International Journal of
Science and Engineering Investigations vol. 7, issue 76, May 2018
ISSN: 2251-8843

Applying the Particle Swarm Optimization Algorithm to Find the Dynamic Time Intervals in Controlling the Service Level Agreements in Cloud Computing

Fatemeh Saadatjoo[1], Mohammad Javad Rezaei[2], Marziyeh Rahmani[3]
[1,2,3]Department of Computer Engineering, Sciense and Art University, Yazd, Iran
([1]saadatjou@sau.ac.ir, [2]Mj.rezaei@stu.sau.ac.ir, [3]Rahmani69yazdstudent@gmail.com)

Abstract-Cloud computing is a technology which provides its services based on demand and reduction of costs. Service level agreement is an agreement between cloud service providers and cloud users. When the contents of this agreement are not observed, it is violated, and the service provider should pay a fine. To examine the service level agreement, this paper proposes a method whose goal is to find time intervals in which the greatest violation of the agreement is detected. This detection does not increase the time overhead imposed on the service provider. In order to detect the violation, the Particle Swarm Optimization algorithm and the Genetic algorithm were compared to find the time interval with a higher number of detected violations and a minimal time overhead. The findings indicated that, as compared to two algorithms could establish a better balance between the time overhead and the number of violations.

Keywords-*Particle Swarm Optimization Algorithm, Cloud Computing, Service Level Agreements, Gentic Algorithm*

I. INTRODUCTION

The main concepts of cloud computing were formed in the 1960s, when John McCarthy pointed out that cloud computing may be organized one day as one of the public industries [1]. Cloud computing is a new model in which computing resources, ranging from storing the data to completing the configuration of systems distributed based on demand, are available as scalable services on the Internet. In the case of the presence of parallelization in the software, users will be able to use the cloud computing solution to decrease the computational time. From the infrastructure point of view, it includes a set of distributed and parallel systems which are interconnected [2, 3, 4]. Cloud systems provide the conditions for users to implement their computing requirements on cloud servers, rather than buying several servers, and pay only for their own use. Also, in these conditions, when the users face an increase in costs, they can use more resources based on their needs. This technology is known as virtualization, through which the dynamic sharing of the hardware resources is provided [5]. Buyya et al. of the University of Melbourne have defined cloud as follows: Cloud is a parallel distributed computing system consisting of a set of virtual and interconnected computers, and, as one or more integrated computing resources, agreements are achieved based on the agreements related to level of service presentation; these agreements are established after negotiation between the service provider and the customer [6]. The most comprehensive or most suitable definition for cloud computing might be the one proposed by the National Institute of Technology and Standards (NIST). It has been defined in this way: Cloud computing is a model providing the conditions for easy access, based on the user's demand, to a set of changeable computing resources and configurations through a network. This system can be provided quickly without a need for resource management or direct service interference of the provider [7]. In cloud computing, a workload is a set of text files including the samples of percentages for using the processor in each interval as well as the data collected from more than 500 data centers throughout the world. Also, there are two types of time interval including static interval and dynamic interval. Static intervals are those whose values are fixed during the program implementation, but dynamic intervals are those whose values vary during the program implementation [8]. Researchers often cannot work in a real cloud environment due to some cloud computing system bottlenecks. So, simulation to model the mechanism and to evaluate the findings is essential [9]. The computing function has been based on the service level agreement model. This model includes information such as the characteristics of service and its name and values [10]. In the case of loss of data, agreement can help the users prove their claims against the cloud server. Agreement parameters include input and output bandwidths, processor, and memory [11-12-13]. Using the PSO algorithm, the best dynamic interval has been proposed as the output in this paper. The best dynamic interval is the one in which there is a balance between the violation of the service level agreement and the system time overhead.

This paper is organized in the following manners. The second section reviews the studies related to the subject along with their advantages and disadvantages. Section 3 describes the proposed algorithm. Section 4 examines and evaluates the findings of the research. Finally, section 5 concludes the paper.

