Cluster Comput (2014) 17:129–137
DOI 10.1007/s10586-013-0275-6

HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems

Arash Ghorbannia Delavar · Yalda Aryan

Received: 1 March 2012 / Accepted: 7 May 2013 / Published online: 30 May 2013

Abstract In heterogeneous distributed computing systems like cloud computing, the problem of mapping tasks to resources is a major issue which can have much impact on system performance. For some reasons such as heterogeneous and dynamic features and the dependencies among requests, task scheduling is known to be a NP-complete problem.

In this paper, we proposed a hybrid heuristic method (HSGA) to find a suitable scheduling for workflow graph, based on genetic algorithm in order to obtain the response quickly moreover optimizes makespan, load balancing on resources and speedup ratio.

At first, the HSGA algorithm makes tasks prioritization in complex graph considering their impact on others, based on graph topology. This technique is efficient to reduction of completion time of application. Then, it merges Best-Fit and Round Robin methods to make an optimal initial population to obtain a good solution quickly, and apply some suitable operations such as mutation to control and lead the algorithm to optimized solution. This algorithm evaluates the solutions by considering efficient parameters in cloud environment.

Finally, the proposed algorithm presents the better results with increasing number of tasks in application graph in contrast with other studied algorithms.

Keywords Heterogeneous distributed computing systems · Cloud computing · Workflow scheduling · Heuristic · Genetic Algorithm

A. Ghorbannia Delavar (✉) · Y. Aryan
Department of Computer, Payame Noor Universtiy,
P.O. Box 19395-3697, Tehran, Iran
e-mail: a_ghorbannia@pnu.ac.ir

Y. Aryan
e-mail: yld_Aryan@yahoo.com

1 Introduction

Heterogeneous distributed computing (HDC) system consists of potentially millions of heterogeneous computing nodes interconnected through arbitrary network architecture and is made for the creation of high-throughput computing resource pools, and domain virtual organizations, to meet the requirements of widely varying applications.

Cloud computing is a paradigm for HDC system promised to deliver the utility computing vision with some appealing properties such as sharing the resources as services on the Internet on-demand forming, and includes many dynamic resources and requests.

It consists of a collection of heterogeneous computing nodes, virtualized computers, and software services that are dynamically provisioned among the competing end-user's applications based on their availability, performance, capability, and Quality of Service (QoS) requirements [1]. The cloud environment provisions services in main three levels: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These levels support virtualization and management of differing levels of the solution stack [2].

Cloud computing service providers, make the large-scale network servers form the pool of large-scale virtual resources. IaaS level is the delivery of hardware (server, storage and network), and associated software (operating systems virtualization technology, file system), and provides a large amount of computational capacities to service remote users in a flexible and efficient way. In this level, the resources have provisioned in the form of Virtual Machines (VMs) deployed within the cloud equipments consisting of: Data-center, physical resources etc. for fulfilling the requests. Resource management sub systems in HDC systems are designated to schedule incoming tasks in getting the service.

Task scheduling is a key process for IaaS that is mapping the requests on resources in an efficient manner by considering cloud characteristics. It takes VMs as scheduling units for mapping physical heterogeneous resources to tasks. Each VM is an abstract unit of computing and storage capacities in cloud. By some reason such as heterogeneous and dynamic properties of resources, in addition to many number of entry tasks with different characteristics this issue is known to be a NP-Complete problem.

Since a good scheduling method would enhance the performance of the distributed system significantly and there is no direct method to find an optimal solution in polynomial time, so the schedule decisions must rely on heuristic ways on finding a solution as good as possible.

The software running on HDC system is called an application. Each application is a workflow, which is composed of intercommunicated tasks. These tasks are scheduled to run over different processors in the systems [3].

Many methods are proposed for this problem. Some of them have used heuristic and evolutionary methods. Each method often focuses on limited number of parameters and main objectives such as the completion time of all tasks (makespan), such as Min-min, Max-min or Sufferage. Moreover some classical solution scheduling algorithms often rely on list-based methods, such as Heterogeneous Earliest Finish Time (HEFT) or Critical Path, that often are used in simple model systems and do not accurately reflect real parallel systems [4]. The Heterogeneous Earliest Finish with Duplicator (HEFD) [5] used duplication of parent tasks in scheduling. A dynamic list scheduling (DLS) method is presented in [6] for cloud computing systems. It does not consider task priority of DAG workflow, also there are some evaluated workflow scheduling algorithms in [7] that are useful in homogeneous systems.

