5

Task Scheduling in Grid Environment Using Simulated Annealing and Genetic Algorithm

Wael Abdulal[1], Ahmad Jabas[1], S. Ramachandram[1] and Omar Al Jadaan[2]
[1]*Department of Computer Science and Engineering, Osmania University*
[2]*Medical and Health Sciences University*
[1]*India*
[2]*United Arab Emirates*

1. Introduction

Grid computing enables access to geographically and administratively dispersed networked resources and delivers functionality of those resources to individual users. Grid computing systems are about sharing computational resources, software and data at a large scale. The main issue in grid system is to achieve high performance of grid resources. It requires techniques to efficiently and adaptively allocate tasks and applications to available resources in a large scale, highly heterogeneous and dynamic environment.

In order to understand grid systems, three terms are reviewed as shown below:

1. **Virtualization**: The Virtualization term in grids refers to seamless integration of geographically distributed and heterogeneous systems, which enables users to use the grid services transparently. Therefore, they should not be aware of the location of

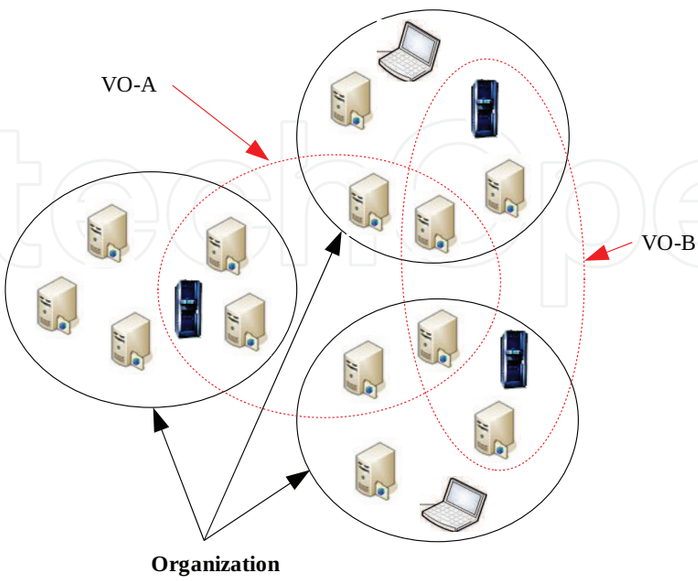


Fig. 1. Two virtual organizations are formed by combining a three real organizations

computing resources and have to submit their service request at just one point of entry to the grid system. Foster introduced the concept of Virtual Organization (VO) (Foster et al., 2001). He defines VO as a *"dynamic collection of multiple organizations providing flexible, secure, coordinated resource sharing"*. Figure 1 shows three actual organizations with both computational and data resources to share across organizational boundaries. Moreover, the same figure forms two VOs, A and B, each of them can have access to a subset of resources in each of the organizations (Moallem, 2009). Virtualization is a mechanism that improves the usability of grid computing systems by providing environment customization to users.

2. **Heterogeneity**: The organizations that form part of VO may have different resources such as hardware, operating system and network bandwidth. Accordingly, VO is considered as a collection of heterogeneous resources of organizations.
3. **Dynamism**: In the grid system, organizations or their resources can join or leave VO depending on their requirements or functional status.

Grid systems provide the ability to perform higher throughput computing by usage of many networked computers to distribute process execution across a parallel infrastructure. Nowadays, organizations around the world are utilizing grid computing in such diverse areas as collaborative scientific research, drug discovery, financial risk analysis, product design and 3−D seismic imaging in the oil and gas industry (Dimitri et al., 2005).

Interestingly, task scheduling in grid has been paid a lot of attention over the past few years. The important goal of task scheduling is to efficiently allocate tasks as fast as possible to avialable resources in a global, heterogeneous and dynamic environment. Kousalya pointed out that the grid scheduling consists of three stages: First, resource discovery and filtering. Second, resource selection and scheduling according to certain objective. Third, task submission. The third stage includes the file staging and cleanup (Kousalya & Balasubramanie, 2009; 2008). High performance computing and high throughput computing are the two different goals of grid scheduling algorithm. The main aim of the high performance computing is to minimize the execution time of the application. Allocation of resources to a large number of tasks in grid computing environment presents more difficulty than in conventional computational environments.

The scheduling problem is well known NP-complete (Garey & Johnson, 1979). It is a combinatorial optimization problem by nature. Many algorithms are proposed for task scheduling in grid environments. In general, the existing heuristic mapping can be divided into two categories (Jinquan et al., 2005):

First, online mode, where the scheduler is always in ready mode. Whenever a new task arrives to the scheduler, it is immediately allocated to one of the existing resources required by that task. Each task is considered only once for matching and scheduling.

Second, batch mode, the tasks and resources are collected and mapped at prescheduled time. This mode takes better decision because the scheduler knows the full details of the available tasks and resources. This chapter proposes a heuristic algorithm that falls in batch mode Jinquan et al. (2005).

However, this chapter studies the problem of minimizing makespan, i.e., the total execution time of the schedule in grid environment. The proposed Mutation-based Simulated Annealing (MSA) algorithm is proved to have high performance computing scheduling algorithm. MSA algorithm will be studied for random and Expected Time to Compute (ETC) Models.

2. Related works

One salient issue in grid is to design efficient schedulers, which will be used as a part of middleware services to provide efficient planning of users' tasks to grid resources. Various scheduling approaches that were suggested in classical parallel systems literature are adopted for the grid systems with appropriate modifications. Although these modifications made them suitable for execution in grid environment, these approaches failed to deliver on the performance factor. For this reason, Genetic Algorithm (GA) and Simulated Annealing (SA) algorithm, among others, used to solve difficulties of task scheduling in grid environment. They gave reasonable solutions comparing with classical scheduling algorithms. GA solutions for grid scheduling are addressed in several works (Abraham et al., 2000; Carretero & Xhafa, 2006; Abraham et al., 2008; Martino & Mililotti, 2002; Martino, 2003; Y. Gao et al., 2005). These studies ignored how to speed up convergence and shorten the search time of GA.

Furthermore, SA algorithm was studied in previous works Fidanova (2006); Manal et al. (2011). These works show important results and high quality solutions indicating that SA is a powerful technique and can be used to solve grid scheduling problem. Moreover, Jadaan, in Jadaan et al. (2009; 2010; 2011), exposed the importance of rank in GA.

The authors Wook& Park (2005) proved that both GA and SA algorithms have complementary strengths and weaknesses, accordingly, they proposed a new SA-selection to enhance GA performance to solve combinatorial optimization problem. The population size which they use is big that makes time consumed by algorithm large, specially when problem size increases. While Kazem tried to solve a static task scheduling problem in grid computing using a modified SA (Kazem et al., 2008). Prado propose a fuzzy scheduler obtained by means of evolving a fuzzy scheduler to improve the overall response time for the entire workflow (Prado et al., 2009). Rules of this evolutionary fuzzy system is obtained using genetic learning process based on Pittsburgh approach.

Wael proposed an algorithm that minimizes makespan, flowtime and time to release as well as it maximizes reliability of grid resources (Wael & Ramachandram, 2011). It takes transmission time and waiting time in resource queue into account. It uses stochastic universal sampling selection and single exchange mutation to outperform other GAs.

Lee et al. (2011) provided Hierarchical Load Balanced Algorithm (HLBA) for Grid environment. He used the system load as a parameter in determining a balance threshold. the scheduler adapts the balance threshold dynamically when the system load changes. The loads of resource are CPU utilization, network utilization and memory utilization.

P.K. Suri & Singh Manpreet (2010) proposed a Dynamic Load Balancing Algorithm (DLBA) which performs an intra-cluster and inter cluster load balancing. Intra-cluster load balancing is performed depending on the Cluster Manager (CM). CM decides whether to start the local balancing based on the current workload of the cluster which is estimated from the resources below it. Inter-cluster load balancing is done when some CMs fail to balance their workload. In this case, the tasks of the overloaded cluster will be transferred to another cluster which is underloaded. In order to check the cluster overloading, they introduced a balanced threshold. If the load of cluster is larger than balanced threshold, load balancing will be executed. The value of balanced threshold is fixed. Therefore, the balanced threshold is not appropriate for the dynamic characteristics in the grid system.

