International Journal of Soft Computing 10 (1): 6-18, 2015
ISSN: 1816-9503

Hybrid Multi-Objective Workflow Scheduling on Utility Grids

[1]Sunita Bansal and [2]Chittaranjan Hota
[1]Department of Computer Science and Information Systems,
Birla Institute of Technology and Science, Pilani Campus, Pilani, 333031 Rajasthan, India
[2]Department of Computer Science and Information Systems,
Birla Institute of Technology and Science, Jawahar Nagar,
Shameerpet, Secunderabad, 500078 A.P., India

Abstract: Workflow scheduling is solved using heuristics and meta-heuristics. Heuristics are used to get an optimal solution while meta-heuristics are used to get near optimal solution. Meta-heuristics are general purpose method of solving different types of problem. This study puts Double Hybrid Multi-objective Non-Dominated Sorting Genetic Algorithm (DHNSGA-II) that gears up the convergence of the algorithm. DHNSGA-II does hybridization at two levels. At the first level it uses pre-selection operator. At the second level it uses Memetic algorithm. Pre-selection operator seeds the DHNSGA-II with the previously generated solutions. Memetic algorithm improves the current population using multi-objective local search. Apart from DHSNGA-II researchers introduced an approach to rank the Pareto frontiers because Pareto frontier has many solutions; it is nearly impossible to choose the best solution. The experimental result reveals that the proposed approach in this research performs well in optimizing the workflow scheduling jobs.

Key words: NSGA-II, hybrid, ranking, workflow, scheduling

INTRODUCTION

Due to advances in wide-area network technologies and low cost of computing resources, grid and cloud computing (Foster and Kesselman, 1999) have been established as an active research area for large-scale collaborative and distributed e-Business and e-Science applications. One of the motivational factors for Grid computing is to aggregate the capabilities of widely distributed resources and to provide non-trivial service to users. A utility grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities.

The key technologies that affect the grid efficiency involve the grid resource allocation and management. The goal of the utility grid is to utilize all available free computational resources to overcome difficulties generated by complicated tasks with enormous computing workloads. One of the current research problems is to devise new and efficient methods for resource management.

Unlike traditional services in distributed and parallel computing systems, grid scheduler does not have direct control over different nodes. Thus, grid scheduler is known as a meta scheduler. Meta scheduler has to select a node on which job should be executed where local scheduler has to execute tasks on node. Grid scheduling works in three phases, namely, resource discovery, system selection and scheduling, launching and monitoring. Discovery phase performs authorization and gathers initial information about resources. Selection phase collects information about application using application profiling. Resource information is collected from application benchmarking. After that, it generates a schedule based on a Scheduling algorithm. In the last phase, it prepares the job for transfer, transfers the jobs to local resources and monitors the execution.

Selection and scheduling phase schedules the independent and dependent tasks over different nodes. Independent tasks do not have parent and child relationship. Dependent task has the precedence orders among tasks, i.e., some tasks cannot start until other tasks are finished. Scheduling algorithms are classified as static and dynamic. Static algorithm assumes that all the information about resources is priory known and that resource properties do not vary over time. Dynamic Scheduling algorithm considers resource properties that may change unpredictably during a job execution.

Utility grid applies two kinds of algorithms, namely, heuristic and meta-heuristics. Heuristics are domain specific algorithms where meta-heuristics are general

Corresponding Author: Sunita Bansal, Department of Computer Science and Information Systems,
Birla Institute of Technology and Science, Pilani Campus, Pilani, 333031 Rajasthan, India

algorithms. Meta-heuristics are single objective or multi-objectives. Single objective Meta-heuristics minimize/maximize the single objective. Example of single objective meta-heuristics are Genetic Algorithms (GA) (Zomaya and Teh, 2001) and Simulated Annealing (SA) (Bandyopadhyay *et al*., 2008). Multi-objective meta-heuristics minimize/maximize more than one objectives. Example of multi-objective meta-heuristics are Strength Pareto Evolutionary Algorithm (SPEA-2) (Deb, 2007) and Pareto Archived Evolution Strategy (PEAS) (Deb, 2007). A Simulated Annealing-Based Multi-objective Optimization Algorithm (AMOSA) (Bandyopadhyay *et al*., 2008), Particle Swarm Optimization (PSO) (Pandey *et al*., 2010) and Ant Colony Optimization (Chen and Zhang, 2009).

This study describes and compares the three versions of NSGA-II in order to increase the convergence of NSGAII. The first version is seeded NSGA-II. Seeded GA guarantees to explore in promising region. The seeded NSGA-II differs from the unseeded NSGA-II. In seeded NSGA-II population is seeded with Meta heuristics while in unseeded NSGA-II entire population is randomly generated. The second version is multi-objective local search, known as Memetic GA. Memetic NSGA-II performs neighborhood search, on the best solutions that are near to optimal solutions. The last version is combination of Preselection and Memetic and denotes as DHNSGA-II. Apart from this, researchers have also introduced Pareto front ranking since, Pareto front contains many solutions to pick the best optimal solution, researchers have ranked the solutions. Rank of solutions is computed using Entropy and Technique for Order Performance by Similarity to Ideal Solution (TOPSIS) Method. Entropy Method (Zeleny, 1982) calculates the weight of each objective based on information. TOPSIS (Sen and Yang, 1998) algorithm ranks the solution based upon the closeness value of each alternative with reference to the ideal solution.

