articles based on the type of nature-inspired meta-heuristic algorithms. Table 5 demonstrates details of nature-inspired meta-heuristic algorithms for load balancing in the SDN. To perform a more accurate and detailed analysis, we chose four articles of each nature-inspired meta-heuristic algorithm based on full text and quality assessment. But, in some of nature-inspired meta-heuristic algorithms, the number of the published papers is low such as PSO and SA. Therefore, the number of articles investigated in these categories is less than 4. In the next sections, we surveyed the selected nature-inspired meta-heuristic algorithms in aforementioned groups.

5 | LOAD BALANCING METHODS

As previously mentioned, in SDN, nature-inspired meta-heuristic algorithms can be classified into five categories including ACO, GA, PSO, greedy, and SA. In the next subsections, we surveyed the selected nature-inspired meta-heuristic algorithms.

5.1 | Ant colony optimization technique

As a heuristic method in evolutionary computation, Dorigo[87] has proposed ACO.[88] This algorithm was inspired by a model of the natural behavior of real ants that is finding the shortest path from the nest to food sources.[89] In this process, pheromone plays a key role, which is used by the ants as an orientation and excreted by themselves during motion.[90] A single ant moves almost randomly. However, if it encounters pheromone tracks, then it follows them with a probability that increases with the strength of the track. She moves on the track and excretes pheromone on it.

TABLE 4 Distribution of the articles based on the type of nature-inspired meta-heuristic algorithms

Nature-Inspired Algorithm	Numbers
Ant colony optimization	5
Genetic algorithm	6
Particle swarm optimization	3
Greedy	7
Simulated annealing	2

TABLE 5 Details of nature-inspired meta-heuristic algorithms for load balancing in the SDN

Nature-Inspired Algorithms	Publisher	Year	Paper	Author	Conference/Journal	Selected?
Ant colony optimization	IEEE	2014	Load balancing using software-defined networking in cloud environment	Koushika and Selvi[42]	International Conference on Recent Trends in Information Technology	Yes
		2016	Joint route-server load balancing in software-defined networks using ant colony optimization	Sathyanarayana and Moh[74]	High Performance Computing and Simulation (HPCS)	Yes
			An ACO-based link load-balancing algorithm in SDN	Wang, Zhang[75]	7th International Conference on Cloud Computing and Big Data	Yes
		2017	Network aware VM load balancing in cloud data centers using SDN	Tsygankov and Chen[76]	Local and Metropolitan Area Networks (LANMAN)	No
	Other publications	2016	The load balancing research of SDN based on ant colony algorithm with job classification	Lin and Zhang[16]	2nd Workshop on Advanced Research and Technology in Industry Applications	Yes
Genetic algorithm	IEEE	2016	A fast and load-aware controller failover mechanism for software-defined networks	Fang, Wang[77]	10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)[77]	Yes
		2017	Entropy-based load-balancing for software-defined elastic optical networks	Mahlab, Omiyi[44]	Transparent Optical Networks (ICTON), 19th International Conference	Yes
	Springer	2014	A genetic-based load balancing algorithm in OpenFlow network	Chou, Yang[20]	Advanced Technologies, Embedded and Multimedia for Human-centric Computing	Yes
		2016	The multi-objective routing optimization algorithm for hybrid SDN	Gu, Luo[78]	Conference of Spacecraft TT&C Technology in China	No
	ACM	2017	A highly reliable and load balance supporting domain division algorithm for software-defined networks	Li and Shi[79]	Proceedings of the International Conference on High Performance Compilation, Computing and Communications	No
	Other publications	2016	Load balancing of software-defined network controller using genetic algorithm	Kang and Kwon[80]	Contemporary Engineering Sciences	Yes
Greedy	IEEE	2014		Han, Seo[81]		No

(Continues)

TABLE 5 (Continued)

Nature-Inspired Algorithms	Publisher	Year	Paper	Author	Conference/Journal	Selected?
			Software-defined networking-based traffic engineering for data center networks		Network Operations and Management Symposium (APNOMS)	
		2015	Towards adaptive elastic distributed software-defined networking	Chen, Li[40]	Computing and Communications Conference (IPCCC)	Yes
			Design of a load-balancing middlebox based on SDN for data centers	Tu, Wang[82]	Computer Communications Workshops (INFOCOM WKSHPS)	Yes
		2017	Client-network collaborative load balancing mechanism for WLAN based on SDN and 802.11 u	Sang, Wu[47]	Wireless Communications and Mobile Computing Conference (IWCMC)	Yes
			A switch migration-based decision-making scheme for balancing load in SDN	Wang, Hu[39]	IEEE Access	No
	Elsevier	2016	BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch	Boero, Cello[83]	Computer Networks	No
	Other publications	2017	An initial load-based green software-defined network	Hu, Luo[36]	Applied Sciences	Yes
Particle swarm optimization	IEEE	2016	A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles	He, Ren[34]	China Communications	Yes
		2017	An intelligent load balancer for software-defined networking (SDN) based cloud infrastructure	Govindarajan and Kumar[84]	Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)	Yes
	Springer	2015	Energy saving and load balancing for SDN based on multi-objective particle swarm optimization	Zhu, Wang[85]	International Conference on Algorithms and Architectures for Parallel Processing	Yes
Simulated annealing	IEEE	2013	Dynamic controller provisioning in software-defined networks	Bari, Roy[41]	Network and Service Management (CNSM)	Yes
	Elsevier	2016	Efficient routing for middlebox policy enforcement in software-defined networking	Li, Wu[86]	Computer Networks	Yes

Therefore, the track has more pheromone, and it is now strengthened to attract more ants. For solving discrete problems, the ACO is originally proposed, and it solved a lot of discrete problems effectively.[91,92] The remainder of this section is organized as follows. Section 5.1.1 consists of the overview of the selected mechanisms that were shown in Table 5, and summary of the reviewed articles together with their pros and cons propounds is presented in Section 5.1.2.

5.1.1 | Overview of the selected mechanisms

Wang, Zhang[75] have proposed a link load balancing algorithm based on ACO. The algorithm has used the search rule of the ACO and takes link load, delay, and pack-loss as impact factors that ants select next node. For maintaining link load-balancing and reducing end-to-end transmission delay, the widest and shortest path in all paths can be gained by ants. Simulation results have shown that the algorithm can balance the link load of network effectively, improve the QoS, and decrease network overhead. However, it may become trapped in some local optimal. Also, delay and throughput of the algorithm have not been considered. Furthermore, since this method applies the ACO algorithm, the rate of convergence may be slow down. Also, this method exploits single controller; hence, it suffers from low availability and low scalability. Moreover, the system has a bottleneck. A bottleneck occurs within a server when workloads arrive too quickly. It causes an interruption in processing packet-in messages and delays across the network.

Furthermore, Lin and Zhang[16] have offered a dynamic load balancing algorithm combined with job classification and ACO algorithm in SDN network cloud computing environments. They classify the server nodes that have the same processing capability of the CPU into a small clustered sub-network in SDN with the classifying SDN network server nodes. They have created a central controller whose work is to monitor the entire network, and there is a sub-controller in each sub-network. ACO algorithm runs in each controller. When a job comes, the central controller first finds the corresponding sub-network based on the job demand for CPU performance and sends the job to the sub-network controller; then, in the sub-network, the ACO algorithm can calculate the minimum load link road. In the end, the controller sends forwarding policy to the switch. Experimental results have proved that the optimized ACO algorithm can achieve better load balancing in each link which results in improving the utilization of resources and enhancing the usability of the system. However, it may become trapped in some local optimal. Moreover, overhead and delay of the algorithm have not been considered. Also, since this approach has used ACO algorithm, it needs a lot of memory. Furthermore, they use a central controller to monitor the entire network, and there is a sub-controller in each sub-network; hence, complexity of the method is high.

