A Goal-Oriented Workflow Scheduling in Heterogeneous Distributed Systems

Arash Ghorbannia Delavar
Payam Noor University
Department of Computer
PO BOX 19395-3697, Tehran, IRAN

Yalda Aryan
Payam Noor University
Department of Computer
PO BOX 19395-3697, Tehran, IRAN

ABSTRACT
In heterogeneous distributed systems like grid and cloud computing infrastructures, the major problem is the task scheduling which can have much impact on system performance. For some reasons, such as heterogeneous and dynamic features and the dependencies among the requests, this issue is known as a NP-hard problem. In this article a hybrid meta-heuristic method based on Genetic Algorithm (GMSW) is being proposed in order to find a suitable solution for mapping the requests on resources. The proposed method tries to obtain the response quickly, with some goal-oriented operations. It begins, through making a good initial population by merging some features of the Best-Fit and Round Robin methods and a bi-directional tasks prioritization in unbalanced-structured workflow, considering their impact on each other, based on graph topology. Some other operations control and lead the algorithm steps in order to obtain the solution by using efficient parameters in the mentioned systems. Here the focus is on optimizing the makespan and reliability, by considering a good distribution of workload on resources. The experiments here indicate that the GMSW improves the results, with the increasing number of tasks in application graph, for the mentioned objectives. The results are compared with other studied algorithms.

General Terms
Internet and Distributed Computer Systems, Heterogeneous distributed systems, Cloud computing, Grid computing, Scheduling.

Keywords
Heterogeneous distributed systems, Grid computing, Cloud computing, Workflow scheduling, Reliability, Genetic Algorithm.

1. INTRODUCTION
Heterogeneous distributed systems (HDS) consist of millions of heterogeneous computing nodes interconnected through arbitrary network architecture and are made to achieve high-throughput computing resource pools [1]. Grid and Cloud computing are paradigms for HDS system promised to deliver the utility computing view with some desirable properties such as sharing the many dynamic resources by virtualization technology in order to meet the requirements of widely varying requests.

Task scheduling is a key process in these systems, that is, mapping the requests on resources in an efficient manner by considering the environment characteristics. Due to its heterogeneous and dynamic properties of resources, in addition to many number of tasks with different characteristics this issue is referred to as a NP-hard 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, the scheduling decisions must rely on finding the best solution within possibilities.

Many methods are proposed for this problem. Each method often focuses on limited number of parameters and main objectives such as the completion time of all tasks (makespan), reliability or distribution of workload on resources. For example, many fundamental heuristic methods like greedy (First-fit) [2] and Round-Robin (RR) [3], Min-min, Max-min or Sufferage [4] try to achieve the makespan. Moreover, some different dynamic list scheduling methods are presented for HDS systems [5] which often do not consider the latency among resources and focus on the makespan and/or another main issue.

Some meta-heuristic based methods are presented to solve NP problems such as: particle swarm optimization (PSO) [6], tabu search (TS) [7], simulated annealing (SA) [8], genetic algorithm (GA) etc. In contrast, GA by [9] [10] [11] 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 methods apply random manner in some steps like preparing the initial population or adopting a different prioritization method, for task ordering in the same level in the workflow[12], and some of them use simple graph with the most two output nodes.

In our last article, we proposed a QoS-based dynamic scheduling method for independent task scheduling with focusing on deadline [13]. Here, we suggest a hybrid meta-heuristic method (GMSW) based on GA in order to find a proper scheduling solution with unbalanced-structure computational applications. The proposed algorithm here would make a reduction in number of GA operation iterations by making an optimized initial population, with respect to the reliability and suitable distribution of workload on resources, simultaneously. This method obtains the solutions by two evaluation functions, one function measures priority of each task in DAG for bi-directional form based on their influence on the other tasks, and another function evaluate the value of the produced solutions.

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

2. RELATED WORK

The known policies, First-Fit or RR are non-preemptive methods and are used by some cloud systems such as Eucalyptus. 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 distribution.

Some heuristic proposed algorithms in HDS, like the ones proposed by [14], [15] and [16], focus on makespan and the workload distribution. However these algorithms do not consider communication latency among resources for data transferring and related cost. Although the bandwidth of links will be dedicated in distributed systems, many parameters influence the data transfer speed rate, such as distance, noise etc. The list-based scheduling [17] focused on the three mentioned objectives. A duplication task method is adopted in [18]. It should be noted that, since in HDS the workload against resources is great, task duplication is not an efficient method because it increases the workload and makespan of other application.