Due to the importance of workflow applications, many grid projects such as DAGMan (Thain *et al*., 2005) was developed to schedule jobs on Condor, Pegasus (Deelman *et al*., 2005) map executes complex workflows based on full planning. Another workflow includes UNICORE (Almond and Snelling, 1999) and ASKALON (Fahringer *et al*., 2005).

LITERATURE REVIEW

In the past, researchers had developed heuristics for workflow scheduling, namely Heterogeneous Earliest First Time (HEFT) (Topcuouglu *et al*., 2002), cluster (Bajaj and Agrawal, 2004) and duplicate scheduling (Wieczorek *et al*., 2008; Kang and Agrawal, 2000). These scheduling algorithms first map a priority to each task and then generate an overall optimal schedule. Cluster and Duplicated Scheduling algorithms map a task to a machine so that the communication time is minimized.

Researcher have also developed QoS based scheduling algorithm that minimizes the budget with deadline or vice versa. Two popular scheduling algorithms backtrack and deadline distribution have been developed. Backtrack algorithm (Menasce and Casalicchio, 2004) assigns the unmapped task to the fastest resources. At each iteration, it computes the makespan time of the assigned tasks. If the deadline of assigned task is more than total budget, it removes the most costly resources from the resource group and reiterates the scheduling algorithm. Deadline distribution (Yu, 2007) divides the workflow into sub workflow and assigns the sub-deadline to each sub workflow. Then, mapping is performed on each sub workflow. Chunlin and Layuan (2006) have introduced Dynamic Multi-Objective Grid Scheduling algorithm. They have used three QoS dimensions namely cost, deadline and reliability. The cost includes computation cost and network cost. Deadline is broken into delay and processing time. They try to satisfy the task agent and resource agent using Iterative algorithm.

Yu (2007) have explored various constraint based single and Multi-Objective Optimization Genetic algorithms on the workflow grid. Single Objective Genetic algorithm minimizes execution time/cost while meeting a specified budget/time constraint. Multi-Objective Scheduling algorithm finds the trade-off between two conflicting objectives i.e., execution time and the cost while meeting user's requirement. Chitra *et al*. (2011) have introduced new local search operator and fuzzy logic to select the best solution among available solutions in Pareto optimal way. Garg and Singh (2011) have proposed workflow scheduling based on referenced point NSGA-II. It generates multiple tradeoff schedules which minimize the time and cost together with the maximization reliability within the given quality of service constraints. Garg *et al*. (2008) have applied Hybrid Genetic Algorithm on independent tasks where each task requires more than one processing element to execute. This algorithm first assigns the task to resources using linear programming without considering the PE requirement then converts this assignment into multiple PE requirement (single node) if possible, otherwise assigns it to a dummy node.

Various methods are proposed for local search in literature (Siarry and Michalewicz, 2008). These methods replace the current solution with first, best or random solution among neighboring solutions. First, best and

random improved local search, replace current solution with first, best and random chosen better solution from neighboring solution, respectively.

Ishibuchi *et al.* (2010) have proposed various Multi-Objective Local Search Methods. Local Search methods can be used in various orders for example before during or after the genetic operators. It can also be applied on either population or offspring or both.

Various versions of NSGA-II has been introduced. Steady state and Thread based parallel version of NSGAII is famous one. Steady state NSGA-II (Durillo *et al*., 2009) does not use auxiliary population, it generates single offspring, performs non-dominated sorting on offspring and population. Thus, it's time complexity is more than simple NSGA-II. Thread based parallel NSGA-II (Nebro *et al*., 2008) uses multiple thread to compute the objectives of solutions. They have reported thread based parallel NSGA-II does not perform well.

The research differs from the above mentioned research efforts. Researchers focus on DHNSGA-II that is seeded with GA and perform local search on best solutions. Researchers have also considered three objectives namely computation cost, makespan time and network cost while others considered only two objectives. This research has also developed a new decision maker to select the best solution from Pareto front.

PROBLEM DEFINITION

This study formulates the grid resource scheduling problem into grid resource market. It considered the utility grid that is a collection of heterogeneous clusters and network resources. Clusters are heterogeneous in processor architecture and pricing. Network resources have different speed and cost. Utility grid architecture is shown in Fig. 1. Utility grid consists of four entities namely users, grid scheduler, information server and clusters connected with network resources. Users submit the application to the resource broker. Application profiling tool is used to collect the information about the application. Application is converted into workflow. The grid scheduler gets contact information of computing node and network resources from the information server. Then, resource broker contacts the resources and gets an updated status and maps the workflow task on the resources.

