Cumhuriyet Üniversitesi Fen Fakültesi
Fen Bilimleri Dergisi (CFD), Cilt:36, No: 4 Özel Sayı (2015)
ISSN: 1300-1949

Cumhuriyet University Faculty of Science
Science Journal (CSJ), Vol. 36, No: 4 Special Issue (2015)
ISSN: 1300-1949

Cloud Computing Resource Planning Based on Imperialist Competitive Algorithm

Armita ABARGHOEI[1,*], Ebrahim MAHDIPOUR[2], ,Majid ASKARZADEH[1]

[1]*Department of computer, Qeshm international Branch, Islamic Azad University, qeshm ,Iran.*

[2]*Department of Computer Engineering, Science And Research Branch, Islamic Azad University, Tehran, Iran.*

Received: 22.03.2015; Accepted: 29.05.2015

Abstract. Resource allocation in cloud computing and the scheduling of each user works on existing virtual machines is an NP-Complete problem, in which many algorithms are provided for solving the problem so far. But none of these algorithms are able to meet the requirements related to speed and accuracy in cloud computing environments. In this paper, a combination of imperialist competitive and local search optimization algorithms is proposed to solve this problem. The algorithm attempts to improve the possible responses by creating an initial empire, through applying imperialist competitive algorithm. To avoid premature convergence, imperialist competitive algorithm is combined with a local search algorithm. The hybrid proposed algorithm used a convergence diagnosis mechanism based on semblance coefficient, and in times of premature convergence in imperialist competitive algorithm, local search algorithm runs. Quality and efficiency of the proposed algorithm are compared with round-robin, ant colony and genetic algorithms. The results show that the proposed algorithm in terms of time and the responses' quality is faster than the ant colony optimization and genetic algorithms.

Keywords: resource allocation, cloud computing, imperialist competitive algorithm, local search, NP-Complete

INTRODUCTION

In recent years, attention is increased in cloud computing. The most reception of this technology took place in multimedia industry and analysis systems. In cloud computing, some decisions are known as planning decisions. Planning process determines required resources for production of required activities for scheduling [1]. Scheduling is the allocation of limited resources to activities over time, to optimize one or more objective function [1]. Scheduling of cloud computing resources is one of the most important issues in scheduling, which has attracted the attention of many researchers. Resource Scheduling is a very hard problem among NP-hard problems [2, 3].

The issue of scheduling and resources allocation in a cloud and a net is a very difficult problem. As a result, the search space is large enough that if an algorithm tries to examine the space in order to find the best answer, needs exponential time. Some innovative algorithms that have been used in scheduling and resource allocation are ant colony optimization [4, 5] and genetic algorithms [6].

Ant colony algorithm is an algorithm based on collective intelligence, which is inspired by studying the colonies of ants [7]. In real life, the ants are searching in random manner to find food. Then, they come back to the nest and leave a trace of pheromones on the track. When an ant moving accidently and solely, is faced with a path that has more pheromones, it most likely choose that path. And it strengthens the path through more creating more pheromone. Although presented algorithms based on ant colony [4, 5] provide the ability of parallel scheduling resources allocation in cloud computing, for the implementation of efficient algorithms a powerful parallel hardware is needed. And it will require considerable time.

In [6] scheduling algorithm and resources allocation in cloud and net computing are studied through genetic algorithm is. Genetic algorithm is an innovative algorithm that uses Darwin's

* Corresponding author. Armita ABARGHOEI

Special Issue: Technological Advances of Engineering Sciences theory of "survival of the fittest" [8]. Genetic algorithms can be classified as an evolutionary algorithm in which potential solutions to a particular problem are coded in form of data structures similar to the chromosome, and applies recombination functions on these structures to protect vital information.

Accordingly, in this paper, a method based on imperialist competitive algorithms [9] to solve the problem of scheduling is provided. This algorithm is classified in innovative algorithms which are based on population, and therefore it does not have the problems of traditional systems. In addition, the proposed method uses local search algorithm to avoid being trapped in a local optimum. For this reason, this algorithm is more intelligent than the genetic algorithm in local optimum conditions. In addition, the proposed algorithm requires no specific hardware to run. And in regard to time issues it can do so much better than ant colony optimization algorithms.

Imperialist competitive

Figure 1 shows imperialist competitive algorithm flowchart [9]. Like other evolutionary algorithms, this algorithm begins with some random initial population, each of them called a "country". Some of the best elements of the population (equal to elites in genetic algorithm) are selected as imperialist. The remaining population is considered as colony. Depending on imperialists' power, they drag the colonies through a special process. The power of each empire depends on both parts; the imperialists' country (the core) and its colonies. In mathematics, the dependence is modeled by defining empire's power as the sum of the imperialist's empire, plus a percentage of the colonial power average.

By forming empires, imperialist competitive between them begins. Every empire that fails to compete and increase its power (or at least prevent the decline of its influence), will be removed from the scene of imperialist competitive. Thus survival of an empire depends on its ability to attract rival empires' colonies and ruling. As a result, during the imperialist competitive, the power of larger empires is gradually increased. And weaker empires will be removed. Empires will be forced to develop their own colonies to increase their power. Over time, the colonies, in terms of power will be closer to the empires, and we will see a convergence. The end of imperialist competitive is when we have a united empire in the world, with colonies that are very close to imperialist countries in manner of position.

