# Meta-heuristically Seeded Genetic Algorithm for Independent Job Scheduling in Grid Computing

Muhanad Tahrir Younis, Shengxiang Yang, and Benjamin Passow

Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester LE1 9BH, UK
p14017957@my365.dmu.ac.uk, syang@dmu.ac.uk, benpassow@ieee.org

**Abstract.** Grid computing is an infrastructure which connects geographically distributed computers owned by various organizations allowing their resources, such as computational power and storage capabilities, to be shared, selected, and aggregated. Job scheduling problem is one of the most difficult tasks in grid computing systems. To solve this problem efficiently, new methods are required. In this paper, a seeded genetic algorithm is proposed which uses a meta-heuristic algorithm to generate its initial population. To evaluate the performance of the proposed method in terms of minimizing the makespan, the Expected Time to Compute (ETC) simulation model is used to carry out a number of experiments. The results show that the proposed algorithm performs better than other selected techniques.

**Keywords:** meta-heuristic algorithms; seeded Genetic Algorithm; Ant Colony Optimization; job scheduling; grid computing; makespan

## 1 Introduction

Grid Computing has been defined as a type of parallel and distributed infrastructure which allows the geographically distributed autonomous and heterogeneous resources to be shared, selected and aggregated dynamically depending on their availability, capability, performance, cost, and users quality-of-service requirements. This infrastructure offers to its users the same processing capabilities provided by supercomputers by creating a virtual supercomputer from connecting various networked and loosely coupled computers together allowing their resources to be shared among users. Computers, processing elements, software applications, printers, network interfaces, storage space and data are examples of resources. Middleware, computer software which provide basic services for resource management, security, monitoring, and so forth, are used to connect all these resources to a network. Due to the fact that resources are owned by various administrative organizations, local policies are defined to specify what is shared, who is allowed to access what and when, and under what conditions. The Grid architecture is based on the creation of Virtual Organizations (VOs), a set of rules defined by individuals and institutions to control resource sharing [8]. By sharing some or all of its resources, a physical organization can be part of one or

more VOs [10]. Grid Computing has been increasingly used by commercial and non-commercial clients as a utility for solving scientific, complex mathematical, and academic problems, as well as for diverse applications [9].

In grid computing, a key concern is job scheduling. In general, job scheduling can be defined as the mapping of jobs to corresponding appropriate resources in order to process them [13]. To evaluate the job scheduling performance, an objective function should be defined such as maximizing resources utilization, minimizing the makespan, and maximizing load balancing [20]. The schedulers efficiency strongly relays on the algorithm applied to do the scheduling. Different algorithms could be used to do the scheduling which varies from simple heuristic methods to meta-heuristic methods. However, to enhance the overall performance of the grid, the meta-heuristic approaches are more likely preferred [19].

Job scheduling in grid computing is known to be a NP-complete optimization problem, and hence, many meta-heuristic methods, which are capable of searching large search spaces very efficiently and provide optimal or near optimal solutions, have been proposed [2]. One of these methods is Ant Colony Optimization (ACO) which is a search algorithm that mimics the behavior of ants in searching for a path between their nest and a source of food [5]. While ants move back and forth from their colony to a source of food, they leave a substance called pheromone on the ground. This substance can be sensed by other ants and this allows indirect communications among them. Ants which find the shortest path to the source of food will, then, return back to their colony earlier than ants with longer paths, which means that the shortest path has been marched over more than other paths and its pheromone density will be higher. Ants will choose the path with the highest pheromone concentration, i.e. the path with high pheromone levels will attract more ants and will contain even more pheromone. However, if this path remains after the consumption of food, it would seriously obstruct the ants' ability to find food. To cope with this situation, pheromone trails evaporate over time, which is a mechanism to forget old decisions, is used [7]. The ACO algorithm has been used to solve various NP-complete optimization problems such as Travelling Salesman problem, assignment problem, graph coloring, and job-shop scheduling successfully [22]. As a result, the ACO algorithm is a good candidate for job scheduling in Grid computing.

In addition to ACO, there are many other meta-heuristic methods which have been proposed to solve the problem of job scheduling in grid computing. One of these methods is Genetic algorithm (GA), a population-based meta-heuristic search method which is inspired by the evolution of living beings. The traditional GA starts with an initial population, a group of solutions, then seeks to find the approximately best solution by applying selection, crossover and mutation operators. The simplest way to generate the initial population is the random method. However, the quality of the final solution found by GA could be effected by the quality of the initial solution as generating an initial population randomly may cause the situation where the population has more individuals with worst quality and, sometimes, infeasible solutions than best quality solutions which

means more time is required to find an optimal solution, more generations are required to evolve best solution, and the convergence rate is reduced. Therefore, a new method to generate the initial generation is needed and actually, several studies have been suggested in this context [21].