II. REVIEW OF LITERATURE

Several methods have been used to control and monitor service level agreements in cloud environments. These methods, each with both advantages and disadvantages, are divided into two general classes. The first class includes methods whose values do not change during program computations. They are known as static methods. The second-class methods, whose values change during the program implementation, are known as dynamic methods. An automated monitoring process system, referred to as Sandpiper, has been presented in a static environment. The advantages of this system are considered to be automated monitoring, detection of important points, resetting the virtual machine when required, and prevention of service level agreement violations. However, lack of mapping the low-level metrics such as processor and memory to the parameters of the service level agreement is considered as the disadvantage of the system [14]. Reference [15] presents the Grid eye, which is a service-oriented monitoring system with a flexible architecture. Managing the resources with a specific approach in a grid environment is considered as the advantage of that study, while its disadvantage is the lack of mapping the low-level criteria and management of the service level agreement in static and dynamic environments. The study in reference [16] presents Netlogger, which is a distributed monitoring system and can collect information from a network and monitor it. Netlogger is a user programming interface to investigate the load before and after some requests or operations. Monitoring the network resources is the advantage of that study, but it is disadvantageous due to failing to manage service level agreements. In [17], to manage service level agreements, a framework is added to the grid discussed. This approach to service level agreements is applied to the grid network, when the goal is managing the agreement in the cloud environment. In [18], cloud monitoring of the program is presented by using the mOSAIC method. Development of monitoring techniques for such applications is described by using mOSAIC API, and the use of those techniques is expanded to cloud environments to collect information. In that paper, detection of violations from service level agreements to avoid any cost is not taken into account, nor is it commonly done. This is because the method only controls the programs which have been developed by using mOSAIC API. A lot of discussion has been made on providing dynamic service level agreements by using GRIA. Monitoring based on service level agreements is one of the advantages of this method, and failing to map low-level criteria to high-level criteria as well as being limited to a grid network is regarded as one of its disadvantages [19]. Independent management of quality of service (QoS) in a static environment has been discussed by using a method, like a proxy, whose implementation is based on a web-service agreement. In this regard, using service level agreements to prevent the abuse of parameters of quality of service is an advantage, but being limited to web services is a disadvantage [20]. Accessibility of historical data to provide a service level agreement and ability to evaluate the conditions of the agreement are considered as the advantages, while the lack of monitoring the low-level criteria and failure to map it to high-level contents are the disadvantages of that study [21]. The study in reference [22] presents a cloud exchange method with such strengths as better performance, reduced cost in cloud environments, and lack of support for dynamic scheduling. Indeed, an approach is presented to adapt a user service level agreement template to general agreements. One advantage of this method is that the restriction of static agreements is resolved with this template and mapping is defined between the public and private agreements. On the other hand, imposing costs on users and lack of linear management of the violation of service level agreements are regarded as the disadvantages of the method [10]. In [11], an infrastructure is provided to detect violations from the Desvi architecture agreement. This architecture automatically detects the violation of service level agreement in static intervals. Minimizing the interaction with users and preventing the violation of service level agreement are the advantages of this approach, but increased time overhead in short intervals is a disadvantage. In reference [23], the Bee-MMT method has been presented to reduce violations of service level agreements in static intervals as a requirement to enhance the quality of services and satisfy the customers. Reduced energy consumption, leading to more violations of service level agreements, as compared to other methods, is the advantage of that study. However, it is disadvantageous due to the reduced time overhead of system and the lack of dynamization. Using a genetic algorithm, a smart mechanism is devised to find the dynamic time intervals to monitor and optimize the conflict of interests in Desvi architecture. Failing to reduce the overhead time applied to the system as well as failure to provide accurate information on implementation is considered as a disadvantage of that study [24]. In [25], a colonial competition algorithm is presented to deal with the conflict of interests between system overhead and violation in cloud computing. This algorithm is intended to find dynamic intervals in Desvi architecture. Establishing a relative balance between a situation in which intervals are longer and one in which the intervals are selected from short intervals is an advantage of the study. However, it may be criticized for the lack of implementation in the Cloud sim environment. Evolutionary algorithms can be used to choose dynamic time intervals. The use of such algorithms depends on the number of intervals which can be examined to detect violations. Given a high workload of the used Planet Lab, at each attempt to achieve a dynamic interval, the workload increases, and, using a manual method, it is not possible to calculate the time overhead or to detect violations. A PSO algorithm is used in that paper owing to its high convergence speed in achieving the personal best (PBest).

