A BI-OBJECTIVE WORKFLOW APPLICATION SCHEDULING IN CLOUD COMPUTING SYSTEMS

Yalda Aryan[1] and Arash Ghorbannia Delavar[2]

[1] Educational Computer Group, Research Centre of Teachers, Isfahan, Iran
[2] Department of Computer, Payam Noor University, PO BOX 19395-3697, Tehran, Iran

ABSTRACT

The task scheduling is a key process in large-scale distributed systems like cloud computing infrastructures which can have much impressed on system performance. This problem is referred to as a NP-hard problem because of some reasons such as heterogeneous and dynamic features and dependencies among the requests. Here, we proposed a bi-objective method called DWSGA to obtain a proper solution for allocating the requests on resources. The purpose of this algorithm is to earn the response quickly, with some goal-oriented operations. At first, it makes a good initial population by a special way that uses a bi-directional tasks prioritization. Then the algorithm moves to get the most appropriate possible solution in a conscious manner by focus on optimizing the makespan, and considering a good distribution of workload on resources by using efficient parameters in the mentioned systems. Here, the experiments indicate that the DWSGA amends the results when the numbers of tasks are increased in application graph, in order to mentioned objectives. The results are compared with other studied algorithms.

KEYWORDS

Cloud computing, Heterogeneous distributed computing systems, market oriented systems, Workflow scheduling, Genetic Algorithm

1.INTRODUCTION

The cloud computing system is a distributed system which offers the utility computing vision with some attractive qualities such as sharing the many dynamic resources by virtualization technology in order to meet the requirements of widely varying Requests.

Generally, task scheduling process is an important issue in distributed systems because the requests should be mapped on resources in an efficient manner by considering the environment properties. Because of heterogeneous resources and many numbers of tasks with different characteristics in these systems, this issue is known as a NP-hard (Non-deterministic Polynomial-time hard) problem. Since a good scheduling method would impress on the system performance and there is no direct method to find an optimal solution in polynomial time, the scheduling process must rely on finding the best solution within possibilities.

Many methods are proposed for this problem. Each method often focuses on main objectives such as the completion time of all tasks (makespan), or distribution of workload on resources. For example, many fundamental heuristic methods like greedy (First-fit) [1] and Round-Robin (RR) [2], Min-min, Max-min or Sufferage [3] try to achieve the makespan. Some different dynamic list scheduling methods are presented for heterogeneous distributed systems [4] which often do not consider the latency among.

A number of meta-heuristic based methods were presented to solve NP problems such as: particle swarm optimization (PSO) [5], tabu search (TS) [6], simulated annealing (SA) [7], genetic algorithm (GA) etc. In contrast, GA by [8] [9] [10] 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 proposed scheduling algorithms exert random methods in its steps like providing the initial population or ordering tasks in the same level in the workflow[11], and some of them use a simple workflow graph.

Already, we proposed another hybrid meta-heuristic workflow scheduling method by considering response time and reliability in heterogeneous distributed systems [12]. Here, we suggest a dynamic bi-objective method (DWSGA) in order to find a proper scheduling solution with computational applications according to cloud system environments. This algorithm tries to reduce the number of GA operation iterations by making an optimized initial population, in order to appropriate workload diffusion on resources, simultaneously.

The rest of this paper is organized as follows: the related works are discussed in section 2, in section 3 the problem definition is described, and in section 4 the proposed algorithm is presented. The simulation result of DWSGA is presented in section 5, and the conclusion in section 6 would end the article.

2. RELATED WORKS

Some non-preemptive methods such as First-Fit and Round-Robin are used in some cloud systems such as Eucalyptus. Task starvation is a major problem in these methods that causes non-optimal usage and workload balancing of resources.
Also some other heuristic proposed algorithms in distributed systems, like the ones proposed by [13], [15], [15] and [16] by considering with Grid and Cloud systems, focus on makespan and the workload distribution. Since many parameters influence the data transfer speed rate, such as distance, noise etc, so there is communication latency among resources. But the mentioned methods do not consider the communication latency. A duplication task method is adopted in [17]. Because of the high-workload and delay in these systems, task duplication is not an efficient method because it increases the makespan of other application.

