Int. J. Nonlinear Anal. Appl.
Volume 12, Special Issue, Winter and Spring 2021, 343-360
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.5188

Resource allocation optimization in cloud computing using the whale optimization algorithm

Seyed Hasan Hosseini[a], Javad Vahidi[b,d,*], Seyed Reza Kamel Tabbakh[c], Ali Asghar Shojaei[a]

[a]Deportment of Computer Islamic Azad University Neyshabur Branch, Neyshabur, Iran
[b]Iran University of Science and Technology, Tehran, Iran
[c]Deportment of Computer Engineering, Mashhad Branch, Islamic Azad University Mashhad, Iran
[d]Department of Mathematical Sciences, University of South Africa, UNISA0003, South Africa

(Communicated by Ali Jabbari)

Abstract

Cloud computing is a massively distributed system in which existing resources interact with user-requested tasks to meet their requests. In such a system, the problem of optimizing Resource Allocation and Scheduling (RAS) is vital, because recourse allocation and scheduling deals with the mapping between recourses and user requests and also is responsible for optimal allocating of tasks to available resources. In the cloud environment, a user may face hundreds of computational resources to do his work. Therefore, manually recourse allocation and scheduling are impossible, and having a schedule between user requests and available recourses seems logical. In this paper, we used Whale Optimization Algorithm (WOA) to solve resource allocation and task scheduling problem in cloud computing to have optimal resource allocation and reduce the total runtime of requested services by users. The proposed algorithm is compared with the other existed algorithms. Results indicate the proper performance of the proposed algorithm than other ones.

Keywords: Cloud Computing, Makespan, Task, Resource, Whale Optimization Algorithm.

1. Introduction

Cloud computing is defined as a new generation of computing that make it possible to implement computational services and resources available in data centers (such as machines, networks, storage space, operation systems, application software) in the network platform [1, 26]. The logic of the cloud computing is based on virtualization (a technique that separates computational functions from physical hardware). This technique allows users to divide and share the infrastructure of physical machines (such as CPU, memory, I / O, storage space, and network interface) [3]. Programs are not

*Corresponding Author
Email addresses: seyyed.h12@gmail.com (Seyed Hasan Hosseini), jvahidi@iust.ac.ir (Javad Vahidi), drkamel@mshdiau.ac.ir (Seyed Reza Kamel Tabbakh), shojaei2012@gmail.com (Ali Asghar Shojaei)

Received: January 2021 *Accepted:* May 2021

run on physical machines, but they are run on virtual machines instead. Virtual Machine (VM) is a implementation of software for a computational environment related to simulation of a physical machine running directly on a physical hardware [5, 27]. Virtual Machine Monitor (VMM) is also applied to create and manage VMs (including Xen, VMware, VirtualBox, and KVM) [6]. Virtual machine configuration or resource allocation controls the sharing of assigned physical resources (CPU, memory, I/O bandwidth) to VMs. The problem of optimizing the performance of virtual programs (such as applications running on VMs) is very important in the success of cloud computing paradigm, since the VM configuration affects the program's performance[2, 4].

Recently, cloud computing has emerged like a new paradigm for service provisioning on the Internet using a dynamic pool of virtual computing resources. Most cloud service providers such as Microsoft, Amazon, and Google, have increasingly expanded their data centers to meet customer needs, which in turn increased energy consumption and CO2 emissions to the environment.

According to studies, the energy consumption of large data centers in the world equals to that of 25000 homes. Between the years 2005 and 2010, data centers' power consumption has increased by 56% and it is expected that if this trend continues, the annual energy costs of data centers' operators will exceeds their equipment costs. It is also estimated that 2% of global carbon emission is related to information and communication technologies and 14% of it is related to data centers. With respect to the economic and environmental impact of data centers' energy consumption, the existence of energy-aware approaches that lead to significant reduction in operational costs of data centers and negative effects of CO2 emission into the environment is necessary [7, 27, 29].

In this paper whale optimization algorithm is used to solve resource allocation optimization in cloud computing and then an optimal solution is suggested to solve the mentioned problem.

The structure of the paper is organized as follows: Section 2 describes the related works. Whale optimization algorithm is explained in Section 3. Proposed algorithm is introduced in Section 4. Simulation results and conclusions are presented in Sections 5 and 6 respectively.

2. Related work

Many researchers have been working on the optimization problems and the applications of the computers in order to optimize them[[32] -[34]]. In particular, many kinds of research have been done on the optimization running programs in virtual environments and resource allocation. In the following we investigate some of them.

Baranwal and Vidyarthi [8] proposed a multi-attribute combinatorial double auction for Cloud resource allocation by considering price and other quality of service parameters. Proposed approach imposes a penalty on the provider for false QoS assurance in order to win the auction and at the same time customer is compensated.

Ma et al. [9] explored existing RAS policies and algorithms with their associated parameters and then classified five major topics in cloud computing, namely locality-aware task scheduling, reliability-aware scheduling, energy-aware RAS, Software as a Service (SaaS) layer RAS and workflow scheduling into three parts: performance-based RAS, cost-based RAS, and performance- and cost-based RAS.

Shrimali and Patel [10] proposed an energy-efficient resource allocation using Multi-Objective Optimization (MOO) method and implemented it in CloudSim environment. They considered power consumption and SLA violations and achieved on average, 32% power consumption and 5.41% SLA violations over a 24-h period with maintaining QoS requirements, 51% improvement in SLAV and 13% reduction in power consumption.

Pradhan et al. [11] have used round robin resource allocation algorithm to satisfy customer demands and reducing waiting time. Authors have answered the question "what is the optimal time

quantum in round robin algorithm?" using dynamic time quantum instead of fixed time quantum.

Lin et al. [12] discussed on resource allocation at the application level and proposed a threshold-based dynamic resource allocation scheme for cloud computing that can allocate virtual resources (virtual machines) dynamically based on their load changes and therefore improve resource utilization and reduce user usage cost.

Kheiri et al. [13] proposed an Genetic Algorithm (GA) based approach to guarantee service quality through providing adequate resources and improving system performance, meeting users' requirements and providing maximum resource efficiency. To do so, they considered the effects of resource allocation on the service performance and cost.