A study presented by [23] developed several versions of GA and studied the different configuration issues of GA to solve the Travelling Salesman Problem (TSP) and claimed that using heuristic methods to seed the initial population can improve significantly the efficiency of GA. An improved GA for the rectilinear Steiner problem by [14] proposed a hybrid seeding population technique. The author has compared the efficiency of the proposed seeding technique with the random technique of generating the initial population of GA and concludes that the seeding technique significantly improves the performance. The authors in [11] have proposed a new method to generate the initial population of GA for the optimization of 2d and 3d truss structures. Their work has examined the effect of seeding the initial population on the performance of the GA in terms of capturing the global optimum and concluded that the proposed seeding method reduces the number of generations needed to find the optimal solution and enhances the convergence capability which consequently enhances the overall performance of the GA.

In this work, a GA for static independent job scheduling problem in grid computing is proposed. The proposed algorithm uses a meta-heuristic algorithm, which is ACO, to seed its initial population. The work focuses on minimizing the makespan and the Expected Time to Compute (ETC) model is used to test the performance. The performance of the proposed GA has been compared with Min-Min heuristic, GA with random initial generation, GA with initial population seeded by Min-Min heuristic, and ACO.

The rest of the paper is organized as follows. Section 2 presents the related work on ACO and GA in solving the problem of job scheduling in grid computing. The simulation model used to test the performance of the proposed method is described in section 3. Section 4 explains the use of ACO for job scheduling in grid computing while section 5 describes the proposed method. Section 6 presents the results of applying the proposed GA in grid computing. Finally, the conclusions and future works are provided in Section 7.

## 2 Related work

An ant colony optimization based scheduler for job scheduling in grid computing was proposed by [16]. Minimization of the total job waiting time was the main goal of the proposed scheduler which consists of four steps. The proposed scheduler used local update and global update rules to update the pheromone value on each resource. In addition, the scheduler has used the Completion Time (CT), which is the time a machine needs to finish executing a job measured as clock time. The authors defined a grid environment in which jobs arrive to the system at different times, the availability of resources is regularly changing, one job could be processed by each processor per unit time and jobs are independent

of each other. In the study, the performance of ACO based scheduler has been compared with First Come First Serve (FCFS), Earliest Due Date (EDD) and Earliest Release Date techniques (ERD). The results showed that ACO has the best average-case of the waiting time.

A more efficient ACO-based grid scheduling algorithm was introduced by [18]. The developed scheduler has modified the original ACO algorithm presented in [5] by changing the basic pheromone updating rule. This modification increased efficiently the algorithm performance in terms of makespan compared to the original ACO.

The authors in [22] have developed a hybrid algorithm to improve the performance of other similar techniques described in [3]. The hybrid algorithm has combined ACO with local search. The results show that the use of a local search with ACO increases the quality of the solution.

ACO is not the only algorithm used to solve the problem of job scheduling in grid computing. The literature shows that there are many other meta-heuristic methods which have been suggested in this field. One of these methods is Genetic Algorithm (GA), a population-based heuristic search method which is inspired by the evolution of living beings. GA starts with an initial population, which is a group of solutions usually generated randomly, and then seeks to find the approximately best solution by applying selection, crossover and mutation operators. GA has been quite used for solving many combinatorial optimization problems. Eleven static heuristic methods have been applied in [3] to solve the problem of job scheduling in heterogeneous environment by minimizing the makespan. The experimental results show that GA outperforms the other ten methods used in the study. The authors used a population of 200 individuals which is generated either randomly or by seeding the population with one individual generated using Min-Min heuristic method [12] and 199 individuals generated randomly.

A study proposed by [4] presented the use of GA for efficient multi-objective job scheduling in grid computing systems. To introduce diversity, two heuristics methods, which are Longest Job to Fastest Resource  Shortest Job to Fastest Resource (LJFR- SJFR) [1] and Minimum Completion Time (MCT) [17], have been used beside the random method to initialize the initial population. The authors considered two encoding schemes, namely the direct and the permutation methods, and implemented several GA operators.

