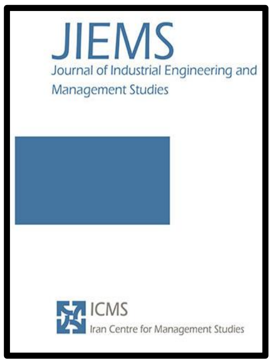
JIEMS
Journal of Industrial Engineering and Management Studies

Vol. 4, No. 1, 2017, pp. 69-89

DOI: 10.22116/JIEMS.2017.51964

www.jiems.icms.ac.ir


JIEMS
Journal of Industrial Engineering and Management Studies
ICMS
Iran Centre for Management Studies

Integrated modeling and solving the resource allocation problem and task scheduling in the cloud computing environment

Maryam Shams[1,*], Ahmad Jafarzadeh Afshari[1], Amir Khakbaz[1]

Abstract

Cloud computing is considered to be a new service provider technology for users and businesses. However, the cloud environment is facing a number of challenges. Resource allocation in a way that is optimum for users and cloud providers is difficult because of lack of data sharing between them. On the other hand, job scheduling is a basic issue and at the same time a big challenge in reaching high efficiency in the cloud computing environment. In this paper, "the cloud resources management problem" is investigated that includes allocation and scheduling of computing resources, such that providers achieve the high efficiency of resources and users receive their needed applications in an efficient manner and with minimum cost. For this purpose, a group technology based non-linear mathematical model is presented with an aim at minimization of load difference of servers, number of transfers between servers, number of active virtual machines, maximum construction time, the cost of performing jobs and active servers energy consumption. To solve the model, a meta-heuristic multi-objective hybrid Genetic and Particle Swarm Optimization algorithm is proposed for resource allocation and scheduling. In order to demonstrate the validity and efficiency of the algorithm, a number of problems with different dimensions are randomly created and accordingly the efficiency and convergence capability of the suggested algorithm is investigated. The results indicated that the proposed hybrid method has had an acceptable performance in generating high quality, diverse and sparse solutions.

Keywords: Cloud computing; Resource allocation; Task scheduling; Non-dominated sorting genetic algorithm; Particle swarm optimization.

Received: December 2016-15

Revised: April 2017-16

Accepted: May 2017-18

1. Introduction

Cloud computing is a model for providing easy access to a collection of changeable and configurable computing resources (like networks, servers, storage space, applications, and services) through a network based on the user demand and the access should rapidly be provided or freed with the least need to resource management or direct intervention of service provider. In the meantime, the issue of resource allocation in cloud computing is of great importance.

* Corresponding Author; mrym.shams@gmail.com
[1] Department of industrial engineering, Shomal University, Amol, Iran.

A resource allocation system for cloud computing can be referred to any type of mechanism whose goal is to guarantee the provision of making requirements of applications.
The issue of resource allocation is a major challenge in cloudy environments and has a direct relationship with the amount of energy consumption, service providers' profit, and users' cost; hence, many works have been done to reduce the number of physical or virtual resources used, by applying virtualization methods and creation of load balance and integration of physical resources. Cloud resources can be physical or virtual and the cloud provider requests them from the cloud. Generally, real resources of a data center which are shared between several users should dynamically be allocated according to the user's request or retaken. The resource allocation system should deal with unpredicted requests.
Having allocated the computational resources to users, there is a need to make decisions on the scheduling of jobs. The goal of scheduling is to determine a processing resource from a collection of the resources which are needed for processing of a job, such that more duties can be processed in less time. The scheduling system controls various duties in the cloud system to increase the rate of job accomplishment and productivity of resources and as a result, increase the computational power. Job scheduling is one of the important challenges of cloud computing systems. Owing to limitations and heterogeneity of resources, the issue of scheduling is among the NP-Complete issues (Garey, Johnson, 1979). An appropriate scheduling method is very effective on the increase of time of performing jobs and productivity of resources.
Many studies have been conducted on the issue of resource allocation and scheduling of jobs in the cloud computing environment (Daji et al., 2013). However, no deterministic mathematical model has been proposed on the integration of allocation and scheduling in the cloud computing environment. Allocation and scheduling of resources can seamlessly lead to the optimization of criteria and the main goals of users and service providers including transfer charges, servers load balance and management on machines. Thus, the issue needs further investigation.
In solving constrained optimization problems, there are two subjects of optimization and binding. Due to the complexity of the problem, a hybrid algorithm based on Non-dominated Sorting Genetic Algorithm (NSGA II) and Particle Swarm Optimization (PSO) has been proposed to solve it in an acceptable time. The goal of the algorithm is to use capabilities of the PSO algorithm due to its suitable rate of convergence to the optimum solution. In the first step, the proposed algorithm tries to use Genetic Algorithm capabilities to find the local optimal solution. In addition, we know that the penalty function method is the most common method for constrained optimization due to simplicity and ease of implementation. In the second step, using the co-evolutionary concept, the Co-evolutionary Particle Swarm Optimization approach (CPSO) is used to adapt the local optimal solutions to penalty factors to find the global optimal point out of local optimal solutions. In the next sections of this paper, after a brief overview of previous studies, the problem of resource allocation and scheduling using the concepts of group technology in the cloud computing environment is studied and a non-linear programming mathematical model is proposed for the problem. Then, the problem-solving method is considered and having explained the primary concepts and definitions, an efficient meta-algorithm is proposed combining Genetic Algorithm and PSO.

2. Literature review

Numerous criteria are found in the literature about the scheduling problem. Most studies are focused on minimization of the ending time of all jobs. Some of the studies conducted on the effect of scheduling criteria show that the use of the criterion alone for real problems of scheduling is not satisfactory enough. On the other hand, a general investigation of the studies conducted on scheduling problems indicates that utilization of a comprehensive view on the effective criteria on scheduling decisions has less been considered. It is obvious that in the real world, numerous criteria are suggested for the scheduling problem, some of which are different depending on the type of industry. Here, some of the studies conducted in the literature are investigated.

In 2008, Jaeger et al., explored the nature and potential of cloud computing, the policy issues raised, and research questions related to cloud computing and policy. In 2013, Eguia et al. proposed a linear programming model for the display of the problem of cellular manufacturing and jobs scheduling of reconfigurable cellular manufacturing systems (RCMS) with an aim of minimizing reconfiguration and efficiency costs for the condition of lack of use of RCMS resources. Then, in order to solve greater examples of this problem and find a suitable schedule without the need to long computation times, they used an efficient Tabu Search (TS) algorithm. In 2014, Kim et al. proposed a multi-agent-based scheduling method for intercellular and intracellular scheduling problem of configuration of the system and flow of materials in a production line of hybrid cells and then presented a real-time heuristic scheduling method to solve the problem.

The problem of constrained resource allocation is also one of the problems with an important position in planning and project control. The desirability of scheduling of activities of a project depends not only on the project completion time but also other factors. Among these factors, the way of using resources can be mentioned. The non-deterministic nature of activities in a project causes deterministic models not to be responsible for many practical projects. Therefore, in 2015, Wang and Su proposed a dynamic hierarchical resource allocation algorithm for multiple nodes of big data environments that using the fuzzy allocation pattern, dynamically classified jobs and nodes in different levels according to the computational power and storage factors.

Since solving real world problems with deterministic methods has been inefficient due to their large size, heuristic methods have always been considered in this area. Among the methods presented in the literature, the following studies can be mentioned. The results of the studies show that the proposed heuristic methods are more efficient and present solutions which are closer to the optimum solution.

In 2011, Cagnina et al., introduced a PSO-based method called CPSO-Shake for the constrained optimization. Making relatively simple changes in the conventional PSO algorithm, the method can better maintain the diversity and explore the limited search spaces more. In 2014, Moazami et al., proposed a new and rapid method for load shedding scheduling in which active and reactive values of power flow were prioritized via a dynamic list and optimized using culture–particle swarm optimization–co-evolutionary algorithm and artificial neural network method. In 2014, M.A. Rodriguez and R. Buyya put forward a scheduling and resource allocation strategy for scientific workflows in infrastructure clouds as a (laaS) service. They presented a PSO based algorithm with an aim at minimizing the total cost of workflow considering deadline constraints.

As mentioned, there are many models for scheduling problems in the literature. Given the nature of scheduling problems and especially with regard to the concept of group technology, which is the topic of the present study, there is no willingness for mathematical modeling and achieving the optimal solution of this kind of problems. Appearance of objective functions and non-linear constraints in the problem, binary variables, numerous variables and limitations that itself increases the size of the problem and naturally makes it more complicated and other issues have caused mathematical models and their solution not is used frequently in the literature because a lot of time is needed to solve them and given the current methods, types of software and the problem volume, achieving the desirable solution is never assured and even if it is achievable it’s a very time-consuming job. As mentioned before, in this paper, a resource allocation and scheduling problem is modeled using the concept of group technology considering special assumptions.