GVNS algorithm has proposed for heterogeneous systems [18]. This method combines the GA and variable neighbourhood search (VNS) algorithms. At first some solutions are produced in normal GA by considering a task priority similar to the HEFT. Then using two steps in VNS phase (two novel neighbourhood structures) predecessor for each task will be selected. It should be noted that, in the step one a task on the highest computation workload resource is chosen randomly, and will be reallocated to another randomly selected resource; then in second step, for all tasks on the highest communication workload resource, the VNS randomly selects a predecessor for each task, and reallocates the predecessors to the mentioned resource. Although it can be occurred a good selection in first phase, the second phase maybe cancels it. The GVNS tries to obtain a near minimum completion time, but here the bandwidth of the links is dedicated and the runtime of algorithm is long.

DCLS algorithm [19] is proposed for cloud computing systems. It is a dynamic list based scheduling that focuses on reducing the makespan. For each application, the DCLS lists the tasks by considering the graph topology without being influenced by the other parameters. Then the tasks will be allocated on resources by the order list. The task on the top of this list will be assigned to the resource that can finish the task at the earliest time. It continues until unassigned tasks are entered (as dynamic method). Here the communication cost of resources is considered.

In this approach we seek to obtain a suitable task scheduling with respect to the fully connected resource graph with different communication to computation ratios. This method makes initial population by merging the optimal characteristics in Best-Fit and Round-Robin and bi-objective resource allocation methods with a different GA stages such as mutation and selection generation.

3. PROBLEM DEFINITION

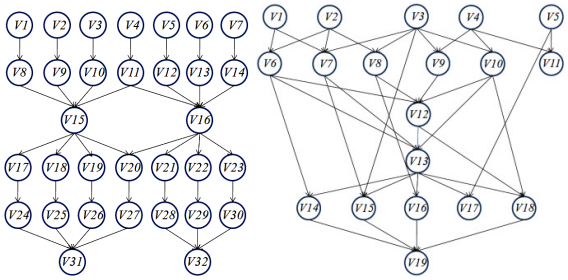
In cloud system environments, there is a data-center unit which is undertaken to collect and save the information of resources and tasks. Some static and dynamic key information such as: physical and virtual memory storage space, disk storage space, the load average of the resource node, the number of the running tasks, the current running tasks' number of threads, and the status of these tasks, CPU usage, etc, are collected. These data are updated frequently, in real-time [20].

Some static and dynamic information that impress for scheduling more than the other could be used here. Since a lot of tasks are received instantaneously, the workload and other dynamic information influence the selection of a good candidate resource for task.

An application is a workflow that contains a set of tasks that are connected to each other by precedence constraint. Each task will be executed and give an output dataset. This dataset is then sent to the next task as defined by the structure of the workflow. Generally, a workflow has the structure of a DAG (Directed Acyclic Graph): a graph where the nodes are the tasks and the edges are the precedence constraints [21].

According to many workflow projects, the workflow application structures can be categorized as either balanced-structure or unbalanced-structure [9]. In the balanced-structure workflows, nodes are related together considering certain level, but the unbalanced-structure application is complex, like [22] and has more relation among nodes where the level of some nodes is not certain (see Figure 1). When the size of the workflow is increased the processing time may become very long. Because of their heterogeneity, the heterogeneous system platform has 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 [23]. The CCR factor is computed by the average communication cost divided by the average computation cost on target system.
An application DAG graph is represented as $G=(V,E)$, where 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 this 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 the edges.



a) A balanced-structure graph b) An unbalanced-structure graph

Figure 1. Two types of graphs in workflow applications

At a cursory glance, the used notations in this paper are described as follow:

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
r_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

No task is dispatched until its precedence tasks are completes. The start time of each task is determined in accordance with the following equation:

$$T_{ST}(v_i,r_j) = \max\{ T_{avail}(r_j) , T_{ready}(v_i,r_j)\} \quad (1)$$

where, $T_{avail}(r_j)$ is the time when resource r_j is available for the execution of task v_i , $T_{ready}(v_i,r_j)$ is the time when all the predecessors of task v_i are executed, and all the necessary input data are available and could be transmitted to the processor r_j, through the following equation:

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

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:

$$T_{FT}(v_i,r_j)=T_{ST}(v_i,r_j)+(w(v_i)/r_j^{MIPS})+CC(v_i,v_j) \quad (3)$$

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 unbalanced-structured graph presented in figure 1(b). In these graphs some tasks are not in certain level, so the task selection to send to the ready queue is important. The resources are fully connected by different links' capabilities (see Figure 2):

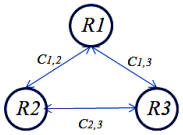


Figure. 2. An example of resource's connections

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

$$CC(v_i,v_j) = T_c(v_i,v_j) / C(_{k,l}) \quad (4)$$

Since heterogeneous distributed systems like cloud, have some properties that should be considered in the scheduling process, the important constraints are:

- The amount 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, and etc.
- The cloud environment is a collection of heterogeneous resources with dynamic hardware and software features such as: the node workload average, CPU usage, etc.

4. PROPOSED ALGORITHM

In this approach, a hybrid meta-heuristic method, based on GA is used, by considering the cloud computing system characteristics. Generally, the pseudo-code of the proposed algorithm is as follow:

Pseudo code of DWSGA method

Input: Available resources and unmapped tasks of an application
Output: An optimum derived scheduling

1. Make a virtual list of available resources from Data center information
2. **Repeat**
3. **For** all tasks $v_i \in V$ in each application graph do
4. Find the depth (in critical path)
5. **End** for
6. Set the priority of each task by equation (5) considering the graph topology
7. Update virtual list of resources
   // make initial population
8. **For** each chromosome **do**
9. Find the best fit resources for each task based on the execution time order by Best-fit && RR methods (is described in *4.2.* section)
10. Go to the next place in resource list for finding candidate resources for next chromosome
11. **If** the counter=last resource index **then**
12. Go to the first place in resource list
13. **End** for
    // doing other operations
14. Evaluate all chromosomes using equation (7)
15. **While** the stop conditions are met
16. One-point crossover operation
17. Goal-oriented mutation operation
18. Select the best chromosomes as elites

19. **End while**
20. Save the best solution
21. Dispatch all mapped tasks on candidate resources due to obtain the best solution
22. **Until** there are unscheduled application

In the following sections, the algorithm steps are described.

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 presents a chromosome in DWSGA method.

V3	V2	V4	V1	V7	…	V100
R7	R4	R1	R8	R8	…	R20

Figure. 3. A sample chromosome in schedule encoding in DWSGA

Here, first, the tasks of the graph are ordered on priority based on their influence on the other tasks in the graph for execution according to section 4.3; second, the tasks should mapped on 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.

4.2. Initial Population

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, for making 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 selected with minimum running time for the task from virtual list of available resources based on $w(v_i)$ and r_j^{MIPS} as resource speed at the first time.

Resource	R7	R4	R9	R1	R8	…
Workload (r_w)	100*103	70*103	120*103	50*103	90*103	…

Figure 4. A part of virtual list of available resources

The virtual list, consist of workload of available resources at current schedule time.
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 available resource virtual list, but for the second chromosome, it finds the best resource from the second place in list and so on like Round-Robin method but here for resource selection. If the counter is finished, the resource selection will be continued from first place. This process will continue until a population is made.

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.

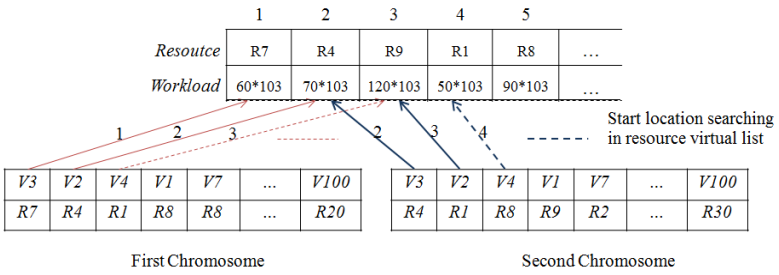


Figure 5. A sample of selecting candidate resources

A good property of this technique for making initial population is leading the algorithm to find an optimized solution, faster than other algorithms.

After an available resource is selected as the candidate, the virtual list will be updated based on the rest processing capacity on current workload of resources for designated tasks.