A heuristic method called Min-Max has been proposed in [13] for job scheduling in heterogeneous environments. The performance of the proposed method has been compared with five popular heuristics which are: Min-Min, Max-Min, LJFR-SJFR, sufferage, and WorkQueue. The authors investigated the effect of using these heuristics for initializing simulated annealing (SA) and found that the Min-Min and Min-Max heuristics are more efficient than others.

## 3 Simulation model

In order to simulate several heterogeneous scheduling scenarios in a realistic way and to allow a fair comparison of the presented methods, a well-known

benchmark has been used. In this study, the Expected Time to Compute (ETC) benchmark simulation model is used which has been introduced in [3] to address the problem of static scheduling algorithms for Heterogeneous Computing (HC) such as grid computing. The expected execution time of the jobs on each machine has been assumed to be available in advance in a two dimensional array. This assumption is realistic since it is easy to gather information about the jobs requirements and the computation power of resources from the users, by predications or from historic data [25]. To capture the various characteristics of HC environments, the model defines three different types of metrics: consistency, job heterogeneity, and resource heterogeneity. A matrix is said to be consistent if it contains a resource $R_n$ which is capable of processing a job $J_k$ faster than another resource in the system $R_m$, and $R_n$ processes all other jobs $J_t$ faster than $R_m$. If a resource $R_n$ executes some jobs faster than $R_m$ and some slower, then the matrix is said to be inconsistent. A semi consistent matrix is an inconsistent matrix which has a sub matrix of a consistent matrix. Job heterogeneity models the statistical distribution for jobs execution times with two values high or low. Resource heterogeneity models the distribution when executing the same job in all machines, and also has two values high or low. Therefore, we need 12 distinct combinations of ETC matrix to consider all these various characteristics. Table 1 shows a 15 x 10 subset of the ETC matrix with semi-consistent, low job heterogeneity and low resource heterogeneity. The results provided in this study used ETC matrices of size 512 x 16.

The problem description under the ETC model is defined as follows:

1. A number of jobs, n, that has to be scheduled. These jobs are independent to each other, any job can be processed by any resource, and are non-preemptive, which means that a job must be processed entirely by a single resource.
2. A number of resources, m, to process the submitted set of independent jobs. These resources are heterogeneous.
3. The ETC matrix of size n x m, where ETC[i][j] represents the estimated time for executing the job i in the resource j.

The job scheduling problem in grid computing is known to be multi-objective, therefore, several objective functions can be considered for this problem such as makespan, load balancing, and flowtime [24]. In this study, we consider the minimization of makespan, which is defined as the finishing time of the latest job and can be calculated by Equation (1).

$$makespan = min_{s \in Solutions} max_{j \in Jobs}(F_j) \tag{1}$$

where Solutions is the set of all possible solutions and Jobs is the set of all jobs submitted to the system and $F_j$ represents the time when job j is finished [15].

## 4 Applying ACO to the job scheduling problem

ACO has been applied for several problems closely related to job scheduling problem in grid computing. Therefore, it seems an appropriate candidate in this

**Table 1.** A 15 x 10 subset of the Expected Time to Compute (ETC) matrix with semi-consistent, low job heterogeneity and low resource heterogeneity. $r_i$ $(1 \leq i \leq 10)$ is a resource