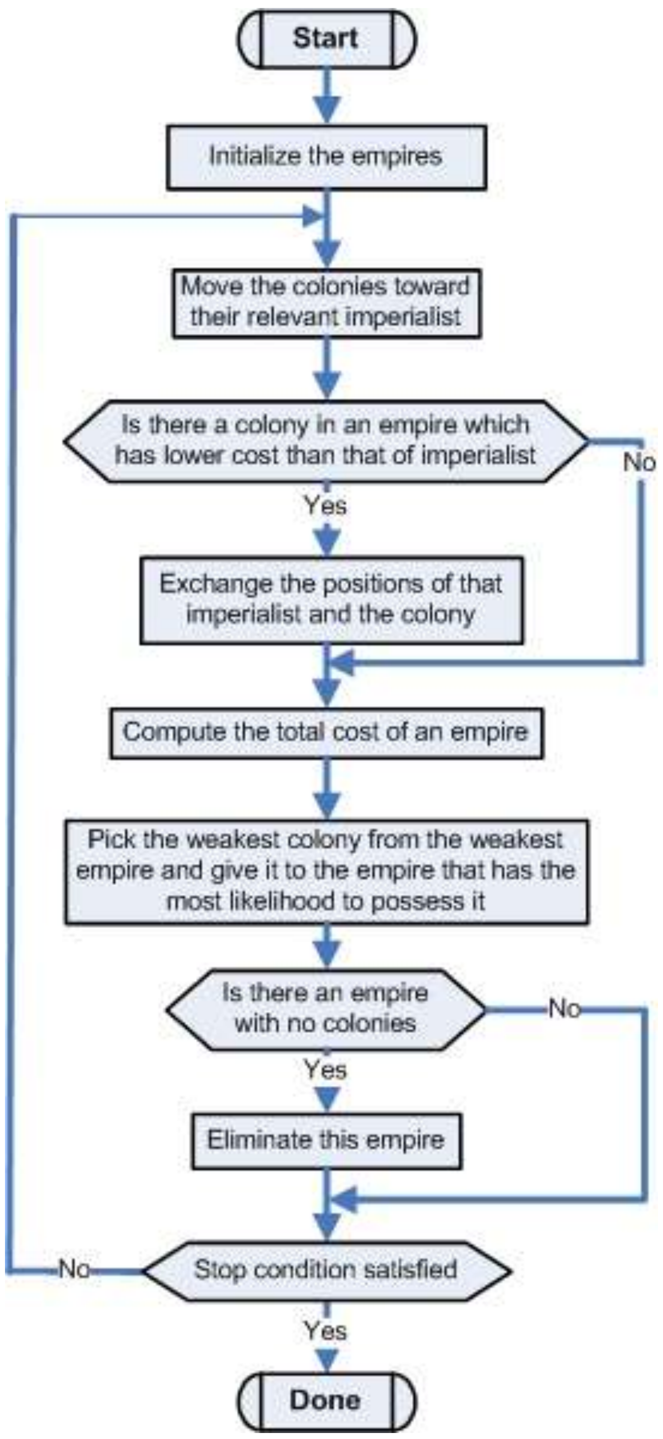
Start
Initialize the empires
Move the colonies toward their relevant imperialist
Is there a colony in an empire which has lower cost than that of imperialist
No
Yes
Exchange the positions of that imperialist and the colony
Compute the total cost of an empire
Pick the weakest colony from the weakest empire and give it to the empire that has the most likelihood to possess it
Is there an empire with no colonies
No
Yes
Eliminate this empire
No
Stop condition satisfied
Yes
Done

Figure 1. Imperialist competitive algorithm flowchart [9].

Coding

The permutation encryption algorithm is used in the proposed algorithm. The coding is composed of two phases: 1) for each m_i machine, S_i sequence of tasks that are assigned to the machine is created, and 2) the sequence of the tasks created in the first stage, are joined to each other. Result of these two steps is the permutation of the tasks that have been assigned to the machines. Figure 2 shows an example of this coding for 10 tasks and 5 machines. The code consists of two parts: 1) the scheduling and tasks allocation, and 2) related data structures to show number of tasks allocated to each machine.

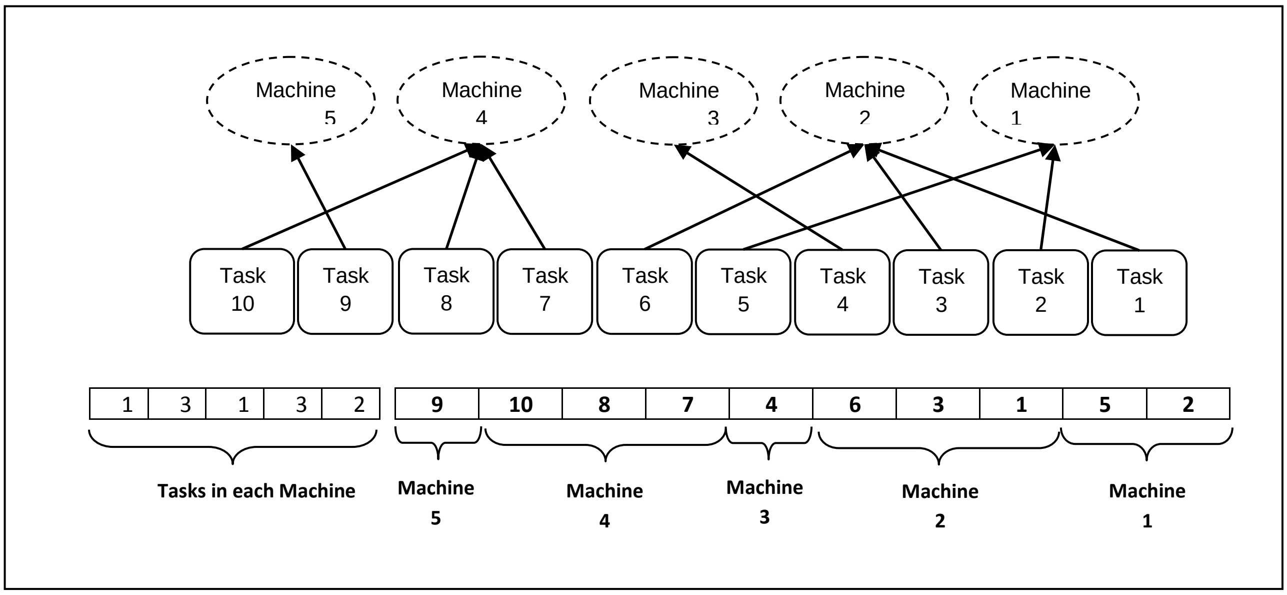

Figure 2. Example of permutative coding for the scheduling of 10 tasks on 5 machines.

