International Research Science and Development Center

International Research Science and Development Journal
www.IRSDJournal.com

International Research Science and Development Journal
Vol. 2, No. 2, 2021, pp. 25-45.
ISSN 2348-3008

A new meta-heuristic algorithm for placing controllers in software networks to reduce latency

Mohammad Erfan Mehrabian[1]

[1] Master of Computer Engineering, Islamic Azad University (IAU), Kerman Branch

Abstract

Software Defined Networks (SDNs) try to increase the intelligence of networks by transferring the data control section from the hardware switch and router to the virtual network software layers and using a defined software controller, capabilities such as scheduling. Provide scalability, flexibility, automation, intelligence and network software development by organizations. This new architecture has made the network very dynamic and many of the previous problems in the network have been solved. As the size of the network increases, using a controller across the network will cause problems such as increasing the average latency between the switches and the controller, as well as forming a bottleneck in the controller. Since this is an NP-Hard problem, methods based on meta-heuristic algorithms can be effective in solving it. In this paper, we have solved the problem of locating controllers in software-defined networks with the aim of reducing latency using genetic algorithms. Finally, the results of this study are compared with the SA algorithm.

Keywords: Controllers, Meta-Heuristic Algorithms, Software Networks, Delay Reduction, Controllers Placement.

International Research Science and Development Journal
Vol. 2, No. 2, 2021, pp. 25-45.

Introduction:

The evolution of mobile devices and peripherals, the virtualization of servers, and the advent of cloud computing services have led to a re-examination of common network architecture. The architecture of many traditional networks is a hierarchy formed by using groups of network switches in a tree structure. This architecture will be more tangible when it comes to client-server communications [1]. But such a static architecture is not enough for dynamic communication and the needs of companies in the field of data centers and media servers. Computer networks cover the whole world and no serious and new development has been formed in this industry for many years [2]. But companies and consumer organizations complain about not adding new features to their networks and like to do a lot of things automatically according to their needs, and also like to develop networks as software and Expand and not have to resort to expensive new hardware for many of their network activities and needs. The software-defined network architecture (SDN) and the Open Flow protocol make data and control levels separate, make the network smarter and more controllable, and separate the core network infrastructure from applications, and companies are able to There will be more programming, automation and network control [3].

Software-defined networks try to increase the intelligence of networks and by transferring the data control section from the switch and hardware router to the virtual software layers of the network and using a defined software controller, capabilities such as scheduling, scale Provide flexibility, flexibility, automation, intelligence and network software development by organizations [4]. SDN can be called the biggest evolution of four decades of computer networks. SDN was first introduced in 2005 and accelerated in 2010 and entered a new phase in 2011 with the formation of the DNF Foundation and the membership of more than eighty major companies in the network industry and the development of the Open Flow standard. The first SDN products entered the market in 2012 and more in 2013, and it was predicted that by 2017, these types of networks would gradually replace traditional TCP / IP-based networks.

State the problem:

Software defined networks have been introduced as an emerging phenomenon in network architecture and today, like the cloud, are a hot topic in the IT world. Basically, transformation in networks is formed by several important factors: a) providing new and

better services b) advancing technology c) increasing bandwidth and reducing costs, which create new architectural requirements for networks [5]. SDNs or Software defined networks try to increase the intelligence of networks and by transferring the data control part from the control of hardware switch and router to Software defined networks layers and using the software defined controller, capabilities such as scheduling Provide scalability, flexibility, automation, intelligence and network software development by organizations. Software defined networks are a new type of network that separates the control layer from the data transfer layer and the network is logically centralized. In these networks, the control layer manages the network optimally with a single view of the entire network [6].

The control layer does this by using controllers that are the masterminds of the network. The controller can act as a set of distributed controllers that provide a single view of the network. SDN consists of separating the data page from the control panel. The task of managing the control panel in an SDN network is the responsibility of the logical centralized controller [7]. Given the importance of controllers in SDN architecture and the diversity of architecture and implementation in the market and research areas, there is a need to evaluate and benchmark all of these choices with different performance indicators. A variety of controllers are implemented with processes of several thousand currents per second to several million currents per second, with multiple language methods, architectures [8], APIs and protocols. Software defined networks are a new architecture in computer networks in which network intelligence is logically concentrated in the software controller and the network hardware becomes a simple device for navigation that can be closed through an open interface [9].

The importance of research:

Software defined networks were first introduced in research on the OpenFloe protocol at Stanford University. The SDN network has great potential for reorienting the performance of computer networks, and Open Flow has been touted as a radical idea in particular [10]. While open flow has attracted a lot of attention in the industry today, the idea of programmable networks and segregation of control logic has long been considered. Given that the structure of many common networks today is hierarchical, the use of such a static architecture is not sufficient for dynamic communication and the needs of today's companies [11].

One of the major challenges facing Software defined networks is how to choose the right places to place and distribute controllers so that the latency between controllers and switches

can be extended across wide area networks [13]. In this regard, most of the methods presented to date have focused on reducing latency, but latency is one of the factors that play a role in network performance and reduce the overall cost between controllers and related switches. The proposed scheme significantly improves service quality parameters such as end-to-end latency, packet loss, and network life, and appears to be a viable alternative in the near future [14].

