A Survey on Routing Algorithms and Techniques Used to Improve Network Performance in Software-Defined Networking

Suryanarayan Menon A.
Department of Information Technology
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
suryanarayan.trv19it055@gecbh.ac.in

Sanjay J Prakash
Department of Information Technology
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
sanjay.trv19it048@gecbh.ac.in

Vinayak Naveen
Department of Information Technology
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
vinayak.trv19it058@gecbh.ac.in

Roshan Aji Cherian
Department of Information Technology
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
roshan.trv19it046@gecbh.ac.in

Ron Regi Zacharia
Department of Information Technology
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
ron.trv19it045@gecbh.ac.in

Suryapriya S.
Asst.Professor, Dept. of IT
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
suryapriya.s@gecbh.ac.in

Josna VR
Asst.Professor, Dept. of IT
Government Engineering College,
Barton Hill
Thiruvananthapuram, Kerala, India
josna.vr@gecbh.ac.in

***Abstract*—The growth of network infrastructure over the years, including the increasing number of devices and users connecting to networks, the increasing complexity of network environments, and the need for more flexible and efficient network management, has led to the emergence of Software-Defined Networking (SDN) as a feasible means of support. SDN is a networking approach that uses software to abstract, manage, and control the network infrastructure. In SDN, the control plane, which determines how data is routed through the network, is separated from the data plane, which forwards the traffic. This separation allows the control plane to be more flexible and programmable, enabling administrators to easily configure and manage the network using the software. There are several routing algorithms and techniques that can be used to improve network performance in Software-Defined Networking. Shortest Path Routing is a common routing algorithm along with Segment Routing and may be capable of speeding up the network's overall performance by decreasing the time it takes for traffic to reach its destination. Load Balancing is a technique that involves distributing traffic across multiple resources, such as servers or network devices, to optimize the use of resources and improve the performance of the network. For this, there exist several techniques such as Round Robin and Ant Colony Optimization. Another such technique used to optimize the use of resources and improve the overall performance of the network is Traffic Engineering - where software is used to route traffic through the network based on various criteria, such as the type of traffic, the source, and destination of the traffic, or the current load on the network. There have been numerous studies in these fields investigating the feasibility of novel systems and solutions that have been proposed for improving network performance. This paper goes into the methodologies followed for these systems and the impacts they have had on generating increments in network performance.**

Keywords—Software Defined Networking; Segment Routing; Traffic Engineering; Load Balancing; Bio-inspired Routing; Ant Colony Optimization; Multi-path routing; Path Encoding

I. Introduction

Over the years, traditional Internet Protocol (IP) networks have needed amelioration to be able to tackle huge amounts of network traffic brought about by the growth of network devices. One of the solutions for this has been the proposal of Software Defined Networking (SDN). The principle of SDN is to have the network control logic be separated from underlying network hardware like routers and switches, thereby focusing on the centralization of network control. In SDN, OpenFlow is a protocol that has been used as a communication standard between the controller plane and the hardware plane.

SDN is different from existing traditional networks utilizing physical routers and switches. In SDN, the control plane is implemented in software, while the data plane is implemented in hardware. This separation allows for the control plane to be more flexible and easy to modify, as it can be changed through software updates rather than requiring physical changes to the network infrastructure. As a result, One of SDN's key benefits is its ability to centrally control and manage the network. In traditional networks, each network device has its own control plane and is responsible for routing packets to the correct destination. In an SDN network, a central controller becomes responsible for managing the data flow through the network, thereby making it easier to optimize performance and troubleshoot issues.