Some meta-heuristic based methods are presented to solve NP problems such as: particle swarm optimization (PSO) [8], tabu search (TS) [9], simulated annealing (SA) [10], genetic algorithm (GA) etc. In contrast, GA by [11, 12, 14] are known to give good results in several optimization domains and provide robust search techniques that allow a high-quality solution to be obtained from a large search space and parallel search in polynomial time by applying the principle of evolution. It could present several solutions to evaluate the efficient parameters.

Some of GA based methods have already been used in the scheduling of workflows [13, 14]. But they don't consider heterogeneous components characteristics and is not suitable for HDC system like cloud systems. Some of GA based methods apply random manner to prepare the initial population, or use different prioritization method for task ordering in the same level in the workflow, such as Most Completed Tasks, Most Outstanding Tasks, Least Dependent Tasks, Most Dependent Tasks [15], and some of them use simple graph with the most two output nodes. The number of GA based solutions are proposed in [16, 21] for heterogeneous systems too.

In this paper, we suggest a hybrid heuristic method (HSGA) based on GA to find a proper scheduling solution for HDC system like cloud environments with complex computational applications. The HSGA proposed algorithm tries to decrease the number of GA operation iteration with starting the algorithm by an optimized initial population, considering the balance the load on resources. It makes the solutions by two evaluation functions, a function measures priority of each task in workflow, based on their influence on the others, and another function evaluate the value of the produced solutions. It focuses to decrease the completion time of application as makespan and failure rate, and increasing the load balancing subjects.

The rest of this paper is organized as follow: in Sect. 2 the related work are discussed, in Sect. 3 the Problem definition is described, Sect. 4 is about proposed algorithm, in Sect. 5 the evaluating of performance of HSGA is presented, and the conclusion in Sect. 6 would end the paper.

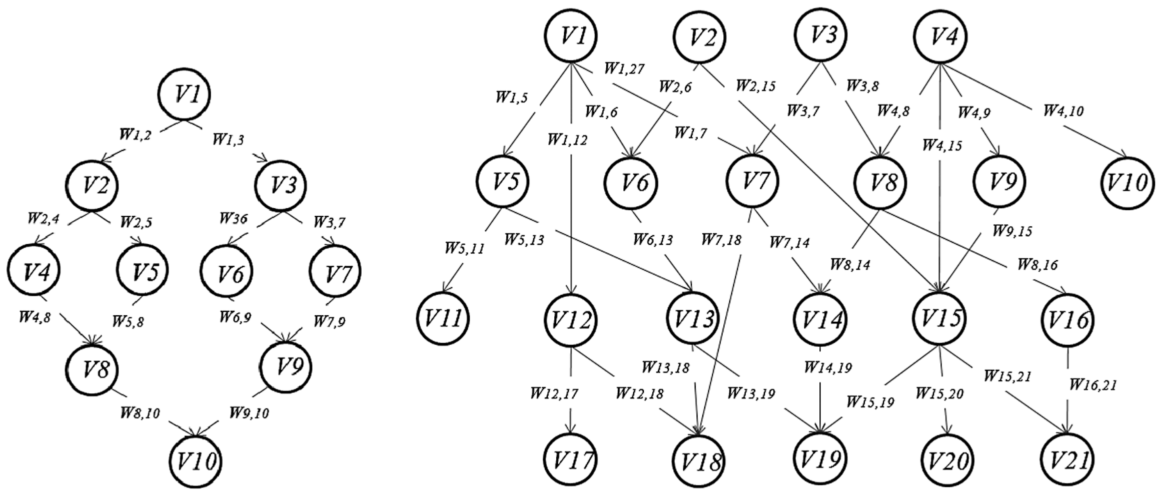
2 Related works

In this section, we mention some algorithms such as Greedy (First-Fit), Round Robin, LAGA and NGA which are heuristic and based on GA methods.

In some policies, First-Fit [22] or RR (Round Robin) are used by some cloud system such as Eucalyptus [23]. In these methods the starvation problem is almost solved and the makespan is decreased. But, the requests will run on all resources and would not support the optimal usage of resources and a proper load balancing.

The LAGA algorithm has proposed for large-scale distributed systems like Grid and Cloud, based on GA. It is a computation-intensive and reliability-driven reputation algorithm that considers the tasks' runtime using the task failure rate (task failures per unit time) of resources in order to define the reputation and evaluate the reliability of resources. This method computes a task ordering procedure by resource completion time in each generation and selects a resource with the least failure rate in mutation operation. It focuses on completion time and schedule failure rate [16].