The current methods of the resource allocation such as FIFO, Round-Robin, ... which are used in the cloud, are unfairly allocated at times of high workload system. The clients are interested

in that their works are completed in the shortest possible time and the minimal cost which cloud service providers must receive. On the other hand, also the cloud service providers tend to maximize resource utilization and increase their profits, that these two are often in conflict with each other and does not match with the traditional methods of resource allocation and the available scheduling mechanisms. The examined problem of this study with considering to the specified requirements in the Service Level Agreement tries to find the best scheduling and resource allocation that be able to complete the works with maximum utilization of resources, at minimum cost, and in minimum time, by providing a deterministic integrated mathematical model. As yet, no deterministic mathematical model has been proposed on the integration of allocation and scheduling in the cloud computing environment.
In addition, given the difficulty of workshop flow and permutation workshop flow problems, deterministic approaches have less been considered to solve this type of problems. Few of these methods only investigated very small size problems. As a result, over the last fifty years, researchers have proposed many heuristic algorithms and methods to solve this type of problems of medium and large size. Thus, this study is looking for presenting a co-evolutionary mechanism based differential evolutionary method to solve this constrained optimization problem. At first, a penalty function is designed to manage constraints. Then, a co-evolutionary model is implemented to do a thorough search in the spaces of solutions and penalty factors. Given its co-evolutionary property, it seems it is significantly superior to the best existing methods.

3. Modeling the problem

In the upcoming problem, there are m virtual machines and n jobs in general. It should be noted that the m machines are settled in c cells or in other words, c servers and there is one or more of each one. Each job has a special process and the sequence of performing various operations to fulfill the job is definite. Furthermore, each job has a number of avocations or duties. Performing any task of each job has its special processing time on each machine. The hardware and software capacity of servers and virtual machines is known and each machine can only perform one operation or task at a time. It is assumed that in the case of starting an operation, its pause is not allowed. In this problem, the operation related to each job processing is scheduled on the virtual machines needed for processing in the cell where the machine is settled. The goal of solving this problem is to determine the sequence of jobs in each cell in such a way that goals of users as service providers are met, such that the following objective are optimized and the constraint in using the existing resources is observed.
1) Minimization of the intracellular movement in order to reduce communication costs
2) Equiponderating loads of servers and virtual machines for the purpose of increase of productivity of resources and the system
3) Minimization of the number of active machines to reduce intervention
4) Minimization of the construction time of cellular generation with the workflow structure in order to reduce servicing cycle
5) Minimization of the cost of performing task in order to increase users and service providers' profit
6) Minimization of energy consumption of servers to optimize consumption
In real problems, instead of dealing with one criterion we deal with several of them. Therefore the problem is considered as a multi-objective one in this study.

Problem Formulation

All the constants and variables used in the model are listed for easy reference below:

- t_{ijp} : $Process\ time\ \ i^{th}\ task\ in\ job\ j\ on\ the\ p-type\ virtual\ machine\ \ (1 \leq i \leq s, 1 \leq j \leq t, 1 \leq p \leq m)$

- D_j : $The\ number\ of\ demand\ of\ job\ j$
- T_p : $Total\ availability\ time\ of\ p-type\ virtual\ machine\ across\ servers$
- $a_{ijp} = \begin{cases} 1 & if\ i^{th}\ task\ in\ job\ j\ needs\ for\ p-type\ virtual\ machine \\ 0 & otherwise \end{cases}$
- TE_j : $The\ j^{th}\ job\ transmission\ time$
- N_{pk} : $The\ maximum\ number\ of\ p-type\ virtual\ machines\ that\ can\ be\ on\ k^{th}\ server\ (1 \leq k \leq c)$
- SC_{ij} : $Workload\ size\ of\ task\ i\ in\ job\ j$
- SD_{ij} : $Data\ size\ of\ task\ i\ in\ job\ j$
- DTI_k : $The\ data-transfer-in\ price\ of\ server\ k\ (\$/MBs)$
- DTI_k : $The\ data-transfer-out\ price\ of\ server\ k\ (\$/MBs)$
- Com_k : $The\ executing\ price\ of\ server\ k, represents\ the\ cost\ per\ executed\ million\ instructions\ (\$/MIs)$
- Str_k : $Storage\ price\ of\ server\ k\ (\$/MBs)$
- Bg_j : $User-defined\ budget\ of\ executing\ job\ j$
- cpu_{ij} : $The\ amount\ of\ required\ CPU\ of\ executing\ the\ i^{th}\ task\ in\ job\ j$
- mem_{ij} : $The\ amount\ of\ required\ memory\ of\ executing\ the\ i^{th}\ task\ in\ job\ j$
- sto_{ij} : $The\ amount\ of\ required\ storage\ of\ executing\ the\ i^{th}\ task\ in\ job\ j$
- CPU_k : $The\ CPU\ capacity\ of\ server\ k$
- MEM_k : $The\ memory\ capacity\ of\ server\ k$
- STO_k : $The\ storage\ capacity\ of\ server\ k$
- $E_{k,max}$: $The\ maximum\ power\ consumption\ of\ server\ k$
- $E_{k,idle}$: $the\ power\ consumption\ of\ server\ k\ when\ it\ is\ idle$
- E_k : $The\ dynamic\ power\ consumption\ of\ server\ k$
- ES_k : $The\ start/current\ power\ consumption\ of\ server\ k$
- $Y_{ijk} = \begin{cases} 1 & if\ i^{th}\ task\ belongs\ to\ job\ j\ is\ executed\ in\ server\ k \\ 0 & otherwise \end{cases}$
- $M_{rpk} = \begin{cases} 1 & if\ r^{th}\ p-type\ virtual\ machine\ which\ assigned\ to\ server\ k\ is\ activated \\ 0 & otherwise \end{cases}$
- $Z_{ijrpk} = \begin{cases} 1 & if\ i^{th}\ task\ belongs\ to\ job\ j\ is\ assigned\ to\ r^{th}\ p-type\ virtual\ machine\ in\ server\ k\ (1 \leq r \leq N_{pk}) \\ 0 & otherwise \end{cases}$
- $W_{ijlh} = \begin{cases} 1 & if\ i^{th}\ task\ belongs\ to\ job\ j\ is\ done\ earlier\ than\ l^{th}\ task\ belongs\ to\ job\ h\ (1 \leq l \leq s, 1 \leq h \leq t) \\ 0 & otherwise \end{cases}$
- $H_{ijlhrpk} = \begin{cases} 1 & if\ i^{th}\ task\ belongs\ to\ job\ j\ and\ l^{th}\ task\ belong\ to\ job\ h\ are\ assigned\ to\ r^{th}\ p-type\ virtual\ machine\ in\ server\ k \\ 0 & otherwise \end{cases}$
- $Act_k = \begin{cases} 1 & if\ server\ k\ is\ used \\ 0 & otherwise \end{cases}$
- Wt_{jpk} : $Workload\ of\ job\ j\ on\ p-type\ virtual\ machine\ on\ server\ k$
- Av_{jk} : $Workload\ average\ of\ job\ j\ on\ all\ virtual\ machines\ on\ server\ k$
- gs_{ij} : $Completion\ time\ of\ the\ i^{th}\ task\ in\ job\ j$
- gt_j : $Completion\ time\ of\ job\ j$
- g : $Total\ completion\ time\ of\ all\ jobs$
- $T_{i(i+1)j}$: $Transfering\ time\ of\ two\ consecutive\ tasks\ belongs\ to\ job\ j$
- Cst_{ijk} : $The\ executing\ price\ of\ i^{th}\ task\ belongs\ to\ job\ j\ on\ server\ k$

The problem of this work can be expressed in the fallowing mathematical model. The goal of this optimization problem is to minimize the total operational cost of the system including power and executing costs and maximize the efficiency of computing resources.

$$Min\ \ F = F_1 , F_2 , F_3 , F_4 , F_5 , g \tag{1}$$

$$F_1 = \sum_{j=1}^{t}\sum_{p=1}^{m}\sum_{k=1}^{c}(Wt_{jpk} - Av_{jpk})^2 \tag{2}$$

$$F_2 = \sum_{i=1}^{s-1}\sum_{j=1}^{t}\sum_{k=1}^{c} Y_{ijk}\,(1 - Y_{i+1jk}) \tag{3}$$

$$F_3 = \sum_{r=1}^{N_{pk}}\sum_{p=1}^{m}\sum_{k=1}^{c} M_{rpk} \tag{4}$$

$$F_4 = \sum_{i=1}^{s}\sum_{j=1}^{t}\sum_{k=1}^{c} Cst_{ijk}.Y_{ijk} \tag{5}$$

$$F_5 = \sum_{k=1}^{c}(Act_k.E_{k,idle} + E_k\sum_{i=1}^{s}\sum_{j=1}^{t} Y_{ijk}) \tag{6}$$