GVNS algorithm is proposed for heterogeneous systems [19]. This algorithm incorporates GA with the variable neighborhood search (VNS) algorithm. Here, some solutions are made in normal GA by considering a task ordering similar to the HEFT. Then two novel neighborhood structures as VNS phase are applied in the solutions. In one of the steps of VNS, a task on resource with the highest computation workload is chosen randomly, and will be reallocated to another randomly selected resource. In the other step of VNS, for all tasks on resource with the highest communication workload, the GVNS randomly selects a predecessor for each task, and reallocates the predecessors to the mentioned resource. The GVNS seeks to obtain a near minimum completion time where task-machine matching, and task scheduling are integrated. Here the bandwidth of the links is dedicated, the runtime of algorithm is long and in the VNS phase, two steps' effects cancel each other.

CMMS algorithm [20] is proposed for cloud computing systems. It is a list based scheduling that focuses on reducing the makespan using min-min algorithm. For each application, the CMMS lists the tasks by considering the graph topology without being influenced by the other parameters. Then the tasks will be allocated on resources in the order indicated by the list. The task on the top of this list will be assigned to the resource that can finish the task at the earliest time. Here the communication cost of resources is considered.

The LAGA algorithm [21] 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.

In this study the unbalanced-structured workflow is used which should be assigned in the pool of resources with fully connected graph and different communication latency. The algorithm tries to obtain a good proportion among the three mentioned objectives by making initial population method through a contribution to the optimal characteristics in Best-Fit and Round Robin methods, in order to find the response and other goal-oriented operations in a rapid manner.

3. PROBLEM DEFINITION

In Grid and Cloud environments, there is an information service or data-center system which is assumed to collect and save the information of resources and tasks. Some key static information such as: physical memory storage space, virtual memory storage space, disk storage space, etc and some dynamic information such as: the load average of the 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 [22].

Some static and dynamic information for scheduling 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 data set is then sent to the next task as defined by the structure of the workflow. Generally, a workflow has the structure of a DAG (Direct Acyclic Graph): a graph where the nodes are the tasks and the edges are the precedence constraints [17].

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 [23] 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 [24]. 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.

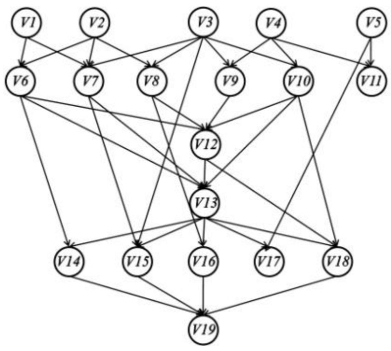

Fig 1: An unbalanced-structure graph.

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:

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

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

$$CC(v_i, v_j) = T_c(v_i, v_j) * C(r_k, r_j) \quad (4)$$

where, $T_c(v_i,v_j)$ is the output size of v_i to v_j and $C(r_k,r_j)$ is the communication latency between resources r_k and r_j for v_i and v_j respectively.

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):

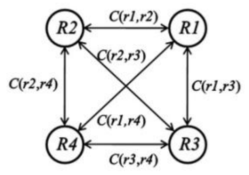

Fig 2: An example of resources' connections

Since HDS systems like grid and 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 system is a collection of heterogeneous resources with dynamic hardware and software features such as: the node workload average, CPU usage, etc.

4. THE PROPOSED ALGORITHM

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

Pseudo code of GMSW method

```
Input: Available resources and unmapped tasks of an application
Output: An optimum derived scheduling
1.  Make a virtual list of available resources (ARVL) from Data center
2.    For all tasks $v_i \in V$ in each application graph do
3.        Find the depth (in critical path)
4.    End for
5.    Set the priority of each task by equation (5) considering the graph topology
6.      Update virtual list of resources
        // make initial population
7.    For each chromosome do
8.        Find the best fit resources for each task based on the execution time order by Best-fit && RR methods (is described in 4.3. section)
9.        Go to the next place in resource list for finding candidate resources for next chromosome
10.     If the counter=last resource index then
11.       Go to the first place in resource list
12.   End for
      // doing other operations
13.   Evaluate all chromosomes using equation (9)
14.   While the stop conditions are met
15.     One-point crossover operation
16.     Goal-oriented mutation operation
17.     Select the best chromosomes as elites
18.   End while
19.   Save the best solution
20.   Dispatch all mapped tasks on candidate resources due to obtain the best solution
```

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 GMSW method.

V17	V2	V8	V0	V12	…	V24
R2	R9	R3	R1	R8	…	R5

Fig 3: A sample chromosome in GMSW

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 4.2; 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 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:

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

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:

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

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.

