Optimal Grid Scheduling Using Improved Artificial Bee Colony Algorithm

T. Vigneswari, M. A. Maluk Mohamed

Abstract—Job Scheduling plays an important role for efficient utilization of grid resources. The grid environments are dynamic, heterogeneous, and unpredictable. The scheduling algorithm is required to find suitable resource allocation for different jobs to the available resources. The goal of grid scheduling is to find an effective resource utilization to reduce job tardiness, when scheduled. To get optimal scheduling plan, evolutionary algorithms and swarm intelligence algorithms have been effectively used in the literature for grid scheduling. Meta-heuristics techniques like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Genetic Algorithm (GA) have been efficiently used for the grid scheduling problem in literature. The proposed model utilizes a novel Artificial Bee Colony algorithm (ABC) along with Min-Min algorithm to identify the initial food source. Simulation results show the performance improvement of the proposed algorithm over other swarm intelligence techniques.

Keywords—Grid Computing, Grid Scheduling, Artificial Bees Colony Algorithm (ABC) Algorithm, Resource Management.

I. INTRODUCTION

GRID is used for large scale distributed and parallel computing systems. Each node in a grid environment shares their resources dynamically during the execution of an application. Usually the resources are heterogeneous and distributed geographically. The selection of a resource depends on the availability, cost, and Quality of Service (QoS) requirement of the applications [1]. Grids have been widely used for Computational Services, Data Services, Application Services, Information Services and Knowledge Services applications [2].

Grid scheduling is the activity of allocating different jobs to the available resources. Some of the resources available in grid computing are storage space, network bandwidth, CPU cycles and software. The assignment of jobs to the resources should be optimal to minimize the makespan, minimize the cost of allocated resources and maximize the throughput [3]. Various scheduling algorithms have been proposed in literature for scheduling resources in the grid environment [4], [5].

Grid environments are dynamic, heterogeneous and unpredictable computing systems sharing different services between users. Due to the grid’s heterogeneous and dynamic nature traditional methods are not applicable for grid scheduling. Scheduling is important area that needs to be addressed to achieve high grid environment performance as it

T. Vigneswari is with Kings College of Engineering, Tamilnadu, India (Phone: +91 9066002047; e-mail: vigneswari.grt@gmail.com).

M. A. Maluk Mohamed is with M. A. M College of engineering, Tamilnadu, India (e-mail: malukmohammed.mam@rediffmail.com).

This paper aims to find suitable resources allocation for every job. Scheduling decision should address effective resource utilization to reduce job tardiness, when scheduled. Finding optimal resource allocation for specific jobs which reduce jobs schedule length is a challenging research area. Scheduling problem is NP-complete problem [6] and not trivial.

To get optimal scheduling plan, evolutionary algorithms and swarm intelligence algorithms have been effectively used in the literature. Meta-heuristics techniques like Particle Swarm Optimization (PSO) [8], Ant Colony Optimization (ACO) [9], and Genetic Algorithm (GA) [10] have been efficiently used for the grid scheduling problem in literature. Meta heuristic algorithms have been found to be more effective if the initial population is selected from existing sub optimal scheduling algorithms like First Come First Serve (FCFS) earlier used for solving queuing problems [11]. Longest Job First (LJF) and FCFS as initial schedule was used for the Fuzzy Particle Swarm based scheduling [12], Shortest Job First (SJF) as initial scheduler for Swift Scheduler algorithm [13] and SJF and LJF as initial schedule for GA based scheduler [14] have been used in literature.

There is increasing interest in Multi-Objective Evolutionary Algorithm (MOEA) [15] which combines evolutionary algorithms with theoretical frameworks of multi-criteria decision making. Though some real world problems are reduced to a single objective usually it is hard to define a single objective’s aspects. Defining multiple objectives provides a better idea of a task. Multi-objective evolutionary algorithms yield potential solutions, optimal in some sense. Multi-objective optimization environment’s main challenge is minimizing distance of generated solutions to Pareto set and maximizing developed Pareto set diversity [16]. A good Pareto set is obtained by guiding the search process through reproduction operators/fitness assignment design strategies. To diversify, special care is ensured in the selection process. Similar care prevents non-dominated solutions from getting lost.