The NGA has proposed for heterogeneous multiprocessor systems, based on GA method too, that focus on completion time of application and communication delay time. The fitness function in NGA does the evolution in two different phases. First phase is fitness of task that the authors said it equipped system with the knowledge all the tasks are executed and scheduled in legal order. Here legal order is schedule a pair of tasks on single processor while the pair of



(a) A simple-structure graph (b) A complex-structure graph

Fig. 1 The simple and complex graphs in workflows

tasks is independent to each other. The second phase is fitness of processor which attempts to minimize the processing time [21].

Although to send and receive the requests and data among resources, the communication cost affects the response time to requests as a major factor, these algorithms do not consider it. They just used computation time to calculate the completion time of request without effective communication time.

In this study we use in complex computational intensive requests which is assigned on pool of resources with fully connected graph, and suggest an entry task ordering using in terms of request execution time, and making initial population method by contribution of the optimal characteristics by Best-Fit and Round Robin methods, to find the response quickly. This proposed algorithm considers the completion and communication time.

3 Problem definition

In cloud computing environment, there is data-center system, which is assumed to collect and save the static and dynamic information of resources and tasks. Some key static information are: physical memory storage space, virtual memory storage space, disk storage space etc., and some dynamic information such as: the load average of the node itself, the number of the running tasks, the current running tasks' number of threads, and the status of these tasks, CPU usage etc., are captured periodically or based on the polling strategy or others, and are sent to the Data Receiver of the master node through the communication component. These data are updated frequently, and in real-time form to provision the response as services [24]. Clearly, this heterogeneity in cloud systems is effective in response to requests.

We can use some static and dynamic information for scheduling. Since a lot of tasks are received in the cloud system in each time, so the workload and other dynamic information impress to select a good candidate resource for task.

Each application in distributed systems is a workflow that contains a set of tasks, which tasks are connected to each other by precedence constraints. Each task will be executed and give an output dataset. This data set is then sent to the next task as defined by the structure of the workflow. Most workflow application can be represented in the form of a DAG (Direct Acyclic Graph): a graph whose nodes are the tasks and whose edges are the precedence constraints [4], such as Montage, Epigenomics, SIPHT, LIGO [17], Cyber Shake, SPHINX projects, etc. [8, 18–20].

According to many workflow projects, the workflow application structures can be categorized as either simple-structure or complex-structure. In the simple workflows, nodes are related together considering certain level, but the complex-structure application like [25] has more relation among nodes where the level of some nodes is not certain (see Fig. 1).

When the size of the workflow is increased the processing time may become very long. Because of their heterogeneity—the cloud platform have hosts with different properties and calculation capacities.

Some of applications are computation-intensive and some are communication-intensive. The communication to computation ratio (CCR) is a measure that indicates whether a task graph is communication-intensive, computation-intensive or moderate [26]. The CCR factor is computed by the average communication cost divided by the average computation cost on target system.

At a cursory glance, the used notations in this paper are described in Table 1.

Table 1 Definitions of the notations

Notation	Definition
V	The set of v nodes that present as tasks in application graph
E	The set of e edges or dependencies between tasks in application graph
v_i	A task in application graph
$w(ei, j)$	The weight of edge between task v_i and v_j in graph
p_j^{MIPS}	The number of resource instruction per minutes as processing speed
$T_{pred}(v_i)$	Set of predecessors of task v_i
$T_{succ}(v_i)$	Set of successors of task v_i
$T_{ready}(v_i, p_j)$	The time when all the predecessors of task v_i have been executed
$T_{avail}(p_j)$	The time when processor p_j is available to the execution of task v_i
$T_{ST}(v_i, p_j)$	The start time of task v_i
$T_{FT}(v_i, p_j)$	The finish time of task v_i
$T_{priority}(v_i)$	The priority of task v_i
$CC(v_i, v_j)$	The cost of data transferring between two tasks v_i and v_j
$T_c(v_i, v_j)$	Size of output data from v_i to v_j
$C_{k,l}$	The bandwidth of link between two resources p_k and p_l
$d(v_i)$	Depth of task v_i in critical path
p_i^f	Probability of task failure per unit time
p_i^{fr}	The failure factor of resource
p_i^t	The time when resource p_i completes all its assigned tasks of workflow in current scheduling

Generally, an application graph with precedence constrained tasks is represented by a directed acyclic graph (DAG), $G = (V, E)$ that V is the set of v nodes presented as tasks and E is the set of e edges or dependencies among tasks indicating the relation and precedence constraints. Each task in graph has a weight $w(v_i)$ that is the length or the same number of instructions of the task, and the data transfer rate among tasks is introduced by the weigh $w(e_{i,j})$ of edges.