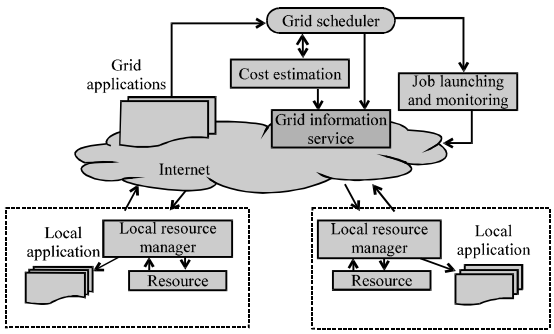

Fig. 1: Grid scheduler1

Researchers use Directed Acyclic Graph (DAG) to model an application as shown in Fig. 2a. A workflow w is represented by a DAG G = (v, e, x, c) where v and e is set of tasks and directed edges, respectively. A node in the task graph represents a task that runs non-preemptively on any cluster. Each edge is denoted by (e_i, e_j) corresponding to the data communication between t_i and t_j where t_i is called immediate parent task of t_j. Child task cannot be started until all of its parent tasks are completed. A task which does not have a parent task is called entry task t_{entry}. A task that does not have a child called exit task t_{exit}. x is computation matrix in which x_{ij} is computation time of task i on cluster j. c is the communication matrix where c_{ij} communication time between (t_i, t_j). Figure 2a-c show the DAG of 8 tasks, communication matrix and computation matrix, respectively. Researchers have considered the makespan, computation cost and network cost as objective functions.

Makespan time: The makespan time calculates as completion time. The completion time com_{ij} of a task t_i on the cluster r_j is given by:

$$com_{ij} = st_{ij} + x_{ij} \quad (1)$$

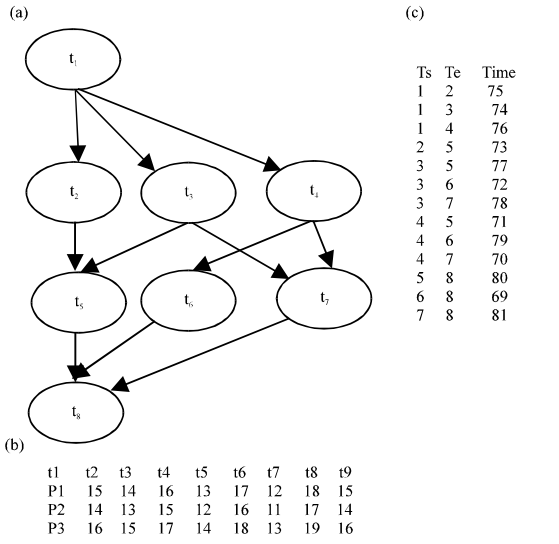

Fig. 2: Workflow

Here, st_{ij} is the start time. Start time of the entry task is zero. Other tasks start times is computed by considering the completion time of all immediate predecessors of the task. The computation time x_{ij} is added:

$$\text{Makespan (s)} = \max(com_i) \quad (2)$$

Here, com_j is the latest completion time of cluster r_j. Makespan of workflow is the maximum of com_j.

Computation cost: The computation cost is occurred when task is executed on cluster. The computation cost of task t_i on cluster r_j is determined by:

$$Cost_{ij} = rc_j \times x_{ij} \quad (3)$$

Where:
rc_j = Cluster cost in unit
x_{ij} = Computation time of task i on cluster j

Network cost: The network cost is occurred when cluster transfer task result to other cluster. The network cost is defined as:

$$Network_{ij} = nc_j \times c_{ij} \quad (4)$$

Here:
nc_j = The network cost
c_{ij} = The communication time between edge i and j. Network cost is not reciprocal of communication time

PRINCIPLE OF NSGA-II ALGORITHM

NSGA-II is proposed by Deb (2007) is one of the most efficient and famous Multi-Objective Evolutionary algorithms. NSGA-II algorithm is better than other Existing Non-Dominated Sorting algorithms because NSGA-II employees non-dominated sorting without using an external memory, time complexity is also $O(MN^2)$ and sharing parameter is not required. It uses crowded comparison operator to select best solution for mating. Following study illustrate non-domination sorting and crowded comparison operator.

Non-dominated sorting: The multi-objective optimization problem consists of a number of objectives and several equality and inequality constraints is as follows:

$$\text{Min} = (f_i(x), f_{i+1}(x), f_n(x)),\ i = 1, 2, 3,..., n \quad (5)$$

$$\text{Subject to } g_i(x) \geq 0,\ i = 1, 2, 3,..., n \quad (6)$$

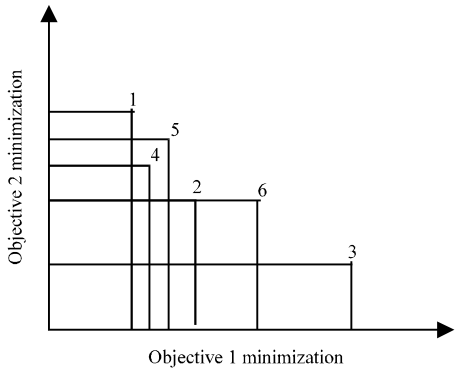

Fig. 3: Unconstrained non-dominated

$$h_i(x) = 0,\ i = 1, 2, 3,..., h \quad (7)$$

Where:
x = The decision variable vector representing a feasible solution, i.e., satisfying
m = Inequality constraints
h = Equality constraints
f_i = The ith objective function to be minimized
n = The number of objective functions

For unconstrained optimization problem, a solution x dominates y, if (a) the solution x is no worse than solution y in all objectives and (b) x is strictly better than y in at least one objective. If any one of (a) and (b) is violated, the solution x does not dominate solution y.

The dominating concept is illustrated using the Fig. 3. Here, solution '1', '2' and '3' are the non-dominating solutions. But solution '4' is dominated by solution '2' as the solution '2' is better in one objective and is equal in other objective. On the other hand, solution '6' is also dominated by solution '2'. In this case, solution '6' is not worse than solution '2' with respect to the second objective but the solution '2' is strictly better than solution '6' with respect to the first objective. Solution '5' is dominated by solution '2' and '4' as '2' and '4' are better than solution '5' in both the objectives.