This paper proposes a new scheduling technique based on Artificial Bee Colony algorithm (ABC) for Grid scheduling. ABC algorithm is a meta-heuristic approach based on foraging behaviour of honey bee swarm [17]. It does not require cross over rate and mutation rate as in case of genetic algorithm to solve the problem. ABC algorithm has been effectively used to solve constrained and unconstrained function optimization problems. ABC’s advantage over other optimization algorithms includes its [18]:

- Simplicity, flexibility and robustness
- Use of reduced control parameters compared to other

Abstract—Job Scheduling plays an important role for efficient utilization of grid resources available across different domains and geographical zones. Scheduling of jobs is challenging and NP-complete. Evolutionary / Swarm Intelligence algorithms have been extensively used to address the NP problem in grid scheduling. Artificial Bee Colony (ABC) has been proposed for optimization problems based on foraging behaviour of bees. This work proposes a modified ABC algorithm, Cluster Heterogeneous Earliest First Min-Min Artificial Bee Colony (CHMM-ABC), to optimally schedule jobs for the available resources. The proposed model utilizes a novel Heterogeneous Earliest Finish Time (HEFT) Heuristic Algorithm along with Min-Min algorithm to identify the initial food source. Simulation results show the performance improvement of the proposed algorithm over other swarm intelligence techniques.

Keywords—Grid Computing, Grid Scheduling, Heterogeneous Earliest Finish Time (HEFT), Artificial Bee Colony (ABC) Algorithm, Resource Management.

I. INTRODUCTION

GRID is used for large scale distributed and parallel computing systems. Each node in a grid environment shares their resources dynamically during the execution of an application. Usually the resources are heterogeneous and distributed geographically. The selection of a resource depends on the availability, cost and Quality of Service (QoS) requirement of the applications [1]. Grids have been widely used for Computational Services, Data Services, Application Services, Information Services and Knowledge Services applications [2].

Grid scheduling is the activity of allocating different jobs to the available resources. Some of the resources available in grid computing are storage space, network bandwidth, CPU cycles and software. The assignment of jobs to the resources should be optimal to minimize the makespan, minimize the cost of allocated resources and maximize the throughput [3]. Various scheduling algorithms have been proposed in literature for scheduling resources in the grid environment [4], [5].

Grid environments are dynamic, heterogeneous and unpredictable computing systems sharing different services between users. Due to the grid’s heterogeneous and dynamic nature traditional methods are not applicable for grid scheduling. Scheduling is important area that needs to be addressed to achieve high grid environment performance as it

T. Vigneswari is with Kings College of Engineering, Tamilnadu, India (Phone: +91 9066002047; e-mail: vigneswari.grt@gmail.com).

M. A. Maluk Mohamed is with M. A. M College of engineering, Tamilnadu, India (e-mail: malukmohammed.mam@rediffmail.com).

This paper proposes a new scheduling technique based on Artificial Bee Colony algorithm (ABC) for Grid scheduling. ABC algorithm is a meta-heuristic approach based on foraging behaviour of honey bee swarm [17]. It does not require cross over rate and mutation rate as in case of genetic algorithm to solve the problem. ABC algorithm has been effectively used to solve constrained and unconstrained function optimization problems. ABC’s advantage over other optimization algorithms includes its [18]:

- Simplicity, flexibility and robustness
- Use of reduced control parameters compared to other
search techniques
• Hybridization ease with other optimization algorithms
• Ability to handle objective cost with stochastic nature
• Easy implementation with basic mathematical/logical operations.

Section II reviews some of the related works available in the literature, Section III details about the methods used. Section IV discusses the simulation environment and the results obtained, Section V concludes the paper.