Subject to:

$$\sum_{r=1}^{N_{pk}} Z_{ijrpk} = a_{ijp}\,.Y_{ijk} \qquad \forall\, i\,, j\,, p\,, k \tag{7}$$

$$M_{rpk} \le \sum_{i=1}^{s}\sum_{j=1}^{t} Z_{ijrpk} \qquad \forall\, r\,, p\,, k \tag{8}$$

$$M_{rpk} \ge \frac{\sum_{i=1}^{s}\sum_{j=1}^{t} Z_{ijrpk}}{s \times t} \qquad \forall\, r\,, p\,, k \tag{9}$$

$$\sum_{r=1}^{N_{pk}}\sum_{p=1}^{m}\sum_{k=1}^{c} Z_{ijrpk} = 1 \qquad \forall\, i\,, j \tag{10}$$

$$\sum_{i=1}^{s}\sum_{j=1}^{t}\sum_{r=1}^{N_{pk}}\sum_{p=1}^{m} D_j\,.\,mem_{ij}\,.Z_{ijrpk} \le MEM_k \qquad \forall\, k \tag{11}$$

$$\sum_{i=1}^{s}\sum_{j=1}^{t}\sum_{r=1}^{N_{pk}}\sum_{p=1}^{m} D_j\,.\,cpu_{ij}\,.Z_{ijrpk} \le CPU_k \qquad \forall\, k \tag{12}$$

$$\sum_{i=1}^{s}\sum_{j=1}^{t}\sum_{r=1}^{N_{pk}}\sum_{p=1}^{m} D_j\,.\,sto_{ij}\,.Z_{ijrpk} \le STO_k \qquad \forall\, k \tag{13}$$

$$Wt_{jpk} = \frac{\sum_{i=1}^{s}\sum_{r=1}^{N_{pk}} D_j\,.\,t_{ijp}\,.\,Z_{ijrpk}}{T_p} \qquad \forall\, j\,, p\,, k \tag{14}$$

$$Av_{jk} = \frac{\sum_{p=1}^{m} Wt_{jpk}}{\sum_{r=1}^{N_{pk}}\sum_{p=1}^{m} M_{rpk}} \qquad \forall\, j\,,\ k \tag{15}$$

$$T_{i(i+1)j} = \sum_{k=1}^{c} Y_{ijk}\,(1 - Y_{i+1jk})\ .TE_j \qquad \forall\, i\,, j \tag{16}$$

$$gs_{i+1j} - gs_{ij} \geq t_{ijp} + T_{i(i+1)j} \qquad \forall\, i, j, p \tag{17}$$

$$gs_{lh} - gs_{ij} + M(1 - W_{ijlh}) + M(1 - H_{ijlhrpk}) \geq t_{lhp} \qquad \forall\, i, j, l, h, r, p, k \;\&\; i \neq l, j \neq h \tag{18}$$

$$gs_{ij} - gs_{lh} + M.W_{ijlh} + M(1 - H_{ijlhrpk}) \geq t_{ijp} \qquad \forall\, i, j, l, h, r, p, k \;\&\; i \neq l, j \neq h \tag{19}$$

$$W_{ijlh} + W_{lhij} = H_{ijlhrpk} \qquad \forall\, i, j, l, h, r, p, k \;\&\; i \neq l, j \neq h \tag{20}$$

$$W_{ijlh} + W_{lhuv} \leq W_{ijuv} + 1 \qquad \forall\, i, j, l, h, u, v \;\&\; i \neq l \neq u, j \neq h \neq v \tag{21}$$

$$gs_{ij} \geq t_{ijp} \qquad \forall\, i, j, p \tag{22}$$

$$gt_j \geq gs_{ij} \qquad \forall\, i, j \tag{23}$$

$$g \geq gt_j \qquad \forall\, j \tag{24}$$

$$Cst_{ijk} = cpu_{ij}.Com_k + sto_{ij}.Str_k + mem_{ij}.(DTI_k + DTO_k) \qquad \forall\, i, j, k \tag{25}$$

$$\sum_{k=1}^{c}\sum_{i=1}^{s} Cst_{ijk}.Y_{ijk} \leq Bg_j \qquad \forall\, j \tag{26}$$

$$\sum_{i=1}^{s}\sum_{j=1}^{t} Y_{ijk}.E_k \leq E_{k,max}.Act_k - E_{k,idle} - ES_k \qquad \forall\, k \tag{27}$$

$$\sum_{k=1}^{c} Act_k \geq \left\lceil \frac{\sum_{k=1}^{c} E_{k,idle} + ES_k}{E_{k,max}} \right\rceil \tag{28}$$

$$gs_{ij}, gt_j, g, \quad Cst_{ijk} \geq 0 \tag{29}$$

$$Y_{ijk}, Z_{ijrpk}, W_{ijst}, H_{ijstrpk}, Act_k, M_{rpk} \in \{0, 1\} \tag{30}$$

The objective function is composed of six terms that are shown in equations (2) to (6). The first term is related to load balance of servers. Using variance, the amount of load unbalance of the integrated load in servers can be calculated. The second term is related to the calculation of the number of transits between servers and in the case of the existence of a transfer, communication costs will increase. The third term is related to the calculation of the number of active virtual machines. The fourth term is related to the calculation of the cost of performing all duties of all jobs on all servers. It should be noted that the cost of calculation and storage is different on different servers depending on the capacity of each server and also network traffic etc. The fifth term calculates the energy consumption in the servers that are active. It is worth mentioning that the cost of power consumption of communication resources, coolers and air conditioning of modules (units) across servers and communication equipment/network is amortized in a data center and as a result, it is relatively independent of the user job volume. More exactly, these costs are not included in the cost and power equation of data center (Gudarzi et al., 2012) The last term, which is denoted by g in equation (1), is related to the maximum duration of time of performing jobs. Putting F_1, F_2, F_3, F_4, F_5 and g in the objective function causes the values to be minimized.

Constraint (7) is related to the definition of variable Z_{ijrpk} and shows that task i of job j can be done on the machine No. r of type p in server k (Z_{ijrpk}=1), only if the task is performed in server k (Y_{ijk}=1) and also the machine of type p is needed (a_{ijp}=1). Constraints (8) and (9) ensure us that in the case of allocation of at least one job to each of the machines, the related machine will be considered as active. Constraint (10) guarantees that each task of each job is allocated to one machine of one type and on one server.

Constraints (11), (12) and (13) guarantee that capacity of each server including memory, CPU, and storage space don't exceed the limit. Constraint (14) calculates the sum of tasks i={1,…,s} of job j on all active machines r of type p on server k. Calculation of a machine load is carried out according to the efficiency ratio of power/demand or (available time of a machine/the time needed for processing). The numerator of the above equation determines the total processing time of all duties allocated to all the machines of type p in server *k*. Constraint (15) calculates the average processing time within server for job *j* on all machines in sever *k*. In other words, like equation (14), the equation calculates the efficiency of a server.

Constraint (16) calculates the transfer time for two consecutive duties *i* and *i*+1 of job *j*. Constraint (17) guarantees that each task is processed on the machine of type p in accordance with the defined operation (tasks) priority in processing. It means that the difference between the completion times of task *i* and *i*+1 of job *j* should at least be equal to the sum of the time span of processing and transfer of task *i* of job *j*. It means task *i*+1 cannot be completed before task *i*. Constraints (18) and (19) shows the sequence of performing jobs on a machine. It means if task *i* of job *j* and task *l* of job *h* are both accomplished on machine *r* of type *p* in server *k*, the completion time of each task will be in accordance with the priority of their entry to the machine. Constraint (20) shows that if duties i and l of jobs j and h are both accomplished on machine No. r of type p in server k ($H_{ijlhrpk}$=1), one of the duties i or *l* of jobs *j* and *h* will be done before the other. In other words, we will have W_{ijlh}=1 or W_{lhij}=1. Constraint (21) expresses the trinity relation in the priority of jobs done on a machine and shows that jobs related to a machine are done in a queue. Constraint (22) guarantees that the completion time of each task is at least equal to its accomplishment time. Constraint (23) shows the completion time of each task. It means the completion time of job *j* should at least be equal to the completion time of its task. Constraint (24) shows that the time span of construction is more than or equal to the completion time of each job.