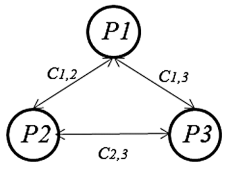
In the workflow graph, each task does not dispatch until its precedence tasks are completed. So the start time of each task is:

$$T_{ST}(v_i, p_j) = \max\{T_{avail}(p_j), T_{ready}(v_i, p_j)\} \tag{1}$$

where $T_{avail}(p_j)$ is the time when processor p_j is available for the execution of task v_i, $T_{ready}(v_i, p_j)$ is the time when all the predecessors of task v_i are executed and all necessary input data are available and could be transmitted to the processor p_j, it can be computed as:

$$T_{ready}(v_i, p_j) = \max\{T_{FT}(v_k) + w(e_{k,i})\},\quad v_k \in T_{pred}(v_i) \tag{2}$$

Fig. 2 An example of resource's connections



where $T_{FT}(v_k)$ is the finish time of predecessor, and $w(e_{k,i})$ is the weight between task and its predecessor. The finish time of each task can be computed as:

$$T_{FT}(v_i, p_j) = T_{ST}(v_i, p_j) + \left(w(v_i)/p_j^{MIP}\right) + CC(v_i, v_j) \tag{3}$$

where $w(v_i)$ is the workload or length of task v_i. In the workflow graph, the tasks with $T_{pred} = \emptyset$ are entry tasks, and the tasks with $T_{succ} = \emptyset$ are exit tasks.

The experiment is conducted on the complex graph that was depicted in Fig. 1(b). In these graphs some tasks are not in certain level. So the task selection to send to the ready queue is important.

Also in our study, the resources are fully connected by different links, like in Fig. 2.

When the tasks are being assigned to the resources, the cost of data transfer between two tasks can be computed as follow:

$$CC(v_i, v_j) = T_c(v_i, v_j)/C_{k,l} \tag{4}$$

Since the cloud systems have some properties that should be considered in the scheduling process, the important constraints used in method are:

- The amounts of entry requests are always more than the amount of resources. So each resource can process more than one request
- The request characteristics are always variable and indeterminate, such as: arrival time, execution time etc.
- The cloud environment is a collection of heterogeneous resources
- The resources have dynamic hardware and software characteristics such as: average of workload on node, CPU usage, failure rate, etc.

As mentioned, in several methods the GA is used to find a suitable solution in NP problems such as workflow scheduling. Typically the normal genetic algorithm steps are so:

1. Create initial population considering the task graph topology
2. Evaluate the solutions by fitness function
3. Select the some solution for next population
4. Doing the crossover and mutation operations
5. If the stop conditions are not met, repeat 2–5 steps.

4 Proposed algorithm

In this approach, a hybrid heuristic method, based on GA is used, by considering the cloud system characteristics. Generally, the pseudo-code of proposed algorithm is in Algorithm 1.

Algorithm 1 Pseudo code of HSGA method

Input: Set of available resources and unmapped tasks of an application.

Output: An optimized generated schedule.

```
1.  Find depth of each task in application graph
2.  Set the priority if each task by (5) considering the
    graph topology
3.  Update resource properties
4.  Make a list of available resources
    // make initial population
5.  for each chromosome do
6.      Find the best fit resources for each task based on the
    execution time
7.      Go to the next place in resource list
8.      If the counter of index=last resource then
9.          Go to the first place in resource list
10. End
11. Evaluate all chromosomes using (8)
12. While the stop conditions are met
13.     Random selection gene crossover operation
14.     mutation operation
15.     Select the best chromosomes as elites
16. End while
17. Save the best solution
18. Dispatch all mapped tasks on candidate resources due
    to obtain the best solution
```

4.1 Encoding

In GA method, every solution is encoded as a chromosome. Each chromosome has N genes, as the chromosome length. In workflow scheduling each schedule appears in a chromosome form. Each schedule contains the tasks of application and the related candidate resources. Figure 3 depicts a chromosome in HSGA method.

Here, first, the tasks of graph are ordered on priority based on their influence on the other tasks in the graph for execution according to Sect. 4.3, the tasks should mapped on the suitable resources from a set of available resources.

In this algorithm, to make a chromosome, each task is mapped to a selected resource from a virtual list of available resources, according to the data-center information. The virtual list will be updated in some operations such as initial population.

V17	*V2*	*V8*	*V0*	*V12*	...	*V24*
P2	*P9*	*P3*	*P1*	*P8*	...	*P5*

Fig. 3 A sample encoding of a schedule as a chromosome in HSGA

4.2 Initial population

Let us say, a set of multiple possible solutions (chromosomes) is assumed to be referred to as a population. The initial population is made randomly in normal genetic algorithm.

