Priority-based task scheduling on heterogeneous resources in the Expert Cloud

Mehran Ashouraie and Nima Jafari Navimipour
Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran

Abstract
Purpose – Expert Cloud as a new class of Cloud systems provides the knowledge and skills of human resources (HRs) as a service using Cloud concepts. Task scheduling in the Expert Cloud is a vital part that assigns tasks to suitable resources for execution. The purpose of this paper is to propose a method based on genetic algorithm to consider the priority of arriving tasks and the heterogeneity of HRs. Also, to simulate a real world situation, the authors consider the human-based features of resources like trust, reputation and etc.
Design/methodology/approach – As it is NP-Complete to schedule tasks to obtain the minimum makespan and the success of genetic algorithm in optimization and NP-Complete problems, the authors used a genetic algorithm to schedule the tasks on HRs in the Expert Cloud. In this method, chromosome or candidate solutions are represented by a vector; fitness function is calculated based on several factors; one point cross-over and swap mutation are also used.
Findings – The obtained results demonstrated the efficiency of the proposed algorithm in terms of time complexity, task fail rate and HRs utilization.
Originality/value – In this paper the task scheduling issue in the Expert Cloud and improving pervious algorithm are pointed out and the approach to resolve the problem is applied into a practical example.
Keywords Information systems, Networking, Optimization techniques, Knowledge management, Social networks, Telecommunications
Paper type Research paper

1. Introduction
Resources such as software and information are shared through the network and retrieved by computers and devices on demand (Yigit *et al.*, 2014). Cloud computing offers a promising new platform to execute large programs. In addition to providing multiple Virtual Machines (VMs) to execute tasks which are contained in a program, Cloud computing also offers on-demand scaling and pay-per-use metered service. That is, computing resources are dynamically allocated to user programs based on needs, and users just pay for the resources their programs actually consume, similar to the conventional pay-per-use metered service for utility consumptions of water, electricity and natural gas (Su *et al.*, 2013). Cloud computing offers many types of services such as Infrastructure as a Service (Wickboldt *et al.*, 2014; Manvi *et al.*, 2014), Platform as a Service (Anselmi *et al.*, 2014) and Software as a Service (Zheng *et al.*, 2011; Wu *et al.*, 2012). Also, Expert as a Service (EaaS) is a new type of service in the Cloud computing which enables its users to request the skill, knowledge and expertise of people without any information of their location by employing internet infrastructures and Cloud computing concepts (Jafari Navimipour, 2015; Jafari Navimipour *et al.*, 2015a, b).

Expert Cloud is a new class of Cloud systems that provides the knowledge and skills of human resources (HRs) as a service using Cloud concepts. The Expert Cloud enables

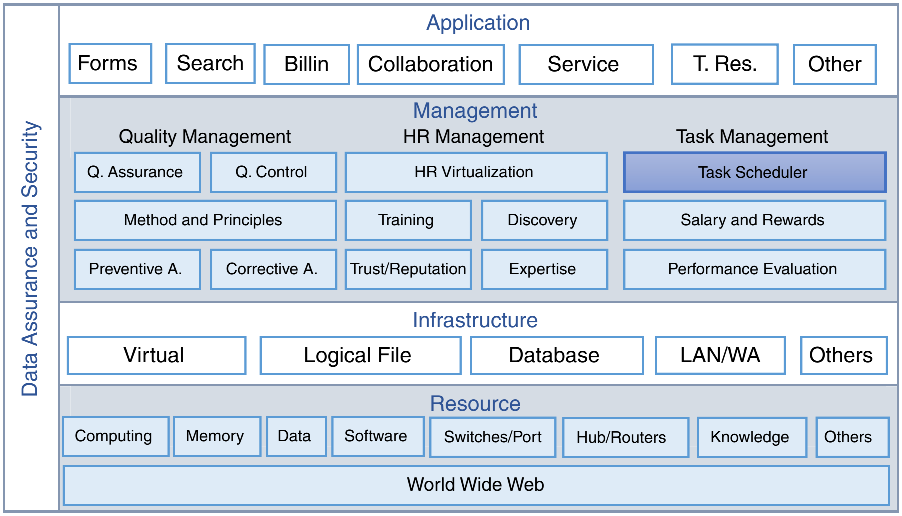
Kybernetes
Vol. 44 No. 10, 2015
pp. 1455-1471

0368-492X
DOI 10.1108/K-12-2014-0293

all human societies to share the knowledge, skills and experiences of HRs to meet the demands. In the Expert Cloud there are many HRs with different profession and expertise. These resources which are called Virtualized-HRs are spread over geographic, governmental and organizational boundaries and collaborate together via the Expert Cloud.

EaaS, as a primary service in the Expert Cloud provides customers the transparent access to expert, knowledge and skills of HRs remotely over the internet via a layered structure that corresponds to the Cloud architecture. The Expert Cloud architecture consists of four layers: application, management, infrastructure and resource layer (see Figure 1). A web-based application of the Expert Cloud is ready to use which is developed by using PHP[1] and MySQL.

Task scheduling is a vital part of any distributed system like Grid, Cloud and P2P networks (Jafari Navimipour and Sharifi Milani, 2014) which assigns tasks to suitable resources for execution. However, it is NP-Complete to schedule tasks to obtain the minimum makespan (Su *et al.*, 2013; Jafari Navimipour and Mohammad Khanli, 2008). A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks (Tawfeek *et al.*, 2013). Also, priority of tasks is an important issue in scheduling because some tasks should be serviced earlier than other those tasks cannot stay for a long time in a system. Therefore a suitable task scheduling algorithm must consider priority of tasks (Ghanbari and Othman, 2012). The goal of a task scheduling algorithm is scheduling all the subtasks on a given number of available resources in order to minimize makespan without violating precedence constraints (Xu *et al.*, 2014). In the Expert Cloud, efficient task scheduling mechanism can meet users' requirements and improve the resource utilization, therefore it enhances the overall performance of the Expert Cloud. In other words, task scheduling is one of the important issues in the Expert Cloud and impacts on its efficiency and customer satisfaction (Jafari Navimipour *et al.*, 2014). The optimal trade-off between the HR and task demands is a challenging problem (Navin *et al.*, 2014). The pervious GA-based algorithm was proposed by Jafari Navimipour *et al.* (2014) to



Source: Jafari Navimipour *et al.* (2015b)

Figure 1.
Layered architecture of the Expert Cloud

schedule independent tasks with a complexity of *O (n)* on homogeneous HRs. The main contributions of this paper are fourfold:

- improving complexity of proposed algorithm by Jafari Navimipour *et al.* (2014) to *O (1)*;
- considering priority of arriving tasks;
- considering heterogeneity of HRs; and
- engaging human-based features of resources like expertise, reputation and agility to simulate a real world situation.

