International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:11, 2014

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 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 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 [7]. 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

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

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

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. Heuristic 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

methodology is shown in Fig. 1.

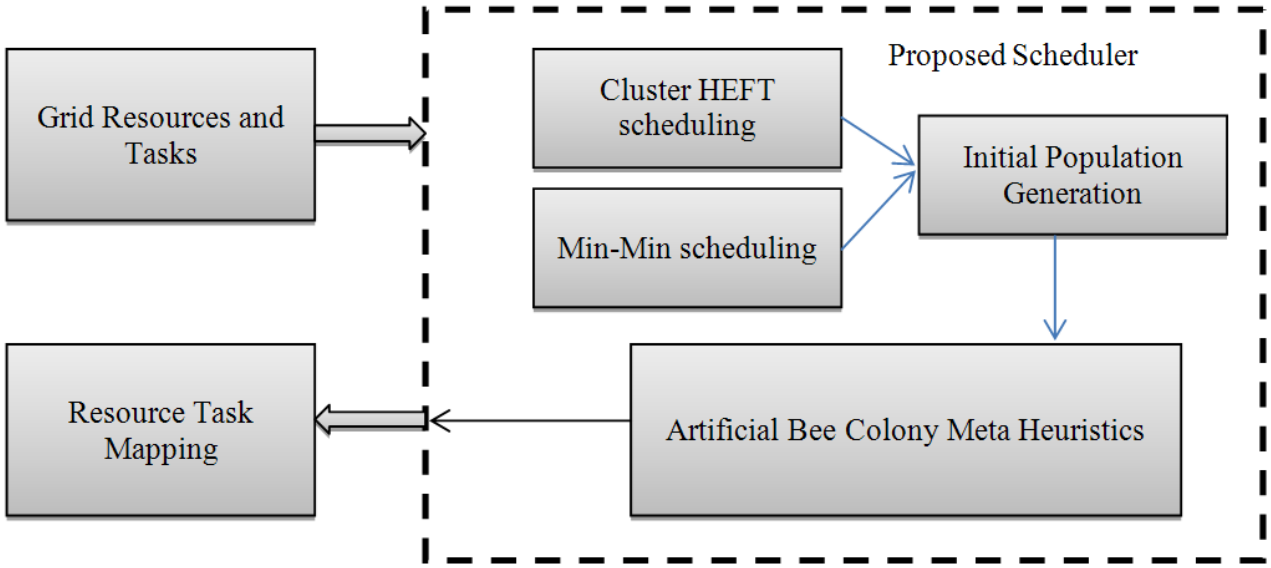

Fig. 1 Block diagram of the Proposed Methodology

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.

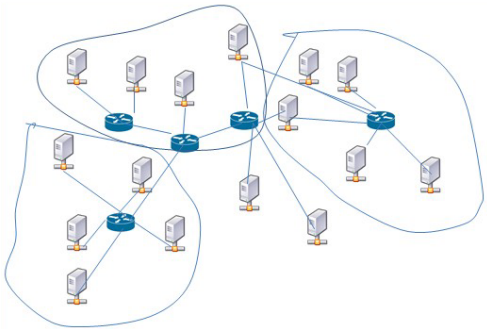

Fig. 2 Cluster formed before HEFT

The resources in the grid are clustered into a sparser network so that the maximum fan-out is minimized. Thus, 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.

Last, the newly scheduled task is removed from U and the process repeats until all tasks are scheduled [45]. The flow of min- min algorithm is shown in Fig. 4. In Fig. 4 rj denotes the expected time which resource Rj will become ready to execute a task after finishing the execution of all tasks assigned to it. First, the Cij entries are computed using the Eij (the estimated execution time of task Ti on resource Rj) and rj values. For each task Ti, the resource that gives the earliest expected completion time is determined by scanning the ith row of the C matrix (composed of the Cij values). The task Tk that has the minimum earliest expected completion time is determined and then assigned to the corresponding resource. The matrix C and vector r are updated and the above process is repeated for tasks that have not yet been assigned to a resource.

CLUSTER(resource, vector, i)
Represent fan-in resources of i as vector
While test vector
    If vector>0
        i = a randomly selected resource from vector
        j = numbers of resource that can be reached by i
        If current cluster size + j <max cluster size
        Assign the set S of resource that can be reached by i to the current cluster
    Remove the clustered resource from vector
Do While
HEFT(cluster, s)
  Store processing capability of each cluster in descending order in s
  $S_{max} = \sum_{j=1}^{q} s(j)$ (cluster with q cpu time)
  While set of task $u \neq 0$
  For each task i
  Compute the optimal capacity S_i^*
  If $\sum_i S_i^* = S_{MAX}$
    For each ready task i
    $si' = (\frac{S_{MAX}}{\sum_i S_j^*} > S_i^*$
  Else
    For each ready task i, $si'=S_i^*$
 End If
  Compute actual t-level
  While ready task =∅
  i=task with minimum t-level,$tl_i = t\ level\ of\ i$
  K=fastest cluster free at tl_i OR first cluster free after tl_i
    $est'_i = est_i = MAX(tl_i, availability, Time_K)$ p=1, s_i=0
  While $S_i < S'_i$
  If p=1 OR $est_i + t_i$, p < $est_i' + t_i$, p -1 # add cluster
    V (p) =k, $est'_i = est_i, p = p + 1, s_i = s_i + s(k)$
  K=next fastest cluster. Free at tl_i OR
    Next cluster. free after tl_i
  If k=null break # all clusters analyzed
  $est_i = MAX(tl_i, availability, Time_K)$
    Compute b-level of ready task ,w=1
    While true
    $hbl(w)$ task with higher b-level
    $r = task$ with minimum b-level
    $if\ r$ exists in hbl break
    Remove fastest cluster assigned at task r
    Assign it to task $hbl(w)$
    Re-evaluate proc. Time of r and $hbl(w)$
    Re-evaluate b-level of r and $hbl(w)$
    $W = W + 1$
 End If

Fig. 3 The proposed HEFT algorithm

```
for all tasks Ti in meta-task Mv
 for all resources Rj
     Cij=Eij+rj
do until all tasks in Mv are mapped
 for each task in Mv find the earliest
                    find the task Tk with the minimum earliest completion time
 assign task Tk to the resource Rl that gives the earliest completion time
 delete task Tk from Mv
 update rl
 update Cil 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 x 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 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 found by scout bee, which in turn involves a random calculation of a new solution.

Foraging bee's emergent intelligent behaviour can be summarized as:

1. During initial foraging phase, bees start exploring environment randomly to locate a food source.
2. After locating a food source, bee is an employed forager starting to exploit the discovered source. The employed bee returns to hive with nectar and unloads it. After unloading she goes back to discovered source site directly or shares information about the source through a dance on the dance floor. When her source is exhausted, the scout starts a random search for a new source.
3. Onlooker bees in the hive watch dances advertising profitable sources 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)

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

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 high quality food source is selected. A neighbour food source v_{mi} is determined/calculated by using (2)

$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}) \tag{2}$$

where x_k is randomly selected food source, i is randomly chosen parameter index, mi φ 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_m.

$$fit_m(x_m) = \begin{cases} \dfrac{1}{1+f_m(x_m)}, f_m(x_m) > 0 \\ 1+\left|f_m(x_m)\right|, f_m(x_m) < 0 \end{cases} \tag{3}$$

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)

$$P_m = \frac{fit_m(x_m)}{\sum_{m=1}^{SN} fit_m(x_m)} \quad (4)$$

where $fit_m(x_m)$ 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).

$$x_m = l_i + rand(0,1)*(u_i - l_i) \quad (5)$$

where x_m is new generated food source, *rand (0,1)* is a random number within range [0,1], u_i and l_i are upper and lower bound of objective function's solution space.

In the proposed CHMM-ABC algorithm's first step, the initial swarm $\vec{x}_i$ *(i = 1, . . ., 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).

$$F = \frac{\alpha}{\sum P_i} + \beta \frac{\beta_{avg} e^{M_i}}{\sum_{i=1}^{j} \left(B_{0,i} - B_{i,i+1}\right)^2} \quad (6)$$

where $\alpha + \beta = 1$, P_i are the selected clusters, B_{avg} is the average bandwidth among the selected clusters, and $B_{i,1+1}$ 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).

$$x_{ij} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) * rand \quad (7)$$

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.

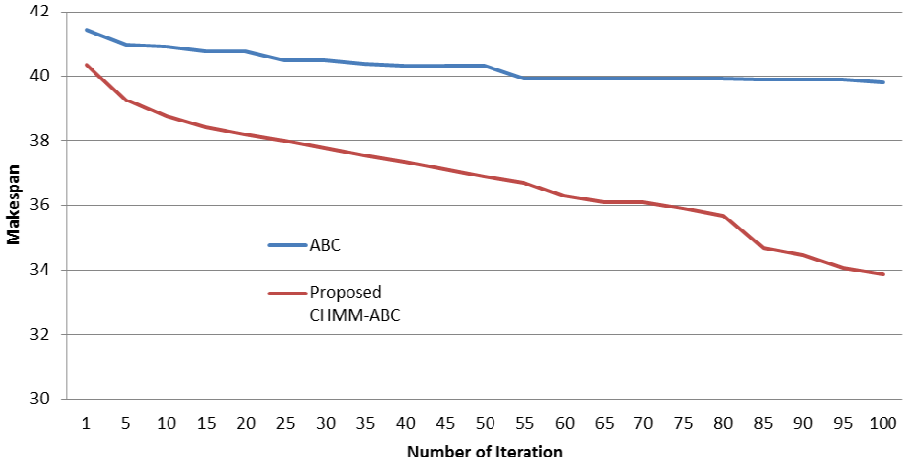

Fig. 5 Average makespan RMSE

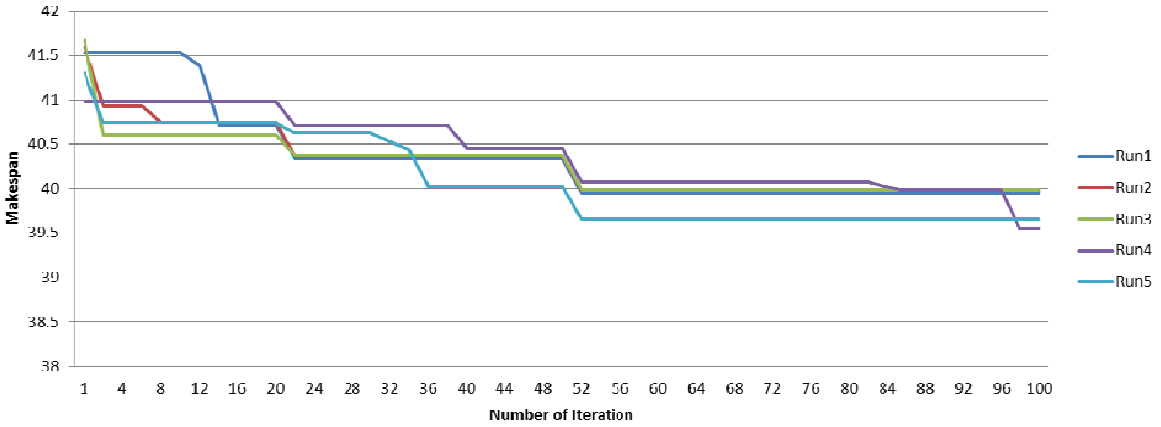

Fig. 6 Makespan of all runs for ABC

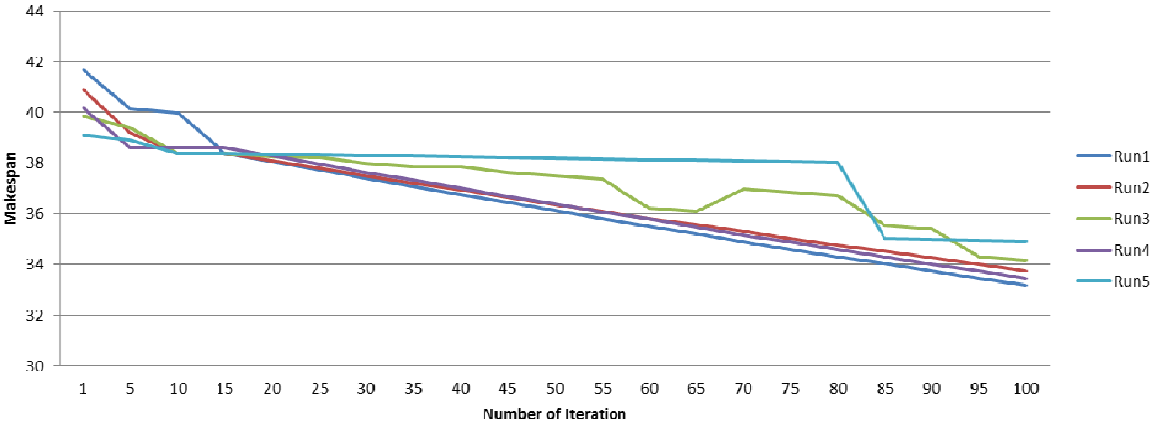

Fig. 7 Makespan of all runs for Proposed CHMM-ABC

Fig. 6 shows the makespan for all the 5 runs for 100 iterations of ABC. Result shows that all the runs had highest makespan during starting iterations and makespan reduces significantly when the number of iterations is increased. Although, it is observed in Fig. 7 that the initial run itself achieves a low makespan in the proposed CHMM-ABC. The utilization of CHEFT and Min-min algorithm for creating the initial swarm drastically improves the performance of the proposed method. Fig. 8 shows the resource utilization for the proposed system compared with ABC algorithm.

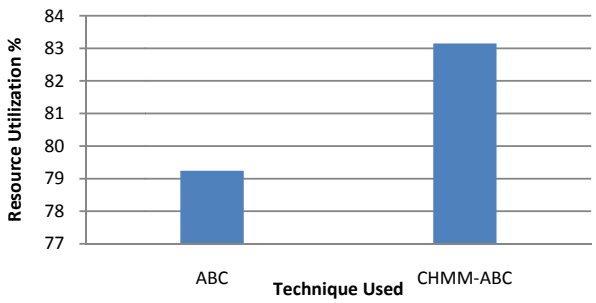

Fig. 8 Resource utilization

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 II
STANDARD DEVIATION

	Average Makespan	Standard Deviation
ABC	39.8277	0.2079
Proposed CHMM-ABC	33.8830	0.6877
Fuzzy Particle Swarm Algorithm [12]	38.0428	0.6613

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

[1] Baker, M., Buyya, R., & Laforenza, D. (2002). Grids and Grid technologies for wide-area distributed computing. Software: Practice and Experience, 32(15), 1437-1466.
[2] Foster, I. (2005). Globus toolkit version 4: Software for service-oriented systems. In Network and parallel computing (pp. 2-13). Springer Berlin Heidelberg.
[3] Garg, S. K., Buyya, R., & Siegel, H. J. (2010). Time and cost trade-off management for scheduling parallel applications on utility grids. Future Generation Computer Systems, 26(8), 1344-1355.
[4] Casavant, T. L., & Kuhl, J. G. (1988). A taxonomy of scheduling in general-purpose distributed computing systems. Software Engineering, IEEE Transactions on, 14(2), 141-154.
[5] Dong, F., & Akl, S. G. (2006). Scheduling algorithms for grid computing: State of the art and open problems. School of Computing, Queen's University, Kingston, Ontario. 1-55.
[6] Lorpunmanee, S., Sap, M. N., Abdullah, A. H., & Chompoo-inwai, C. (2007). An ant colony optimization for dynamic job scheduling in grid environment.International Journal of Computer and Information Science and Engineering, 1(4), 207-214.
[7] Sarath Chandar A P, Priyesh V, & Doreen Hephzibah Miriam D. (2012). Grid Scheduling using Improved Particle Swarm Optimization with Digital Pheromones. International Journal of Scientific & Engineering Research, 3(6).
[8] Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on (Vol. 5, pp. 4104-4108). IEEE.
[9] Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 2). IEEE.
[10] Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), 95-99.
[11] Cao, J., & Zimmermann, F. (2004, April). Queue scheduling and advance reservations with COSY. In Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International (p. 63). IEEE.
[12] Liu, H., Abraham, A., & Hassanien, A. E. (2010). Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Generation Computer Systems, 26(8), 1336-1343.
[13] Somasundaram, K., & Radhakrishnan, S. (2009). Task Resource Allocation in Grid using Swift Scheduler. International Journal of Computers, Communications & Control, 4(2).
[14] Carretero, J., & Xhafa, F. (2006). Use of genetic algorithms for scheduling jobs in large scale grid applications. Technological and Economic Development of Economy, 12(1), 11-17.
[15] Grosan, C., Abraham, A., & Helvik, B. (2007). Multiobjective evolutionary algorithms for scheduling jobs on computational grids. In International Conference on Applied Computing (pp. 459-463).
[16] Coello Coello, C. A. (2006). Evolutionary multi-objective optimization: a historical view of the field. Computational Intelligence Magazine, IEEE, 1(1), 28-36.
[17] Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. Applied soft computing, 8(1), 687-697.
[18] Bolaji, A. L. A., Khader, A. T., Al-Betar, M. A., & Awadallah, M. A. (2013). Artificial Bee Colony Algorithm, Its Variants and Applications: A Survey. Journal of Theoretical & Applied Information Technology, 47(2).
[19] Casanova, H., Legrand, A., Zagorodnov, D., & Berman, F. (2000). Heuristics for scheduling parameter sweep applications in grid environments. In Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th (pp. 349-363). IEEE.
[20] Baraglia, R., Ferrini, R., & Ritrovato, P. (2005). A static mapping heuristics to map parallel applications to heterogeneous computing systems. Concurrency and Computation: Practice and Experience, 17(13), 1579-1605.
[21] Fidanova, S., & Durchova, M. (2006). Ant algorithm for grid scheduling problem. In Large-Scale Scientific Computing (pp. 405-412). Springer Berlin Heidelberg.
[22] Chen, W. N., & Zhang, J. (2009). An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 39(1), 29-43.
[23] Ma, Y., & Wang, Y. (2012, December). Grid task scheduling based on Chaotic Ant Colony Optimization Algorithm. In Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on (pp. 469-472). IEEE.
[24] Garg, R., & Singh, A. K. (2013). Enhancing the Discrete Particle Swarm Optimization based Workflow Grid Scheduling using Hierarchical Structure. International Journal of Computer Network and Information Security (IJCNIS), 5(6), 18.
[25] Karimi, M., & Motameni, H. (2013). Tasks Scheduling in Computational Grid using a Hybrid Discrete Particle Swarm Optimization. International Journal of Grid & Distributed Computing, 6(2).
[26] Zhi-yun, Z., Tian, Z., Yong-tao, Z., & Li-ping, L. (2010, July). Optimization of grid resource allocation using improved particle swarm optimization algorithm. In Information Technology and Applications (IFITA), 2010 International Forum on (Vol. 3, pp. 99-103). IEEE.
[27] Tao, Q., Chang, H., Yi, Y., Gu, C., & Yu, Y. (2009, August). QoS constrained grid workflow scheduling optimization based on a novel PSO algorithm. In Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on (pp. 153-159). IEEE.
[28] Gao, Y., Rong, H., & Huang, J. Z. (2005). Adaptive grid job scheduling with genetic algorithms. Future Generation Computer Systems, 21(1), 151-161.
[29] Khanli, L. M., Far, M. E., & Ghaffari, A. (2010). Reliable job scheduler using RFOH in grid computing. Journal of Emerging Trends in Computing and Information Sciences, 1(1), 43-47.
[30] Kashyap, R., & Vidyarthi, D. P. (2011). Security-driven scheduling model for computational Grid using genetic algorithm. In Proceedings of the World Congress on Engineering and Computer Science (Vol. 1).
[31] Kim, S. S., Byeon, J. H., Liu, H., Abraham, A., & McLoone, S. (2012). Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization. Soft Computing, 1-16.
[32] elvi, V., & Umarani, R.. (2013) "Comparative Study of GA and ABC for Job Scheduling ", International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2, Issue-6.
[33] Mandloi, S., & Gupta, H. (2013). Adaptive job Scheduling for Computational Grid based on Ant Colony Optimization with Genetic Parameter Selection. International Journal. of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume-3 Number-1 Issue-9.
[34] Xue, X., & Gu, Y. (2010). Global optimization based on hybrid clonal selection genetic algorithm for task scheduling. J Comput Inf Syst, 6(1), 253-261.
[35] Hu, C., Wu, M., Liu, G., & Xie, W. (2007, August). QoS scheduling algorithm based on hybrid particle swarm optimization strategy for grid workflow. In Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on (pp. 330-337). IEEE.
[36] Fidanova, S. (2006, October). Simulated annealing for grid scheduling problem. In Modern Computing, 2006. JVA'06. IEEE John Vincent Atanasoff 2006 International Symposium on (pp. 41-45). IEEE.
[37] Di Martino, V., & Mililotti, M. (2004). Sub optimal scheduling in a grid using genetic algorithms. Parallel computing, 30(5), 553-565.
[38] Pooranian, Z., Shojafar, M., Abawajy, J. H., & Singhal, M. (2013). GLOA: a new job scheduling algorithm for grid computing. International journal of interactive multimedia and artificial intelligence, 2(1), 59-64.
[39] Garg, S. K., Konugurthi, P., & Buyya, R. (2011). A linear programming-driven genetic algorithm for meta-scheduling on utility grids. International Journal of Parallel, Emergent and Distributed Systems, 26(6), 493-517.
[40] Rao, C. S., & Babu, B. R. (2013). DE Based Job Scheduling in Grid Environments. Journal of Computer Networks, 1(2), 28-31.
[41] Zhao, H., & Sakellariou, R. (2003). An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. In Euro-Par 2003 Parallel Processing (pp. 189-194). Springer Berlin Heidelberg.

[42] Abdelkader, D. M., & Omara, F. (2012). Dynamic task scheduling algorithm with load balancing for heterogeneous computing system. Egyptian Informatics Journal, 13(2), 135-145.
[43] Tang, X., Li, K., Liao, G., Fang, K., & Wu, F. (2011). A stochastic scheduling algorithm for precedence constrained tasks on Grid. Future Generation Computer Systems, 27(8), 1083-1091.
[44] Suter, F., Desprez, F., & Casanova, H. (2004, January). From heterogeneous task scheduling to heterogeneous mixed parallel scheduling. In Euro-Par 2004 Parallel Processing (pp. 230-237). Springer Berlin Heidelberg.
[45] Braun, T.D., H. Jay Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys and B. Yao, (2001). 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: 810-837.
[46] Kıran, M. S., & Gündüz, M. (2012). A novel artificial bee colony-based algorithm for solving the numerical optimization problems. International Journal of Innovative Computing, Information & Control, 8(9), 6107-6121.
[47] Saab, S. M., El-Omari, N. K. T., & Hussein, H. O. (2009). Developing optimization algorithm using artificial bee colony system. Ubiquitous Computing and Communication Journal, 4(3), 391-396.
[48] Mezura-Montes, E., Damián-Araoz, M., & Cetina-Domingez, O. (2010, July). Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization. In Evolutionary Computation (CEC), 2010 IEEE Congress on (pp. 1-8). IEEE.
[49] Yan, G., & Li, C. (2011). An effective refinement artificial bee colony optimization algorithm based on chaotic search and application for pid control tuning. J Comput Inf Syst, 7(9), 3309-3316.
[50] Karaboga, D., & Akay, B. (2009). Artificial bee colony (ABC), harmony search and bees algorithms on numerical optimization. In Innovative Production Machines and Systems Virtual Conference.