| job | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_8$ | $r_9$ | $r_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $j_1$ | 4.3 | 135.9 | 194.5 | 223.8 | 303.0 | 346.6 | 418.4 | 419.0 | 487.5 | 516.7 |
| $j_2$ | 353.5 | 472.9 | 478.4 | 117.5 | 338.8 | 573.7 | 431.4 | 655.5 | 545.8 | 207.0 |
| $j_3$ | 17.2 | 18.9 | 24.5 | 33.2 | 39.7 | 47.2 | 58.4 | 62.1 | 76.2 | 83.5 |
| $j_4$ | 182.2 | 358.3 | 180.1 | 539.6 | 511.8 | 474.3 | 131.0 | 89.7 | 571.8 | 363.1 |
| $j_5$ | 55.3 | 99.7 | 107.0 | 198.0 | 219.1 | 237.3 | 238.0 | 244.3 | 306.8 | 353.1 |
| $j_6$ | 405.8 | 88.7 | 59.9 | 82.9 | 442.6 | 352.3 | 62.9 | 207.1 | 49.5 | 413.2 |
| $j_7$ | 55.8 | 79.0 | 84.7 | 110.2 | 143.9 | 176.0 | 180.7 | 182.5 | 189.3 | 203.8 |
| $j_8$ | 166.6 | 334.4 | 310.1 | 194.7 | 88.5 | 349.0 | 310.6 | 118.6 | 124.6 | 110.5 |
| $j_9$ | 108.2 | 119.3 | 138.7 | 144.7 | 169.7 | 193.5 | 223.4 | 248.1 | 293.3 | 312.9 |
| $j_{10}$ | 109.8 | 127.7 | 30.2 | 125.0 | 136.7 | 56.3 | 92.7 | 25.5 | 89.7 | 187.8 |
| $j_{11}$ | 114.1 | 118.3 | 121.8 | 318.0 | 332.2 | 391.5 | 391.7 | 477.3 | 541.0 | 568.9 |
| $j_{12}$ | 186.5 | 635.7 | 727.6 | 308.9 | 747.8 | 563.8 | 146.4 | 485.3 | 728.3 | 205.8 |
| $j_{13}$ | 212.8 | 216.0 | 248.4 | 269.2 | 306.1 | 381.9 | 413.4 | 439.2 | 486.6 | 582.6 |
| $j_{14}$ | 250.6 | 70.4 | 106.1 | 136.2 | 163.5 | 100.1 | 211.3 | 108.0 | 113.5 | 174.7 |
| $j_{15}$ | 114.9 | 213.2 | 233.7 | 261.5 | 280.4 | 317.3 | 354.6 | 385.0 | 453.0 | 513.1 |

environment. An ACO-based scheduler is introduced in this section which follows the ACO algorithm design explained in [6].

The first step in any ACO algorithm is to determine what information the pheromone trail encodes. The pheromone trail allows the ants to communicate indirectly to each other and share useful information about optimal solutions. Since we have n jobs to be scheduled into m resources, a pheromone matrix, $\tau$, of size n x m is needed in which the value of $\tau$[i][j] represents the favorability of assigning job i to resource j. In addition to the information encoded in the pheromone trial, the ants use heuristic information to build their solutions. In this study, the following heuristic has been used:

$$\eta_{ij} = \frac{1}{free[j]} \tag{2}$$

where the function free[j] denotes the time when machine j will be free.

If free[j] is small, $\eta_{ij}$ will be a very large value. Thus if a resource is free earlier, it will be more desirable.

To measure the quality of the solutions, a fitness function must be defined. As mentioned earlier, the makespan is considered in this study. The makespan of a solution is a good indicator of the general throughput of the grid system. Small value of makespan means that the algorithm is finding efficient mapping of jobs to resources.

To allow the communication among ants about the current states of the resources, a rule for updating the pheromone trial is needed and it is defined by Equation (3):

$$\tau_{ij} = \rho * \tau_{ij} + \Delta\tau_{ij} \tag{3}$$

**Algorithm 1** The ACO-based scheduler

**Step 1: While (Number of generation ≤ Maximum number of generations) do steps 2-5.**

**Step 2: Initialization**

1. Let n be the number of jobs, m be the number of resources and k be the number of ants.
2. Initialize the pheromone deposit value $\tau_{ij}$to a small value.
3. Let free[0..m-1]=0.
4. Initialize the pheromone evaporation.

**Step 3: For each ant A, do the following:**

1. Randomly select the job-resource pair (i, j) and add it to the scheduled list.
2. For all unscheduled jobs, do the following:
   a. description free[j] = free[j]+ETC[i, j].
   b. Calculate the heuristic function using Equation (2).
   c. Calculate the probability matrix using Equation (6).
   d. Determine the highest $\rho_{xy}$ value.
   e. Select the next job-resource pair (i=x, j=y) and add it to the scheduled list.

**Step 4: Find the best solution**

1. Calculate the makespan of all solutions using Equation (1).
2. The best solution is the one with the minimum makespan.

**Step 5: Pheromone update**

1. Calculate $F_k$ using Equation (5).
2. Calculate $\Delta\tau_{ij}$ using Equation (4).
3. Calculate the new pheromone trail using Equation (3).