II. RELATED WORKS

Job scheduling in a grid environment are based on popular heuristic algorithms such as min-min, fast greedy, Tabu Search and Ant System. Heuristics algorithms proposed for job scheduling rely on static environment and expected value of execution times. Casanova et al. [19] and Baraglia et al. [20] proposed heuristic algorithms to solve scheduling issues based on differing static data, for example, execution time and system load.

Fidanova et al. [21] presented a heuristic scheduling algorithm based on ACO, designed to achieve high throughput computing with load balancing in grid environment. An ACO algorithm to schedule large-scale work-flows with various QoS parameters was proposed by [22] which enabled users to specify their QoS preferences and also define minimum QoS thresholds for specific applications. A task scheduling strategy based on Chaotic ACO Algorithm, using the randomness periodicity and regularity of chaotic motion to improve the quality of ant's individuals was presented by [23].

Garg and Singh [24] proposed the design/implementation of hierarchical discrete PSO (H-DPSO) for grid environment's dependent task scheduling to minimize makespan and total cost. In H-DPSO particles are dynamic hierarchically arranged with good particles lying above and having higher influence on swarm. Karimi and Motameni [25] presented Tasks Scheduling in Computational Grid using an H-DPSO where particles were initialized by Min-Min algorithm. In [26] searching of particle in each sub-swarm, a scenario for grid resource allocation was produced. Tao et al. [27] proposed a rotary hybrid discrete particle swarm optimization (RHDPSO) algorithm for QoS Constrained Grid Workflow Scheduling Optimization.

Genetic algorithms minimize average job completion time through optimal job allocation on each node in application level scheduling [28]. Khanli et al. [29] presented a Reliable Job Scheduler using Resource Fault Occurrence History (RFOH) in Grid Computing. RFOH stored the number of faults occurred and number of jobs in execution using this resource. Based on the RFOH information, GA was used to find an optimum schedule. Kashyap et al. [30] proposed a security driven scheduling model for large computational grid using genetic algorithm which helps to incorporate security into task scheduling.

A binary artificial bee colony (BABC) algorithm for grid computing was proposed by [31]. The proposed technique incorporates a flexible ranking strategy (FRS) to improve balance between exploration and exploitation. Simulation results for benchmark job scheduling issues showed that the proposed method’s performance is better than alternatives like simulated annealing, genetic algorithms, and particle swarm optimization. Selvi and Umarani [32] presented Comparative Study of GA and ABC for Job Scheduling. Simulations were conducted with five different job sets. Numerical results showed that hybrid GA-ABC job scheduling gave lower makespan compared to GA and ABC scheduling algorithms.

To overcome local minima problem various hybrid techniques have been proposed in the literature. Mandloi and Gupta [33] presented adaptive job scheduling based on ACO with Genetic Parameter Selection. GA was used to control the parameters of ACO algorithm. Failure in execution of jobs and job completion were taken for evaluating the performance of proposed scheduling algorithm. Evaluation of results proved that optimization algorithm using GA and PSO was better than FCFS algorithm. Xue et al. [34] proposed a hybrid clonal selection genetic algorithm (HCSGA) for solving task scheduling problem. Hu et al. [35] presented HPSOA Hybrid Particle Swarm Optimization (HPSO) algorithm to resolve dynamic web services selection with QoS global optimal in grid workflow.

Many techniques other than swarm intelligence and Evolutionary algorithms have been proposed for grid scheduling. Fidanova et al. [36] introduced a grid computing tasks scheduling algorithm based on simulated annealing (SA). Martino [37] designed a two level scheduling system, with first level being formed by a computing node set – each with a local scheduling policy – and a second level formed by super scheduler. The proposed technique used local search strategy to improve convergence when number of jobs is large as in the real world operations. Pooranian et al. [38] presented Group Leader Optimization Algorithm (GLOA) for Job Scheduling. The capability of Linear Programming (LP) and GA was combined with LP-driven GA algorithm which aims the best meta-scheduling that minimizes the combined cost of all users with negligible time overhead [39]. Rao [40] proposed a Differential Evolution approach to generate an optimal scheduling which helps to complete the jobs within a minimum period of time.