The rest of this paper is organized as follows: the related works and background are reviewed in Section 2; Section 3 deals with the proposed method; experiments and results are presented in Section 4; and in the last section we will conclude the paper and suggest some indications for future researches.

2. Related works

There are many papers that address the problem of scheduling in distributed systems, like Grid, and multiprocessor systems whereas there are a few works on this problem in the Clouds and only one work in the Expert Cloud is done. The multi-objective nature of the scheduling problem in Clouds makes it difficult to solve, especially in the case of complex tasks (Abrishami and Naghibzadeh, 2012). Some papers are reviewed as follow:

Su *et al.* (2013) have presented a cost-efficient task scheduling algorithm using two heuristic strategies. The first strategy dynamically maps tasks to the most cost-efficient VMs based on the concept of Pareto dominance. The second strategy, a complement to the first strategy, reduces the monetary costs of non-critical tasks. The simulation results showed that the algorithm can substantially reduce monetary costs while producing makespan as good as the best known task scheduling algorithm can provide. However, this algorithm is inapplicable to multiple types of VMs with different pricing models.

Mateos *et al.* (2013) have proposed a new Cloud scheduler based on Ant Colony Optimization (ACO). The goal of scheduler is to minimize the weighted flowtime of a set of PSE jobs, while minimizing makespan. Simulated experiments performed with real PSE job data and other Cloud scheduling policies indicate that this proposal allows for a more agile job handling while reducing PSE completion time. Also, the evaluation results showed that ACO performs better than random and best effort algorithms.

Wang *et al.* (2012) have proposed a trust dynamic level scheduling algorithm named Cloud-DLS by integrating the existing DLS algorithm. The main contribution of this study is extending the traditional formulation of the scheduling problem so that both execution time and reliability of applications are simultaneously accounted. Theoretical analysis and simulations proved that the Cloud-DLS algorithm can efficiently meet the requirement of Cloud computing workloads in trust, sacrificing fewer time costs and assuring the execution of tasks in a security way.

Ghanbari and Othman (2012) have proposed a new priority-based job scheduling algorithm in cloud computing based on multiple criteria decision-making model and the theory of AHP. This scheduling algorithm is consisted of three levels of priorities including: scheduling level (objective level), resources level (attribute level) and job level (alternative level). The algorithm calculates priority vector of scheduling jobs (PVS) then chooses a job with maximum amount of priority value based on PVS and allocates suitable resources. Result of this paper indicated that the proposed algorithm has reasonable complexity.

Up to now, there is just one specialized paper which studied the scheduling issue in the Expert Cloud (Jafari Navimipour *et al.*, 2014), in which independent tasks are scheduled on homogeneous resources by a GA-based algorithm. A set of tasks arrived in a period $BT = (T_1, T_2, \ldots, T_n)$ with a service request for each task that is represented as $(T_i, H_i, E_i, A_i, R_i, D_i)$, where T_i is the time that the request is received, H_i is a type of required HR, E_i, A_i and R_i represent the minimum value of expertise, agility and reputation (in the range of [0, 1]) that T_i want, and D_i is the deadline. To find a suitable scheduling scenario, a population of solutions is created and genetic operators such as mutation and cross-over are applied. The method selects one of N produced chromosome using search algorithms of *O (N)* or *O (logN)* complexity. After that, in each iteration the best individuals are selected and the worst ones are replaced with new generated ones. The algorithm is terminated if no further improvement in the fitness value of the best chromosome in the population occurs (Jafari Navimipour *et al.*, 2014).

3. Proposed method

In this section, we propose a genetic-based approach for task scheduling in the Expert Cloud where the resources are HRs. First, a directed acyclic graph (DAG) is introduced for modeling the problem. Then, we describe the algorithm and improve it using stochastic acceptance (Lipowski and Lipowska, 2012) with *O (1)* complexity. Also, we assume that the resources are heterogeneous and the tasks have precedence order. Heterogeneity of HRs refers to the different capability of HRs that execute tasks. This helps us to model a realistic Expert Cloud situation.

3.1 DAG model

Tasks with precedence order in the Expert Cloud can be modeled as a DAG, $G = (V, E)$, where V is the set of v tasks to be executed and E is the set of e edges representing the precedence constrains between tasks. Figure 2 illustrates a sample of seven tasks with eight precedence relation among them. To start processing T1 and T2, first T0 must be completed. Also, execution of T3 and T4 depends on completion of T1 and execution of

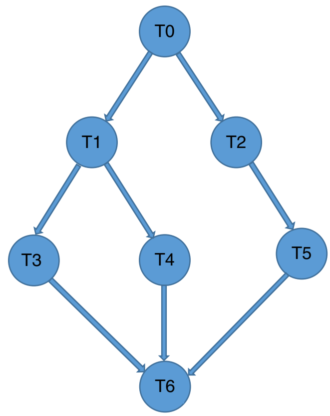


Figure 2.
Sample DAG model for the Expert Cloud tasks

T5 depends on completion of T2. At the end, T6 cannot be started unless all T3, T4 and T5 are completely done.

The weight of a node v_i, denoted by w_{v_i}, represents the execution time of the task v_i on each HRs as an array (see Table I). For example HR4 claims to execute T1 in six cycles of time unit and the same task can be done faster by HR2 in five cycles. If an HR is unable to execute a given task the weight becomes null.