Chang et al. (2009) introduced Balanced Ant Colony Optimization algorithm (BACO) to choose suitable resources to execute tasks according to resources status. The pheromone

update functions perform balancing to the system load. While local pheromone update function updates the status of the selected resource after tasks assignment. Global pheromone update the status of each resource for all tasks after completion of all tasks.

In this chapter MSA maintains two solutions at a time, and it uses single exchange mutation operator as well as random-MCT heuristic (demonstrated in subsection 5.2).

Previous works, namely, Wael et al. (2009c;b;a) considered the minimization of the makespan using GA based on Rank Roullete Wheel Selection (RRWSGA). They use standard deviation of fitness function as a termination condition of the algorithm. The aim of using standard deviation is to shorten the the time consumed by the algorithm with taking into account reasonable performance of Computing resources (97%).

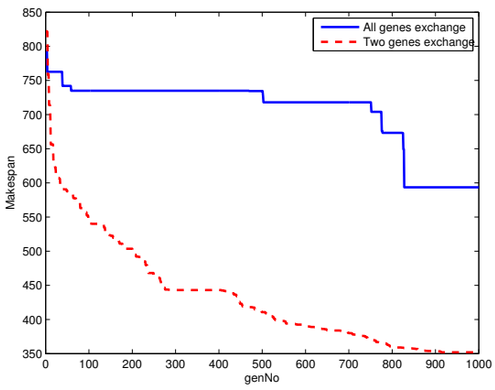
Yin introduced GA which used standard deviation less than (0.1) as stopping criterion to limit the number of iterations of GA (Yin et al., 2007). This algorithm has drawbacks such as low quality solutions (almost same as low quality solutions of standard GA), generating initialization population randomly (even though the time consumed by algorithm is small comparing with standard GA), and mutation depends on exchange of every gene in the chromosome. This mutation will destroy the good information in subsequent chromosomes in next generations. In order to illustrate the usefulness of this work, next section explains the motivation behind it.

3. Motivation

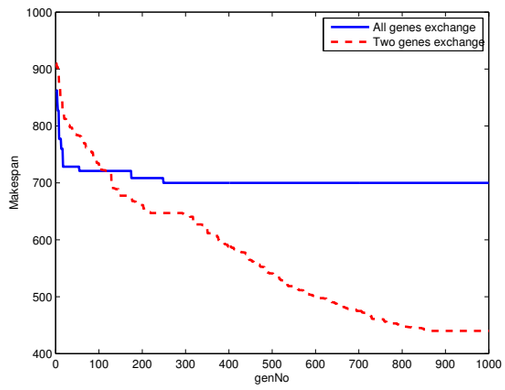
The motivation of this chapter is to develop a grid scheduling algorithm that can introduce a high utilization of grid resources, speed up convergence to the optimal or sub-optimal solution, shorten time consumed by algorithm as much as possible, improve load balancing of grid resources to the best optimization level, and minimize the schedule length, i.e., makespan.

The quality of solution produced by Yin's algorithm for grid scheduling is low. In other words, Yin proposed GA where mutation with all genes of chromosome are changed (Yin et al., 2007). This type of mutation is not always suitable to solve complex optimization problem such as grid task scheduling. It destroys the information in the chromosomes and does not help to find the optimal or near-optimal solution for the problem at hand. Moreover, the population's initialization of Yin's algorithm is generated randomly without using any heuristic in initialization phase of GA. The heuristics allow GA to search closer to the optimal solution area reducing the time consumed by an algorithm to reach the reasonable level of solution.
The four subfigures 2(a), 2(b), 2(c) and 2(d) show that the single exchange mutation (two genes exchanges), represented by dashed curves, approaches the optimal solution faster than the all genes changed mutation, represented by straight curves, in terms of the number of generations. Furthermore, subfigures 2(c) and 2(d) highlight the importance of using heuristic algorithm random-MCT (demonstrated in subsection 5.2) at the initialization stage of GA.

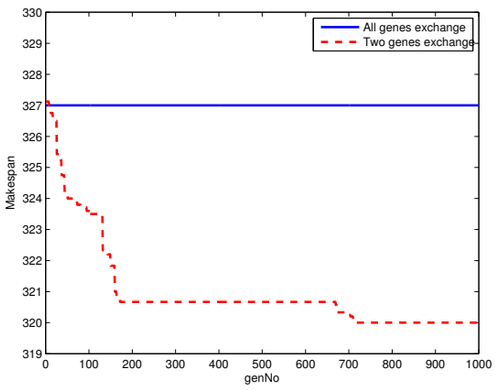
In this chapter, two models are introduced in term of random initialization of tasks and resources. First, Random Model (RM), which follows the uniform distribution in generating the matrix EET_{ij} with low-value ranges for computing capacities of resources and the workload of tasks. Second model, Expected Time to Compute (ETC) 11, in which the workload of tasks and computing resource capacity are generated randomly, in different ranges, low and



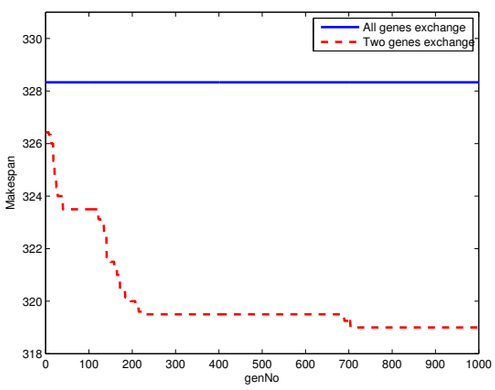
(a) Makespan results with one point crossover and random initialization of GA



(b) Makespan results with two points crossover and random initialization of GA



(c) Makespan results with Random-MCT and one point crossover



(d) Makespan results with Random-MCT and two points crossover

Fig. 2. Makespan results of experiment 8 (mentioned in table 3) which consists of 1000 tasks and 50 resources with one/two point(s) crossover, with/without Random-MCT, and two/all genes exchanged.

high. Figure 3 shows the relationships among RM and ETC models, on one hand, and both algorithms RGSGCS and MSA, on the another hand.

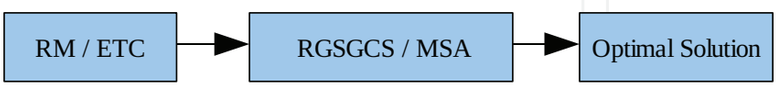


Fig. 3. The relationship among RM/ETC model and the RGSGCS/MSA

4. Problem formulation

For any problem formulation is fundamental issue which help to understand the problem at hand. This chapter considers a grid with sufficient arriving tasks to GA for scheduling. Let N be the total number of tasks to be scheduled and W_i, where $i = 1, 2, \cdots, N$, be the workload of

each task in number of cycles. The workload of tasks can be obtained by analyzing historical data, such as determining the data size of a waiting task. Let M be the total number of computing resources and CP_j, where $j = 1, 2, \cdots, M$, be the computing capacity of each resource expressed in number of cycles per unit time. The Expected Execution Time EET_{ij} of task T_i on resource R_j is defined in the following formula:

$$EET_{ij} = \frac{W_i}{CP_j} \tag{1}$$

5. Rank Genetic Scheduler for Grid Computing Systems (RGSGCS) algorithm

GA is a robust search technique that allows a high-quality solution to be derived from a large search space in polynomial time, by applying the principle of evolution. In other words, GA is used to solve optimization problems by imitating the genetic process of biological organisms (Goldberg, 1989). In GA, a potential solution to a specific problem is represented as a chromosome containing a series of genes. A set of chromosomes make up population. GA evolves the population, that generates an optimal solution, using selection, crossover and mutation operators.

Therefore, GA combines the exploitation of best solutions from past searches with the exploration of new regions of the solution space. Any solution in the search space of the problem is represented by a chromosome.