Data forwarding, packet routing, energy management, and load balancing are a few areas that can increase performance but are currently being worked on since the idea of software-defined networking is still relatively new. Routing presents challenges in conventional networks due to the ineffective utilization involved. This is especially true when the network is overrun with packets [1]. By dividing the data plane from the control plane, SDN makes it possible for routing and traffic control algorithms to be far more effective than they are in conventional networks today. However, as the network management requirement becomes more exact, network traffic increases dramatically. Due to their sluggish convergence and poor reaction to network changes, conventional routing algorithms like Open Shortest Path First (OSPF) are inappropriate for these situations [2]. Conversely, the Ant-Colony Optimization (ACO) algorithm is able to adapt to network changes and identify effective solutions even in the face of traffic or other interruptions [3]. It can choose pathways that are less crowded or have lower latency by taking into consideration the status of the network at the moment. Their usefulness for routing in dynamic and extremely dynamic networks is a result of this.

II. Performance parameters

The several performance metrics that were taken into account in the publications that we analyzed are listed in this section. Along with this, we have also spoken about the approaches used by the respective authors to resolve problems and make advancements in the aforementioned aspects. In Table II.1, a summary of these data has been provided.

TABLE II.1 LIST OF METHODOLOGIES

S.No	**Method**	**Parameters Considered**	**Technique Used**
1	Dynamic Load Balancing of SDN [1]	Load Balancing - Round Trip Time, Packet Loss, Optimal Path	Genetic Algorithm coupled with Ant Colony Optimization Algorithm
2	GOP-SDN for Load Balancing [4]	Load Balancing - Throughput, Response Time, Migration Numbers	Genetic Algorithm and Particle Swarm Optimization
3	DROM [2]	Delay, Throughput	Deep Reinforcement Learning, Deterministic Policy Gradient
4	SmartBlock-SDN [16]	Throughput, Resource Utilization	Blockchain, Cluster Head Selection
5	Ant Colony Optimization-based Method [3]	Delay, Throughput	Ant Colony Optimization Algorithm
6	Ant Colony Genetic Fusion Routing [17]	Delay, Throughput, Resource Utilization	Ant Colony Genetic Fusion Routing Algorithm
7	An Optimization Routing Algorithm Method [18]	Load Balancing - Path Cost, Optimal Link Utilisation	Segment Routing, MOPSO
8	RLMR [24]	Optimal Link Utilization	Reinforcement Learning Based Multipath Routing
9	PASR [27]	Flow Entries, Flow Table Overflow, Coincident paths	Segment Routing, Path Aggregation
10	LFOD [31]	Flow Entries, Flow Table Overflow	Flow Table Optimization based on Flow Length Distribution

A. Load Balancing

Load Balancing (LB) is a procedure for allocating load to network components in order to enhance Quality of Service (QoS) [4] and improve network performance. LB has become an important task in maximizing network performance, scalability and robustness. This has only become even more important because of the development of Software-Defined Networking. In SDN, by distributing network traffic among resources, LB makes sure that no one resource is overburdened. hence maximizing performance. For this, there have been several studies on several existing conventional methods for Load Balancing [5], however there has also been considerable developments in load balancing using bio-inspired optimization algorithms [6], [7].

Xue et al. [1] proposed a novel approach for dynamic load balancing of SDN that combined a Genetic Algorithm (GA) [8] with Ant Colony Optimization (ACO) [9] that aimed at enhancing the Round Robin time and the ACO algorithm in terms of optimal path search rate, round trip duration, and packet loss rate. At present, the ACO algorithm in load balancing of SDN employs a positive feedback method to update flow path information, although this may result in a local optimum solution and thus incorrect path selection. A random selection policy could be used to avoid a locally optimum solution, but as a tradeoff, this doesn't even guarantee a suboptimal solution. These issues hinder ACO's performance in path selection. However, GA can be adopted with ACO to alleviate these problems. In the second stage, applying GA can reduce the search space for ACO to more efficiently find paths for LB. Simulations showed that as compared to Round Robin and

ACO algorithms on their own, this approach would dramatically improve Round Trip Time (RTT) and packet delivery ratio [1].

To evaluate the performance of the system, an experiment was conducted in [1] using OpenDayLight and Mininet. It was observed that The proposed G-ACO technique provided a 95% success rate of finding the optimal path, which was notably higher than those for ACO and GA schemes on their own. Using the iperf software tool [10], Round Robin, ACO and the proposed G-ACO system were simulated. In a 5-minute simulation, it was discovered that the RTT and packet loss rate of the G-ACO scheme were nearly identical to those of the ACO scheme. For cases where the simulation time was almost as small as 5 minutes, computation overhead for GA in the initial phase was high and compounded information for path selection was insufficient [1]. With an increase to 10 minutes of simulation, the proposed system dramatically lowered the RTT and packet loss rate in contrast to the RR and ACO techniques. For the Round Robin scheme specifically, the packet loss rate was quite high due to path congestion for a particular load. For the same load, the ACO scheme was seen to have slightly improved Round Trip Time when compared to the RR setup whereas the packet loss rate was significantly reduced. However, the suggested G-ACO method was successful for LB because time delay and packet loss were consistent regardless of destination.

Zahra et al. [4] proposed GOP-SDN: an improved load balancing solution in distributed SDNs based on the Genetic Algorithm and the Optimized Particle Swarm Optimization (OPSO) algorithm. This method combines two optimization algorithms, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [11], to improve load balancing performance in distributed SDN environments. The scheme incorporates multiple controllers and switches instead of a single centralized controller and addresses the issues of scalability and reliability that are commonly seen in multi-controller environments by creating load balancing on controllers themselves. The model uses a combination of GA and PSO algorithms to allocate jobs to resources in the most effective manner possible. GOP-SDN uses PSO algorithm to select the target controller, while also taking controller load, propagation delay, and load balancing into account. It also uses GA with variant PSO based mutation for generic controller placement in SDNs [12]. This lessens the number of switch migrations among controllers while increasing network throughput.

For performance evaluation, a case study was performed with an example assuming 5 controllers connected to 8 switches [4]. The experiment involved calculating workload of each controller which was then compared with a threshold value to determine whether there is load imbalance and whether migration cycle should be entered or not. The Genetic Algorithm was used to select a sample of enhanced populations with the best fitness function, which were then entered as particles into the OPSO algorithm. The OPSO algorithm here actually comprises two PSO algorithms - one for particle movement and the other for optimizing the particle's weight in the speed formula used [4]. Through the case study and simulations run for scenarios of 300 seconds, output obtained was compared with existing metrics for Online Controller Load Balancing (OCLB) method proposed in [13]. Results showed an average improvement in throughput of 24.72% over OCLB, mainly due to the use of variable thresholds in detection of load imbalance [4]. In terms of Migration numbers, the system showed significant reduction in the number of migrations whilst at the same time not producing a tangible increase in response time, which is an advantage over the OCLB model due to the network's higher input traffic rate of two to three times.

B. Throughput And Delay

Throughput refers to how much data can be transferred from the source to the destination within a given timeframe. Delay is the latency for a bit of data to travel from one endpoint to another. As the size of the network increases, reducing the delay and improving the throughput are essential to maintain the network quality.

One of the proposed systems to improve these parameters is DDPG Routing Optimization Mechanism (DROM) [2]. It is a routing protocol based on deep reinforcement learning that can be implemented in Software-Defined Networks. DROM combines the perception ability of deep learning with the decision-making ability of enhanced learning. It utilizes neural networks to generate the strategy function and Q function, enabling it to learn and adapt to changing network conditions. DROM agents interact with the environment using three parameters: state, action, and reward. The state is the present traffic matrix [2], action refers to the action taken by the agent i.eworks to improve throughput and minimize the delay.Th. the change made in the current traffic matrix. By changing the weight of the link, the data flow path can be altered. The reward can be any performance parameter such as delay or throughput obtained after the change in the data flow path.

DROM when compared with other standard routing protocols such as OSPF under different traffic loads for latency in the network, DROM showed a substantial reduction in delay. Also when DROM was compared with throughput optimal solutions such as SDN-LB [14] and QAR [15] under different traffic loads, showed that DROM had better throughput as compared to its alternatives.

Anichur et al. [16] proposed a solution, SmartBlock-SDN, to enhance the network throughput by integrating Software-Defined Networking (SDN) with Blockchain technology. Their proposed framework, the Hierarchical Blockchain-Enabled SDN, guarantees secure network communication while optimizing various network metrics, with a special focus on throughput.

The architecture of the framework is composed of three layers: the Perception Layer, the Edge Layer, and the Cloud Layer. The IoT environment consists of IoT sensors and devices, the SDN environment comprises the data and control planes, and the Cloud Layer encompasses the data centers and the Blockchain Environment. Two separate blockchain systems are employed: one for the control layer and the other for the data layer. The control layer blockchain stores the distributed flow rules, ensuring the consistency of flow rules across each cluster. In contrast, the data layer blockchain operates differently. All switches dump their flow rules in the blockchain, and their consistency is verified. In case any switch doesn't maintain the same ruleset, the record is not updated, and the switch is isolated from the environment. This hierarchical Blockchain-Enabled SDN framework provides a secure, reliable, and efficient solution to improve the network's throughput and overall performance.

The purpose of the study in [3] is to design and implement Ant Colony Optimisation (ACO), a strategy for dynamic routing in SDN inspired by biological activity.The proposed system in Shreya et al. [3] is comprised of three planes, namely, a control plane, a data plane, and an application plane. These planes are interconnected via the OpenFlow API. The control plane employs a bio-inspired Ant Colony Optimization technique, which includes three modules. Firstly, the Compute Multiple Path module is responsible for computing multiple paths from the source to the destination. Secondly, the Select Optimal Path module is responsible for determining the optimal routing based on the link's cost and the number of transmitted packets. Lastly, the Validate Path module is utilized to assess the validity of the selected path.

One of the issues faced in network routing is adaptability. In addition to dealing with unpredictable traffic volumes, network topologies can undergo changes due to the addition of new nodes and removal of old ones. To handle network routing, Dijkstra's algorithm is commonly employed as a link state method. The algorithm effectively routes traffic through the network while minimizing the number of connected nodes. However, this system does not account for incoming traffic or load balancing, which can lead to reduced throughput and delays. As a result, the algorithm's functionality can be limited in dynamic network environments where traffic patterns and network loads can fluctuate over time. In their research, Shreya et al. [3] evaluated the proposed algorithm's effectiveness using Quality of Service (QoS) metrics, such as average throughput and average delay. The algorithm utilizes Ant Colony Optimization (ACO) to enhance QoS by computing multiple paths from the source to the destination, taking into account the link's load, and selecting the path with the lowest load. In contrast, Dijkstra's algorithm only computes the shortest path. By performing a comprehensive comparison of the two algorithms, the authors showed that ACO can significantly improve network performance, particularly in terms of throughput and delay.

The study conducted by [3] employed One-way Analysis of Variance (ANOVA) to examine the impact of two levels of a single factor. It was observed that the delay variation was higher than the throughput variation, signifying that the ACO method performed better in terms of delay and is therefore suitable for multimedia applications. The obtained results demonstrated that the ACO-based routing outperformed Dijkstra's routing by optimizing throughput and reducing delay. This highlights the potential of the ACO algorithm to offer improved performance in network routing compared to conventional methods.

The research done by Zhao et. al [17] presents a new routing algorithm for Software-Defined Networking (SDN) that incorporates both Ant Colony Optimization (ACO) and Genetic Algorithm (GA) techniques. The objective of the suggested algorithm is to generate a robust, efficient, and effective routing solutions in virtual networks. The algorithm first employs ACO to extract the path characteristics and then utilizes GA to minimize the connection cost and identify the optimal path [17].

The performance of the proposed algorithm was evaluated through simulations and compared to four existing routing algorithms based on key metrics such as network throughput, average delay, packet loss rate, and resource utilization. The results indicated that the proposed algorithm was superior to the alternative methods in terms of routing accuracy, data throughput, and packet delivery ratio, and also improved the overall network efficiency by reducing the average end-to-end delay. This innovative approach has the potential to address the challenges posed by complex network environments and holds significant value for future SDN network routing.