3.2 Proposed method

An improved genetic algorithm is used to achieve an accurate schedule to process multiple tasks in the Expert Cloud. At first, a random initial population of chromosomes are generated. Then, cross-over and mutation operators are performed to generate new populations. Finally, survival individuals are selected according to the defined fitness function. If the termination criteria have not been reached, the final survival individuals are used in cross-over and mutation operator and new population is generated until the termination criteria is reached.

3.2.1 Chromosome. Genetic algorithm starts with a population of strings (represented by chromosomes), which encodes candidate solutions called individuals. Each chromosome is represented by an array with the length of j, where j is the height of a given DAG and with width of i, where i is the number of HRs (see Table II). Each element of array contains index of a task that will be executed. Also, parent(s) of each task must be saved as an additional information in order to avoid violating precedence constraints.

For five HRs and six tasks, a sample chromosome might be an array represented in Table II showing that at the first level of DAG, T0 is executed by HR1 while other HRs and tasks are just waiting. Here, T0 has no parent, so it can be executed independently. At the second level of DAG, two tasks, T1 and T2, are waiting for completion of T0. When T0 is successfully done, T1 and T2 could start. At the third level, T3 and T4 are waiting for T1 and T5 waits for T2 to be completed. Finally, T6 will be executed by HR2 when all three parents, T3, T4 and T5 are completely done. This array represented a solution for the given DAG.

HR No.	Task 0	Task 1	Task 2	Task 3	Task 4	Task 5
HR 1	4	9	**N**	3	6	4
HR 2	5	5	9	**N**	4	6
HR 3	9	12	**N**	11	11	**N**
HR 4	**N**	6	2	1	2	4
HR 5	5	8	**N**	8	9	6

Table I. A sample execution time table for six tasks and five HRs

HR No.	Height 1	Height 2	Height 3	Height 4
HR 1	0	–	3	–
HR 2	–	1	–	6
HR 3	–	–	–	–
HR 4	–	–	4	–
HR 5	–	2	5	–

Table II. A sample chromosome for the given DAG (see Figure 2) and five HRs

3.2.2 Fitness function. Fitness of a chromosome is defined as the minimum time for fulfilling all given tasks called total execution time (TET), least missed tasks (M.T.) and higher Reputation (Rep.), Agility (Agi.), Expertise (Exp.). Fitness is defined to evaluate efficiency of a built chromosome as a solution to a given DAG. At the end of this algorithm a chromosome with highest fitness will be the best found solution:

$$\boldsymbol{FF} = \alpha\left(\frac{1}{(TET of all\ leaves) \times M.T.}\right) + \beta\left(\gamma_1 \times Rep. + \gamma_2 \times Agi. + \gamma_3 \times Exp.\right) \quad (1)$$

where:

$$\alpha + \beta = 1 \quad (2)$$

$$\gamma_1 + \gamma_2 + \gamma_3 = 1 \quad (3)$$

and TET can be calculated by a recursive function as:

$$\mathbf{TET_{leaf}} = \begin{cases} w_{v_i} & \textit{if leaf } v_i \textit{ has no parent} \\ max(TET\ of\ each\ parent) + w_{v_i} & \textit{otherwise} \end{cases} \quad (4)$$

3.2.3 Initial population. The initial population defines the speed and the convergence of the genetic algorithm (Abdoun *et al.*, 2012). We uses random method to initial the population with 50 chromosomes (population size) in which satisfy the criteria of problem. The initial population is generated randomly after that, in each iteration the best individuals are selected and the worst ones are replaced with new generated ones (Diaz-Gomez and Hougen, 2007).

3.2.4 Selection operator. Roulette wheel selection is a common technique used in GA implementations to select the chromosome. Existing mechanism select one of N individuals using search algorithms of $O\ (N)$ or $O\ (logN)$ complexity. We use a selection algorithm, which typically has $O\ (1)$ complexity and is based on stochastic acceptance instead of searching. The first step in the selection process is to run all of the chromosomes from the initial population through the schedule builder. After all of the chromosomes have been scheduled and scored, the sum of all of the individual fitness's is calculated which represents the total fitness for the population (Sun *et al.*, 2010).

Definition: let a population contains N individuals where their fitness $f_i > 0\ (i = 1, 2, \ldots, N)$ are calculated. The selection probability of the i-th individual is (Lipowski and Lipowska, 2012):

$$pi = \frac{wi}{\sum_{i=1}^{N} wi} \quad (i = 1, 2, \ \ldots \ , N) \quad (5)$$

(1) Select randomly one of the individuals. The selection is done with uniform probability (1/N), which does not depend on the individual's fitness W_i.

(2) With probability W_i/W_{max} where $W_{max} = \max\{W_i\}$ is the maximal fitness in the population, the selection is accepted. Otherwise, the procedure is repeated from step 1 (i.e. in the case of rejection, another selection attempt is made).

3.2.5 Cross-over operator. The cross-over operator combines more than one chromosome to generate the chromosomes of the new generation. The new chromosome inherits some features from first parent and the rest features from the other parent [9]. With definition of P_c (probability of cross-over) the cross-over operation starts. Also, a cross-over point, C_p, is defined to divide a chromosome into right and left sections. C_p is a random integer number between 1 and height of the DAG (H_{DAG}). The Pseudo code of the cross-over operation is described by these steps:

(1) define P_c in [0, 1]; set $i=j=k=0$;

(2) generate a random number $r1$ in [0, 1];

(3) i++; If $r1 < P_c$ chromosome i is a parent (first one) otherwise proceed from 2;

(4) generate a random number $r2$ in [0, 1];

(5) j++; If $r2 < P_c$ chromosome i is a parent (second one) otherwise proceed from 4;

(6) define C_p;

(7) divide each parent chromosome into two sections with C_p;

(8) swap right side of first parent with left side of second parent;

(9) if swapped chromosomes satisfy the DAG save them and k++; and

(10) set $i=j=0$; proceed from 2 if $k < pop\_size$.