RGSGCS is GA for solving task scheduling in grid environment. It is presented in the algorithm 2 and the flowchart 5, ((Wael et al., 2010) and (Wael et al., 2011)).

In order to successfully apply RGSGCS to solve the problem at hand, one needs to determine the following :

1. The representation of possible solutions, or the chromosomal encoding.
2. The fitness function which accurately represents the value of the solution.
3. Genetic operators (i.e., selection, crossover, Mutation and Elitism) which have to be used and the parameter values (population size, probability of applying operators, maximum number of generatons, etc.), which are suitable.

The main steps in RGSGCS are as follows:

5.1 Chromosome representation of RGSGCS algorithm

In GA, a chromosome is represented by a series of genes. Each gene, in turn, represents an index of computing resource R_j as shown below:

$$Chromosome = gene_i(R_j) \tag{2}$$

Where $i = 1, 2, \cdots, N$, and $j = 1, 2, \cdots, M$. Figure 4 shows an example of the chromosome's representation consists of three resources and thirteen tasks.

Task No.	1	2	3	4	5	6	7	8	9	10	11	12	13
Resource No.	1	3	1	3	2	2	3	1	2	1	3	2	2

Fig. 4. Task-Resource Representation for the Grid Task Scheduling Problem

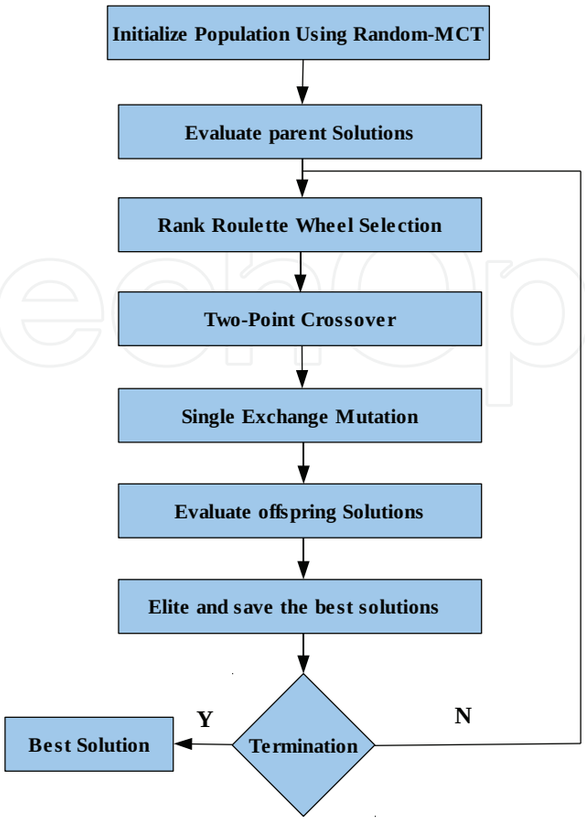


Fig. 5. Flow Chart of RGSGCS Algorithm

5.2 Population initialization of RGSGCS algorithm

One of the important steps in GA is initialization of population. This initialization supports GA to find best solutions within the available search space. In this step, in GA, if bad solutions are generated randomly, the algorithm provides bad solutions or local optimal solutions. To overcome the posed problem, generating individuals using well-known heuristics in the initial step of the algorithm is required. These heuristics generate near-optimal solutions and the meta-heuristic algorithm combines these solutions in order to obtain better final solutions.

Scheduling heuristics such as Min-Min, Minimum Completion Time (MCT), Minimum Execution Time (MET) (Braun et al., 2001), are proposed for independent tasks. Most of these heuristics are based on the following two assumptions.

First, the expected execution time EET_{ij} is deterministic and does not vary with time.

Second, each task has exclusive use of the resource. Traditional MCT heuristic assigns each task to the resource that completes it earliest. The new algorithm, Random-MCT, is described below: For the first tasks in the grid, which their number equals to total resources number in the grid, the resources are assigned randomly. The remaining tasks in the grid are assigned according to earliest finishing time. Where RT_j is the Ready Time of resource j. The time complexity of Random-MCT heuristic is O($M.N$). After completion of RGSGCS's initialization, the evaluation phase is introduced.

Algorithm 1 Random-MCT

for $Ti = 1$ *to* M tasks in the grid **do**
  for all resources R_j in the grid **do**
    $C_{ij} = EET_{ij} + RT_j$
    find resource R_p randomly for T_i
    attach T_i to R_p
  end for
end for
for remaining tasks T_i in the grid **do**
  for all resources R_j in the grid **do**
    $C_{ij} = EET_{ij} + RT_j$
    find resource R_p which will finish T_i earliest
    attach T_i to R_p
  end for
end for

Algorithm 2 RGSGCS

Generate Initial Population P of size $N1$ using Random-MCT (algorithm 1).
for $g = 1$ to $MaximumGenerations$ **do**
  Calculate the fitness of each chromosome using equations (3, 4 and 5)
  Generate offspring Population from P
  {Rank based Roulette Wheel Selection
  Recombination and Mutation
  Calculate the fitness of each chromosome using equations (3, 4 and 5) }
  (elitism) Select the members of the combined population based on minimum fitness, to make the population P of the next generation.
end for

5.3 The evaluation phase of RGSGCS algorithm

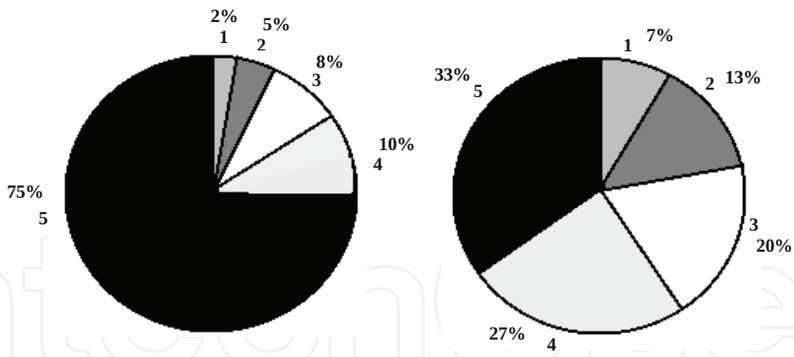
The evaluation phase evaluates the quality of resulted schedule depending on a fitness equation. The fitness equation must be devised to determine the quality of a given chromosome instance and always returns a single numerical value. In this chapter, the fitness function is the makespan, i.e., the minimum completion time of the last finishing task. In other words, makspan is the schedule length. The main goal is to maximize the throughput of the grid by minimizing makespan through an intelligent load balancing. The makespan is calculated using the equations 3 and 4.While the fitness function is expressed as in equation 5.

$$C_m = \sum_n EET_{n,m} \tag{3}$$

$$makespan = Max\{C_m\} \tag{4}$$

$$fitness = makespan \tag{5}$$

Where $m = 1, 2, \cdots, M$; $n = 1, 2, \cdots, N$; M is the total number of resources; and N is the total number of tasks. C_m is the sum of EET of each task T_n assigned to resource R_m, which denotes the completion time of the last task on resource. After the completion of the evaluation Phase, selection phase is used.



(a) Roulette Wheel Selection (b) Rank Selection

Fig. 6. Example of Roulette Wheel Selection and Rank Selection

5.4 Rank Roulette Wheel Selection (RRWS) of RGSGCS algorithm

This phase chooses chromosomes from a population for later breeding (crossover). RRWS combines the advantages of two types of selections, namely, Rank Selection (RS) and Roulette Wheel Selection (RWS).

5.4.1 Roulette Wheel Selection (RWS) of GA

GA uses proportional selection. The population of the next generation is determined by n independent random experiments; the probability that chromosome xi is selected from the pool $(x1, x2, \cdots, xm)$ to be a member of the next generation at each experiment is given by the following equation:

$$RP(c) = \frac{Fitness(c)}{\sum_n^N Fitness}; \tag{6}$$

This process is also called roulette wheel selection. Where each chromosome of the population is represented by a slice that is directly proportional to the chromosome's fitness. A selection step is a spin of the wheel, which in the long run tends to eliminate the least fit population chromosomes. Figure 6 (a) explains the selection method of (RWS) (Obitko, 1998). A roulette wheel where are placed all chromosomes in the population, every chromosome has its place big accordingly to its fitness function. The procedure now is to assign to each chromosome a part of roulette wheel then spin the wheel n time to select n individuals.