Crowded comparison: Crowded comparison operator uses non-domination count of each solutions in a population. Then, it makes the different front of solutions based on non-domination count. Each solutions in a particular front computes the density of solutions with other solutions in the front. Density of individual is computed using average distance of two points on either side of this point along each of the objectives of the problem. This value is called crowding distance. After that, the solution that resides in the less crowded region is preferred in a particular group.

Non-Dominated Sorting algorithm:

- Initialize a population using uniformly random distribution
- Apply crossover and mutation operators to generate children solutions
- Combined the children and parent population to compute non-domination sorting
- Computes the objective value of each solutions
- Computes the non-domination rank of each solutions and assign different front. The solutions having lesser rank are better candidates for next generation
- Computes the crowding distance of each solution within the front. For a minimization type optimization problem, a solution x wins with another solution y if (a) solution x has better rank than solution y or (b) if the solutions x and y have the same rank but solution x has large crowding distance than solution y

DHNSGA-II

DHNSGA-II employs pre-selection and Memetic Genetic algorithm on NSGA-II. NSGA-II is not well suited for fine tuning solution which are very close to optimal solution and it does not search in promising region. Pictorial diagram of DHNSGA-II is shown in Fig. 4. It is first seeded with previously generated solution of GA. After that genetic operators are applied to obtain the offspring. Then, multi-objective local search is applied to improve the offspring. At last, non-dominated sorting is applied on off spring, population and improved solutions. The step by step procedure of DHNSGA-II is described in Algorithm 1.

Algorithm 1 (Pseudo code DHNSGA-II):

```
1: Input = G(v, e, x, c)
2: Output = multiple schedule
3: evaluation = 0, seeds = 6;
4: for i = 0 → seeds do
5:     Solution = getPreGeneratedSolution()
6:     evaluatedObjectives(Solution);
7:     Pop = add(Solution);
8:     evaluation++;
9: end for
10: for i = seeds → popSizedo
11:    Solution = generateSolution();
12:    evaluated Objectives(Solution);
13:    Pop = add(Solution);
14:    evaluation++;
15: end for
16: for evaluation → maxEvaldo
17:    Solution Pp[] = Pc[] = Pm[] = new Solution[2];
18:    Solution offSpring[] = new Solution[popSize];
19:    for i= 0 → popSizedo
20:    Pp = selectParents(Pop);
21:    Pc = crossOver(Pp,0.8);
22:    Pm = mutation(Pc,0.3);
23:    evaluatedObjectives(Pm);
24:    evaluation = evaluation+2;
25:    offSpring = add(Pm);
26:    end for
27:    Pop' = localSearch(offSpring);
28:    Pop = nonDominatedSorting
(Pop',offSpring,Pop);
29:    end for
```

In Algorithm 1, the initial population is seeded through GA (lines 2-7). Remaining population is generated randomly and fitness is evaluated (lines 8-13). Off spring, Pp is selected from population Pop using Binary Tournament2 selection (line 18). Offspring Pp generates a new offspring Pc through two-point crossover with probability of 0.8 (line 19). Offspring Pc is mutated through insertion based mutation with probability 0.3 (line 20). Then, newly generate offspring are evaluated to determine their fitness (line 21-22). This process is repeated until the offspring population size is less than popSize. After that simple neighborhood search is applied to newly generated offspring (line 25). Non-dominated sorting among offspring, population and improved solutions is performed to assign different fronts (line 26). This process is repeated until evaluation is less than maxEval.

Implementation of DHNSGA-II: The chromosome is represented using two strings namely matching string and scheduling string. Scheduling string represents the schedule. Matching string represents the task order. Following genetic operators are used.

Chromosome representation: The process of representing a solution that conveys required meaning is necessary. Researchers have used solution string Ss and matching string Ms of length v. Matching string contains the value between 0 to max cluster. Ms[i] = k means the task v_i is assigned to cluster k. Solution string Ssis generated using topology sort of length v. Solution string Ss[i] = k indicates that v_k is ith sub task of the DAG.

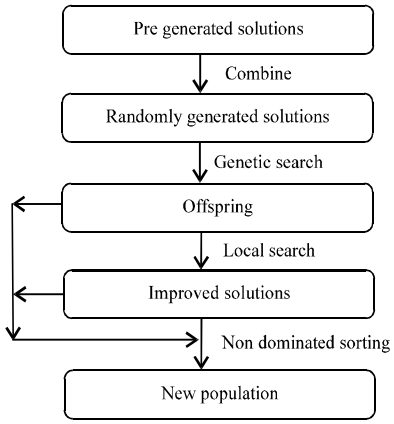

Fig. 4: DHNSGA-II

Selection/replacement: Selection phase is used to allocate reproductive trials to chromosomes according to their fitness. There are different approaches that can be applied during the selection phase. Researchers have used Binary Tournament 2 operator (Deb, 2007) with probability of 0.8. This operator selects the population based on the rank and crowding distance. An individual selected has either its rank lesser (better) than the other or its crowding distance greater than the other.