Constraint (25) calculates the cost of doing a job. A general cloud service provider receives the cost of three cases: calculations, storage, and data transfer. The data transfer is from/to the internet, private or public IP addresses etc. and because of the negligible cost of data transfer between cloud servers, it can be ignored in the modeling of the cost equation and its cost is set to zero (Wang et al, 2013). Constraint (26) is defined for budget control and guarantees that the cost of performing all duties of all jobs doesn't exceed the defined budget.

Each sever has an energy limit $E_{k,max}$ that cannot be exceeded. Constraint (27) guarantees that the servicing or hosting of virtual machines is done according to the remaining capacity. A constant or idle energy plus the start/recent energy of server *k* determines recent energy consumption of the server. For the severs that meet the conditions $E_{k,max}>E_{k,current}$ and $E_{k,current}\neq0$, the total number of used servers is constrained by the inequality (28) ($E_{k,current}=E_{k,idle}+ES_k$). Constraints (29) and (30) are obvious and introduce the type and range of variables.

4. The proposed algorithm

Due to the complexity of the problem, in this study, a hybrid meta-algorithm based on Non-dominated Sorting Genetic Algorithm (*NSGA II*) and co-evolutionary Particle Swarm Optimization (*PSO*) has been designed and implemented to solve it.

In the *NSGA II* there will be two groups of solutions; one group is parents' population from the previous stage, and the another group is the population obtained from the operation of integration and mutation operators on parents. Now, out of the two populations, some should be eliminated so that the primary number of population remains constant for the beginning of the next cycle. At first, the members of these two groups of solutions should be ranked. According to non-dominated sorting, the group of population members that has never been dominated is determined first and they get rank 1 (Rank=1). Then, for the rest of members, ignoring the effect of members with rank 1 on the population, the non-dominated sorting is accomplished again and members that are never dominated in this stage are distinguished with rank 2. This process continues until the rank of all members of the population is determined. For members that all have the same rank and another criterion should be exerted for the selection, which is the criterion of keeping the diversity of solutions.

This stage operator is called congestion distance operator. Each point with more congestion distance covers a wider area and its elimination leads to losing solution diversity in a wide range of solutions.

Each particle in the *PSO* algorithm consists of three d-dimensional vectors where d is the dimension of the search space. For the i^{th} particle, the three vectors are x^i current position of the particle, v^i the particle speed and $x^{i,\text{ best}}$ is the best position the particle has experienced so far. x^i is a set of coordinates that displays the current position of a particle. If the position is better than previous solutions, it is stored in $x^{i,best}$. f^i is the value of the objective function for x^i and $f^{i,best}$ is the value of objective function for $x^{i,best}$, both of which are regarded as constituent elements of a particle. In each iteration, a new x^i and v^i are obtained and the purpose of executing the algorithm is to improve $x^{i,best}$ with the probability of x^i.

As we know, Genetic Algorithm and *PSO* algorithm has respectively been successful in solving many discrete and continuous optimization problems (Pandian and Vasant, 2010). Given the discrete nature of scheduling problems and also multi-objectivity of the mathematical model proposed for this study, *NSGA II* internally deals with solving the problem optimally. On the other hand, the binding part of the problem which is expressed by the concepts of penalty functions, an external PSO algorithm adjusts the penalty factors. The two algorithms interactively and in parallel are used to solve constrained multi-objective optimization problems in a co-evolutionary way. In the next section, the meta-algorithm will be described in detail.

4.1. Co-evolutionary mechanism

Given the simplicity of rule and ease of implementation, the penalty function method is regarded as the most common technique in the management of constraints for application uses. In fact, new methods have only been used normally for dealing with very special problems (and generally unreal). In this study, inequality constraints are only considered because penalty functions are not as appropriate for managing equality constraints as for hard constraints and there are other suitable methods for their management (Michalewicz, 1996).

Since adjustment of penalty factors for the penalty function method is difficult, *Michalewicz* demonstrated that a self-adaptive plan is a hopeful management in co-evolutionary optimization. In his study, *Coello* also proposed the concept of co-evolutionary and combined it with a genetic algorithm to solve constrained optimization problems. *He*, *Wang* and *Huang* et al., respectively proposed a co-evolutionary *PSO* approach and a co-evolutionary mechanism based differential evolutionary method to solve constrained optimization problems. In the present research, some changes are made in the co-evolutionary process and Non-dominated Sorting Genetic Algorithm (*NSGA II*) and *PSO* are combined to solve multi-objective optimization problems. The proposed technique, using the concept of co-evolutionary concept, deals with the implementation of the idea that two (or more) populations are evolved simultaneously or interactively while the populations exchange the information in the process. As a result, by this combined optimization more worthy solutions that meet more constraints can be achieved.

The rule for multi-objective Non-dominated Sorting Co-evolutionary Genetic Algorithm – Particle Swarm Optimization (*NSCGA-PSO*) is shown in figure 1.

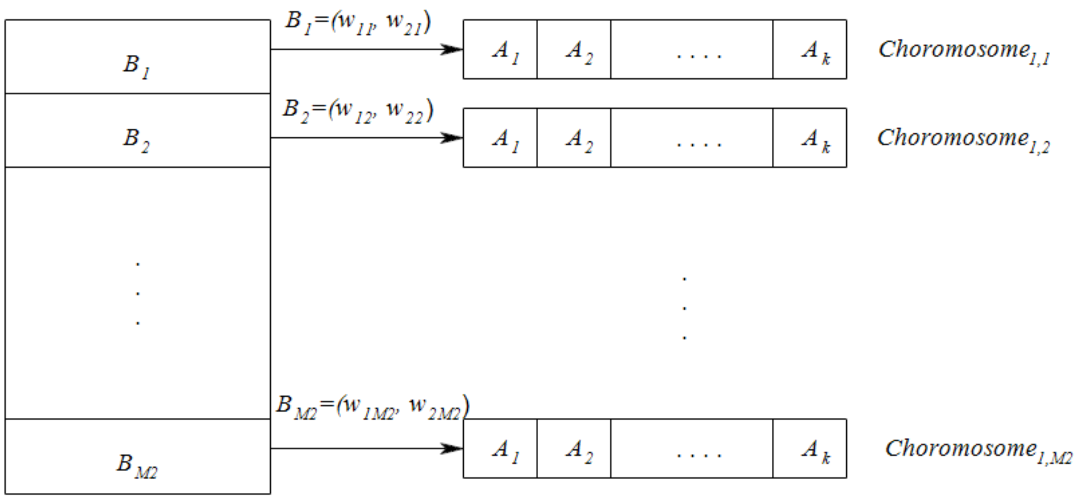


Figure 1. Graphical illustration for the notion of co-evolution

Two populations of Swarm and Chromosome are used in *NSCGA-PSO*. Population Swarm (designated by *Swarm*$_2$) is used with size M_2 for the adjustment of suitable penalty factors. In another section, the multiple population Chromosome (designated by *Chromosome*$_{1,1}$, *Chromosome*$_{1,2}$,….,*Chromosome*$_{1,M2}$), each with size M_1, are used in parallel to search for suitable decision solutions.
The problem that will be solved optimally is as follows:

$$\begin{aligned} &\text{Optimize } f_k(\text{x}). && k = 1, \dots , m. \\ &\quad \text{Subject to:} && \\ &\quad g_i(x) \le 0 && i = 1, \dots , p. \\ &\quad g_i(x) \ge 0 && \end{aligned} \tag{31}$$

When people are evaluated in Chromosome$_1$, a special penalty function is designed in which not only the number of violated constraints but also the amount of violation by the constraints is important. Specifically, the i^{th} Chromosome with regard to the k^{th} objective function in *Chromosome*$_{1,j}$ in *NSCGA-PSO* is evaluated by the following formula:

$$F_i(\text{x}) = f_{ik}(x) + (sum\_viol \times w_1 + num\_viol \times w_2) \times \beta \tag{32}$$

Where f_{ik}(x) is the fitness value of the i^{th} individual with regard to the k^{th} objective, *sum_viol* is the sum of total values violated by the constraints and *num_viol* is the number of violations of the constraints which is initialized with zero. w_1 and w_2 are penalty factors corresponding to particle B_j in *Swarm*$_2$. β is the effect coefficient of violation in order to increase its effect and prevent regeneration of infeasible solutions.
As we know, the amount of violation of inequality constraints is calculated as follows:

$$V_{gi} = \max\left\{0, 1 - \frac{g_i(x)}{g_{0i}}\right\} \qquad \forall g_i(x) \ge 0 \tag{32}$$

$$V_{gi} = \max\left\{0, \frac{g_i(x)}{g_{0i}} - 1\right\} \qquad \forall g_i(x) \le 0 \tag{33}$$