4.3. Task Prioritization

Before making the initial population, all tasks of entered application should be sorted based on priority in graph topology. Because some tasks on the unbalanced application graph cannot be categorized in levels, and each task's length and successor are different from 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 successor, the predecessor task should be executed before children. Completion time of each task influences the application completion time. So a bi-directional ordering method is being proposed that could be computed as the priority of tasks in both horizontal and vertical direction in graph topology, according to the following equation:

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

where, $d(v_j)$ is the depth of the 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 unbalanced-structured graph shown in figure 1(b) the set of successors of v_6 are: $T_{succ}(v_6)=\{v_{12},v_{13},v_{14}\}$. So the execution of task v_6 is efficient for its successor and the execution of each successor of v_6 is efficient for their successor alternatively, as well. Also v_{11} can be executed with v_{19} at the same time, because the depth of task v_{11} is 0. So we can select the more important successor of each task with an important depth. Thus, the limited range of depth selection is between α and β for selecting the most important successor for equation (5).

where, β represents the depth of successor with the longest sequence and α can be computed as:

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

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 one-point crossover is used. Two parents and their two genes are selected randomly. Then two other solutions by a change in resource sections of selected genes are created. For example in two selected chromosomes for the randomly selected point such as second to the last genes, the

candidate resources are changed by each other. Figure 6 illustrates the before and after crossover in mentioned example.

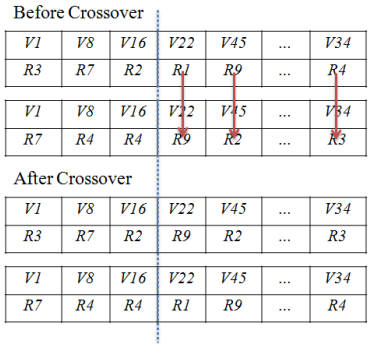


Figure. 6 Crossover operation method in this algorithm

4.5. Mutation

To make a mutation in solutions, a goal-oriented method that tries to lead the algorithm to reduce the makespan and workload of heaviest resource is used. The mutation steps are:

Pseudo code of mutation method

1. Select a chromosome, randomly
2. Compute the finished time of all resources, and find the resources with minimum (as r_min_w), and maximum (as r_max_w) workload in current selected chromosome
3. Select a task on r_max_w randomly
4. time_max=makespan by r_max_w
5. time_min= makespan by r_min_w
6. **If** (time_min < time_max) **then**
7. Assign the selected task on r_min_w
8. **Else**
9. **while** (! select a suitable resource)
10. Select next-best resource in list with lower workload
11. Assign the selected task on this resource
12. **End** while

The mutation operation causes the GA not to 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

In GA, to determine the value of a solution, it should be evaluated by a fitness function with efficient parameters in quality of solution. The fitness function is applied on all solutions and computes their values, and then a solution with the best value, based on parameters placement policy, is obtained as the minimum or maximum for the fittest solution.
Here, the fitness value of each solution is computed through:

$$Fitness = \max\ (T_{FT}(v_i, r_j)) \qquad (7)$$

where, $T_{FT}(v_i, r_j)$ is the completion time of task v_i mapped on resource r_j based on equation (3). So the maximum value of $T_{FT}(v_i, r_j)$ presents completion time of last task or the completion time of all tasks in workflow by current scheduling.

The chromosome with minimum fitness value is considered as the best solution among the others. Some of the best of chromosomes are will be selected by elitism method for next iteration.

Target is: Minimizing (*Fitness*) (8)

4.7. Stop Conditions

The algorithm would stop upon meeting a stop conditions. The conditions to end the process are [12]:

- Number of generations, will reach to a maximum bound
- The makespan of the best solution will not be changed after the certain number of generations
- All chromosomes converge to the same mapping

5. PERFORMANCE EVALUATION

This section presents the comparative evaluation DWSGA with two algorithms, GVNS and DCLS and demonstrates and evaluates the makespan and speedup rate subjects. The experiments are conducted considering cloud systems with respect to heterogeneity in resources and tasks properties as in some previous works such as random DAG generator in [17, 25] with different complexity rate, in order to simulate workflow applications. Some other parameters applied are from [18, 17, 24, 12] that are listed in tables 2. The parameters used in GA values such as probability for crossover operation and mutation operation are 0.2 and 0.125 respectively. Here the initial population size is 30.

Table 2. Simulation Parameters

Parameter	Value
Number of tasks in application	40 ~ 200
Task size	12 ~ 72 (*10^3 MI)
The number of resources	200
Resource speed	500~1000 (MIPS)
The bandwidth among resources	10 ~ 100 (mbps)
CCR value	~0.5 ~1.0 ~2.0

The average results are described and the figures are presented as follows:

5.1. The Makespan Evaluation

As mentioned, one of the main objectives in all methods is makespan. In this experiment, the proposed method was compared with two mentioned algorithms in different CCR rates. Figure 7 to 9 and Table 3 illustrate how the DWSGA reduces makespan in CCR 0.5 at least about by 0.7%, in CCR 1.0 at least about by 2.9% and in CCR 2.0 at least about by 4.7 in comparison with other mentioned algorithms in this section with respect to the different number of tasks and 1000 iterations. This is because of a good initial population and search space, that leads the search in mutation operation by reducing the workload from heaviest resource to the lightest and if not found it selects the next lightest resource.

5.2. The Speedup Evaluation

To recognize the quality of distribution of workload among resources, the speedup value is evaluated. This can be computed by equations in [10] and [18], that is, dividing the completion time of tasks in one resource t (U^{ct}), in parallel resources.

Speedup= U^{ct} / makespan (9)

So a higher value represents better result. According to experiments in Figure 7 to 9 and Table 3, the DWSGA distributes better workload then the two mentioned algorithms. This is because the search is being led to reduce workload from high to low in resources in mutation and selection operations. The results are optimized at least about by 1.2%, 2% and 5.7% in CCR 0.5, 1.0 and 2.0 respectively.

Table 3. The comparison of DWSGA with others per cent

	Makespan		Speedup	
	DCLS	**GVNS**	**DCLS**	**GVNS**
CCR 0.5	0.7%	9.1%	1.2%	16.5%
CCR 1.0	2.9%	13.0%	2.0%	14.7%
CCR 2.0	4.7%	13.6%	5.7%	15.4%

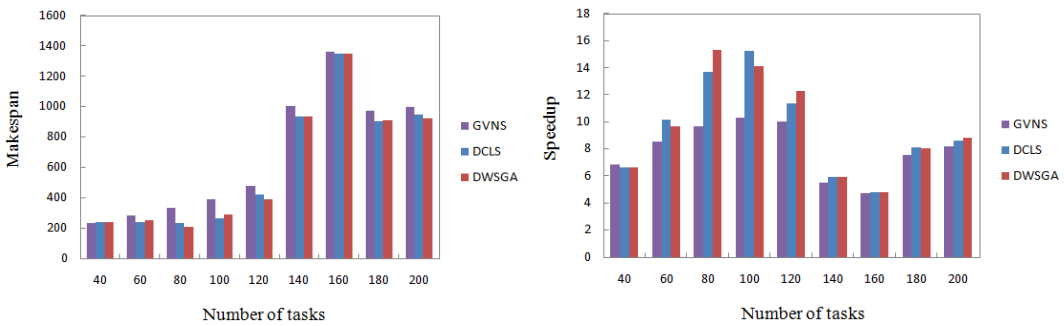


Figure7. The makespan and Speedup in CCR 0.5

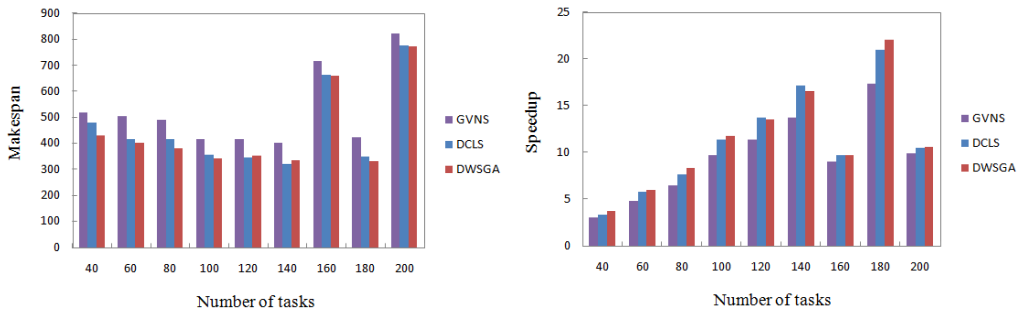


Figure 8. The makespan and Speedup in CCR 1.0

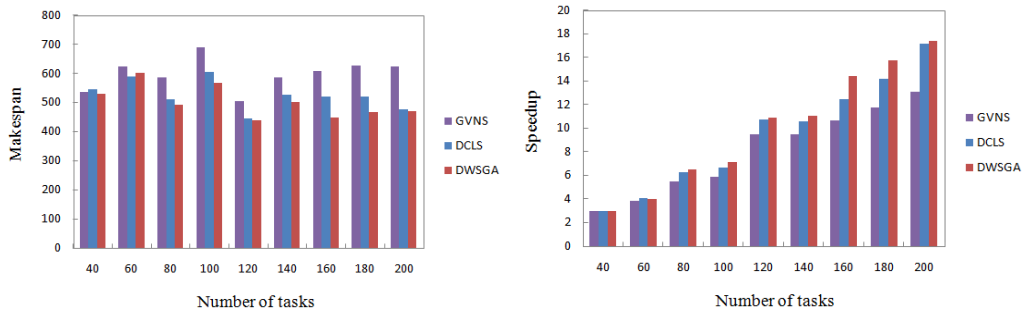


Figure 9. The makespan and Speedup in CCR 2.0

6. CONCLUSIONS

In this approach, we proposed a hybrid heuristic scheduling method for workflow applications in cloud systems by considering efficient parameters. The GA based proposed algorithm, tries to get an optimized solution in a rapid manner with respected to completion time and distribution of workload on resources. It uses some conversant methods such as making initial population by ordering the tasks, based on a bi-directional priority method and other goal-oriented operations that lead the algorithm to fulfill the objectives with speeding up the good solution finding process. The DWSGA controls the search by an especial mutation method that reassigns resources based on workload in addition to considering the most effective task. The DWSGA results are compared to DCLS and GVNS algorithms. The produced solution through this proposed algorithm is perceived and it improves the results in comparison with them. In the next work we want to present a method that would support mapping the resources on tasks in other objectives with a new perspective.

REFERENCES

[1] R. P. BRENT, (July 1989), " Efficient Implementation of the First-Fit Strategy for Dynamic Storage Allocation" Australian National University, ACM Transactions on Programming Languages and Systems, Vol. 11, No. 3.

[2] Daniel Nurmi & Rich Wolski & Chris Grzegorczyk & Graziano Obertelli & Sunil Soman & Lamia Youseff & Dmitrii Zagorodnov, (2009), "The Eucalyptus open-source cloud-computing system", IEEE International Symposium on Cluster Computing and the Grid (CCGrid).

[3] Hesam Izakian & Ajith Abraham & Václav Snášel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments".

[4] Henri Casanova & Frédéric Desprez & Frédéric Suter, (2010), "On cluster resource allocation for multiple parallel task graphs", ELSEVIER, J. Parallel and Distributed Computing, 70, 1193–1203.

[5] Suraj Pandey, (2010), "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.

[6] S. Porto & C. Ribeiro, (1995), "A tabu search approach to task scheduling on heterogeneous processors under precedence constraints", International Journal of High Speed Computing, 7, 45–72.

[7] A. Kalashnikov & V. Kostenko, (2008), "A parallel algorithm of simulated annealing for multiprocessor scheduling", International Journal of Computer and Systems Sciences 47, 455–463.

[8] Jia Yu & Rajkumar Buyya, K. Ramamohanarao, (2009), "Workflow Scheduling Algorithms for Grid computing", Department of Computer Science and Software Engineering, The University of Melbourne, VIC 3010, Australia. http://www.cloudbus.org/reports.

[9] Myungryun Yoo, (2009), "Real-time task scheduling by multiobjective genetic algorithm", ELSEVIER, The Journal of Systems and Software, 82, 619–628.

[10] Fatma.A. Omara & Mona. M. Arafa, (2010), "Genetic algorithms for task scheduling problem", ELSEVIER, J. Parallel and Distributed Computing, 70, 13_22.

[11] ADNAN Fida, (2008), "Workflow Scheduling for Service Oriented Cloud Computing", MSc Thesis, College of Graduate Studies and Research In Partial Fulfillment, Department of Computer Science University of Saskatchewan Saskatoon.

[12] Arash Ghorbannia Delavar & Yalda Aryan, (2012), "A Goal-Oriented Workflow Scheduling in Heterogeneous Distributed Systems", International Journal of Computer Applications, 0975 – 8887.

[13] E. Ilavarasan & P. Thambidurai, (2007), "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments", Journal of Computer Sciences 3 (2): 94-103.

[14] S. Padmavathi & S. Mercy Shalinie, (2010), "Scable Low Complexity Task Scheduling Algorithm for Cluster of Workstations", Journal of Engineering Science and Technology Vol. 5, No. 3, 332 – 341.

[15] Zhiao Shi & Jack J. Dongarra, (2006), "Scheduling workflow applications on processors with different capabilities", Future Generation Computer Systems 22, 665–675.

[16] Xiaofeng Wang & Chee Shin Yeo & Rajkumar Buyya & Jinshu Su, (2011), "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm", ELSEVIER, Future Generation Computer Systems 27, 1124–1134.

[17] Xiaoyong Tang & Kenli Li & Guiping Liao & Renfa Li, (2010), "List scheduling with duplication for heterogeneous computing systems", J. Parallel and Distributed Computing, 70, 323_329.

[18] Yun Wen & Hua Xu & Jiadong Yang, (2011), "A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system", ELSEVIER, Information Sciences, 181, 567–581.

[19] Jiayin Li & Meikang Qiu & Zhong Ming & Gang Quan & Xiao Qin & Zonghua Gue, (2012), "Online optimization for scheduling preemptable tasks on IaaS cloud systems", ELSEVIER, Journal of Parallel and Distributed Computing, 72, 666–677.

[20] Junwei Ge & Bo Zhang & Yiqiu Fang, (2010), "Research on the Resource Monitoring Model Under Cloud Computing Environment", WISM, LNCS 6318, pp. 111–118, Springer, Verlag Berlin Heidelberg.

[21] Xiaoyong Tang & Kenli Li & Renfa Li & Bharadwaj Veeravalli, (2010), "Reliability-aware scheduling strategy for heterogeneous distributed computing systems", J. Parallel and Distributed Computing, 70, 941_952.

[22] Arash Ghorbannia Delavar & Vahe Aghazarian & Sanaz Litkouhi & Mohsen Khajeh naeini, (2011), "A Scheduling Algorithm for Increasing the Quality of the Distributed Systems by using Genetic Algorithm", International Journal of Information and Education Technology, Vol. 1, No. 1, ISSN: 2010-3689,.

[23] M. Mezmaz & N. Melab & Y. Kessaci & Y.C. Lee c & E.-G. Talbi & A.Y. Zomaya & D. Tuyttens, (2011) "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems", ELSEVIER, J. Parallel and Distributed Computing.

[24] Young Choon Lee & Albert Y. Zomaya, (2010) "Rescheduling for reliable job completion with the support of clouds", Future Generation Computer Systems 26, 1192_1199.

[25] P. Chitra & R. Rajaram & P. Venkatesh, (2011) "Application and comparison of hybrid evolutionary multiobjective optimization algorithms for solving task scheduling problem on heterogeneous systems", Applied Soft Computing 11, 2725–2734.

Authors

Yalda Aryan received the B.Sc. from Azad University, Arak, IRAN, and M.Sc. in Payam Noor University, Tehran, IRAN, in 2012, both in computer engineering. She achieved the top student award in M.Sc. course. She is a teacher in computer science since 2002. She is the head of Educational Computer Group of Research Center of Teachers, Isfahan, Iran. Her research interests include computational intelligence, Cluster computing and cloud computing, Data mining in medicine.

Arash Ghorbannia Delavar received the MSc 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.