Crossover: The different crossover operators have been developed for a GA. Researchers have used the most popular two-point crossover operator on the scheduling string. Two-point crossover operator randomly selects two crossover points within a chromosome. Then, interchanges the two-parent chromosomes between these points to produce new offspring. The probability of performing mutations is determined by experimentation and is set to 0.8.

Mutation: Mutation is a background operator which produces spontaneous random changes in the chromosome. A simple way to achieve a mutation would be to alter one or more genes. Researchers have used the insert mutation with probability 0.3 on the scheduling string. Insertion picks two random values and moves the second value to follow the first in scheduling string. It preserves most of the order and the adjacency information. The probability of performing mutations was determined by experimentation and set to 0.3.

Pre-selection: Pre-selection seeds the population with pre generated solutions of GA. Researchers have seeded through three Genetic algorithms. Fitness function of GAs minimizes the makespan time, computation cost and network cost. Crossover, mutation, selection and number of evaluations of NSGA-II are used in seeded GA.

Local search: This algorithm selects current solution c from offspring and searches its neighborhood solution c^0 using local search operator namely move and swap mutation. Move mutation randomly moves a task from one cluster to other cluster. Swap mutation exchanges tasks between two clusters. Utility 11 is computed of c and c^0. The utility is calculated for mxobjectives. Here, three weights are randomly generated for each iteration of local search such that sums of weights are one. Add the c^0 into temporary population, if it has better utility than c. This process is repeated on small number of solutions at random time:

$$u_s = \sum_{i=1}^{m} w_i \frac{f_{in} - min_i}{max_i - min_i} \quad (8)$$

Evaluation: Evaluation of population is performed over three objectives that are described earlier. Non-domination count of solutions are computed in order to find different fronts. If particular front size is more than the remaining population size, it performs crowded comparisons operator for clustering and selects the remaining chromosome.

SIMULATION AND EVALUATION

Simulation has been performed on intel core, i-5, 2.4 GHz, 8 GB Ram processor using jMetal (Durillo and Nebro, 2011) tool kit.jMetal toolkit provides an environment to solve multi-objective optimization problems. To test the effectiveness of the DHNSGA-II real world DAG of Gauss Elimination algorithm is used. Gauss elimination graph is introduced by Topcuouglu *et al.* (2002). Gauss Elimination algorithm finds the upper triangle of square matrix. This application requires matrix size m as an input which should be 2^m.

The total number of tasks in a Gauss elimination graph is equal to $(m^2+m-2)/2$. It requires several input parameters and these are matrix size in the graph, average computation cost (10, 15, 100, 200) computation to communication ratio (0.1, 1, 2, 10) heterogeneity factor (0.2), range of network cost (1-10), computation cost (0.1-0.9), local search operator is performed from (10, 10, 5) times and mutation for local search operator is performed (1, 2, 5). Fitness functions are designed to measure the quality of solutions. Three fitness functions are considered of a schedule.

Makespan: The objective function f_1 of schedule s isdetermined using Eq. 9 where makespan (s) ofa schedule s is calculated using Eq. 2:

$$Minimze(f_1) = Makespan(s) \quad (9)$$

Computation cost: The objective function f_2 iscalculated using Eq. 10 and computation cost of s is calculated using Eq. 3:

$$Minimze(f_2) = Cost(s) \quad (10)$$

Network cost: The objective function f_3 of schedulesis calculated using Eq. 11. Network cost of s iscalculated using Eq. 4:

$$Minimze(f_3) = Network(s) \quad (11)$$

To access the search ability of the proposed algorithm, researchers have generated 16 Gauss elimination graphs of particular matrix that is combination

Table 1: NSGA-II parameters

Parameters name	Values
Population size	100
Cross over rate	0.8
Mutation	0.3
Max evaluation	50000
Crossover operator	Two point crossover
Mutation operator	Insertion based mutation
Selection operator	Binary tournament 2
Local selection operator	Roulette wheel
Local operator	Move and swap
Seeds	6

of different average computation cost and communication ratio. For each graph, ten times network cost and computation cost of resources are generated and three times local search operator is applied. Thus, particular matrix graph is evaluated 480 times. Researchers have compared the seeded NSGA-II and Memetic NSGA-II with DHNSGA-II. Seeded and Memetic is denoted as NSGA-II*, NSGA-II**, respectively. DHNSGAII is denoted as NSGA-II***. NSGA-II* is seeded with GA. Researchers seed two best solutions of each GA. NSGA-II** applies the local search on best solutions. Best solutions are pick out using Roulette wheel selection method. Local search applied on number of solutions at number of times that vary at each iterations (Table 1).

Performance index: Multi-Objective Optimization (MOO) algorithms, measure two parameters, regarding the obtained solution set and reference solution set. It should converge close to the reference solution set and it should maintain diverse solution set. The first condition clearly ensures that the obtained solutions are near optimal and the second condition ensures that a wide range of tradeoff solutions is obtained. Hyper Volume (HV) indicator (Zitzler and Thiele, 1999) is used that computes both convergence and diversity. It computes difference between non-dominated solution set obtained from algorithm and reference solution set. Reference solution set is obtained by merging all of the non-dominated solutions generated by all of the algorithms. The higher value is better for HV. Statistical significance with alpha value (0.05) is computed.