where i, j ∈ the best solution, $\rho$ $(0 < \rho \leq 1)$ is the decay parameter which is used to allow the ants to forget poor information and $\Delta\tau_{ij}$ is the amount of pheromone deposit to the path and is defined by Equation (4) as:

$$\Delta\tau_{ij} = \frac{1-\rho}{F_k} \tag{4}$$

where

$$F_k = max(free[i]) \tag{5}$$

As mentioned earlier, the ants use both the information encoded in the pheromone trial and the heuristic function to build their solutions. Each ant starts with two lists: scheduled list which is empty and unscheduled list which contains n jobs. The first job-resource mapping will be selected randomly. The heuristic function $\eta_{ij}$, then, will find the best resources available to process the unscheduled jobs. A job x is selected probabilistically to be processed by resource y using the transition rule defined by Equation (6) as follows:

$$p_{xy} = \frac{[\tau_{xy}]^{\alpha} * [\eta_{xy}]^{\beta} * \frac{1}{ETC[x,y]}}{\sum[\tau_{xy}]^{\alpha} * [\eta_{xy}]^{\beta} * \frac{1}{ETC[x,y]}} \tag{6}$$

where $\alpha$ and $\beta$ are two parameters used to define the relative weight of the pheromone and the heuristic respectively.

This process is repeated until all unscheduled jobs have been mapped and that represents a complete solution. Each ant in the colony does the same procedure to build a solution. The colony size, which is the number of ants, is k. The pheromone trail update rule is applied as described in Equation (3) when all k ants built their solutions. Similar to other meta-heuristics algorithms, ACO has many parameters that need to be tuned. To achieve an efficient performance, these parameters should be chosen carefully. The maximum number of generations is 5, each generation (or colony) has 50 ants. The decay parameter $\rho$ was 0.5 while $\alpha = 2$ and $\beta = 30$. Algorithm 1 shows the complete ACO-based scheduler for job scheduling in grid computing.

## 5 Applying GA to the job scheduling problem

Genetic Algorithm (GA), a population-based heuristic search method which is inspired by natural evolution. GA starts with an initial population, a group of solutions usually generated randomly, then seeks to find the approximately best solution by applying selection, crossover and mutation operators. GA has been quite used for solving many optimization problems closely related to the job scheduling problem.

In this study, we used the GA-based scheduler proposed in [3] to solve the problem of job scheduling in grid computing. The scheduler main steps are explained as follows:

1. The solution representation: the authors used the direct representation to encode the individuals. In direct representation, each individual is represented as a list called solution of size equals to the number of jobs. The value of solution[x] represent the resource where job x is allocated. Therefore, the values in this list are integers in the range [1, m], where m is the total number of resources.
2. The initial generation: they used a population of 200 individuals which is generated either randomly or by seeding the population with one individual generated using the Min-Min heuristic method [12] and 199 individuals generated randomly.
3. The fitness function: the makespan was used to evaluate the fitness of individuals.
4. Create a new generation by applying GA operators:
    a. The selection operator: the rank-based roulette wheel method has been used. To guarantee that the best solution remains in the population, the elitist generational was implemented.
    b. The crossover operator: the one-point scheme is used with a probability of 0.6.
    c. The mutation operator: the mutation is done with a probability of 0.4 by randomly select an individual then randomly select a job and reassign it to a different resource.

5. Repeat from step 3 until the stopping criteria are true. The proposed GA stops when one of the following conditions is occurred: (a) 1000 generations, (b) the elite individual remain the same for 150 iterations, or (c) all individuals have the same solution.

The Min-Min heuristic [12] used in generating the initial population starts by computing the minimum completing time CT[i,j] for all jobs and resources. Then finds the job x with minimum CT[i, j] and allocates it to the resource that obtains it.

In this study, a meta-heuristic method, which is ACO, will be used to seed the initial population of GA for solving the problem of static independent job scheduling in grid computing. The proposed method will run ACO first for a specific number of iterations. The solutions found by ACO, then, will be used to seed the initial population of GA together with the random method. Three versions of the GA proposed by [3] are then considered here to study the effects of seeding the initial population of GA. All the three versions follow the main steps explained above. The only difference is the way the initial population is generated, The first version, called Random-GA, uses the random method to generate the initial population while the second version, called Min-GA, generates the initial population by seeding it with one individual generated using the Min-Min heuristic and 199 individuals generated randomly. The third version, called ACO-GA, runs first the ACO meta-heuristic algorithm, then uses the 50 solutions found by the ACO and 150 individuals generated randomly to seed the initial population.

## 6 Experiments and Results

Experiments have been carried out using an Intel T2080 CPU @ 1.73GHz with 2GB RAM and all programs were written in Java language. To obtain the best, worst and median values, each algorithm was executed 100 times for each instance of the 12 ETC matrices. Table 2 provides the actual makespan while figure 1 shows the median makespan values. In the table, the first column represents the instance name, the second column represents the type of the results, namely Best, Worst or Median, the third, fourth, fifth, sixth and seventh columns represent the makespan found by Min-Min, Rnd-GA, Min-GA, ACO, and ACO-GA. The best results are indicated in bold.