Shaping the primary empires

In optimization, the goal is to find an optimal solution based on variables of the problem. We create an array of problem variables that must be optimized. In genetic algorithm the array is called chromosomes; we call it here a "country".

To start the algorithm, we create $N_{country}$ number of countries. N_{imp} best members of the population (countries with the lowest cost function) are picked as imperialists. The rest N_{col} countries form colonies, each belonging to an empire. Then we assign colonies to the imperialists according to imperialists' power. To do this, through the cost of all the imperialists, the cost of normalizing will be considered as the following:

$$C_n = \max_i\{c_i\} - c_n \qquad (1)$$

In which c_n is the cost of the imperialist n-th, $\max_i\{c_i\}$ is the highest cost of the imperialists and C_n is the cost of the normalized imperialist, respectively. Any imperialist with significant higher costs (lower is imperialist), has less normalization costs. Bu obtaining normalization cost, normalized relative strength of the imperialist is calculated as follows. And based on this issue colonial countries are distributed between imperialist.

$$p_n = \left|\frac{C_n}{\sum_{i=1}^{N_{imp}} C_i}\right| \qquad (2)$$

On the other hand, the power of an imperialist normalization is the ratio of the colonies governed by the imperialist. The initial number of an imperialist's colonies will be equal to:

$$N.C_n = round\{p_n.(N_{col})\} \qquad (3)$$

In which $N.C._n$ is the number of colonies of an empire as N_{col} is the total number of colonial countries in early countries population.

Figure 3 shows how the early empires are shaped. As shown in this figure, larger empires have more colonies. In this figure, it imperialists 1has created the most powerful empire has the most number of colonies.

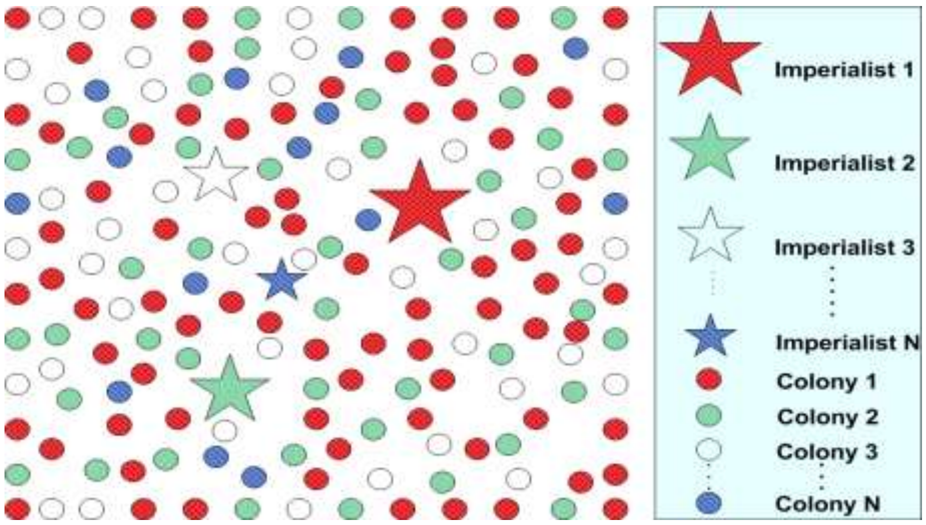

Figure 3. The formation of primary empires.

Modeling assimilation policy: the colonies movement to the imperialist

Assimilation policy aims to analyze culture and social structure of colonies in the central government's culture. With regard to the method of representing a country in solving an optimization problem, in fact, the central government tried to use assimilation policy in order to make social and political dimensions of the colony move closer. This part of colonization process in the optimization algorithm is modeled in form of the colonies movement to the imperialist countries, the model is. Figure 4 shows the overview of this movement.

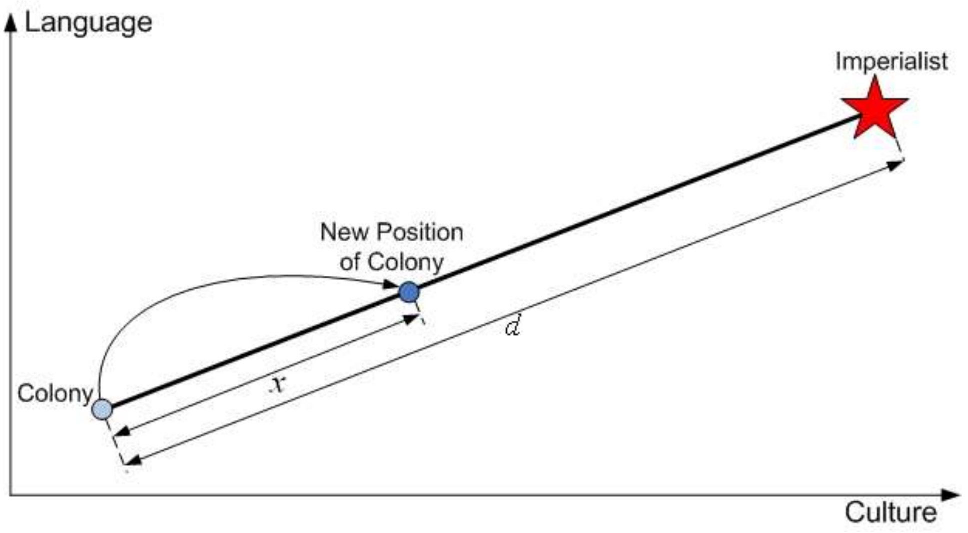

Figure 4. Colonies movement to the imperialist.