Results of DHNSGA-II: To compare the different Multi-Objective Scheduling algorithm references solution set is obtained by combining the non-dominated solution generated from three algorithms. The reference solution set of different size of matrix is shown from Fig. 5-8. From Fig. 5-8, it can be observed as the size of matrix is increased cost and time is also increased and Pareto front size is also increased.

Figure 9-11 show the comparison between reference solution set and non-dominated solution set obtained

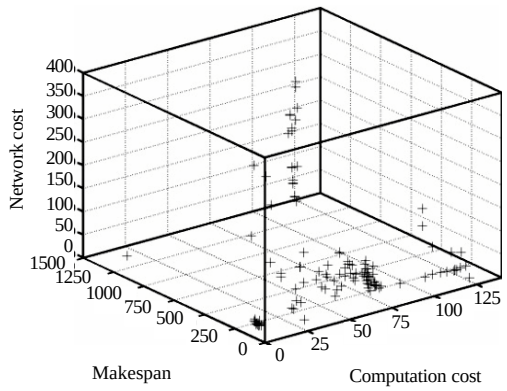

Fig. 5: Pareto front approximation of matrix size 8

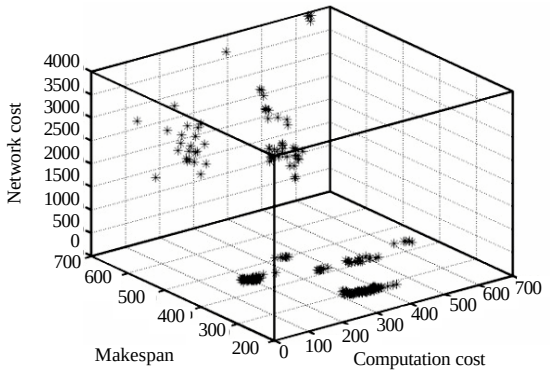

Fig. 6: Pareto front approximation of matrix size 16

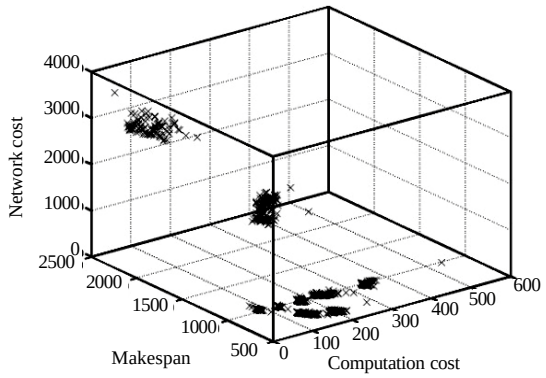

Fig. 7: Pareto front approximation of matrix size 32

from matrix size 16. In Fig. 9-11 blue color shows the reference solution set and red color shows the solution set of different algorithms. Researchers can observe that NSGAII*** get the better spread and convergence with respect to reference solutions set. NSGA-II*** obtained less makespan time, computation cost and network cost.

Figure 12-15 show HV with different size of matrix. From the Fig.12-15, it can be observed that HV value of