For example, the cross-over operation swap first and second level of each chromosome in Table III and generates two new chromosomes (see Table IV). The newly generated ones will remain alive if they satisfy their given DAG, otherwise cross-over operation is repeated.

3.2.6 *Mutation operator*. In GA, mutation operator is used to maintain the diversity of the population by changing chromosome with a small probability from the interval [0, 1], which is known as the probability of mutation P_m. Also two mutation points M_{p1} and M_{p2} are defined to perform mutation operation. These mutation points are random integers between 1 and number of HRs. The Pseudo code of the mutation operation is described by these steps:

(1) define P_m in [0, 1]; set $k=0$;

(2) generate a random number r in [0, 1];

(3) if $r < P_m$ chromosome i is a parent otherwise go to 7;

(4) define M_{p1} and M_{p2};

HR No.	Height 1	Height 2	Height 3	Height 4
HR 1	0	–	3	–
HR 2	–	1	–	6
HR 3	–	–	–	–
HR 4	–	–	4	–
HR 5	–	2	5	–
HR 1	–	2	–	–
HR 2	–	1	–	–
HR 3	0	–	3	–
HR 4	–	–	4	6
HR 5	–	–	5	–

Table III. Two chromosomes before cross-over operation

Table IV. Two chromosomes after cross-over operation

HR No.	Height 1	Height 2	Height 3	Height 4
HR 1	–	2	3	–
HR 2	–	1	–	6
HR 3	0	–	–	–
HR 4	–	–	4	–
HR 5	–	–	5	–
HR 1	0	–	–	–
HR 2	–	1	–	–
HR 3	–	–	3	–
HR 4	–	–	4	6
HR 5	–	2	5	–

(5) swap two rows (row M_{p1} and M_{p2}) of the selected parent chromosome;

(6) if the new chromosome satisfies the DAG save it; and

(7) k++; proceed from 2 if $k < pop\_size$.

As an example, a chromosome of the DAG in Table V is being mutated. The algorithm randomly decides to swap second and fourth row of this chromosome and a new chromosome is generated (see Table VI). The newly generated chromosome will continue to live if it satisfies the given DAG.

3.2.7 Termination condition and pseudo code. The algorithm is terminated if no further improvement in the fitness value of the best chromosome in the population is not occurred for ten iterations or maximum number of generation is reached. The Pseudo code of the algorithm is described by these steps:

(1) initial *pop_size* chromosomes randomly;

(2) calculate the fitness of each chromosome;

(3) perform cross-over;

Table V. A chromosome before mutation operation

	Height 1	Height 2	Height 3	Height 4
HR 1	0	–	3	–
HR 2	–	1	–	6
HR 3	–	–	–	–
HR 4	–	–	4	–
HR 5	–	2	5	–

Table VI. A chromosome after mutation operation

	Height 1	Height 2	Height 3	Height 4
HR 1	0	–	3	–
HR 2	–	–	4	–
HR 3	–	–	–	–
HR 4	–	1	–	6
HR 5	–	2	5	–

(4) perform mutation;
(5) evaluate the fitness of the offspring;
(6) select the survive individuals; and
(7) proceed from 3 if the termination criteria have not been reached (Table VII).

4. Experimental results

The proposed task scheduler method in the Expert Cloud is implemented and evaluated on the Expert Cloud networks which is based on PHP. Detailed information of running environment (including software and hardware) is given in Table VIII. Subsection 4.1 indicates a scenario and the related data sets. Subsection 4.2 shows the obtained results in term of HR utilization rate, fail rate and fitness value.

4.1 Scenarios

In this subsection, we develop data sets to evaluate performance of proposed algorithm. A generation containing of 50 chromosomes starts the process. Probability of cross-over is 90 percent and probability of mutation is 5 percent. Maximum number of generations to reach a solution is 1,000 while algorithm could end if after ten iterations no improvement in the fitness value of the best chromosome occurs. A set of seven incoming tasks with particular demands including arrival time, service time and deadline are assumed (see Table IX) and Table X illustrates data set of five HRs to deal with tasks. For example T1 could be executed completely in two cycles of time unit if it is taken by HR1. Also, HR3, HR4 and HR5 claim to complete T1 in 5, 4, 6 cycles of time unit. But, HR2 is not capable of executing T1, so it should not be allocated to T1. At the end of table, capability of each HR is shown. HR1 offers to fulfill all tasks with a background of high reputation as it is 0.7. HR3 is the most experienced resource as its expertise score is the highest among all resources.

Symbol table	
pop_size	Population size
C_p	Cross-over point
M_{p1}, M_{p2}	Mutation point
P_c	Probability of cross-over
P_m	Probability of mutation
H_{DAG}	Height of DAG
r, i, j, k	Temporary variables

Table VII. Defined symbols in proposed method

Software/Hardware	Type
PHP version	5.4.24
MySQL version	5.0.10
Processor	Intel Core i7-2700K
RAM	16GB-DDR3
System	64-bit

Table VIII. Running environment details

For the given data set in scenario 1, proposed method found an answer. HR1 is assigned to fulfill T1, HR5 to T2, HR4 to T3, and etc. With knowing the precedence order of each task, no violation of orders is detected (Table XI).

Another scenario (see Table XII) is solved using the proposed method which Table XIII illustrates best assignment table. There are 13 tasks waiting to be executed. HR8 can deal with all tasks while HR6 is not able to complete T1 and T2, so the final answer would not contain an assignment of HR6 to T1 or T2.

HR8 will execute T1, then HR8 will deal with T2, HR4 with T3, and etc. With the best found answer, all tasks will be assigned to appropriate resources to be executed. A summary of important parameters is given in Table XIV.