According to this figure, imperialist country assimilates the colonial country in regard to culture and language dimensions. As shown in this figure, colonial country moved x unit, and is brought to a new location. In this figure, the gap between the colonizer and the colonized is shown d . x is a random number with uniform distribution (or any other distribution). That is, for any x we have:

$$x \equiv U(o, \beta \times d) \qquad (4)$$

In which β is a number greater than one and close to 2. The proposed algorithm it is considered $\beta = 2$. Because $\beta > 1$ makes the colonial country get close to colonizer from different directions during the movement.

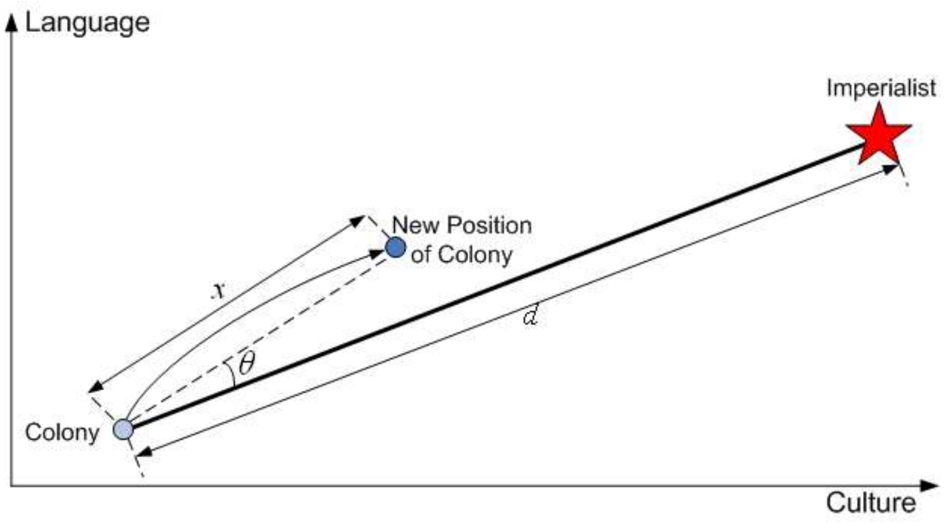

Figure 5. The real colonies' movement towards imperialist.

In colonies' movement toward colonizer, we add a little bit random angle to the direction movement. Figure 6 shows this case. Therefore, this time, instead of moving x unit to the side of the colonizer and colonizer vector direction, in the same amount, but with θ shift in the direction we continue the movement. We determine θ randomly and with uniform distribution (but any other desired and suitable distribution can also be used). So:

$$\theta \equiv U(-\gamma,\gamma) \qquad (5)$$

γ is an optional parameter. Its increase leads to an increase in searching around imperialists its increase leads colonies move close to vector as much as possible. Considering the radian units for θ a number close to π /4, is a good choice in most implementations. And therefore, in the proposed algorithm, this value is considered the same amount.

Colonial and imperialist displacement

Some of these colonies may reach to better position than their imperialist (the points in the cost function in which produce less cost than the amount of cost in imperialist position). In this case, the colonizer and the colonized will change positions with each other, and the algorithm is continued with the colonizer in the new position, and this time, the new imperialist country that begins to apply assimilation policy on its colonies. Of the colonizer and the colonies is shown in Figure 6. Figure 7 shows the whole empire after changing positions.

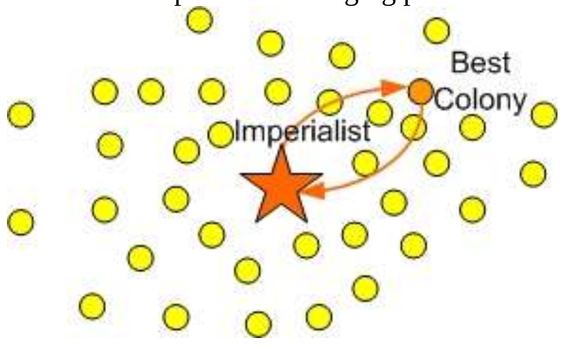

Figure 6. Changing positions of the colonizer and the colonized.

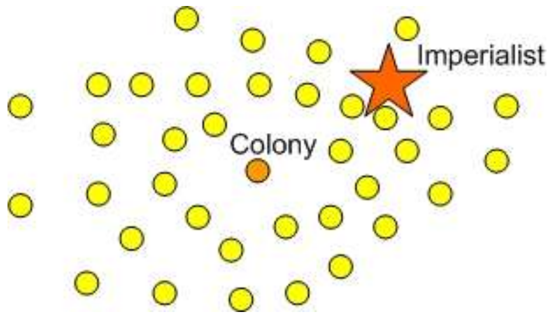

Figure 7. The empire, after changing positions.

Empire's Power is equal to the power of an imperialist country, plus a percentage of total power of its colonies. Thus, the total cost of empire is:

$$T.C._n = Cost(imperialist_n) + \xi \, mean\{Cost(colonies\, of\; empire_n)\} \quad (6)$$

In which $T.C._n$ is the total cost of Empire n-th and ξ is positive integer which is usually between zero and one and is considered close to zero. Considering ξ small, makes the cost of an empire, almost equal to the cost of the central government (the imperialists). And ξ increase, also increases the effects of empire's colonies cost in determining the total cost. In the proposed algorithm $\xi = 0.05$, since this value led to a more efficient response in most implementations.

Local Search

When applying population-based algorithms such as genetic algorithms and imperialist competitive algorithms on a problem, two scenarios may occur. First, the algorithm may converge to the global optimum solution. In this way the algorithm's performance will be improved over time, and will generate acceptable results. Second, the algorithm converges to local optimum solutions. In this way the algorithm's performance will be severely degraded, and the produced results are not reliable [11]. Considering scheduling and resources allocation in cloud computing is highly non-linear, the risk of second scenario is very likely. To overcome this problem proposed method uses a method based on local search. This algorithm works in such a way that, every time solutions converge to a local optimum, the algorithm creates some random countries, and adds them to the countries set. This way the algorithm finds a way to exit local optimum.