III. PROPOSED ALGORITHM

The current challenge in detecting the violations of service level agreements is that, as the smaller the time interval is, the more violations are detected, and the longer the overtime will be. On the other hand, when the time interval is selected to be larger, the number of violations detected and the time overhead applied to the system will be lower. The main goal is to establish a balance between the time overhead of the system and the number of violations detected. The time interval in static intervals increases due to detection of more violations [11]. When there is a direct relationship between the time overhead and the number of violations, a system failure will

result. In order to solve this problem, we search for dynamic intervals. To select the best time intervals with which to achieve the mentioned balance, a method is presented in this paper. These time intervals should be able to establish a balance between the system time overhead and the number of violations detected. To this end, for each of the parameters of a service level agreement, a workload is generated based on the pre-assumed workload of the cloud sim library. This workload needs to evaluate the rate of violations at each time interval in minutes. As Planet Lab data report only the number of violations detected in an interval, which is an integer multiple of 5 minutes, they are not appropriate for our method. For this reason, we require to generate a workload so that we can determine the number of violations detected at any certain time. Each workload is evaluated as 1440 minutes for one day. Given the selected time interval m, the number of times a system violation is recorded is obtained using equation (1) as follows:

$$p = \frac{1440}{m}, 1 \le m \le 1440 \quad (1)$$

where m represents minutes. For each time interval m, it is required that the cost of measurement, which has a direct relationship with the length of the interval, and the number of violations detected be calculated. To create a balance between the cost of measurement and the number of violations detected, equation (2) is used [11]:

$$c = \mu \times C_m \sum_{\varphi \in [cpu, memory, Storge]} \alpha(\varphi) \times C_v \quad (2)$$

where μ is the number of measurements, C_m is the cost of measurements, $\alpha(\varphi)$ is the number of the undetected violations of the service level agreement, and C_v is the cost of the loss of agreement violation. In equation (2), the number of measurements has a direct relationship with time. For example, if the measurement intervals are 5 minutes and our workload is implemented for one day (24 hours), the number of measurement intervals will be 1440 minutes. Accordingly, the number of measurements will be 522 for 10-minute intervals. It is clear that this number will be lower for intervals with a larger length and larger for intervals with a smaller length. The sum of these parameters should establish a balance between the cost of measurement and the cost of undetected violations. As the number of the selected intervals is high and since the total workload should be evaluated at each selected interval in a 24-hour period, we are looking for a time interval which can establish a balance between the cost of measurement and the number of violations detected. For this purpose, a time interval of m is selected, among all other intervals, minimizing the value of equation (2). Considering what is stated above, to find an interval among the current time intervals, an evolutionary algorithm is required. Accordingly, the PSO algorithm is used in this paper. As previously explained, finding the best time interval that can establish a balance between the measurement cost and the number of undetected violations is considered as an innovation of this study. In other words, the study aims to find a dynamic interval determined based on the conditions of the problem. It means that this interval can vary by a change in the workload, the parameters of Service level Agreement, the number of virtual machines available in the cloud space, and the way to allocate each job of the workload to a virtual machine. The main objective of this paper is to dynamize the selection of time intervals to measure the number of violations detected per a workload in any condition.

A. Particle swarm optimization

Particle Swarm Optimization (PSO) method has been inspired by the social behavior of a group of birds or fish in search for food to direct the population to a promising area in a search space [26, 27]. Each bird uses the past experiences of other members to find food. The main base of the particle swarm optimization algorithm is sharing of information among the group members. In the particle swarm optimization algorithm, the solution of each problem is the bird position in the search space, known as 'particle'. All the particles have a merit gained through a fitness function, which is the optimization goal. In addition, each particle has a component called 'speed', which specifies its path in the search space. The particle swarm optimization algorithm begins with a group of random solutions. Then, it begins searching to find an optimal solution in the problem space by updating the situation and the speed of each particle. Each particle is defined in a multidimensional form (depending on the problem nature) with two values, X_{id} and V_{id} , which represent the place and the speed related to the dth dimension and the ith atomic particle respectively. At any step of the population movement, each particle is updated based on two best values. The first value is the best solution in terms of merit, obtained for each particle separately so far. This value is the best person called 'pbest'. The other best value obtained by the particle swarm optimization algorithm is the one obtained by all the particles in the population. This value is the general best called 'gbest'. After finding the two values of pbest and gbest, each particle updates its new speed and place through the following two equations [27]:

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (3)$$

$$V_{ij}(t+1) = w(V_{ij}(t) + c_1 r_{1i}(t)[pbest_{ij} - X_{ij}(t)] .c_2 r_{2i}(t)[gbest_i(t) - X_{ij}(t)]) \quad (4)$$

where w is the inertial weight, c_1 and c_2 are the acceleration coefficients, and r_1 and r_2 are the random numbers in the interval (0 and 1). The final value of the speed of each particle is limited to $[-V_{MAX}, V_{MAX}]$ interval to prevent the divergence of the algorithm. c_1 , c_2 and w are the PSO parameters, and the convergence of the algorithm depends on the value of these parameters. c_1 and c_2 are the numbers between 1.5 and 2, but the best selection is $c_1 = c_2 =$ 2.05 and $0 \le w < 1$. The convergence strongly depends on the value of w , and it is better to define it defined dynamically. In the interval (0.2-0.9), this value is reduced linearly during the population evolution. First, it should be large enough, but, in later steps, a

small w will result in better convergence. PSO, updating its population by using equations (3) and (4), is called base or standard PSO [28].

B. Generation of particles in the PSO algorithm

Particle needs to be defined if the PSO algorithm is to be used in selecting the best dynamic interval to detect violations. Considering the number of parameters in the service level agreement, each particle is composed of five parts including speed, the best value, and three parameters of detecting the violations. These parameters include memory, processor, and bandwidth. Since the workload related to planetlab is implemented in the intervals of multiples of five, they cannot be used for the proposed method. For this reason, a new workload based on planetlab will be created, which can be implemented and measured in each time interval. This workload is implemented in different time intervals, and utilization of each workload is determined based on the number of physical and virtual machines. To calculate the utilization, the value of each job of the workload allocated to the processor compares the memory and the bandwidth to the requested value of each job. In case of any difference or lack of allocation of the job request, the lack of request is calculated in the form of violation of that parameter. To use the solutions obtained by implementing the workload in each interval, the utilization obtained from each job needs to be stored along with its corresponding implementation time on one map. The map has the role of updating each particle. Accordingly, the particles are generated randomly. Then, 12 jobs corresponding to them are selected through the map based on the value of each parameter of detecting the violation. The value of the fitness function related to each selected map is implemented on the basis of equation (2). The mean value of the fitness function for 12 maps is selected as the best value for that particle. This best value is the same as the length of the considered interval. Each particle is updated based on its speed and its best value. The operation continues as long as the value of equation (2) for the best particle in each repetition of the algorithm is not less than the threshold value. The best particle is selected, and the best value is introduced as the dynamic interval for the generated workload after the algorithm is ended. It should be noted that the best particle in each implementation of the algorithm is a particle whose best value is among the four particles. Fig. 1 displays the flowchart of the introduced algorithm.