In the table of results, the following abbreviation has been used to identify the type of ETC matric: x-y-zzww, where:

- x represents the type of probability distribution. In this study, the uniform distribution (u) has been used only.
- y represents the type of consistency, which has one of: c: consistent, i: inconsistent, s: semi-consistent.
- description zz represents the job heterogeneity, which could be either high (hi) or low (lo).

**Table 2.** The best, worst, and median makespan results

| Instance | B/W/M | Min | Rnd-GA | Min-GA | ACO | ACO-GA |
|---|---|---|---|---|---|---|
| | **Best** | 16613907.00 | 11081565.00 | 9050105.00 | 8570055.00 | **8095185.00** |
| **u-c-hihi** | **Worst** | 16613907.00 | 12423588.00 | 9830633.00 | 8845919.00 | **8355396.00** |
| | **Median** | 16613907.00 | 11709830.00 | 9495092.00 | 8680503.00 | **8210842.00** |
| | **Best** | 254880.70 | 193386.30 | 163221.10 | 159259.60 | **155335.40** |
| **u-c-hilo** | **Worst** | 254880.70 | 201751.70 | 171102.40 | 161058.40 | **157831.00** |
| | **Median** | 254880.70 | 196916.15 | 166404.4 | 159787.10 | **155945.10** |
| | **Best** | 558377.30 | 357089.90 | 295008.20 | 279429.20 | **263120.30** |
| **u-c-lohi** | **Worst** | 558377.30 | 400989.20 | 317053.70 | 289153.40 | **272172.70** |
| | **Median** | 558377.30 | 374781.20 | 305801.80 | 282148.90 | **266419.40** |
| | **Best** | 7789.00 | 6463.50 | 5496.90 | 5358.20 | **5221.90** |
| **u-c-lolo** | **Worst** | 7789.00 | 6758.30 | 5684.00 | 5499.60 | **5289.60** |
| | **Median** | 7789.00 | 6614.10 | 5574.75 | 5456.90 | **5246.45** |
| | **Best** | 3943275.90 | 6074364.30 | 3239078 | 3253050.00 | **3056285.00** |
| **u-i-hihi** | **Worst** | 3943275.90 | 7653952.60 | 3488099.00 | 3385014.00 | **3232376.00** |
| | **Median** | 3943275.90 | 7052776.30 | 3358354.00 | 3314572.00 | **3132457.00** |
| | **Best** | 85887.20 | 143994.60 | 77287.60 | 78051.70 | **75524.60** |
| **u-i-hilo** | **Worst** | 85887.20 | 176494.20 | 79323.70 | 80282.70 | **78018.50** |
| | **Median** | 85887.20 | 160123.45 | 78262.25 | 79462.65 | **76563.10** |
| | **Best** | 138091.40 | 211945.90 | 110578.30 | 111224.20 | **104657.00** |
| **u-i-lohi** | **Worst** | 138091.40 | 266207.60 | 118082.70 | 115520.40 | **111124.30** |
| | **Median** | 138091.40 | 238080.55 | 114355.80 | 113785.90 | **107216.20** |
| | **Best** | 3112.20 | 4538.00 | 2547.10 | 2594.90 | **2484.20** |
| **u-i-lolo** | **Worst** | 3112.20 | 5773.00 | 2755.60 | 2646.40 | **2552.70** |
| | **Median** | 3112.20 | 5182.50 | 2674.15 | 2621.20 | **2513.45** |
| | **Best** | 10591575.00 | 8580789.60 | 6534393.00 | 6044180.00 | **5578565.00** |
| **u-s-hihi** | **Worst** | 10591575.00 | 11212404.00 | 7950509.00 | 6405586.00 | **5877259.00** |
| | **Median** | 10591575.00 | 9492592.40 | 7191440.00 | 6133150.00 | **5707337.00** |
| | **Best** | 150658.20 | 152887.30 | 115180.30 | 108923.20 | **104959.00** |
| **u-s-hilo** | **Worst** | 150658.20 | 192228.80 | 125965.10 | 111047.00 | **108439.20** |
| | **Median** | 150658.20 | 175001.90 | 120388.80 | 110017.70 | **106000.80** |
| | **Best** | 317909.00 | 241510.20 | 188685.70 | 169244.60 | **158247.80** |
| **u-s-lohi** | **Worst** | 317909.00 | 328735.10 | 236169.00 | 180159.50 | **168558.20** |
| | **Median** | 317909.00 | 282073.55 | 210498.70 | 175702.80 | **161362.30** |
| | **Best** | 5357.20 | 5320.30 | 3927.20 | 3717.50 | **3589.90** |
| **u-s-lolo** | **Worst** | 5357.20 | 6600.50 | 4350.20 | 3792.40 | **3681.40** |
| | **Median** | 5357.20 | 5807.75 | 4167.85 | 3763.40 | **3632.85** |

- description ww represents the resource heterogeneity, which could be either high (hi) or low (lo).