5.4.2 Rank Selection (RS) of GA

RWS has problem when the fitness value differs very much. In RS (as shown in figure 6 (b)) the worst value has fitness value equals to 1, the second worst value equals to 2, $\cdots$, etc, and the best will have fitness value equals to N (Obitko, 1998).

Therefore, the RGSGCS's selection process is accomplished by applying RRWS (Wael et al., 2009c; Jadaan et al., 2009), RRWS orders the chromosome's fitness values of population in ascending order, and save this order in array, say *Rank*. After that, it associates the probability shown in equation(7) with each individual chromosome, calculates cumulative proportion of each chromosome, and selects solutions from the population by repeated random sampling

based on cumulative proportion.

$$RP(c) = \frac{Rank(c)}{\sum_{n}^{N} Rank}; \tag{7}$$

RRWS determines how many and which individuals will be kept in the next generation. Next, crossover operator and mutation operator are explained.

5.5 Two-point crossover operator of RGSGCS algorithm

Two-point crossover operator (figure 7) controls how to exchange genes between individuals. Two chromosomes are selected randomly from mating pool. Where the middle part in each chromosome is reversed between two parent chromosomes. It is applied to the chromosomes from selection phase. After that, the mutation operator allows for random gene alteration of

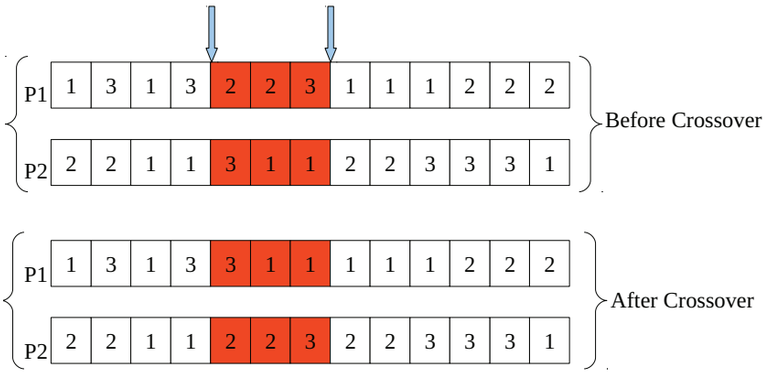


Fig. 7. Two-Point Crossover Operator

an individual.

5.6 Single exchange mutation operator of RGSGCS algorithm

In this phase, single exchange mutation operator (figure 8) is applied to the output of crossover phase. Mutation operator exchanges only two genes according to a mutation rate P_m. It is useful to avoid premature convergence of GA.

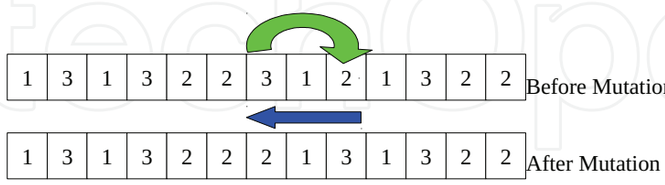


Fig. 8. One Exchange Mutation Operator

5.7 Elitism of RGSGCS algorithm

Besides the standard genetic operators (i.e., crossover and mutation operators). The elitism Phase is used finally to preserve the best candidates for the next generation, so that the algorithm always converges to the global optimum. It combines the parent population with the modified population (the candidates generated by crossover and Mutation operators), and

takes the best chromosomes. After this phase, the algorithm continues to the next iteration. The algorithm terminates after it reaches the maximum generations number.

6. Time complexity analysis of RGSGCS algorithm

Time complexity analysis of RGSGCS algorithm can be analyzed step by step as shown in table 1. From table 1 time complexity of RGSGCS algorithm is expressed as: O($Q.PS.N.M$). Where PS is population size, and Q is Maximum number of iterations of RGSGCS algorithm.

Phase	Complexity
Initialization	O($PS.M.N$)
Selection	O($PS^2 + PS.log(PS) + 2PS$)
Crossover	O($PS.N$)
Mutation	O(PS)
Evaluation	O(2.$PS.N.M$)
Elitisim	O($PS + PS.log(PS)$)
RGSGCS algorithm	O($Q.PS.N.M$)

Table 1. Time complexity analysis of RGSGCS algorithm

7. Mutation based simulated annealing algorithm for minimizing makespan in grid computing systems (MSA)

SA is a global optimization technique. It derived from the concept of metallurgy which crystallizes the liquid to the required temperature. Traditional SA, as shown in figure 9 (Akella, 2009), explores globally, spending more time in regions which on average have fitter solutions. In the later stages, the search is confined to a small area, and SA optimizes within that area. In these final stages, it is similar to local search.

Instead of population which is used in GA, two solutions are maintained in MSA algorithm at a time.

Interestingly, MSA works iteratively keeping two tentative chromosomes at every iteration. For the first iteration, single exchange mutation operator is applied to initial chromosome, with different genes to produce two new solutions. For the remaining iterations, First chromosome is generated from the previous one by applying single exchange mutation operator to it, while the second chromosome is generated by applying single exchange mutation operator to initial chromosome (generated by Random-MCT algorithm 1), and each one either replaces it or not depending on acceptance criterion. The acceptance criterion works as follows: the old and the new solutions have an associated makespan *Makespan*, determined by a fitness function. If the new solution is better than the old one, then it will replace it, if it is worse, it replaces it with probability P. This probability depends on the difference between their *Makespan* values and control parameter T named temperature. This acceptance criterion provides a way of escaping from local minimum. Therefore, the probability of moving to the new solution decreases exponentially as its fitness gets worse, and also temperature gets lower. Usually temperature is gradually decreased so that uphill movements (for minimization problem) become less and less as the run progresses.

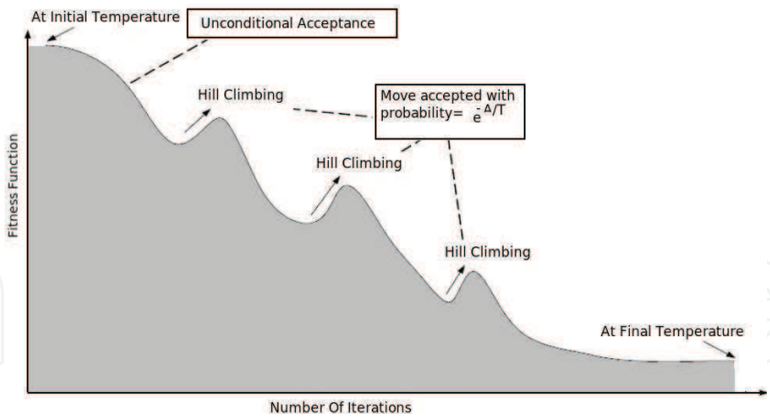


Fig. 9. Convergence of Simulated Annealing.