Table IX. Seven tasks with demands

Task No.	Arrival time	Service time	Deadline
T1	2	2	4
T2	3	2	6
T3	5	1	20
T4	5	1	20
T5	6	1	12
T6	7	2	10
T7	7	1	18

Table X. Execution parameters for seven tasks and five HRs (Scenario 1)

HR No.	T1	T2	T3	T4	T5	T6	T7	Rep.	Agi.	Exp.
HR 1	2	6	9	5	8	N	5	0.7	0.5	0.5
HR 2	N	2	1	2	1	2	6	0.6	0.2	0.3
HR 3	5	7	2	1	4	8	1	0.6	0.4	0.8
HR 4	4	5	1	6	2	6	N	0.2	0.6	0.6
HR 5	6	4	2	6	3	8	6	0.4	0.2	0.4

Table XI. Best found answer (Scenario 1)

Task	1	2	3	4	5	6	7
HR	1	5	4	3	5	2	3

Table XII. Execution parameters for 13 tasks and eight HRs (Scenario 2)

HR No.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	Rep.	Agi.	Exp.
HR 1	4	5	4	N	7	7	3	4	7	9	7	5	2	0.5	0.2	0.4
HR 2	8	5	1	7	N	7	4	2	6	8	7	6	5	0.7	0.6	0.3
HR 3	6	8	9	4	9	N	5	1	N	4	8	2	2	0.4	0.4	0.6
HR 4	8	3	3	7	9	4	6	2	6	9	7	4	4	0.6	0.5	0.6
HR 5	N	9	6	8	5	1	6	3	4	8	9	3	6	0.3	0.3	0.4
HR 6	N	N	4	6	7	4	7	9	6	7	6	2	1	0.7	0.8	0.6
HR 7	6	9	8	N	5	N	3	9	3	6	6	1	7	0.5	0.5	0.7
HR 8	2	5	9	5	4	7	4	6	1	2	1	7	8	0.5	0.3	0.9

4.2 Results

In this subsection, we evaluate the performance of the proposed method in comparison to IWD (Mokhtari, 2015), GA (Jafari Navimipour *et al.*, 2014), CS (Yildiz, 2013; Jafari Navimipour and Sharifi Milani, 2015) and BA (Musikapun and Pongcharoen, 2012).

Figures 3 and 4 show the overall time needed to fulfill tasks in each scenarios. The results show that the proposed method finds better solutions to fulfill tasks in a shorter time with higher HR utilization and lower fail rate. The proposed method finds better solutions to fulfill tasks in a shorter time.

Table XIII. Best found answer (Scenario 2)

Task	1	2	3	4	5	6	7	8	9	10	11	12	13
HR	8	8	4	6	5	8	4	6	7	2	8	7	6

Table XIV. Important parameters used in scenarios

GA parameters	
Size of population	50
Maximum number of generations	1,000
Selection mechanism	Roulette wheel
Fitness function	*As defined in Equation 1*
Cross-over rate	One point ($P_c = 0.9$)
Mutation rate	Swap ($P_{m=} 0.05$)
Termination conditions	No improvement in the fitness value of the best chromosome for ten iterations

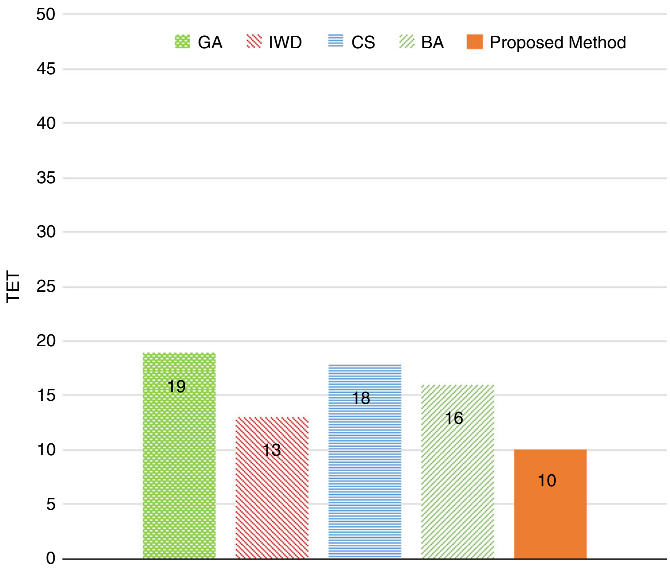


Figure 3. TET of all tasks in different methods (scenario 1)

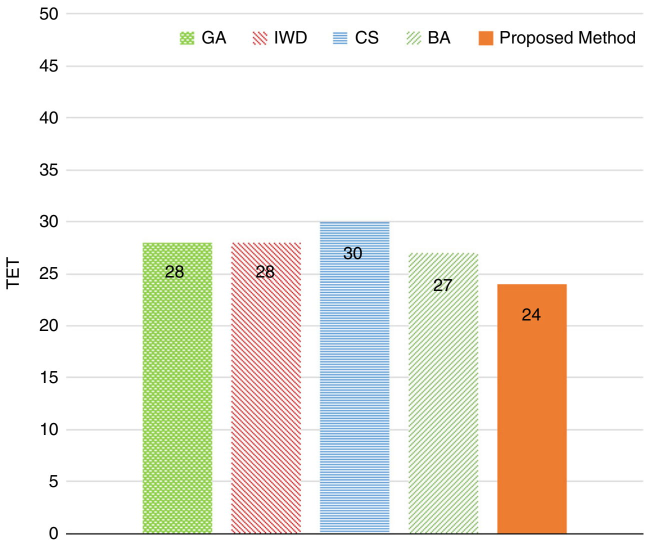


Figure 4.
TET of all tasks in different methods (scenario 2)

Task failure rate is another important factor for evaluation of the proposed scheduler. Results as illustrated in Figures 5 and 6 show that this method has better action in term of failure rate in comparison to other policies.

HR utilization is a key factor for evaluation of any scheduler. Results as illustrated in Figures 7 and 8 show that this method acts better in term of HR utilization rate in comparison to other policies.

