Controller Placement in Software Defined Networks using Honey Badger Algorithm

Mahnaz Khojand[1], Kambiz Majidzadeh[2] (Corresponding Author), Mohammad Masdari[3], Yousef Farhang[4]
[1] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
[2] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
[3]Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
[4]Department of Computer Engineering, Khoy Branch, Islamic Azad University, Khoy, Iran
Email: m.khojand@gmail.com [1], Kambiz.majidzadeh@iau.ac.ir [2], masdari2012@gmail.com [3], yfarhang@iaukhoy.ac.ir [4]
Receive Date: 01 Jan 2024 Accept Date:12 April 2024

Abstract

Software-defined networking (SDN) is a network structure where the control and data planes are separated. In traditional SDN, a single controller was in charge of control management, but this architecture had several constraints. To address these constraints, it is advisable to incorporate multiple controllers in the network. Selecting the number of controllers and connecting switches to them is known as the controller placement problem (CPP). CPP is a key hurdle in enhancing SDNs. In this paper a metaheuristic algorithm called Honey Badger Algorithm (HBA), is used to determine the optimal alignment between switches and controllers. HBA is modified using genetic operators (GHBA). The assessments are conducted with a diverse range of controllers on four prominent software-defined networks sourced from the Internet Topology Zoo and are compared to numerous algorithms in this field. It is noted that GHBA outperforms other competing algorithms in terms of end-to-end delay and energy consumption.

Keywords: Software Defined Network, Controller Placement, Honey Badger Algorithm, Heuristic algorithms, Genetic operators.

1. Introduction

Modern developments in information and communication technologies, including cloud computing, Internet of Things (IoT), video conferencing, online gaming, increased traffic volume, and social networking, highlight the inadequacy of traditional networks to address evolving traffic demands and diverse application requirements. Consequently, network management encounters substantial challenges, and upgrading, managing, and providing new services without adding new hardware are essential in next-generation networks[1, 2]. Software Defined Networking (SDN) is a new generation of network that has the capability to transform traditional network infrastructures into more compatible, agile, flexible, and controllable network topologies by separating the control plane from the data plane[1, 3, 4]. In traditional SDN networks, only one controller was responsible for control management, but this architecture had several limitations. One limitation of having only one controller in the network is that it can become a single point of failure. If the controller fails or encounters issues, it can compromise the performance and management of the entire network. Additionally, a single controller may struggle to meet the increasing demands of a

large-scale network, leading to performance bottlenecks and scalability problems. To address these limitations, it is recommended to have multiple controllers in the network. Having multiple controllers enhances fault tolerance and scalability within the network[5, 6]. Typically, multiple controllers are physically distributed across the network to improve various performance metrics such as switch latency, fault tolerance, and controller response time. In a multi-controller architecture, each switch is assigned to only one controller, and each controller serves a specific set of switches. [7]. Having more controllers increases the efficiency of the network, but on the other hand, due to the high cost of controllers, it is not possible to use any number of controllers. Finding the optimal number of controllers to be placed in SDN and the position of these controllers is called the controller placement problem (CPP). The CPP takes the network topology as input and typically determines the number of controllers needed for the network and their locations and the switches assigned to each of them[8]. In static networks, the problem of controller placement can be easily solved at the beginning of the network. But since we have assumed a dynamic network in this work, the determination of this number and allocation must be dynamic processes and be done based on the dynamic changes of the network. Determining the switches controlled by any controller is challenging because it has an exponential number of possible solutions and belongs to the class of computational problems that are difficult to solve efficiently. The number of possible network configurations and solutions to the switch mapping problem increases exponentially with the size of the network so that a comprehensive search for the optimal solution becomes computationally impossible. Due to the computational complexity and lack of efficient algorithms for its optimal solution, CPP is classified as an NP-hard problem[9].

Meta-heuristic algorithms are high-level problem-solving techniques used to find approximate solutions to complex optimization problems. These algorithms are designed to explore the search space efficiently and effectively, even when the problem has many possible solutions or lacks a known mathematical formula. For this reason, we have used meta-heuristic methods to determine the mapping method. Meta-heuristic techniques have been widely used in recent years due to their efficiency in solving complex and large-scale problems[10]. Unlike traditional optimization algorithms that rely on explicit and problem-specific information, meta-heuristics are general-purpose and can be applied to a wide range of problems. Meta-heuristic algorithms draw inspiration from natural processes or phenomena like evolutionary biology, crowd intelligence, or physical phenomena[11]. Typically, these algorithms engage in iterative enhancements of a potential solution by traversing diverse areas within the search space. This study employs heuristics methods to steer the search process, enabling evasion from local optima and striving to locate high-quality solutions within acceptable timeframes. The primary objective of the proposed methodology is to enhance parameters related to delay and energy consumption in SDNs.