C. *Optimal link utilization*

Users or networks need different network services based on their various requirements. Optimal link utilization, load balancing, and minimizing the cost of the network path are some of them. Link utilization can be expressed as a percentage of the total link capacity on a network. When the maximum link utilization is smaller, the remaining bandwidth will be greater, and there is very little chance of network congestion. There are many ways to improve these parameters in the SDN environment

The paper [18] uses Segment Routing for optimizing routing algorithms in SDN for optimal link utilization. There are many traditional routing algorithms, such as Minimum Interference Routing Algorithm (MIRA) [19], Shortest Path First (SPF) [20], Shortest Widest Path (SWP) [21], and Widest Shortest Path (WSP) [22]. Segment Routing (SR) in SDN is a forward routing technique that enables the creation of traffic paths by encoding instructions, or "segments," into packets. The working principle of the above method using SR includes using Multiple Objective Particle Swarm Optimization (MOPSO) [23] to enhance the link weight for the cost of the path and load balance. K shortest paths [18] between the source and the destination nodes are taken for the improved weighted matrix. An improved weighted matrix was maintained to find the final evaluation value for the K shortest paths [18]. From this matrix, the path with the highest evaluation value was taken according to the user's preference. When this proposed method was compared with other routing algorithms there was much improvement in the network path being chosen for the routing, which in turn helped in improving the link utilization and a more accurate solution was obtained.

In the paper [24], the authors propose a novel method for routing algorithms in software-defined networks (SDN). The authors aim to optimize link utilization by utilizing Reinforcement Learning-based Multipath Routing (RLMR)[24]. The authors use a deep reinforcement learning framework to make more informed routing decisions by choosing the best path out of multiple available paths based on parameters such as state representation, action space, reward function, and a deep reinforcement learning framework [25]. The system is built on a deep Q-network (DQN) [26] which adjusts the routing decisions dynamically based on feedback from the reward function and action space. The authors conducted experiments to demonstrate the effectiveness of their approach in balancing traffic across multiple paths and improving link utilization, network performance, and quality of service in SDN. Their results revealed that the suggested approach surpassed traditional routing algorithms in link utilization and network performance, with a more balanced traffic distribution and reduced congestion. The authors also discussed the challenges and constraints of the proposed method and suggested directions for future projects. Overall, the paper offers insightful information about Reinforcement Learning's use in SDN routing and illustrates how it can enhance network performance and service quality.

D. Reduction of Flow Entries to Avoid Flow Table Overflow

Software Defined Networks (SDN) achieve fine-grained path control by installing flow entries [27]. When these flow entries are installed, a switch consults its flow table to determine the direction of the data flow. Flow entries are stored by utilizing a Ternary Content Addressable Memory (TCAM) which can search all entries in one single clock cycle. However, there is an issue that this system poses; it leads to flow-table overflow. Due to a lack of space in the flow table, inefficiencies arise when there are several incoming data flows. This inhibits the normal transmission of data flow as it prevents further installation of entries. As we scale up our SDN, the traffic of flows increases and flow-table overflow becomes more significant and critical.

Many schemes have been put forward to mitigate the issue of high volumes of flow entries. The traditional OpenFlow method requires the installation of flow entries for every switch. Due to the installation of several entries for scaled-up systems, this technique proves to be inefficient. Another method that Dong et al. [28] have mentioned is the Multiprotocol Label Switching (MPLS) forwarding scheme which utilizes labels to route a path for the flow. Although this system reduces flow entries, it gives rise to another problem; the label load incurred is very large. Furthermore, Arbitrary Jump Source Routing (AJSR), a system proposed by Dong et al. [28] which takes inspiration from the JumpFlow scheme by Guo et al. [29] helps to resolve the issue of label load. The entire routing path of a specific flow is divided by AJSR into various pieces of arbitrary sizes.

Even though we see a reduction in the number of flow entries in the above systems, it is still not sufficient. Therefore a scheme based on segment routing and an intelligent path encoding algorithm, namely Path Aggregation Segment Routing (PASR) has been proposed by Li et al. [27]. PASR has two databases; Flow Path Database (FPD) and a Path Segment Database (PSD). The FPD is used to store information on the status of the network flow in real-time and PSD stores path aggregation information along with previous path segmentation rules. As a flow request is received by the SDN Controller, the Path Computation Module figures out the optimal path using algorithms like Shortest Path First, considering the network resources and QoS requirements. Then the Path Encoding Module performs path segmentation and aggregation using information contained in the FPD and PSD, after which the SDN Controller installs flow entries to the switches considering the aggregated paths.

Through path aggregation, PASR proves to be an effective scheme for reducing flow entries. When compared to OpenFlow and AJSR there is quite a significant difference in flow entries. OpenFlow installs entries for every switch along the path and flow table redundancies are seen. In the case of AJSR, we see that PASR is able to reduce the flow entries by 40% as observed by Li et al. [27]. Although AJSR performs segmentation on the routing path, a lot of flow table redundancies can still be noted. PASR, on the other hand, works based on the coincidence degree of data flows. By utilizing the coincidence degree, PASR can aggregate paths and reduce the number of entries in aggregated paths. In large-scale SDNs, due to large size, there are more instances of path aggregation and therefore more instances of reduction of installation of flow entries. PASR therefore also has lower flow rejection rates, thereby increasing the performance of the network.

Another approach to solving the issue of flow table overflow is to increase the space of flow tables by designing new architectures like SRAM (Static Random Access Memory) at the tail of TCAM [30]. But this is very difficult to achieve and implement due to the high cost of hardware. In order to improve the forwarding behaviour of the OpenFlow protocol so that the controller doesn't install flow entries right away after receiving packet-in messages from switches, we use another method to deploy the Lightweight Flow table Optimization scheme in SDN based on flow length Distribution in the Internet (LFOD) proposed by He et al. [31]. Using an elephant flow detection mechanism, mice flows and elephant flows are identified. Mice flows are flows that take up large amounts of space in the flow table but contribute to low traffic. Elephant flows, on the other hand, are less common but carry the majority of the traffic. A hybrid forwarding method is used by LFOD to handle both elephant and mice flows. The packet is handled in accordance with OpenFlow standard definitions when it comes to elephant flows. The flow entries of elephant flows are installed on switches and forwarded by OpenFlow switches. For mice flows, the SDN controller uses a packet-out message to transmit the packets straight to the destination. LFOD helps to reduce the high volumes of space occupied by mice flows and increases utilization by delegating those spaces for high-traffic elephant flows.

III. Conclusion

Software-Defined Networking centralizes network control and makes networks more flexible and diverse. However, the traditional OpenFlow scheme of SDNs is very inefficient as it leads to redundancies in flow entries and degrades network performance by causing an increase in delay and a decrease in throughput. On the bright side, many alternative solutions have been proposed to increase the efficiency of SDNs. This paper enlists and discusses some of the methodologies used to improve network efficiency in SDNs. Selecting optimal paths through various methods to reduce delay and increase throughput is an important task. DROM technique [18] and Ant-Colony-based methods [3] are effective in doing the same. In addition to increasing throughput, cluster head selection and blockchain concepts employed in Block-based-SDNs [16] and Ant Colony Genetic Fusion Routing algorithm [17] ensure proper resource utilization. When scaling up SDNs, issues like load balancing and flow entry reduction become much more visible. Due to high traffic SDNs tend to become inefficient without proper techniques to deal with huge volumes of network resources. Ant Colony Optimization [3] method and Genetic Algorithm [8] are bio-inspired algorithms [6], [7] that help to reduce Round Trip Time and Packet Loss Rate. Likewise, through aggregation of path segments of contemporaneous flows and by employing a hybrid forwarding technique, systems like PASR [27] and LFOD [31] facilitate the reduction of number of flow entries. Implementing MOPSO method [23] improves the link weight for the cost of the path and load balance. The RLMR [24] method, which uses reinforcement learning techniques, can also aid in optimizing link utilization. The discussions in this paper aid engineers in determining the methodology to be employed while optimizing Software-Defined Networks. Comparison between these methodologies enables the engineer to decide which methodology is feasible in a particular situation. Overall, we observe that using SDN networks along with the optimization techniques mentioned above can truly centralize control of networks without issues in managing traffic of huge sizes, thereby enabling engineers to optimize networks more effectively and efficiently.

DECLARATION

The authors of this paper declare that they have cited paper references to the corresponding authors and have avoided all forms of plagiarism to the best of their abilities.

REFERENCES

[1] Xue, H.; Kim, K.T.; Youn, H.Y. Dynamic Load Balancing of Software-Defined Networking Based on Genetic-Ant Colony Optimization. Sensors 2019, 19, 311. https://doi.org/10.3390/s19020311

[2] C. Yu, J. Lan, Z. Guo and Y. Hu, "DROM: Optimizing the Routing in Software-Defined Networks With Deep Reinforcement Learning," in IEEE Access, vol. 6, pp. 64533-64539, 2018, doi: T10.1109/ACCESS.2018.2877686.

[3] T. Shreya, M. M. Mulla, S. Shinde and D. G. Narayan, "Ant Colony Optimization-based Dynamic Routing in Software Defined Networks," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-7, doi: 10.1109/ICCCNT49239.2020.9225287.

[4] Kabiri, Z., Barekatain, B. & Avokh, A. GOP-SDN: an enhanced load balancing method based on genetic and optimized particle swarm optimization algorithm in distributed SDNs. *Wireless Netw* **28**, 2533–2552 (2022).

[5] M. R. Belgaum, S. Musa, M. M. Alam and M. M. Su'ud, "A Systematic Review of Load Balancing Techniques in Software-Defined Networking," in *IEEE Access*, vol. 8, pp. 98612-98636, 2020, doi: 10.1109/ACCESS.2020.2995849.

[6] Almufti, S., R. Marqas, and V. Ashqi. "Taxonomy of bio-inspired optimization algorithms." *Journal Of Advanced Computer Science & Technology* 8.2 (2019): 23.

[7] Johnvictor, Anita Christaline, et al. "Critical review of bio‐inspired optimization techniques." *Wiley Interdisciplinary Reviews: Computational Statistics* 14.1 (2022): e1528.

[8] Mirjalili, Seyedali, and Seyedali Mirjalili. "Genetic algorithm." *Evolutionary Algorithms and Neural Networks: Theory and Applications* (2019): 43-55.

[9] Lin, W.C.; Zhang, L.C. The Load Balancing Research of SDN based on Ant Colony Algorithm with Job Classification. In Proceedings of the 2nd Workshop on Advanced Research and Technology in IndustryApplications, Dalian, China, 14–15 May 2016.

[10] iPerf—The TCP, UDP and SCTP Network Bandwidth Measurement Tool. Available online: https://iperf.fr

[11] Gad, Ahmed G. "Particle swarm optimization algorithm and its applications: a systematic review." *Archives of computational methods in engineering* 29.5 (2022): 2531-2561

[12] Liao, Lingxia and Victor C. M. Leung. "Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs." *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (2017): 1-6.

[13] Zhang, Shaojun, et al. "Online load balancing for distributed control plane in software-defined data center network." *IEEE access* 6 (2018): 18184-18191.

[14] W. Yong, T. Xiaoling, H. Qian, and K. Yuwen, ''A dynamic load balancing method of cloud-center based on SDN,'' China Commun., vol. 13, no. 2, pp. 130–137, Feb. 2016

[15] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, ''QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach,'' in Proc. IEEE Int. Conf. Services Comput., Jun./Jul. 2016, pp. 25–33

[16] A. Rahman et al., "SmartBlock-SDN: An Optimized Blockchain-SDN Framework for Resource Management in IoT," in IEEE Access, vol. 9, pp. 28361-28376, 2021, doi: 10.1109/ACCESS.2021.3058244.

[17] Zhao, Kaixin, Yong Wei, and Yang Zhang. "An ant colony genetic fusion routing algorithm based on soft define network." IET Networks (2022).

[18] Hou, X.; Wu, M.; Zhao, M. An Optimization Routing Algorithm Based on Segment Routing in Software-Defined Networks. Sensors 2019, 19, 49. https://doi.org/10.3390/s19010049

[19] E. Akin and T. Korkmaz, "Comparison of Routing Algorithms With Static and Dynamic Link Cost in Software Defined Networking (SDN)," in IEEE Access, vol. 7, pp. 148629-148644, 2019, doi: 10.1109/ACCESS.2019.2946707.

[20] D. Todorov, H. Valchanov and V. Aleksieva, "Shortest path routing algorithm with dynamic composite weights in SDN networks," 2021 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2021, pp. 193-197, doi: 10.1109/ICAI52893.2021.9639512.

[21] Wang, Z.; Crowcroft, J. Quality-of-service routing for supporting multimedia applications. IEEE J. Sel. Areas Commun. 1996, 14, 1228–1234

[22] Apostolopoulos, G.; Kama, S.; Williams, D.; Guerin, R.; Orda, A.; Przygienda, T. QoS routing mechanisms and OSPF extensions. In Proceedings of the IEEE Global Telecommunications Conference, Phoenix, AZ, USA, 3–8 November 1997.

[23] Fei Han, Wen-Tao Chen, Qing-Hua Ling, Henry Han, "Multi-objective particle swarm optimization with adaptive strategies for feature selection," Swarm and Evolutionary Computation, Volume 62, 2021, 100847, ISSN 2210-6502.

[24] Chao Chen, Feifan Xue, Zhengyong Lu, Zhongyun Tang, Chuanhuang Li, "RLMR: Reinforcement Learning Based Multipath Routing for SDN," Wireless Communications and Mobile Computing, vol. 2022, Article ID 5124960, 12 pages, 2022.

[25] G. Kim, Y. Kim and H. Lim, "Deep Reinforcement Learning-Based Routing on Software-Defined Networks," in *IEEE Access*, vol. 10, pp. 18121-18133, 2022, doi: 10.1109/ACCESS.2022.3151081.

[26] EL Hocine Bouzidi, Abdelkader Outtagarts, Rami Langar, Raouf Boutaba, "Deep Q-Network and Traffic Prediction based Routing Optimization in Software Defined Networks," Journal of Network and Computer Applications, Volume 192, 2021, 103181, ISSN 1084-8045.

[27] Z. Li and Y. Hu, "PASR: An Efficient Flow Forwarding Scheme Based on Segment Routing in Software-Defined Networking," in IEEE Access, vol. 8, pp. 10907-10914, 2020, doi: 10.1109/ACCESS.2020.2964800.

[28] X. Dong, Z. Guo, X. Zhou, H. Qi and K. Li, "AJSR: an Efficient Multiple Jumps Forwarding Scheme in Software-Defined WAN," in IEEE Access, vol. 5, pp. 3139-3148, 2017.

[29] Z. Guo, Y. Xu, M. Cello, J. Zhang, Z. Wang, M. Liu, and H. J. Chao, "JumpFlow: Reducing flow table usage in software-defined networks," Computer Networks, vol. 92, pp. 300–315, 2015.

[30] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense," Security and Communication Networks, vol. 2018, p. 4760632, 2018.

[31] H. He, Z. Peng, X. Zhou and J. Wang, "LFOD: A Lightweight Flow Table Optimization Scheme in SDN Based on Flow Length Distribution in the Internet," 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), Takamatsu, Japan, 2022, pp. 1-6, doi: 10.23919/APNOMS56106.2022.9919927.