4.3 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 as ARVL 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	…
Failure rate (r_f)	0.0001	0.001	0.0005	0.0004	0.001	…

Fig 4: A part of an ARVL

The ARVL consist of workload and failure rate of available resources at current schedule time. The failure rate factor can be computed as in [21].

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 ARVL, but for the second chromosome, it finds the best resource from the second place in ARVL 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.

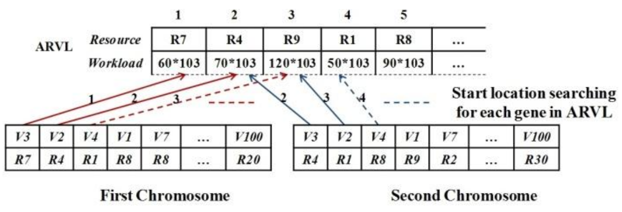

Fig 5: A sample of selecting candidate resources

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 ARVL list will be updated based on the rest processing capacity on current workload of resources for designated tasks.

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.

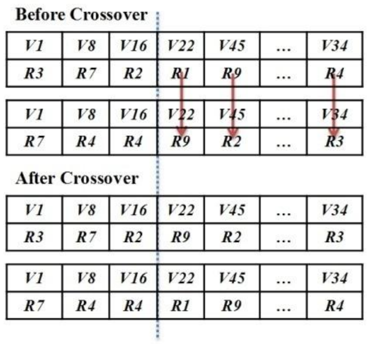

Fig 6: Crossover operation method in this algorithm

4.5 Mutation

To make a mutation in solutions, a goal-oriented method that tries to lead the algorithm to reduce makespan considering reliability 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 task with highest $d(v_j)$ on r_max_w
4. time_max=makespan by r_max_w
```