Making a good and goal oriented initial population that would lead to find the response in a rapid manner is the concern here. For this purpose, to build the initial population, after the tasks are sorted by priority, they will be placed in the first row of genes in the chromosome, and for each task, a suitable resource will be select with minimum running time for task from virtual resource list.

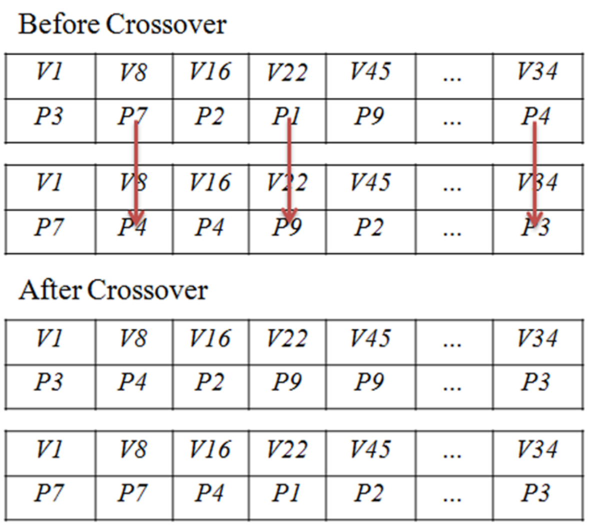
This process is repeated for all genes as Best-Fit. In this manner, in first chromosome for each gene, the algorithm selects the fittest resource from the first place in virtual list, but for the second chromosome, it finds the best resource from the second place in virtual list. The fittest candidate-resource will be searched from the next point, after last chromosome was started from the last place in virtual list and so on like Round-Robin method but for resource selection. This process, continue to making a population. But in making each chromosome, if the counter is finished, the resource selection will be continued from first place.

This method assures that all resources will be selected for making population. Thus, all possible solutions can almost be made, and attended the balance the load on resources.

After each available resources are selected as the candidate, the virtual list will be updated based on the rest processing capacity on current resources' workload for designated tasks.

4.3 Task prioritization

Because of, the tasks on complex application graph cannot be partitioned into levels easily, and each task's length and successors are different with the others, the task selection based on graph topology is an important problem. Since each task produces some outputs as input data set for its successors, the predecessor task should be executed before children. Completion time of each task influences the completion time of application. So an ordering method is being proposed that could be computed the priority of tasks based on task influence, according to the follow equation:



Before Crossover

V1	V8	V16	V22	V45	...	V34
P3	P7	P2	P1	P9	...	P4

V1	V8	V16	V22	V45	...	V34
P7	P4	P4	P9	P2	...	P3

After Crossover

V1	V8	V16	V22	V45	...	V34
P3	P4	P2	P9	P9	...	P3

V1	V8	V16	V22	V45	...	V34
P7	P7	P4	P1	P2	...	P3

Fig. 4 Crossover operation method in proposed algorithm

$$T_{priority}(v_i) = w(v_i) + \sum_{d(vj)=\alpha}^{\beta} \big(d(v_j) * w(e_{i,j}) \big),$$
$$v_j \in T_{pred}(v_i) \tag{5}$$

where $d(v_j)$ is the depth of task v_j in critical path. The critical path for each task is the longest path from it to an exit task. Each task has some successors, and each successor has a depth. According to the complex graph shown in Fig. 1(b) the set of task successors of v_4 are: $T_{succ}(v_4) = \{v_8, v_{15}, v_9, v_{10}\}$. So the execution of task v_4 is efficient for its successors and the execution of each successor of v_4 is efficient for their successors alternatively, as well. Also v_8 should be executed more quickly than $v_{15,}$ but v_{10} can be execute with v_{21} at the same time. The depth of task v_{10} is 1. So we can select the more important successors of each task with an important depth. Thus the limited range of depth selection is between α and β for selecting the most important successors for (4). Where β is the most depth of successors with the longest sequence and α can be computed as:

$$\alpha = \beta - \text{floor of } (\beta/2) \tag{6}$$

The priority of each task with respect to its dependencies in graph topology will be computed, and a list of task ordering is prepared by descending to make the first row of chromosomes.

4.4 Crossover

Here, a random gene selection crossover is used. Two parents randomly and their some genes are selected randomly. Then two other solutions by change in resource sections of selected genes are created. For example in some selected genes randomly such as second, fourth and last genes, the candidate resources are changed by each ether, in two random selected chromosomes. Figure 4 illustrates the before and after crossover in mentioned example.