Figure 9 is a progress diagram of a large scenario in addition to two previous scenarios which is simulated separately. This scenario includes 2,000 dependent tasks and just five HRs to simulate reaction of proposed algorithm under a condition of heavy

Figure 5.
The failure rate values in different methods (scenario 1)

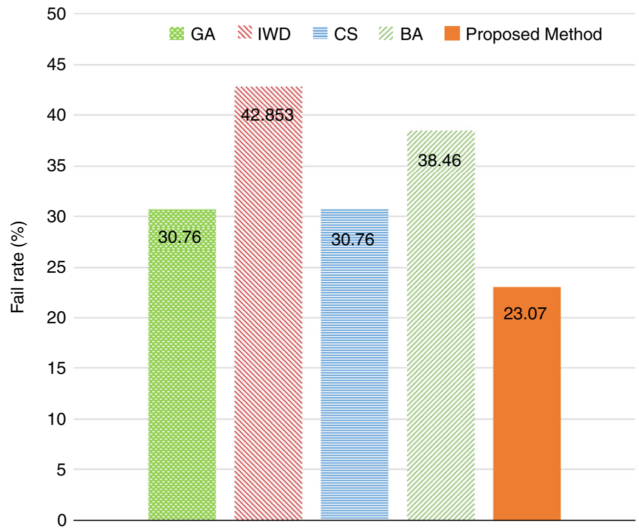


Figure 6.
The failure rate values in different methods (scenario 2)

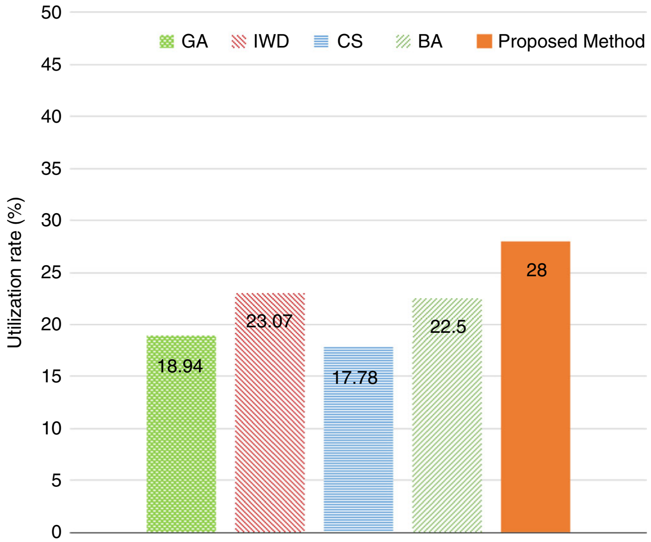


Figure 7.
The HR utilization rate values in different methods (scenario 1)

traffic while few resources are available. With given task demands, a solution is found after 32 iterations as termination condition was defined to be no improvement in the fitness value of the best chromosome after five iterations.

Finally, a summary of the important parameters, including supported task dependency and resource types, complexity order and some human-based features, is illustrated in Table XV. It is obvious that the proposed method acts better in term of considering human-based feature of realistic world while providing precedence order support on heterogeneous resources while some other methods support only homogeneous resources and task dependency is not discussed in some of them.

Figure 8.
The HR utilization rate values in different methods (scenario 2)

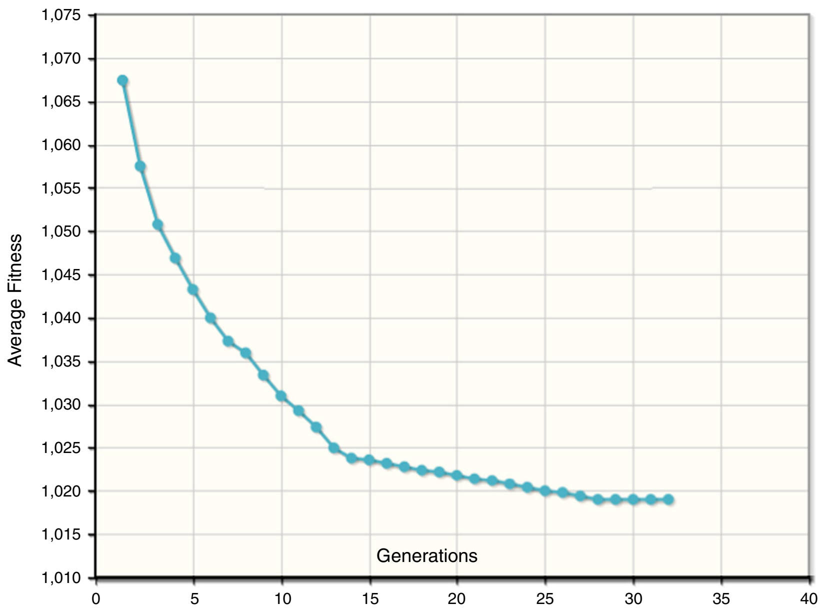


Figure 9.
The average fitness in generation progress

5. Conclusion and future works

With the rapid development of Cloud computing, more and more industries took Cloud computing uses in business models into account. To enhance overall performance of the Expert Cloud, as a new class of Cloud systems, efficient task scheduling is vital. There are several scheduling methods for different environments like Cloud, Grid or P2P networks which have various characteristics. In this paper we introduced a new

Method	Task dependency	Resource type	Complexity order	Deadline	Rep.	Agi.	Exp.
GA-based (Jafari Navimipour *et al.*, 2014)	Independent	Homogeneous	n	✓	X	X	X
Cloud-DLS (Wang *et al.*, 2012)	Independent	Homogeneous	n	X	X	X	X
MPQGA (Xu *et al.*, 2014)	Dependent	Heterogeneous	n	✓	X	X	X
PJSC (Ghanbari and Othman, 2012)	Dependent	Heterogeneous	$n^{2.81}$	X	X	X	X
Aravind (2013)	Independent	Homogeneous	n	X	X	X	X
Wang *et al.* (2011)	Dependent	Homogeneous	n	✓	✓	X	X
Proposed method	Dependent	Heterogeneous	1	✓	✓	✓	✓