The results show clearly that ACO-GA outperforms the other approaches for all instances of ETC matrix tested in finding the minimum makespan. The performance order for all cases of the approaches used from best to worst was: ACO-GA, ACO, Min-GA, Rnd-GA, and then Min-Min. ACO provided the second best performance as it outperformed Min-GA, Rnd-GA, and Min-Min in

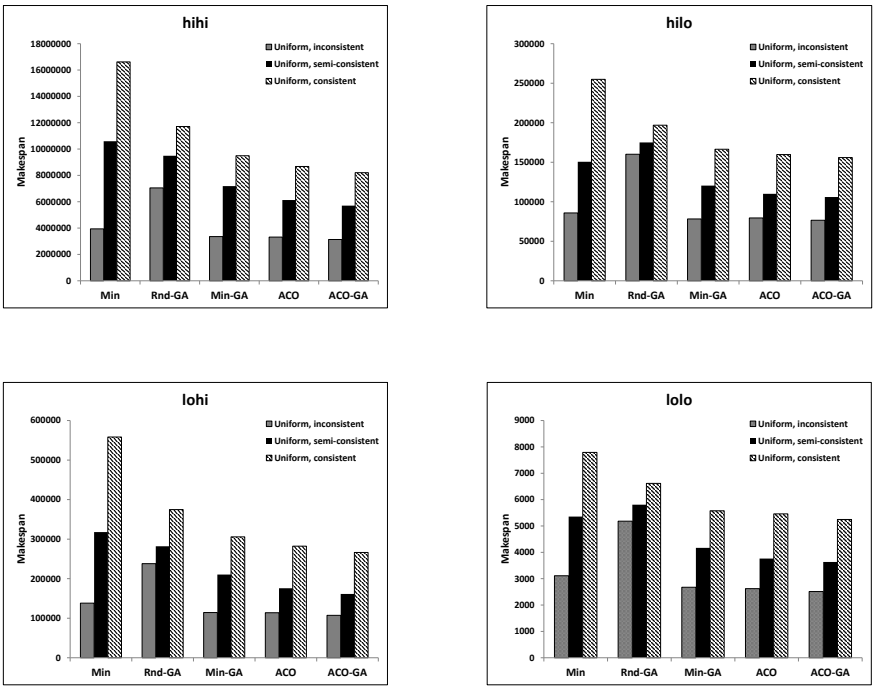


**Fig. 1.** Median makespan obtained by five methods in four different cases

consistent and semi-consistent cases. However, Min-GA performed better than ACO, GA, and Min-Min in some of inconsistent cases. The results also show the significant effect of seeding the initial population of GA either in the case of Min-GA which outperforms the normal Rnd-GA or in the case of ACO-GA which outperforms all other approaches used in this study.

## 7 Conclusions and future work

One of the major difficult tasks in grid computing systems is job scheduling. An efficient job scheduler will significantly improve the overall performance of grid computing systems. Similar to scheduling problems in conventional distributed systems, job scheduling in these systems is known to be NP-complete. However, in grid computing systems job scheduling problem is much more complex due to the fact that the jobs and resources in these environments have a high degree of heterogeneity, the environment is dynamic, and the problem is multi-objective. Therefore, the use of meta-heuristics, such as ACO and GA, is necessary to cope in practice with its complexity and difficulty. GA is a robust search method that has been used successfully to solve the problem of job scheduling in computa-

tional grid. However, the solution found by GA could be improved by providing diversity in its initial population. One method to provide diversity is seeding the initial population with solutions generated by heuristics. In this study, a meta-heuristic method, which is ACO, has been used to seed the initial population of GA for solving the problem of static independent job scheduling in grid computing. The proposed algorithm can find better mappings than other approaches found in the literature in terms of minimizing the makespan. The ETC matrix model has been used to examine the performance of the proposed method. The experimental results show that the proposed method is outperforming the other methods used in this study which are Min-Min heuristic, GA with an initial population generated randomly, GA with an initial population generated using Min-Min heuristic and random method, and ACO.