In the proposed algorithm, iterative hill-climbing method is used for finding random solutions, which was introduced by Michalewicz [10] Hill-climbing method structure is shown in figure (8).

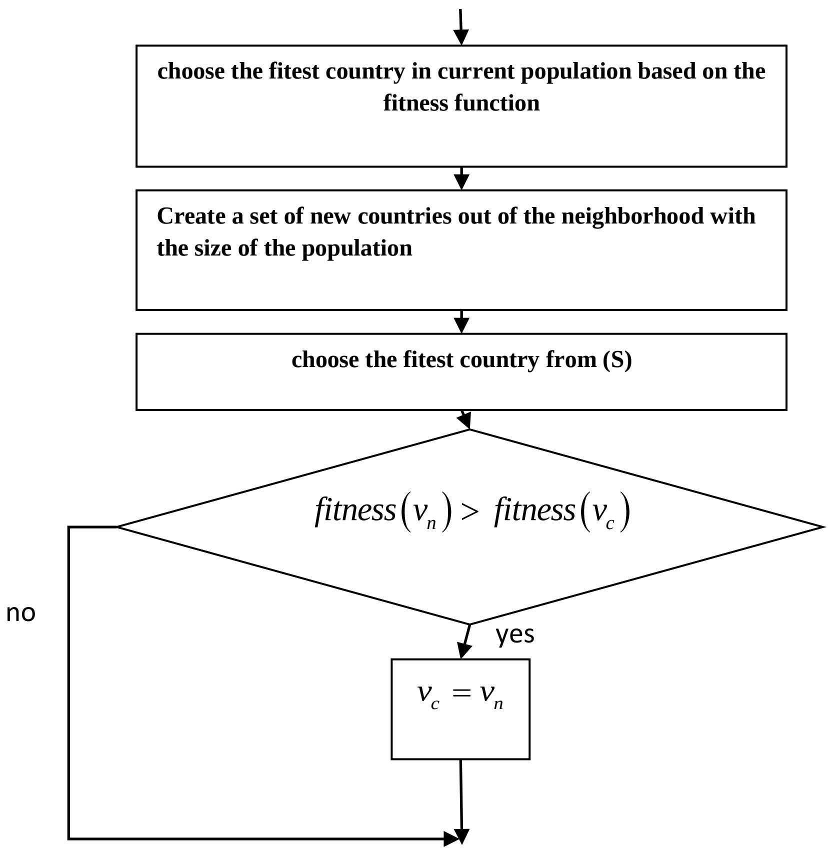
choose the fitest country in current population based on the fitness function
Create a set of new countries out of the neighborhood with the size of the population
choose the fitest country from (S)
$fitness(v_n) > fitness(v_c)$
no
yes
$v_c = v_n$

Figure 8. Iterative hill-climbing method introduced by Michalewicz.

Local search control mechanism

The more genetic algorithm converge to a solution, the more increase occur in population. And thus the values of competency become more similar, and variety of population decreases. The proposed algorithm uses genetic algorithm to control applying time of hill-climbing algorithm logic. In this method, a mechanism based on similarity coefficient method is used to measure the similarity between population's members.

Suppose $M = [v_{1m}, v_{2m}, \ldots, v_{Nm}]$ and $L = [u_{1l}, u_{2l}, \ldots, u_{Nl}]$ are two countries of the current empire. The similarity coefficient between L and M, are (6-5):

$$SC_{lm} = \frac{\sum_{k=1}^{N} \partial(v_{km}, u_{kl})}{N_1} \tag{7}$$

In which k the position of the gene, v_{km} is the gene's k-th value of the country m, u_{kl} is the gene's k-th value of the country l, and N_1 is the number of genes. The value of $\partial(v_{km}, u_{kl})$ can be stated in formula (7):

$$\partial(v_{km}, u_{kl}) = \begin{cases} 1 & ,if\, |v_{km} - u_{kl}| \leq \alpha \\ 0 & ,otherwise \end{cases} \tag{8}$$

In which α is a predetermined coefficient to measure the similarity of the two chromosomes. With regard to countries coding on the issue of scheduling includes positive integer values, $\alpha = 0$ in the proposed algorithm.

The similarity coefficient of population $\left(\overline{SC_{lm}}\right)$ is formulated as the equation (10-5):

$$\overline{SC_{lm}} = \frac{\sum_{l=1}^{N_{country}-1}\sum_{m=1}^{N_{country}} SC_{lm}}{n_1} \quad (9)$$

In which $N_{country}$ is the size of the population and n_1 is total SC_{lm}.

Now, hill-climbing method can be formulated (10):

$$\begin{cases} apply\ iterative\ hill\ climbing\ method\ to\ ICA\ loop, & if\ \overline{SC_{lm}} \geq \beta \\ \text{apply ICA alone} & , \text{otherwise} \end{cases} \quad (10)$$

In which β is a predefined threshold.

Criteria

To evaluate the proposed method, four criteria in Table 1 together with optimization periods for criteria scheduling are presented. Criteria have been selected through OR Library [12]. There a lot of criteria in this library. But they all do not have certain optimal scheduling. In other words, still no optimal scheduling is found. These four criteria, in addition to have a certain optimum scheduling, cover a range of criteria in the library, in terms of hardness. FT10 and FT20 were introduced first in [13]. Because of researchers' high interest, they as test questions were accepted by everyone. LA40 is a kind of a confusing problem, because the number of tasks and the number of machines are a lot [14]. SWV14 [15] is also a very difficult problem, which includes 50 tasks.