In this case, the value of *sum_viol* is calculated as follows:

$$sum\_viol = \sum_{i=1}^{N} V_{gi} \tag{34}$$

Where *N* is the number of inequality constraints (here it is assumed that all equality constraints are converted to inequality constraints).
Each individual $B_j(1\leq j\leq M_2)$ in *Swarm₂* displays a set of penalty factors (w_1 and w_2). The generated combined weights (that is penalty factors) are used in a definite number of generations (G_1) for the evolution of Chromosome$_1$. Having evolving each *Chromosome$_1$* corresponding to it, the j^{th} individual B_j in *Swarm$_2$* is evaluated as described below:
1- If there is at least one feasible solution in *Chromosome$_{1,j}$*, then individual B_j is called a valid individual, which is evaluated as follows:

$$P(B_j) = \frac{\sum f_{feasible}}{num\_feasible} - num\_feasible \tag{35}$$

Where $\sum f_{feasible}$ is the sum of values of the objective function of feasible solutions in Chromosome$_{1,j}$ and *num_feasible* shows the number of feasible solutions in *Chromosome$_{1,j}$*.
If infeasible solutions are not kept out of the calculation, the selection mechanism of genetic algorithm may cause the population to be affected by regions of the search space in which solutions with there are very low weight combinations (w_1 and w_2). Such solutions may have good fitness values but are infeasible as well. In fact, the use of β coefficient in equation (33) is to prevent the emergence of such problems. In addition, deduction from *num_feasible* in equation (35) is for preventing from the recession of *Chromosome$_{1,j}$* in special regions that have very small number of feasible individuals with good objective values.
2- If there is no feasible solution in *Chromosome$_{1,j}$* (the penalty can be considered very low), then B_j will be known as an invalid individual, which is evaluated as follows:

$$P(B_j) = \max(P_{valid}) + \frac{\sum sum\_viol}{\sum num\_viol} + \sum num\_viol \tag{36}$$

Where max(P$_{valid}$) shows the maximum value of the objective function of all valid individuals in P$_2$. $\sum sum\_viol$, shows the sum of violations of constraints for all people in Chromosome$_{1,j}$, and $\sum num\_viol$ counts the total number of violations of constraints for all people in Chromosome$_{1,j}$.
Obviously, using equation (36), an individual in Swarm$_2$ that results in the lower amount of constraints violation of Chromosome$_{1,j}$ is paid more attention. As a result, the search may lead Chromosome$_{1,j}$ towards regions with the small sum of constraints violations (that is the boundary of the feasible region). In addition, adding the term max (P$_{valid}$) ensures that valid people are always better than invalid people, so that the search is led toward the feasible region.
The process shown above is repeated until all individuals in *Swarm$_2$* have a fitness value (the best P(B_j) corresponding to *Chromosome$_1$*). It is worth mentioning that the interaction between *Chromosome$_1$* and *Swarm$_2$* introduces the diversity in both populations and prevents genetic algorithm easily from trapping in convergence to a local optimal. The proposed algorithm framework is shown in figure 2.

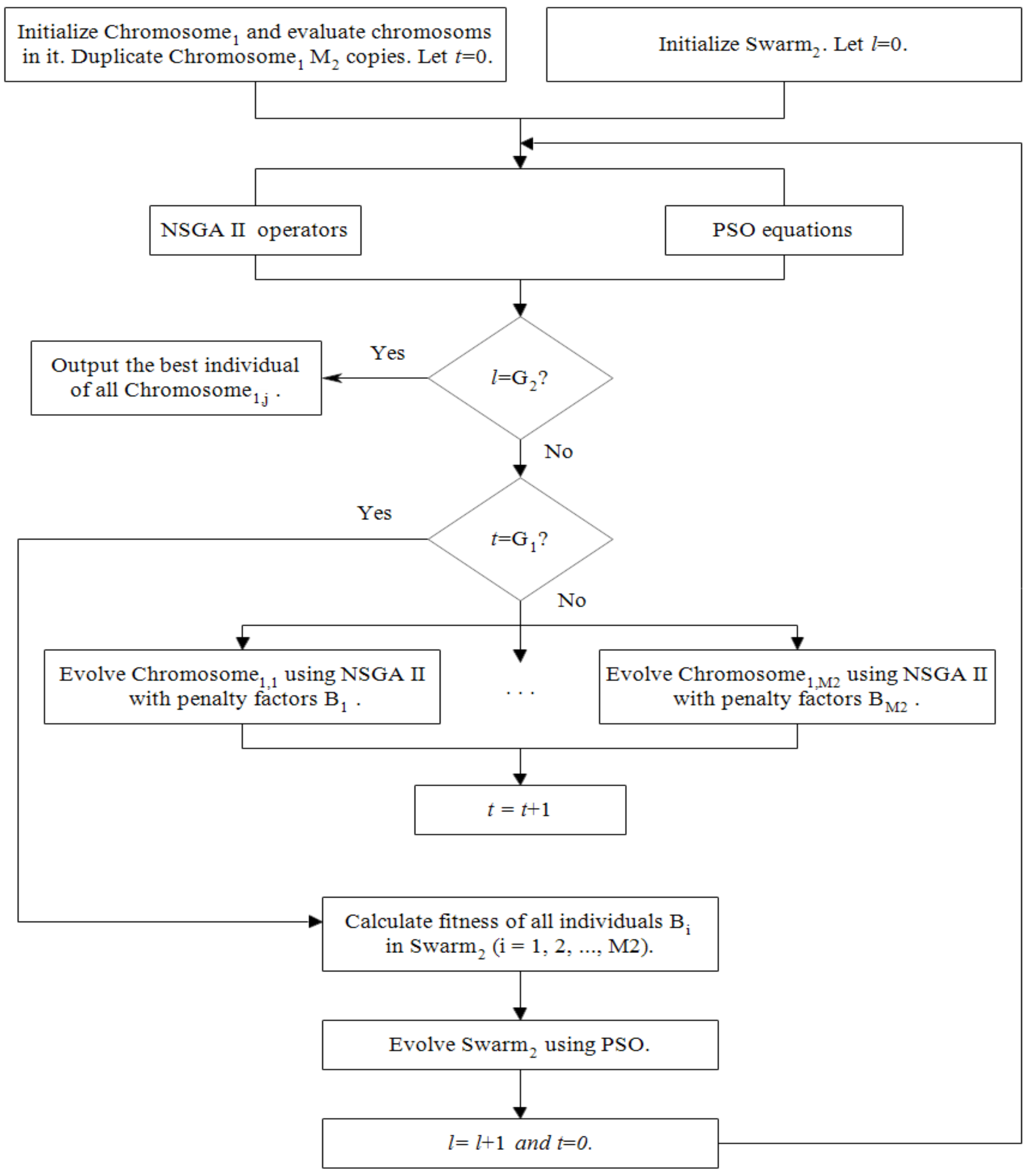

Initialize Chromosome$_1$ and evaluate chromosoms in it. Duplicate Chromosome$_1$ M$_2$ copies. Let t=0.
Initialize Swarm$_2$. Let l=0.
NSGA II operators
PSO equations
Output the best individual of all Chromosome$_{1,j}$.
Yes
l=G$_2$?
No
Yes
t=G$_1$?
No
Evolve Chromosome$_{1,1}$ using NSGA II with penalty factors B$_1$.
. . .
Evolve Chromosome$_{1,M2}$ using NSGA II with penalty factors B$_{M2}$.
t = t+1
Calculate fitness of all individuals B$_i$ in Swarm$_2$ (i = 1, 2, ..., M2).
Evolve Swarm$_2$ using PSO.
l= l+1 and t=0.

Figure 2. Flow chart of *NSCGA-PSO*.