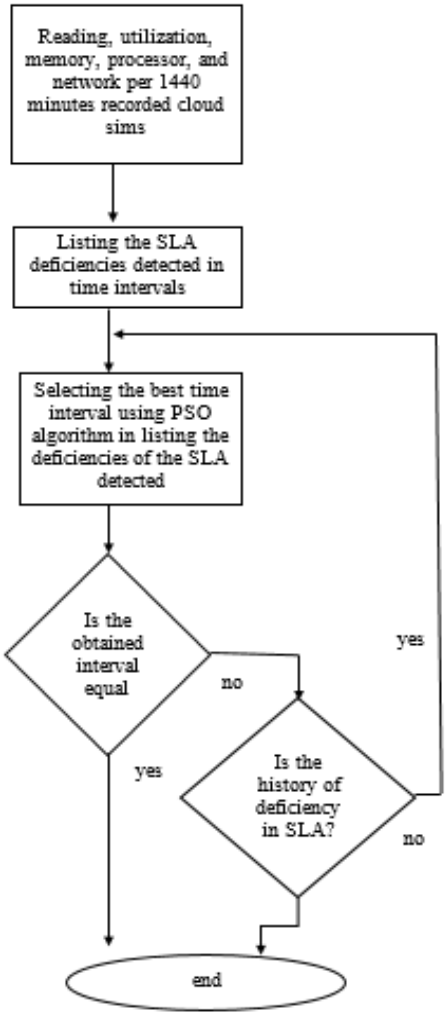

Figure 1. Flowchart of the proposed algorithm in finding the dynamic interval

IV. SETTING THE PARAMETERS AND TESTS AND EVALUATION OF THE RESULTS

In order to evaluate the PSO algorithm for utilization of memory, processor, and network, among the Planetlab data, 1440 data are randomly selected as the workload. These data are such that, if we consider a day as long as 24 hours, we will have $24 \times 60 = 1440$; that is to say, each day equals 1440 minutes. It is assumed that the interval value is in minutes. This assumption means that calculation in seconds is overlooked because it seems that the workload applied to the system is too great to investigate the variations second by second. Calculations in minutes are, however, more rational.

As stated before, prior to applying the PSO algorithm, it is required that a map be generated. In this study, the number of particles for each implementation of the algorithm is considered four, and the threshold value is set to be 0.5.After implementing the PSO algorithm once, all the particles are arranged in a descending order according to their fitness function values. Updating of each particle and determination of its direction is done on the basis of its speed. The PSO algorithm output, which is the best time interval proposed, can be evaluated in two aspects. In one aspect, the highest number of violations taking place in time intervals is detected. In the second aspect of evaluation, the time overhead is applied to system, which has been reduced in some time intervals. To simulate this operation, the Cloud sim Library version 0, 3, 3 is used. This simulation is done by the NetBeans IDE 8.1 software. In the simulation environment, 50 physical devices have been designed, and 50 machines have been designed in each device. Table 1 illustrates the features of the selected physical and virtual machines.

TABLE I. FEATURES OF THE SELECTED PHYSICAL AND VIRTUAL MACHINES

	Operating system	Processor	RAM memory
physical machine	Linux	4-core	4GB
virtual machine	Ubunta	1-core	1 GB

In order to examine the rate of violations of the service level agreement in the PSO algorithm by the request of the memory user, it is assumed that the user's request of the memory has been at least 700 MB. As shown in (Fig. 2), in different minutes of a day, different values of the memory are allocated by the host to the user. The horizontal axis is the time in minutes, and the vertical axis is the amount of memory allocated to the host in MB.

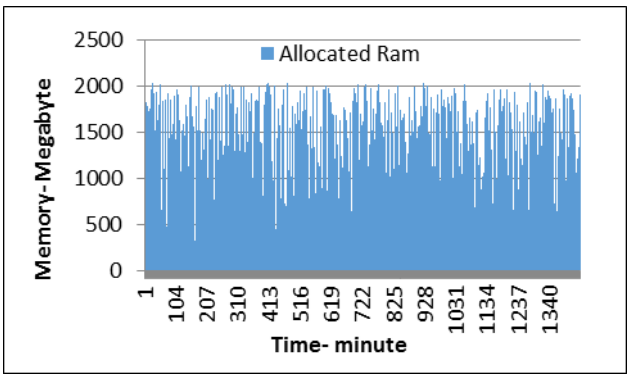

Figure 2. Memory allocated to host

Fig. 3 illustrates the relationship between time intervals and SLA (Service Level Agreement) violations. The diagram indicates that, in most cases, more violations are detected in shorter time intervals. For example, for a 5-minute interval, 130 MB of memory is obtained. These values have been obtained before giving them to the PSO algorithm.

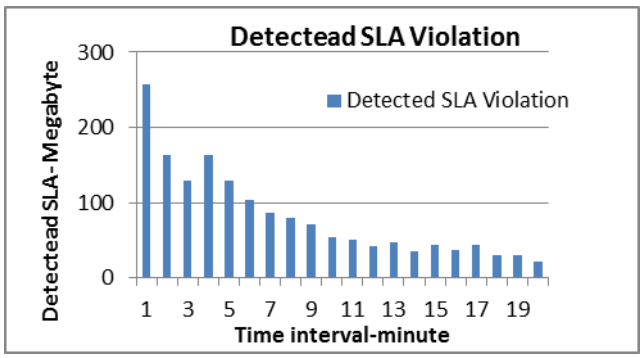

Figure 3. SLA violations detected in time intervals

Fig. 4 shows the relationship between the time used to detect the violations and the time intervals. As suggested by the figure, the time used to detect 130 violations is equal to 7056343 nanoseconds. The goal of PSO is to obtain the time interval in which the SLA violations are increase and the time overhead applied to the system is reduced.