The concept of Software-defined networks:

SDN is a new network architecture in which data and control levels are separated. In these networks, by separating applications from the main network infrastructure and flexibility in the control layer, networks become smarter and more controllable [15]. In such architecture, the ability to control is removed from network devices, and thus these devices will become simple sending (packet) components. The control logic is passed to an external element called the SDN controller. Infrastructure devices are simply sent motors in which input packets are sent based on a set of rules generated by one (or more) controllers [16]. The controller's decision is based on the logic of predefined programs. In fact, instead of policies and protocols being implemented separately on a set of devices, it is done through a single central controller throughout the network, regardless of the hardware and the manufacturer of that network [17]. The foundation of software defined companies is based on the Open Flow contract. Data and control layer abstraction helps programmers operate on an abstract network layer instead of thousands of different physical devices through user interfaces [18]. To achieve this, the SDN architecture requires physical devices with different features compared to the classic switch. SDN switches are used to achieve this goal in network infrastructure. SDN switch is a switch that can be programmed through a central controller. This planning is done through a southern interface, which in most cases is an Open Flow contract [19].

SDN controller

SDN controllers are based on protocols such as Open Flow, which allow servers to command the switch where packets are sent [20]. The controller is the core of SDN networks, which is located between the network hardware and the application layer, and any communication between the network hardware and the application must be done through the controller. The

controller also uses protocols such as Open Flow to perform tasks such as: configuring network hardware and selecting the optimal route for traffic [21].

As we said, the controller is the main part of NOS and is located between network equipment (bottom layer) and applications (top layer). It was also stated that an SDN controller is responsible for handling any current in the network, and does so by creating current interference on each switch. Two modes can be considered to adjust the flow: active and inactive [22].

In the active mode, the flow rules are already placed in the flow tables. Therefore, the current settings are made before the first packet of a stream reaches the Open Flow switch [23]. The main advantage of actively adjusting the current is the slight adjustment delay and the reduction of the number of times the communication with the controller [24]. However, there is a possibility of overflow of switch flow tables. In contrast, we have a passive switch setting, in which the current law is set when there are no entries in the flow tables, and this setting is made when the first packet of a current reaches the Open Flow switch. Therefore, only the first packet creates a connection between the switch and the controller. These inputs will expire after a deadline and must be eliminated.

Given the above about SDN, the purpose of SDN is to transfer the intelligent portion of the network from the switching equipment depending on the centralized controller. Sending decisions are first made in the controller and then applied to the switches, so they can easily execute these decisions. This has many advantages, including a comprehensive control and overview of the entire network when it comes to automating network operations, better server server efficiency, and other benefits.

Open Flow Protocol

Open Flow is a communication protocol that allows network switches and routers to be configured. Common switches consist of two parts, control and transmission [25]. The control part, using routing protocols, determines the output port based on the destination address for the input packets and registers it in the routing table, and transmits the transmission part of these packets. The basis transmits the information obtained [26].

Open Flow is the most common communication protocol between the controller and the switch in the SDN network. Switches that support this protocol include one or more routing tables and a group table that find the output port for input packets and Transfers packets [27].

The Open Flow protocol defines an API for communication between the control and the switch. The controllers in the network use this protocol to connect to the switch and make changes to the routing table. Input packets on the switch are compared to the routing table, and if they comply with existing rules, routing begins, and if the packet does not comply with any rules, it is sent back to the controller to be decided [28].

An Open Flow switch consists of one or more flow groups and flow tables that search and direct the packet and have an Open Flow channel to the external controller. The switch communicates with the controller and manages the switch controller using the Open Flow protocol. Using this protocol, the switch can record, delete and update the flow entry in the flow tables. Each flow table contains a set of flow inputs, each of which contains a matching field, a counter, and a set of instructions for performing packet matching operations. The match starts from the first table and may be transferred to other tables [29].

Current inputs are routed to a port that is a physical or logical input defined by the switch is a port reserved based on a characteristic. Reserved ports may be a specific routing operation such as sending to a controller, flood action Asa or normal switch operation without considering the Open Flow methods of operations at the input of the flow may direct packets to the group, which requires more processing [30].

Formulate the problem:

Software defined networks, although they solve many of the problems of traditional networks, have their own challenges. One of the major challenges in this type of network is to place the least number of controllers within the network so that goals such as reducing latency can be achieved. The symbols and presuppositions used in this dissertation are as follows:

The communication network is represented by a graph without direction $G(V, E)$

Sets V and E represent the set of nodes and links in the network, respectively.

Each $i \in V$ node represents a switch. Each switch has the ability to accommodate only one controller. Each controller can be positioned on any of the network nodes. The set of placements where the controllers are located is called V_c.

$y_i = \{1,0\}$ is a binary variable in which a value of 1 means that node $i \in V$ hosts a controller and otherwise no.

$x_{ij} = \{1,0\}$ is a binary variable whose value 1 indicates that the $j \in V$ switch is controlled by the controller in node i. Otherwise it will have a value of 0. Obviously if $y_i = 1$ is $x_{i,i} = 1$.