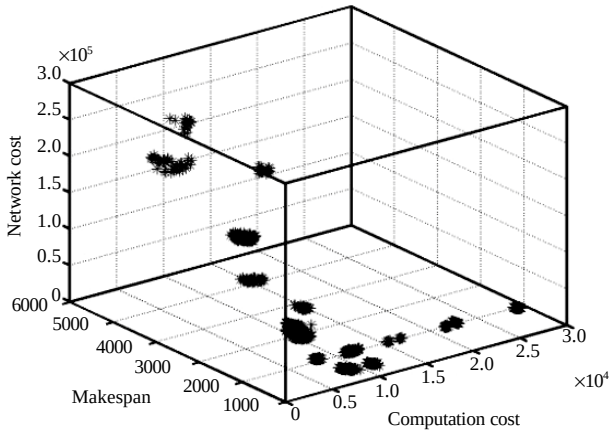

Fig. 8: Pareto front approximation of matrix size 64

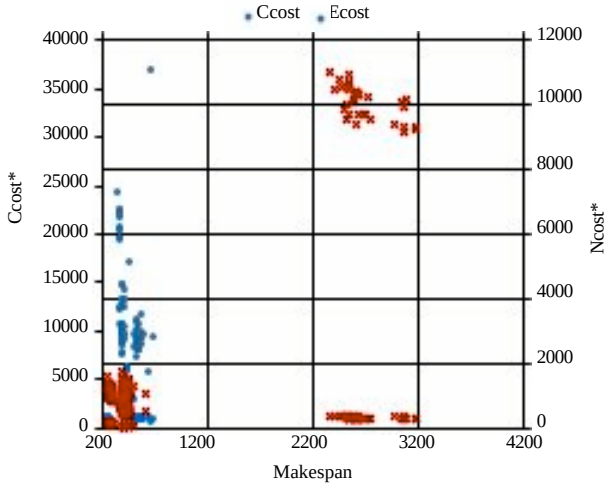

Fig. 9: Comparison of NSGA-II*

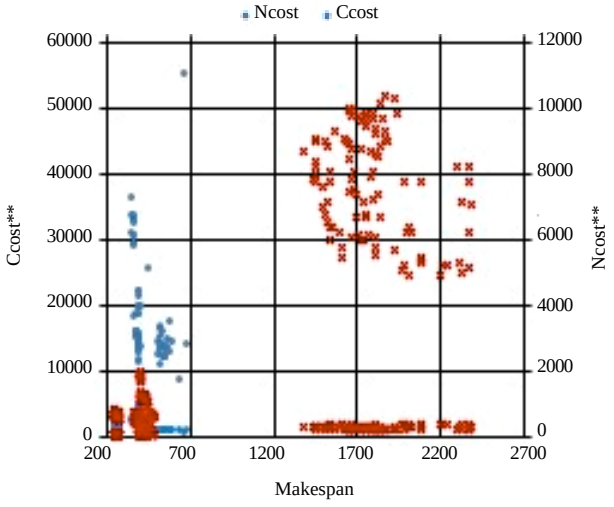

Fig. 10: Comparison of NSGA-II**

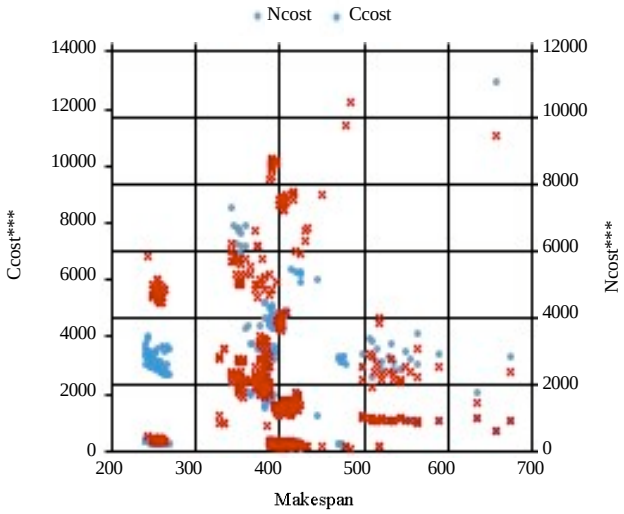

Fig. 11: Comparison of NSGA-II***

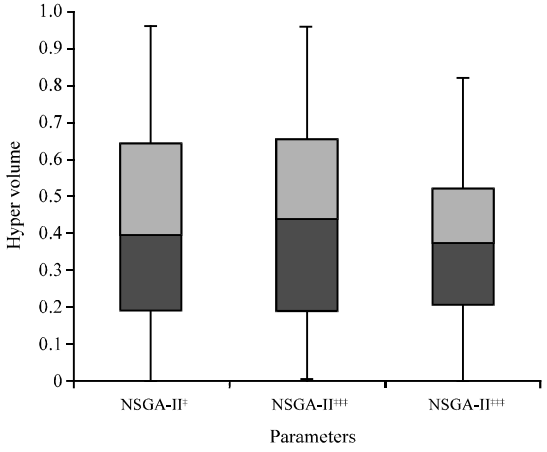

Fig. 12: HV of matrix size 8

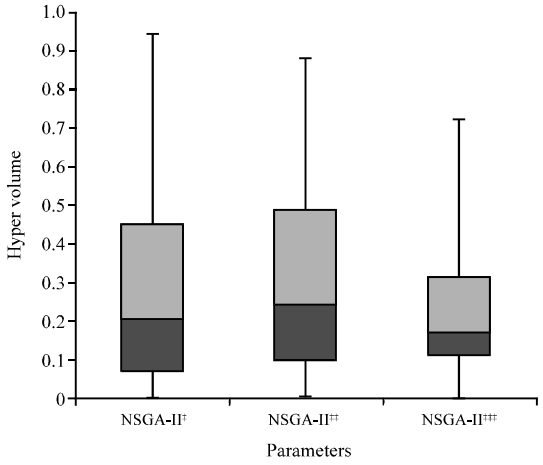

Fig. 13: HV of matrix size 16

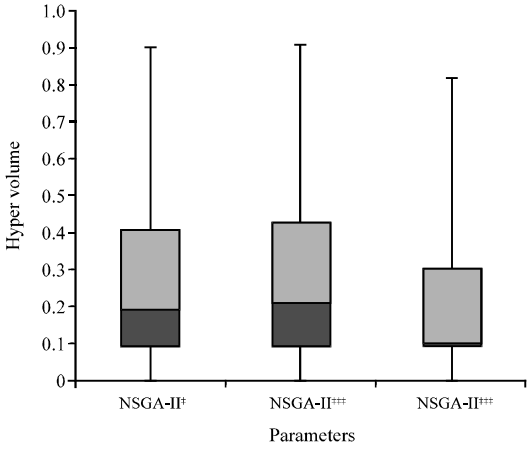

Fig. 14: HV of matrix size 32

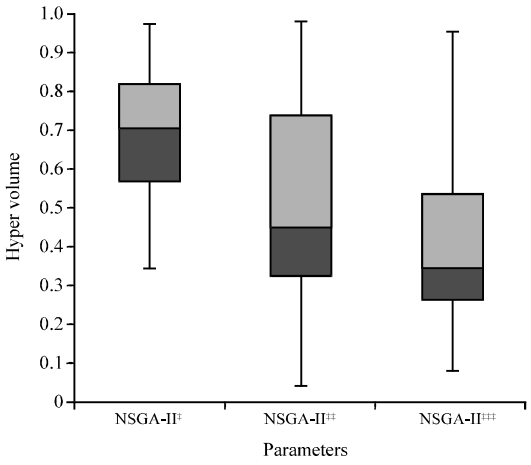

Fig. 15: HV of matrix size 64

NSGA-II*** is lesser than the other versions of NSGA-II. In some case NSGA-II* performs better than NSGA-II**.