In single exchange mutation operator, it picks up two genes of a chromosome selected randomly, then exchanges resource indices between them if these indices are not the same vaue (Wael et al., 2011). The MSA algorithm is described by algorithm 3. In the traditional SA,

Algorithm 3 MSA

Choose an initial chromosome $s \in S$ using Random-MCT
$i = 0$, $s^* = s$, $T_{initial} = 1000$, $best = Makespan(s)$
$t_i = T_{initial}$
repeat
    Generate chromosome $s1 \in Neighbor(s)$ by using Single Exchange mutation.
    Apply using Single Exchange mutation on initial chromosome s to generate chromosome $s2$.
    Calculate $Makespan(s1)$ and $Makespan(s2)$.
    for $l = 1$ to 2 **do**
        Δ=exp($\frac{Makespan(s_l) - Makespan(s)}{t_i}$)
        if random[0,1] $< \Delta$ **then**
            $s = s_l$
        else
            if Makespan(s_l) $>$ best **then**
                s^*=s_l
                $s = s^*$
                $best = Makespan(s^*)$
            end if
        end if
    end for
    t_{i+1}=$t_i \times 0.99$;
until *Neighbours* is reached
return s^*, and best $Makespan(s^*)$ // Where *Makespan* is makespan value of candidate chromosome.

only one neighborhood solution is created in each temperature. The main difference between MSA and traditional SA is that MSA creates two neighborhood solutions in each temperature using single exchange mutation, and selects one of them according to the probability and the fitness function. Applying this modification to the traditional SA causing MSA algorithm to

find better solutions in less average time. Note that in table 2 stopping criterion is refered as the total number of *Neighbours*. As *Neighbours* increases, more solutions will be evaluated and larger areas of the search space may be searched. This enhances MSA chances to find the global optimum.

In order to make additional comparison with other algorithms, Min-Min algorithm is chosen as a benchmark because most related works evaluated this algorithm (Wael et al., 2011). The idea of algorithm Min-Min is as follows: calculate the shortest execution time of every task, select the shortest task to match a resource, then delete the task. Repeat the steps until all tasks finished. This algorithm takes $O(N^2.M)$ time. While computational time complexity of MSA algorithm is expressed as O($M.N$+$Q.M.N$) or O($Q.M.N$), where Q is total number of MSA iterations.

8. Performance analysis of RGSGCS and MSA

In order to measure the final schedule of both algorithms MSA, and RGSGCS, the following parameters are used:

First, Load Balancing Factor LBF, which is in the following equation:

$$LBF = \frac{Makespan}{mean(C_m)} - 1 \tag{8}$$

Note that LBF measures load balancing of an algorithm's solution, when LBF minimizes, algorithm's quality is better. Finally, when LBF equals to zero the load balancing of algorithm's solution is optimal. MSA algorithm needs to spend very less time to come up with an optimal solution. Second, average resource utilization is given by the following equation:

$$U = \frac{mean(C_m)}{Makespan} \tag{9}$$

Where $m = 1, 2, \cdots, M$. However, according to the simulation results, it is proved that MSA is effective in speeding up convergence while providing an optimal result.

9. Simulation results Of RGSGCS and MSA

The simulation results of algorithms RGSGCS, MSA and Min-Min of RM are shown in table 3 for eight different experiments. The reason of testing experimenting is to explore the dynamic behavior of grid environment. For the experiment 1, workloads of tasks are $(6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52, 60)$ cycles, and the computing capacities of resources are $(4, 3, 2)$ Cycle Per Unit Time(CPUT). The computing capacities of resources, experiments 2 to 8 in the table 3, diverse randomly from 2 to 8 CPUT for each resource. Furthrmore, the workload of tasks ranges randomly from 10 to 150 cycles. The parameters of MSA and RGSGCS are listed in table 2. MSA, Min-Min and RGSGCS are simulated using MATLAB. In table 3, the values of makespan, average time and LBF for both algorithms RGSGCS and MSA are compiled together for purpose of comparison. Table 3 and figures(10 (a), (b) and (c)) lay out that:

The average gains in time of MSA expressed as percentages are 97.66%, 91.03%, 96.83%, 89.6%, 93.35%, 90.95%, 92% and 92.29% for experiments 1 upto 8 respectively. Hence the total average gain in time for all the experiments is 92.964%.

RGSGCS algorithm	Parameters
Crossover Rate	0.8
Mutation Rate	0.1
Population Size for experiment 1	20
Population Size for experiment 2	80
Population Size for experiment 3	150
Population Size for experiment 4	250
Maximum Generations for experiments from 1 up to 3	1000
Population Size for experiments from 5 up to 8	TasksNo.
Maximum Generations for experiments from 4 up to 8	1500
MSA algorithm	
Stopping Criterion for experiments 1 and 2	$M * N$*100
Stopping Criterion for experiment 3	$M * N$*20
Stopping Criterion for experiment 4	$M * N$*10
Stopping Criterion for experiments 5 up to 8	$M * N$*5
Initial Temperature	1000
Cooling rate	0.99

Table 2. Parameters used in RGSGCS/MSA

MSA algorithm has reduction in makespan value equals to eighteen (18) when it is compared with algorithm Min-Min and equals to three (3) when it is compared with algorithm RGSGCS. Moreover, LBF values of algorithms MSA, Min-Min and RGSGCS are in ranges [0–1.53], [6.4–29.56] and [0–8.12] respectively.

From results discussed above, it can be concluded that MSA algorithm dynamically optimizes output schedule closer to global optimal solution.

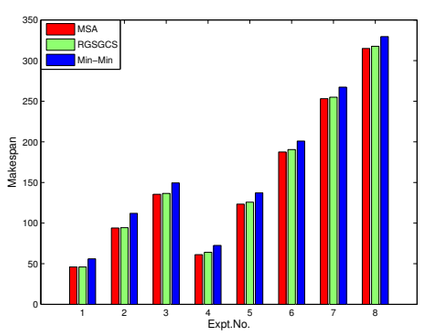
Note that the MSA algorithm outperforms RGSGCS algorithm within very less time to run the algorithm. Depending on SA algorithm and random-MCT heuristic, MSA algorithm is powerful when it is compared with RGSGCS algorithm, while RGSGCS algorithm has less convergence to the optimal solution.

The results of the comparison among algorithms RGSGCS , Min-Min and MSA in each experiment in the table 3, prove that MSA algorithm provides an effective way to enhance the search performance, because it obtains an optimal schedule within a short time along with high resource utilization.

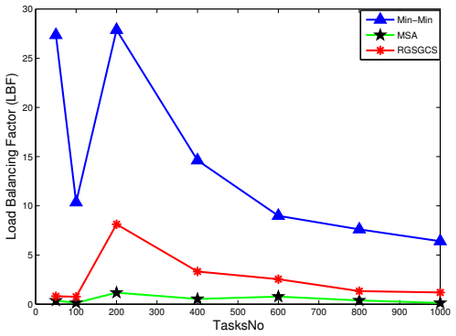
Notably, the solutions of MSA algorithm are high quality and can be used for realistic scheduling in grid environment. The simulation results are consistent with the performance analysis in section 8, which clarifies that the improvement to the evolutionary process is reasonable and effective.

10. Improvment in time consumed by algorithm of RGSGCS and MSA

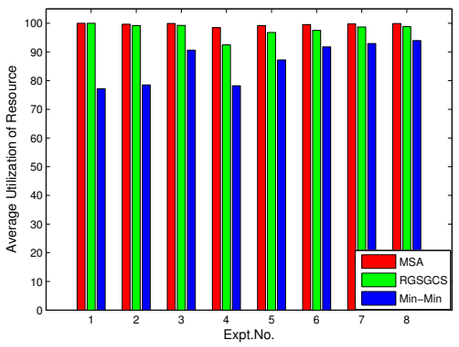
According to the table 3, two genes exchanged in single exchange mutation has shown good performance in both algorithms RGSGCS and MSA. Moreover, the times taken by both algorithms RGSGCS and MSA are still big, and it is useful to get reasonable results along with less time consumed by algorithms RGSGCS and MSA. One way to do this is to assign



(a) Makespan values of MSA, Min-Min and RGSGCS



(b) LBF values of MSA, Min-Min and RGSGCS



(c) Utilization of Resource of MSA, Min-Min and RGSGCS

Fig. 10. Simulation Results of Random model

probability of crossover, probability of mutation, ppopulation size and maximum generations number to the values 1 , 1, 50 and 1000, respectively in the experiments in table 3. Table 4 dispays new values of solution for both algorithms MSA and RGSGCS. This table provides the results of MSA algorithm for two different termination criterions, namely $T < e^{-300}$ for MSA(1) and $T < e^{-50}$ for MSA(2). Note that, in tables 3, 4, MR denotes reduction in makespan, which is difference between makespan values for both algorithms RGSGCS and MSA. Two experiments 9 and 10 are added to ensure scalability of both algorithms MSA and RGSGCS.