The Heterogeneous Earliest Finish Time (HEFT) algorithm achieves shorter schedule lengths compared to other algorithms [41]. In HEFT algorithm, ranking function is used to compute a value of the subtasks and accordingly scheduled. Abdelkader and Omara [42] proposed a Clustering Based HEFT with Duplication (CBHD) for dynamic task scheduling. The proposed CBHD is a combination of HEFT and triplet clustering algorithm which achieves better execution time and load balancing. Tang et al. [43] developed stochastic HEFT (SHEFT) scheduling algorithm which includes stochastic attributes to facilitate efficient scheduling precedence constrained stochastic tasks.

III. METHODOLOGY

In this work a hybrid ABC using modified HEFT based clustering along with min-min algorithm to create the initial population is proposed. The block diagram of the proposed
A. Generation of Initial Populations

Heterogeneous Earliest Finish Time (HEFT) is a list scheduling heuristics [41] based on the two components: a priority function, which is used to order all nodes in the task graph at compile time; and an objective function which must be minimized. The priority function used by HEFT is based on “upward ranking” which is the length of the critical path from a task to the exit task, including the computation cost of this task. The upward rank of a task is the sum of the average execution cost of this task over all available processors and a maximum computed over all its successors. The terms of this maximum are the average communication cost of an edge and the upward rank of the successor.

The Earliest Start Time (EST) is the moment when the execution of a job can actually begin on a resource. An execution can start either when a processor becomes available or when all needed data has arrived on the resource. Adding the execution cost, the Earliest Finish Time (EFT) of a task is obtained. HEFT uses the EFT as the objective function for selecting the best processor for a job [44].

The weights assigned to the nodes are calculated based on the predicted execution times of the jobs. The weights assigned to the edges are calculated based on predicted times of data transferred between the resources. In homogeneous environments the weights are equal to the predicted times. In heterogeneous environments, the weights must be approximated considering different predictions for execution times on different resources, and for different data transfer times on different data links.

The ranking phase is performed traversing the workflow graph upwards, and assigning a rank value to each of the tasks. Rank value is equal to the weight of the node plus the execution time of the successors. The successor execution time is estimated, for every edge being immediate successors of the node, adding its weight to the rank value of the successive node, and choosing the maximum of the summations. A list of resources is arranged, according to the decreasing rank values.

In the proposed Cluster HEFT (CHEFT), the grid is first partitioned into clusters for better utilization of the distributed resources. The grid can be represented by an acyclic graph G(V,E) where V is a set of nodes which represents the resources and E is a set of directed edges which represents the interconnection between the resources. The fan-out of the data communication equipment is the number of edges incident from it and the fan-in is the number of edges incident to it. Primary input is a resource with zero fan-in and primary output is a resource with zero fan-out. In the given acyclic graph G, each node in V is assigned weights except for the primary input resource which is assigned zero weight.
The total number of edges is reduced in the clustered network. After clustering, the number of edges is equal to the number of clusters as only one fan out is radiated from each cluster. This helps to produce a good initial solution for the scheduling process. Fig. 2 shows a sample clustering of the resources. The pseudo code of the proposed Cluster HEFT (CHEFT) is shown in Fig. 3.

Min-Min algorithm schedule tasks by considering the execution time of the tasks on the resources. The Min-min algorithm begins with the set U of all unscheduled tasks. The set of minimum completion times for each of the tasks exiting in U is found. Next, the task with the overall minimum completion time from unscheduled tasks is selected and assigned to the corresponding resource.