```
5.  time_min= makespan by r_min_w
6.  **If** ((failure frequency of r_min_w<failure frequency
    of r_max_w) && (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, and lower failure rate
11.                 Assign the selected task on this
                resource
12.         **End** while
```

The failure frequency of resource *i* can be computed by:

$$r_i^{ff} = r_i^f / r_i^{MIPS} \qquad (7)$$

where, r_i^f is the failure rate of resource *i*.

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:

$$fp = \sum_{j=1}^{n} (\sum_{i=1}^{m} T_{FT}\,(v_i, r_j)) * r_j^f \qquad (8)$$

$$Fitness = \max(T_{FT}(v_i, r_j)) + fp \qquad (9)$$

where, fp is the failure probability in scheduling and $T_{FT}(v_i,r_j)$ 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.

$$\text{Target is} = \text{Minimizing}(\text{Fitness}) \qquad (10)$$

4.7 Stop conditions

The algorithm would stop upon meeting a stop conditions. The conditions to end the process are [13]:

- 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 GMSW with three algorithms, GVNS, CMMS and LAGA and demonstrates and evaluates the three makespan, reliability, and speedup rate subjects. The experiments are conducted considering HDS systems with respect to heterogeneity in resources and tasks properties as in some previous works such as random DAG generator in [21, 25] with different complexity rate, in order to simulate workflow applications. Some other parameters applied are from [19, 21, 26, 13] that are listed in tables 1. 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 1. 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)
Resource failure rate	10^{-3} ~ 10^{-4} (failure/h)
The communication latency among resources	10 ~ 100 (ms)
CCR value	~0.2 ~0.5 ~1.0 ~2.0

The average results of the three objectives 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 three mentioned algorithms. Figure 7 illustrates how the GMSW reduces makespan for at least about by 2% and at most about by 15%in comparison with other mentioned algorithms in this section with respect to the different number of tasks and 1000 iterations. Also in different iterations for 200 tasks (in GA-based methods), a faster convergence in finding the optimum result is perceived. 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 and consideration task with the highest *d*(v_j).

5.2 The reliability evaluation

The proposed method here is compared to LAGA method. The experiments showed that the failure probability of tasks in GMSW is close to that of the LAGA, in about by 0.6%. Because in mutation operation in the proposed method, both failure frequency and workload are tested and finding a resource. So reducing the makespan causes to reduce the failure probability. Reducing the failure probability or increasing the reliability by a faster convergence is shown in Figure 8 by different iterations for 200 tasks. This is because the convergence in the completion time of application based on the equation (8). The reliability in scheduling can be computed through e^{-fp} [21].

5.3 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 [11] and [19], that is, dividing the completion time of tasks in one resource t (U^{ct}), in parallel resources.

$$\text{Speedup} = \text{U}^{\text{ct}}/\text{makespan} \qquad (11)$$

So a higher value represents better result. According to experiments in Figure 9, the GMSW distributes better workload then the three 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 4%. The convergence of results in different iterations, are indicated in Figure 9.

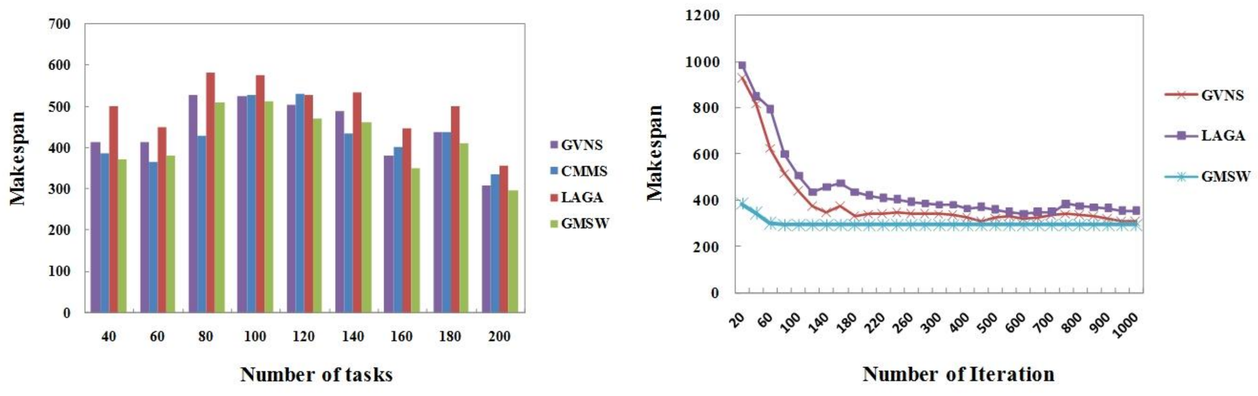

Fig7: Average results of makespan

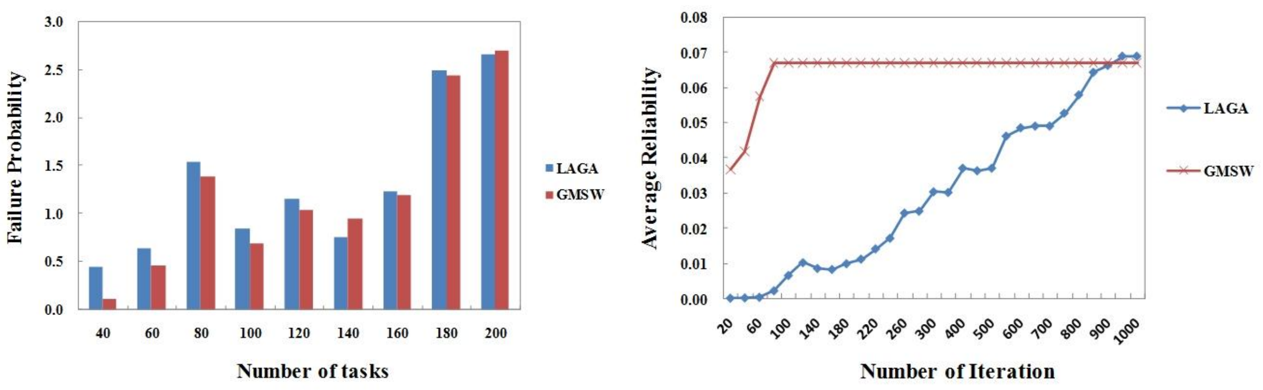

Fig 8: Average results of reliability

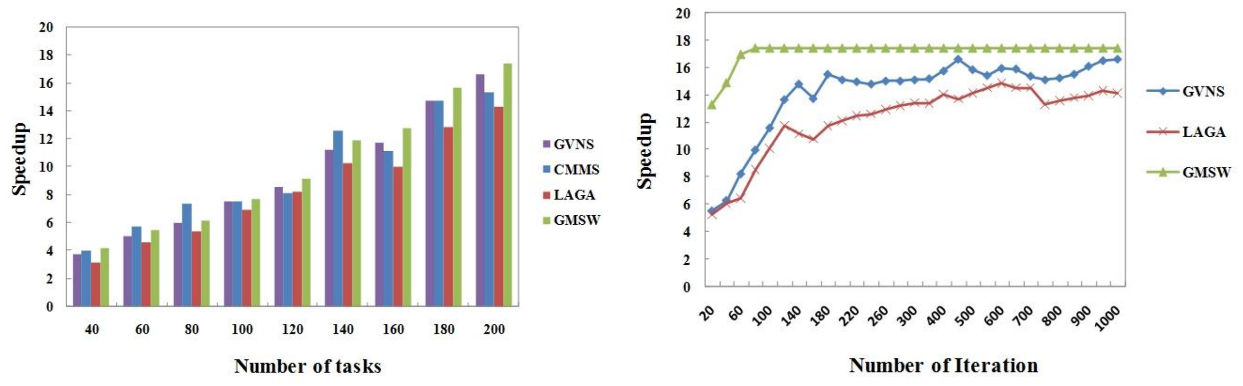

Fig 9: Average results of speedup

6. CONCLUSION

In this article, a hybrid meta-heuristic scheduling method is suggested for workflow applications in HDS systems by fully connected resources with different communication costs.

The GA based proposed algorithm here, tries to get an optimized in a rapid manner with respected to completion time, reliability and distribution of workload on resources. It orders all tasks, by a bi-orientation priority method considering their horizontal and vertical influences based on graph topology.