4.5 Mutation

A chromosome and one of its genes will be chosen randomly and a resource will be selected randomly from virtual list of resources. The selected resource will be replaced with selected gene if its failure rate is better than the last candidate resource as [16], but here there is another condition. This candidate resource is not the resource with most workload in current schedule. The failure factor of resource can be computed as:

$$p_i^{fr} = p_i^f / p_i^{MIPS} \tag{7}$$

where, p_i^f is probability of task failure per unit time on resource p_i as failure rate of resource, and p_i^{MIPS} is the computing power of resource as number of machine instructions per second. The selecting and checking new candidate resource for a better result will be repeated in a limited case.

The mutation operation causes, GA does not stop in the local minimum, but this method in mutation leads to the finding of a good solution in a rapid manner.

4.6 Evaluation and selection solutions

To recognize the value of a solution, we should evaluate it by a fitness function with efficient parameters in quality of solution. In GA, the fitness function is applied on all solutions and computes their value, and then a solution with the best value, based on parameters placement policy, is obtained as the minimum or maximum for the fittest solution.

The fitness value of each solution is computed through:

$$Fitness = \sum_{i=1}^{m} (p_i^t \times p_i^f) \tag{8}$$

where p_i^t is the time when resource p_i completes all its assigned tasks of workflow in current scheduling. In this function, total spent time of each resource for current application is considered, that is effective on the makespan.

Also we consider p_i^f in order to reduce failure rate of executing of tasks as much possible.

The chromosome with minimum fitness value is considered as the best solution among the others.

$$\text{Target is: minimizing } (\mathit{fitness}) \tag{9}$$

The algorithm tries to find the solution by minimizing the fitness value as much possible by crossover and mutation operations. Some of the best of chromosomes are will be selected by elitism method for next iteration.

5 Performance evaluation

This section presents the comparative evaluation HSGA with two algorithms, LAGA and NGA and demonstrates and

Table 2 Simulation parameters

Parameter	Value
Number of tasks in application	20–100
Task lengths	12–72 ($\times 10^5$ MI)
The number of resources	30
Resource speeds	500–1000 (MIPS)
Bandwidth between resources	10–100 (mbps)
Failure rates of resource	10^{-4}–10^{-3}
CCR value	0.25

Table 3 HSGA algorithm parameters

Parameter	Value
Population size	20
Crossover rate	0.5
Mutation rate	0.5
Elitism selection rate	0.75
Number of iteration	20

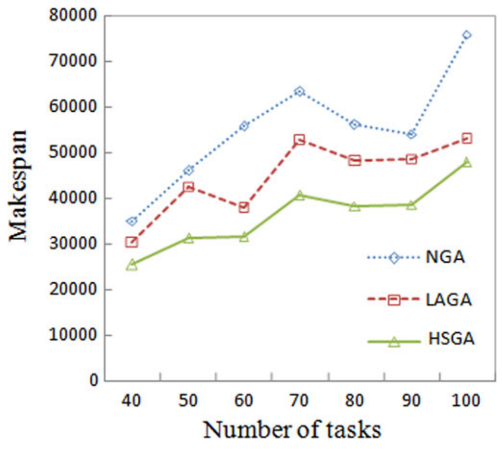


Fig. 5 Makespan for application with increasing the number of tasks

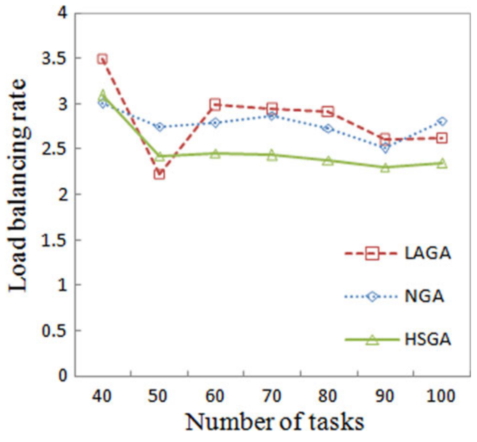


Fig. 6 The load balancing with increasing the number of tasks

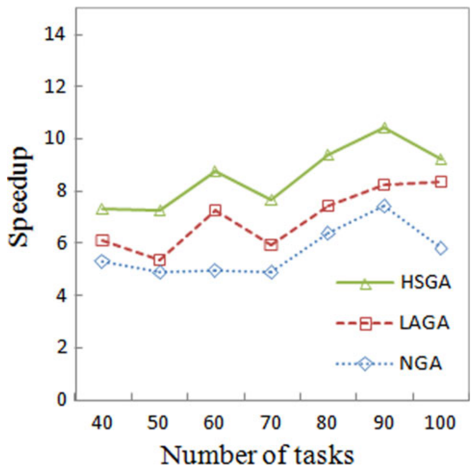


Fig. 7 The Speedup of algorithms in different number of tasks

evaluates the three subjects: makespan of application, failure rate and load balancing of resources. The experiments are conducted considering cloud characteristics with respect to heterogeneously in resource properties and various properties for applications.