Table 1. Lists of criteria used in evaluation.

name	tasks	machines	optimization cost
FT10	10	10	930
FT20	20	5	1165
LA40	15	15	1222
SWV14	50	15	2968

Various parameters of the proposed algorithm

The proposed algorithm is a combination of imperialist competitive and hill-climbing method algorithms. In this algorithm, the problem of scheduling and resource allocation is resolved using imperialist competitive algorithm and creating a $N_{countary}$ number of initial country. When the imperialist competitive algorithm stuck in premature convergence and local optimal, convergence diagnosis mechanism recognizes the convergence base on the average similarity of countries. If the value of similarity is greater than a threshold ofβ, with the implementation of hill-climbing algorithm, it tries to add a number of new countries to the world. According to this explanation, the most important parameters that affect the proposed hybrid algorithm, is the value of threshold β parameter. In this section we examine the effect of different values of this parameter on the performance of the proposed algorithm in terms of algorithm's running time, quality of (cost) scheduling and time of algorithm's convergence. Algorithm execution time is equal to CPU time for 200 generation of running algorithm. Scheduling quality is calculated based on the country's competency, which is calculated as follows.

$$fitness = \frac{makespan + mean(flowtime)}{2} \quad (11)$$

In which $makespan$ represents the finished time of last task and $mean(flowtime)$valso shows the average execution time of all tasks. The algorithm convergence time, is when the empire's competency do not change more than 0.001.

The number of countries in FT10 is 100 countries, and 150 for FT20, 200 for LA40, and 300 for SWV14, 300.

Figures 9 to 12 show the result of the proposed algorithm with different values ofβ.

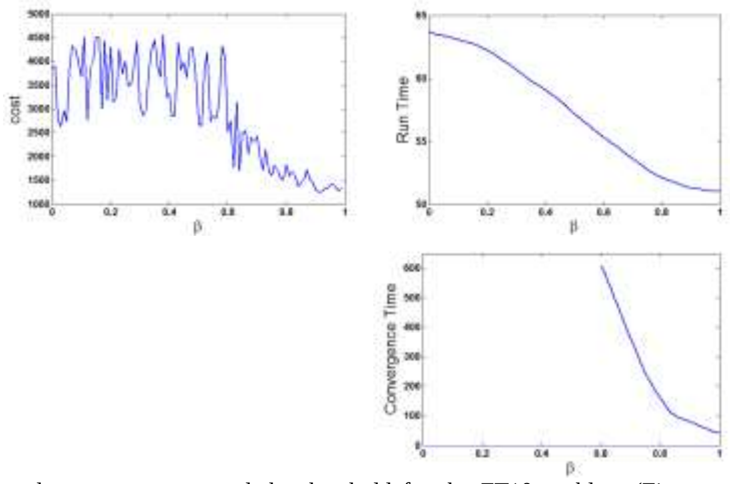

Figure 9. Changes in the cost, time, and convergence toward the threshold for the FT10 problem (Figures are arranged from right to left).

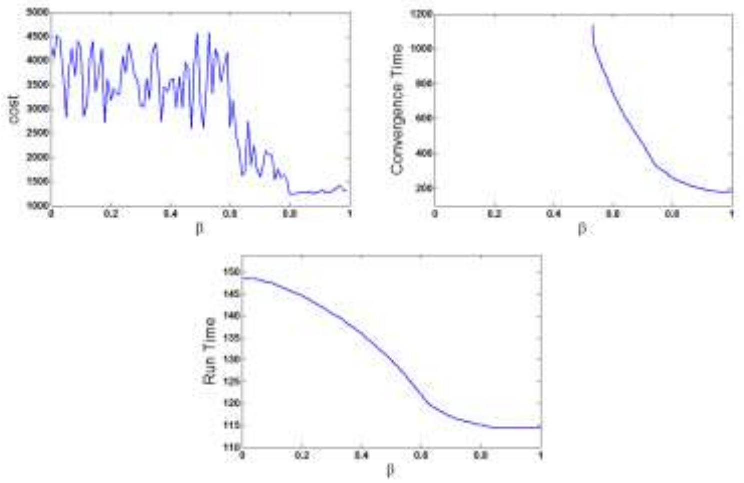

Figure 10. changes in the cost, time, and convergence toward the threshold for the FT20 problem.

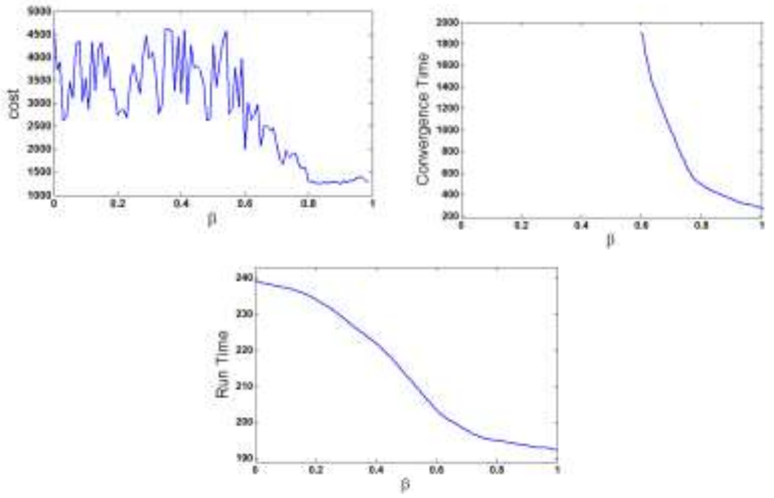

Figure 11. Changes in the cost, time, and convergence toward the threshold for the LA40 problem.