It is assumed that each switch is controlled by a single controller. Suppose $Del_{i,j}$ represents the amount of delay calculated between nodes $i,j \in V$ so that node j has a controller. The purpose of locating controllers is to ensure with the least number of controllers that the delay between the switch-controller does not exceed the D threshold.

$$\min \quad \sum_{i \in V_C} y_i \qquad (1)$$

s.t.

$$x_{ij} \leq y_i ; \qquad i \in V, j \in V_C \qquad (2)$$

$$\sum_{i \in V} x_{ij} y_i = 1 ; \qquad j \in V_C \qquad (3)$$

$$Del_{i,j} \leq \mathfrak{D} ; \qquad i \in V, j \in V_C, \; x_{ij} = 1 \qquad (4)$$

Equation (1) indicates that the number of controllers placed in the network should be minimized. (2) Indicates that switches can only be assigned to a node in which the controller is located. (3) states that each switch in each node, such as j, must be connected exactly to a controller. (4) Indicates the delay limit.

Genetic algorithm:

Genetic algorithm is one of the most important heuristic algorithms that is used to optimize various functions. In this algorithm, past information is extracted due to the inheritance of the algorithm and is used in the search process. These algorithms have fundamental differences from conventional search and optimization methods, which Goldberg summarizes as follows:

1. The genetic algorithm works with a set of encoded answers, not with them.
2. The genetic algorithm starts searching in a population of answers and with a set of them, not with one answer.
3. The genetic algorithm uses fitting function information, not derivatives or other ancillary sciences.
4. The genetic algorithm uses the rules of probabilistic transmission, not the definite rules.
5. The genetic algorithm works with a set of encoded answers, not with them themselves.
6. The genetic algorithm starts searching in a population of answers and with a set of them, not with one answer.
7. Genetic algorithm uses fitting function information, not derivatives or other ancillary sciences.

8. The genetic algorithm uses the rules of probabilistic transmission, not the definite rules.

Because genetic algorithms are derived from both computer science and natural genetics, the terms used are a mixture of natural and artificial terms. The basic concepts that are critical to understanding the genetic algorithm are:

Chromosome: The basis of the genetic algorithm is to convert each set of answers into a coding. This code is called a chromosome. Chromosomes are also called individuals, strands, and structures. They can also be called genotypes.

Phenotype: Each chromosome corresponds to a set of answers to a problem. The corresponding set for each chromosome is called a phenotype.

Gene: The elements that make up a chromosome, which are usually numbers, are called genes. Genes have also been called composition, expression, and decoding.

Placement: The placement of a gene on a chromosome is called a placement.

Population: A set of chromosomes is called a population.

Generation: Each iteration of an algorithm is called a generation.

The structure of the proposed genetic algorithm is as follows:

Procedure: Genetic Algorithm

Step 1: Set t:=0

Step 2: Generate initial population, p(t).

Step 3: Evaluate p(t) to creat fitness values

Step 4: While (not termination cordination) do:

Step 5: Recombine p(t) to yield c(t), selecting from p(t) according to the fitness values.

Step 6: Evaluate c(t)

Step 7: Generate p(t+1) from p(t) and c(t)

Step 8: Set t:=t+1

Step 9: End.

Step 10: Stop

P (t) = parents of generation t

C (t) = infant of generation t

Primary population production:

1- First we create a random sequence of different operations and orders.

2-Then, based on the problem constraints, a machine is selected for each operation based on the set of machines that are capable of performing the said operation.

3-This process is done until the machines are assigned to all operations (the first matrix is completed).

4- Then, in the last step, based on the sequence of different operations, we randomly create a sequence of different operations (completion of the second matrix).

5- At the end, based on the first and second matrices, the start and end time of each operation is calculated and the objective function is determined.

Mutation operator and composition and stop condition of genetic algorithm:

The pairs that were considered as parents in the selection section exchange their genes with each other in this section and create new members. The intersection in the genetic algorithm causes the dispersal or genetic diversity of the population to disappear. This type of crossover operator ensures that the offspring produced are always regular (ie, it is never possible to produce chromosomes that do not correspond to any member of the response space).

Common methods are single-point, two-point, multi-point, and uniform displacement. The simplest mode of displacement is single-point displacement. In single-point translocation, first the parent chromosome pair (binary strand) is cut at a suitable point along the strand, and then parts of the cut point are swapped. This results in two new chromosomes, each of which inherits genes from the parent chromosomes.

Simulation Annealing Algorithm:

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a meta heuristic to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete (e.g., the traveling salesman problem). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to exact algorithms such as gradient descent, Branch and Bound. The name of the algorithm comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Both are attributes of the material that depend on their thermodynamic free energy. Heating and cooling the material affects both the temperature and

the thermodynamic free energy or Gibbs energy. Simulated annealing can be used for very hard computational optimization problems where exact algorithms fail; even though it usually achieves an approximate solution to the global minimum, it could be enough for many practical problems.

The problems solved by SA are currently formulated by an objective function of many variables, subject to several constraints. In practice, the constraint can be penalized as part of the objective function.