Jena and Mohanty [14] proposed a two-phase approach including genetic algorithm-based resource allocation and shortest task first scheduling in multi-cloud computing in order to remove the gap existed between customers fluctuating requirements and existed infrastructure for the service with the aim of mapping tasks to VMs to minimize makespan and maximize customer satisfaction.

Boloni and Turgut [15] explored applications ranging from weather prediction to financial modeling in which output quality increases with the deployed computational power and proposed a computation scheduling by considering both the financial cost of the computation and the predicted financial benefit of the output named value of information (VoI)-based scheduling algorithm.

Hallawi et al. [16] proposed two algorithms, namely Combinatorial Ordering First-Fit Genetic Algorithm (COFFGA) and Combinatorial Ordering Next Fit Genetic Algorithm (CONFGA) and then combined them. The aim of the combined algorithm is to reduce the total number of running servers and resources wastage per server. Authors compared their algorithm with existing algorithms in terms of performance and robustness.

Samimi et al. [17] proposed a new market model called "Combinatorial Double Auction Resource Allocation" (CDARA) to meet both users and providers requirements and then evaluated its efficiency from an economic point of view. Li et al. [18] developed comprehensive models of cloud resource provisioning for both private and public cloud-based automotive system by considering issues such as stochastic communication delays and task deadlines. In fact they developed centralized resource provisioning model for private cloud by using chance constrained optimization to utilize the cloud resources for best Quality of Services and a decentralized auction-based model for public cloud by using reinforcement learning to obtain an optimal bidding policy for a "selfish" agent.

Maguluri et al. [19] divided resource allocation problem into a routing or load balancing problem and a scheduling problem in which jobs arrive according to a stochastic process and request resources like CPU, memory and storage space and used the join-the-shortest-queue routing and power-of-two-choices routing algorithms with MaxWeight scheduling algorithm to solve the mentioned problem which are queue length optimal in the heavy traffic limit.

Xiaoying et al. [20] proposed two dynamic resource allocation methods: (1) speed switching (SS) method and (2) speed increasing (SI) method in order to enhance storage utilization and reduce download time in cloud download service. Muthu and Enoch [21] proposed a new optimized approach named GABFO by combining bacterial foraging optimization algorithm and GA in order to schedule jobs and allocate resources in a efficient manner then compared their method with PSO, GA, BFO.

Sheuly et al. [22] investigated three different allocation-based GA algorithms, Particle Swarm Optimization (PSO) and Best-fit heuristic algorithm for cost optimization and computational time minimization as well they investigated the effect of the algorithm parameters (i.e. population size, probability of mutation and probability of crossover) on the objective function in GA and found that the performance of the algorithm is condition-dependent, the conditions such as available resource, number of VMs etc.

Ficco et al. [23] oroposed a bio-inspired coral-reefs optimization based approach to model cloud

elasticity in a cloud-data center in which they utilized classic Game Theory to optimize resource reallocation by considering cloud provider's objectives and customer requirements through formalizing Service Level Agreements using a fuzzy linguistic method.

HU et al. [24] focused on the advantages and disadvantages of ACO algorithm and its combination with GA for increasing the speed of the searching ability in order to overcome the shortage of initial pheromone decomposition and increase the convergence speed to solve resource allocation problems in cloud computing.

3. Whale optimization algorithm (WOA)

The WOA is a novel meta-heuristic algorithm that is inspired by the Humpback whales [25]. In this algorithm, the optimization process starts with producing a randomly generated population for whales. These whales try to find the prey's (optimum) location, then append (optimize) them by encompassing or bubble-net or method. In the encompassing method [25], the Humpback whales improve their current location according to the best location as follows:

$$D = |C \odot X^*(t) - X(t)| \tag{3.1}$$

$$X(t+1) = |X^*(t) - A \odot D|, \tag{3.2}$$

In which, D is the distance between the position vector of both prey $X(t)^*$ and whale $X(t)$, and t is the current iteration number. A and C are coefficient vectors, and are defined as follows:

$$A = 2a \odot r - a \tag{3.3}$$

$$C = 2r, \tag{3.4}$$

In which, r is a random vector $\in [0, 1]$, and the value of a is decreased linearly from 2 to 0 in the iterations.

Whereas the bubble-net method can be performed in two ways. The first is the shrinking encompassing in which, the value of a in equation (3.3) and A are decreased. The second is the spiral updating position which is applied to get inspired by the helix-shaped movement of Humpback whales around prey:

$$X(t+1) = D' \odot e^{bl} \odot \cos(2\pi l) X^*(t).$$

In which, $D' = |X^*(t) - X(t)|$ is defined as the distance between the whale and prey, b is a constant value used for specifying the logarithmic spiral shape, $\odot$ is an element-by-element multiplication, and l is a randomly generated value $\in [-1, 1]$.

The whales can swim around the victim through a shrinking circle and along a spiral-shaped path concurrently:

$$X(t+1) = \begin{cases} X^*(t) - A \odot D & if \quad p \geq 0.5 \\ D' \odot e^{bl} \odot \cos(2\pi l) + X^*(t) & if \quad p \leq 0.5. \end{cases}$$

In which, $p \in [0, 1]$ is a random value for describing the probability of taking either the shrinking encompassing method or the spiral model to set the whales position. In the discovery phase, the Humpback whales explore for prey with a random manner. The position of a whale is set by specifying a random search agent rather than the best search agent as below:

$$D = |C \odot X_{rand} - X(t)| \tag{3.5}$$

$$X(t+) = |X_{rand} - A \odot D|, \tag{3.6}$$

In which, X_{rand} is a position specified randomly among the current population. The first algorithm depicts the total structure of the WOA.

4. The proposed algorithm

Evolutionary algorithms have been widely used to solve complex optimization problems. Given that the resource allocation problem is an NP-complete problem, evolutionary algorithms can be used to find an optimal solution for that. In this paper, a new evolutionary technique called whale optimization algorithm is used to solve the resource allocation problem in cloud computing. The whale algorithm is inspired by the collective hunting way of a kind of Whale named Humpback Whale. In the following we will investigate the details of whale optimization algorithm to solve resource allocation problem.