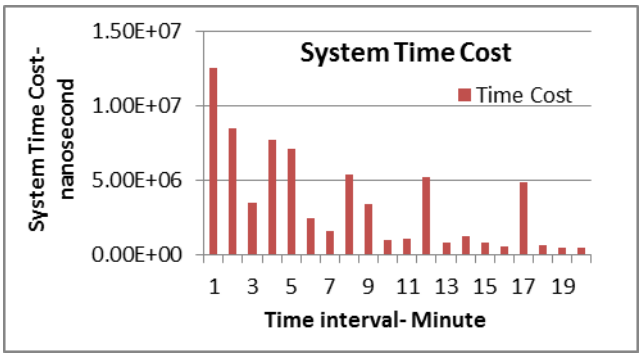

Figure 4. Time overhead applied to the system

As stated before, the generated map includes the rate of violations corresponding to the service level agreement parameters and the time of finding them in the determined interval for one day, given as the input to the PSO algorithm. According to the history of the map, this algorithm decides in which interval the balance between the implementation time and the number of the detected violations has been established. In the present study, after completion of the algorithm, the best particle is obtained with the time interval of seven. Fig. 5 suggests the reason for selecting this interval. In this figure, the horizontal axis represents the time interval, and the vertical axis represents the value of the cost function. As the figure further shows, in the time interval of seven, the lowest value is obtained from equation (2). In other words, in this interval, there is a balance achieved between the number of the detected violations and the implementation time.

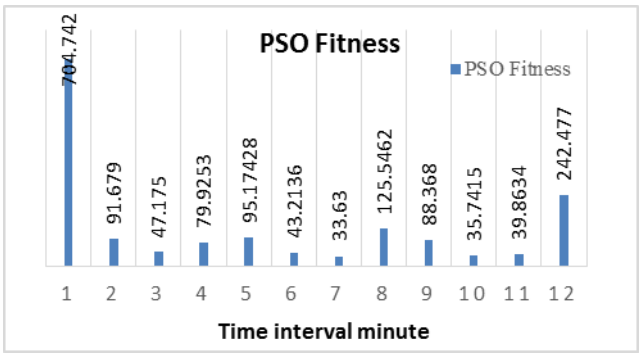

Figure 5. Fitness function value in the PSO algorithm

There are twelve intervals selected. Selection of this number is due to the fact that, in intervals more than twelve, the number of detected violations is reduced significantly and the number of undetected violations is increased. This increases the value of the fitness function in these intervals. To examine the utilization of the PSO algorithm, another test is conducted using a genetic algorithm. This test environment has been considered similar to that of the PSO algorithm. On this account, the input map of the genetic algorithm is the same as the input map of the PSO algorithm. Similarly, the number of genes is equal to the number of particles, and the fitness function has been considered the same as equation (2). As illustrated in Fig. 6, the genetic algorithm in the sixth interval has the minimum value of the fitness function. In general, by comparing the values of the fitness functions of the PSO and genetic algorithms, one can find out that the PSO algorithm can detect more violations in time intervals. This is because the value of equation (2) is higher when the minimum number of the undetected violations is high.

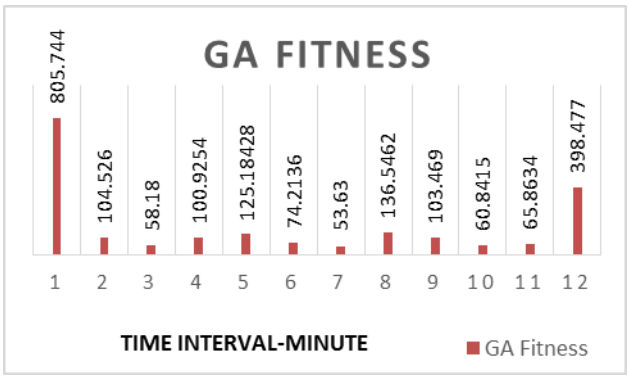

Figure 6. The value of fitness function in the GA algorithm

Fig. 7 compares the PSO algorithm with the genetic algorithm. The goal of both PSO and genetic algorithms is reducing the time overhead applied to the system and reducing the number of violations.