This notion of slow cooling implemented in the simulated annealing algorithm is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored. Accepting worse solutions allows for a more extensive search for the global optimal solution. In general, simulated annealing algorithms work as follows. The temperature progressively decreases from an initial positive value to zero. At each time step, the algorithm randomly selects a solution close to the current one, measures its quality, and moves to it according to the temperature-dependent probabilities of selecting better or worse solutions, which during the search respectively remain at 1 (or positive) and decrease towards zero.

The simulation can be performed either by a solution of kinetic equations for density functions or by using the stochastic sampling method. The method is an adaptation of the Metropolis–Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, published by N. Metropolis et al. in 1953.

Algorithm evaluation:

In this section, the parameters used in the proposed genetics, which include the initial population, mutation rate, composition rate, number of replications in each model run are adjusted. The Taguchi experimental design method was used to adjust the parameters of the proposed algorithms. For GA method, four factors of initial population number (npop), maximum number (max_it), mutation coefficient and crossover coefficient are considered. The criterion of the value of the objective function is also considered as the criterion of response. Also, for each factor, three levels are considered as follows.

Initial population: 40, 50, 60

Maximum number of repetitions: 50, 150, 200

Displacement coefficient: 0.7, 0.8, 0.9

Mutation coefficient: 0.3, 0.2, 0.1

In Taguchi method, the criterion (S / N) is used. This criterion shows the amount of changes that have occurred in the response variable. For each factor, the optimal surface value is less than the standard value (S / N).

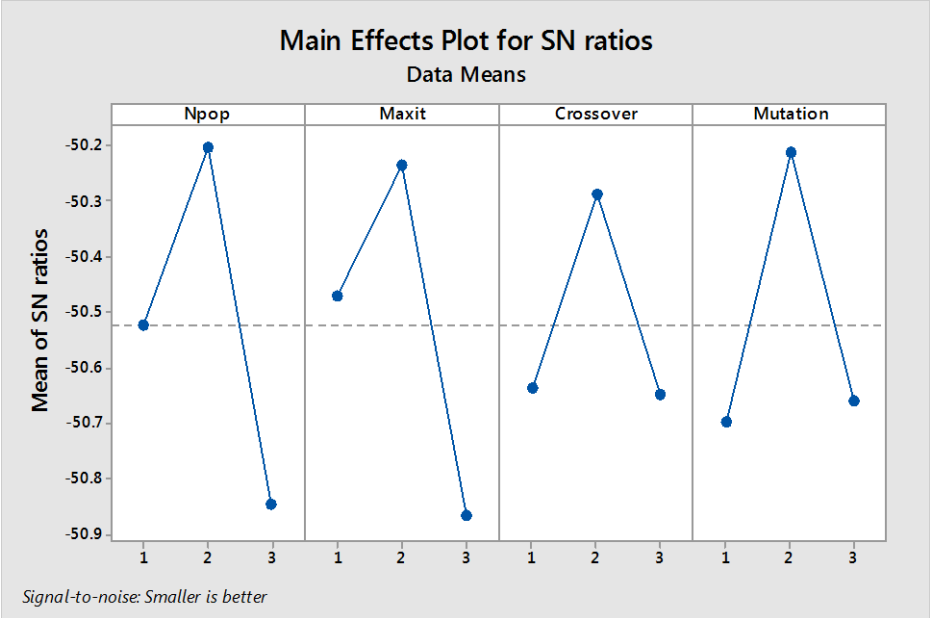

Figure 1: S/N rate for genetic algorithm coefficients

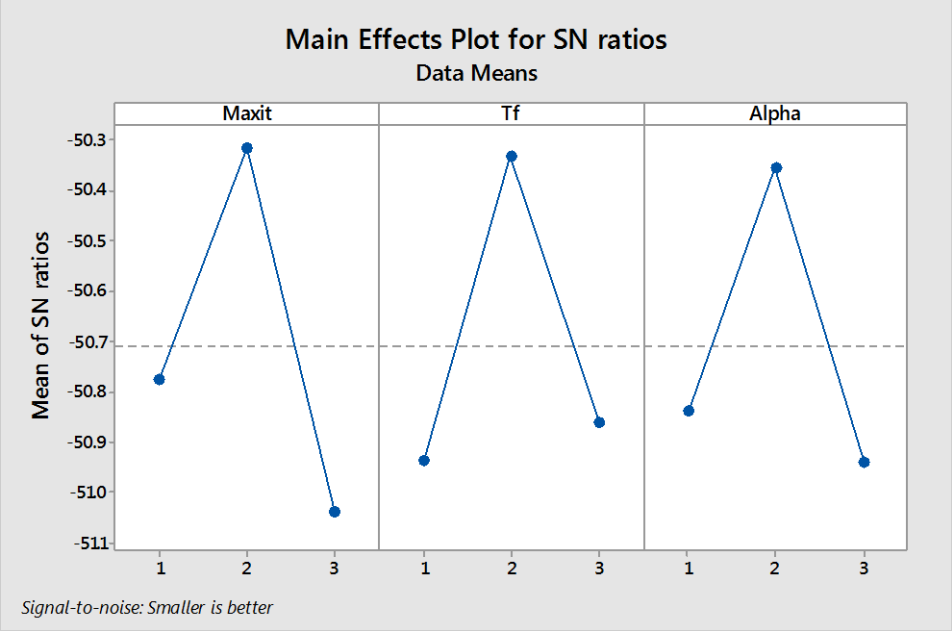

Figure 2: S/N rate for SA algorithm coefficients

Computational results of the proposed algorithm:

In this section, the exact method is used to validate the model. All the results of running programs in a notebook system with Intel ® Core ™ 2 Duo CPU and 2.0 GHz processor and

2.00 Gb RAM and Microsoft Windows 7 Ultimate operating system. Gomez software has been used to solve the problem. Due to the NP-Hard model, with increasing the size of the problem, the computational solution time increases dramatically. Therefore, in this research, the best answer found by Gomez software is reported in two hours or 7200 seconds.