<table>
<thead>
<tr>
<th>pseudo code</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUSTER(resource, vector, i)</td>
<td>Represent fan-in resources of i as vector</td>
</tr>
<tr>
<td>While test vector</td>
<td></td>
</tr>
<tr>
<td>i = a randomly selected resource from vector</td>
<td></td>
</tr>
<tr>
<td>j = numbers of resource that can be reached by i</td>
<td></td>
</tr>
<tr>
<td>If current cluster size + j &lt; max cluster size</td>
<td>Assign the set S of resource that can be reached by i to the current cluster</td>
</tr>
<tr>
<td>Remove the clustered resource from vector</td>
<td></td>
</tr>
<tr>
<td>Do While</td>
<td></td>
</tr>
<tr>
<td>HEFT(cluster, s)</td>
<td>Store processing capability of each cluster in descending order in s</td>
</tr>
<tr>
<td>$S_{max} = \sum_{i=1}^{n} s(i)$ (cluster with q cpu time)</td>
<td></td>
</tr>
<tr>
<td>While set of task u ≠ 0</td>
<td></td>
</tr>
<tr>
<td>For each task i</td>
<td>Compute the optimal capacity $S_i^*$</td>
</tr>
<tr>
<td>If $\sum_i S_i^* = S_{max}$</td>
<td>For each ready task $i$, $s_i = \frac{T_{task}}{S_i^<em>} &gt; S_i^</em>$</td>
</tr>
<tr>
<td>Else</td>
<td>For each ready task $i$, $s_i = S_i^*$</td>
</tr>
<tr>
<td>End If</td>
<td></td>
</tr>
<tr>
<td>Compute actual t-level</td>
<td></td>
</tr>
<tr>
<td>While ready task ≠ Ø</td>
<td></td>
</tr>
<tr>
<td>i = task with minimum t-level, $t_i = t$ level of i</td>
<td></td>
</tr>
<tr>
<td>K = fastest cluster free at $t_i$, OR first cluster free after $t_i$</td>
<td></td>
</tr>
<tr>
<td>$est_i = est_i = MAX(t_u, availability, Time_{max})$ p=1, $s_i = 0$</td>
<td></td>
</tr>
<tr>
<td>While $S_i &lt; S_i^*$</td>
<td></td>
</tr>
<tr>
<td>If p=1 OR $est_i + t_p &lt; est_i^* + t_p$, p-1 # add cluster</td>
<td></td>
</tr>
<tr>
<td>V (p) ~k, $est_i = est_i + t_p = p + 1$, $s_i = s_i + s(k)$</td>
<td></td>
</tr>
<tr>
<td>K = next fastest cluster. Free at $t_i$, OR</td>
<td></td>
</tr>
<tr>
<td>Next cluster. free after $t_i$</td>
<td></td>
</tr>
<tr>
<td>If k=null break # all clusters analyzed</td>
<td></td>
</tr>
<tr>
<td>$est_i = MAX(t_u, availability, Time_{max})$</td>
<td></td>
</tr>
<tr>
<td>Compute b-level of ready task s=1</td>
<td></td>
</tr>
<tr>
<td>While true</td>
<td></td>
</tr>
<tr>
<td>$hbl(w)$ task with higher b-level</td>
<td></td>
</tr>
<tr>
<td>r = task with minimum b-level</td>
<td></td>
</tr>
<tr>
<td>if r exists in hbl break</td>
<td></td>
</tr>
<tr>
<td>Remove fastest cluster assigned at task r</td>
<td></td>
</tr>
<tr>
<td>Assign it to task $hbl(w)$</td>
<td></td>
</tr>
<tr>
<td>Re-evaluate proc. Time of r and $hbl(w)$</td>
<td></td>
</tr>
<tr>
<td>Re-evaluate b-level of r and $hbl(w)$</td>
<td></td>
</tr>
<tr>
<td>$W = W + 1$</td>
<td></td>
</tr>
<tr>
<td>End If</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3 The proposed HEFT algorithm
for all tasks $T_i$ in meta-task $M_i$
for all resources $R_j$
$C_{ij} = E_{ij} + r_i$
do until all tasks in $M_i$ are mapped
for each task in $M_i$ find the earliest
assign task $T_i$ to the resource $R_j$ that gives the earliest completion time
delete task $T_i$ from $M_i$
update $r_i$
update $C_{ij}$ for all $i$
end do

Fig. 4 Steps in Min-Min scheduling algorithm

B. Scheduling Using Artificial Bee Colony (ABC) Algorithm

Artificial Bee Colony (ABC) algorithm is based on the social behaviour of honey bee colonies. Honey bees share information about the location, quantity and quality of foods. This information sharing activity can be used for resource management problems of grid scheduling. There are three types of honey bees in a bee colony. They are Onlooker bees, employed bees and scout bees. Employed bees search the locations of food in parallel and inform to other bees by dancing. Onlooker bees evaluate and select the best solution among the solutions given by all the employed bees. Scout bees start a new search for solution. Detailed description of ABC can be found in [46]-[48].