11. ETC model

The Expected Time to Compute (ETC) model is another model can also test performance of MSA algorithm. Interestingly, ETC matrix model allows to capture important characteristics of task scheduling. For example, ETC model introduces possible inconsistencies among tasks and resources in grid system by assigning a large value to $ETC(t, m)$ to indicate that task t is incompatible with resource m.

Moreover, ETC matrix considers three factors: task heterogeneity, resource heterogeneity and consistency. The task heterogeneity depends upon the various execution times of the

Algo.	*Makespan*	Time	*LBF*	*MR*	*Utilization*
	experiment 1 (13 tasks,3 resources)				
MSA	46	0.43	0		100
Min-Min	56	0.012	29.56	10	77.18
RGSGCS	46	18.38	0	0	100
	experiment 2 (50 tasks,10 resources)				
MSA	94	5.78	0.317		99.68
Min-Min	112	0.072	27.35	18	78.5
RGSGCS	94.33	64.44	0.81	0.33	99.196
	experiment 3 (100 tasks, 10 resources)				
MSA	135.5	4.5	0.063		99.94
Min-Min	149.5	0.077	10.35	14	90.62
RGSGCS	136.5	142.1	0.75	1	99.25
	experiment 4 (200 tasks, 50 resources)				
MSA	61.14	41.12	1.53		98.49
Min-Min	72.5	0.36	27.87	11.36	78.2
RGSGCS	64	395	8.12	2.86	92.49
	experiment 5 (400 tasks, 50 resources)				
MSA	123.33	67.86	0.823		99.18
Min-Min	137.2	2.92	14.62	13.87	87.24
RGSGCS	125.75	1021	3.32	2.42	96.78
	experiment 6 (600 tasks, 50 resources)				
MSA	187.5	142.58	0.452		99.55
Min-Min	201	8.92	8.98	13.5	91.76
RGSGCS	190.5	1575	2.54	3	97.52
	experiment 7 (800 tasks, 50 resources)				
MSA	253.167	248.7	0.235		99.76
Min-Min	267.37	21.66	7.6	14.203	92.94
RGSGCS	255	2826	1.33	1.833	98.69
	experiment 8 (1000 tasks, 50 resources)				
MSA	315	380.14	0.108		99.89
Min-Min	329.5	41.46	6.4	14.5	93.98
RGSGCS	317.6	4928	1.19	2.6	98.82

Table 3. Simulation Results of MSA, Min-Min and RGSGCS with probability of crossover, probability of mutation, population size and maximum generations number values taken from table 2.

tasks. The two possible values are defined high and low. Similarly the resource heterogeneity depends on the running time of a particular task across all the resources and again has two values: high and low.

In the real scheduling, three different ETC consistencies are possible. They are consistent, inconsistent and semi-consistent. The instances of benchmark problems are classified into twelve (12) different types of ETC matrices, they are generated from model of Braun (Braun et al., 2001). Each type is obtained by calculating the average value of makespan of ten runs of each algorithm except Min-Min algorithm which it runs just once by default. The

Algo.	*Makespan*	Time	*LBF*	*MR*	*Utilization*
	experiment 1 (13 tasks,3 resources)				
MSA(1)/MSA(2)	46/46.5	5.49/0.964	0/1.45		100/98.57
RGSGCS	46	5.84	0	0/0.5	100
	experiment 2 (50 tasks,10 resources)				
MSA(1)/MSA(2)	94/94	7.277/1.31	0.34/0.32		99.63/99.68
RGSGCS	94	10.28	0.33	0/0	99.67
	experiment 3 (100 tasks, 10 resources)				
MSA(1)/MSA(2)	135.5/135.5	9.33/1.64	0.05/04		99.95/99.96
RGSGCS	136	16.39	0.38	0.5/0.5	99.63
	experiment 4 (200 tasks, 50 resources)				
MSA(1)/MSA(2)	61.5/64	31.66/5.587	2.66/6.37		97.79/94.01
RGSGCS	65	42.46	7.7	3.5/1	92.285
	experiment 5 (400 tasks, 50 resources)				
MSA(1)/MSA(2)	123.33/126	57.92/10.167	0.97/2.76		99.04/97.31
RGSGCS	127.75	76.01	2.42	4.42/1.75	95.46
	experiment 6 (600 tasks, 50 resources)				
MSA(1)/MSA(2)	188/192.5	84.02/14.78	0.66/2.86		99.34/97.22
RGSGCS	190.62	107.44	3.12	2.62/-1.88	97.89
	experiment 7 (800 tasks, 50 resources)				
MSA(1)/MSA(2)	255/261	109.9/19.23	0.69/2.86		99.32/97.22
RGSGCS	259	139.41	5.83	4/-2	97.52
	experiment 8 (1000 tasks, 50 resources)				
MSA(1)/MSA(2)	316/319	136.23/23.876	0.4/1.3		99.6/98.69
RGSGCS	318	170.81	1.13	2/-1	98.88
	experiment 9 (2000 tasks, 50 resources)				
MSA(1)/MSA(2)	629/635	268.23/46.94	0.37/1.1		99.63/98.9
RGSGCS	630	329.36	0.63	1/-5	99.38
	experiment 10 (3000 tasks, 50 resources)				
MSA(1)/MSA(2)	948/953	402.8/70.62	0.47/0.84		99.53/99.17
RGSGCS	947.33	488.68	0.48	-0.67/-5.67	99.52

Table 4. Simulation Results of MSA and RGSGCS with probability of crossover, probability of mutation, population size and maximum generations number equal to 1 , 1, 50 and 1000, respectively.

instances depend upon the above three factors as task heterogeneity, resource heterogeneity and consistency. Instances are labeled as u-x-yyzz where:

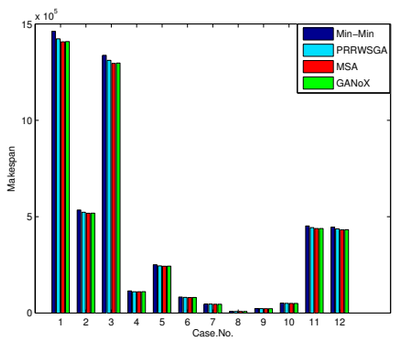
1. u - is a uniform distribution, used to generate the matrix.
2. x - is a type of consistency.
   (a) c - consistent. An ETC matrix is said to be consistent if a resource *Ri* execute a task Ti faster than the resource *Rk* and *Ri* executes all other tasks faster than *Rk*.
   (b) s - semi consistent. A semiconsistent ETC matrix is an inconsistent matrix which has a sub matrix of a predefined size.
   (c) i - inconsistent. An ETC matrix is said to be inconsistent if a resource Ri executes some tasks faster than Rj and some slower.

3. yy - is used to indicate the heterogeneity of the tasks(hi-high, lo-low).
4. zz - is used to indicate the heterogeneity of the resources (hi-high, lo-low).

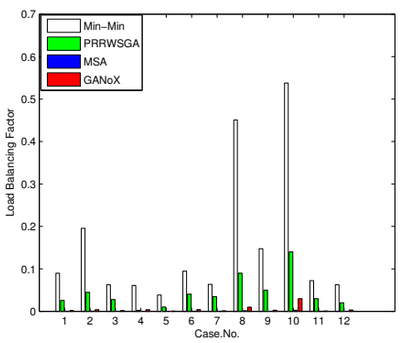
All the instances consist of 512 tasks and 16 resources. This Model is studied for the following algorithms:

1. GANoX algorithm is the same algorithm 2, but without using crossover operator. Single exchange mutation is used at probability of mutation equals to one.
2. PRRWSGA algorithm is the same algorithm 2, probability of crossover equals to 0.8, probability of Mutation equals to 0.01, and full chromosome will be altered.
3. MSA-ETC is the same algorithm 3. Initial chromosome can be taken as the best run of 1000 runs of the algorithm 1. Moreover, stopping criterion which is used equals to $(M \times N \times 20)$;
4. Min-Min algorithm is pointed out in section 7.