In this section, the results of calculations of the solved examples in following Tables with the proposed algorithm are presented. This algorithm is programmed in MATLAB software version 2013b. The 15 problems designed in the table below are solved by the genetic algorithm, refrigeration simulation and the combined genetic algorithm and refrigeration simulation.

Table 1: Results of the proposed algorithms

Test Problem Number	GA		SA		GA-SA	
	Objective	GAP%	Objective	GAP%	Objective	GAP%
۱	359.2	0	359.2	0	359.2	0
۲	386.6	0	387.8	0.3 %	386.6	0
۳	485.5	0.2%	488.9	0.9%	484.5	0
۴	644.2	0.7%	647.4	1.2%	639.7	0
۵	643.6	1.8%	646.7	2.3%	637.3	0.8%
۶	771.3	2.5%	778.1	3.4%	760.8	1.1%
۷	1021.7	3.2%	1032.6	4.3%	1007.8	1.8%
۸	898.1	3.8%	910.2	5.2%	883.4	2.1%
۹	948.7	4.2%	967.9	6.3%	936.9	2.9%
۱۰	1212.0	5.3%	1232.7	7.1%	1190.1	3.4%
۱۱	1147.4	7.2%	1160.2	8.4%	1114.2	4.1%
۱۲	1413.2	6.9%	1452.9	9.9%	1386.8	4.9%
۱۳	1627.9	7.6%	1691.4	11.8%	1597.6	5.6%
۱۴	2495.0	10.4%	2535.7	12.2%	2409.2	6.6%
۱۵	3309.6	9.3%	3439.8	13.6%	3264.2	7.8%
میانگین	---	4.2 %	---	7.8 %	----	2.7 %

Table 2: Results of computational solution time of the proposed algorithms

Test Problem Number	GA	SA	GA-SA
١	**17.1**	**10.2**	**24.3**
٢	**29.1**	**14.3**	**41.3**
٣	**46.45**	**25.71**	**62.6**
٤	**72.53**	**49.8**	**101.1**
٥	**104.72**	**76.2**	**147.4**
٦	**143.61**	**101.3**	**173.7**
٧	**187.01**	**138.8**	**234.2**
٨	**241.74**	**182.6**	**314.0**
٩	**324.55**	**270.54**	**438.6**
١٠	**459.76**	**350.1**	**552.8**
١١	**576.54**	**410.9**	**651.3**
١٢	**676.84**	**507.3**	**879.7**
١٣	**804.03**	**705.6**	**1061.0**
١٤	**1042.7**	**820.6**	**1214.3**
١٥	**1237.6**	**1085.8**	**1606.9**

The figure below shows the computational chat time of three algorithms with parameters set by Taguchi method. The vertical axis shows the chat value. Each of these diagrams shows the first, third, and middle quarters of the computational gap obtained for the 15 problems solved by each algorithm in each case. As can be seen from the figure, the least gap is for the hybrid algorithm and the highest gap is for the refrigeration simulation algorithm.

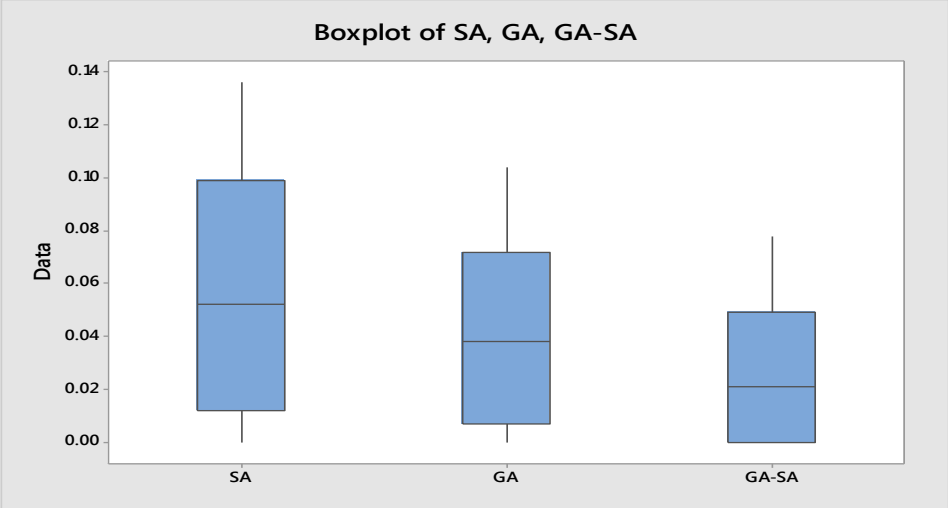

Figure 3: Comparison of the computational gap of the proposed solution methods with the set parameters

The following figure shows the solution time of three algorithms with parameters set by Taguchi method. The vertical axis shows the calculation time, which is in seconds. Each of these diagrams shows the minimum and maximum solution times and the median solution times obtained for the 15 problems solved by each algorithm in each case. As can be seen from the figure, the minimum solution time and the average solution time for the refrigeration simulation algorithm are less and the maximum solution time of the hybrid algorithm is longer than the genetic algorithm. As a result, the cooling time of the refrigeration simulation algorithm is less.