4.2. Design of Taguchi experiment

The effectiveness of meta-heuristic algorithms is very much dependent on the correct selection of parameters (Al-Aomar and Al-Okaily, 2006). In this section, the effect of various parameters on the proposed algorithm is studied. The full factorial design, which tests all possible combinations of factors, is an extensive method which has been used in a variety of studies. However, when the number of factors increases significantly, the full factorial design may not be implementable. In this study, there are four factors at three levels and each combination should be run five times. Thus, the total number of runs is equal to$(3^4) \times 5 = 405$. In order to perform a robust test, several experiment designs have been suggested to reduce the number of experiments. Among them, Taguchi method has been successfully used in a wide range of experiments. In order to distinguish and obtain the optimum conditions out of the experimental runs, the analysis of *S/N* ratio was used. The ratio of S/N is calculated as follows in Taguchi method (Abd-El-Wahed et al., 2011):

$$\frac{S}{N}\, ratio = -10 \log_{10} \frac{sum(objective\ function)^2}{n} \qquad (37)$$

Where the objective function is the output value for each run of experiment and n is the number of repetition of the experiment. The quality of solutions generated by various algorithms is assessed using the following equation (Hamilton and Vairaktarakis, 2013):

$$Relative\ Deviation\ Index\ (RDI) = \frac{MethodSol - BestSol}{WorstSol - BestSol}\ 100\% \qquad (38)$$

Where *BestSol* and *WorstSol* are respectively the best and the worst values of objection function by the tested method (values are shown by *MethodSol*).
Parameters of the proposed algorithm and their level are shown in table 1. The probability of mutation and crossover for each pair of chromosomes is normally considered between 0 and 1 and the effect rate of mutation (mutation intensity) is considered very small and from 0.001 to 0.1. Here, values related to levels of each of control factors are considered according to previous studies.

Table 1. Controllable factors and their levels for DOE.

	Low	**Medium**	**High**
Parameters	**1**	**2**	**3**
Population size	10	15	**20**
Crossover percentage	0.75	0.8	**0.9**
Mutation percentage	0.2	0.3	**0.4**
Mutation rate	0.01	0.02	**0.05**

Having determined performance criteria, control factors, and their levels, Taguchi method can be used for the design of experiments. In this example, the MINITAB statistical software has been used to design the experiments. The experiments designed by Taguchi method and also the results of experiments for each one of response variables for a sample problem has been shown in table 2. Furthermore, according to equation (38), the best-calculated cost for each experiment has been normalized. The total number of designed experiments is equal to 9, each of which was repeated five times. All experiments were carried out using MATLAB software.

Table 2. The design of Experiments Using the Taguchi method and expected results.

Run	**Population size**	**Crossover percentage**	**Mutation percentage**	**Mutation rate**	**Best Cost**	**RPD**
1	1	1	1	1	20.9155	100.00
2	1	2	2	2	20.8901	99.86
3	1	3	3	3	12.2337	53.89
4	2	1	2	3	11.7723	51.44
5	2	2	3	1	11.8631	51.92
6	2	3	1	2	11.1855	48.33
7	3	1	3	2	2.7849	3.71
8	3	2	1	3	2.0844	0
9	3	3	2	1	2.8265	3.94

In table 3, the value of S/N of the run of each experiment has been given to find the best values for parameters and each factor has been assigned a ratio of S/N. Out of the obtained values of S/N, the maximum value showed the optimum condition. In figure 3, the data are drawn.

Table 3. The average of $\frac{S}{N}$ ratio for expected Best Cost.

Parameters	**Mean of $\frac{S}{N}$ ratio**			
	Level 1	**Level 2**	**Level 3**	**Max**
Population size	-24.8531	-21.2914	-8.1002	**-8.1002**
Crossover percentage	-18.9076	-18.0875	-17.2497	**-17.2497**
Mutation percentage	-17.9207	-18.9470	-17.3771	**-17.3771**
Mutation rate	-18.9728	-18.7560	-16.5160	**-16.5160**

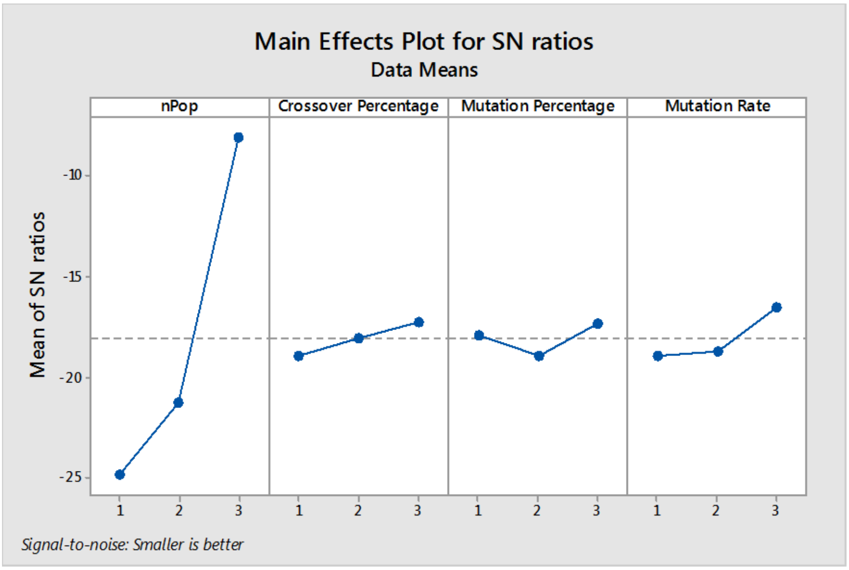


Figure 3. The average of $\frac{S}{N}$ ratio plot at each level for the objective function values.

In figure 3, the first graph is related to the size of the population of Non-dominated Sorting Genetic Algorithm in which the third factor value of -8.10026 has the optimum condition out of three ratios, while the factor with value -24.8531 has the worst situation. Given the high slope of the diagram, it can be concluded that the variable of population size has the highest effect on the optimum value of objective function.
According to the results obtained from table 3, the optimum conditions for the adjustment of genetic algorithm parameters is when all variables are considered at level 3.

5. Calculation results

In this section, the proposed approach is executed on small and medium size problems. For each of the generated problems, each algorithm is run five times and the results are expressed as the average value of runs. The BC value in table 4 represents the fitness of the best individual in all chromosomes and particles, which has been obtained as a result of various iterations of the two algorithms. The assumptions considered for these problems are:

- The number of iterations for the external algorithm is considered as 30.
- Processing times for small and large problems are randomly generated respectively in the interval [1,5] and [1,10].
- In small and large problems, the size of the population is considered respectively as 30 and 50.

Table 4. Obtained Pareto front solutions for small and large problems.

No.	**Number of jobs**	**Number of machines**	**Number of servers**	***BC***	***Objective functions***						***Time(s)***
					F_1	***F_2***	***F_3***	***F_4***	***F_5***	***g***	
1	3	3	2	0.9449	0.0395	2	8	5.2043	412	21	90
2	4	2	2	2.2134	0.7814	2	6	5.4752	422	19	98
3	3	4	2	2.612	0.3938	4	7	5.7443	431	22	124
4	4	4	2	2.9878	0.3669	3	8	5.0182	484	16	130
5	4	5	2	5.4027	0.1080	0	10	5.3327	502	16	133
6	5	5	2	6.6349	0.3834	4	13	6.3607	544	19	160
7	5	7	2	3.3196	0.4645	4	14	6.3635	789	22	181
8	6	7	2	13.7789	0.5539	5	16	8.5198	893	19	237
9	6	8	2	18.4312	0.7886	9	19	9.3692	1142	22	296
10	7	8	3	20.4043	0.8222	11	20	9.6759	1477	25	359
11	8	8	3	29.3238	1.8638	8	20	10.8851	1677	22	410
12	10	8	3	31.1645	7.5325	29	34	7.5325	1741	76	1636
13	15	10	3	49.037	18.0073	41	51	63.0193	2877	82	5100
14	20	10	4	98.2095	59.8728	60	82	75.7724	4135	129	13258
15	25	10	4	129.5309	98.329	69	94	110.142	5044	139	18756
16	30	12	5	184.5671	126.9102	97	122	133.3003	5461	135	21342
17	35	15	5	193.5029	188.5085	112	138	150.932	7017	138	26049
18	40	15	6	214.1988	221.08	131	159	163.2509	8593	141	30470
19	45	20	6	264.6019	287.4831	149	182	175.834	9149	142	38706
20	50	20	6	312.4821	325.7313	173	199	181.6119	10128	144	52806

The nature of the problem solving algorithms is to reach the best solution in an acceptable time. The two parameters, that is calculation time and the optimum value of solutions, are suggested as the main factors of comparison and conclusion. In the proposed mathematical model, weights of all objective functions are assumed equal. As we know, when cost functions become non-linear, calculations need more time than the linear case. The obtained results of the optimum Pareto solutions show that *NSCGA-PSO* algorithm has an acceptable performance when finding the optimum value of problems. Moreover, the time to reach the optimum solution corresponding to development and extension of problems has an intangible increase in comparison with the Lingo software. As a result, it increases considerably the speed of decision making for resource allocation and jobs scheduling in the cloud computing environment. The diagram of the changes of calculation time of sample solved problems is shown in figure 4.

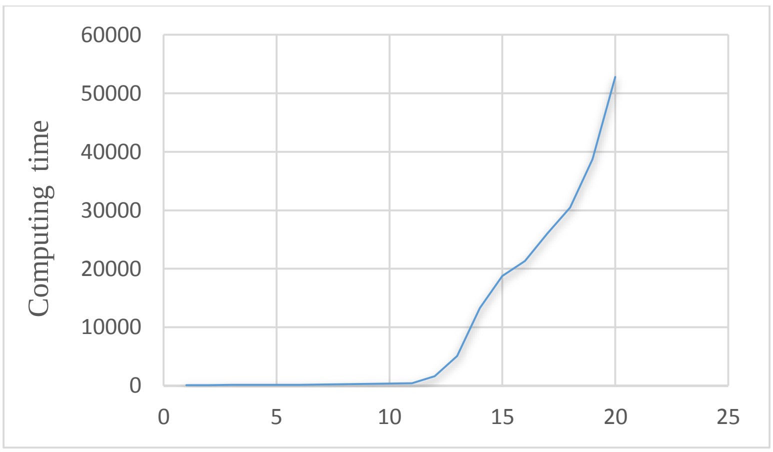


Figure 4. The computing time changes plot.

Non-dominated Sorting Genetic Algorithm is one of the most powerful methods for multi-objective optimization, but optimization by this method is quite time consuming and costly. Combining this algorithm with the *PSO* algorithm in a co-evolutionary manner, as described in the previous section, is an attempt to for reducing the optimization time of the method. In *PSO* algorithm, members of population communicate with each other and reach the solution by the exchange of information, which results in a high convergence rate. In this way, the advantages of both *NSGA* and *PSO* are used and the optimization and convergence rate is increased significantly. In order to test the effect of different iterations (Max-iterations in the improvement step) on the final solution quality, the convergence curve of the best individual position cost is all chromosomes is drawn for different iterations.

Convergence curves of objective functions for populations with size 30 and 50 and number of iterations of 500 generations are respectively shown in figures 5 and 6. According to figure 5, optimization process continues to 100 generations, which leads to finding an individual with the cost value of -3. According to the figure, in generations No. 10, 20, 70, 100 and 470, local search by *NSGA* has led to a considerable mutation in the convergence process. In figure 6 with a number of generations of 50, the optimization process has continued to 170 generations, which has led to finding an individual with the cost value of -4. According to the figure, in generations with No. 5, 60 and 170, local search by *NSGA* has led to a considerable mutation in the convergence process. Thus, not changing the generations, the decision solution will approach the absolute optimal as much as possible. The results indicate that increasing the convergence rate of multi-objective optimization problems, *NSCGA-PSO* is capable of presenting an excellent solution.

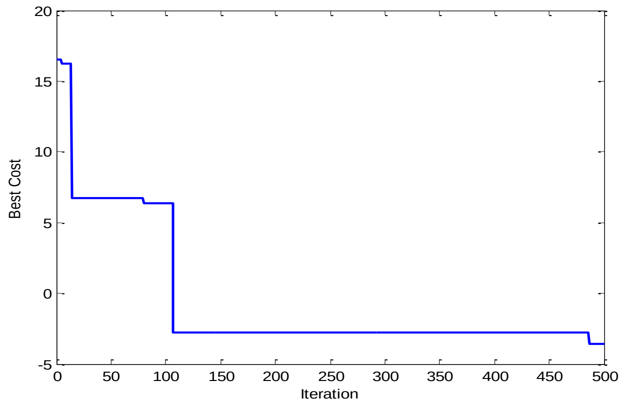


Figure 5. Convergence curves with the population size of 30.

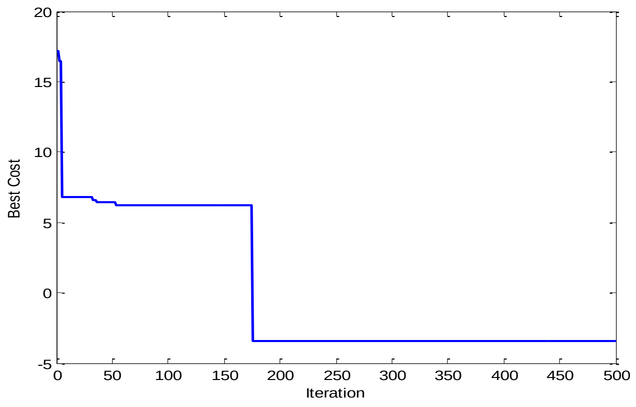


Figure 6. Convergence curves with the population size of 50.

The another criterion for comparing the results is the capability of the proposed algorithm to meet all constraints of the problem in this study. Recorded results of table 4 in the previous section indicate that except for the constraints of the sequence of doing jobs on a machine at a time in which there is deviation in some of the primary populations, the algorithm has had good performance in meeting nearly all constraints of problems of various sizes.

The last comparison criterion of the study is to show how much the proposed algorithm is able to regenerate the optimum Pareto deploy and that it is possible to identify and generate the highest size of the optimum deploy using a hybrid of two algorithms of *NSGA* and *PSO*. As mentioned in previous sections, instead of optimizing a single objective function, several objective functions should simultaneously be optimized in multi-objective problems. In such circumstances, instead of resulting in one single solution, the optimization output leads to a set of optimum solutions (Pareto optimum solutions) such that none of the Pareto deploy solutions is preferred to the others and depending on the situation, each one can be considered as the optimum choice.

In figures 7 to 9, the Pareto deploy obtained by the algorithm for different objective functions are given in a pairwise manner. In the figures, the points designated by the red color are the same as the efficient or optimum Pareto points, which have been detected through domination calculations. As is seen, Pareto solutions are uniformly distributed across the search space and also different objective functions have a similar behavior in the direction of optimization. In this way, if the value of one function increases, so does the value of other functions. The lowest point on the Pareto curve is chosen as the most optimum solution. Selection of the optimum value of each function is accomplished while considering their importance for the system designer as well as the balance between the cost and the system efficiency.

Thus, given the obtained results, it seems that in the space of constrained optimization problems, *NSCGA-PSO* is an efficient method from the perspective of identification of the optimum Pareto deploy, meeting constraints as well as acceptable running time.

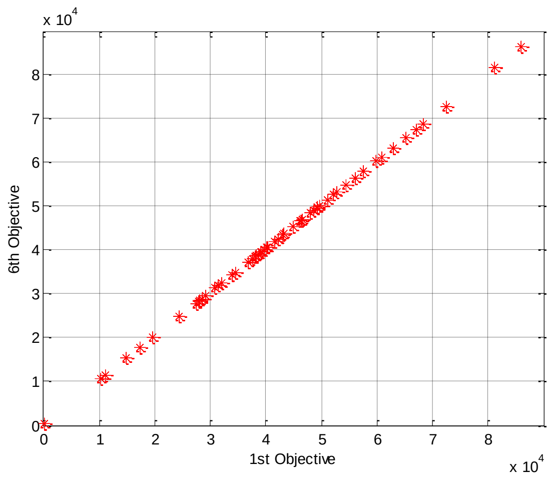


Figure 7. Comparing of the first and sixth objective functions in the range of obtained solutions from Pareto plot.

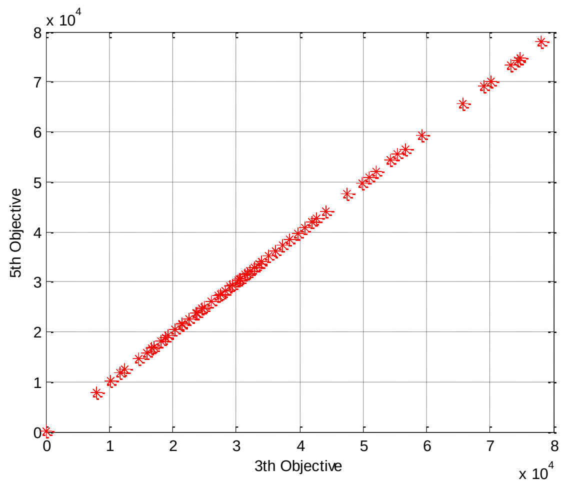


Figure 8. Comparing of the third and fifth objective functions in the range of obtained solutions from Pareto plot.

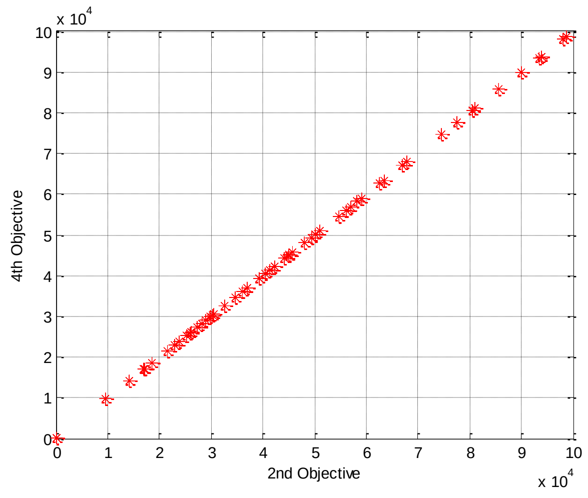


Figure 9. Comparing of the second and forth objective functions in the range of obtained solutions from Pareto plot.

6. Conclusion

In this paper, a resource allocation and jobs scheduling problem in the cloud computing environment was first described in accordance with group technology concepts and the problem of basic cellular generation, which is an appealing issue in the literature, was taken into account. The problem is the allocation of resources and scheduling of a number of jobs (each containing a definite number of duties) in a cellular generation system as a one by one procedure. Presenting an integrated deterministic mathematical model, the problem under investigation tries to find the best scheduling and resource allocation so as to be able to fulfill duties with the highest degree of resource utilization, minimum cost as well as minimum time.

The criterion selected for the objective function is the popular criterion of the total completion time, which has been considered more by many researchers in the literature than any other criterion. Other optimization criteria of the study are a minimization of servers load difference, a number of intra-server movements, a number of active virtual machines, cost of doing jobs and energy consumption of servers. Given its complex nature, it is classified as an NP-Hard problem. In the beginning, a mathematical model was developed to solve the problem. Owing to the existence of many constraints and variables in the model, solving problems of higher size (more jobs and cells) necessitates spending much time. Thus, a hybrid Non-dominated Sorting Co-evolutionary Genetic Algorithm and Particle Swarm Optimization, namely *NSCGA-PSO,*

was introduced to solve problems of higher size. The proposed *NSCGA-PSO* was presented to achieve an absolute solution with desirable quality. In this method, a penalty function for each constraint as a penalty factor for all constraint was considered. Then, an internal *NSGA* solved the problem for different penalty factors and obtained a locally optimal solution. At the same time, the external algorithm of *PSO* resulted in the global optimum solution by choosing the best penalty factor in a co-evolutionary way and in parallel. When the two algorithms were used to optimize objective of the problem considering the constraints, much more effective outcomes were obtained. The results obtained from solving small and medium size problems with *NSCGA-PSO* method were presented in the fifth section to demonstrate the proposed meta-algorithm efficiency.
In the end, analyzing the convergence of the proposed algorithm, meeting constraints and Pareto solutions archive, its effectiveness and optimum performance in expediting the achievement of the final desirable solution was proved. The results indicated that the proposed hybrid Non-dominated Co-evolutionary Genetic Algorithm and *PSO* have had an acceptable performance in generating high quality, diverse and sparse solutions.
Elimination of all restricting assumption of the problem which was mentioned in the third section of the paper can be regarded as appropriate areas for future studies. For example, the inclusion of stochastic or fuzzy processing times of each job which may have examples in the real world. Consideration of other objective functions with regard to the nature of each industry, considerations of preparation times and available times of machines, investigation of the problem in the dynamic mode, consideration of cut of jobs or use of other meta-heuristic methods to solve the problem is suggested for future research.

References

Abd-El-Wahed, W.F., Mousab, A.A. & El-Shorbagy, M.A., (2011). "Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems", *Journal of Computational and Applied Mathematics*, Vol. 235, No. 5, pp 1446–1453.

Al-Aomar, R. and Al-Okaily, A., (2006). "A GA-based parameter design for single machine turning process with high-volume production", *Computers & Industrial Engineering*, Vol 50, pp 317–337.

Cagnina, Leticia Cecilia; Esquivel, Susana Cecilia, Coello Coello, Carlos A., (2011). "Solving constrained optimization problems with a hybrid particle swarm optimization algorithm", *Engineering Optimization*, Vol. 43, No. 8, pp 843-866.

Coello, C.A.C., (2000). "Use of a self-adaptive penalty approach for engineering optimization problems", *Computers in Industry*, Vol. 41, pp 113–127.

Eguia, Ignacio, Racero, Jesus, Guerrero, Fernando, Sebastian Lozano, (2013). "Cell formation and scheduling of part families for reconfigurable cellular manufacturing systems using Tabu search, Simulation", *Transactions of the Society for Modeling and Simulation International*, pp 1–17.

Emmons, Hamilton, Vairaktarakis, George, (2013). Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications, Springer, New York.

Ergu, Daji, Kou, Gang, Peng, Yi, Shi, Yong and Shi, Yu, (2013). "The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment", *The Journal of Supercomputing*, Vol. 64, No. 3, pp 835-848.

Goudarzi, H., Ghasemazar, M. and Pedram, M., (2012). "SLA-based Optimization of Power and Migration Cost in Cloud Computing, Cluster, Cloud and Grid Computing (CCGrid)", 2012 12th IEEE/ACM International Symposium on Ottawa, pp 172 – 179.

He, Qie and Wang, Ling, (2007). “An effective co-evolutionary particle swarm optimization for constrained engineering design problems”, *Engineering Applications of Artificial Intelligence*, Vol. 20, pp 89–99.

Huang, Fu-zhuo, Wang, Ling and He, Qie, (2007). “An effective co-evolutionary differential evolution for constrained optimization”, *Applied Mathematics and Computation*, Vol. 186, pp 340–356.

Jaeger, Paul T., Lin, Jimmy, Grimes, Justin M., (2008). “Cloud Computing and Information Policy: Computing in a Policy Cloud?”, *Journal of Information Technology & Politics* , Vol. 5, No. 3, pp 269-283.

Javid, J., (2010). “Modeling supply chain management”, 2nd International Industrial Engineering Conference. Tehran, Iran, 14-20.

Kim, Jungyun, Chung, Seong Youb, Yoon, Hyun Joong, (2014). “Multi-agent-based scheduling methods for hybrid cellular production lines in semiconductor industry”, *Journal of Engineering Manufacture*, Vol. 228, No. 12, pp 1701–1712.

Kim, S.J., Kim, K.S., Jang, H., (2003). “Optimization of manufacturing parameters for a brake lining using Taguchi method”, *Journal of Materials Processing Technology*, Vol. 136, pp 202–208.

Kolischa, R. and Padman, R., (2001). “An integrated survey of deterministic project scheduling”, *Omega*, Vol. 29, pp 249–272.

Kuo, R.J. & Han, Y.S., (2011). “A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem – A case study on supply chain model”, *Applied Mathematical Modelling*, Vol. 35, No. 8, pp 3905–3917.

Moazzami, Majid, Khodabakhshian, Amin, Hooshmand, Rahmat-Allah, (2014). “A New Optimal Under-frequency Load-shedding Method Using Hybrid Culture–Particle Swarm Optimization–Co-evolutionary Algorithm and Artificial Neural Networks”, *Electric Power Components and Systems*, Taylor & Francis Group, Vol. 43, pp 69–82.

M. Garey and D. Johnson, (1979). *Computers and intractability: a guide to the theory of NP-Completeness*, 1st edition New York: WH Freeman and Company.

Mozdgir, Ashkan, Mahdavi, Iraj, Seyedi Badeleh, Iman, Solimanpur, Maghsud, (2013). “Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing”, *Mathematical and Computer Modelling*, Vol. 57, No. 1–2, pp 137–151.

Pandian M., Vasant, (2012). *Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance*, IGI Global, 1 edition, United States of America.

Rodriguez, M.A., Buyya, R., (2014). “Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds”, *Cloud Computing, IEEE Transactions*, Vol. 2, No. 2, pp 222 – 235.

Valdez, Fevrier, Melin, Patricia, Castillo, Oscar, (2011). “An improved evolutionary method with fuzzy logic for combining Particle Swarm Optimization and Genetic Algorithms”, *Applied Soft Computing*, Vol. 11, No. 2, pp 2625–2632.

Wang, Wei-Jen, Chang, Yue-Shan, Lo, Win-Tsung, Lee, Yi-Kang, (2013). “Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments”, *The Journal of Supercomputing*, Vol. 66, No. 2, pp 783-811.

Wang, Zhanjie and Su, Xianxian, (2015). *Dynamically hierarchical resource-allocation algorithm in cloud computing environment*, Springer Science+Business Media New York.

Z. Michalewicz, (1996). *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd edn., Springer-Verlag.

Z. Michalewicz, N. Attia, (1994). “Evolutionary optimization of constrained problems” , Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific Publishing, River Edge, NJ, pp 98–108.