Some goal-oriented operations are applied to fulfill the objectives. To make the initial population and to select good candidate resources the aggregation features of two methods, the Best-Fit and Round Robin are used. Consequently this technique speeds up the good solution finding process. The GMSW leads the search by an especial mutation method that reassigns resources based on workload and failure frequency in addition to considering the most effective task. The GMSW results are compared to LAGA, CMMS 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 for QoS-constraint with a new perspective.

7. REFERENCES

[1] Sahoo, B. Avinash Ekka, A. 2006. Performance Analysis Of Concurrent Tasks Scheduling Schemes In a Heterogeneous Distributed Computing System. National Conference on Computer Science & Technology.

[2] BRENT, R. P. 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, (July 1989).

[3] Nurmi, D. Wolski, R. Grzegorczyk, C. Obertelli, G. So-man, S. Youseff, L. and Zagorodnov, D. 2009. The Eucalyptus open-source cloud-computing system. IEEE International Symposium on Cluster Computing and the Grid (CCGrid).

[4] Izakian, H. Abraham, A. Member, S. Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments.

[5] Casanova, H. Desprez, F. Suter, F. 2010. On cluster resource allocation for multiple parallel task graphs. ELSEVIER, J. Parallel and Distributed Computing, 70 (2010) 1193–1203.

[6] Pandey, S. 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 (December 2010).

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

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

[9] Yu, J. Buyya, R. Ramamohanarao, K. 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.

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

[11] Omara, F.A. Arafa, M. M. 2010. Genetic algorithms for task scheduling problem. ELSEVIER, J. Parallel and Distributed Computing, 70 (2010) 13_22.

[12] Fida, A. 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.

[13] Ghorbannia Delavar, A. Aryan, Y. 2011. A Synthetic Heuristic Algorithm for Independent Task Scheduling in Cloud Systems. IJCSI International Journal of Computer Science Issues, 1694-0814.

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

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

[16] Shi, Z. Dongarra, J. J. 2006. Scheduling workflow applications on processors with different capabilities. Future Generation Computer Systems 22 (2006) 665–675.

[17] Tang, X. Li, K. Li, R. Veeravalli, B. 2010. Reliability-aware scheduling strategy for heterogeneous distributed computing systems, J. Parallel and Distributed Computing, 70 (2010) 941_952.

[18] Tang, X. Li, K. Liao, G. Li, R. 2010. List scheduling with duplication for heterogeneous computing systems.J. Parallel and Distributed Computing, 70 (2010) 323_329.

[19] Wen, Y. Xu, H. Yang, J. 2011. A heuristic-based hybrid genetic-variable neighbourhood search algorithm for task scheduling in heterogeneous multiprocessor system. ELSEVIER, Information Sciences, 181 (2011) 567–581.

[20] Li, J. Qiu, M. Ming, Z. Quan, G. Qin, X. Gue, Z. 2012. Online optimization for scheduling preemptable tasks on IaaS cloud systems. ELSEVIER, Journal of Parallel and Distributed Computing, 72 (2012) 666–677.

[21] Wang, X. Shin Yeo, Ch. Buyya, R. Su, J. 2011. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm, ELSEVIER, Future Generation Computer Systems 27 (2011) 1124–1134.

[22] Ge, J. Zhang, B. and Fang, Y. 2010. Research on the Resource Monitoring Model Under Cloud Computing Environment. WISM 2010, LNCS 6318, pp. 111–118, Springer, Verlag Berlin Heidelberg.

[23] Ghorbannia Delavar, A. Aghazarian, V. Litkouhi S. and Khajeh naeini, M. 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.

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

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

[26] Lee, Y. C. Zomaya, A. Y. 2010. Rescheduling for reliable job completion with the support of clouds. Future Generation Computer Systems 26 (2010) 1192_1199.