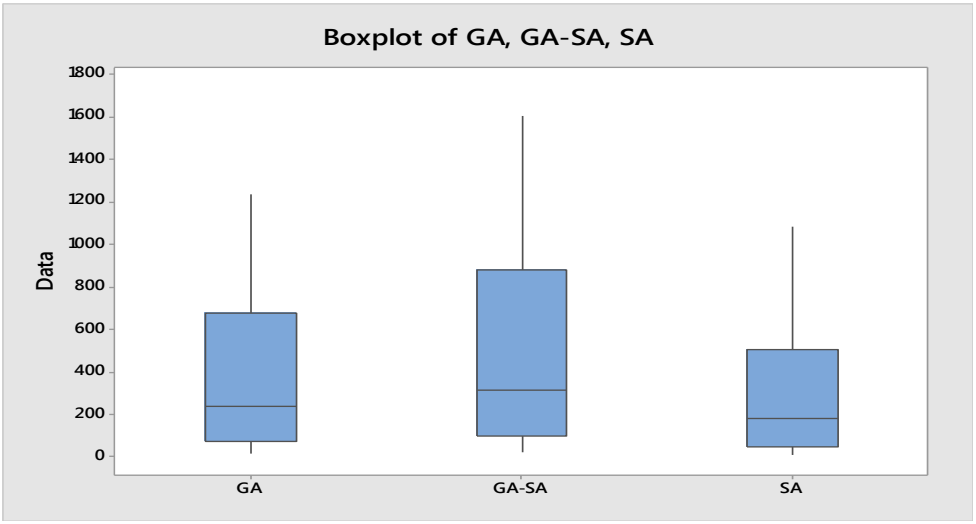

Figure 4: Comparison of computational time of proposed solution methods with set parameter

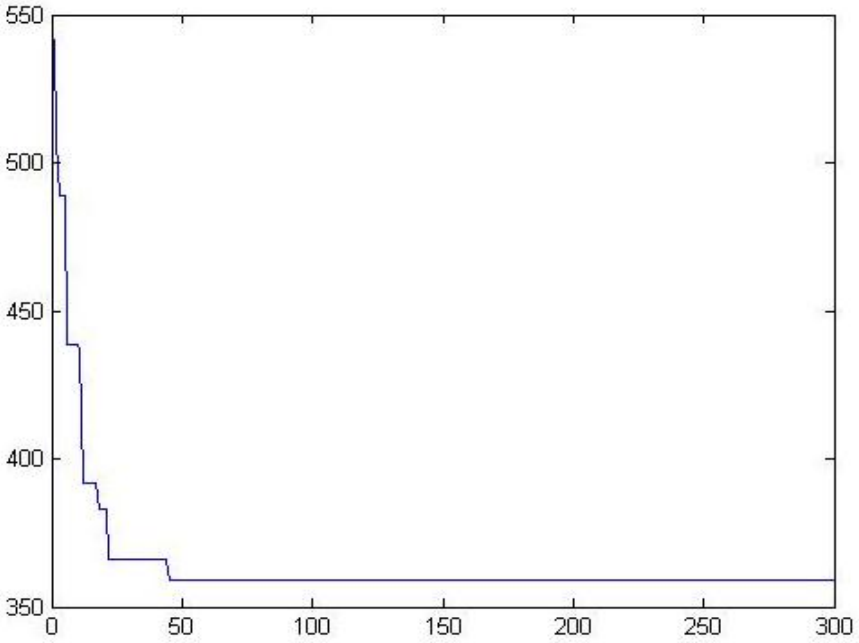

Figure 5: Problem convergence diagram for a hybrid algorithm

Analysis of variance to evaluate the quality of algorithms:

To better evaluate the algorithms, it is necessary to compare the algorithms with the help of statistical analysis. As mentioned, in this area we have used the hypothesis testing technique. In this research, we have used the hypothesis test of equality of the mean of three two-way societies (Tukey test). Thus, we considered the assumption of zero to be equal to the means of evaluation criteria in the three algorithms with a 95% confidence level. If the obtained P-value is less than 0.05 (1-0.95), the null hypothesis is rejected and we conclude that there is a significant difference between the performance evaluation criteria of the three algorithms and vice versa. In this study, two evaluation criteria have been examined. The first criterion of RPD is the best answer and the second criterion is the solution time of the algorithm. They are then compared. As shown in Figure 4-13, the p-value of the F test for the value of the objective function is 0.0. This value is less than 0.05. As a result, the null test hypothesis is rejected and there is a significant difference between the performance of the algorithms.