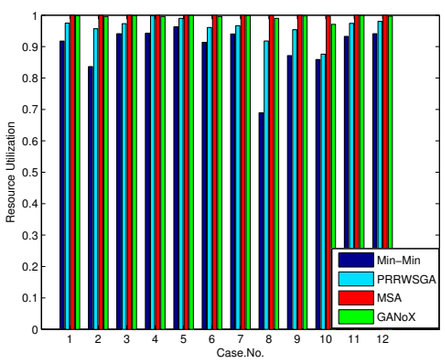
Maximum Generation is 1000 and Population Size is 50 for both algorithms GANoX and PRRWSGA. It can be seen from figures 11 (a), (b) and (c), and table 5, that MSA-ETC has superior performance on all remaining algorithms, namely, Min-Min, GANoX, and PRRWSGA, in terms of *LBF*, *Makespan*, *ResourceUtilization*, and time taken by the algorithm. Saving in average time is about 90%, except when it is compared with Min-Min.



(a) Makespan values of ETC model (sec.)



(b) LBF values of ETC model



(c) Resource Utilization values of ETC model

Fig. 11. Simulation Results of ETC model

Furthermore, the resource utilization value in range [0.9975-0.9999], and LBF value in range [0.0025-0.000085]. From the analysis of time complexity of RGSGCS algorithm in the table 1,

ETC matrix / Algo.	Min-Min	MSA-ETC	GANoX	PRRWSGA
$u-c-hihi$	1462108.59	1407383.05	1408816.72	1423161.96
$u-i-hihi$	534521.65	517909.73	518207.61	522950.76
$u-s-hihi$	1337720.25	1295674.81	1297192.19	1311406.26
$u-c-hilo$	114600.52	110549.32	110774.58	110661.56
$u-i-hilo$	250758.61	243168.05	243295.24	244844.35
$u-s-hilo$	83094.47	80568.32	80680.07	81423.61
$u-c-lolo$	47104.35	45800.13	45822.55	46398.43
$u-i-lolo$	8659.29	8422.85	8430.91	8483.34
$u-s-lolo$	23337.57	22546.96	22561.49	22841.92
$u-c-lohi$	51556.38	49786.42	49907.59	50254.31
$u-i-lohi$	452016.81	438583.36	438728.74	443424.34
$u-s-lohi$	445906.75	431898.36	432666.09	437031.44
LBF	0.5379-0.0385	0.0025-0.000085	0.03-0.0008	0.14-0.002
$ResourceUtilization$	0.8365-0.963	0.9975-0.9999	0.971-0.999	0.876-0.99

Table 5. Simulation Results of ETC Model of Min-Min, MSA-ETC, GANoX, and PRRWSGA

the time complexity of GANoX algorithm and PRRWSGA algorithm is O($Q.PS.M.N$), and for MSA-ETC algorithm is O($Q.M.N$).

As a result, MSA-ETC has less time complexity when it is compared with both algorithms GANoX and PRRWSGA. In the study, the algorithm is designed and compared to different grid environments. Using MSA-ETC it can get good workload balancing results.

The proposed MSA-ETC algorithm can consistently find better schedules for several benchmark problems as compared to other techniques in the literature.

12. Conclusion

This chapter studies problem of minimizing makespan in grid environment. The MSA algorithm introduces a high throughput computing scheduling algorithm. Moreover, it provides solutions for allocation of independent tasks to grid computing resources, and speeds up convergence. As a result load balancing for MSA algorithm is higher than RGSGCS algorithm, and the gain of MSA algorithm in average time consumed by an algorithm is higher than RGSGCS algorithm for both RM and ETC models, which makes MSA algorithm very high QoS and more preferable for realistic scheduling in grid environment.

The initialization of MSA algorithm plays important role to find a good solution and to reduce the time consumed by algorithm.

Furthermore, the improvments on the performance of MSA algorithm, and RGSGCS, give another salient feature, which reduces the time consumed by algorithm to the low reasonable level.

Regarding MSA algorithm for ETC Model, MSA algorithm has superior performance among other algorithms along with resource utilization and load balancing factor values.

Other benefits of MSA algorithm include robustness and scalability features. the disadvantage of MSA algorithm is that flowtime is higher more than Min-Min, RGSGCS, and GANoX.

It can be concluded that MSA algorithm is a powerful technique to solve problem of minimizing makespan in grid environment with less time to be consumed by the intended algorithm.

13. References

A. A. P. Kazem, A. M. Rahmani, & H. H. Aghdam. A modified simulated annealing algorithm for static task scheduling in grid computing. In: *The International Conference on Computer Science and Information Technology*, September 2008, pp. (623-627), IEEE, ISBN 978-0-7695-3308-7, Singapore.

Abdulal, W., Jadaan, O. A, Ahmad Jabas, A. & S. Ramachandram. Genetic algorithm for grid scheduling using best rank power. In: *IEEE Nature & Biologically Inspired Computing (NaBIC 2009)*, December 2009, pp. (181-186), IEEE, ISBN 978-1-4244-5053-4, Coimbatore, India.

Abdulal, W., Jabas, A., Jadaan, O. A. & S. Ramachandram. An improved rank-based genetic algorithm with limited iterations for grid scheduling. In: *IEEE Symposium on Industrial Electronics and Applications (ISIEA2009)*, October 2009, pp. (215-220), IEEE, ISBN 978-1-4244-4681-0. Kuala Lumpur, Malaysia.

Abdulal, W., Jadaan, A. O., Jabas, A., Ramachandram, S., Kaiiali M. & C.R. Rao. Rank-based genetic algorithm with limited iterations for grid scheduling. In: *The First IEEE International Conference on Computational Intelligence, Communication Systems, and Networks (CICSyN2009)*, July 2009. pp. (29-34), ISBN 978-0-7695-3743-6, Indore, India.

Abdulal, W. & S. Ramachandram. Reliability-Aware Genetic Scheduling Algorithm in Grid Environment. In: *IEEE International Conference on Communication Systems and Network Technologies*, June 2011, pp. (673-677), IEEE, ISBN 978-0-7695-4437-3/11, Katra, Jammu, India.

Abdulal, W., Jadaan, O. A., Jabas, A.& S. Ramachandram. Mutation Based Simulated Annealing Algorithm for Minimizing Makespan in Grid Computing Systems. In: *IEEE International Conference on Network and Computer Science (ICNCS 2011)*. April 2011, Vol. 6. pp. (90-94), IEEE, ISBN 978-1-4244-8679-3, Kanyakumari, India.

Abdulal, W., Jadaan, o. A., Jabas, A. & S. Ramachandram. Rank based genetic scheduler for grid computing systems. In: *IEEE International Conference on Computational Intelligence and Communication Networks (CICN2010)*, Nov. 2010, pp. (644-649), IEEE, ISBN 978-1-4244-8653-3, Bhopal, India.

Abraham, A., Rajkumar Buyya & Baikunth Nath. Nature's heuristics for scheduling jobs on computational grids. In: *8th IEEE International Conference on Advanced Computing and Communications (ADCOM2000)*. `www.buyya.com/papers/nhsjcg.pdf`, pp. (45-52).

Akella, P. `http://www.ecs.umass.edu/ece/labs/vlsicad/ece665/slides/SimulatedAnnealing.ppt`.

Braun, Tracy D. and Siegel, Howard Jay and Beck, Noah and Bölöni, Lasislau L. and Maheswaran, Muthucumara and Reuther, Albert I. and Robertson, James P. and Theys, Mitchell D. and Yao, Bin and Hensgen, Debra and Freund, Richard F. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, In: *Journal of Parallel Distributing Computing*. June 2001, pp. (810-837), ISSN 0743-7315.

C. Wook Han & J. Il Park. SA-selection-based genetic algorithm for the design of fuzzy controller. In: *International Journal of Control, Automation, and Systems*, 2005, Vol. 3, pp. (236-243).

