Dynamic Resource Allocation in Cloud Computing

Seyedmajid Mousavi[1], Amir Mosavi[2,3,4], Annamria R. Várkonyi-Kóczy[2,5], Gabor Fazekas[1]

[1]Faculty of Informatics, University of Debrecen, Kassai Str. 26, 4028 Debrecen, Hungary, {majid.mousavi & fazekas.gabor}@inf.unideb.hu

[2]Institute of Automation, Kandó Kálmán Faculty of Electrical Engineering, Óbuda University, Bécsi út 94-96, 1431 Budapest, Hungary, amir.mosavi@kvk.uni-obuda.hu, varkonyi-koczy@uni-obuda.hu

[3]Norwegian University of Science and Technology, Department of Computer Science, 7491 Trondheim, Norway

[4]Institute of Structural Mechanics, Bauhaus Universität-Weimar, Marienstraße 15, 99423 Weimar, Germany

[5]Department of Mathematics and Informatics, J. Selye University, Elektrarenska cesta 2, 945 01 Komarno, Slovakia

Abstract: Utilizing dynamic resource allocation for load balancing is considered as an important optimization process in cloud computing. In order to achieve maximum resource efficiency and scalability in a speedy manner this process is concerned with multiple objectives for an effective distribution of loads among virtual machines. In this realm, exploring new algorithms, as well as development of novel algorithms, is highly desired for technological advancement and continued progress in resource allocation application in cloud computing. Accordingly, this paper explores the application of two relatively new optimization algorithms and further proposes a hybrid algorithm for load balancing which can contribute well in maximizing the throughput of the cloud provider's network. The proposed algorithm is a hybrid of teaching-learning-based optimization algorithm (TLBO) and grey wolves optimization algorithm (GW). The hybrid algorithm performs more efficiently than utilizing every single one of these algorithms. Furthermore, it well balances the priorities and effectively considers load balancing based on time, cost, and avoidance of local optimum traps, which consequently leads to minimal amount of waiting time. To evaluate the effectiveness of the proposed algorithm, a comparison with the TLBO and GW algorithms is conducted and the experimental results are presented.

1 Introduction

A cloud is created from numerous physical machines. Each physical machine runs multiple virtual machines which are presented to the end-users, or so-called clients, as the computing resources. The architecture of virtual machines is based on physical computers with similar functionality. In cloud computing, a virtual machine is a guest program with software resources which works like a real physical computer [1]. Yet, high workload on virtual machines is one of the challenges of cloud computing in the allocation of virtual machines. The task, requested by a client, has to wait to be allocated to the work and the resources needed. This strategy is independent of the executive priority of the tasks. However, the client who owns the task may offer larger value for it to try to raise his/her priority and eventually may succeed in taking control over the resources needed. Users can consume services based on the service level agreement that defines their needs of quality of service (QoS) parameters [2]. Yet, the multipurpose nature of the scheduling in the cloud computing environment has made it extremely difficult to manage. Therefore, scheduling has to create a compromise between service quality costs to come up with a suitable service which belongs to a multi-objective optimization problems family [3]. Several methods are available for a multi-objective scheduling problem [4]. The current methods of allocation of resources, such as FIFO [1] and Round-Robin [5] which are used in the cloud, do unfair allocation regardless of priority between tasks.

Resource allocation in the cloud environment is utilized to achieve customer satisfaction with minimal processing time. Reducing the fees of leasing resources in addition to ensuring quality of service and improving throughput for trust and satisfaction of the service provider is considered as another objective. In dynamic scheduling, the basic idea is the request allocation at the time of implementation of programs. In addition to the cost estimation, in the static method, dynamic scheduling consists of two other main sections of system state estimation and decision-making [6]. To recap, clients are interested in having their tasks completed in the shortest possible time and at the minimum cost which cloud servers should receive. On the other hand, the cloud providers are interested to maximize the use of their resources and also to increase their profits. Obviously these two objectives are in conflict with each other and often they are not satisfied with the traditional methods of resource allocation and scheduling mechanisms available [7]. Yet the goal is to direct the resource allocation to be performed in a way that is acceptable to both the users and the suppliers.

Resource allocation is a technique that ensures the allocation to virtual machines when multiple applications need different resources of CPU and input/output memory [2]. In cloud computing there are two technical restrictions. Firstly, the capacity of the machines is physically limited; secondly, priorities for the implementation of the tasks should be in harmony with maximizing the efficiency of resources. Ultimately, the waiting time and the completion time are to be reduced, in order to decrease the cost of system implementation. Classical methods for achieving a fully optimized solution are very time-consuming and in some cases are impossible. Traditional approximate methods [5] are reported inconclusive and inaccurate for solving optimization problems and are often trapped in local optimum.

Virtual machines in distributed systems have different usage conditions including; the cost of utilizing them and also different processing power. The tasks initiated by users may also have a different amount of information. In addition, to assign any task on any machine, a preparation time between tasks is also considered. This time-delay, which varies in different resources, is considered negligible in this study. The most important problem in this process is the order process and how the placement of tasks on resources is conducted. In fact by increasing the productivity of resources, the response time can be reduced and, simultaneously, can improve the total cost for resource utilization and load balancing. The load-balancing index is calculated based on target variables. The target variables include:

- The time of completing the latest task among virtual machines
- The average cost paid by the user for use of the resources
- Efficiency caused by the impact of load balancing based on completion time and cost of doing them.

To conduct this study in the realm of cloud computing system development, a number of assumptions are made with the following characteristics. In these assumptions, the resource and virtual machine are considered as one entity.

- Tasks are independent.
- Distributed environment is heterogeneous and dynamic.
- All tasks must be done.
- Each task is performed only by one virtual machine.
- Everything is done exactly once.
- Each virtual machine has different and specified processing speed.
- Each virtual machine has a special price that must be paid for the use of it in time.

Traditional approximation and resource allocation methods (see, e.g. [7]) due to the multi-objective and dynamic nature of the problem and also difficulties in dealing with local optimum need advancement and major improvement. Consequently, the purpose of this paper is set to address the research gap in cloud computing. To do so a hybrid approximation algorithm for resource allocation is proposed. In this study the performance of two algorithms i.e. teaching-learning-based optimization (TLBO) [8] and grey wolves optimization (GW) [9], in comparison with the proposed algorithm, are discussed. These two algorithms are currently used as approximation algorithms for establishing load balancing, based on time and cost between resources and efficiency. The balance is established between three target assessment variables for evaluating the proposed approximation algorithm.

2 Related Works

For resource allocation in distributed scheduling, Xu et al. [5] present a non-dominated sorting genetic algorithm-based multi-objective method (NSGA-II) [10]. They aimed at minimizing the time and cost in load balancing using resources to achieve Pareto optimal front. They used self-adaptive crowding distance (SCD) to overcome the crowding distance. In addition, in their proposed method, a mutation operator is included in the traditional algorithm of NSGA-II to avoid premature convergence. In this method, the strategy that the algorithm uses for improving the efficiency and performance of the intersections, has not been effective, as the solutions are trapped in local optimum.

Salimi et al. [14] introduced a multi-objective tasks' scheduling using fuzzy systems and standard NSGA-II algorithms for distributed computing systems in [6]. The authors aimed at minimizing implementation time and costs while increasing the productivity of resources. Their study is associated with the load balancing in the distributed system. They use the indirect method and fuzzy systems and do implementation of the third objective function to solve this problem. However dealing with three objectives has not been efficiently done in their work. In [7], Cheng provides an optimized hierarchical resource allocation algorithm for workflows using a general heuristic algorithm. In this model, the main objective is the coordination between the tasks and duties assigned to the service. The purpose is to service in accordance with the operational needs to perform properly the tasks and observe the priority between them. This model accomplishes workflow tasks scheduling aimed at load balancing by(?) dividing the tasks to different levels. Further, mapping and allocation of each level of tasks to resources is directed according to the processing power.

Gomathi and Karthikey [8] introduce a method for assigning tasks in a distributed environment using Hybrid Particle Swarm Optimization algorithm (HybPSO) [11]. HybPSO is used to meet the user needs and increase the amount of load balancing with productivity. The goal is to minimize the task completion time among processors and create load balancing. This method assures that each task is assigned to exactly one processor. In this method, each solution is shown as a particle in the population; each particle is a vector with *n* dimension which is defined for scheduling *n* independent tasks.

In [9], authors introduce a heuristic method based on particle swarm algorithm [12] for tasks' scheduling on distributed environment resources. Their model considers the computational cost and the cost of data transfer. Their proposed algorithm optimizes dynamic mapping tasks to resources using classical particle swarm optimization algorithm and ultimately balances the system loads. This optimization method is composed of two components. One of them is the scheduling operations task and the other one is particle swarm algorithm (PSA) to obtain an optimal mix of the tasks to resources' mapping.

Table1 presents a summary of the related works done in the field of tasks' scheduling. This table includes the objectives of tasks' scheduling, the algorithms used in these methods, the simulation environment of the algorithms, and the year in which they were developed. Table1 does not include the GW and education-based learning algorithms and/or any variations of these algorithms.

Table1
Summary of the works done in the field of resources' allocation

Author	Evolutionary algorithm	Environment	Targets	Year	Simulation tool
Xue et al [13]	multi-target genetic	Cloud	• Reduce the longest termination time among resources • Reduce the resources cost • Load balancing	2014	Matlab
Salimi et al [14]	multi-target genetic	Grid	• Reduce the longest termination time among resources • Reduce the resources cost • Load balancing	2014	GridSim
Cheng [15]	genetic	Cloud	• Reduce the longest termination time among resources • Load balancing	2012	Java environment
Gomathi & Karthikey [8]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Load balancing	2013	Java environment
Pandey et al [17]	swarm optimization	Cloud	• Reduce costs associated with load balancing	2010	Amazon EC2
Wu et al [18]	swarm optimization	Grid	• Reduce the longest termination time among resources • Reduce the workflow time • Load balancing	2012	Ad-hoc VC++ toolkit
Izakian et al [19]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Reduce the workflow time • Load balancing	2010	Java environment
Banerjee et al [20]	Ant colony	Cloud	• Reduce the longest termination time among resources • Load balancing	2009	Simulated cloud
Mousavi & Fazekas [21]	Ant colony	Cloud	• Reduce the longest termination time among resources • Reduce the workflow time • Load balancing	2016	CloudSim
Ludwig & Moallem [22]	Ant colony	Grid	• Reduce the longest termination time among resources • Load balancing	2011	GridSim
Babu & Krishna [23]	Bee colony	Cloud	• Reduce the longest termination time among resources • Load balancing	2013	CloudSim
Zhao [24]	swarm optimization	Cloud	• Reduce the longest termination time among resources • Reduce the resources cost	2015	CloudSim
Abdullah & Othman [25]	Simulated Annealing	Cloud	• Reduce the longest termination time among resources • Load balancing	2014	CloudSim

The literature review shows that traditional methods which are used for optimization, may be definitive and accurate, yet they are often trapped in local optimum. In fact, due to the dynamic nature of distributed environment and heterogeneous resources, in such a system, the scheduling process must be done automatically and very quickly. That is why the scheduling process is recognized as an NP-complete problem [26]. Traditional approaches are not dynamic and suitable to solve such a scheduling problem. These approaches contain a large search space; facing a large number of possible solutions and a tedious process to find the optimal solution. There is currently no efficient method available to solve these problems. In such circumstances, the traditional approach has been set to find a fully optimized solution instead of finding the semi-optimal solution, but in a shorter time. In this context, IT professionals are focused on exploratory methods. Therefore, metaheuristic algorithms which have a global overview, as they ensure convergence to solution and do not fall into the trap in local optimum, are of importance. Consequently, the GW algorithm is chosen for this purpose. In addition, the TLBO algorithm is used in a hybrid form with GW to improve local optimization and increase accuracy.

3 Proposed Method

Methodology is based on bonding the algorithms of TLBO and GW. With such hybridization, it is aimed at speeding up the process while maintaining the improvement of local optimization and increasing the accuracy. In the following, the problem is described. Further TLBO and GW algorithms are introduced as the primary solutions to the described problem. The proposed methodology then emerges from bonding of two algorithms.

3.1 Description of the Problem

There is a distributed network in a cloud environment with resource systems $S_1,S_2,S_3,\dots,S_n$. The resources are ready to serve in the distributed network for various nodes. Different jobs are sent for the source systems by nodes. The overall goal of this system is that, an agreed scheduling on the resources' allocation is to be obtained to perform the jobs. Consequently, with the resources allocation, the load balancing will be increased [27]. Here the scheduler is responsible to allocate one or more jobs to artificial machines in a distributed system [28]. In other words, the agreement on job scheduling is done by the scheduler. The scheduler provides a scheduling for resource allocation [29].

Several jobs are allocated and processed in parallel with each other at time - t - in the distributed system. The number of variables T_k is permutation between jobs and resources, this variable is called P, and its value is calculated as follows:

$$P = n^m \quad \left(n \text{ is number of tasks and } m \text{ is the number of sources}\right) \qquad (1)$$

As it is described in Figure 1 each node includes several jobs. Each job requires a series of specific resources. The problem can be introduced as *Job* → $j_1,j_2,\dots,j_n$ *and Resource* → $R_1,R_2,\ \dots\ ,R_m$.

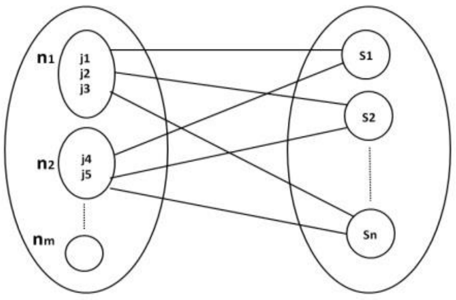

Figure 1
Resource allocation in a distributed environment

If in the particular example, the resources $R_1,R_2,\ \dots\ ,R_m$ have the same capacity and processing power and $j_1,j_2,\dots,j_n$ all need 1% of the processing. The advanced model can be defined in a form describing what jobs in which resources should be used in order to achieve the maximum load balancing, average response time, and minimum cost. For the exact solution of the problem, all possible allocation modes must be calculated and the best mode chosen. Due to the large number of modes (exponential), the problem is an example of set packing problems, which is of *NP*-complete type.
Optimization function is defined for resource *i* and job *j*. y_i is the number of resources (package). The objective function and mathematical programming model that should be optimized are as follows:

$$Min\ B = \ a*\left(1-\ L_{(y\,j)}\right)+b*C_{(y\,j)}+c*T_{(y\,j)} \tag{2}$$

S.t.

$$\sum_{i=1}^{n} w_i x_{i\,j} \leq Ky_j, \forall j \quad , \quad \sum_{j=1}^{n} x_{i\,j} \leq b_j, \forall j \qquad x_{i\,j}, y_i = 0,1 \ \ \forall i, j$$

Where :

$$x_j = \begin{cases} 1 & job\ j\ is\ used \\ 0 & job\ j\ isnot\ used \end{cases} \quad , \quad y_j = \begin{cases} 1 & resource\ j\ is\ used \\ 0 & resource\ j\ isnot\ used \end{cases}$$

$x_{i\,j}$ represents that job *j* is assigned to resource *i*. C is the maximum capacity for each resource. w_i represents the amount of job *i* that is covered by the resource. The aim is to find the minimum number of virtual machines - Y_j - that minimize the objective functions. The values of *L*, *C*, and *T* (load balancing, cost, and response time) are considered based on the number of virtual resources, Y_j, where a, b, c are variable based on cloud system. The variable of X_{ij} demonstrates that the i^{th} job is in j^{th} virtual machines, and if its value is equal to 0, it means that there is not any resource in j^{th} virtual machine and if its value is equal to 1, it means that there is enough resource to allocate the j^{th} virtual machine. Every job has the capacity of W_i. The first limitation indicates that total capacity of jobs can be placed at the maximum - K - available resources. The second limitation shows the maximum capacity of each virtual resource. b_j is the capacity of each virtual resource.

3.2 Grey Wolf Algorithm

Mirjalili et al. [30] introduce GW for solving engineering problems. GW is a new optimization algorithm inspired by behavior of grey wolves' hunting and their role hierarchies. The GW algorithm is benchmarked on 29 well-known test functions and the results on the unimodal functions show the superior exploitation of GW. Capability of exploration of new solutions in GW algorithm is confirmed by the results of multimodal function. The hierarchical structure and social behavior of wolves during the hunting process is modeled in the form of mathematical models and is used to design an optimization algorithm. The GW optimization algorithm emulates the hierarchical leadership and hunting mechanism of Grey Wolves in nature. Four types of GWs are considered as hierarchical structure in the social behavior of wolves for simulating, such as alpha, beta, delta, and omega. Furthermore, the three main steps of hunting - searching for prey, encircling prey, and attacking prey are implemented (Figure 2).

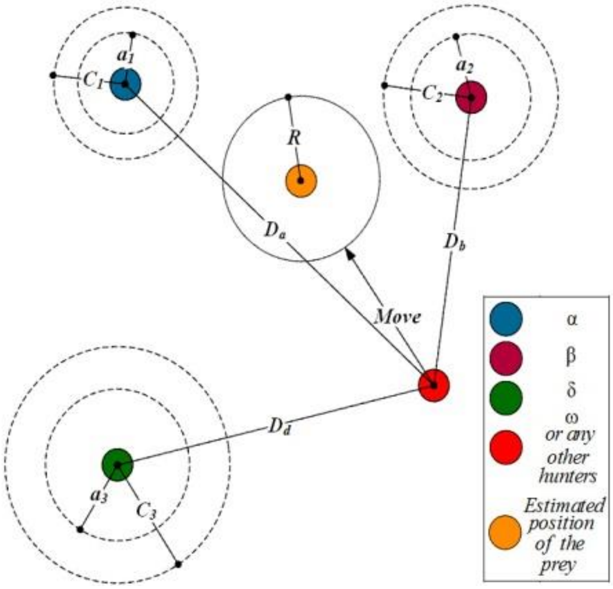

Figure 2
Grey wolves' motion in haunting [30]

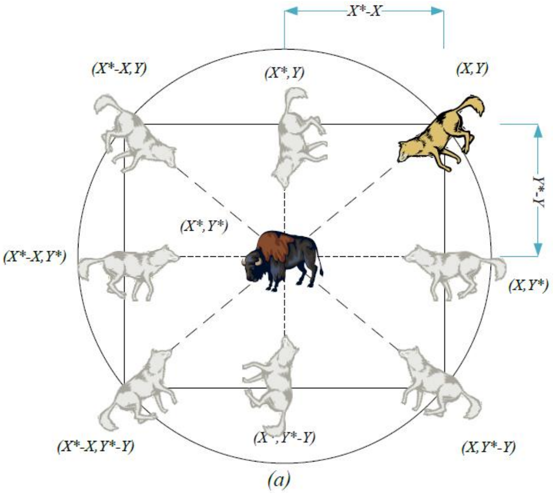

Figure 3

Updating wolves' position [30]

The Wolves’ leader is called alpha and is primarily responsible for the prey. The second level of wolves, which helps the leader, is called beta. The third level of wolves is called delta and is designed to support alpha and beta. The lowest level is called Omega [31]. In general, the algorithm steps can be summarized as follows:

- The fitness of all solution levels are computed and three top solutions are selected as Alpha, Beta, and Delta (Figure 2) until the end of the algorithm. In fact, Alpha is the best fitness of solution. After Alpha, the Beta and Delta are the best solutions respectively.
- In each iteration, the three top solutions e.g. Alpha, Beta, and Delta have the ability to estimate the prey position, conducting it in each iteration using the following equations [30]:

$$\vec{D}_\alpha = \left|\vec{C}_1 . \vec{X}_\alpha - \vec{X}\right|, \quad \vec{D}_\beta = \left|\vec{C}_2 . \vec{X}_\beta - \vec{X}\right|, \quad \vec{D}_\delta = \left|\vec{C}_3 . \vec{X}_\delta - \vec{X}\right| \tag{3}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad where \quad \left\{ \begin{array}{l} \vec{X}_1 = \vec{X}_a - \vec{A}_1.(\vec{D}_a) \\ \vec{X}_2 = \vec{X}_a - \vec{A}_2.(\vec{D}_a) \\ \vec{X}_3 = \vec{X}_a - \vec{A}_3.(\vec{D}_a) \end{array} \right\} \tag{4}$$

First, the wolves put a ring around the prey, where *Xp*, is the hunting position vector. *A* and *C* are hunting vector coefficients. *X* is the wolves' positions and t stands for the stage of each iteration. *D* indicates the behavior of putting the ring around the hunt [30]. In each iteration, after determining the positions of Alpha, Beta, and Delta, the other solutions are updated in compliance with them. Hunting information is defined by Alpha, Beta, and Delta. And the rest update their *x* positions accordingly. In each iteration, vector and consequently vectors b and c are updated. At the end of an iteration, Alpha wolf position is considered as the optimal point. This value is *A*. A value of *A* is an option value which is between (-2a, 2a). The absolute value of A is less than 1, so when the wolves are at the A distance from the prey, attack happens. At a distance of more than one, it is still(??) necessary that the wolves must converge toward each other [4]. In the GW algorithm, some main parameters like initial population size, vector coefficients, number of iterations, and the number of wolf levels are to be determined [32]. Then, the cost function of optimization which is minimized in this study is introduced. Afterward, the initial population is formed randomly and the fitness function is introduced. Then, in a loop on a regular basis, the position of the wolves' level is determined and the fitness function is calculated, and, using them, the new positions are calculated again. Iteration of this loop is specified according to the initial parameters.

3.3 Teaching-Learning-based Algorithm

Teaching-learning-based optimization algorithm (TLBO) [33] provides a novel approach to explore a problem space to find the optimal settings and parameters to satisfy the problem's objectives. The algorithm was introduced by Rao et al [33]. Similar to other evolutionary optimization techniques, TLBO algorithm is an algorithm derived from nature and works based on a teacher's teaching in a classroom. A teacher in the classroom, by expressing material, plays an important role in student learning and, if the teaching is effective, students learn the material better. In addition to the teacher factor, review of lessons by students would lead to better learning. This algorithm uses a total population of solutions to achieve the overall solution. A teacher tries to increase the level of students' knowledge by teaching and repeating the materials. Therefore the students can achieve a good score. In fact, a good teacher makes students closer to the level of his/her knowledge. The teacher is the most knowledgeable person of the class that shares his/her knowledge with the students. So the best solution (the best student of the class population) in the same iteration can act as a teacher. It should be considered that the students acquire knowledge based on the quality of teaching by the teacher and students' status (the average of class scores). This idea is the basis of Teaching-Learning-Based Optimization algorithm for solving optimization problems. The algorithm operates in two phases. The first phase is the teacher who shares his/her knowledge with students and the second phase is the review of courses by students in the same class. At the first stage, a teacher tries to improve scores of a class. In Figure 4, the Gaussian distribution function is used and the average scores acquired by students in the classroom is shown as *M*. *M* Parameter indicates the degree of the teacher's success in the classroom. In this figure, M_1 and M_2, respectively, show average scores of two separate classrooms with the same students.

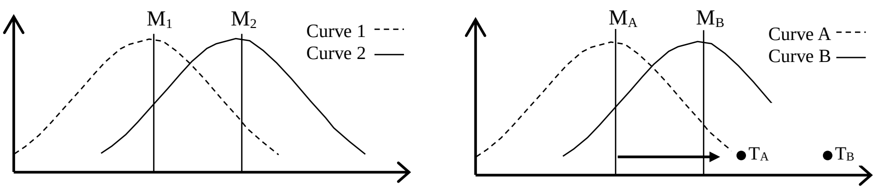

Figure 4
Average scores of students who were in classroom [33]

As it is shown in figure 4, the second teacher, with average scores of M2, has acted better than the first teacher with average score of M_1. T_A is the first grade teacher, who, with the best-case average scores of the first grade, M_A, moves to the T_A. It means that the academic level of students is approaching that of their teacher or equal with him/her. This creates a new population of the classroom which has shown an average of M_B and T_B. In fact, the students do not reach the knowledge level of the teachers, just close to it, which also depends on the level of classroom ability. Gaussian probability function is defined as follow [34, 36]:

$$f(X) = \frac{1}{s\sqrt{2p}} e^{\frac{-(x-m)^2}{2s^2}} \qquad (5)$$

In this formula, μ is the average score of students, which is shown as M_1 and M_2 in Figure 4.

3.4 Proposed Algorithm

Given more convergence power in the global optimality, the GW algorithm is used as a base algorithm in the proposed algorithm. This algorithm can also perform multi-objective optimization. The steps are as follows. In the initial state, a series of random numbers as the initial population are considered with uniform distribution and a basic solution is considered for the problem. Coefficients a, b, and c are initialized. Each solution is known as a wolf. In another word, each wolf is considered as a solution to the problem. These solutions or wolves have an answer. Wolves are divided into three categories; alpha, beta, and gamma. Yet on the basis of the fitness function, one of them gives a better answer to the fitness function. Then, the solution enters the main loop where after a few iterations the best solution for the fitness function is discovered. Based on the equations of the GW algorithm, the wolves' position is updated.

According to the first class of wolves, the new positions are fitted. Later on, more values for the probability of solution are considered. Correspondingly, the values of beta and gamma classes, the new positions of wolves, and their classifications can be obtained. If a suitable solution is found in the new classification, the algorithm is to be improved further. The best solution between the wolves is considered as the initial solution (initial population) for the teaching and learning algorithm. Further, the problem of the teaching and learning algorithm is solved and the solution is considered as initial population to start again. In this stage the GW algorithm is implemented. If there is no improvement in GW algorithm, according to the teaching and learning, it tries to find a better solution. If the solution is trapped in local optimum, an algorithm based on teaching and learning can introduce the new area of space based on training, which may improve solution. Since the accuracy of GW algorithm in the local behaviour is high, after each stage the position of wolves is determined. This position can be improved by learning and training algorithms, and GW algorithm is implemented again. This process increases the accuracy of GW algorithm. It should be considered that in the GW algorithm, every wolf represents a solution in the solution space. The best and the most successful solution will be chosen among them at any stage according to the position of other wolves. The best solution of the GW is defined as the initial solution for teaching and learning. After learning and training, its output is implemented as the initial solution for the next iteration in the GW algorithm. The proposed algorithm is presented in table 2:

Table 2
Pseudo code of the proposed algorithm

```
Initialize the grey wolf population Xi=(i=1,2,...,n)
Initialize a,b and c
Calculate the fitness of each search agent
X1=the best Search gent
X2=the second best Search Agent
X3=the third best Search agent
While t<Max number of iterations)
    For each search agent
      Update the position of the current search agent by equation
      End for
Calculate the fitness of all search agents
Update X1,X2,X3
t=t+1
If not improve solution
  Begin
     sol_wolf=Solution_grey_wolf
           Initialize sol_wolf for initialize_solution for TLBO
          Sol_TLBO=Do TLBO with Initialize Population with sol_wolf
          Intialize the grey wolf population Xi= Sol_TLBO, Initialize a,b and c
          Calculate the fitness of each search agent
X1=the best Search agent, X2=the second best Search agent, X3=the third
best Search agent
  end
end while
return X1
```

The main advantage of this algorithm is that if there was no improvement in grey wolf algorithm, according to the teaching-learning process, we try to find a better solution. If the problem is stuck in local optimum, teaching-learning process can introduce the new area of space based on training phase, which may improve the solution. Because of the accuracy of grey wolf algorithm in the local behavior (defect of the grey wolf algorithm), after each iteration, the position of wolves is updated. These positions can be improved by the teaching-learning algorithm, and then grey wolf algorithm is repeated again. This process increases the accuracy of grey wolf algorithm. Figure 5 illustrates flow diagram of proposed algorithm.

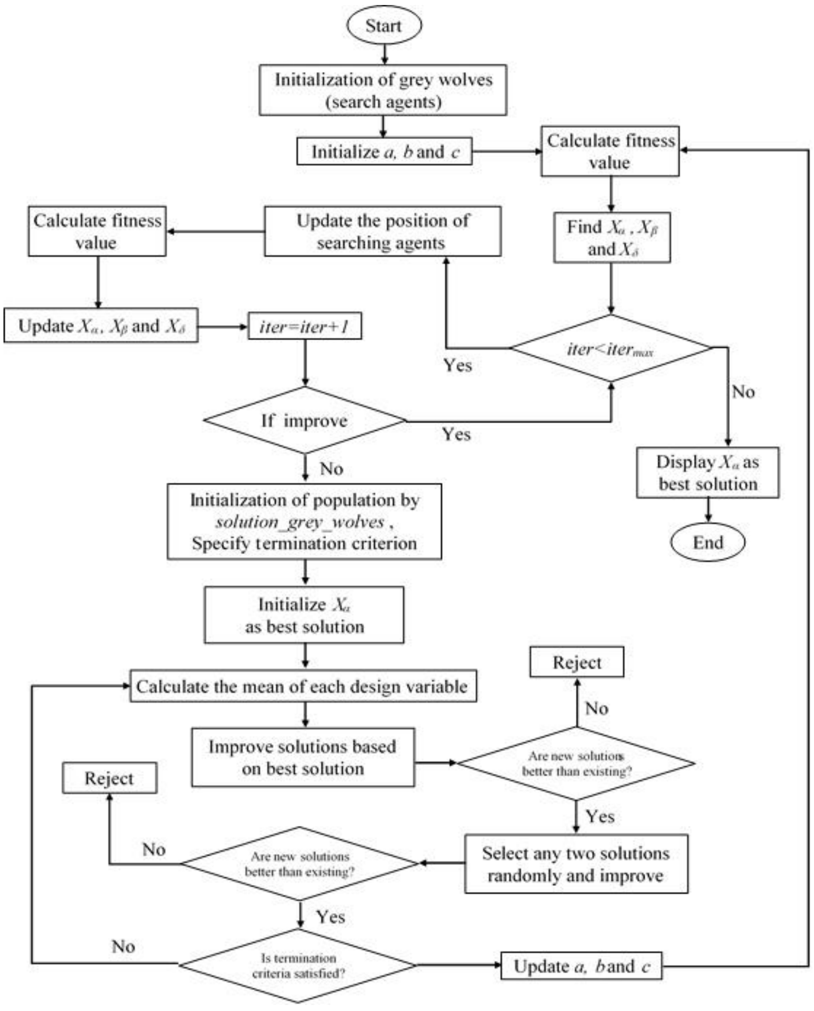
Start
Initialization of grey wolves (search agents)
Initialize a, b and c
Calculate fitness value
Find X_α, X_β and X_δ
Update the position of searching agents
Calculate fitness value
Update X_α, X_β and X_δ
iter=iter+1
iter<iter_max
Yes
No
If improve
Yes
No
Display X_α as best solution
End
Initialization of population by solution_grey_wolves, Specify termination criterion
Initialize X_α as best solution
Calculate the mean of each design variable
Improve solutions based on best solution
Are new solutions better than existing?
Reject
No
Yes
Select any two solutions randomly and improve
Are new solutions better than existing?
No
Reject
Yes
Is termination criteria satisfied?
No
Update a, b and c

Figure 5

Flow diagram of proposed algorithm

4 Simulation

According to the assessment index, load imbalance and the number of resources allocated are compared with each other. In this problem, Matlab is used to simulate the proposed method and set a packing problem which is considered as a model of resource allocation in cloud computing. Packages in the set packing are considered as requests that are processed in the cloud virtual servers.

Amount of efficiency of each resource - $Ri_{efficiency}$ - is equivalent to what percentage of resource has been used compared to the total resource [28, 30, 35]. Formally, the coefficient of variation of resource efficiency is called lack of load balancing. This variable indicates to what extent, there is a deviation of productivity. According to statistical indicators, if this variable is zero, it means that absolutely all resources are used. This variable is equal to the quotient of productivity of standard deviation in resources when there are a number of resources. When the variable is close to zero, load balancing is done better. Standard deviation is:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Ri_{efficiency} - mean\,R_i)^2} \qquad (6)$$

Load balancing factor *(Flb)* and load imbalance factor *(NFlb)* are introduced using the following equations:

$$NFlb = \frac{\sigma . R_i}{n} \quad i = 1 \ldots n \qquad (7)$$

$$Flb = \frac{C - \sigma}{n} \qquad (8)$$

Where *C* is the capacity of each resource or virtual machine.

Teaching-Learning simulation parameters have been set as follows:

maxIt = 500; *% Maximum Number of Iterations*
nPop = 250; %for each test must be updated Population Size

maxIt is the number of iterations for TLBO algorithm. *nPop* is the proportion of the number of packages in an appliance packaging problem.

Grey Wolf simulation parameters have been set as follows:

max_iteration=500
Dim = number of your variables
a=0.0354, b=38.3055, c=1243.531

Coefficients a, b, c, have been selected according to the resource. *Dim* is the number of packets in the packaging of appliance problem and *max_iteration* is the number of iterations for the algorithm.

5 Experimental Results

There are 60 packages in the dataset binpack5 with the capacity of 100. Each package has different value. The best way to obtain optimal solution is dividing the sum of the values of packages by the capacity of 100. The optimal solution of the allocation is 20 boxes. Our proposed method has achieved an approximate solution of 23. The solution has acted better than the GW and TLBO algorithms alone, as is shown in Table 3.

Table 3

Comparison of allocation with the proposed algorithm in dataset binpack5

Dataset	GW	TLBO	Hybrid method	Optimal result
First part of binpck5	24	26	23	20

In Table 4, allocation in dataset boxes binpack5 is shown. Each box is shown as a bin. Each bin indicates the packages with weights. For example bin1 contains packages weighing 49, 5, 47 and 4. Other boxes are allocated different packages in the same manner.

Table 4

Allocated boxes for proposed method

bin1: 49.5,47.4,	bin7: 41,39.5,	bin13: 35,35,29.9,	bin19: 27.2,26.9,26.9,
bin2: 47.3,47.2,	bin8: 37.2,37,25.5,	bin14: 34.7,32,31.5,	bin20: 26.8,26.2,26.1,
bin3: 46.6,45,	bin9: 36.6,36.6,26.3,	bin15: 30.7,30.3,29.8,	bin21: 25.9,25.8,25.4,
bin4: 44.5,44.4,	bin10: 36.6,36.3,27.1,	bin16: 29.8,28.8,28.7,	bin22: 25.2,25.2,25.2,
bin5: 43.9,43,	bin11: 36.1,35.7,27.5,	bin17: 28.3,27.5,27.4,	bin23: 25.1,
bin6: 41.9,41.4,	bin12 35.5,35.1,29.2,	bin18: 27.3,27.3,27.2,	

In the second experiment, a set of binpacks was used. Hence, according to data dispersion and increased capacity of boxes, the problem became more difficult, but the results demonstrate that the proposed method is desirable. Figure 6 indicates this process. A comparison between the differences of optimum solution show a high performance.

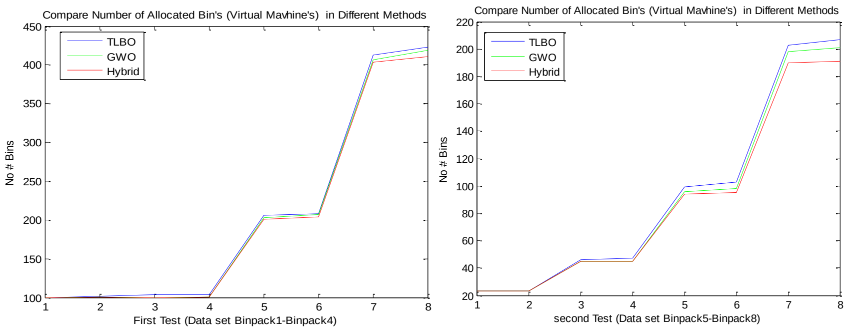

Figure 6

Comparison of the hybrid method with other methods to achieve optimal solution

According to a recent article [36], CGA-CGT and HI-HB methods are providing the best solution to set packing. The proposed hybrid method indicates a better performance than the other two methods (Table 5).

Table 5
Comparison of the proposed method with HI_BP and CGA-CGT

Experiment	Dataset	Number of Job	Resource capacity	HI_BP	CGA-CGT	Hybrid	Optimal solution
First	Binpack1-U250_00	250	150	100	100	100	99
First	Binpack1-U250_01	250	150	101	101	101	100
First	Binpack2-U250_00	250	150	100	100	100	99
First	Binpack2-U250_01	250	150	101	101	101	100
First	Binpack3-U500_00	500	150	204	201	201	198
First	Binpack3-U500_01	500	150	204	204	204	201
First	Binpack4-U1000_0	1000	150	404	404	403	399
First	Binpack4-u1000_1	1000	150	414	413	411	406
Second	Binpack5-T60_00	60	100	23	23	23	20
Second	Binpack5-T60_01	60	100	23	23	23	20
Second	Binpack6-T120_00	120	100	45	45	45	40
Second	Binpack6-T120_01	120	100	47	45	45	40
Second	Binpack7-T249_00	249	100	96	94	94	83
Second	Binpack7-T249_01	249	100	101	97	95	83
Second	Binpack8-T501_00	501	100	202	194	190	167
Second	Binpack8-T501_01	501	100	204	199	191	167

Figure 7 demonstrates the difference between the proposed approximate solution and the optimal solution. The difference is within acceptable limits. When data is increased, the proposed hybrid method has a better performance than the other two methods.

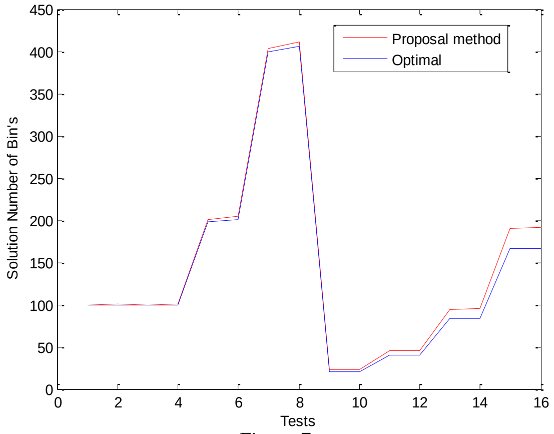

Figure 7
Solution's difference between the proposed method and the optimal solution

Nflb index indicates lack of load balancing. Decreasing this index demonstrates the increasing of load balancing in the cloud system. Increased load imbalance further indicates that maximum resource capacity is used. This means that the capacity of the resources is low. Therefore by increasing data, load balancing is performed better in the proposed algorithm. Figure 8 shows the load imbalance between the methods in the first and second experimental set. The load imbalance index acts in a reverse manner compared with the load balancing index. Increasing of data will increase the performance of load balancing. The training learning method, due to the structure of incorrect understanding of training in the experiments, doesn't have a good load balancing with more data. The proposed method shows a good performance with increased data in load balancing. The load balancing can be calculated by the lack of load balancing. Load balancing can indicate appropriate allocation. The second experiment in Table 6 confirms the fifth to eighth data set, and then related charts are presented in Figure 8.

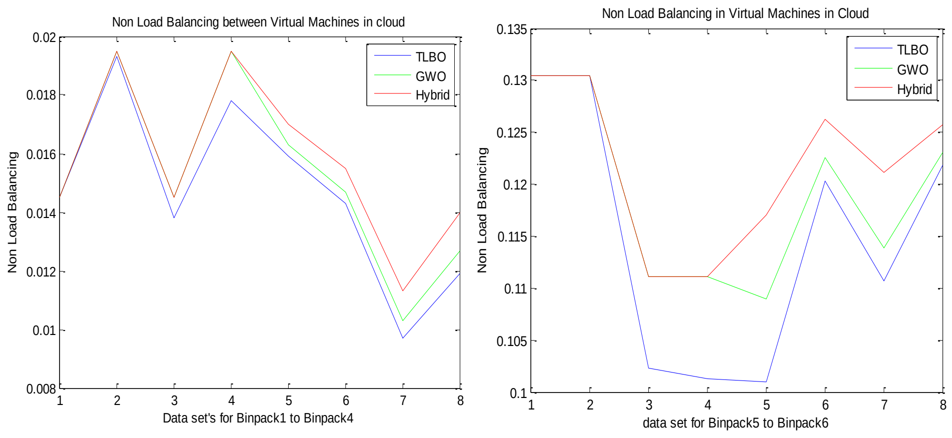

Figure 8

Load imbalance in the first test set (left) and load imbalance in the second test set (right)

Table 6

Comparison of the load imbalance between the proposed algorithm and other algorithms

Experiment	Dataset	Number of Job	Resource capacity	Nflb-TLBO	Nflb-GWO	Nflb-Proposed method
First	Binpack1-U250_00	250	150	0.0145	0.0145	0.0145
First	Binpack1-U250_01	250	150	0.0193	0.0195	0.0195
First	Binpack2-U250_00	250	150	0.0138	0.0145	0.0145
First	Binpack2-U250_01	250	150	0.0178	0.0195	0.0195
First	Binpack3-U500_00	500	150	0.0159	0.0163	0.0170
First	Binpack3-U500_01	500	150	0.0143	0.0147	0.0155
First	Binpack4-U1000_00	1000	150	0.0097	0.0103	0.0113
First	Binpack4-u1000_01	1000	150	0.0119	0.0127	0.140
Second	Binpack5-T60_00	60	100	0.1304	0.1304	0.1304
Second	Binpack5-T60_01	60	100	0.1304	0.1304	0.1304
Second	Binpack6-T120_00	120	100	0.1023	0.1111	0.1111
Second	Binpack6-T120_01	120	100	0.1013	0.1111	0.1111
Second	Binpack7-T249_00	249	100	0.1010	0.1090	0.1170
Second	Binpack7-T249_01	249	100	0.1203	0.1226	0.1263
Second	Binpack8-T501_00	501	100	0.1107	0.1139	0.1211
Second	Binpack8-T501_01	501	100	0.1219	0.1231	0.1257

Variable of relative changes percentage in comparison with the best answer can demonstrate the accuracy of the algorithm. Therefore, we calculate Robust Parameter Design *(RPD)* parameter, which is normalized change percentage against the best answer where $f_{heuristic}$ is metaheuristic value and $f_{optimal}$ is optimal value. The *RPD* equation is:

$$RPD = 100 \times \frac{f_{heuristic} - f_{optimal}}{f_{optimal}} \qquad (9)$$

Figure 9 indicates improvements of the relative changes' percentage. With increasing jobs this performance is observed to be stable in the proposed method.

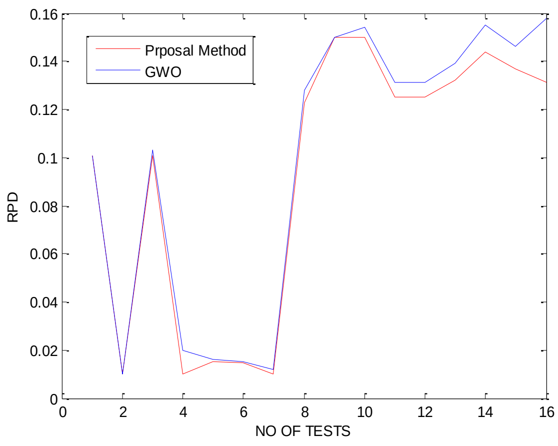

Figure 9

Relative changes' percentage of the optimal solution

This criterion shows the performance of the algorithm with increase of the jobs and indicates that the proposed method has appropriate performance with increasing data. Table 7 demonstrates relative changes' percentage in proposed algorithm. The average of relative changes' percentage is increased in an appropriate way in experimental data and has a better flow than GW and TLBO algorithms.

Table 7

RPD index in performance of proposed method

Experiment	Dataset	Number of Job	Resource capacity	Hybrid	Optimal solution	RPD
First	Binpack1-U250_00	250	150	100	99	1/99
First	Binpack1-U250_01	250	150	101	100	1/100
First	Binpack2-U250_00	250	150	100	99	1/99
First	Binpack2-U250_01	250	150	101	100	1/100
First	Binpack3-U500_00	500	150	201	198	3/198
First	Binpack3-U500_01	500	150	204	201	3/201
First	Binpack4-U1000_00	1000	150	403	399	4/399
First	Binpack4-u1000_01	1000	150	411	406	5/406

Second	Binpack5-T60_00	60	100	23	20	3/20
Second	Binpack5-T60_01	60	100	23	20	3/20
Second	Binpack6-T120_00	120	100	45	40	5/40
Second	Binpack6-T120_01	120	100	45	40	5/40
Second	Binpack7-T249_00	249	100	94	83	11/83
Second	Binpack8-T501_00	501	100	190	167	23/167
Second	Binpack8-T501_01	501	100	191	167	22/167

Conclusions

The performance of two relatively new optimization algorithms, i.e., TLBO and GW, along with a hybrid form of these two algorithms in dynamic resource allocation is described. To evaluate the performance of the proposed hybrid algorithm, a comparison with the TLBO and GW algorithms is conducted and the experimental results presented. It is reported that the proposed hybrid algorithm performs more efficiently than utilizing only one of these algorithms. It is further concluded that the main problem in the resource allocation of cloud scheduler is the lack of convergence in the optimal solution. Optimization of objective functions for the resource allocation at any time is one of the main problems in dynamic resource allocation. The evaluation of experimental results indicate that the proposed hybrid approach in high-volume data for resource allocation in cloud scheduler has better performance than the other two methods.

Acknowledgment

This work has been sponsored by Hungarian National Scientific Fund under contract OTKA 105846 and Research & Development Operational Program for the project “Modernization and Improvement of Technical Infrastructure for Research & Development of J. Selye University in the Fields of Nanotechnology and Intelligent Space”, ITMS 26210120042, co-funded by the European Regional Development Fund.

References

[1] J. Yao and H. Ju-Hou: Load Balancing Strategy of Cloud Computing Based on Artificial Bee Algorithm, Information Management, Vol. 51, 2012, pp. 185-189

[2] S. Zhao, X. Lu, and X. Li: Quality of Service-based Particle Swarm Optimization Scheduling in cloud Computing, Networks, Vol. 12, 2015, pp. 235-242

[3] A. Mosavi and A. Vaezipour: Reactive Search Optimization; Application to Multiobjective Optimization Problems, Applied Mathematics, Vol. 3, 2012, pp. 1572-1582

[4] S. Zhang and Y. Zhou: Grey Wolf Optimizer Based on Powell Local Optimization Method for Clustering Analysis, Discrete Dynamics in Nature and Society, Vol. 3, 2015

[5] U. Ayesta, M. Erausquin, E. Ferreira, and P. Jacko: Optimal Dynamic Resource Allocation to Prevent Defaults, Operations Research, Vol. 6, 2016, pp. 451-456

[6] A. Abur, and A. G. Exposito: Power System State Estimation: Theory and Implementation. CRC press, 2004

[7] A. Khetan, V. Bhushan, and S. Gupta: A Novel Survey on Load Balancing in Cloud Computing, Journal of Engineering & Technology, Vol. 6, 2013, pp.1-9

[8] B. Gomathi and K. Karthikeyan: Task Scheduling Algorithm Based on Hybrid Particle Swarm Optimization in Cloud Computing Environment, Journal of Theoretical and Applied Information Technology, Vol. 89, 2013, pp. 33-38

[9] E. Emary, Hossam M. Zawbaa, Crina Grosan, and Abul Ella Hassenian: Feature Subset Selection Approach by Grey-Wolf Optimization. Industrial Advancement, Springer International Publishing, Vol. 63. 2015 pp. 1-13

[10] A. Mosavi: Multiple Criteria Decision-Making Preprocessing Using Datamining Tools. International Journal of Computer Science Issues, Vol. 7, 2010, pp. 26-34

[11] P-Y. Yin, S-S. Yu, P-.P Wang, and Y-T. Wang: A Hybrid Particle Swarm Optimization Algorithm for Optimal Task Assignment in Distributed Systems. Computer Standards & Interfaces, Vol. 28, 2006, pp. 441-450

[12] R. C. Eberhart and J. Kennedy: A New Optimizer Using Particle Swarm Theory. Micro Machine and Human Science, Vol. 1, 1995, pp. 39-43

[13] S. Xue: An Improved Algorithm Based on NSGA-II for Cloud PDTs Scheduling, Journal of Software, Vol. 6, 2014, pp. 443-450

[14] R. Salimi, H. Motameni, and H. Omranpour: Task Scheduling Using NSGA II with Fuzzy Adaptive Operators for Computational Grids, Journal of Parallel and Distributed Computing, Vol. 74, 2014, pp. 2333-2350

[15] B. f Cheng: Hierarchical Cloud Service Workflow Scheduling Optimization Schema using Heuristic Generic Algorithm, Telecommunications, Vol. 15, 2012, pp. 92-95

[16] R. V. Rao, V. J. Savsani, and D. P. Vakharia: Teaching–Learning-based Optimization: an Optimization Method for Continuous Non-Linear Large Scale Problems, Journal of Information Sciences, Vol. 7, 2012, pp.1-15

[17] S. Pandey, L. Wu, S. Guru, and R. Buyya: A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, Information Networking and Applications, 2010, pp. 400-407

[18] L. Wu: A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling, Faculty of Information and Communication Technologies Swinburne University of Technology, Melbourne, Australia, 2012, pp. 1-5

[19] H. Izakian, B. Ladani, A. Abraham, and V. Snasel: A Discrete Particle Swarm Optimization Approach for Grid Job Scheduling, International Journal of Innovative Computing, Information and Control, Vol. 16 2014, pp. 4219-4252

[20] S. Banerjee, I. Mukherjee, and P. Mahanti: Cloud Computing Initiative using Modified ant Colony Framework, World Academy of Science, Engineering and Technology, Vol. 12, 2009, pp. 200-203

[21] SM. Mousavi and G. Fazekas: A Novel Algorithm for Load Balancing using HBA and ACO in Cloud Computing Environment, International Journal of Computer Science and Information Security, Vol. 16, 2016, pp. 48-52

[22] S. Ludwig and A. Moallem: Swarm Intelligence Approaches for Grid Load Balancing. J Grid Computing, Vol. 8, 2013, pp. 279-301

[23] LD. Dhinesh Babu and PV. Krishna: Honey Bee behavior Inspired Load Balancing of Tasks in Cloud Computing Environments, Science Direct, Applied Soft Computing, Vol. 13, 2013, pp. 2292-2303

[24] S. Zhao, X. Lu, and X. Li: Quality of Service-based Particle Swarm Optimization in Cloud Computing, Computer Engineering and Networks, 2015, pp. 235-242

[25] M. Abdullah and M. Othman: Simullated Annealing Approach to Cost-based Multi-Quality of Service Job Scheduling in Cloud Computing Environment, American Journal of Applied Sciences, Vol. 18, 2014, pp. 872-877

[26] MR. Garey, DS. Johnson, and L. Stockmeyer: Some Simplified NP-Complete Graph Problems. Theoretical Computer Science, Vol. 1, 1976, pp. 237-267

[27] Z. Bo, G. Ji, and A. Jieqing: Cloud Loading Balance Algorithm, Proceedings of IEEE 2nd International Conference on Information Science and Engineering, China, 2010, pp. 5001-5004

[28] M. Grabowski, C. Rizzo, and T. Graig: Data Challenges in Dynamic, Large-Scale Resource Allocation in Remote Regions, Safety Science, Vol. 34, 2013, pp.76-86

[29] D. Bertsimas, S. Gupta, and G. Lulli: Dynamic Resource Allocation: A Flexible and Tractable Modeling Framework, European Journal of Operational Research, Vol. 23, 2014, pp. 14-26

[30] S. A. Mirjalili, S. M. Mirjalili, and A. Lewis: Grey Wolf Optimizer, Advances in Engineering Software, Vol. 69, 2014, pp. 46-61

[31] M. H. Suleiman, Z. Mustafa, and M. R. Mohmed: Grey Wolf optimizer for Solving Economic Dispatch Problem with Valve-Loading Effects, APRN Journal of Engineering and Applied Sciences, Vol. 2, 2015, pp. 1619-1628

[32] C. Selvaraj: A Survey on Application of Bio-Inspired Algorithms, International Journal of Computer Science, Vol. 11, 2014, pp. 366-370

[33] R. V. Rao, V. J. Savsani, and D. P. Vakharia: Teaching–Learning-based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems, Computer-aided Design, Vol. 23, 2011, pp. 303-315

[34] N. Malarvizhi, V. R. Uthariaraj: Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment, First International Conference in Advanced Computing, 2009, pp. 97-104

[35] A Várkonyi-Kóczy, A. R.: A Load Balancing Algorithm for Resource Allocation in Cloud Computing. In Recent Advances in Technology Research and Education, Vol. 660, 2017, pp. 289-299

[36] B. Yagoubi and Y. Slimani: Task Load Balancing Strategy for Grid Computing, Journal of Computer Science, Vol. 12, 2007, pp. 186-194