One-way ANOVA: RPD versus Alg

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	2	0.006996	0.003498	2.66	0.082
Error	42	0.055327	0.		
Total	44	0.062323			

Plot Area

S	R-sq	R-sq(adj)	R-sq(pred)
0.0362949	11.22%	7.00%	0.00%

Figure 6: Results of analysis of variance for the best value of the objective function

```
                Individual 95% CIs For Mean Based on Pooled StDev

Factor     N      Mean       StDev          95% CI
GA         15     0.04207    0.03452    (0.02315,  0.06098)
SA         15     0.0579     0.0457     ( 0.0390,   0.0768)
GA-SA      15     0.02740    0.02583    (0.00849,  0.04631)

Pooled StDev = 0.0362949
   Grouping Information Using Tukey Method

Factor     N      Mean       Grouping
SA        15      0.0579        A
GA        15      0.04207       A
GA-SA     15      0.02740       A
```

Figure 7: Output from the Tukey test for the best value of the objective function

As can be seen from the figure below, the Tokay test puts all algorithms in one category and does not make a significant difference in terms of performance between the algorithm and the performance of the three algorithms. Following figure shows the average diagram of the three algorithms. Although the average computational gap of the hybrid algorithm is less than the other two algorithms, performance difference is not significantly better than the other two algorithms.

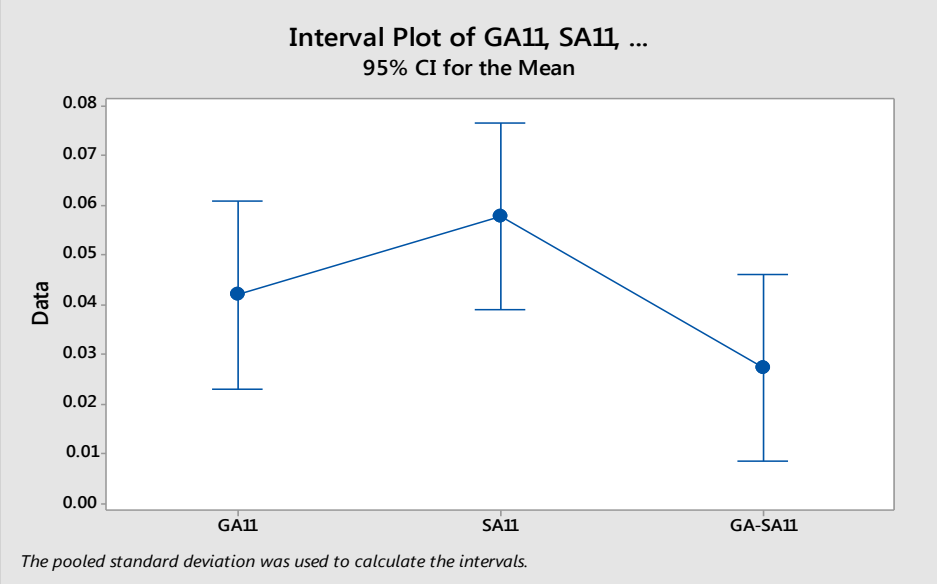

Figure 8: Interval diagram of the three proposed algorithms for the best value of the objective function

Conclusion:

The placement of controllers in software defined networks is a very important and fundamental issue in these networks. Numerous and sometimes contradictory goals can be considered for arranging controllers in Software defined networks. Goals such as load balancing, increased reliability, and latency are examples of these goals, the last of which,

latency, is more important than others. However, locating controllers for any purpose is an NP-Hard problem and can be solved by methods such as genetic algorithms and the SA algorithm. In this paper, the validation of the research model and the proposed algorithms were discussed. First, how to produce sample problems and the characteristics of sample problems are stated. Then the problem was solved with meta-heuristic algorithms, refrigeration simulation and hybrid algorithms and it was shown that the answers obtained from these algorithms are also valid. In order to compare the meta-heuristic algorithm introduced in this paper, first the parameters of the proposed algorithms were adjusted and the most appropriate value for their effective parameters was determined using the Taguchi method. Then the optimality criteria and solution time of the algorithms are compared and finally the performances of the algorithms are compared using statistical hypothesis tests. The results show that

-The minimum solution time and the average solution time for the refrigeration simulation algorithm is less and the maximum solution time of the hybrid algorithm is more than the genetic algorithm. As a result, the cooling time of the refrigeration simulation algorithm is less.

- The lowest gap is for the hybrid algorithm and the highest gap is related to the refrigeration simulation algorithm.

References

1. Mehr, S. Y., & Ramamurthy, B. (2019, December). An SVM Based DDoS Attack Detection Method for Ryu SDN Controller. In Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies (pp. 72-73.(
2. Akyildiz, I. F., Lee, A., Wang, P., Luo, M., & Chou, W. (2014). A roadmap for traffic Rengineering in SDN-OpenFlow networks. Computer Networks, 71, 1-30.
3. Rowshanrad, S., Abdi, V., & Keshtgari, M. (2016). Performance evaluation of SDN controllers: Floodlight and OpenDaylight. IIUM Engineering Journal, 17(2), 47-57.
4. Jarraya, Y., Madi, T., & Debbabi, M. (2014). A survey and a layered taxonomy of software -defined networking. IEEE communications surveys & tutorials, 16(4), 1955-1980.