Note: ✓, signifies support and X, otherwise

Table XV. Scheduling metrics considered by different methods

genetic-based algorithm to schedule tasks in the Expert Cloud which is a combination of some existing scheduling techniques and some newly added features. Newly presented algorithm supports priority of tasks as dependency is common in real world. Also, order of selection operation as a key factor of genetic algorithms performance is reduced to *O (1)* by applying stochastic acceptance. Another advantage of this algorithm is considering human-based features. A major disadvantage of proposed method might be slow growth of generations in large DAGs which is referred to low speed nature of genetic algorithm that we put effort to improve by applying stochastic acceptance in selection part. The experimental results also showed that the proposed method performs faster with higher accuracy and lower failure rate in comparison to IWD, GA, CS and BA. In the future, we can improve current method by adding more human-based features like mutual trust. Also, further improvements can be made using new optimization techniques (e.g. metaheuristic). The proposed method could be applied to the other practical problems like load balancing, resource discovery and energy efficient Cloud computing. To prove and explain the convergence of the proposed method some mathematical analysis or formal verification might be useful.

Note

1. www.php.net

References

Abdoun, O., Abouchabaka, J. and Tajani, C. (2012), "Analyzing the performance of mutation operators to solve the travelling salesman problem", *International Journal of Emerging Sciences*, Vol. 2 No. 1, pp. 61-77.

Abrishami, S. and Naghibzadeh, M. (2012), "Deadline-constrained workflow scheduling in software as a service cloud", *Scientia Iranica*, Vol. 19 No. 3, pp. 680-689.

Anselmi, J., Ardagna, D. and Passacantando, M. (2014), "Generalized nash equilibria for SaaS/PaaS clouds", *European Journal of Operational Research*, Vol. 236 No. 1, pp. 326-339.

Aravind, A.A. (2013), "Simple, space-efficient, and fairness improved FCFS mutual exclusion algorithms", *Journal of Parallel and Distributed Computing*, Vol. 73 No. 8, pp. 1029-1038.

Diaz-Gomez, P.A. and Hougen, D.F. (2007), "Initial population for genetic algorithms: a metric approach", *International Conference on Genetic and Evolutionary Methods, Las Vegas, NV*, pp. 43-49.

Ghanbari, S. and Othman, M. (2012), "A priority based job scheduling algorithm in cloud computing", *Procedia Engineering*, Vol. 50 No. 1, pp. 778-785.

Jafari Navimipour, N. (2015), "A formal approach for the specification and verification of a trustworthy human resource discovery mechanism in the expert cloud", *Expert Systems with Applications*, Vol. 42 No. 15, pp. 6112-6131.

Jafari Navimipour, N. and Mohammad Khanli, L. (2008), "The LGR method for task scheduling in computational grid", *Advanced Computer Theory and Engineering, ICACTE, International Conference, IEEE*, pp. 1062-1066.

Jafari Navimipour, N. and Sharifi Milani, F. (2014), "A comprehensive study of the resource discovery techniques in peer-to-peer networks", *Peer-to-Peer Networking and Applications*, Vol. 8 No. 3, pp. 1-19.

Jafari Navimipour, N. and Sharifi Milani, F. (2015), "Task scheduling in the cloud computing based on the cuckoo search algorithm", *International Journal of Modeling and Optimization*, Vol. 5 No. 1, pp. 244-247.

Jafari Navimipour, N., Navin, A.H., Rahmani, A. and Hosseinzadeh, M. (2014), "Job scheduling in the expert cloud based on genetic algorithms", *Kybernetes*, Vol. 43 No. 8, pp. 1262-1275.

Jafari Navimipour, N., Navin, A.H., Rahmani, A.M. and Hosseinzadeh, M. (2015a), "Behavioral modeling and automated verification of a cloud-based framework to share the knowledge and skills of human resources", *Computers in Industry*, Vol. 68 No. 1, pp. 65-77.

Jafari Navimipour, N., Rahmani, A.M., Navin, A.H. and Hosseinzadeh, M. (2015b), "Expert cloud: a cloud-based framework to share the knowledge and skills of human resources", *Computers in Human Behavior*, Vol. 46 No. 1, pp. 57-74.

Lipowski, A. and Lipowska, D. (2012), "Roulette-wheel selection via stochastic acceptance", *Physica A: Statistical Mechanics and its Applications*, Vol. 391 No. 391, pp. 2193-2196.

Manvi, S.S. and Krishna Shyam, G. (2014), "Resource management for infrastructure as a service (IaaS) in cloud computing: a survey", *Journal of Network and Computer Applications*, Vol. 41 No. 1, pp. 424-440.

Mateos, C., Pacini, E. and Garino, C.G. (2013), "An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments", *Advances in Engineering Software*, Vol. 56 No. 1, pp. 38-50.

Mokhtari, H. (2015), "A nature inspired intelligent water drops evolutionary algorithm for parallel processor scheduling with rejection", *Applied Soft Computing*, Vol. 26 No. 1, pp. 166-179.

Musikapun, P. and Pongcharoen, P. (2012), "Solving multi-stage multi-machine multi-product scheduling problem using bat algorithm", *2nd International Conference on Management and Artificial Intelligence*, pp. 98-102.

Navin, A.H., Jafari Navimipour, N., Rahmani, A.M. and Hosseinzadeh, M. (2014), "Expert grid: new type of grid to manage the human resources and study the effectiveness of its task scheduler", *Arabian Journal for Science and Engineering*, Vol. 39 No. 8, pp. 6175-6188.

Su, S., Li, J., Huang, Q., Huang, X., Shuang, K. and Wang, J. (2013), "Cost-efficient task scheduling for executing large programs in the cloud", *Parallel Computing*, Vol. 39 No. 4, pp. 177-188.

Sun, B., Wang, W., Xie, X. and Qin, Q. (2010), "Satellite mission scheduling based on genetic algorithm", *Kybernetes*, Vol. 39 No. 8, pp. 1255-1261.

Tawfeek, M.A., El-Sisi, A., Keshk, A.E. and Torkey, F.A. (2013), "Cloud task scheduling based on ant colony optimization", *Computer Engineering & Systems (ICCES), 8th International Conference, IEEE*, pp. 64-69.