The simulation parameters are listed in Table 2 and the parameters used by HSGA algorithm based on GA are listed in Table 3.

The average results are presented in Fig. 5.

In Fig. 5, other algorithms are compared with proposed one. The results show, the completion time of application (makespan) for HSGA is less than LAGA in about by 19.145 %, and in about by 34.39 % in contrast with NGA. It shows the makespan by HSGA is better than the others.

Also the balance the load on the resources in Fig. 6 is demonstrated. According to the used method in [14], the load balancing of proposed method is better than LAGA in about by 11.88 %, and it was reduced in about by 10.38 % better than NGA.

By illustrated chart in Fig. 7, we understand the speedup (is computed by dividing the sequential execution time by the makespan of parallel execution) in Different number of tasks by HSGA algorithm is better than LAGA in about by 18.22 % and than NGA in about by 33.77 %.

We compute the failure rate of produced solution like the method in [21], that has suggested to distributed systems, such as grid and cloud. So according to the Fig. 8, we perceive the produced solution by proposed algorithm has failure frequency near to the LAGA method in about by −0.16 % (a little value) that it is little important in cloud systems, because the failure probability of resources is so low, due to software level agreement (SLA) and more consistency of resources than grid computing. Also it is better than NGA about 8.67 %.

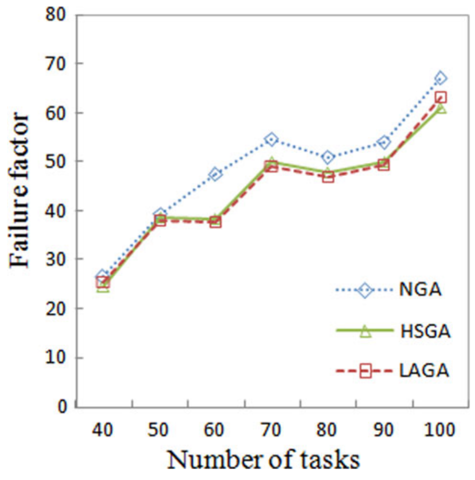


Fig. 8 The failure frequency of algorithms in different number of tasks

6 Conclusion

In this paper, a hybrid heuristic scheduling method is suggested for workflow of applications in cloud computing environment by fully connected resources with different communication costs. It is based on GA, that uses the optimize characteristics Best-Fit and Round-Robin algorithms.

The proposed method, makes a goal oriented initial population by the virtual list of resources and their updated properties. For making the initial population, it uses two stages evaluation. At first stage, all tasks of application will be ordered by a priority method with respect to their influence to each other based on graph topology. In the second stage, the candidate resources will be assigned by combination of features of two methods, Best-Fit and Round robin to select good candidate resources. Consequently this technique for making initial population makes to find the good solution in a rapid manner considering the load balancing, that is as close as the best possibility. Also we use some suitable parameters to increasing the quality of evaluation of population in fitness function.

The HSGA results are compared to LAGA and NGA algorithms. The produced solution through this proposed algorithm is perceived, the HSGA decreases the running time of tasks, in comparison with LAGA, and NGA, whereas it supports the load balancing and reliability with important parameters. Also it improves the makespan of application in about by 19.14 %, load balancing in about by 10.38 % and speedup ratio in about by 18.22 % at least. In the next work we want to present a method that support mapping the resources on tasks for communication intensive applications with efficient result.

Acknowledgements This research is sponsored by the Payam Noor University.