5. Ali, J., Lee, S., & Roh, B. H. (2018, April). Performance analysis of POX and Ryu with different SDN topologies. In Proceedings of the 2018 International Conference on Information Science and System (pp. 244-249).
6. Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. (2012). On Controller Performance in Software-Defined Networks. In Presented as part of the 2nd {USENIX} Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services.
7. ONF, "Software-Defined Networking: The New Norm for Networks," Open Networking Foundation, Tech. Rep., April 2013.
8. Farhady, H., Lee, H., & Nakao, A. (2015). Software-defined networking: A survey. Computer Networks, 81, 79-95.
9. Hu, J., Lin, C., Li, X., & Huang, J. (2014, May). Scalability of control planes for software defined networks: Modeling and evaluation. In 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS) (pp. 147-152). IEEE.
10. Kumbhare, A. P., Kamath, D. G., Pandey, V. A., & Mukherjee, N. (2015). U.S. Patent No. 9,225,635. Washington, DC: U.S. Patent and Trademark Office .
11. Dixit, A., Hao, F., Mukherjee, S., Lakshman, T. V., & Kompella, R. R. (2014, October). ElastiCon; an elastic distributed SDN controller. In 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS) (pp. 17-27). IEEE.
12. T. M. M. D. M. Yosr Jarraya, "A Survey and a Layered Taxonomy of Software-Defined Networking," IEEE COMMUNICATIONS SURVEYS & TUTORIALS, vol. 16, 2014.
13. V. P. R. L. S. Alexander Shalimov, "Advanced study of SDN/OpenFlow controllers," in CEE-SECR, Moscow, Russia, 2013.
14. Rotsos, C., Antichi, G., Bruyere, M., Owezarski, P., & Moore, A. W. (2015, June). OFLOPS-Turbo: Testing the next-generation OpenFlow switch. In 2015 IEEE International Conference on Communications (ICC) (pp. 5571-5576). IEEE.
15. Kandoi, R., & Antikainen, M. (2015, May). Denial-of-service attacks in OpenFlow SDN networks. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) (pp. 1322-1326). IEEE.
16. Gheorghe, G., Avanesov, T., Palattella, M. R., Engel, T., & Popoviciu, C. (2015, April). SDN-RADAR: Network troubleshooting combining user experience and SDN

capabilities. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft) (pp. 1-5). IEEE.

17. Pérez, M., Perez, G. M., Giardina, P. G., Bernini, G., Neves, P., Alcaraz-Calero, J. M., ... & Koutsopoulos, K. (2018). Self-organizing capabilities in 5G networks: NFV & SDN coordination in a complex use case. Proceedings of EuCNC, 1-5.
18. Betzler, A., Quer, F., Camps-Mur, D., Demirkol, I., & Garcia-Villegas, E. (2016, June). On the benefits of wireless SDN in networks of constrained edge devices. In 2016 European Conference on Networks and Communications (EuCNC) (pp. 37-41). IEEE.
19. Camacho, F., Cárdenas, C., & Muñoz, D. (2018). Emerging technologies and research challenges for intelligent transportation systems: 5G, HetNets, and SDN. International Journal on Interactive Design and Manufacturing (IJIDeM), 12(1), 327-335.
20. R.Rajabioun, "Cuckoo Optimization Algorithm", Applied soft Computing , 2011 ,Vol.11, PP.5508-5518.
21. Bari, Md Faizul, et al. "Dynamic controller provisioning in software defined networks." Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013). IEEE, 2013.
22. Hu, Y., Wang, W., et al.: Reliability-aware controller placement for swaoftre-defined networks. In: IEEE International Symposium on Integrated Network Management (2013)
23. Yao, G., Bi, J.: On the capacitated controller placement problem in software defined networks. IEEE Commun. Lett. 18(8), 1339–1342 (2014)
24. Sallahi, A., St-Hilaire, M.: Optimal model for the controller placement problem in software defined networks. IEEE Commun. Lett. 19(1), 30–33 (2015)
25. Hock, David, et al. "Pareto-optimal resilient controller placement in SDN-based core networks." Teletraffic Congress (ITC), 2013 25th International. IEEE, 2013.
26. Heller, Brandon, Rob Sherwood, and Nick McKeown. "The controller placement problem." Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, 2012.
27. Wang, Guodong, et al. "A K-means-based network partition algorithm for controller placement in software defined network." Communications (ICC), 2016 IEEE International Conference on. IEEE, 2016.

28. Ksentini, Adlen, et al. "On using bargaining game for Optimal Placement of SDN controllers." Communications (ICC), 2016 IEEE International Conference on. IEEE, 2016.
29. Liao, Jianxin, et al. "Density cluster based approach for controller placement problem in large-scale software defined networkings." Computer Networks (2017).
30. Lange, Stanislav, et al. "Specialized heuristics for the controller placement problem in large scale SDN networks." Teletraffic Congress (ITC 27), 2015 27th International. IEEE, 2015.