Also, network performance can be greatly ameliorated by managing resource utilization. Since there are no efficient ways to obtain network statistics from a device of the network, conventional load balancing approaches are usually inflexible and inefficient. The problem can be solved by adopting the SDN approaches. Sathyanarayana and Moh[74] have presented a dynamic load balancing method for SDN to balance both the paths leading to the servers load and the servers load. They have used the least loaded server load balancing policy to select the best server. Also, they have accepted the ant colony system algorithm to select the best path to reach the selected server. As an SDN controller load balancing module, they joined dynamic server load balancing algorithm and ant colony system routing. To detect the best server and path for network flows, this algorithm handles network statistics and server-load prepared by the controller. By supplying fewer network delay and upper network throughput, the evaluations show that the suggested algorithm fulfills considerably better than the shortest path round robin and shortest path random algorithms. Also, this algorithm has the least average response time and lowest packet loss compared with the other two algorithms. However, due to using a single controller, this approach suffers from system bottleneck, low availability, and low scalability. Moreover, analyzing the control overhead has not been performed. As well as, utilization and execution time of the proposed algorithm are not evaluated. Moreover, falling into a local optimum problem is not solved.

Moreover, when resources are added to data centers, most network resources are underutilized. When routing the data between the data centers, this underutilization causes congestion. Load balancing over the bottlenecked links can ameliorate this problem. Hence, Koushika and Selvi[42] have fulfilled the load balancer with SDN using the heuristic approach to help for solving the problem. The load balancer module is performed as the controller application. They have implemented the ACO-based load balancer. With the global view supplied by the controller, the suggested objective function calculates the best server and the optimized best path. When the request of a client comes at the switches, by balancing both the host and server load, the load balancer determines the best path and the best host. Results of the simulation have shown that proposed method achieves a higher degree of load balancing and also lower response time and delay than round-robin algorithm, which is performed in the SDN environment. However, evaluating the proposed method in a real testbed has not been done. Also, the packet loss rate and throughput of the method have not been investigated. Furthermore, this method uses single controller; hence, the system has a bottleneck. Moreover, scalability and availability are not achieved. Also, falling into a local optimum problem has not been solved.

5.1.2 | Summary of the reviewed ACO-based mechanisms

In the former subsection, based on SDN load balancing, four chosen ACO algorithms are reviewed, and their pros and cons are explained. Table 6 demonstrates a complete comparison of the most important pros and cons of each article. Some of the advantages of these methods are reduction of the (network) overhead, diminishing delay, and ameliorating load balancing degree. Moreover, since most of the methods use a single controller, they suffer from low availability, low scalability, and system bottleneck. Furthermore, these methods use ACO algorithm and hence may fall into local optima.

5.2 | Genetic algorithm technique

The GA inspired Darwin's theory of evolution that an organism's survivorship is depended on the fact that individuals whose fitness value is low will die while individuals whose fitness value is high will survive.[93] Darwin also said that the survivorship of a critter can be retained via the process of reproduction, mutation, and crossover. Then, a computational algorithm is adapted from the theory that a solution is found for an optimization problem. This algorithm has been

TABLE 6 Selected ACO-based load balancing techniques and their properties

Paper	Main idea	Advantages	Disadvantages	Missing Metrics
Wang, Zhang[75]	Link load balancing algorithm based on ant colony optimization	• Balancing the link load of network effectively • Improving the QoS • Reducing network overhead	• The possibility of falling into some local optimal • The rate of convergence may be slow down • Low scalability • Low availability • Bottleneck	• Delay and throughput of the algorithm have not been considered
Lin and Zhang[16]	Load balancing based on ant colony algorithm With job classification	• Achieving better load balancing in each link • Improving the utilization of resources • Enhancing the usability of the system	• It needs a lot of memory • The possibility of falling into some local optimal • High complexity	• Overhead and delay of the algorithm have not been evaluated
Sathyanarayana and Moh[74]	Load balancing the servers and the paths leading to the servers. Using ant colony system to select the best path to reach the chosen server	• Supplying fewer network delay • Improving network throughput • reducing average response time • reducing packet loss rate	• Low scalability • Low availability • Bottleneck • The possibility of falling into some local optimal	• Analyzing the control overhead has not performed • Utilization and execution time of the method is not considered
Koushika and Selvi[42]	Implementing the ACO and defining a new objective function That classifies the best path and best server	• Achieving a higher degree of load balancing • Reducing response time • Diminishing delay	• Have not evaluated in a real testbed • Low scalability • Low availability • Bottleneck • The possibility of falling into some local optimal	• Packet loss rate and throughput metrics have not been investigated.

proposed by Holland[94] in 1975 and called GA. This algorithm obtains a value of an optimal solution to a problem that has many feasible solutions. Genetic algorithms are started with the collection of casually generated solutions named population. In the population, each individual is named chromosome which is a delegate of each solution and appraised the level of robustness (fitness) by a predestinated function. Via the natural selection process on genetic operators, it is expected that a new chromosome with a higher level of fitness as a new generation (child) is generated by the genes of the two chromosomes. Chromosomes are recalculated, and output of this operation is called as generation. A fitness function evaluates the chromosomes.[95] We hopefully get the optimal solution while the GA converges to the best chromosome after several generations.[96] This algorithm is easy to combine with other algorithms, and also it has been designed for problems that have real or binary search space.[97]

5.2.1 | Overview of the selected mechanism

Chou, Yang[20] have used the GA to provide an OpenFlow-based load balancing system. Based on the GA, the data can be efficiently distributed between clients and various servers by this system. Additionally, with the preconfigured flow table entries, each flow can be directed in advance. When the traffic burst or server loading unanticipatedly increases, the proposed method can help to balance the workload of server farms. By comparison with other methods including load-based, random, and round-robin approaches, the evaluation results have shown that the suggested method has the high efficiency. However, the arithmetic average for the coefficient of variation metric has been only used by them. Moreover, because this method uses a single controller, system has a bottleneck, and also scalability and availability are not achieved. In addition, overhead and utilization of the algorithm have not been investigated. Furthermore, this method suffers from high computation time because it uses the GA.

Moreover, in the SDN when there are considerable requests for routing information from multiple switches to a controller, the controller overflows and the performance deteriorates. OpenFlow allows the use of multiple controllers in the network to solve this problem. All the controllers should be well balanced to avoid failure of a heavy-loaded controller. Kang and Kwon[80] have suggested a load balancing algorithm based on a GA. If controller load is not well balanced, their algorithm performs selection, crossover, and mutation to determine optimal load balancing. They have calculated controller load with CPU utilization and packet_in messages. They have simulated an SDN environment and have proved that an optimized mapping solution can be determined through the proposed algorithm. If the switches generate high complexity packet_in messages in this simulation, similar performance can be revealed in the real network. However, the evaluation of the algorithm is based on only load balancing factor, and the delay, overhead, and energy consumption of the algorithm have not been investigated. Moreover, the performance of the algorithm has not been compared with other approaches. Furthermore, due to using multiple controllers, the proposed method complexity is high. In addition, due to using the GA, the proposed method has slow convergence rate.

Also, one controller has been used to manage the entire network in the conventional SDN. It causes single point of failure (SPOF) problem. To solve this problem, Fang, Wang[77] have presented a fast and load-aware controller failover (FLCF) for SDN. To gather all failure notifications from other controllers about a failed controller, the FLCF applies a detecting controller so as to aid the detecting controller to construct the final decision. If the controller has failed, each controller precomputes its recovery scheme and synchronizes the scheme with other controllers that will adopt the switches. They have suggested a GA to prepare a near optimal recovery scheme for failure recovery. Based on simulation and experimental results, they have shown that the FLCF can reduce failover time compared to fast controller failover for multi-domain (FCF-M)[98] method. Also, after controller failure and switch reassignment, the FLCF can attain better controller load balancing. Also, the FLCF considers both switch-controller delay and controller loading. Therefore, the switches might not reassign only to some qualified controllers; hence, compared to non-stop network controller[99] and FCF-M methods, the controller load imbalance might not occur. However, the degree of load balancing of the proposed method is low compared to survivor method.[100] Furthermore, switch-controller delay of the method is higher than as FCF-M, non-stop network controller, and pre-partitioning failover[101] methods. Moreover, other metrics such as execution time, utilization, and energy consumption of the proposed method have not been evaluated. In addition, the suggested method has a weak local search because it uses the GA.

Furthermore, metro optical networks evolve over time with the surplus of new links to rescue network congestion and bottlenecks to support exuding traffic requests. For minimizing the cost of the novel deployment, new resources location should be optimally determined in the network. Network-wide load imbalance and spectrum-fragmentation prevent the optimal allocation of network resources. Lately, with the convergence of advanced programmable and flexible optical devices with emanating SDN paradigms, it is possible to balance the load on the network with dynamic and flexible defragmentation of the spectrum. For fiber-load balancing across the network, Mahlab, Omiyi[44] have introduced an optimization strategy in the software-defined elastic optical networks. Also, by using this strategy, the cost of the resulting service disruption is minimized. For gauging load imbalance, they have suggested an entropy-based metric and applied to design a joint entropy/hits utility function with a GA for the optimization. Evaluation results have illustrated that using the joint entropy/hits affects load balancing on the links in the network. Also, bandwidth is abandoned from more weighty loaded fibers and services redistributed between less applied fibers. However, the GA is very slow.[102] Hence, it is one of the main disadvantages of the proposed method. Moreover, this method used a single controller in the SDN, so it suffers from system bottleneck, low scalability, and low availability. In addition, a real testbed has not been used to evaluate the proposed method, and it is the possibility of falling into some local optimal. Moreover, throughput and packet loss rate metrics have not been investigated.

5.2.2 | Summary of the reviewed genetic-based mechanisms

Based on SDN load balancing, four chosen GAs are reviewed, and their benefits and drawbacks are explained in the former subsection. Table 7 illustrates a complete comparison of the most important pros and cons of each article. Some of the advantages of these methods improve the load balancing degree and high efficiency. Most of the methods use a single controller; hence, they suffer from system bottleneck, low scalability, and low availability. Furthermore, these

TABLE 7 Selected genetic-based load balancing techniques and their properties

Paper	Main idea	Advantages	Disadvantages	Missing Metrics
Chou, Yang[20]	Using the genetic algorithm to provide an OpenFlow-based load balancing system	• Balancing the workload of server farms • High efficiency in comparison to other methods	• Using one metric for evaluation • High computation time • Low scalability • Low availability • Bottleneck	• Utilization and overhead of the algorithm have not been evaluated
Kang and Kwon[80]	Performing selection, crossover, and mutation to specify optimal load balancing if controller load is not balanced	• Improving the load balancing degree	• Evaluation of the algorithm is based only load balancing factor • The performance of the algorithm was not compared with other approaches • Slow convergence rate High complexity	• The delay, overhead, and energy consumption of the algorithm have not been investigated
Fang, Wang[77]	Suggesting a genetic algorithm to prepare a near optimal recovery scheme for failure recovery cause to attain better controller load balancing	• Reducing failover time • Better controller load balancing • The controller load imbalance might not occur compared to investigated methods	• The degree of load balancing is low • Weak local search • Delay is high	• Utilization, energy consumption, and execution time metrics have not been evaluated
Mahlab, Omiyi[44]	Introducing an optimization strategy in the software-defined elastic optical networks and suggesting an entropy-based metric for gauging load imbalance	• Bandwidth is abandoned from weightier loaded fibers • A better degree of load balancing	• Without evaluation in a real testbed • Slow convergence rate • The possibility of falling into some local optimal • Low scalability • Low availability • Bottleneck	• Throughput and packet loss rate metrics have not been considered.

methods use GA and so may fall into local optima and convergence rate is slow. These are some of the disadvantages of the selected mechanisms.

5.3 | Methods that have used a greedy algorithm

To detect a global optimum, a greedy algorithm makes the locally optimal selection at each step; hence, it pursues the problem-solving heuristic.[103] To solve multi-step and complex problems, it uses a mathematical process that searches simple and easy-to-implement solutions.[104] In detecting a solution, a greedy algorithm is not prosperous because all the search spaces are not searched entirely.[105] Moreover, to solve the problems, traditional approximate optimization approaches such as greedy-based algorithms make multiple suppositions. Sometimes, the validation of these suppositions is hard in each problem.[97]

5.3.1 | Overview of the selected mechanism

Due to the rapid development of cloud computing technology, the number of users has increased dramatically as well as modern cloud-based applications are becoming more communication-intensive and need more bandwidth; hence, existing data centers are not improved for cloud-based software services.[106] Because of the changeability of traffic load, real data centers may toil from high latency, lower throughput, and low QoS.[107,108] To solve this problem, A programmable middlebox has been presented by Tu, Wang[82] that can evenly distribute traffic. Clos network is a multi-stage network which is proposed by Clos.[109] The middlebox that has been used in this method is based on a clos network. It designs and uses SDN to improve bandwidth utilization while ensuring QoS. To discover the optimal path for the traffic within the data center, the middlebox has been used. To exhibit the cost of transferring data from one server to another, it uses a matrix called price matrix. To calculate the price matrix, a greedy algorithm is exploited. The information is gathered from switches by the SDN controller of the middlebox. It achieves load balancing by using the traffic distribution and server loads information. When it is fulfilled in a data center, the middlebox can significantly minimize delay and enhance bandwidth utilization. The middlebox can simply be utilized in the available data centers because it does not rely on any specific the data center's feature. However, energy consumption of the approach and the overhead of the middlebox have not been evaluated. Also, they have not investigated more QoS constraints.

Also, at present, research on energy saving in SDN is mainly focused on the static optimization of the network with zero loads when new traffic arrives, changing the transmission path of the uncompleted traffic which arrived before the optimization, possibly resulting in route oscillation and other deleterious effects. To avoid this, Hu, Luo[36] have designed a dynamical energy saving optimization scheme in which the paths of the uncompleted flows will not be changed when new traffic arrives. To find the optimal solution for energy saving, the problem has modeled as a mixed integer linear programming problem. As the high complexity of the problem prohibits the optimal solution, an improved heuristic routing algorithm called improved constant weight greedy algorithm has been proposed to find a sub-optimal solution. Simulation results have shown that the energy saving capacity of improved constant weight greedy algorithm is close to that of the optimal solution, offering desirable improvement in the energy efficiency of the network. Also, in the different link densities, average bandwidth utilization is the same. However, the performance of the algorithm has been compared only with the Dijkstra algorithm. Also, execution time and delay of the approach have not been measured.

Furthermore, the congestion of both clients and access points (APs) have increased because the use of wireless local area network (WLAN) is growing. In the throng WLANs, load balancing problem becomes important. For solving load balancing AP selection problem, Sang, Wu[47] have suggested a client-network collaborative architecture based on 802.11u and SDN. The SDN is used to collect and store the load information in order to manage the network centrally. Also, to chaffer the metrics of AP selection among network based on 802.11u and client, they have designed Network Resource Query Protocol. Using this architecture, they have proposed an innovative load balancing algorithm that can consider client traffic, AP load, and link state. This algorithm is a greedy solution, and it can complete the calculation in polynomial time. Experimental results in a real SDN testbed have shown the proposed method that improves the throughput. However, they have used a single controller in the proposed architecture; hence, it has a SPOF. Also, response time and utilization of the method have not been investigated.

On the other hand, to deal with the problems of the single controller SDN networks, multiple controllers have been used in the network to decrease the delay between switches and controllers and ameliorate network reliability. However, the mapping between controllers and SDN switches is statically configured in the currently distributed SDN schemes, which can lead to a load imbalance between controllers. For solving this problem, Chen, Li[40] have suggested an adaptive elastic distributed SDN architecture. A minimum number of active controllers that switches attached to them are selected dynamically by architecture. Then, the mapping between controllers and switches according to the network load are changed. Particularly, a switch can migrate from one controller domain to another so that the mapping is compatible with the load of the network. The controller selection problem has been formalized by them as an optimization problem and using greedy algorithms. The controller selection problem has been formalized by them as an optimization problem, and they chose to use greedy algorithms. The results of the simulation have illustrated that the number of active controllers is reduced and that the entire load of the network decreases. But, due to using multiple controllers, this method has high complexity. Also, in the experimental evaluations, throughput and packet loss rate of the method have not been considered.

5.3.2 | Summary of the reviewed greedy-based mechanisms

Four chosen greedy algorithms for load balancing in the SND, as well as their disadvantages and advantages, were investigated in the prior subsection. These properties are shown in Table 8.

5.4 | Techniques that have used particle swarm optimization algorithm

An algorithm that simulates the flock of birds or insects motion to find the best solution is called PSO. It is suggested by Eberhart and Kennedy[110] in 1995. In the algorithm, the insects or birds are called particles. They are initialized by velocities and random positions.[111,112] Each particle is defined by a group of vectors including the position and velocity of the particle and the personal best position. Solving optimization problems starts with preparing a set of random potential solutions (or particles), and then each particle is given a velocity to move in the search space, which contains all possible solutions.[113] Each solution (or particle) adjusts its velocity using its best situation and the situation of the best solution (or particle) of the whole population in each repetition. To balance the utilization in this technique, local search methods integrate with global search methods. PSO is easy to fulfill. But, in the complex problems, it may encounter with the parameter selection problem, slow convergence rate, and easily falling into a local optimum.[97]

TABLE 8 Selected greedy-based load balancing techniques and their properties

Paper	Main Idea	Advantages	Disadvantages	Missing Metrics
Tu, Wang[82]	Programmable middlebox based on a clos network	• Improving bandwidth utilization • Reducing delay • It can be deployed in current data centers	• Have not been evaluated more QoS constraints	• Energy consumption has not been evaluated • Overhead of the middlebox has not been considered
Hu, Luo[36]	Improved constant weight greedy algorithm	• Improving the energy efficiency Average bandwidth utilization of different link densities is the same	• The performance of the algorithm has been compared only with the Dijkstra algorithm	The execution time of the algorithm has not been evaluated • Delay of the approach has not been investigated
Sang, Wu[47]	Client-network collaborative architecture based on 802.11u and SDN	• Have been evaluated in a real SDN testbed • Improving throughput	• Single point of failure	• Response time and utilization of the method have not been measured.
Chen, li[40]	Adaptive elastic distributed SDN architecture	• Reducing the number of active controllers • Decreasing entire load of the network	• High complexity	• Throughput and packet loss rate metrics have not been considered.

5.4.1 | Overview of the selected mechanism

Most of the energy saving approaches on existing IP network only accumulate traffic into a part of links. It results in imbalance link utilization and seriously affects the QoS. Zhu, Wang[85] have taken benefit of the centralized control and global view of SDN in order to obtain the network load balancing and energy saving by aggregating and balancing the traffic dynamically while guaranteeing QoS. To formulate a multi-objective mixed integer programming model, actual QoS constraints have been added to the basic maximum concurrent flow problem. A multi-objective PSO algorithm named MOPSO has also been proposed to solve this NP-hard problem. MOPSO distributes optimal paths for dynamic traffic demands and turns idol switches and links into sleeping mode. The experimental results on real topologies and traffic requirements have shown the efficiency of the method on the goal of energy saving and load balancing in comparison to other algorithms. However, since the proposed method utilizes centralized controller, it suffers from low availability and scalability. Also, overhead, delay, and throughput of the algorithm have not been considered.

On the other hand, in the Intelligent Transportation System, the Internet of Vehicles (IoV) is considered as a typical application of the Internet of Things (IoT). High processing latency, less mobility support, and location awareness still give rise to IoV suffering. He, Ren[34] have united the fog computing and cloud computing with SDN to deal with the aforementioned problems. Fog computing expands storing and computing to the edge of the network, which could significantly decrease latency and also enable location awareness and mobility support. Withal, centralized control and global view of the network have been provided by SDN. In order to handle the Software Defined Cloud/Fog Networking architecture in the IoV, they have suggested a new SDN-based adapted constrained optimization PSO algorithm. To eschew falling into local optimum, they have proposed an algorithm which utilizes linear inertia weight and the reverse of the mutation particles flight to improve the efficiency of constrained PSO algorithm. The evaluation results have illustrated that the QoS has been improved, and the delay has been decreased in the Software Defined Cloud/Fog Networking architecture by the suggested algorithm. However, utilization, degree of load balancing, and energy consumption metrics have not been taken into account. Also, other QoS parameters such as capacity and security have not been investigated.

Moreover, the cloud provides on-demand storage, computing, and networking resources for cloud consumers. What acts as an intermediary between cloud providers and cloud consumers is called a Cloud Resource Broker (CSB). Application requests effectively are managed by CSB, and it allocates cloud resources efficiently. When CSB meets large amounts of application requests, it must distribute these application requests among the available cloud resources. To distribute cloud consumer application requests in a balanced and optimal manner, Govindarajan and Kumar[84] have presented a PSO-based load balancing method. Also, they have investigated the SDN cloud infrastructure, which, based on the network load, dynamically configures and provides network paths on-demand. Evaluation of the method in a real testbed has shown that proposed load balancing minimizes average response time. Also, it maximizes throughput, resource utilization, and cloud consumer satisfaction value. But they use a single controller in the SDN cloud infrastructure that causes system bottleneck, low scalability, and low availability. Also, the proposed method execution time and energy consumption have not been measured.

5.4.2 | Summary of the reviewed PSO-based mechanisms

In the former sub-section, three chosen PSO algorithms for load balancing in the SDN were analyzed; also, their pros and cons were considered. Table 9 illustrates these properties. The analyzation of the selected algorithms shows that high utilization is an advantage of the most reviewed methods. But, most of them may fall into local optima; also, in the complex problems, their convergence rate is slow.

5.5 | Simulated annealing technique

It is a local search meta-heuristic presented by Kirkpatrick.[114] It works based on the way thermodynamic systems go from one energy level to another. In the search space of this method, each point x represents a state, and function $f(x)$ represents an internal energy of a physical system in that state. The goal of this method is a transferring system from casual initial state to the least energy state. Consequently, at each stage, some neighboring states x' of the present state x is considered, and the basis of the probabilities is determined whether system goes to state x' or remains in the state x.

TABLE 9 Selected PSO-based load balancing techniques and their properties

Paper	Main idea	Advantages	Disadvantages	Missing Metrics
Zhu, Wang[85]	Obtaining the network load alancing and energy saving by aggregating and balancing the traffic dynamically with using SDN properties	• Energy saving • Medium computation time • High utilization	• Low scalability • Low availability	• Overhead, delay, and throughput of the algorithm have not been considered
He, Ren[34]	Integrated the fog computing, cloud computing with SDN.	• Decreasing the delay • Improving the QoS • Eschewing to fall into local optimum	• Security and capacity have not evaluated	• Energy consumption has not been considered • Utilization metric has not been evaluated • Load balancing degree has not been evaluated
Govindarajan and Kumar[84]	Distributing cloud consumer application requests in a balanced and optimal manner with using PSO based load balancing method in SDN cloud infrastructure	• Have evaluated in a real SDN testbed • Minimizing average response time • Maximizing throughput • Maximizing utilization • Maximizing cloud consumer satisfaction value	• Low scalability Low availability • Bottleneck	• Execution time and energy consumption have not been measured

Eventually, until the system attains a good state for the application, these probabilities propel the system to states of lower energy.[97] SA is free to fall into local optimal, and its implementation is easy. But, its initial temperature and number of iteration affect on solution quality.[97]

5.5.1 | Overview of the selected mechanism

A conventional SDN implementation uses a single controller. This preliminary centralized method has multiple limitations relevant to scalability and performance in a large-scale network. To handle these issues, using multiple controllers that work collaboratively to manage a network has been suggested. Nevertheless, this approach causes a problem, which is called the Dynamic Controller Provisioning Problem. This problem is defined as determining the controller's number and their locations with modifying network situations, in order to minimize communication overhead and flow setup time. For deploying multiple controllers within a large-scale network, Bari, Roy[41] have presented a framework. This framework, according to network dynamics, dynamically regulates the number of active controllers and delegates each controller with a subset of OpenFlow switches while warranting minimal communication overhead and flow setup time. Also, as an Integer Linear Program, they have formulated the optimal controller provisioning problem and have suggested two heuristic approaches, based on SA and greedy algorithms, to solve it. The method based on SA uses two algorithms that the aim of algorithm 1 is to produce a possible switch to controller assignment from the current unfeasible assignment. The output of this algorithm is presented as an input to algorithm 2 which runs the SA algorithm to ameliorate the switch to-controller assignment. Their method solves Dynamic Controller Provisioning Problem by considering changing traffic load. Based on evaluation results, flow setup time is minimized while causing very low communication overhead. But, they have not considered convergence time of the method. Also, they have not tested the method in a real testbed. Moreover, response time, execution time, and utilization metrics have not been evaluated.

On the other hand, to raise network operations, for example, guaranteeing performance and providing security, applications of the network need traffic to go through multiple kinds of middleboxes. Sequenced-middlebox policy routing over regular layer 2/3 flow routing is challenging network administrators to manage it. Moreover, different kinds of middlebox resources concurrently procured by innumerable applications entangle network-resource management. Also, security challenges and the malfunction of the whole network can be created by middlebox failures. In the context of sequenced-middlebox policy routing, to get a network load-balancing objective, a mixed integer linear programming problem is formulated by Li, Wu.[86] By balancing the whole network, simplifying candidate path selections, and using the SA algorithm, network resources are managed by their method effectively. Based on evaluation results, their method increases throughput compared with shortest source-destination path method. Also, it has lower end-to-end delay and loss rate. However, they use a single controller in the SDN; hence low scalability, low availability, and system bottleneck are the disadvantages of the proposed method. Also, response time, energy consumption, and load balancing degree of the suggested method have not been measured.

5.5.2 | Summary of the reviewed SA-based mechanisms

Two selected SA algorithms for load balancing in the SND, as well as their pros and cons, were investigated in the prior subsection. These properties have been shown in Table 10. Both two method uses a single controller in the SDN; hence, these methods suffer from a SPOF. Furthermore, energy consumption of these methods has not been measured.

6 | RESULTS AND COMPARISON

In the prior sections, some load balancing approaches based on nature-inspired meta-heuristic algorithms in the SDN have been explained. Reviewing the articles showed that some of the metrics have been used to compare the reviewed papers together. Delay, energy consumption, execution time, load balancing degree, overhead, response time, the rate of packet loss, throughput, and utilization are metrics that have been described in subsection 2.3.

Based on the reviewed articles and acquired results in the previous section, most of the nature-inspired meta-heuristic load balancing algorithms have used delay, load balancing degree, and throughput metrics to evaluate its proposed method but none of the papers are used to the whole metrics and most of the papers do not consider rate of packet loss and energy consumption metrics. According to the reviewed articles, these methods can increase throughput, load balancing degree, and utilization metrics and reduce delay metric. Most of them may be falling into local optimal and suffer from slow convergence rate. Also, these methods often use single controller hence suffer from system bottleneck, low scalability, and low availability. An overview of the argued load balancing nature-inspired meta-heuristic methods and their essential properties are shown in Table 11.

Figure 3 illustrates researchers that were epitomized on qualitative metrics so as the delay is 21%, the degree of load balancing is 18%, throughput is 13%, utilization is 11%, response time is 11%, execution time is 8%, overhead is 8%, rate of packet loss is 5%, and energy consumption is 5%. This figure shows that the most crucial qualitative metrics that are utilized by researchers are delay, load balancing degree, and throughput metrics and rate of packet loss and energy consumption less used.

7 | OPEN ISSUE

This research illustrates that there are some essential issues that have not been comprehensively and extensively perused in the load balancing of SDN. Accordingly, in this section, some open research problems exist.

TABLE 10 Selected SA-based load balancing techniques and their properties

Paper	Main Idea	Advantages	Disadvantages	Missing Metrics
Bari, Roy[41]	Dynamically regulates the number of active controllers and delegates each controller with a subset of OpenFlow switches	• Minimizing flow setup time • Low communication overhead	• Convergence time is not specified Without evaluating a real testbed	• Response time, execution time, and utilization of the method have not been considered
Li, Wu[86]	Getting a network load balancing in the context of sequenced-middlebox policy routing by balancing the whole network, simplifying candidate path selections, and using the simulated annealing algorithm	• Increasing throughput • Lower delay • Lower loss rate	• Low scalability • Low availability • Bottleneck	• Energy consumption has not been considered • Response time metric has not been evaluated • Load balancing degree has not been evaluated

TABLE 11 Load balancing metrics in reviewed nature-inspired meta-heuristic algorithms

Algorithm	Reference	Delay	Energy Consumption	Execution time	Load Balancing Degree	Overhead	Response time	The Rate of Packet loss	Throughput	Utilization
ACO	Wang, Zhang[75]	+	-	-	+	+	-	+	-	-
	Lin and Zhang[16]	-	-	-	-	-	+	-	+	-
	Sathyanarayana and Moh[74]	+	-	-	-	-	+	+	+	-
	Koushika and Selvi[42]	+	-	-	+	-	+	-	-	-
GA	Chou, Yang[20]	-	-	-	+	-	-	-	-	-
	Kang and kwon[80]	-	-	-	+	-	-	-	-	-
	Fang, Wang[77]	+	-	-	+	-	-	-	-	-
	Mahlab, Omiyi[44]	-	-	-	+	-	-	-	-	-
Greedy	Tu, Wang[82]	+	-	-	-	-	-	-	-	+
	Hu, Luo[36]	+	+	-	-	-	-	-	-	+
	Sang, Wu[47]	-	-	-	-	+	-	-	+	-
	Chen, li[40]	-	-	+	+	-	-	-	-	-
PSO	Zhu, Wang[85]	-	+	+	-	-	-	-	-	+
	He, Ren[34]	+	-	-	-	-	-	-	-	-
	Govindarajan and Kumar[84]	-	-	-	-	-	+	-	+	+
SA	Bari, Roy[41]	-	-	+	-	+	-	-	-	-
	Li, Wu[86]	+	-	-	-	-	-	-	+	-

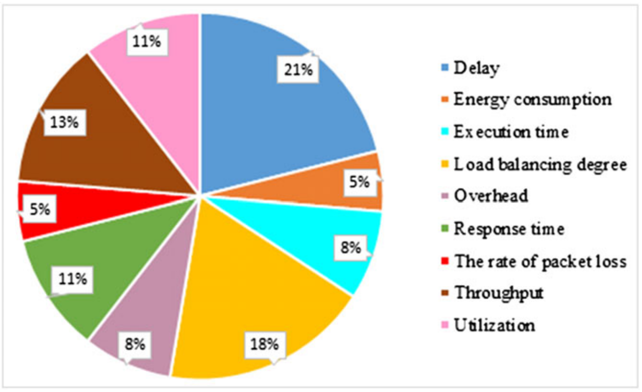

FIGURE 3 Percentage of load balancing metrics in reviewed nature-inspired meta-heuristic algorithms

In some of the reviewed approaches, researchers have not considered some factors such as packet priorities and traffic patterns. Therefore, for future research, one of the issues is applying these factors to the load balancing decisions.

Also, in most of the reviewed methods, energy consumption of the method has not been evaluated. Because the energy saving is very important in the large-scale network, to make load balancing and reduce energy consumption and carbon emission using with a new meta-heuristic nature-inspired method is a very attractive line for future studies.

Furthermore, none of the reviewed papers cooperate with two meta-heuristic algorithms for solving load balancing problem in the SDN. Since each meta-heuristic algorithm has defects, it is very interesting in the future to combine some of them and create a hybrid algorithm that can use the advantages of both algorithms and reduce their disadvantages.

Moreover, in other analogous networks such as mobile cloud computing,[115] peer-to-peer networks,[116] machine-to-machine networks,[117] and mobile ad hoc networks[118] can use an SDN-based solution to cater load balancing. In the future, it is another research direction.

On the other hand, for future works, applying novel optimization algorithms can be very intriguing. Optimization approaches such as bat optimization algorithm,[119] variable flight mosquito flying optimization algorithm,[120] sun and leaf optimization algorithm,[121] gray wolf optimization algorithm,[122] whale optimization algorithm,[123] lion optimization algorithm,[124] and etc can be used by researchers.

Also, most of the reviewed articles suffer from premature convergence and trapping into local optimal. Researchers can apply some of the meta-heuristic algorithms to avoid the occurrence of these two problems. Some of these algorithms are adaptive genetic algorithm,[125] genetic algorithm self-organizing map,[126] quantum-behaved PSO,[127] and elite mating pool.[128]

Furthermore, using SDN, we can facilitate network configuration and conserve resources. This SDN feature can be unified with the IoT to improve network performance. Generally, IoT devices are low powered devices and have less computing power. In data transmission across the network, energy saving plays an important role.[129] Hence, in the IoT, presenting a load balancing method that considers energy consumption criteria is interesting in the future.

Finally, in discussed articles, the researchers have mainly used simulator-based tools to build the SDN network as well as to simulate multiple controllers for evaluations. Therefore, it is very interesting in the future to implement the mentioned approaches in an actual large-scale network with more real-world traffic as well as appraise the performance.

8 | CONCLUSION AND LIMITATION

A systematic review of nature-inspired meta-heuristic algorithms for load balancing in the SDN has been provided in this paper. SDN structure and load balancing problem are investigated, and some of the metrics that researchers have used to evaluate their method are described. We defined research methodology and selected 17 articles among the basic 23 principal studies from our search query. The most articles were published in 2016, and the least papers were published in 2013 according to the SLR. The most published papers belong to IEEE with 61% of articles in journals and conferences. However, between selected publishers, ACM with 4% have the least published articles. Most of the papers have been published in conferences. We classified 17 selected articles in five principle categories including

ACO, GA, greedy, PSO, and SA. We also specified the main idea, pros, and cons of the load balancing methods in each category. Most of the reviewed techniques suffer from trapping into local optima and premature convergence. According to obtained results, ACO-based articles can improve load balancing degree, delay, overhead, throughput, utilization, and response time, whereas GA-based papers only improve load balancing degree. Greedy, PSO, and SA-based approaches have improved fewer criteria in comparison with ACO. We investigated load balancing metrics in reviewed articles and illustrated them in Table 11. According to obtained results, the most important qualitative metrics that are utilized by researchers are delay, degree of load balancing, and throughput metrics. Also, In the reviewed articles, rate of packet loss and energy consumption metrics are considered less. In this paper, the challenges that can be considered by researchers in future are discussed. Exclusively, the responses to the research questions summarized load balancing's main goal, existing challenges, mechanisms, and open issues in SDN. We aspire that the results of this research will aid researchers to propose more effective load balancing methods in SDN.

This survey is presented to steer a comprehensive and systematic review of nature-inspired load balancing techniques in SDN. But, there may be some constraints. Hence, some constraints must be considered in future studies as follows:

- Research scope: several sources such as patents, thesis, academic publications, technical reports, editorial notes, and web pages have covered the load balancing application in SDN. But, to attain the best qualification, we heeded only academic conferences and journals. Also, we have relinquished papers not written in the English language.
- Study and publication: based on prior review experiment, eight online databases are selected. Whereas, the statistics detect that the most reliable and relevant articles are presented by these eight online databases. However, selection of all related articles could not be guaranteed. There is an eventuality that some appropriate articles were overlooked throughout the processes argued in Section 4.
- Categorization: We classified articles in five categories including ACO, GA, greedy, PSO, and SA because these nature-inspired meta-heuristic algorithms have suggested to untangle load balancing problem in the SDN. If the novel nature-inspired meta-heuristic algorithm is suggested to untangle load balancing problem in the SDN similar whale optimization algorithm, their presented category can be modified and completed.
- Study queries: We defined three questions to develop this article; however, there is the possibility of adding other questions.

ORCID

Nima Jafari Navimipour https://orcid.org/0000-0002-5514-5536
Mehdi Hosseinzadeh https://orcid.org/0000-0003-1088-4551

REFERENCES

1. Ortiz S. Software-defined networking: on the verge of a breakthrough? *IEEE Comp*. 2013;46(7):10-12.
2. Abdelaziz A, Fong AT, Gani A, et al. Distributed controller clustering in software defined networks. *PloS Oone*. 2017;12(4):e0174715.
3. Rego A, Canovas A, Jiménez JM, Lloret J. An intelligent system for video surveillance in IoT environments. *IEEE Access*. 2018;6:31580-31598.
4. Dao N-N, Kim J, Park M, Cho S. Adaptive suspicious prevention for defending DoS attacks in SDN-based convergent networks. *PloS Oone*. 2016;11(8):e0160375.
5. Rego A, Garcia L, Sendra S. *Lloret J.* Future Generation Computer Systems: Software Defined Network-based control system for an efficient traffic management for emergency situations in smart cities; 2018.
6. Askar SK. Adaptive load balancing scheme for data center networks using software defined network. *Sci J Univ Zakho*. 2016;4(2):275-286.
7. Xu H, Li X-Y, Huang L, Deng H, Huang H, Wang H. Incremental deployment and throughput maximization routing for a hybrid SDN. *IEEE/ACM Trans Network(TON)*. 2017;25(3):1861-1875.
8. OpenFlow Switch Specification March 2015 [Available from: https://www.opennetworking.org
9. Chen H, Li L, Ren J, et al. A scheme to optimize flow routing and polling switch selection of software defined networks. *PloS Oone*. 2015;10(12):e0145437.

10. Li W, Meng W, Kwok LF. A survey on OpenFlow-based software defined networks: security challenges and countermeasures. *J Network Comp Appl*. 2016;68:126-139.
11. Al-Najjar A, Layeghy S, Portmann M, editors. Pushing SDN to the end-host, network load balancing using OpenFlow. 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops); 2016 14-18 March 2016.
12. Pakzad F, Portmann M, Tan WL, Indulska J. Efficient topology discovery in openflow-based software defined networks. *Comp Comm*. 2016;77:52-61.
13. Gude N, Koponen T, Pettit J, et al. NOX: towards an operating system for networks. *ACM SIGCOMM Comp Comm Rev*. 2008;38(3):105-110.
14. Shenker S, Casado M, Koponen T, McKeown N. The future of networking, and the past of protocols. *Open Network Summit*. 2011;20.
15. Fundation ON. Software-defined networking: the new norm for networks. ONF White Paper. April 2012;2:2.6-.1.
16. Lin W, Zhang L. The load balancing research of SDN based on ant colony algorithm with job classification. 2016.
17. Lin T-L, Kuo C-H, Chang H-Y, Chang W-K, Lin Y-Y, editors. A parameterized wildcard method based on SDN for server load balancing. 2016 International Conference on Networking and Network Applications (NaNA) ; 2016: IEEE.
18. Long H, Shen Y, Guo M, Tang F, editors. LABERIO: dynamic load-balanced routing in OpenFlow-enabled networks. Advanced Information Networking and Applications (AINA), 27th International Conference on 2013 IEEE; 2013: IEEE.
19. Karakus M, Durresi A. A survey: control plane scalability issues and approaches in software-defined networking (SDN). *CompNetworks*. 2016.
20. Chou L-D, Yang Y-T, Hong Y-M, Hu J-K, Jean B. A genetic-based load balancing algorithm in openflow network. In: *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*. Springer; 2014:411-417.
21. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comp Comm Rev*. 2008;38(2):69-74.
22. Hu F, Hao Q, Bao K. A survey on software-defined network and openflow: from concept to implementation. *IEEE Comm Surveys Tutor*. 2014;16(4):2181-2206.
23. Vaughan-Nichols SJ. OpenFlow: the next generation of the network? *Comput*. 2011;44(8):13-15.
24. OpenFlow Switch Specification,1.5.1 [Available from: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.
25. Foundation ON. SDN architecture June, 2014 [Available from: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf.
26. Zhou W, Yang S, Fang J, Niu X, Song H, editors. Vmctune: a load balancing scheme for virtual machine cluster using dynamic resource allocation. 9th International Conference on Grid and Cooperative Computing (GCC), 2010; 2010: IEEE.
27. Chin ML, Tan CE, Bandan MI, editors. Efficient load balancing for bursty demand in web based application services via domain name services. 8th Asia-Pacific Symposium onInformation and Telecommunication Technologies (APSITT), 2010; 2010: IEEE.
28. Salman MA, Bertelle C, Sanlaville E, editors. The behavior of load balancing strategies with regard to the network structure in distributed computing systems. Tenth International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), 2014; 2014: IEEE.
29. Chen S-L, Chen Y-Y, Kuo S-H. CLB: a novel load balancing architecture and algorithm for cloud services. *Comp Electric Eng*. 2017;58:154-160.
30. Milani AS, Navimipour NJ. Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends. *J Network Comp Appl*. 2016;71:86-98.
31. Alakeel AM. A guide to dynamic load balancing in distributed computer systems. *Int J Comp Sci Info Secur*. 2010;10(6):153-160.
32. Celenlioglu MR, Mantar HA, editors. An SDN based intra-domain routing and resource management model. IEEE International Conference on Cloud Engineering (IC2E), 2015; 2015: IEEE.
33. Tuncer D, Charalambides M, Clayman S, Pavlou G. Adaptive resource management and control in software defined networks. *IEEE Trans Network Service Mgmt*. 2015;12(1):18-33.
34. He X, Ren Z, Shi C, Fang J. A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles. *Chin Comm*. 2016;13(Supplement 2):140-149.
35. Han T, Ansari N. A traffic load balancing framework for software-defined radio access networks powered by hybrid energy sources. *IEEE/ACM Trans Network(TON)*. 2016;24(2):1038-1051.
36. Hu Y, Luo T, Beaulieu NC, Wang W. An initial load-based green software defined network. *Appl Sci*. 2017;7(5):459.
37. Carlinet Y, Perrot N, editors. Energy-efficient load balancing in a SDN-based Data-Center network. 17th International Telecommunications Network Strategy and Planning Symposium (Networks), 2016; 2016: IEEE.

38. Lin Y-D, Wang CC, Lu Y-J, Lai Y-C, Yang H-C. Two-tier dynamic load balancing in SDN-enabled Wi-Fi networks. *Wireless Networks*. 2017;1-13.
39. Ca W, Hu B, Chen S, Li D, Liu B. A switch migration-based decision-making scheme for balancing load in SDN. *IEEE Access*. 2017;5:4537-4544.
40. Chen Y, Li Q, Yang Y, Li Q, Jiang Y, Xiao X, editors. Towards adaptive elastic distributed software defined networking. IEEE 34th International PerformanceComputing and Communications Conference (IPCCC), 2015; 2015: IEEE.
41. Bari MF, Roy AR, Chowdhury SR, Zhang Q, Zhani MF, Ahmed R, et al., editors. Dynamic controller provisioning in software defined networks. 9th International Conference on Network and Service Management (CNSM), 2013; 2013: IEEE.
42. Koushika A, Selvi ST, editors. Load balancing using software defined networking in cloud environment. International Conference on Recent Trends in Information Technology (ICRTIT), 2014; 2014: IEEE.
43. Rangisetti AK, Tamma BR. QoS aware load balance in software defined LTE networks. *Comp Comm*. 2017;97:52-71.
44. Mahlab U, Omiyi PE, Hundert H, Wolbrum Y, Elimelech O, Aharon I, et al., editors. Entropy-based load-balancing for software-defined elastic optical networks. 19th International Conference on Transparent Optical Networks (ICTON), 2017; 2017: IEEE.
45. Hai NT, Kim D-S, editors. Efficient load balancing for multi-controller in SDN-based mission-critical networks. IEEE 14th International Conference on Industrial Informatics (INDIN), 2016; 2016: IEEE.
46. Guo Z, Su M, Xu Y, et al. Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Comp Networks*. 2014;68:95-109.
47. Sang X, Wu Q, Li H, editors. Client-network collaborative load balancing mechanism for WLAN based on SDN and 802.11 u. 13th International Wireless Communications and Mobile Computing Conference (IWCMC), 2017; 2017: IEEE.
48. Kang B, Choo H. An SDN-enhanced load-balancing technique in the cloud system. *J Supercomp*. 2016;1-24.
49. Zhong H, Fang Y, Cui J. LBBSRT: an efficient SDN load balancing scheme based on server response time. *Future Gen Comp Syst*. 2017;68:183-190.
50. Fizi FS, Askar S, editors. A novel load balancing algorithm for software defined network based datacenters. International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom); 2016: IEEE.
51. Daraghmi EY, Yuan S-M. A small world based overlay network for improving dynamic load-balancing. *J Syst Softw*. 2015;107:187-203.
52. Subramanian R, Manoranjitham T. Dynamic scheduling for traffic management and load balancing using SDN.
53. Deepika DW, Kumar N. Performance analysis of load balancing algorithms in distributed system. *Adv Electron Electric Eng*. 2014;4(1):59-66.
54. Adami D, Giordano S, Pagano M, Santinelli N, editors. Class-based traffic recovery with load balancing in software-defined networks. Globecom Workshops (GC Wkshps), 2014; 2014: IEEE.
55. Sharma S, Singh S, Sharma M. Performance analysis of load balancing algorithms. *World Acad Sci, Eng Technol*. 2008;38(3):269-272.
56. Li L, Xu Q, editors. Load balancing researches in SDN: a survey. 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), 2017; 2017: IEEE.
57. Benzekki K, El Fergougui A, Elbelrhiti EA. Software-defined networking (SDN): a survey. *Sec Comm Networks*. 2016;9(18):5803-5833.
58. Raghul S, Subashri T, Vimal K, editors. Literature survey on traffic-based server load balancing using SDN and open flow. Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), 2017; 2017: IEEE.
59. Kumari P, Thakur D. *Load Balancing in Software Defined Network*. 2017, 5, 12, 227, 232.
60. Neghabi AA, Navimipour NJ, Hosseinzadeh M, Rezaee A. Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature. *IEEE Access*. 2018;6:14159-14178.
61. Kitchenham B. *Procedures for Performing Systematic Reviews*. Keele, UK, Keele University. 2004;33(2004):1-26.
62. Charband Y, Navimipour NJ. Online knowledge sharing mechanisms: a systematic review of the state of the art literature and recommendations for future research. *Info Syst Front*. 2016;18(6):1131-1151.
63. Google. Google scholar [Available from: https://scholar.google.com/.
64. publications E. Sciencedirect [Available from: https://www.sciencedirect.com/.
65. publications I. IEEE Xplore [Available from: https://ieeexplore.ieee.org/Xplore/home.jsp.
66. publications S. *SAGE J*.
67. publications A. Association for Computing Machinery [Available from: https://www.acm.org/.
68. publications S. Springer link [Available from: https://link.springer.com/.
69. publications W. Wiley online library [Available from: https://onlinelibrary.wiley.com/.
70. publications E. EmeraldInsight [Available from: https://www.emeraldinsight.com/.
71. publications I. Inderscience publishers [Available from: www.inderscience.com.

72. Reim W, Parida V, Örtqvist D. Product-service systems (PSS) business models and tactics—a systematic literature review. *J Clean Prod*. 2015;97:61-75.

73. Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Linkman S. Systematic literature reviews in software engineering—a systematic literature review. *Info Softw Technol*. 2009;51(1):7-15.

74. Sathyanarayana S, Moh M, editors. Joint route-server load balancing in software defined networks using ant colony optimization. International Conference on High Performance Computing & Simulation (HPCS), 2016; 2016: IEEE.

75. Wang C, Zhang G, Xu H, Chen H, editors. An ACO-based link load-balancing algorithm in SDN. 7th International Conference on Cloud Computing and Big Data (CCBD), 2016; 2016: IEEE.

76. Tsygankov M, Chen C, editors. Network aware VM load balancing in cloud data centers using SDN. IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 2017; 2017: IEEE.

77. Fang K-C, Wang K, Wang J-H, editors. A fast and load-aware controller failover mechanism for software-defined networks. Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2016 10th International Symposium on; 2016: IEEE.

78. Gu S, Luo L, Zhao Z, Li X, editors. The multi-objective routing optimization algorithm for hybrid SDN. Conference of Spacecraft TT&C Technology in China; 2016: Springer.

79. Li L, Shi J, editors. A highly reliable and load balance supporting domain division algorithm for software defined networks. Proceedings of the International Conference on High Performance Compilation, Computing and Communications; 2017: ACM.

80. Kang S-B, Kwon G-I. Load balancing of software-defined network controller using genetic algorithm. 2016;9:881-888.

81. Han Y, Seo S-S, Li J, Hyun J, Yoo J-H, Hong JW-K, editors. Software defined networking-based traffic engineering for data center networks. Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific; 2014: IEEE.

82. Tu R, Wang X, Zhao J, Yang Y, Shi L, Wolf T, editors. Design of a load-balancing middlebox based on SDN for data centers. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2015; 2015: IEEE.

83. Boero L, Cello M, Garibotto C, Marchese M, Mongelli M. BeaQoS: load balancing and deadline management of queues in an OpenFlow SDN switch. *Comp Networks*. 2016;106:161-170.

84. Govindarajan K, Kumar VS, Editors. An intelligent load balancer for software defined networking (SDN) based cloud infrastructure. 2017 second international conference on electrical, Computer and Communication Technologies (ICECCT); 2017 22-24 Feb. 2017.

85. Zhu R, Wang H, Gao Y, Yi S, Zhu F, editors. Energy saving and load balancing for SDN based on multi-objective particle swarm optimization. International Conference on Algorithms and Architectures for Parallel Processing; 2015: Springer.

86. Li X, Wu H, Gruenbacher D, Scoglio C, Anjali T. Efficient routing for middlebox policy enforcement in software-defined networking. *Comp Networks*. 2016;110:243-252.

87. Dorigo M. Optimization, learning and natural algorithms (Ph. D. Thesis). Dipartimento diElettronica, Politecnico di Milano, Italy. 1992.

88. Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. *IEEE Trans Syst, Man, Cybernet, Prt B (Cybernetics)*. 1996;26(1):29-41.

89. Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evol Comp*. 1997;1(1):53-66.

90. Wang Z, Xing H, Li T, Yang Y, Qu R, Pan Y. A modified ant colony optimization algorithm for network coding resource minimization. *IEEE Trans Evol Comp*. 2016;20(3):325-342.

91. Narasimha KV, Kivelevitch E, Sharma B, Kumar M. An ant colony optimization technique for solving min-max multi-depot vehicle routing problem. *Swarm Evol Comp*. 2013;13:63-73.

92. Salama KM, Freitas AA. Classification with cluster-based Bayesian multi-nets using ant colony optimisation. *Swarm Evol Comp*. 2014;18:54-70.

93. Goldberg DE, Holland JH. Genetic algorithms and machine learning. *Machine Learning*. 1988;3(2):95-99.

94. Holland JH. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press; 1992.

95. Gen M, Cheng R. *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons; 2000.

96. Sastry K, Goldberg DE, Kendall G. Genetic algorithms. *Search Methodologies*: Springer; 2014. p. 93-117, Genetic Algorithms.

97. Beheshti ZS, Siti Mariyam HJ. A review of population-based meta-heuristic algorithm. *Int J Adv Soft Comp Appl*. 2013;5.

98. Chan Y-C, Wang K, Hsu Y-H, editors. Fast controller failover for multi-domain software-defined networks. European Conference on Networks and Communications (EuCNC), 2015; 2015: IEEE.

99. Li D, Ruan L, Xiao L, Zhu M, Duan W, Zhou Y, et al., editors. High availability for non-stop network controller. IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014; 2014: IEEE.

100. Müller LF, Oliveira RR, Luizelli MC, Gaspary LP, Barcellos MP, editors. Survivor: an enhanced controller placement strategy for improving SDN survivability. Global Communications Conference (GLOBECOM), 2014 IEEE; 2014: IEEE.

101. Obadia M, Bouet M, Leguay J, Phemius K, Iannone L, editors. Failover mechanisms for distributed SDN controllers. International Conference and Workshop on the Network of the Future (NOF), 2014; 2014: IEEE.

102. Abramson D, Abela J. A parallel genetic algorithm for solving the school timetabling problem. 1991.

103. DeVore RA, Temlyakov VN. Some remarks on greedy algorithms. *Adv Computat Math*. 1996;5(1):173-187.

104. Bang-Jensen J, Gutin G, Yeo A. When the greedy algorithm fails. *Discrete Optimiz*. 2004;1(2):121-127.

105. Ivancevic VG, Ivancevic TT. *Computational Mind: A Complex Dynamics Perspective*. Springer; 2007.

106. Hsieh W-K, Hsieh W-H, Chen J-L, Chou F-Y, Lee Y-S, editors. Load balancing virtual machines deployment mechanism in SDN open cloud platform. 17th International Conference on Advanced Communication Technology (ICACT), 2015; 2015: IEEE.

107. Greenberg A, Hamilton J, Maltz DA, Patel P. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Comp Comm Rev*. 2008;39(1):68-73.

108. Beloglazov A, Buyya R, editors. Energy efficient resource management in virtualized cloud data centers. Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing; 2010: IEEE Computer Society.

109. Clos C. A study of non-blocking switching networks. *Bell Labs Tech J*. 1953;32(2):406-424.

110. Eberhart R, Kennedy J, editors. A new optimizer using particle swarm theory. Micro Machine and Human Science, 1995 MHS'95, Proceedings of the Sixth International Symposium on; 1995: IEEE.

111. Kennedy J. Particle swarm optimization. In: *Encyclopedia of Machine Learning*. Springer; 2011:760-766.

112. Shi Y, editor. Particle swarm optimization: developments, applications and resources. Proceedings of the 2001 Congress on Evolutionary computation, 2001; 2001: IEEE.

113. Sheikholeslami F, Navimipour NJ. Service allocation in the cloud environments using multi-objective particle swarm optimization algorithm based on crowding distance. *Swarm Evol Comp*. 2017;35:53-64.

114. Kirkpatrick S. Optimization by simulated annealing: quantitative studies. *J Statistic Phys*. 1984;34(5-6):975-986.

115. Fernando N, Loke SW, Rahayu W. Mobile cloud computing: a survey. *Future Gen Comp Syst*. 2013;29(1):84-106.

116. Navimipour NJ, Milani FS. A comprehensive study of the resource discovery techniques in peer-to-peer networks. *Peer-To-Peer Network Appl*. 2015;8(3):474-492.

117. Liu L, Zhang T, Zhang J. DAG based multipath routing algorithm for load balancing in machine-to-machine networks. *Int J Distrib Sensor Networks*. 2013;10(1):457962.

118. Geng R, Ning Z, Ye N. A load-balancing and coding-aware multicast protocol for mobile ad hoc networks. *Int J Comm Syst*. 2016;29(17):2457-2470.

119. Yang X-S, Hossein GA. Bat algorithm: a novel approach for global engineering optimization. *Eng Comp*. 2012;29(5):464-483.

120. Alauddin M. V-MFO: variable flight mosquito flying optimization. In: *Applications of Soft Computing for the Web*. Springer; 2017:271-283.

121. Hosseini F, Kaedi M. A metaheuristic optimization algorithm inspired by the effect of sunlight on the leaf germination. *Int J Appl Metaheuristic Comp(IJAMC)*. 2018;9(1):40-48.

122. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Adv Eng Softw*. 2014;69:46-61.

123. Mirjalili S, Lewis A. The whale optimization algorithm. *Adv Eng Softw*. 2016;95:51-67.

124. Yazdani M, Jolai F. Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm. *J Comput Des Eng*. 2016;3(1):24-36.

125. Srinivas M, Patnaik LM. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans Syst Man Cybern*. 1994;24(4):656-667.

126. Amor HB, Rettinger A, editors. Intelligent exploration for genetic algorithms: using self-organizing maps in evolutionary computation. Proceedings of the 7th annual conference on Genetic and evolutionary computation; 2005: ACM.

127. dos Santos Coelho L. A quantum particle swarm optimizer with chaotic mutation operator. *Chaos, Solit Fractals*. 2008;37(5):1409-1418.

128. Choubey NS, Kharat MU. Approaches for handling premature convergence in CFG induction using GA. In: *Soft Computing in Industrial Applications*. Springer; 2011:55-66.

129. Kharkongor C, Chithralekha T, Varghese R. A SDN controller with energy efficient routing in the internet of things (IoT). *Proc Comp Sci*. 2016;89:218-227.

How to cite this article: Akbar Neghabi A, Jafari Navimipour N, Hosseinzadeh M, Rezaee A. Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network. *Int J Commun Syst*. 2019;e3875. https://doi.org/10.1002/dac.3875