References

1. Ranjan, R., Buyya, R.: Decentralized overlay for federation of enterprise clouds (2010). http://www.techrepublic.com/whitepapers/decentralized-overlay-for-federation-of-enterprise-clouds/1828007
2. Ghorbannia Delavar, A., Aryan, Y.: A synthetic heuristic algorithm for independent task scheduling in cloud systems. Int. J. Comput. Sci. Issues **8**(6), 289 (2011)
3. Tanga, X., Li, K., Li, R., Veeravalli, B.: Reliability-aware scheduling strategy for heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **70**, 941–952 (2010)
4. Nicod, J.-M., Philippe, L., Toch, L.: A genetic algorithm to schedule workflow collections on a SOA-Grid with communication costs. LIFC Laboratoire D'informatique de l'Universite de Franche-COMTE, EA 4269 (2011)
5. Tang, X., Li, K., Liao, G., Li, R.: List scheduling with duplication for heterogeneous computing systems. J. Parallel Distrib. Comput. **70**, 323–329 (2010)
6. Li, J., Qiu, M., Niu, J., Gao, W., Zong, Z., Qin, X.: Feedback dynamic algorithms for preemptable job scheduling in cloud systems. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (2010)
7. Casanova, H., Desprez, F., Suter, F.: On cluster resource allocation for multiple parallel task graphs. J. Parallel Distrib. Comput. **70**, 1193–1203 (2010)
8. Pandey, S.: Scheduling and management of data intensive application workflows in grid and cloud computing environments. Doctoral thesis, Department of Computer Science and Software Engineering, the University of Melbourne, Australia (December 2010)
9. Porto, S., Ribeiro, C.: A tabu search approach to task scheduling on heterogeneous processors under precedence constraints. Int. J. High Speed Comput. **7**, 45–72 (1995)
10. Kalashnikov, A., Kostenko, V.: A parallel algorithm of simulated annealing for multiprocessor scheduling. J. Comput. Syst. Sci. Int. **47**, 455–463 (2008)
11. Yoo, M.: Real-time task scheduling by multi objective genetic algorithm. J. Syst. Softw. **82**, 619–628 (2009)
12. Yu, J., Buyya, R., Ramamohanarao, K.: Workflow scheduling algorithms for grid computing. Department of Computer Science and Software Engineering, The University of Melbourne, VIC 3010, Australia (2009). http://www.cloudbus.org/reports
13. Yoo, M.: Real-time task scheduling by multi objective genetic algorithm. J. Syst. Softw. **82**, 619–628 (2009)
14. Omara, F.A., Arafa, M.M.: Genetic algorithms for task scheduling problem. J. Parallel Distrib. Comput. **70**, 13–22 (2010)
15. Fida, A.: Workflow scheduling for service oriented cloud computing. A thesis submitted to the College of Graduate Studies and Research in Partial Fulfillment, Department of Computer Science University of Saskatchewan Saskatoon (2008)
16. Wang, X., Yeo, C.S., Buyya, R., Su, J.: Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. Future Gener. Comput. Syst. **27**, 1124–1134 (2011)
17. http://www.ligo.caltech.edu/advLIGO/
18. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator
19. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the montage example. In: Proc. of the 2008 ACM/IEEE Conference on Supercomputing (SC '08), Piscataway, NJ, USA, pp. 1–12 (2008)
20. In, J.-U., Arbree, A., Avery, P., Cavanaugh, R., Katageri, S., Ranka, S.: Sphinx: a scheduling middleware for data intensive applications on a grid. Technical report GriPhyN 2003-17, GriPhyn (Grid Physics Network) (2003)

21. Singh, J., Singh, H.: Efficient tasks scheduling for heterogeneous multiprocessor using genetic algorithm with node duplication. Indian J. Comput. Sci. Eng. **2**(3), 402 (2011)
22. Brent, R.P.: Efficient implementation of the first-fit strategy for dynamic storage allocation, Australian National University. ACM Trans. Program. Lang. Syst. **11**(3), 388–403 (1989)
23. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., So-man, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: IEEE International Symposium on Cluster Computing and the Grid (CCGrid '09) (2009)
24. Ge, J., Zhang, B., Fang, Y.: Research on the resource monitoring model under cloud computing environment. In: Wang, F.L., et al. (eds.) WISM 2010. LNCS, vol. 6318, pp. 111–118. Springer, Berlin (2010)
25. Ghorbannia Delavar, A., Aghazarian, V., Litkouhi, S., Khajeh Naeini, M.: A scheduling algorithm for increasing the quality of the distributed systems by using genetic algorithm. Int. J. Inf. Educ. Technol. **1**(1), 58–62 (2011)
26. Mezmaz, M., Melab, N., Kessaci, Y., Lee c, Y.C., Talbi, E.-G., Zomaya, A.Y., Tuyttens, D.: A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J. Parallel Distrib. Comput. **71**(11), 1497–1508 (2011)

Arash Ghorbannia Delavar received the M.Sc. and Ph.D. degrees in computer engineering from Sciences and Research University, Tehran, Iran, in 2002 and 2007. He obtained the top student award in Ph.D. course. He is currently an assistant professor in the Department of Computer Science, Payam Noor University, Tehran, Iran. He is also the Director of Virtual University and Multimedia Training Department of Payam Noor University in Iran. Dr. Arash Ghorbannia Delavar is currently editor of many computer science journals in Iran. His research interests are in the areas of computer networks, microprocessors, data mining, Information Technology, and E-Learning.

Yalda Aryan received a B.Sc. in computer engineering from Azad University, Arak, Iran, in 2000. She is M.Sc. student in computer engineering in Payam Noor University. Her research interests include computational intelligence, Grid computing and cloud computing.