4.1. Steps of the proposed algorithm

The proposed algorithm begins by creating a number of whales using 'whalecreat' function. Each whale represents a random solution for the scheduling problem. After this step, the fitness of each whale is calculated and the best solution is considered as the current optimal whale. Fitness function is equal to the total communication required in the resource allocation form of a whale. This function is named 'fitness' and explained in the following sections. After this step, whales begin to move. For each whale, the values of a, A, C, l, p are updated. A and C are constant coefficients. a is a descending number $\in [2, 0]$. p is a random number $\in [2, 0]$ and $l \in [2, 0]$. The use of these numbers is explained in the following sections.

One of the most important functions in the whale algorithm is the distance between two whales. Since the whale algorithm is designed for a continuous problems and the resource allocation problem is a discrete problem, this function needs to be rewritten. This function is used as 'distance' function in the algorithm. In total, three functions are designed for whale motion. The first one is "shrinking" function that reduces the distance between the current whale and the best whale. The second one is "spiral" function that simulates the current whale spin around the best whale. The third one is "searchprey" that moves a whale toward a random whale. The flowchart of the proposed algorithm is shown in Figure 1.

4.2. Whale creating

In this algorithm, each whale represents a solution to the problem of resource allocation. There are two types of nodes in the mentioned problem: gateway and resource. Each gateway is assigned a number of resources, and their information is sent just to this gateway. On the other hand, gateways are connected based on a specific topology in order to create the complete flow of information in the cloud. Due to the cost of data transmission between the gateways, their connection is created with the least edges and in the form of a spanning tree or a ring. Accordingly, each whale represents two sets of communications: edges that show the connection between gateways and edges that show the resources assigned to each gateway. A one-dimensional array w of size $k + n$ is used to display a whale, where k is the number of gateways and n is the number of resources. The first k entries of array w ($w[1.k]$) show the relationship between gateways. $w[i] = j$ means that gateway I sends its information to gateway j ($i \neq j$). The second n entries of array w ($w[k + 1.n]$) represent gateway of each resource. $w[p] = q$ indicates that resource $p - k$ sends its information to gateway q. Table 1 shows an example of how the whale is displayed in the proposed algorithm. There are 4 gateways and 7 resources in this example. The length of the array is equal to 11. To create a random whale, two parts of the array are filled individually. The first part is filled firstly.

If j is the random number generated for i^{th} cell, then $1 \leq j \leq k$ and $i \neq j$. Then next n cells are filled. For each generated random number like j, we have $1 \leq j \leq k$. In fact, each resource is assigned to one gateway.

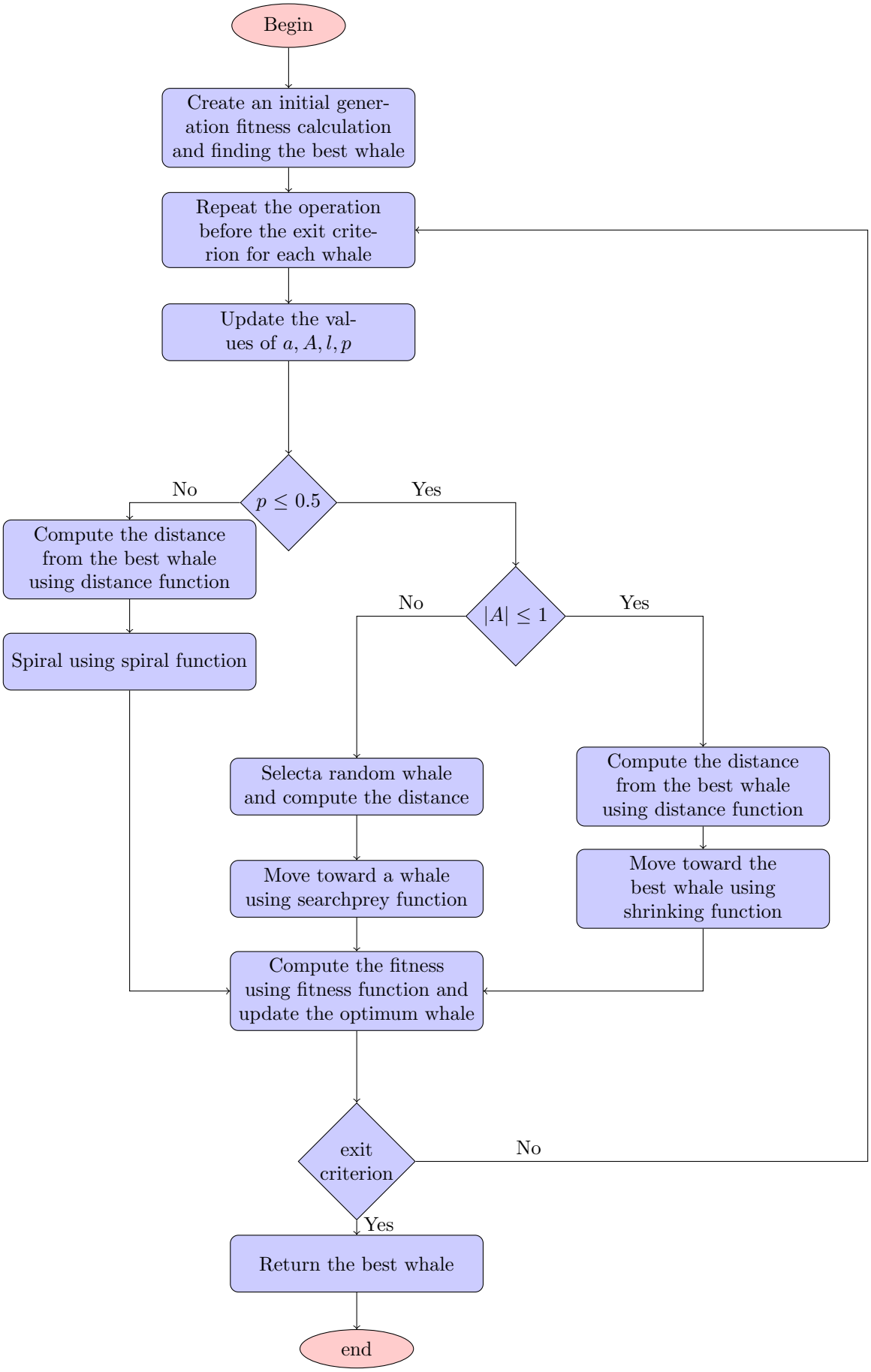

Figure 1: Flowchart of the proposed algorithm

Table 1: A whale with 4 gateways and 7 resources

	Gateway				*Resource*						
Indices	1	2	3	4	5(r_1)	6(r_2)	7(r_3)	8(r_4)	9(r_5)	10(r_6)	11(r_7)
allocate	3	1	4	3	1	4	4	4	3	2	1

4.3. Fitness function

In a cloud computing environment, it is assumed that all resource nodes must communicate with each other. Therefore, for each whale (or the solution of the recourse allocation problem) the total cost of network communications should be calculated. That is assumed that each resource sent a message to all other resources. The total cost of these messages are calculated and considered as whale fitness. The function named total cost and denoted as T_c. The proposed algorithm tries to minimize this function. Equation4.1 calculates T_c.

$$T_c = \frac{\sum_{j=1}^{|V_g|}(d_j^r \times d^g)}{p} \tag{4.1}$$

In which, $|V_g|$ is the total gateways, d_j^r is the total cost of transferring data between j^{th} gateway and all resources connected with it and d^g is the total cost of communication between gateways which is calculated using equation(4.2).

$$d^g = \sum_{j=1}^{|V_g|}\sum_{\substack{j=1 \\ j\neq i}}^{|V_g|} l_{ij} \tag{4.2}$$

In which, l_{ij} is the cost of communication between gateways i and j. d_j^r is calculated using equation (4.3).

$$d_j^r = \sum_{k=1}^{|V_g|} \varepsilon_{jk} \tag{4.3}$$

In which, ε_{jk} is the cost of communication between j^{th} gateway and all resources connected with it. is the number of connected resources to gateway j.

Another part of fitness function is penalty p which is considered for balancing in resource allocation. Given that, the proposed algorithm should reduce T_c , and p is the denominator of this function so $p = 0$ indicates penalty and $p = 1$ indicates normal mode. The value of p is calculated for all gateways. For gateways having resources more than $|V_r|/|V_g|$, $p = 0$ is mentioned to increase the value of T_c. Equation(4.4) calculates p.

$$p = 1 + \sum_{i=1}^{|V_g|} p_i \tag{4.4}$$

In which, $|V_g|$ is the total gateways and p_i is the penalty of each gateway which is calculated using equation (4.5).

$$p_i = \begin{cases} 1 & if \quad g_i^t \leq \varepsilon\frac{|V_r|}{|V_g|} \\ 0 & if \quad g_i^t > \varepsilon\frac{|V_r|}{|V_g|} \end{cases} \tag{4.5}$$

In which, g_i^t is the number of resources assigned to gateway i, $|V_r|$ is the number of resources and ε is a constant number.

4.4. Distance function

One of the most important functions used in the proposed algorithm, is the distance function which calculates the distance between two whales. Since the basic whale algorithm is a continues algorithm and resource allocation problem is a continuous problem, the concept of distance must be re-defined. All operators of the whale algorithm work based on the distance. So the precise design of the distance function will have a great effect on the performance of the proposed algorithm.

Each whale is equivalent to a resource allocation graph that has maximum $k+n$ edges. k is the number of gateways and n is the number of resources. The distance between two whales is defined as the number of non-common edges in two resource allocation graphs. Accordingly, the minimum distance is equal to zero and the maximum distance is equal to $k+n$. The pseudocode of the distance function is given in Algorithm 1.

Algorithm: distance W_1,W_2

$c = 0$;
for $(i = 1; i < k + n; i++)$
    if $(W_1(i) \neq W_2(j))$
        $c = c + 1$;
return$c$;

Below we show an example of how distance function works. There are 2 whales and 4 gateways in table 2. The distance between these two whales is calculated based on the corresponding cells of the array that do not have the same value. These cells are shown in red. Accordingly, the distance between two whales is 5.

Table 2: The distance between two whales

	Gateway				*Resource*						
Indices	1	2	3	4	5(r_1)	6(r_2)	7(r_3)	8(r_4)	9(r_5)	10(r_6)	11(r_7)
W_1	3	1	4	3	1	4	4	4	3	2	1
W_1	1	3	4	2	1	2	2	4	3	4	1

4.5. Spiral function

To determine the type of each whale motion, a randomly generated number p is in $[0, 1]$. If this number is greater than half, whale motion is performed using spiral function which means spiral around the best whale. The concept of spiral must be re-defined for the resource allocation problem. The pseudocode of the proposed spiral is given in Algorithm 2.

In the above pseudocode, W_i is a whale that should spiral and W_{best} is the best whale. $|change|$ denotes the number of entries of whale array that should be changed. If the value of $change$ is positive, then entries having non-equal values are changed using W_{best}. Otherwise, entries having equal values are changed. Table 3 shows an example of spiral function for $l = 0.125$ and $D = 5$ in which $change = \lfloor 5 \times 0.7 \rfloor = 3$. Three changed entries are in red. Green entries show the distance.

4.6. Shrinking function

At the beginning of each run of the algorithm, a random number p is generated. If this number if less than 0.5, another random number A should be considered. If A is lee than 1, shrinking function is performed. This function moves the current whale toward the best whale or prey. The difference between this function and spiral function is that, spiral function spirals around the prey

Algorithm: spiral W_i round W_{best}

$D = distance(W_i, W_{best})$;
l=random value in [-1,1];
$change = \lfloor D \times cos(2\pi l) \rfloor$;
for $(j = 1 : i < |change|; i++)$
    if $change > 0$
        set a random number in $W_i(k)$ where $W_i(k) = W_{best}(k)$;
    ifelse $change < 0$
        set a random number in $W_i(k)$ where $W_i(k) \neq W_{best}(k)$;
returnW_i;

Table 3: An example of spiral function

	Gateway				*Resource*						
Indices	1	2	3	4	5(r_1)	6(r_2)	7(r_3)	8(r_4)	9(r_5)	10(r_6)	11(r_7)
W_1	3	1	4	3	1	4	4	4	3	2	1
Best	3	3	4	2	1	2	2	4	3	4	1
Sprial	3	1	4	1	1	2	4	4	3	4	1

while shrinking function moves directly and faster toward the prey. Hence, increasing the probability of executing the searchprey function at the end of algorithm is desirable.

The shrinking function adjusts its movement toward the prey based on the distance from the best whale. After calculating the distance, a percentage of non-equal entries is changed and moves toward the best whale with the probability of 50%. The pseudo-code of the shrinking function is shown in Algorithm 3.

Algorithm: shrinking W_i to W_{best}

$D = distance(W_i, W_{best})$;
$change = \lfloor A \times D \rfloor$;
for $(j = 0; i < |change|; i++)$
    k= a random indice in W_i where $W_i(k) \neq W_{best}(k)$;
    rand= a random number in $[0, 1]$;
    if $(rand < 0.5)$
        $W_i(k) = W_{best}(k)$;
    else
        set a random number in $W_i(k)$;
returnW_i;

Below we show the shrinking function in an example. In table 4, the distance between W_1 and W_{best} is 4 which is shown in green. If $A = 0.5$, then two entries from entries having non-equal values are changed. One of them become the best whale and the other one had a random change. These two changed entries are shown in red.

4.7. Searchprey function

An important aspect of evolutionary algorithms is the ability to produce new solutions for the problem which prevents stuck in local optimum. In the proposed method, the searchprey function is used for this purpose. The higher probability of performing this function at the beginning is desirable. So that exploitation and optimization phase will start with a greater variety of solutions.

Table 4: An example shrinking function

	Gateway				*Resource*						
Indices	1	2	3	4	5(r_1)	6(r_2)	7(r_3)	8(r_4)	9(r_5)	10(r_6)	11(r_7)
W_1	3	1	4	3	1	4	4	4	3	2	1
Best	3	1	4	2	1	2	2	4	3	4	1
Shrinking	3	1	4	2	1	3	4	4	3	2	1

Searchprey function moves a whale toward a randomly selected whale. The basis of the movement is the distance between the two whales, but different methods have been used for motion as described in the following sections. The pseudocode of the searchprey function is given in Algorithm 4.

Four functions, shrinking, join, swap and randomwalk are used within this function. Shrinking function is explained in the previous section, with this difference that is used instead of . The other three functions are described below.

Algorithm: searchprey W_i

select W_j randomly from whale group;
$D = distance(W_i, W_j)$;
$change = \lfloor A \times D \rfloor$;
rand= a random number in $[0, 1]$;
if $(rand < 0.25)$
    $shirinking(W_i, W_j)$;
elseif $(rand < 0.5)$
    join $(W_i, W_j, change)$;
elseif $(rand < 0.75)$
    swap (W_i, W_j);
else
    random walk $(W_i, change)$;
returnW_i;

4.7.1. Join function

Join function changes some parts of W_i into the entries of W_j. The number of entries being changed is equal to $|change|$. Changing process can be sequential or random. After joining, whale W_i get in in new position, part of which is inherited from W_j.

Table 5 shows an example of performing join function. In this example, W_1 moves toward W_2. The distance is 4 shown in green. Changed entries are selected linearly at the end of the array. These entries are shown in red.

Table 5: An example of join function

	Gateway				*Resource*						
Indices	1	2	3	4	5(r_1)	6(r_2)	7(r_3)	8(r_4)	9(r_5)	10(r_6)	11(r_7)
W_1	3	1	4	3	1	4	4	4	3	2	1
W_2	3	1	4	2	1	2	4	4	2	4	1
join	3	1	4	3	1	3	4	4	2	4	1

4.7.2. Swap function

The swap function is the same as join function with this difference that the number of changed entries in join function is equal to $|change|$, while in swap function is exactly equal to the number of gateways ($|V_g|$) or the number of resources ($|V_r|$). In fact, in swap function, the new position of W_i includes connection sub-graph of gateways in W_i and connection sub-graph of resources in W_j (and vice versa). The value of $|change|$ has no effect on this function.

Table 6 shows an example of swap function, in which W_1 moves toward W_2. The distance is 4 and shown in green and also has no effect on the result. Changed entries belong to the first part of the array which show the type of connection between gateways. These entries are shown in red.

Table 6: An example of swap function

	Gateway				*Resource*						
Indices	1	2	3	4	5(r_1)	6(r_2)	7(r_3)	8(r_4)	9(r_5)	10(r_6)	11(r_7)
W_1	3	1	4	3	1	4	4	4	3	2	1
W_2	3	3	4	2	1	4	4	4	2	4	1
swap	3	3	4	2	1	4	4	4	3	2	1

4.7.3. Randomwalk function

Randomwalk function changes a number of entries of W_i randomly. The number of entries being changed is equal to $|change|$. The values of whale W_j have no effect on the generated numbers for W_i. The only effect of W_j in running this functin is the value of $|change|$ that is generated based on the distance between W_i and W_j.

In fact, randomwalk function produces a completely random movement in W_i and takes the whale to a new position. So it is likely to discover new solutions after performing this function.

4.8. Parameter a

One of the most important parameters in the proposed method is a. The value of A which determines the type of executing function in any steps of the algorithm is calculated based on parameter a according to equation (4.6).

$$A = 2ar - a \tag{4.6}$$

The value of a is considered between 2 to 0 in descending form. So, at the beginning of the algorithm it is likely that the value of A is greater than one and the searchprey function is executed. As the value of a decreases, the value of A also becomes less than one in most cases. Therefor shrinking function will be executed. In this way, after discovery new solutions, the algorithm enters the exploitation phase. In fact, decreasing thevalue of a helps to narrow the spiral movement around the prey and whales focus on optimizing the final solution.

5. Simulation result

This section provides the simulation of the proposed algorithm and then evaluates its performance. Simulation is done in MATLAB software environment on a desktop computer. Given that there is no processing time in the evaluations, it is not necessary to introduce the computer specifications and software version.

None of the prior researches on resource allocation problem in cloud computing hadn't used a common and public dataset. Creating a dataset covers all conditions of the scheduling problem.

Given that, data are not the same, we re-implemented some of them in order to compare with previous works.

5.1. Dataset

The test data are generated in small, medium and large size. The number of gateways varies from4 to 100 and the number of resources varies from 10 to 800. A total of eight test samples are designed for the proposed method. The list of sample tests is shown in table 7.

Table 7: Data set list

Data Set name	Number of Gateways	Number of Resources
DS_1	4	10
DS_2	4	12
DS_3	4	16
DS_4	40	100
DS_5	40	200
DS_6	40	400
DS_7	100	400
DS_8	100	800

In Table 7, three datasets DS_1, DS_2 and DS_3 are of small size, two datasets DS_4 and DS_5 are of medium size and datasets DS_7 and DS_8 are of large size. Regardless of scale, the cost of communication between resources and gateways is a random number $\in [1, 20]$ and the cost of communication between gateways is a random number $\in [20, 40]$. The cost of connection between gateways is always higher than the cost of connection between resources and gateways.

Each dataset has two matrices. If k is defined as the number of gateways and n denotes the number of resources, then one of these matrices is a symmetric matrix called $Gateway_{k\times k}$ of size $k \times k$ which shows the cost of communication between gateways and the other one is a matrix called of size which shows the cost of connection between resources and gateways. You can see the dataset DS_1. In Figure 2.

$$
\begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \end{array}
\begin{array}{c}
\begin{array}{cccc} G_1 & G_2 & G_3 & G_4 \end{array} \\
\left[\begin{array}{cccc} 0 & 24 & 40 & 40 \\ 24 & 0 & 39 & 32 \\ 40 & 39 & 0 & 25 \\ 40 & 32 & 25 & 0 \end{array}\right]
\end{array}
\quad (Gateway)
$$

$$
\begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \end{array}
\begin{array}{c}
\begin{array}{cccccccccc} R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 & R_8 & R_9 & R_{10} \end{array} \\
\left[\begin{array}{cccccccccc} 19 & 7 & 16 & 5 & 4 & 3 & 2 & 2 & 15 & 8 \\ 5 & 1 & 14 & 5 & 4 & 4 & 5 & 2 & 17 & 3 \\ 18 & 2 & 19 & 6 & 1 & 6 & 2 & 4 & 17 & 5 \\ 8 & 6 & 15 & 6 & 14 & 4 & 7 & 4 & 15 & 5 \end{array}\right]
\end{array}
\quad (Resource)
$$

Figure 2: Dataset DS_1

5.2. Simulation results

We applied the proposed algorithm on the generated dataset and in order to compare with other methods, we also applied two methods GA [30] and Search Economics for IOT Resource Allocation (SEIRA) [31] on this data. The main criterion is the comparison of different methods of resource allocation problem is T_c (the total cost of the messages in resource allocation).

Table 8 shows the results of the proposed algorithm in comparison with previous methods. Given the fact that data is randomly generated and the resource allocation problem is a np-complete problem, the optimal solution is not available and only the results of different methods can be used to compare them. For the results shown in Table 8, the proposed algorithm has been executed for 10000 and 20000 times. Other methods are also executed for the relatively same initial population and rounds in order to have fair results.

Table 8: Data set list

	GA [30]	SEIRA [31]	WOA (proposed)
DS_1	180.5	180.5	180.5
DS_2	214	214	214
DS_3	242.5	242.5	242.5
DS_4	841	835	836.6
DS_5	1260.4	1233	1230
DS_6	1920.6	1905.3	1901.4
DS_7	2728.8	2669.1	2661.8
DS_8	3857.3	3753.7	3725

The results of the simulation of the proposed algorithm and its comparison with the other two algorithms indicate that whale algorithm is good at solving resource allocation problem. Obtained results for whale algorithm are always better than GA. In comparison with SEIRA algorithm, whale algorithm has the same results and in some cases better results.

One criterion to evaluate the performance of the proposed algorithm is convergence speed. To do this, we save the fitness of the best whale in each round of whale motions. Figures 3, 4, and 5 show the fitness of three datasets DS_1, DS_5 and DS_8.

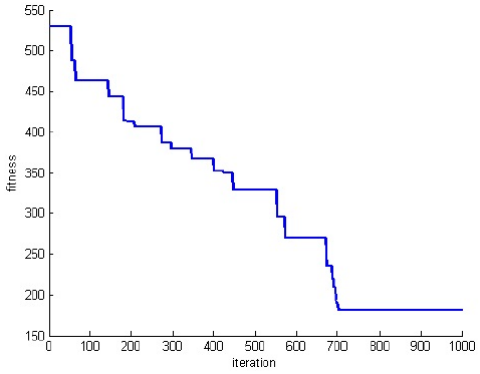

Figure 3: The fitness of the best whale during whales' motion in DS_1

The whale algorithm is inspired by the group haunting of whales. Hence, in each round of the proposed algorithm, all whales should approach the prey. From the resource allocation problem

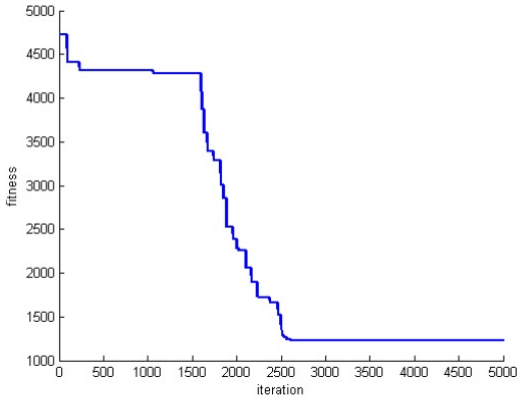

Figure 4: The fitness of the best whale during whales' motion in DS_5

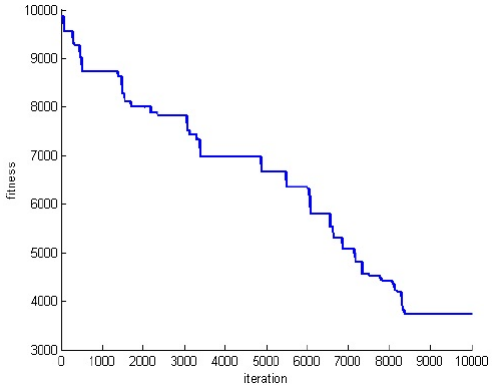

Figure 5: The fitness of the best whale during whales' motion in DS_8

solving point of view, it means that, all whales change over time. Figures 6, 7, and 8 show the fitness of all whales for three datasets DS_1, DS_5 and DS_8.

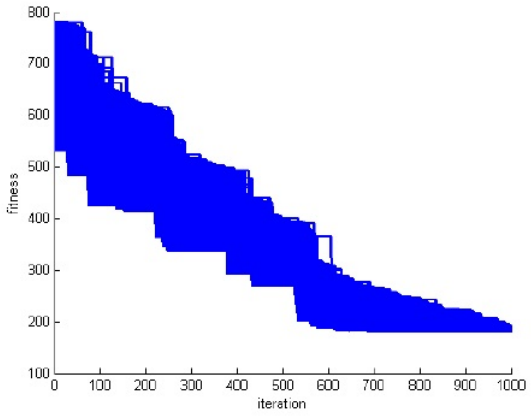

Figure 6: Whale fitness during motion in DS_1

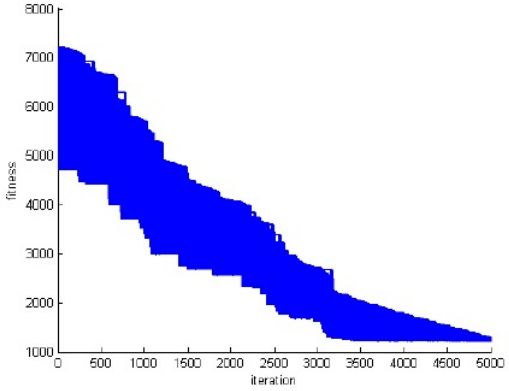

Figure 7: Whale fitness during motion in DS_5

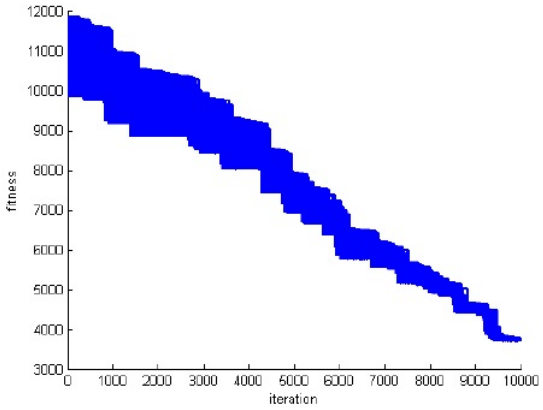

Figure 8: Whale fitness during motion in DS_8

5.3. The effect of spiral, shrinking and searchprey functions

One of the most important points to be considered in the proposed method is the efficiency of the designed functions. Three main functions named spiral, shrinking and searchprey functions have been designed in the proposed method. Spiral function is designed to spiral around the best solution (or prey). The shrinking function moves a whale directly toward the best whale and searchprey function moves a whale toward a random whale. The first and second functions are used to optimize the existing solutions and the third one is used to discover new solutions.

If the designed functions do not have the necessary performance, then the proposed algorithm has the same performance as a random algorithm which produces many random solutions for the problem. In contrast, the high efficiency of these functions makes it possible to quickly converge to an optimum solution. To investigate this issue, we ran the proposed algorithm with different numbers of whales and motions. In all runs, the multiplication of the number of whales in the number of steps is a constant number. That is, in all simulations, the same number of solutions is produced for the problem. The results of three datasets DS_1, DS_5 and DS_8 are shown in tables 9, 10, and 11 respectively.

The results of the three above tables indicate that, using three main functions to produce solution yields to better solutions than random ones. In contrast, If we increase the number of rounds and reduce the number of whales, we will not get good results. We must balance between the number of

Table 9: The best solution of DS_1 for different modes

Number of Whales	Number of rounds	Best solution
1000000	1	270.5
100000	10	233
10000	100	197.5
1000	1000	180.5

Table 10: The best solution of DS_5 for different modes

Number of Whales	Number of rounds	Best solution
25000000	1	1470.1
500000	50	1431.4
50000	500	1298.8
5000	5000	1230

Table 11: The best solution of DS_{18} for different modes

Number of Whales	Number of rounds	Best solution
100000000	1	4357
10000000	10	4124.5
1000000	100	4023.2
10000	10000	3725

whales and the number of rounds.

6. Conclusion

Cloud computing is a popular model for users to use cloud resources because of the "pay-per-use" model. But resource allocation problem is one of the most important challenges in this kind of system due to the high volume of resources and users' requests which attracted many researchers. Resource allocation and scheduling is a method for allocating resources to users and the main objective of a schedule is reducing runtime and allocating optimal resources to tasks. On one hand, improper use of resources leads to increased energy consumption and as a result environmental warming. Therefore task scheduling and resource allocation in large systems such as cloud computing are also important besides runtime and can't be ignored. In this paper a novel algorithm based on whale optimization is proposed with the aim of reducing runtime of tasks and allocating optimal resources to tasks. The proposed method suggested a discrete definition for the whale algorithm in order to optimize resource allocation problem in cloud environment. Each whale is designed in the form of an array. Based on this array, a new concept of distance is defined for distance function. This function is very useful for whale algorithm. Spiral function is designed for spiral movement and shrinking and searchprey functions are designed for direct movement. Obtained results show that the proposed whale algorithm has a high ability to solve the resource allocation problem in the cloud. The functions designed for base operators of the whale algorithm have been able to search the solution space of the problem well and then found reasonable solutions for the mentioned problem by optimizing discovered solutions. An important point in the correct execution of the algorithm is the logical adjustment of the value of the which enters the algorithm to the exploitation phase from discovery phase.

References

[1] Q. Zhang LC and B. Raouf, *Cloud computing: state-of-the-art and research challenges*, J. Int. Serv. Appl. 1 (2010) 7-–18.
[2] AA. Soror, UF. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis and S. Kamath, *Deploying database appliances in the cloud*, IEEE Data Eng. Bull. 32 (2009) 13–20.
[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris and A. Ho, *Xen and the art of virtualization*, Proc. Nineteenth ACM Symp. Oper. Syst. Princ. Bolton Landing, USA, (2003) 164–77.
[4] B. Furht and A. Escalante, *Handbook of Cloud Computing*, Springer, 2010.
[5] RP. Goldberg, *Survey of virtual machine research*, IEEE Comput. 7 (1974) 34–45.
[6] P. Rose, *Survey of system virtualization techniques*, Lisbon, Portugal: Theses (Electrical Engineering and Computer Science) MS non-thesis Research Papers (EECS), 2004.
[7] M. Malekloo, N. Kara and M. E. Barachi, *An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments*, Sustain. Comput. Inf. Syst. 17 (2018) 9–24.
[8] G. Baranwal, D. P. Vidyarthi, *Fair multi-attribute combinatorial double Auction model for resource allocation in cloud computing*, J. Syst. Software 108 (2015) 60–76.
[9] T. Ma, Y. Chu, L. Zhao and O. Ankhbayar, *Resource allocation and scheduling in cloud computing: Policy and algorithm*, IETE Tech. Rev. 31 (2014) 4–16.
[10] B. Shrimali and H. Patel, *Multi-objective optimization oriented policy for performance and energy efficient resource allocation in cloud environment*, J. King Saud Univ. Comput. Inf. Sci. 32(7) (2020) 860–869.
[11] P. Pradhan, P. K. Behera and B. N. B. Ray, *Modified round Robin algorithm for resource allocation in cloud computing*, Procedia Comput. Sci. 85 (2016) 878-890.
[12] W. Lin, J. Z. Wang, C. Liang and D. Qi, *A Threshold-based dynamic resource allocation scheme for cloud computing*, Procedia Engin. 23 (2011) 695–703.
[13] E. Kheiri, M. G. Arani, R. Kheiri and A. Taghizadeh, *An efficient approach based on genetic algorithm for multi-tenant resource allocation in SaaS applications*, Int. J. Software Engin. Appl. 10 (2016) 47–68.
[14] T. Jena and J. R. Mohanty, *GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing*, Arab. J. Sci. Engin. 43 (2018) 4115—4130.
[15] L. Boloni and D. Turgut, *Value of information based scheduling of cloud computing resources*, Future Gen. Comput. Syst. 71 (2017) 212–220.
[16] H. Hallawi, J. Mehnen and H. He, *Multi-capacity combinatorial ordering GA in application to cloud resources allocation and efficient virtual machines consolidation*, Future Gen. Comput. Syst. 69 (2017) 1–10.
[17] P. Samimi, Y. Teimouri and M. Mukhtar, *A combinatorial double auction resource allocation model in cloud computing*, Inf. Sci. 357 (2016) 201–216.
[18] Z. Li, T. Chu, I. V. Kolmanovsky, X. Yin and X. Yin, *Cloud resource allocation for cloud-based automotive applications*, Mech. 50 (2018) 356–365.
[19] S. T. Maguluri, R. Srikant and L. Ying, *Heavy traffic optimal resource allocation algorithms for cloud computing clusters*, Perf. Eval. 81 (2014) 20-39.
[20] T. Xiaoying, H. Dan, G. Yuchun and C. Changjia, *Dynamic resource allocation in cloud download service*, J. China Univer. Posts Telecom. 24 (2017) 53–59.
[21] A. B. A. Muthu and S. Enoch, *Optimized scheduling and resource allocation using evolutionary algorithms in cloud environment*, Int. J. Intel. Engin. Syst. 2017 (2017) 125-133, .
[22] S. S. SHEULY, S. Bankarusamy, S. Begum and M. Behnam, *Resource allocation in industrial cloud computing using artificial intelligence algorithms*, The 13th Scandinavian Conf. Artif. Intel. (2017) 1–9.
[23] M. Ficco, C. Esposito, F. Palmieri and A. Castiglione, *A coral-reefs and Game Theory-based approach for optimizing elastic cloud resource allocation*, Future Gen. Comput. Syst. 78 (2018) 343–352.
[24] W. HU, K. LI, J. XU and J. XU, *Cloud-computing-based resource allocation research on the perspective of improved ant colony algorithm*, Int. Conf. Comput. Sci. Mech. Autom. (2015) 76- 80.
[25] S. Mirjalili and A. Lewis, *The whale optimization algorithm*, Adv. Eng. Software 95 (2016) 51–67.
[26] M. Shojafar, M. Kardgar, A. R. Hosseinabadi, Sh. Shamshirband and A. Abraham, *TETS: A Genetic-based Scheduler in Cloud Computing to Decrease Energy and Makespan*, The 15th Int. Conf. Hybrid Intel. Syst. Chapter Advances in Intelligent Systems and Computing 420, Seoul, South Korea, Springer, 420 (2016) 103–115.
[27] A. R. Hosseinabadi, H. Siar, Sh. Shamshirband, M. Shojafar, M. H. Nizam and Md. Nasir, *Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in small and medium enterprises*, Ann. Oper. Res. 229(1) (2015) 451–474.
[28] Sh. Shamshirband, M. Shojafar, A. R. Hosseinabadi, M. Kardgar, M. H. Nizam Md. Nasir and R. Ahmad, *OSGA:*

genetic-based open-shop scheduling with consideration of machine maintenance in small and medium enterprises, Ann. Oper. Res. 229(1) (2015) 743–758.

[29] A. R. Hosseinabadi, A. B. Farahabadi, M. S. Rostami and A. F. Lateran, *Presentation of a new and beneficial method through problem solving timing of open shop by random algorithm gravitational emulation local search*, Int. J. Comput. Sci. 10 (2013) 745–752.

[30] M. Kim and I. Y. Ko, *An efficient resource allocation approach based on a genetic algorithm for composite services in IoT environments*, IEEE Int. Conf. Web Serv. 2015 543–550.

[31] C. W. Tsai, *SEIRA: An effective algorithm for IoT resource allocation problem*, Comput. Commun. 119 (2018) 156–166.

[32] E. B. Tirkolaee, A. Goli, M. Bakhshi and I. Mahdavi, *Robust multi-trip vehicle routing problem of perishable products with intermediate depots and time win-dows*, Numerical Algebra Cont. Optim. 7 (2017) 417–433.

[33] E. B. Tirkolaee, A. Alinaghian, A. R. Hosseinabadi, M. B. Sasi and A. K. Sangaiah, *An improved ant colony optimization for the multi-trip capacitated arc routing problem*, Comput. Elect. Engin. 77 (2019) 457–470.

[34] A. R. Hosseinabadi, N. S. H. Rostami, M. Kardgar, S. S. Mirkamali and A. Abraham, *A new efficient approach for solving the capacitated vehicle routing problem using the gravitational emulation local search algorithm*, Appl. Math. Model. 49 (2017) 663–679.