For each task $T_i$, which will be processed until completion in resource $R_j$, $C_{ij}$ is the time taken to complete the task. An $T \times N$ matrix $X=x_{ij}$ where $x_{ij}=1$ if task is assigned to the resource else 0 is created. We use a single objective function based on makespan in this work. ABC is based on two natural processes: recruitment of bees to a food source and source abandonment. The difference between ABC and other swarm intelligence algorithms is that in the former, problem’s solutions are represented by food sources, not bees. In comparison, bees act as variation operators discovering (generating) food sources based on existing ones. The employed bees are equal in number to the number of food sources (generating) food sources based on existing ones. The employed bees are equal in number to the number of food sources with an employed bee being assigned to one source. On reaching the source, the bee calculates a new solution (flying to another food source) and retains the best solution using greedy selection technique. When a source fails to improve iteratively, it is dumped and replaced by food source using greedy selection technique. A high quality food source is profitable sources and randomly choose a better food source.

Onlooker bees in the hive watch dances advertising food sources profitability and choose a source site depending on dance frequency proportional to source quality. Classical ABC includes 4 phases [49] [50].

Initialization Phase:

Food sources, with SN population size, are generated randomly by scout bees. The Artificial Bee number is NP. Each food source $x_m$ is a vector to optimization problem, $x_m$ has D variables and D is searching space dimension of objective function needing optimization. Initial food sources are produced randomly by (1)

$$x_m = l_i + \text{rand}(0,1)^*(u_i - l_i)$$

where $u_i$ and $l_i$ are upper and lower bound of the objective function’s solution space, rand (0,1) is a random number within the range [0,1].

Employed Bee Phase:

Employed bees fly and locate a new food source in the food source’s neighbourhood. A neighbour food source $v_{mi}$ is determined/calculated by using (2)

$$v_{mi} = x_{mi} + \phi_{mi} (x_{mi} - x_{g})$$

where $x_g$ is randomly selected food source, $i$ is randomly chosen parameter index, $mi \varphi$ is a random number within range [-1,1]. The food source fitness is essential to find a global optimal. Fitness is computed using (3). After which a greedy selection is applied between $x_m$ and $v_{mi}$.

$$fH_m(x_m) = \frac{1}{1 + f_m(x_m)}, f_m(x_m) > 0$$

$$fH_m(x_m) = \frac{1}{1 - |f_m(x_m)|}, f_m(x_m) < 0$$

where $f_m(x_m)$ is the objective function value of $x_m$.

Onlooker Bee Phase:

Onlooker bees see waggle dance in dance area and calculate food sources profitability and randomly choose a better food source. Food source quantity is evaluated by profitability and profitability of all food sources and determined by (4)
\[ P_i = \frac{\text{fit}_i(x_m)}{\sum_{a=1}^{m} \text{fit}_a(x_m)} \]  

where \( \text{fit}_a(x_m) \) is the fitness of \( x_m \).

Scout Phase:

The scouts randomly search for new solutions. If solution \( x_i \) is abandoned, a new solution \( x_m \) is discovered. The \( x_m \) is defined by (5).

\[ x_m = l_1 + \text{rand}(0,1)*(u_1 - l_1) \]  

where \( x_m \) is new generated food source, \( \text{rand}(0,1) \) is a random number within range \([0,1]\), \( u_1 \) and \( l_1 \) are upper and lower bound of objective function’s solution space.

In the proposed CHMM-ABC algorithm’s first step, the initial swarm \( x_i \) \((i = 1, \ldots, SN)\) solutions are produced using proposed HEFT algorithm and with min-min algorithm. Min-min algorithm runs until all the tasks are assigned.