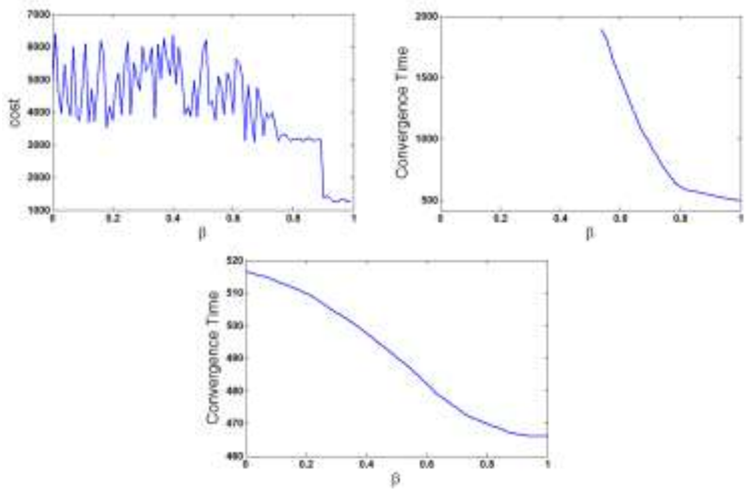

Figure 12. Changes in the cost, time, and convergence toward the threshold for the SWV14 problem.

In figures 9 to 12:

1. The proposed method finds the most economical and best answer for the $\beta \in [0.8,0.9]$ values . The closer threshold is to zero, the frequency of local search calls are more. As a result, random steps in the algorithm will increase. Increased random steps of the algorithm reduce the possibility of convergence. As a result, the proposed algorithm does not find a good answer for small values of the threshold. On the other hand, if threshold be closer to one, decreases the frequency of local search calls. Thus, the proposed algorithm works like a simple imperialist competitive algorithm.
2. Implementation time of the proposed algorithm does not change much according to the threshold value. The proposed algorithm consists of three main parts: imperialist competitive algorithm, similarity control mechanism and repetitive hill-climbing. Different threshold values do not have a strong influence on the first two stages, because they will run in all circles of the proposed algorithm. But repetitive hill-climbing runs only when the similarity population value crosses the value of threshold. Thereby by reducing the threshold, the call frequency of repetitive hill climbing increases. As a result, algorithm execution time increases, through threshold reduction. But this increase is not a significant increase, since execution time of repetitive hill climbing described in Figure 8, is$O(|P|)$. In which $|P|$ is the number of countries.

Comparing the proposed algorithm with other algorithms

In this section, the proposed hybrid algorithm is compared with round-robin algorithm, genetic algorithm and ant colony optimization algorithms. $\beta = 0.85$ for the proposed algorithm. 50 ants have been used in ant colony optimization algorithms for various moves' direction. For genetic algorithm the possibility of cutting equal to 0.9, mutation probability of 0.05 were considered. The number of initial population is considered equal to the number of countries in imperialist competitive algorithm (The number of countries in FT10 is 100 countries, and 150 for FT20, 200 for LA40, and 300 for SWV14, 300). Compared criteria include the algorithm implementation runtime, the quality of obtained results in terms of $makespan$ and the quality of obtained results in terms of $mean(flowtime)$. Round-robin algorithm ends as soon as it finds the answer. But the other three algorithms will continue as long as over 10 consecutive performances, no improvement in response is done. Each algorithm has been run 20 times and all criteria have been achieved on average. All algorithms have been implemented on a system with triple-core 2.8 GHz clock speed system and Windows 7 with 2 GB of main memory.

Comparing runtimes

Figure 5-5 shows compared algorithms' run for the four criteria. As it can be seen, round robin algorithm's runtime, compared with three other algorithms, is very low. This is due to the systematic and greedy structure of the algorithm. In other words, this algorithm returns the first possible answer, while other algorithms return the best answer among feasible answers. Ant colony optimization algorithms take more time, because it uses collective intelligence. Due to the fact that this method has produced a number of artificial ants and the benefits all of their results as parallel, serial implementation of the algorithm has low efficiency. The proposed algorithm in terms of the speed is faster than the genetic algorithm and the ant colony optimization algorithm.

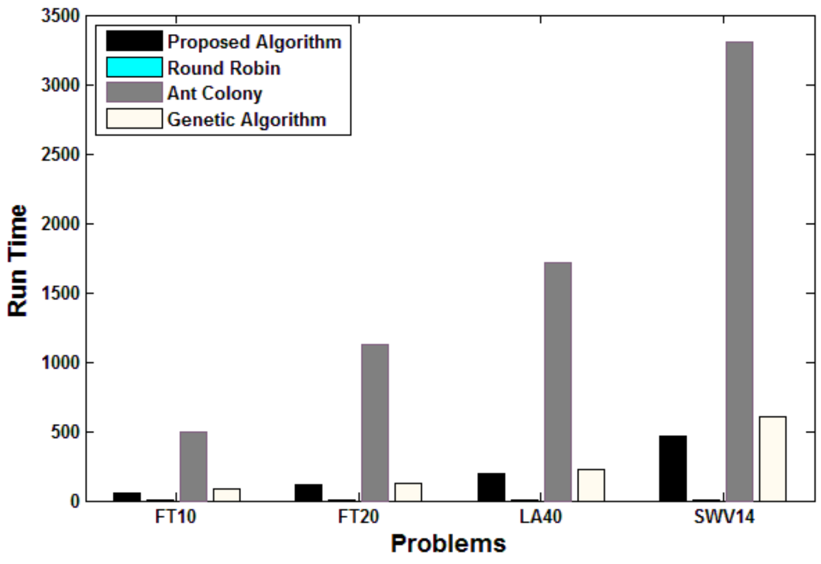

Figure 13. Comparing runtimes.

Comparing solutions' quality

Figures 14 and 15 show the results of the algorithm in form of completion time of all tasks and the average time of each algorithm. As can be seen, although round-robin algorithm is very fast, the quality of the answers is much lower than other algorithms. The result of proposed algorithm in terms of quality is comparable with the ant colony optimization algorithm. But the remarkable thing is that the proposed algorithm is much faster than ant colony optimization algorithm.