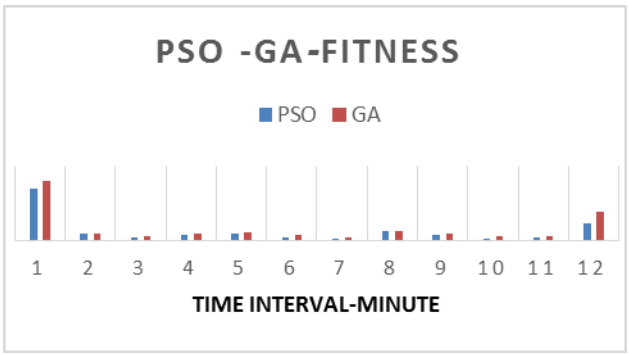

Figure 7. Comparing the PSO and the GA algorithms

V. CONCLUSIONS

In a cloud computing environment, appropriate facilities such as processors, input and output systems, bandwidths, and memory are provided in the form of heterogeneous resources for users. In this paper, a method is presented to detect the highest number of violations of the service level agreement in dynamic time intervals based on the PSO algorithm. To test the utility of PSO versus another algorithm, an experiment is performed through the genetic algorithm. The environment of this experiment has been set to be similar to that of the particle swarm algorithm. For this purpose, the input map of the genetic algorithm is considered to be the same as the input map of the particle swarm algorithm. Also, the number of genes is equated to the number of particles. The findings indicate that the genetic algorithm in the sixth interval has the minimum value of the fitness function.

In general, through a comparison of the values of the fitness functions of the two algorithms, it has emerged that the particle swarm algorithm can detect more violations in time intervals. This is because the value of the fitness function increases when the minimum number of the undetected violations is higher.

REFERENCES

[1] Y. Foster, I. Zhao and S. Lu. Raicu, "Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing. Environment," Workshop, IEEE, 2008, pp. 1–10.

[2] A. Goscinski and B. Michael, "Toward Dynamic and Attribute Based Publication, Discovery and Selection for Cloud Computing. Journal of Future Generation Computer Systems," vol. 27, no.7, 2010, pp. 947-970.

[3] S. r. Pakizeh, "cloud computing simulator," Volume 2, first edition, Tehran, Pardis Danesh, 2012.

[4] M. Goldner and K. Birch, "Resource sharing in a cloud computing age," Interlending and Document Supply, vol.40, no.1, 2012, pp. 4-11.

[5] SH. Jamali, S. Malek Taji and M. Analuei, "Positioning the virtual machines using the colonial competitive Algorithm," Electrical Engineering Magazine, Tabriz University, issue 1, Volume 46, Spring 2016, pp.53-62.

[6] R. Buyya, J. Broberg and A. Goscinski, "Cloud Computing: Principles and paradigms," John Wiley and Sons, vol.87, 2010.

[7] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication, 2011.

[8] M. White, H. Melvin and M. Schukat, "The Impact of Dynamic Monitoring Interval Adjustment on Power Consumption in Virtualized Data Centers", 2014.

[9] W. Long, L. Yuqing and X. Qingxin, "Using cloudsim to model and simulate cloud computing environment. In Computational Intelligence and Security," 9th International Conference on IEEE, 2013, pp.323-328.

[10] M. Maurer, V. C. Emeakaroha, I. Brandic and J. Altmann, "Cost–benefit analysis of an SLA mapping approach for defining standardized Cloud computing goods," Future Generation Computer Systems, vol.28, no.1, 2012, pp. 39-47.

[11] V. C. Emeakaroha, M. A. S. Netto, R. N. Calheiros, I. Brandic, R. Buyya, C. A. De Rose and et al, "Towards autonomic detection of SLA violations in Cloud infrastructures. Future Generation Computer Systems," vol.28, no.7, 2012, pp. 1017-1029.

[12] W. Hussain, F. k. Hussain, O. Hussain and R. Bagia, "Change, E., Risk-based Framework for SLA Violation abatment from the Cloud Servise Provider's perspective.The Computer Journal," 2018.

[13] M. A. T. Rojas, F. F. Redigolo, N. M. Gonzalez, F. V. Sbampato, T. C. M. de Brito Carvalho, K. W. Ullah and A. S. Ahmad, "Managing the Lifecycle of Security SLA Requirements in Cloud Computing, In

Developments and Advances in Intelligent Systems and Applications, Springer," Cham, 2018, pp.119-140.