In the algorithm’s second step, for every employed bee, whose total number equals half the number of food sources, a new source is produced. In the next step, an onlooker bee chooses a food source with probability based on the fitness and produces a new source in chosen food source site. A novel fitness function is proposed and given by (6).

\[ F = \frac{\alpha \sum P_i + \beta M_i}{\sum (B_{i1} - B_{i1-1})^2} \]  

where \( \alpha + \beta = 1 \), \( P_i \) are the selected clusters, \( B_{avg} \) is the average bandwidth among the selected clusters, and \( B_{i1-1} \) is the difference in bandwidth between two hops.

Onlookers are distributed to sources, sources checked on whether they are to be abandoned. If number of cycles by which a source cannot be improved is bigger than a predetermined limit, source is considered exhausted.

Employed bee linked to exhausted source is now a scout searching randomly in the problem domain using (7).

\[ x_p = x_i^{\min} + \text{rand} \times (x_i^{\max} - x_i^{\min}) \]  

IV. RESULTS AND DISCUSSION

Simulations were conducted using 50 jobs, and the resources were grouped into 10 clusters. Each job is independent of each other and the resources are dynamically distributed. Using dynamic arrival time of jobs with different requirement of resources, the proposed scheduling algorithm is run five times and the average computed. During every run makespan value is computed. The initial solution starts with 40 bees of which 20 bees are worker bees and the remaining are onlooker bees. Totally 250 iterations are performed but in all the runs the solution terminated within 100 iterations. The obtained makespan is shown graphically in Fig. 5 and Table I.

| Table I: AVERAGE MAKESPAN FOR THE PROPOSED CHMM-ABC |
|-----------------|-----------------|-----------------|
| Average Makespan| ABC | Proposed CHMM-ABC |
| Run1            | 39.954          | 33.166          |
| Run2            | 39.984          | 33.725          |
| Run3            | 39.993          | 34.167          |
| Run4            | 39.555          | 33.437          |
| Run5            | 39.653          | 34.919          |

Fig. 5 shows the average makespan of 5 runs achieved in the simulation. It is observed that average makespan is reduced after 25 iterations for the proposed CHMM-ABC and the makespan keeps reducing with the iteration as the proposed CHMM-ABC refines the solution iteratively for the proposed fitness function. In all the five runs the solution converged within 100 iterations.

A significant decrease of makespan of 11.94% to 16.99% is achieved by the proposed CHMM-ABC when compared to the classic ABC. The makespan of all the runs for the classical ABC and proposed CHMM-ABC is depicted in Figs. 6 and 7 respectively.

![Fig. 5 Average makespan RMSE](image-url)
Resource utilization improves by 4.7% in the proposed technique. Table II tabulates the standard deviation achieved for ABC, CHMM-ABC and technique proposed by [12].

<table>
<thead>
<tr>
<th>Technique</th>
<th>Average Makespan</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>39.8277</td>
<td>0.2079</td>
</tr>
<tr>
<td>Proposed CHMM-ABC</td>
<td>33.8830</td>
<td>0.6877</td>
</tr>
<tr>
<td>Fuzzy Particle Swarm Algorithm [12]</td>
<td>38.0428</td>
<td>0.6613</td>
</tr>
</tbody>
</table>

From Table II it can be seen that proposed CHMM-ABC offers performance improvement of 10.93% compared to other work in literature.

V. CONCLUSION

Grid computing ensures that multiple machines though located in different places physically act as if they are one huge virtual machine. Grid scheduling is the activity of allocating different jobs to the available resources. This work proposed an improved grid scheduling algorithm using proposed HEFT and min min algorithm to obtain the initial population for the Artificial Bee Colony heuristic. Simulations
are conducted with 50 jobs, and the resources are grouped into 10 clusters. Simulation result shows a decrease of makespan of 11.94% to 16.99% by the proposed ABC when compared to the classic ABC.

REFERENCES