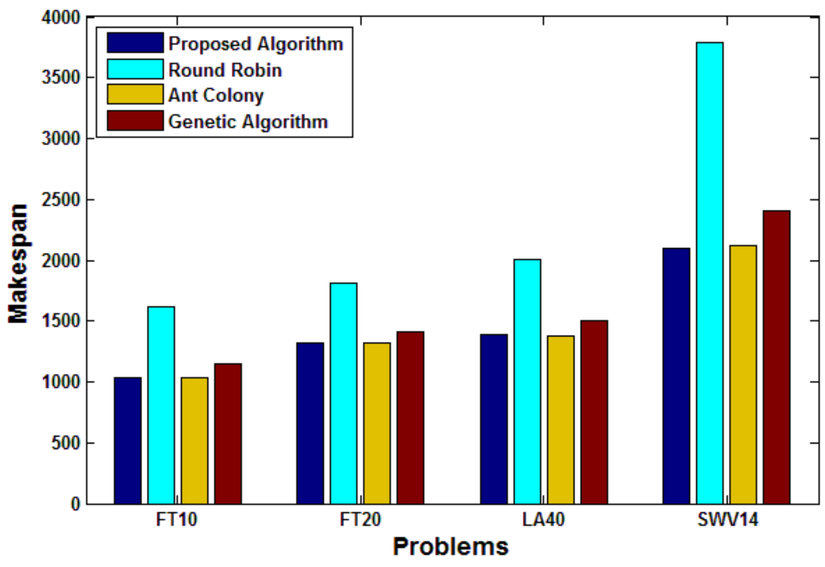

Figure 14. Comparing algorithms in terms of last tasks' completion time.

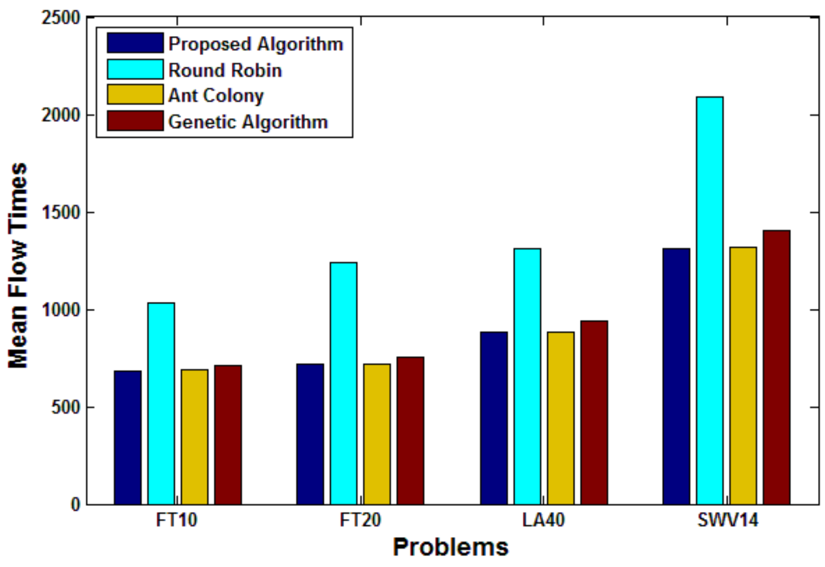

Figure 15. Comparing algorithms in terms of average tasks' completion time.

CONCLUSION

Resource allocation in cloud computing and the scheduling of each user works on existing virtual machines is an NP-Complete problem, in which many algorithms are provided for solving the problem so far. In this paper, a combination of imperialist competitive and local search optimization algorithms is proposed to solve this problem. Quality and efficiency of the proposed algorithm are compared with round-robin, ant colony and genetic algorithms. The results show that the proposed algorithm in terms of time and the responses' quality is faster than the ant colony optimization and genetic algorithms. The results also show that the proposed algorithm in terms of quality, performed better. In other words, the provided scheduling by this algorithm has less cost.

REFERENCES

[1] K. R. Baker, D. Trietsch, Principles of Sequencing and Scheduling, Wiley, 2000.
[2] M. R. Garey, D. S., Johnson, R., Sethi, The complexity of flowshop and jobshop scheduling., Mathematics of operations research, Vol.1, 1976.

[3] Yu, L. Wang, Genetic Algorithm Combined with Simulation for Job Shop Scheduling Problem in Mechanical Engineering, Advances in Mechanical and Electronic Engineering, Vol. 176, pp. 139-144, 2012.

[4] L. Zhu, Q. Li, L. He, Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm, International Journal of Computer Science Issues, Vol. 9 (5), pp. 54-58, 2012.

[5] W. N. Chen, and J. Zhang, An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 39 (1), pp. 29-43.

[6] Javier, C., Fatos, X., Ajith, A.: Genetic Algorithm Based Schedulers For Gird. IEEE, International Journal of innovative Computing, Information and Control 3 (December 2007).

[7] M. Dorigo, V. Maniezzo, and A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26 (1), pp. 29-41, 1996.

[8] E., Eiben, J. E.,Smith, Introduction to Evolutionary Computing, Springer, 2007.

[9] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, Singapore, pp. 4661-4667, 2007.

[10] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Program, Spring-Verlag, 1994.

[11] Y.H. Chang, Adopting co-evolution and constraint-satisfaction concept on genetic algorithms to solve supply chain network design problems, Expert Systems with Applications, vol. 37, pp. 6919–6930, 2010.

[12] J., Beasley, OR-Library: Distributing test problems by electronic mail, Journal of the Operational Research Society, Vol. 41, pp. 1069–1072, 1990.

[13] J., Muth, G., Thompson, Industrial scheduling, Prentice-Hall, 1963.

[14] S., Lawrence, Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement). Pittsburgh, PA: Carnegie-Mellon University, 1984.

[15] R. H., Storer, S. D., Wu, R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, Management Science, Vol. 39, pp. 1495–1509, 1992.