Carretero, J. & Xhafa, F. Using Genetic Algorithms for Scheduling Jobs in Large Scale Grid Applications. In: *Journal of Technological and Economic Development*. http://citeseer.ist.psu.edu/, 2006, Vol. 12, pp. (11-17). ISSN 1392-8619 print/ISSN 1822-3613 Online.

Dimitri Bevc, Sergio, E. Zarantonello, Neena Kaushik, & Iulian Musat. (2005). Grid computing for energy exploration. http://www.ogf.org.

Fatos Xhafa, Enrique Alba, Bernabé Dorronsoro, Bernat Duran, & Abraham, A. Efficient batch job scheduling in grids using cellular memetic algorithms. In: *Studies in Computational Intelligence*. Springer-Verlag, 2008, pp. (273-299), Berlin, Heidelberg

Fidanova, S. Simulated annealing for grid scheduling problem. In: *International Symposium on Modern Computing*, 2006, Vol. 0, pp. (41-45).

Foster, I., Kesselman, C. & Steven Tuecke. The Anatomy of the Grid. (2001). In: *International Journal of Supercomputer Applications*.

Garey, Michael R. and Johnson, David S. Computers and Intractability: A Guide to the Theory of NP-Completeness. In: *W. H. Freeman & Co.*, ISSN 0716710455, New York, NY, USA

Goldberg, D. E. In: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, ISBN 0-201-15767-5, New York.

Jinquan, Z., Lina, Ni. & Changjun, J. A Heuristic Scheduling Strategy for Independent Tasks on Grid. In: *Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, IEEE Computer Society, 2005, pp. (588-593). ISBN 0-7695-2486-9, Washington, DC, USA.

Kesselman, C., & Foster, I. In: *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers.

K. Kousalya and P. Balasubramanie. To Improve Ant Algorithm's Grid Scheduling Using Local Search. In: *International Journal Of Computational Cognition* http://www.yangsky.com/ijcc/pdf/ijcc747.pdf. Vol. 7, No. 4, Dec. 2009. pp. (47–57).

K. Kousalya and P. Balasubramanie. Ant Algorithm for Grid Scheduling Powered by Local Search. In: *International Journal Of Open Problems Compt. Math.*. www.ijopcm.org/files/IJOPCM(vol.1.3.5.D.8).pdf. Vol. 1, No. 3, Dec. 2008, pp. (222-240).

Moallem, A. Using Swarm Intelligence for Distributed Job Scheduling on the Grid. http://library.usask.ca/theses/available/etd-04132009-123250/unrestricted/thesis.pdf. In: *University of Saskatchewan*, March 2009.

Omar Al Jadaan, L. Rajamani, & C. Rao. Parameterless penalty function for solving constrained evolutionary optimization. In: *Hybrid Intelligent Models and Applications*, IEEE, pp. (56-63).

Omar Al Jadaan, Jabas, A., Abdulal, W., Rajamani, L., Zaiton, E., Rao, C.R. & C.R. Rao. Engineering Case Studies Using Parameterless Penalty Non-dominated Ranked Genetic Algorithm. In: *The First IEEE International Conference on Computational Intelligence, Communication Systems, and Networks (CICSyN2009)*, July 2009. pp. (51-56), ISBN 978-0-7695-3743-6, Indore, India.

Omar Al Jadaan, Wael Abdulal, Hameed, M.A, & Ahmad Jabas. & C.R. Rao. Enhancing Data Selection Using Genetic Algorithm. In: *2010 International Conference on Computational Intelligence and Communication Networks (CICN)*, Nov. 2010. pp. (434 - 439), ISBN 978-1-4244-8653-3 , Bhopal, India.

Omar Al Jadaan, Alla Alhaffa, Wael Abdulal & Ahmad Jabas. Rank Based Genetic Algorithm for solving the Banking ATMś Location Problem using convolution. In: *2011 IEEE Symposium on Computers & Informatics (ISCI)*, March 2011. pp. (6–11), ISBN 978-1-61284-689-7, Kuala Lumpur, Malaysia.

Obitko, M. (1998). `http://www.obitko.com/tutorials/genetic-algorithms/selection.php`.

P.K. Suri & Singh Manpreet. An efficient decentralized load balancing algorithm for grid. In: *IEEE 2nd International Advance Computing Conference*. IEEE, pp. (10-13), ISBN 978-1-4244-4790-9 , Patiala, India.

Prado, R. P. and Galán, S. García and Yuste, A. J. and Expósito, J. E. and Santiago, A. J. and Bruque, S .Evolutionary Fuzzy Scheduler for Grid Computing. In: *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence IWANN '09*, Springer-Verlag, pp. (286–293). ISBN 978-3-642-02477-1, Berlin, Heidelberg.

Ruay-Shiung Chang, Jih-Sheng Chang, & Po-Sheng Lin. An ant algorithm for balanced job scheduling in grids. In: *Future Generation Computer Systems*. Vol. 25, pp. (20-27), ISSN 0167-739X.

Suliman, M. O., Vellanki S.S. Kumar, & Abdulal, W. Optimization of Uncertain Construction Time-Cost Trade off Problem Using Simulated Annealing Algorithm. In: *World Congress on information and Communication Technologies*, Dec. 2011.

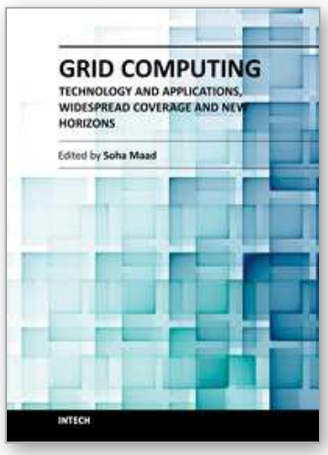
V. D. Martino & M. Mililotti. Scheduling in a grid computing environment using genetic algorithm. In: *The 16th IEEE International Parallel and Distributed Processing Symposium*, April 2002. pp. (235-239). ISBN 0-7695-1573-8.

V. D. Martino. Sub Optimal Scheduling in a Grid Using Genetic Algorithms. In: *Seventeenth International Symposium on Parallel and Distributed Processing*. `ftp://ftp.cs.umanitoba.ca/pub/IPDPS03`, 2003, IEEE Computer Society, ISBN 0-7695-1926-1, Washington, DC, USA.

Y. Gao, H. Rong, & J. Z. Huang. Adaptive grid job scheduling with genetic algorithms. In: *Future Generation Computer Systems*, January 2005, Vol. 21, No. 1, Elsevier Science Publishers, pp. (151-161).

Yin, Hao and Wu, Huilin and Zhou, Jiliu. An Improved Genetic Algorithm with Limited Iteration for Grid Scheduling. In: *Sixth International Conference on Grid and Cooperative Computing*. IEEE Computer Society, pp. (221-227), ISBN 0-7695-2871-6, Washington, DC, USA.

Yun-Han Lee, Seiven Leu, &Ruay-Shiung Chang. Improving job scheduling algorithms in a grid environment. In: *Future Generation Computer Systems*. Vol. 27, pp. (991-998), ISSN 0167-739X.



Grid Computing - Technology and Applications, Widespread Coverage and New Horizons

Edited by Dr. Soha Maad

ISBN 978-953-51-0604-3
Hard cover, 354 pages
Publisher InTech
Published online 16, May, 2012
Published in print edition May, 2012

Grid research, rooted in distributed and high performance computing, started in mid-to-late 1990s. Soon afterwards, national and international research and development authorities realized the importance of the Grid and gave it a primary position on their research and development agenda. The Grid evolved from tackling data and compute-intensive problems, to addressing global-scale scientific projects, connecting businesses across the supply chain, and becoming a World Wide Grid integrated in our daily routine activities. This book tells the story of great potential, continued strength, and widespread international penetration of Grid computing. It overviews latest advances in the field and traces the evolution of selected Grid applications. The book highlights the international widespread coverage and unveils the future potential of the Grid.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen