I. J. Computer Network and Information Security, **2013, 6, 18-26**
Published Online May 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.06.03

Enhancing the Discrete Particle Swarm Optimization based Workflow Grid Scheduling using Hierarchical Structure

Ritu Garg, Awadhesh Kumar Singh
Computer Engineering Department, National Institute of Technology, Kurukshetra, Haryana, India
ritu.59@gmail.com, aksinreck@rediffmail.com

Abstract — The problem of scheduling dependent tasks (DAG) is an important version of scheduling, to efficiently exploit the computational capabilities of grid systems. The problem of scheduling tasks of a graph onto a set of different machines is an NP Complete problem. As a result, a number of heuristic and meta-heuristic approaches are used over the years due to their ability of providing high quality solutions with reasonable computation time. Discrete Particle Swarm Optimization is one such meta-heuristic used for solving the discrete problem of grid scheduling, but this method converge to sub optimal solutions due to premature convergence. To deal with premature convergence, in this paper we proposed the design and implementation of hierarchical discrete particle swarm optimization (H-DPSO) for dependent task scheduling in grid environment. In H-DPSO particles are arranged in dynamic hierarchy where good particles lying above in hierarchy are having larger influence on the swarm. We consider the bi-objective version of problem to minimize makespan and total cost simultaneously as the optimization criteria. The H-DPSO based scheduler was evaluated under different application task graphs. Simulation analysis manifests that H-DPSO based scheduling is highly viable and effective approach for grid computing.

Index Terms — Task Scheduling, Grid Computing, DAG, Particle Swarm Optimization

I. INTRODUCTION

Grid computing have emerged as a popular way of providing high performance computing for solving large scale problems (e.g. physics, astronomy, biology, earthquake science etc) with reasonable time, by using large number of heterogeneous resources available dynamically. Matching and scheduling of jobs or workflows in heterogeneous computing environment is crucial for coordinating the resources and problem solving. Design and implementation of efficient scheduler in grid computing is still a challenging problem. Scheduling in grid computing is the problem considering the different aspects like the environment (static or dynamic), type of objectives (resource centric or application centric), number of objectives (single or multi-objective), job dependencies (independent or dependent) etc. In this paper, we consider the scheduling problem in which 1) expected execution time for each task on each resource is known prior to execution 2) performance of individual application is optimized by considering 3) two conflicting objectives of makespan and economic cost simultaneously 4) tasks are dependent having precedence order (workflow) among them represented by directed acyclic graph (DAG) model.

Grid scheduling problem is well known NP-complete problem [1], so numerous heuristic and meta-heuristic search techniques like genetic algorithm (GA), evolutionary algorithm (EA), simulated annealing (SA), ant colony optimization (ACO), and particle swarm optimization (PSO) are used to solve the problem. Concept of discrete particle swarm optimization [2] is used to effectively map job scheduling solution to PSO particle. Hierarchical particle swarm optimization (HPSO) is another such meta-heuristic proposed by Janson and Middendorf [3] as a variant of PSO. It has increased diversity of population and better convergence. Hence, in this paper, we implemented the efficient grid scheduler for dependent task grid (workflow) based on hierarchical discrete particle swarm optimization strategy. Resulting hierarchical discrete particle swarm optimization (H-DPSO) based grid scheduler enables concurrent search in optimization domain and provides solutions with better convergence. H-DPSO based grid scheduler is evaluated via different sizes application task graphs.

Rest of the paper is organized as follows. Section 2, includes the related work. In section 3, we specified the problem definition. In section 4, we briefly introduced the particle swarm optimization approach and its variants available in literature. Section 5, describes the formulation of hierarchical discrete particle swarm optimization for multi-objective workflow grid scheduling. Section 6, discusses the simulation analysis and finally, section 7 gives the conclusion.

II. RELATED WORKS

The problem of grid scheduling, for DAG based (dependent task) task graph, has already been addressed in literature. Large number of heuristics is proposed which are commonly categorized as:

- List scheduling algorithms in which tasks are assigned the priorities and placed in ordered list [4, 5].
- Duplication based algorithms reduces the makespan by utilizing the idle time of resources for duplicating some of predecessor tasks [6].
- Clustering algorithm reduces the communication delay by grouping heavily communicating tasks to the same cluster and assigning them to same resource. [7]
- Meta-heuristics inspired by natural phenomena include genetic algorithm [8, 9], simulated annealing [10], tabu search [11], ant colony optimization [12], Particle swarm optimization [13] are also used to address the challenges of the problem.

Particle swarm optimization introduced by Kennedy and Eberhart [14] simulates the swarm of birds. Its ability for global searching has made PSO highly suitable for optimization. It also has fewer algorithm parameters than GA and SA. In [15] quality of different solutions found by ACO, PSO, SA and genetic algorithms were compared. The results show that PSO is highly efficient and effective in task scheduling problem. Recent analysis by [16, 17] observed that classical PSO prematurely converge to local optimum. [18] Used the PSO for scheduling workflow applications in distributed environment. [19] used a fuzzy particle swarm optimization and Izakian et al. [2] used discrete version of particle swarm optimization for grid scheduling. Another method for improving the quality of solution is hybridization with local search techniques. Discrete particle swarm optimization hybridized with SA was proposed in [20] by enhancing the local exploration ability of PSO to effectively solve the task assignment problem.

In this work, we proposed the design and implementation of efficient grid scheduler based on hierarchical discrete particle swarm optimization. By using the hierarchical structure particles are arranged in dynamic hierarchy where good particles lying above in hierarchy are having larger influence on the swarm. The changing arrangement of individuals helps preserving diversity, avoid premature convergence in the early iterations and promote convergence towards global optimum. Hierarchical particle swarm optimization introduced in [3] has been successfully applied in solving continuous optimization functions. It was also used as an efficient method for economic load dispatch problem [21].

III. Problem statement

The purpose of dependent task (workflow) Grid scheduling is to efficiently assign various precedence constrained tasks in the workflow to different available grid resources. We model the task workflow as a directed acyclic graph (DAG): Let G = (V, E) be a DAG, with V as the set of vertices representing n different tasks t_i $(1 \le i \le n)$ and E is the set of edges (t_i, t_j) representing precedence constraint among these pair of tasks i.e. task t_i is immediate parent to task t_j and task t_j is immediate child to task t_i. The edge weight e_{ij} between task t_i and t_j denotes data communication between them. In workflow structure child task cannot execute until all its parent tasks have finished their execution. A task without any predecessor is called an entry task and a task without any successor is called an exit task.

To compute a schedule, scheduling algorithm requires:

- List of resources available with the grid. Let a set R represents the m number of resources available in the grid where each resource $r_j \in R$, $(1 \le j \le m)$. Resources have varying processing capability delivered at different prices. So, a Cost vector requires in which $cost_j$ specifies cost of using the resource r_j per unit of time.
- Expected time to compute (ETC) matrix, in which entry ETC_{ij} gives estimated execution time to complete task t_i on resource r_j. Task execution time information can be found from the specifications provided by the user or from the literature [22, 23]
- Bandwidth linkage between any two resources. A m × m Data Transfer Time matrix is taken representing the data transfer time (for a data unit) between two resources i.e. each entry $B_{s,t}$ is used to store the time required to transfer a data unit from resource r_s to r_t .

It is necessary to define some attributes before discussing the objective functions. Let, a task t_i is to be scheduled on resource r_j and the attributes $ST(t_i)$ and $FT(t_i)$ represent the starting time and finish time of a task t_i on resource r_j respectively. These are formally defined as follows:

$$ST(t_i) = \max_{t_p \in pred(t_i)} \{FT(t_p) + CT_{p,i}\} \quad (1)$$

$$FT(t_i) = ST(t_i) + ETC_{ij} \quad (2)$$

where $pred(t_i)$ is the set of immediate predecessor tasks of task $_{ti}$ and $CT_{p,i}$ is the total communication time required to transfer data units from task t_p (scheduled on resource r_s) to task t_i (scheduled on resource r_j), which is calculated as follows:

$$CT_{p,i} = e_{ij} \times B_{s,j} \quad (3)$$

For the entry task t_{entry}, the ST is defined by:

$$ST(t_{entry}) = 0 \quad (4)$$

For the other tasks in the task graph, the starting time and finish time are computed recursively, starting from the entry task, as shown in (1) and (2), respectively.

Instance of the problem is defined by assuming that:

- Every task t_i has to be processed on resource r_j until completion
- One resource can execute one task at a time.
- Task execution can start only after the availability of complete data from all its parent tasks after ensuring task dependency and data transfer starts after finishing the execution of the task.

The problem of task scheduling in computational grid is to assign the tasks of an application to suitable resource after ordering their execution so that task precedence constraint is satisfied. Here, each scheduling solution is represented as task assignment string which maps every task t_i onto a suitable resource r_j to achieve the desired objectives after fulfilling the dependency constraint. Direct encoding method is used to represent task assignment string. In direct encoding each schedule is defined as a schedule vector $S = [s_1, s_2, \ldots s_{no\_task}]$, where every s_i represents the resource number over which respective task is assigned, i.e. $s_i \in [1, no\_resources]$ and $i=1,2,...,no\_tasks$.

In this paper, we consider the scheduling in grid as the bi-objective optimization problem with simultaneous minimization of makespan and total cost. The makespan is defined as the finishing time of the exit task (or the last task) of the task graph and total cost represents the economic cost that an application task graph needs to pay for resource utilization. Thus the two objectives can be defined as:

$$Minimize\ Time(S) = FT(t_{exit}) \qquad (5)$$

$$Minimize\ Cost(S) = \sum C_{i,j} \qquad (6)$$

Subject to $Cost(S) < B\ and\ Time(S) < D$

Where $FT(t_{exit})$ is the finish time of exit task of a task graph and B is the cost constraint (Budget) and D is the time constraint (Deadline) required by users for workflow execution. And $C_{i,j}$ is cost of executing every task t_i on resource r_j.

IV. Overview of Some Particle Swarm Optimization Strategies

A number of PSO strategies are used for grid scheduling problems in literature. Here, overview of these strategies and their significant developments are presented to serve as performance measure for hierarchical Discrete PSO used in this paper.

A. Canonical PSO

It is population based heuristic introduced by Kenedy and Eberhart [14] inspired by the bird flocking behavior. It maintains the swarm of particles, where each particle represents a potential solution. These particles are flown through the multi dimensional search space, where position of each particle is adjusted according to its own experience and that of its neighbor. Let X_i denote the position of i^{th} particle in d-dimensional search space represented as $X_i=\{x_{i1},x_{i2},\ldots,x_{id}\}$. The particle position is updated by velocity represented as $V_i=\{v_{i1},v_{i2},\ldots,v_{id}\}$. Based on evaluation function, the particle best position $Pbest_i=\{pbest_{i1},pbest_{i2},\ldots,pbest_{id}\}$ and swarm best position $Gbest_i=\{gbest_{i1},gbest_{i2},\ldots,gbest_{id}\}$ is determined. For next iteration (t+1), particle updates its position based on individual experience ($Pbest_i$) and swarm intelligence ($Gbest_i$) by using (7) and (8).

$$V_{id}^{(t+1)} = \omega . V_{id}^{(t)}(i,j) + c_1 r_1 (Pbest_{id}^{(t)} - X_{id}^{(t)}) + c_2 r_2 (Gbest_{id}^{(t)} - X_{id}^{(t)}) \qquad (7)$$

$$X_{id}^{(t+1)} = X_{id} + V_{id}^{(t+1)} \qquad (8)$$

where c1 and c2 are the cognitive and interaction coefficients. The higher value of c1 ensures large deviation of particle in search space while higher value of c2 specifies the convergence towards its global best. To have the compromise between exploration and exploitation, time varying acceleration coefficients (TVAC) have been introduced by [16]. It is proposed that c1 decreases linearly over time, while c2 increases linearly. The values of c1 and c2 at iteration t is evaluated as

$$c_1 = (c_{1f} - c_{1i}) \frac{t}{(max\_t)} + c_{1i} \qquad (9)$$

$$c_2 = (c_{2f} - c_{2i}) \frac{t}{(max\_t)} + c_{2i} \qquad (10)$$

Where c_{1f}, c_{2f} are final values and c_{1i}, c_{2i} are initial values of coefficients respectively. The random numbers r_1 and r_2 are generated independently in range [0, 1].

The parameter ω (inertia weight) controls the momentum of particles by weighing the contribution of previous velocity. The value of ω is important to ensure convergent behavior, and to optimize the tradeoff between exploration and exploitation. Time varying inertia weight (TVIW) was introduced in [24]. The higher value of ω helps in global exploration, so it is desired at the initial stages while lower values help in local search and is needed in the later stages. So the inertial weighing function is utilized as

$$\omega = (\omega_{max} - \omega_{min}) \frac{(max\_t - t)}{(max\_t)} + \omega_{min} \qquad (11)$$

B. Binary PSO

Kennedy and Eberhart [25] introduced the first discrete version of PSO to operate on binary search space. For binary PSO, particles represent position in binary space. Each element of particle's position vector can take binary value of 0 or 1. Changes in the particle's position imply a mutation of bits, by flipping a bit from one value to another. Here velocities are defined in terms of probabilities that a bit will be in one state or the other. For updating of velocity (12) is used.

A more natural normalization of velocity is done by using sigmoid function. That is,

$$sig(V_{ij}^{(t+1)}) = \frac{1}{1+exp(-V_{ij}^{(t+1)})} \tag{12}$$

And update of position changes to

$$X_{ij}^{(t+1)} = \begin{cases} 1, \ if \ \ sig(V_{ij}^{(t+1)}) > r_{ij} \\ 0, otherwise \end{cases} \tag{13}$$

C. Discrete PSO

Another version of discrete particle swarm optimization was introduced by Izakian et al. [2] for grid scheduling problem, where swarm of particles represents the allocation of tasks over available resources. Here, solutions or task assignment strings are formulated as m×n matrix, called position matrix where m is the number of available resources and n is the number of tasks. Let X_k is the position matrix of kth particle then

$$X_k(i,j) \in \{0,1\} (\forall i,j), i \in \{1,2,..m\}, j \in \{1,2,..n\} \tag{14}$$

where X_k (i, j) = 1 means that jth task is performed by ith resource. Hence, in each column of the matrix only single element is 1 and others are 0.

Velocity of each particle is also an m×n matrix whose elements are in range [-V_{max}, V_{max}]. If V_k is the velocity matrix of kth particle, then:

$$V_k(i,j) \in [-V_{max}, V_{max}], \ (\forall i, j), \ i \in \{1,2,...m\}, \ j \in \{1,2,..., \ n\} \tag{15}$$

Also, Pbest and Gbest are m×n matrices and their elements assume value 0 or 1 as in the case of position matrices. For particle updating, we are first updating velocity matrix according to (16) and then finally position matrix is updated using (17).

$$V_k^{(t+1)}(i,j) = \omega . V_k^{(t)}(i,j) + c_1 r_1 \ (Pbest_k^{(t)}(i,j) - X_k^{(t)}(i,j)) + c_2 r_2 \ (Gbest_k^{(t)}(i,j) - X_k^{(t)}(i,j)) \tag{16}$$

$$X_k^{(t+1)}(i,j) = \begin{cases} 1, \ if \ V_k^{(t+1)}(i,j) = \max\left\{V_k^{(t+1)}(i,j) \forall i \in \{1,2,...m\}\right\} \\ 0, otherwise \end{cases} \tag{17}$$

In (17) for each column of position matrix, value 1 is assigned to the element whose corresponding element has maximum value in velocity matrix in the respective column.

D. Hierarchical PSO (H-PSO)

In this novel PSO strategy introduced by Janson and Middendorf [3], particles are arranged in hierarchy (tree) to define the neighborhood structure. In hierarchical particle swarm optimization (H-PSO), root node represents the best particle and leaf node represents the worst. Internal nodes are arranged such that particle at parent position have better fitness value than particles at the child nodes. The hierarchy is defined by the height (h), out degree (d) and total number of nodes (n) in the tree. Based on this structure, each particle is neighbored to itself and its parent. In each iteration, fitness of all particles is evaluated first. Then position of every particle within the hierarchy is updated by comparing its own best solution $Pbest_k$ to the best solution found by the particles in the child nodes. If best solution of child node $Pbest_j$ is better i.e. ($Pbest_k$ < $Pbest_j$), then particles k and j swap their places in the hierarchy. These comparisons start from top and proceed in breadth-first traversal. Because of breadth-first traversal, in each iteration particle can move down several levels, but it can move up maximally by one level in the hierarchy. For velocity updating, particle is influenced with its so far best position (Cognitive component) and by the best position of its parent directly above in the hierarchy (Neighborhood best social component). This means that for particle k the value of $Gbest_k$ equals $Pbest_i$ when i is the particle at parent position of particle k. Only the child particles of root node will use the global best position in their velocity update (after at most h iterations). This behavior of moving towards better particles at parent positions facilitates better exploration, diversity of solution and not being trapped in local optima.

V. Workflow Grid Scheduling using Hierarchal Discrete Particle Swarm Optimization (H-DPSO)

The paper presents the efficient grid scheduler for dependent task (Workflow) in order to minimize the two objectives of makespan and total cost simultaneously under deadline and budget constraints by using H-DPSO. Hierarchal PSO is very suitable for discontinuous problems like grid scheduling as it avoids premature convergence and changing arrangement of particles in the hierarchy helps in preserving divergence in the search. In order to control the global and local exploration ability, high value of branching degree is used in the beginning as it lead to faster optimization towards the global best and smaller branching degree afterwards which lead to better local exploration at the end. To effectively map the grid scheduling solution to PSO particle, we used mapping of discrete PSO [2]. The implementation steps are as follows:

1) Let N is the size of swarm. Randomly initialize the particles position matrix $X_k^{(0)} \forall k \in \{1,2,...N\}$. Here every particle $X_k^{(0)}$ as shown in (14) represents the task assignment string(S) after fulfilling the dependency constraints.
2) Initialize all particle velocity matrix ($V_k^{(0)}$) randomly within range [-V_{max} to V_{max}] and personal best of every particle is initialized to itself i.e. $Pbest_k^{(0)} = X_k^{(0)}$.
3) The fitness function is used to measure the quality of solution according the optimization objectives considered. A solution having good fitness value has more chance to be passed to next generation. Here, two objective functions for solution S (an individual) are defined as:

Makespan objective function:

$$f_{makespan}(S) = Time(S)/D \tag{18}$$

Total cost objective function:

$$f_{\cos t}(S) = Cost(S)/B \tag{19}$$

Time(S) (Makespan) and Cost(S) (Total cost) for the scheduling solution S respectively are calculated from (5) and (6) respectively. To handle the deadline and budget constraints the penalty value is added to the respective objective function if they violate the constraint, otherwise not.

Minimizing $f_{makespan}$ results in increase of f_{cost} and vice versa. The weighted aggregation is most common approach to such problems. So, the fitness value of each individual (solution S) can be estimated as:

$$F = \lambda f_{makespan}(S) + (1-\lambda) f_{\cos t}(S) \; where \lambda \in [0, 1] \tag{20}$$

For a solution, smaller value of fitness F represents that it is a better solution.

4) Arrange all the particles in the tree with branching degree (d). Evaluate the individuals according to the fitness function values. To determine the new position of the particle, compare particle's own best solution value $Pbest_k^{(0)}$ to the best solution value found by the particles in the child nodes. If best solution of child node is better than parent node, then two particles are exchanged. These comparisons start from top and proceed in breadth-first traversal.
5) For every kth particle global best i.e. $Gbest_k^{(0)}$ is chosen to be the jth particle personal best $Pbest_j^{(0)}$ where j is the particle at the parent node of kth particle.
6) While maximum number of iterations (max_t) has not been reached Do
   a) Update velocity matrix $V_k^{(t)}(i,j)$ according to (16) and the new position of particle $X_k^{(t)}$ is obtained by (17).
   b) The objective function value is evaluated again at the new position. If the objective function value at new position is better than previous best position then new position is stored as $Pbest_k^{(t)}$. Now the particle is moved in hierarchal tree as in step 4 starting from top and $Gbest_k^{(t)}$ is updated accordingly as mentioned in step 5.
   c) Increment the loop counter.

VI. Simulation Results and Discussion

We used GridSim [26] toolkit in our experiment to simulate the scheduling of workflow tasks. GridSim is a java based toolkit for modeling and simulation of resource and application scheduling in large-scale parallel and distributed computing environment such as Grid. It is flexible to support simulation of grid entities like resources, users, application tasks, resource brokers or schedulers and their behavior using discrete events.

A. Simulation Model

To simulate precedence constraint tasks in workflows, we used the different workflow models represented by random task graph and task graph corresponding to real world problems such as Gaussian elimination (GE) and Fast Fourier Transforms (FFT). We also varied the size of task graph by taking the different number of tasks. The resources were 8 computing entities with various price levels are modeled to simulate heterogeneous environment of grid. Links between resources are established through a router so that direct communication can take place between resources. Computational rating (Million instructions per second) and computational cost (in dollars) of each resource is generated randomly where cost is inversely proportional to computational rating.

In order to generate valid schedule which can meet both deadline and budget constraints specified by the user, two algorithms HEFT [5] and Greedy Cost were used to make the soft constraints of deadline and budget effectively. HEFT is a time optimization scheduling algorithm in which workflow tasks are scheduled on minimum execution time heterogeneous resources irrespective of utility cost of resources. So HEFT gives minimum makespan ($Time_{min}$) and maximum total cost ($Cost_{max}$) of the workflow schedule. Greedy Cost is a cost optimization scheduling algorithm in which workflow tasks are scheduled on cheapest heterogeneous resources irrespective of the task execution time. Thus Greedy Cost gives maximum makespan ($Time_{max}$) and minimum total cost ($Cost_{min}$) of the workflow schedule Thus Deadline (D) and Budget (B) are specified as:

$$D = Time_{max} - 0.1(Time_{max} - Time_{min}) \quad (21)$$

$$B = Cost_{max} - 0.1(Cost_{max} - Cost_{min}) \quad (22)$$

The parameter values for H-DPSO and DPSO was taken as follows.

- It is common in PSO for discrete problems [1, 2] to limit the population size between [20, 40]. Thus, in this paper the population size was set to 21.
- Based on parameter sensitivity analysis, we set $c1=2.5\rightarrow0.5$ and $c2=0.5\rightarrow2.5$.
- Inertia weight (ω) = $0.9\rightarrow0.1$
- $\lambda=0.5$ (to give equal importance to makespan and total cost).
- We have run each algorithm for 20* m*n iterations. (Where m is number of machines and n is number of tasks)

To define the hierarchy in H-DPSO, we have taken branching degree (d) =4 and height (h) =3 initially for swarm size of 21. After 40% of iterations we have reduced the branching degree d equals to 3 and h=4. Finally after 70% of iterations, we have reduced d=2 and h=5. We have done so, because high value of branching degree at the start performs better as all the particles are close to top particle in the hierarchy. Which result in better exploration in the beginning, but as branching degree is reduced it leads to better exploitation at the end. To decrease the branching degree of the tree corresponding to swarm of particles, we removed the sub-tree with best fitness value. The removed nodes are then evenly appended to the bottom of hierarchy.

B. Test Suit1

In this test suit, we used the workflow model represented by randomly generated task graph (Random). The size of random task graph was varied by considering the different number of nodes as 10, 20, 40, 60, 80 and 100. The computation cost of each task is selected randomly by the normal distribution with the mean equal to the twice of specified average computation cost of the graph. The cost of each edge was selected randomly from the normal distribution with mean equal to the product of average computation cost and the communication to computation ratio (CCR). Here CCR is taken as 0.5 to represent the computation intensive application.

The results obtained with H-DPSO and DPSO algorithms corresponding to the first test suit chosen at different size random graph structure are shown in Table I. The makespan, total cost and their fitness values are shown averaged over 10 trials. Table I. clearly specifies that H-DPSO based grid scheduler performs better than DPSO. In H-DPSO, changing arrangement of particles help preserving diversity in search, results in better exploration which ultimately leads to better optimization. Fig. 1 Shows the performance (fitness values) of each method during the search process for the case of random task graph at number of nodes=40. Similar results have been produced for random task graph at different number of nodes considered (Not shown due to similarity). At the start of search process, degree of branching is high so it optimizes faster. As the degree of branching is reducing, algorithm maintains the ability to search wider areas around the better solutions (at parent positions) which improves further the objective function value in the optimization process.

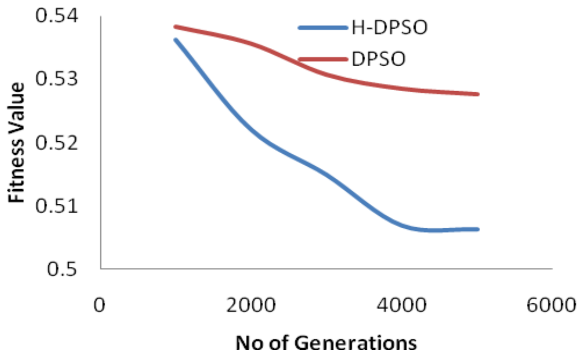

Figure 1: Illustrates the performance of H-DPSO and DPSO algorithms during the search process (Random Task Graph at n=40)

C. Test Suit 2

In this test suit, we generated task graphs corresponding to real life problems such as Gaussian Elimination (GE) and Fast Fourier Transform (FFT). The structure of these task graphs is fixed. In Gaussian Elimination algorithm [27] a parameter matrix size (m) is considered and number of tasks is defined on the basis of m as $(m^2+m-2)/2$. We have taken m as 4, 6, 8, 10, 12, and 14, thus number of tasks considered for GE task graph is 9, 20, 35, 54, 77, 104 respectively. In FFT task graph, corresponding to input vector size m, there are $2*m-1$ recursive call tasks and $m\times\log_2 m$ butterfly operation tasks (where m is 2^k for any integer k). Thus number of tasks considered for FFT is 15, 39, and 95 respectively. Table II-III shows the results obtained for each algorithm at different graph structures corresponding to GE and FFT. It can be observed that H-DPSO based grid scheduler outperforms the DPSO. Fig. 2-3 Shows the performance progress (fitness values) of each method during the search process for the case of GE at number of nodes=35 and FFT at number of nodes=39 respectively.

It is clear that H-DPSO optimizes faster at the start of search progress and further speed increases after the reduction of branching degree of hierarchal tree structure.

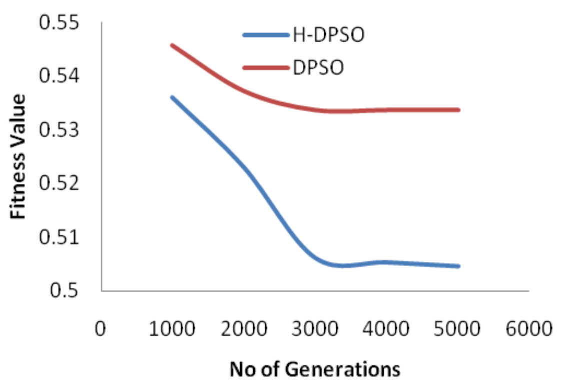

Figure 2: Illustrates the performance of H-DPSO and DPSO algorithms during the search process (GE Task Graph at n=35)

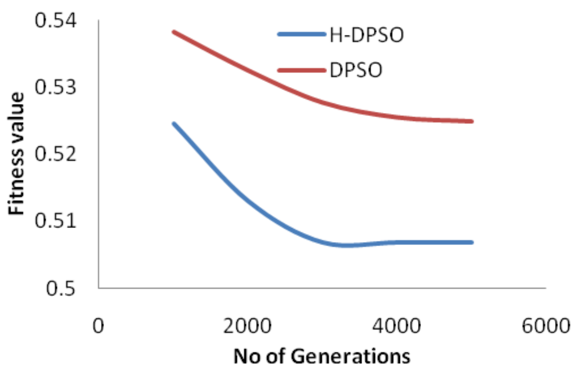

Figure 3 Illustrates the performance of H-DPSO and DPSO algorithms during the search process (FFT Task Graph at n=39)

TABLE I. RESULTS OBTAINED BY GRID SCHEDULER BASED ON H-DPSO AND DPSO FOR RANDOM TASK GRAPH

Tasks	H-DPSO			DPSO		
	Makespan	Total Cost	Fitness Value	Makespan	Total Cost	Fitness Value
10	228	546.06	0.5176884	235	564.50	0.5267853
20	356.0	1201.10	0.5072802	469.0	1236.0	0.5376753
40	612.5	2185.70	0.5034564	695.5	2292.50	0.5387854
60	735	4125.60	0.5146758	745.5	4165.40	0.5302341
80	680	4243.09	0.5063554	698.5	4323.20	0.5352855
100	756	5897.43	0.5067867	767	6123.04	0.5365489

TABLE II. RESULTS OBTAINED BY GRID SCHEDULER BASED ON H-DPSO AND DPSO FOR GAUSSIAN ELIMINATION TASK GRAPH

Tasks	H-DPSO			DPSO		
	Makespan	Total Cost	Fitness Value	Makespan	Total Cost	Fitness Value
09	200.0	536.6	0.5126238	202.0	545.4	0.5218458
20	344.0	1208.10	0.5072802	469.0	1224.0	0.5308406
35	605.5	2164.70	0.5045184	683.5	2278.60	0.533731
54	710	3956.60	0.5140406	714.5	4002.40	0.5412110
77	659	4196.09	0.5086454	670.5	4370.6	0.5352855
104	745	5768.40	0.5074564	756	5990.3	0.5383455

TABLE III. RESULTS OBTAINED BY GRID SCHEDULER BASED ON H-DPSO AND DPSO FOR FAST FOURIER TASK GRAPH

Tasks	H-DPSO			DPSO		
	Makespan	Total Cost	Fitness Value	Makespan	Total Cost	Fitness Value
15	263.0	964.80	0.5116733	290.0	971.89	0.5210113
39	514.0	2081.80	0.5068972	541.0	2158.7	0.52497711
95	691.0	5351.39	0.5055008	702.0	5449.09	0.5259126

VII. Conclusion

Task scheduling decision i.e allocating tasks to resources is crucial in grid computing. The work is motivated by the successful implementation of PSO based scheduler in literature. In this study, we have presented the implementation of hierarchical discrete particle swarm optimization (H-DPSO) based scheduler for workflow (dependent task) applications in grid environment. We considered the two conflicting objectives of minimization of execution time (makespan) and total cost simultaneously. The H-DPSO based grid scheduling is evaluated using randomly generated task graphs and task graphs corresponding to real world problems like Gaussian Elimination and Fast Fourier Transforms and compared with discrete particle swarm optimization (DPSO) based grid scheduling. The simulation results, exhibit that H-DPSO performs better for workflow grid task scheduling in comparison to DPSO.

References

[1] Ullman J: NP-complete Scheduling Problems, Journal of Computer and System Sciences. 1975, vol.10, pp. 384-393.

[2] Izakian H, Ladani BT, Zamanifar K, Abraham A: A novel particle swarm optimization approach for grid job scheduling, in Proceedings of the Third International Conference on information Systems, Technology and Management, Springer: Heidelberg, Germany, 2009, pp. 100-110.

[3] Janson S and Middendorf M: A Hierarchical Particle Swarm Optimizer and Its Adaptive Variant, IEEE Transactions on Systems, Man, Cybernatics. Vol.35 (6), 2005, pp.1272-1282.

[4] Braun T D, Siegal H J, Beck N: A comparison of Eleven Static Heuristics for Mapping a Class of independent Tasks onto Heterogeneous Distributed Computing Systems, Journal of Parallel and Distributed Computing, Vol. 61, 2001, pp. 810-837.

[5] Haluk T, Hariri S, Wu MY: Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, IEEE Transactions on Parallel and Distributed Systems, Vol. 13, 2002, pp. 260-274.

[6] Bajaj R and Agrawal D P: Improving Scheduling of Tasks in a Heterogeneous Environment, IEEE Transactions on Parallel and Distributed Systems, Vol. 15, 2004, pp. 107-118.

[7] Geras A: A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs on Multiprocessors, In J. of Parallel and Distributed Computing, Vol. 16, No.4, 1992, pp. 276-291.

[8] Subrata R, Zomaya Y A, Landfeldt B: Artificial life techniques for load balancing in computational grids, Journal of Computer and System Sciences, Vol. 73, 2007, pp. 1176-1190.

[9] Yu J and Buyya R: Scheduling Scientific Workflow Applications with Deadline and Budget constraints using Genetic Algorithms, Scientific Programming Journal, Vol. 14(1), 2006, pp. 217-230.

[10] Attiya G, Hamam Y: Task allocation for maximizing reliability of distributed systems: A simulated annealing approach, Journal of Parallel and Distributed Computing, Vol. 66, 2006, pp.1259 – 1266.

[11] Chen WH, Lin CS: A hybrid heuristic to solve a task allocation problem, Computers & Operations Research, Vol. 27, 2000, pp. 287-303.

[12] Ritchie G, Levine J: A fast, effective local search for scheduling independent jobs in heterogeneous computing environments, Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[13] Grosan C, Abraham A, and Helvik B: Multi-objective Evolutionary Algorithms for Scheduling Jobs on Computational Grids, IADIS International Conference, Applied Computing 2007, Salamanca, Spain, Nuno Guimares and Pedro Isaias (Eds.), ISBN 978-972-8924-30-0, 2007, pp. 459-463.

[14] Izakian H, Ladani B T, Zamanifar K, Abraham A: A novel particle swarm optimization approach for grid job scheduling, in Proceedings of the Third International Conference on information Systems, Technology and Management, Springer: Heidelberg, Germany,2009, pp. 100-110.

[15] Xhafa F, Abraham A: Metaheuristics for Scheduling in Distributed Computing Environments. ISBN: 978-3-540-69260-7, 2008.

[16] Ratnaweera A, Halgamuge SK and Watson H C: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput., Vol. 8, no. 3,2004, pp. 240-255.

[17] Chen CC: Hierarchical Particle Swarm Optimization for Optimization Problems, Tamkang Journal of Science and Engineering, Vol. 12 (3), 2009, pp. 289-298.

[18] Abraham A, Liu H and Zhao M: Particle swarm scheduling for workflow applications in distributed computing environments, Metaheuristics for Scheduling: Industrial and Manufacturing Applications, Studies in Computational Intelligence, Springer Verlag, Germany, 2008, pp. 327-342.

[19] Abraham A, Liu H, Zhang W, Chang TG: Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, Springer Verlag Berlin Heidelberg, 2006, pp. 500-507.

[20] Chen RM: Application of Discrete Particle Swarm Optimization for Grid Task Scheduling Problem, Advances in grid computing, 2011, DOI: 10.5772/13950.

[21] Chaturvedi K T, Pandit M and Srivastava L: Self-Organizing Hierarchical Particle Swarm

Optimization for Nonconvex Economic Dispatch, IEEE Transactions on Power Systems, Vol. 23(3) 2008, pp. 1079-1087.

[22] Jang SH, Wu X, Taylor V, Mehta G, Vahi K and Deelman E: Using performance prediction to allocate grid resources. GriPhyN Technical report 2004-25, 2004.

[23] Jarvis SA, Spooner DP, Keung HNLC, Cao J, Saini S, Nudd GR: Performance prediction and its use in parallel and distributed computing systems, Future Generation Computer System Vol. 22(7), 2006, pp. 745-754.

[24] Shi Y and Eberhart RC: Empirical study of particle swarm optimization, in Proc. IEEE Int. Congr. Evolutionary Computation, Vol. 3, 1999, pp. 101-106.

[25] Kennedy J, Eberhart RC: A discrete binary version of the particle swarm algorithm, IEEE international conference on Systems, Man, and Cybernetics, 1997, pp. 4104 - 4108.

[26] Buyya R, Murshed M: GridSim: A Toolkit for Modeling and Simulation of Grid Resource Management and Scheduling", http://www.buyya.com/gridsim, Vol. 14, 2002, pp. 1175-1220.

[27] Wu M, Gajski D: Hypertool: A programming aid for message passing system, IEEE Transactions on Parallel and Distributed Systems, Vol. 1, 1990, pp. 330-343.

Ritu Garg received B.Tech degree in Computer Science from Punjab Technical University, Jalandhar, India in 2001. She received M.Tech in the area of Computer Science from Kurukshetra University, Kurukshetra in 2006. Currently, she is pursuing Ph.D in the area of Resource Management in Grid Computing from National Institute of Technology, Kurukshetra, India. Her research interests include Grid Computing, Scheduling and Fault Tolerance.

Prof. Dr. Awadhesh Kumar Singh received B. E. degree in Computer Science & Engineering from Gorakhpur University, Gorakhpur, India in 1988. He received M.E. and Ph.D (Engg) in the same area from Jadavpur University, Kolkata, India. Currently, he is Associate Professor in the Department of Computer Engineering, National Institute of Technology, Kurukshetra, India. His present research interest is mobile distributed computing systems.