The proposed GA seems a promising approach to scheduling in grid computing systems. However, there is much space for further improvements such as adding another objective so that the problem will be multi-objectives, adding a local search scheme to the proposed method and testing it in a dynamic environment.

## Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1.

## References

1. Abraham, A., Buyya, R., Nath, B.: Natures heuristics for scheduling jobs on computational grids. In: The 8th IEEE international conference on advanced computing and communications (ADCOM 2000). pp. 45–52 (2000)
2. Alobaedy, M.M., Ku-Mahamud, K.R.: Scheduling jobs in computational grid using hybrid acs and ga approach. In: Computing, Communications and IT Applications Conference (ComComAp), 2014 IEEE. pp. 223–228. IEEE (2014)
3. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing 61(6), 810–837 (2001)
4. Carretero, J., Xhafa, F., Abraham, A.: Genetic algorithm based schedulers for grid computing systems. International Journal of Innovative Computing, Information and Control 3(6), 1–19 (2007)
5. Dorigo, M., Birattari, M., et al.: Swarm intelligence. Scholarpedia 2(9), 1462 (2007)
6. Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: Algorithms, applications, and advances. In: Handbook of metaheuristics, pp. 250–285. Springer (2003)
7. Eaton, J., Yang, S.: Dynamic railway junction rescheduling using population based ant colony optimisation. In: Computational Intelligence (UKCI), 2014 14th UK Workshop on. pp. 1–8. IEEE (2014)

8. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a new computing infrastructure. Elsevier (2003)
9. Foster, I., Kesselman, C.: The history of the grid. computing 20(21), 22 (2010)
10. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
11. Guntsch, M., Middendorf, M.: Applying population based aco to dynamic optimization problems. In: International Workshop on Ant Algorithms. pp. 111–122. Springer (2002)
12. Ibarra, O.H., Kim, C.E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM (JACM) 24(2), 280–289 (1977)
13. Izakian, H., Abraham, A., Snasel, V.: Performance comparison of six efficient pure heuristics for scheduling meta-tasks on heterogeneous distributed environments. Neural Network World 19(6), 695 (2009)
14. Julstrom, B.A.: Seeding the population: improved performance in a genetic algorithm for the rectilinear steiner problem. In: Proceedings of the 1994 ACM symposium on Applied computing. pp. 222–226. ACM (1994)
15. Kołodziej, J., Xhafa, F.: Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population. Future Generation Computer Systems 27(8), 1035–1046 (2011)
16. Lorpunmanee, S., Sap, M.N., Abdullah, A.H., Chompoo-inwai, C.: An ant colony optimization for dynamic job scheduling in grid environment. International Journal of Computer and Information Science and Engineering 1(4), 207–214 (2007)
17. Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F.: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing 59(2), 107–131 (1999)
18. Mathiyalagan, P., Suriya, S., Sivanandam, S.: Modified ant colony algorithm for grid scheduling. International Journal on Computer Science and Engineering 2(02), 132–139 (2010)
19. Nesmachnow, S., Alba, E., Cancela, H.: Scheduling in heterogeneous computing and grid environments using a parallel chc evolutionary algorithm. Computational Intelligence 28(2), 131–155 (2012)
20. Pacini, E., Mateos, C., Garino, C.G.: Distributed job scheduling based on swarm intelligence: A survey. Computers & Electrical Engineering 40(1), 252–269 (2014)
21. Paul, P.V., Ramalingam, A., Baskaran, R., Dhavachelvan, P., Vivekanandan, K., Subramanian, R.: A new population seeding technique for permutation-coded genetic algorithm: Service transfer approach. Journal of Computational Science 5(2), 277–297 (2014)
22. Ritchie, G., Levine, J.: A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments (2004)
23. Schmitt, L.J., Amini, M.M.: Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study. European Journal of Operational Research 108(3), 551–570 (1998)
24. Xhafa, F., Abraham, A.: Computational models and heuristic methods for grid scheduling problems. Future Generation Computer Systems 26(4), 608–621 (2010)
25. Xhafa, F., Kolodziej, J., Barolli, L., Fundo, A.: A ga+ ts hybrid algorithm for independent batch scheduling in computational grids. In: Network-Based Information Systems (NBiS), 2011 14th International Conference on. pp. 229–235. IEEE (2011)