The findings indicate that the suggested algorithm surpasses other competing algorithms in efficiency, reducing both end-to-end delay and energy consumption. The

key outcomes of this paper are summarized as follows:

- Modifying HBA using genetic operators.
- Applying the GHBA to the CPP.
- Assessing the suggested controller placement algorithm on two actual software-defined networks.
- Contrasting the outcomes of the proposed algorithm with four state-of-the-art meta-heuristic-based algorithms.

The structure of the paper is as follows: Section 2 presents a review of related literature in this field, while section 3 describes the metrics utilized in this study. Section 4 elaborates on the proposed controller placement algorithm for addressing CPP. The evaluation results are outlined in section 5, and the concluding remarks are covered in section 6.

2. Related Works

This section briefly reviews some related works to how to select and place the controller in SDN. Since CPP is an NP method, a heuristic method or a meta-heuristic method is used in most of the works[11].

Dynamic placement of controllers was investigated for the first time in [12]. In this study, the location of the controllers was adjusted to be responsive to the changes occurring in the network traffic. In this work, the problem of dynamic placement of controllers for large samples was raised and for this purpose, two heuristic algorithms were proposed and the setup time and end-to-end delays were significantly reduced.

The authors in [13]proposed a new method based on GSO for the placement of SDN controllers, which used the combination of the 0-1 knapsack problem and Garter snake optimization algorithm to improve the placement of controllers in the network. This method compared to other optimization algorithms in terms of Quality, calculation time, utilization of network resources and reduction of point-to-point delays had better performance. However, this method did not provide good performance in the Colt network. The evaluations showed that the performance of the algorithm was efficient in small-scale (Aarnet) to large-scale (Cogent) networks, but in the Colt network, it was not able to achieve minimum delays close to optimal. In this study, the authors did not provide a clear explanation for the failure of the algorithm in the Colt network and the scalability of the algorithm, which may be a limitation of the proposed approach.

In [14], the authors found the number of controllers using game theory and then optimized the way of mapping switches to each controller by combining two metaheuristic algorithms of golden eagle and gray wolf. In this article, end-to-end delay factors, load imbalance, energy consumption were considered and improved. The drawback of this method was the consumption of more memory and processor.

In [6], Ateya et al proposed a meta-heuristic algorithm based on the Salp Swarm Optimization Algorithm (SSOA) that dynamically determines the optimal number of controllers as well as the optimal paths between switches and controllers. Their proposed method was able to improve network latency and reliability in SDN, but it had high computational complexity and did not guarantee accuracy. In the controller placement method in [15], the focus was on minimizing network propagation delay. The authors used the concept of network

segmentation and the hybrid feeding algorithm (MRFO) of the Salp Swarming Algorithm (SSA). However, the use of more CPU time and the need for additional storage space were cited as drawbacks of this study.

In [16], Gao et al. introduced a new algorithm for the controller placement problem in SDNs. The introduced algorithm considers controllers with capacity constraints, delay between controllers and delay between switches and controllers. In this plan, a meta-heuristic algorithm based on PSO algorithm was proposed to solve the problem and the global delay was defined. Experiments showed that although the proposed algorithm minimized the propagation delay, the static traffic load of the controller was ignored. In addition, in the present article, the discretization problem and approach were not obvious. Furthermore, their proposed method was presented for SDN networks with capacity. In [17], Tehamasbi and his colleagues proposed an optimization algorithm for the placement of controllers for synchronizing WSN networks to optimize network performance. In the proposed algorithm, the optimization process was performed using the Cuckoo Search algorithm, which is a metaheuristic optimization algorithm. The performance of the proposed algorithm was compared with the training and quantum bending algorithms. The results of the comparisons showed that the proposed algorithm performed better than the competing algorithms in terms of network resistance and delay reduction. But the algorithm was not considered in terms of scalability and faced problems in large networks. A scalable placement algorithm for SDN was proposed in [18],which uses poly-stable pairing to distribute switches equally among controllers and assigns switches considering load and delay. This algorithm also reduced the delay between switches and controllers by moving switches. The proposed algorithm was evaluated in three real ISP networks with medium and large scale to check its scalability and efficiency in WAN. The results showed that the proposed algorithm has a better performance compared to the existing algorithms in terms of load distribution and delay reduction and easily provided close to optimal solutions, but this algorithm consumes more memory and processor. The authors in [19] first formulated the controller placement problem as a multi-objective optimization problem. They include reliability, fault tolerance, latency performance, synchronization, and deployment cost. They used the Cuckoo optimization algorithm, the evaluation results showed that this algorithm is superior in performance and synchronization cost, and is significantly more cost-effective, which makes it applicable in large wireless sensor networks.

In [20], a dynamic controller placement method for optical transmission networks was presented that considered the diversity of optical controllers, resource constraints at edge host locations, and delay requirements. The proposed method was a virtual method that allows for greater flexibility and scalability in the network and also enables easy recovery from failures or disasters. This method helps the problem of controller placement by using machine learning algorithms by predicting the active controllers. This method reported the accuracy of the proposed method for different traffic levels and evaluated the performance criteria such as accuracy, ROC

area. Evaluations showed that the proposed method works better in placing controllers in optical transmission networks and can be expanded in other types of networks. In [21], a multi-objective method for placing controllers in SDN was proposed, which was able to reduce the communication delay of the switch to the controller for both link failure and link failure modes. This algorithm generated an initial acceptable solution using a greedy method with grid partitioning and then iteratively generated new solutions by variable local search. Whenever a new solution was generated, the algorithm decided whether to accept the new solution as an acceptable solution to the problem and performed an update operation on the set of Pareto optimal solutions. Meanwhile, to avoid falling into the local optimum, the algorithm used the chaos strategy.

In [9], the authors presented a new algorithm for controller placement in SDN networks. The presented algorithm provided a support technique against the failure of a link and minimized the communication delay by using flexibility. The presented algorithm used particle group optimization algorithm and firefly algorithm to achieve the mentioned goals. Also, delay between controllers, delay between switches and controllers and multi-path connection between switches and controllers were considered in the presented algorithm. The evaluation results showed that the proposed algorithm improves the survivability of the network path and improves the network performance effectively. In addition, Jalili and colleagues presented an innovative algorithm for controller placement using NSGA-II for large software defined networks[22]. In the presented algorithm, the controllers considered maximum and average delay and load balancing as objective functions. Several networks of Zoo Internet topology were evaluated using the proposed algorithm, and the obtained results showed the superiority of the proposed algorithm over other similar algorithms. Multi-objective controller placement algorithm using NSGA-II optimized the delay and load balance of controllers. Although the proposed algorithm was applied to several SDN networks, it was compared with PSA and PSO algorithms, which were not able to accurately express the possible superiority of the proposed algorithm.

In [23], Singh et al. developed an innovative optimization algorithm called Varna-based optimization (VBO) and used it to solve the controller placement problem in SDN. The main goal of the proposed algorithm was to reduce the average network delay. VBO does not assume the same formula for all particles in the population. Also, it was not necessary that particles always remain in a particular class in a generation. VBO could be improved by dividing particles into more than two classes, each class having a specific task. This method had a more optimal result than the results based on clustering and the results based on optimization for the controller placement problem with capacity. At the same time, its convergence rate was better than other algorithms. However, in this method, the discretization method and operators were not transparent and the complexity was not discussed. The results show that optimization-based methods provide better results than clustering-based methods.

3. Problem Formulation

Most of the proposed solutions for CPP focus on determining the optimal number and placement of controllers, as well as devising strategies for assigning switches to controllers[24]. Considering end-to-end delay, this work is an attempt to reduce the average energy consumption.

The SDN can be modeled by an undirected graph $G = (V, E)$ where $V = \{i, \dots n\}$ is the set of nodes (routers) and E is the set of edges. The edge uv corresponds to the bidirectional link between nodes u and v and its weight. Some usual metrics in SDN are defined as follows.

4. End-to-End Delay

Let $S = \{s_1, s_2, \dots, s_m\}$ denotes the switches in the network and $C = \{c_1, c_2, \dots, c_n\}$ denotes the controllers. The end-to-end delay in this method is calculated by Equation (1). It is the sum of the average inter-controller propagation latency denoted by AvgICL and the average switch-to-controller propagation Latency denoted by AvgSCL.

$$D_{e2e} = AvgICL + AvgSCL \qquad (1)$$

5. Energy Consumption

This article examines the energy usage of controllers, switches, and communication links to determine the overall energy consumption of the network. The calculation methodology is outlined as follows:

$$E = \sum_{i=1}^{n} \begin{pmatrix} e_{li} \times E_{swi} + e_{2i} \times E_{ci} \\ +e_{3i} \times L \\ +E_{link} \end{pmatrix} \qquad (2)$$

In this context, E_{swi} signifies the energy consumption of the i_{th} switch, while E_{ci} denotes the energy usage of the i_{th} controller. Additionally, L and E_{link} stand for the latency of the network and the aggregate energy consumption of all operational links within the network. Three normalized weight parameters, namely e_{li}, e_{2i} , and e_{3i} are obtained through simulation runs conducted over a specific duration[25].

6. Proposed Algorithm

In this section, the algorithm that is used in this paper is described.

6.1. Honey Badger Algorithm

The Honey Badger Algorithm (HBA) is inspired by the foraging behavior of the honey badger, which involves two main approaches for locating food sources. In the first approach, known as the digging mode, the honey badger uses its sense of smell to estimate the location of prey. Upon reaching the target area, it then moves strategically to choose the optimal spot for digging and capturing the prey. In the second approach, called the honey mode, the honey badger follows the guidance of the honeyguide bird to directly locate a beehive.

The HBA is structured into two distinct phases: the "digging phase" and the "honey phase", each serving specific functions outlined in further detail as follows:

6.1.1. Algorithmic steps

In theory, the HBA algorithm incorporates both exploration and exploitation stages, qualifying it as a global optimization technique. The mathematical formulation of the proposed HBA algorithm is elaborated as follows.

Step 1: Initialization phase begins by setting the number of honey badgers (population size N) and determining their initial positions using Equation (3). POC is Pupulation of candidates. i_{th} position of

honey badger x_i is indicated by i_{th} pos x_i. r1 is a random number between 0 and 1

$$\text{POC}==\begin{bmatrix} x_1 & \cdots & x_d \\ \vdots & \ddots & \vdots \\ x_{n_1} & \cdots & x_{n_d} \end{bmatrix}$$

$$i_{th} \text{ pos } x_i = [x_i^1, x_i^2, x_i^3, \ldots, x_i^d] \quad (3)$$

$$x_i = lb_i + r_1 \times (ub_i - lb_i),$$

Where x_i is i_{th} honey badger position referring to a candidate solution in a population of N, while lb_i and ub_i denote the lower and upper bounds of the search domain, respectively.

Step 2: Calculating the intensity metric (I), as a measure of the prey's concentration strength and its proximity to the i_{th} honey badger, is crucial. The scent intensity of the prey, represented as Ii, plays a significant role in determining the speed of movement. Specifically, higher scent intensity corresponds to increased pace, while lower intensity results in slower motion. This correlation is precisely expressed through Equation (4) in a mathematical formulation.

$$I_i = r_2 \times \frac{4}{4\pi d^2}, \quad (4)$$

r2 is a random number between 0 and 1

$$S = (x_i - x_{i+1})^2$$

$$d_i = x_{prey} - x_i$$

In this context, S denotes the source strength or concentration level, which signifies the prey's position. Furthermore, d_i represents the distance between the prey and the i_{th} honey badger.

Step 3: Adjust the density factor, represented by α, to introduce variability in randomization over time, aiding in transitioning smoothly from exploration to exploitation. Update the decreasing factor α such that it decreases with each iteration, gradually reducing randomization as time progresses. Implement this modification effectively using Equation (5).

$$\alpha = C \times exp\left(\frac{-t}{t_{max}}\right), \quad (5)$$

$t_{max} = \text{maximum of iterations}$

where C is a constant ≥ 1 (default = 2).

Step 4: In order to avoid getting stuck in local optima during the algorithm, a flag called F is used to adjust the search direction. This modification allows agents to explore the search space more extensively, enhancing the likelihood of discovering favorable opportunities and preventing confinement to suboptimal areas.

Step 5:In the Honey Badger Algorithm (HBA), the adjustment of agent positions is carried out through a procedure known as thex_{new} position update. This process is divided into two main phases, namely the "digging phase" and the "honey phase" as previously outlined.

6.2. Genetic HBA

During the digging phase, a honey badger moves in a manner resembling a Cardioid shape. This Cardioid motion can be calculated using Equation (6):

$$\begin{aligned} &x\_new \quad (6)\\ &= x\_prey + F \times \beta \times I \times x\_prey \\ &+ F \times r\_3 \times \alpha \times d\_i \\ &\times |\cos\ 〚(2\pi r\_4) \times [1 \\ &- \cos\ 〚(2\pi r\_5)〛\ ]〛\ | \end{aligned}$$

Here, x_{prey} represents the location of the prey, which corresponds to the best position discovered thus far – essentially the global best position. β (with a default value of 6) is the honey badger's capability to find food. di denotes the distance between the prey and the i_{th} honey badger, while r_3, r_4, and r_5 are three distinct random numbers ranging from 0 to 1. The flag F adjusts the search direction

and is determined using Equation (7). r6 is a random number between 0 and 1.

$$F = \begin{cases} 1 & if\ r_6 \le 0.5 \\ -1 & else \end{cases} \quad (7)$$

During the digging phase, the honey badger places significant emphasis on the scent intensity I of prey x_{prey}. The badger's position relative to the prey di and the dynamically changing search influence factor (α) are crucial factors during the digging phase. Furthermore, during the digging phase, the badger may experience disturbances labeled as F, potentially aiding in the discovery of a more favorable prey location. The mathematical representation of a honey badger tracking a honey guide bird to find a beehive in the Honey phase is captured by Equation (8). r_7 is a random number between 0 and 1.

$$x_{new} = x_{prey} + F \times r_7 \times \alpha \times d_i \quad (8)$$

In Equation (8), x_{new} represents the updated position of the honey badger, while xprey indicates the location of the prey. The equation demonstrates that the honey badger's search strategy near the currently known prey location xprey is determined by the distance measurement d_i and is affected by the time-varied search behavior represented by α. Additionally, the presence of a disturbance F may influence the honey badger's search process at this point.

In this study, genetic operators are employed in the update phase of the algorithm to facilitate both exploration and exploitation. As previously stated, during the digging phase, it is crucial to uncover new positions. Therefore, genetic operators, such as the mutation operator, are utilized to aid in the exploration of new positions. The random value $r8$, ranging between 0 and 1, plays a role in deciding whether to apply the original Honey Badger Algorithm formula or the genetic operators.

$$x_{new} = \begin{cases} x_{prey} + A + B \times C \\ Mutation(x_{prey}) \end{cases}$$

$$A = F \times \beta \times I \times x_{prey} \quad (8)$$

$$B = F \times r_3 \times \alpha \times d_i$$

$$C = |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]|$$

In the honey phase, a greater emphasis is placed on exploitation. Consequently, in certain scenarios, a crossover operation is performed on x_{prey} and x_i, as indicated by Equation (9). The random valuer_8, falling within the range of 0 to 1, dictates whether to employ the original Honey Badger Algorithm formula or genetic operators.

$$x_{new} = \begin{cases} x_{prey} + F \times r_7 \times \alpha \times d_i \\ crossover(x_i, x_{prey}) \end{cases} \quad (9)$$

By simultaneously utilizing genetic operators and HBA formulas for updating, the exploration and exploitation capabilities of HPA are enhanced, leveraging the advantages of both approaches

7. Performance Evaluation

This study aims to tackle the challenge presented by the dynamic characteristics of networks, which result in variations in the ideal quantity and positions of active controllers, along with their interactions with switches[6]. The primary objective is to reevaluate the allocation of switches to controllers to enhance network efficiency amidst these fluctuations. The article endeavors to address the CPP utilizing the GHBA algorithm.

This section commences by outlining the key elements of the experimental outcomes that evaluate the efficiency of the proposed algorithm. The experiments employ network topologies sourced from the Internet Topology Zoo[26]. The algorithms are

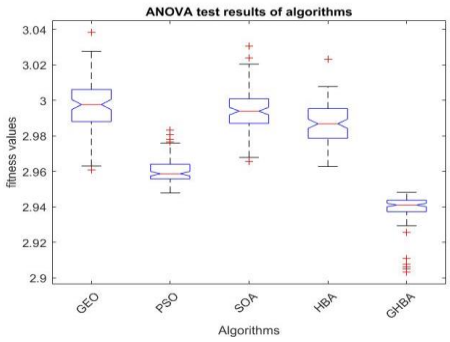
constrained to a maximum of 50 iterations. Within this framework, up to 12 controllers are at disposal, allowing for activation or deactivation based on network traffic levels. Diverse network setups are examined using 6, 8, 10, and 12 controllers. Each network undergoes testing with the transmission of 200 packets, concluding upon reaching the maximum iteration limit. Notably, all experiments are conducted within the same environment defined by the specifications in Table 1. The algorithms are coded in MATLAB, and multiple iterations are executed for each algorithm.

Table 1:The characteristics of the test environment

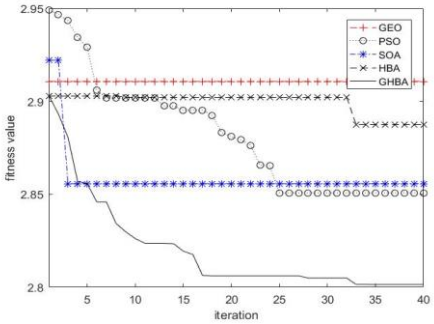
Name	Value
CPU	Core i5
RAM	8GB
HARD driver	500GB
Operating Systems	Windows 10
Language	MATLAB R2016b

Initially, the GHBA method is pitted against four other methodologies in a real-world network scenario to gauge its effectiveness. The comparative analysis involves benchmarking the proposed algorithm against several state-of-the-art metaheuristic algorithms, including GEO[27], PSO[28], SOA[29] and HBA. The efficiency of the GHBA algorithm is scrutinized using Anova diagrams and convergence rate diagrams. Anova diagrams assess algorithm efficiency by scrutinizing variance among random states, helping in evaluating performance across diverse states. On the other hand, convergence diagrams illustrate the speed at which each algorithm converges towards a solution, offering insights into their exploration and exploitation balancing capabilities.

Figure1 shows the Anova test and the convergence rate for the proposed algorithm in the Bics network. The Anova diagrams reveal that the GHBA outperforms its competitors by producing more stable results with fewer random states. GHBA exhibits notable stability, as indicated by its lower standard deviation in comparison to other algorithms. Additionally, the convergence curves highlight GHBA's superior rate of convergence, demonstrating its ability to quickly attain optimal solutions surpassing those of other algorithms. Specifically, in the Bics network featuring 46 switches and 85 edges, both the Anova chart and convergence rate diagrams consistently show the effectiveness of GHBA relative to its counterparts. This visual data effectively showcases GHBA's dominance in the context of the Bics network scenario.



(a)



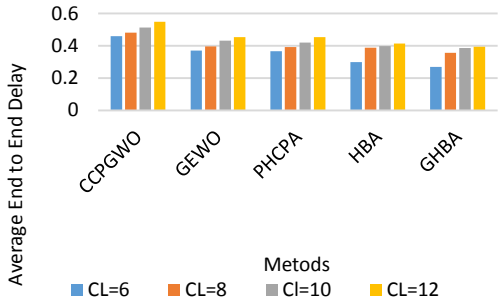
(b)

Fig. 1. (a) the Anova chart and (b) the convergence rate for the proposed algorithm in the Bics network.

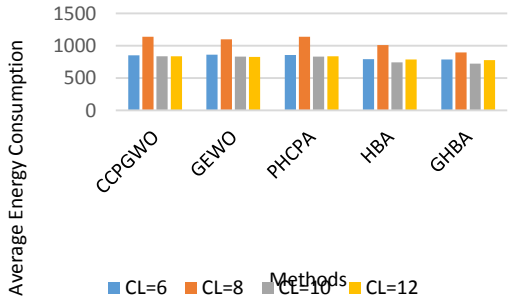
In the subsequent phase, the GHBA algorithm is deployed to address the CPP. Initially, random traffic is introduced into the network to establish connections between switches based on packet exchange volumes. The optimization process focuses on assigning switches to controllers to ensure that switches with the highest number of connections are managed by a controller within the same domain. This strategy aims to minimize connectivity between switches in different domains controlled by separate controllers, ultimately streamlining data transfer and rule enforcement processes between switches. By avoiding unnecessary data transfers and ensuring efficient communication within the network, this configuration enhances overall network efficiency. The performance of GHBA in solving the CPP is compared against recently developed algorithms including CCPGWO[30], GEWO[14], PHCPA[14] and HOA in terms of average energy consumption and end-to-end delay.

This section presents the results obtained for varying numbers of controllers on prominent software-defined networks, Bics, and Colt sourced from the Internet Topology Zoo. Due to the optimized mapping of switches to controllers, the need for continuous loading of additional information into each controller is eliminated, resulting in more efficient end-to-end delay compared to other algorithms. Figure 2 illustrates the comparison of GHBA with CCPGWO, GEWO, PHCPA, and HOA algorithms employing different numbers of controllers in the Bics network. The outcomes demonstrate that the proposed algorithm excels in terms of energy consumption and end-to-end delay efficiency when contrasted with the other algorithms.

Figure 2 depicts the outcomes of comparing the GHBA with the CCPGWO, GEWO, PHCPA, and HOA algorithms employing varying numbers of controllers in the Bics network. The results clearly indicate that the proposed algorithm outperforms the others in terms of energy consumption and end-to-end delay.



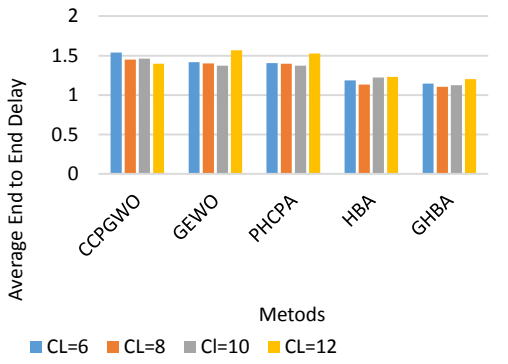
(a)



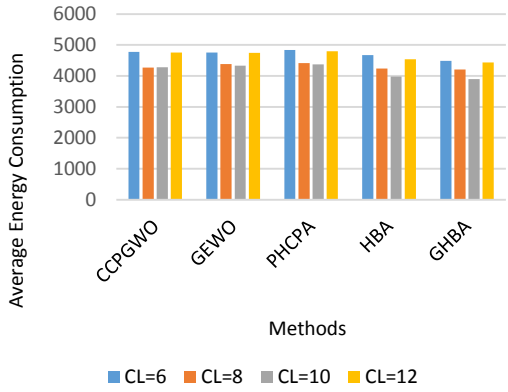
(b)

Fig.2. Bar graph illustrating the results achieved by the algorithms on the Bics network. (a): Average End-to-end delay (b) Average energy consumption.

Fig.3. illustrates a comparative analysis of GHBA against CCPGWO, GEWO, PHCPA, and HOA concerning energy consumption and end-to-end delay on the Colt network, which includes 149 switches and 191 links. The evaluation takes into account different numbers of controllers (6, 8, 10, and 12). The results indicate that the GHBA algorithm surpasses the other algorithms in terms of both energy consumption and end-to-end delay.



(a)



(b)

Fig.3. Bar graph illustrating the results achieved by the algorithms on the Colt network. (a): Average End-to-end delay (b) Average energy consumption.

8. Conclusion

This study aims to address the controller placement problem (CCP) in software-defined networks (SDN). Initially, the Honey Badger Algorithm is enhanced with genetic operators (GHBA). By incorporating genetic operators, the algorithm is better equipped to avoid local optima and improve both exploration and exploitation. The GHBA is then utilized in solving the controller placement problem (CPP) with the goal of reducing end-to-end delay and energy consumption. The performance of the proposed GHBA algorithm is evaluated on two real-world software-defined networks from topology zoo dataset. It is compared against four metaheuristic algorithms, each utilizing a different number of controllers. The results indicate that the GHBA algorithm surpasses its competitors by demonstrating superior efficiency with enhanced convergence rates, exploration, exploitation capabilities, and notable reductions in energy consumption and end-to-end delay.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks,* vol. 54, no. 15, pp. 2787-2805, 2010.

[2] M. A. Bagha, K. Majidzadeh, M. Masdari, and Y. Farhang, "ELA-RCP: An energy-efficient and load balanced algorithm for reliable controller placement in software-defined networks," *Journal of Network and Computer Applications,* p. 103855, 2024.

[3] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Computer Networks,* vol. 112, pp. 279-293, 2017.

[4] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Computer Networks,* vol. 81, pp. 79-95, 2015.

[5] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: A survey," *IEEE network,* vol. 31, no. 5, pp. 21-27, 2017.

[6] A. A. Ateya *et al.*, "Chaotic salp swarm algorithm for SDN multi-controller networks," *Engineering Science and Technology, an International Journal,* vol. 22, no. 4, pp. 1001-1012, 2019.

[7] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 19-24.

[8] V. Ahmadi and M. Khorramizadeh, "An adaptive heuristic for multi-objective controller placement in software-defined networks," *Computers & Electrical Engineering,* vol. 66, pp. 204-228, 2018.

[9] K. S. Sahoo, D. Puthal, M. S. Obaidat, A. Sarkar, S. K. Mishra, and B. Sahoo, "On the placement of controllers in software-defined-WAN using meta-heuristic approach," *Journal of Systems and Software,* vol. 145, pp. 180-194, 2018.

[10] S. Lange *et al.*, "Specialized heuristics for the controller placement problem in large scale

SDN networks," in *2015 27th International Teletraffic Congress*, 2015: IEEE, pp. 210-218.
[11] J. M. Hansen, S. Raut, and S. Swami, "Retail shelf allocation: A comparative analysis of heuristic and meta-heuristic approaches," *Journal of Retailing,* vol. 86, no. 1, pp. 94-105, 2010.
[12] M. F. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, 2013: IEEE, pp. 18-25.
[13] S. Torkamani-Azar and M. Jahanshahi, "A new GSO based method for SDN controller placement," *Computer Communications,* vol. 163, pp. 91-108, 2020.
[14] M. Khojand, K. Majidzadeh, M. Masdari, and Y. Farhang, "Controller placement in SDN using game theory and a discrete hybrid metaheuristic algorithm," *The Journal of Supercomputing,* pp. 1-49, 2023.
[15] N. Firouz, M. Masdari, A. B. Sangar, and K. Majidzadeh, "A novel controller placement algorithm based on network portioning concept and a hybrid discrete optimization algorithm for multi-controller software-defined networks," *Cluster Computing,* vol. 24, pp. 2511-2544, 2021.
[16] C. Gao, H. Wang, F. Zhu, L. Zhai, and S. Yi, "A particle swarm optimization algorithm for controller placement problem in software defined network," in *Algorithms and Architectures for Parallel Processing: 15th International Conference, ICA3PP 2015, Zhangjiajie, China, November 18-20, 2015, Proceedings, Part III 15*, 2015: Springer, pp. 44-54.
[17] S.Tahmasebi, M. Safi, S. Zolfi, M. R. Maghsoudi, H. R. Faragardi, and H. Fotouhi, "Cuckoo-PC: an evolutionary synchronization-aware placement of SDN controllers for optimizing the network performance in WSNs," *Sensors,* vol. 20, no. 11, p. 3231, 2020.
[18] B. R. Killi and S. V. Rao, "Poly-stable matching based scalable controller placement with balancing constraints in SDN," *Computer Communications,* vol. 154, pp. 82-91, 2020.
[19] S. Tahmasebi, N. Rasouli, A. H. Kashefi, E. Rezabeyk, and H. R. Faragardi, "SYNCOP: An evolutionary multi-objective placement of SDN controllers for optimizing cost and network performance in WSNs," *Computer Networks,* vol. 185, p. 107727, 2021.
[20] S. Rahman *et al.*, "Virtualized controller placement for multi-domain optical transport networks using machine learning," *Photonic Network Communications,* vol. 40, pp. 126-136, 2020.
[21] Y. Fan, L. Wang, and X. Yuan, "Controller placements for latency minimization of both primary and backup paths in SDNs," *Computer Communications,* vol. 163, pp. 35-50, 2020.
[22] A. Jalili, M. Keshtgari, and R. Akbari, "Optimal controller placement in large scale software defined networks based on modified NSGA-II," *Applied Intelligence,* vol. 48, pp. 2809-2823, 2018.
[23] A. K. Singh, S. Maurya, and S. Srivastava, "Varna-based optimization: a novel method for capacitated controller placement problem in SDN," *Frontiers of Computer Science,* vol. 14, pp. 1-26, 2020.
[24] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Computer Communication Review,* vol. 42, no. 4, pp. 473-478, 2012.
[25] M. Priyadarsini, S. Kumar, P. Bera, and M. A. Rahman, "An energy-efficient load distribution framework for SDN controllers," *Computing,* vol. 102, no. 9, pp. 2073-2098, 2020.
[26] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications,* vol. 29, no. 9, pp. 1765-1775, 2011.
[27] A. Mohammadi-Balani, M. D. Nayeri, A. Azar, and M. Taghizadeh-Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm," *Computers & Industrial Engineering,* vol. 152, p. 107050, 2021.
[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4: ieee, pp. 1942-1948.
[29] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowledge-based systems,* vol. 165, pp. 169-196, 2019.
[30] K. Kanodia, S. Mohanty, K. Kurroliya, and B. Sahoo, "CCPGWO: A meta-heuristic strategy for link failure aware placement of controller in SDN," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020: IEEE, pp. 859-863.