Wang, W., Zeng, G., Tang, D. and Yao, J. (2012), "Cloud-DLS: dynamic trusted scheduling for cloud computing", *Expert Systems with Applications*, Vol. 39 No. 3, pp. 2321-2329.

Wang, X., Yeo, C.S., Buyya, R. and Su, J. (2011), "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm", *Future Generation Computer Systems*, Vol. 27 No. 8, pp. 1124-1134.

Wickboldt, J.A., Esteves, R.P., De Carvalho, M.B. and Granville, L.Z. (2014), "Resource management in IaaS cloud platforms made flexible through programmability", *Computer Networks*, Vol. 68 No. 1, pp. 54-70.

Wu, L., Kumar Garg, S. and Buyya, R. (2012), "SLA-based admission control for a software-as-a-service provider in cloud computing environments", *Journal of Computer and System Sciences*, Vol. 78 No. 5, pp. 1280-1299.

Xu, Y., Li, K., Hu, J. and Li, K. (2014), "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues", *Information Sciences*, Vol. 270 No. 1, pp. 255-287.

Yigit, M., Gungor, V.C. and Baktir, S. (2014), "Cloud computing for smart grid applications", *Computer Networks*, Vol. 70 No. 1, pp. 312-329.

Yildiz, A.R. (2013), "Cuckoo search algorithm for the selection of optimal machining parameters in milling operations", *The International Journal of Advanced Manufacturing Technology*, Vol. 64 Nos 1-4, pp. 55-61.

Zheng, L., Chen, S., Hu, Y. and He, J. (2011), "Applications of cloud computing in the smart grid", *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2nd International Conference, IEEE*, pp. 203-206.

Corresponding author
Dr Nima Jafari Navimipour can be contacted at: jafari@iaut.ac.ir

This article has been cited by:

1. Bahman Keshanchi, Alireza Souri, Nima Jafari Navimipour. 2017. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *Journal of Systems and Software* **124**, 1-21. [CrossRef]
2. Fariba Aznoli, Nima Jafari Navimipour. 2017. Cloud services recommendation: Reviewing the recent advances and suggesting the future research directions. *Journal of Network and Computer Applications* **77**, 73-86. [CrossRef]
3. Parina Alamir Shabestar Iran (the Islamic Republic of) Nima Jafari Navimipour Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran TABRIZ Iran, Islamic Republic of Magnus Ramage The Open University Milton Keynes United Kingdom of Great Britain and Northern Ireland Magnus Ramage The Open University Milton Keynes United Kingdom of Great Britain and Northern Ireland Magnus Ramage The Open University Milton Keynes United Kingdom of Great Britain and Northern Ireland . 2016. Trust evaluation between the users of social networks using the quality of service requirements and call log histories. *Kybernetes* **45**:10. . [Abstract] [PDF]
4. Fariba Aznoli, Nima Jafari Navimipour. 2016. Deployment Strategies in the Wireless Sensor Networks: Systematic Literature Review, Classification, and Current Trends. *Wireless Personal Communications* . [CrossRef]
5. Saeideh Hazratzadeh Tabriz Iran (the Islamic Republic of) Nima Jafari Navimipour Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran Tabriz Iran, Islamic Republic of Magnus Ramage The Open University Milton Keynes United Kingdom of Great Britain and Northern Ireland David Chapman The Open University Milton Keynes United Kingdom of Great Britain and Northern Ireland MarkWilliam Johnson University of Liverpool Liverpool United Kingdom of Great Britain and Northern Ireland . 2016. Colleague recommender system in the expert cloud using features matrix. *Kybernetes* **45**:9. . [Abstract] [PDF]
6. Bahman Keshanchi, Nima Jafari Navimipour. 2016. Priority-Based Task Scheduling in the Cloud Systems Using a Memetic Algorithm. *Journal of Circuits, Systems and Computers* **25**:10, 1650119. [CrossRef]
7. Sanay Abdollahzadeh, Nima Jafari Navimipour. 2016. Deployment strategies in the wireless sensor network: A comprehensive review. *Computer Communications* **91-92**, 1-16. [CrossRef]
8. Nima Jafari Navimipour, Yeganeh Charband. 2016. Knowledge sharing mechanisms and techniques in project teams: Literature review, classification, and current trends. *Computers in Human Behavior* **62**, 730-742. [CrossRef]
9. Matin Chiregi, Nima Jafari Navimipour. 2016. Trusted services identification in the cloud environment using the topological metrics. *Karbala International Journal of Modern Science* **2**:3, 203-210. [CrossRef]
10. IbrahimRabha Waell Rabha Waell Ibrahim GhaniAbdullah Abdullah Ghani Institute of Mathematical Sciences, University of Malaya, Kuala Lumpur, Malaysia Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia . 2016. Hybrid cloud entropy systems based on Wiener process. *Kybernetes* **45**:7, 1072-1083. [Abstract] [Full Text] [PDF]

11. Zeynab Soltani, Nima Jafari Navimipour. 2016. Customer relationship management mechanisms: A systematic review of the state of the art literature and recommendations for future research. *Computers in Human Behavior* **61**, 667-688. [CrossRef]

12. Alireza Sadeghi Milani, Nima Jafari Navimipour. 2016. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications* **71**, 86-98. [CrossRef]

13. Batool Zareie, Nima Jafari Navimipour. 2016. The effect of electronic learning systems on the employee's commitment. *The International Journal of Management Education* **14**:2, 167-175. [CrossRef]

14. Nima Jafari Navimipour, Bahareh Alami MilaniReplica selection in the cloud environments using an ant colony algorithm 105-110. [CrossRef]

15. Samad Mohammad Aghdam, Nima Jafari Navimipour. 2016. Opinion leaders selection in the social networks based on trust relationships propagation. *Karbala International Journal of Modern Science* **2**:2, 88-97. [CrossRef]

16. Bahareh Alami Milani, Nima Jafari Navimipour. 2016. A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications* **64**, 229-238. [CrossRef]