[14] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," Computer Networks, vol.53, no.17, 2009, pp.2923-2938.

[15] W. Fu and Q. Huang, GridEye: "A service-oriented grid monitoring system with improved forecasting algorithm," In Fifth International Conference on Grid and Cooperative Computing Workshops, 2006, pp. 5-12.

[16] D. Gunter, B. Tierney, B. Crowley, M. Holding and J. Lee, Netlogger: "A toolkit for distributed system performance analysis.8th International Symposium on Modeling," Analysis and Simulation of Computer and Telecommunication Systems, 2000, pp.267-273.

[17] Micsik, H. M. Frutos, I. Kotsiopoulos and B. Koller "Semantically supported SLA negotiation," 10th IEEE/ACM International Conference on Grid Computing, 2009, pp.169-170.

[18] D. Petcu, B. Di Martino, S. Venticinque, M. Rak, T. Máhr, G. E. Lopez, V. Stankovski and et al, " Experiences in building a mOSAIC of clouds. Journal of Cloud Computing": Advances, Systems and Applications, vol.2, no.1, 2013, pp.1-12.

[19] M. Boniface, S. Phillips, S. Sanchez-Macian and M. Surridge, "Dynamic service provisioning using GRIA SLAs," In Service-Oriented Computing-ICSOC Workshops, 2009, pp.56-67.

[20] B. Koller and L. Schubert, "Towards autonomous SLA management using a proxy-like approach," Multiagent and Grid Systems, vol.3, no.3, 2007, pp.313-325.

[21] M. Comuzzi, C. Kotsokalis, G. Spanoudkis and R. Yahyapour, "Establishing and monitoring SLAs in complex service based systems. Proceedings of the 7th International Conference on Web Services," 2009, pp.783–790.

[22] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," Future Generation Computer Systems, vol.28, no.2, 2012, pp.358-367.

[23] S. Saravanan, V. Venkatachalam and S. T. Malligai, "Optimization of SLA violation in cloud computing using artificial bee colony. International Journal," Adv. Eng, vol.1, no.3, 2015, pp.410-414.

[24] S. Zanganeh and A. Farahi, "Proposed approach to find optimal cost of automatic detection of service level agreement in Desvi architecture by allocating dynamic intervals," National Conference on Computer Engineering and Information Technology Management, Tehran, 2014.

[25] N. Ghafuri and F. Saadatjoo, " Find the Best Time Intervals in the Control of Service Level Agreement Commitments in Cloud Computing Using Colonial Competitive Algorithm. International Science and Investigation journal," vol.4, no.4, 2015, pp.20-35.

[26] W. M. Korani, H. T. Dorrah and H. M. Emara, "Bacterial Foraging Oriented by Particle Swarm Optimization Strategy for PID Tuning," International Symposium on Computational Intelligence in Robotics and Automation, 2009, pp.445-4506.

[27] R. Houshmand, H. Mohkami and A. Khodabakhshian, "new method for optimal positioning of capacitors and generators distributed in distribution networks Using PSO-oriented bacteriological search algorithm. Tabriz Electrical Engineering Magazine," Issue 2, Volume 39, Spring 2016, pp.53-65.

[28] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," International Conference on Computational Cybernetics and Simulation, vol.5, 1997, pp.4104-4108.

Fatemeh Saadatjoo received her B.Sc. and M.Sc. and Ph.D. degrees from the computer Science department of Yazd University, Iran, in 1998, 2000 and 2009, respectively. She is currently an assistant professor in the C. department of Science and Art University, Yazd, Iran. Her areas of interest include software engineering, data base, data mining and fuzzy systems.

Mohammad Javad Rezaei Received her B.sc Degree on Computer Software From Science and Art University, Yazd, Iran in 2014.He is Currently a M.sc Student Continuing Computer Software at Science and Art University, Yazd, Iran. His research interests are in the Field of Cloud Computing and Deep learning.

Marziyeh Rahmani Received her B.sc Degree on Computer Software from Pooya University,Yasuj, Iran in 2012.She is Currently a M.sc Student Continuing Computer Software at Science and Art University, Yazd, Iran. Her research interests are in the Field of Cloud Computing.