RANKING OF NON-DOMINATED SOLUTIONS

A large number of non-dominated solutions are provided by the DHNSGA-II so, the subjective ranking of solutions is very difficult and also will not be precise. In the present investigation, a comprehensive approach has been adopted to rank the non-dominated solutions. Ranking process works in two phases. First phase finds weightage of attributes using entropy theory. Second phase ranks the non-dominated solutions using TOPSIS algorithm. Entropy process is used to measure the weights of various criteria based on information. TOPSIS approach is a practical and useful technique among many famous multiple criteria decision-making methods for ranking and selection of possible alternatives through measuring Euclidean distance. TOPSIS was first developed by Sen and Yang (1998). It is based on the concept that the chosen alternative should not only be nearest to the Positive Ideal Solution (PIS) but should also be farthest from the Negative Ideal Solution (NIS).

Weight determination: Zeleny (1982) proposed the concept of deciding the objective weights based on information entropy that measures the amount of useful information in the data provided. The pseudo code for determining the attribute weights is given in Algorithm 2.

Algorithm 2 (Pseudo code for wieght determination):

1: Input : solution matrix
2: max = findMax(solutionMatrix)
3: min = findMin(solutiomMatrix)
4: Output: weights
5: Transform the different scales and units

$$x'_{ij} = \frac{\max - \text{solutionMatrix}[i][[j]}{\max - \min} \quad \forall\, i=1...n, j=1...m \tag{12}$$

6: Remove insignificance of x_{ij}

$$p_{ij} = \frac{1+x'_{ij}}{\sum_{i=1}^{n}\left[1+x'_{ij}\right]} \forall\, i=1...n, j=1...m \tag{13}$$

7: After normalizing the decision matrix, the entropy value e_j of the set of alternatives for criterion j isdetermined as:

$$e_j = -\frac{1}{\text{Inm}}\sum_{i=1}^{m}\left(p_{ij}\,\text{Inp}_{ij}\right) \forall\, j=1...m \tag{14}$$

Here, m is number of alternatives and j is the number of criterion
8: Degree of divergence of criterion j is calculated

$$d_i = 1 - e_j \;\forall\; j=1...m \tag{15}$$

9: The objective weight for each attribute can be obtained:

$$w_j = \frac{d_j}{\sum_{j=1}^{m} d_j} \tag{16}$$

First it finds the maximum and minimum in solution matrix (lines 2-3). Then, it transforms different scales and units among various attributes into common measureable units to allow the comparison of different attributes using Eq. 12. There are various methods available to derive the normalized matrix. From the experiments researchers have found that Eq. 12 is best suited to the problem. In order toremove the insignificance of normalized matrix Eq. 13 is used to achieve projection matrix given by Zhang *et al*. (2011). Next, projection matrix is used to find the entropy value (e_j) using Eq. 14. Entropy value computes the uncertainty in the projection matrix. After that degree of divergence is calculated using Eq. 15. Here, if the entropy value is high then degree of divergence is less thus this criteria are less important. At last objective weight is obtained using Eq. 16.

Ranking the solutions: TOPSIS (Sen and Yang, 1998) Algorithm is used to rank the various obtained solutions. Ranks are given based on relative closeness. Relative closeness is computed between Positive Ideal Solution (PIS) and Negative Ideal Solutions (NIS). The PIS (A+) is the most preferred alternative while NIS (A-) is the least preferred alternative for the particular criterion. The pseudo code of TOPSIS ranking is given in Algorithm 3.

Algorithm 3 (Pseudo code of TOPSIS):

1: The weighted normalized matrix is obtained by:

$$x_{ij} = w_j \times P_{ij} \tag{17}$$

1. Determine idle A^+ and negative idle solution A^-

$$\begin{aligned} A^+ &= \left\{\left(\max x_{ij}, j \in J\right) \middle| i = 1...n\right\} \\ &= \left\{x^+_i, x^+_{i+1}, ...x^+_n\right\} \end{aligned} \tag{18}$$

$$\begin{aligned} A^- &= \left\{\left(\min x_{ij}, j \in J\right) \middle| i = 1...n\right\} \\ &= \left\{x^-_i, x^-_{i+1}, ...x^-_n\right\} \end{aligned} \tag{19}$$

2. Calculate the separation measures for ideal and negative ideal solutions of each resources as follows:

$$R^+_i = \sqrt{\sum_{i=1}^{n}\left(x_{ij} - x^+_i\right)^2}\; i = 1...m \tag{20}$$

Table 2: Working of ranking algorithm

CompuCost	Makespan	CompuCost	NetCost	Makespan	NetCost	Closeness	Rank
2	3	4	5.000	6.0000	7.0000	8.0000	9
9	174	8	0.952	0.9435	0.9915	0.9650	7
9	174	9	0.952	0.9435	0.9905	0.9646	8
9	181	6	0.952	0.9381	0.9936	0.9653	6
9	181	8	0.952	0.9381	0.9915	0.9646	9
8	183	12	0.960	0.9366	0.9873	0.9676	3
8	187	9	0.960	0.9335	0.9905	0.9684	2
9	187	7	0.952	0.9335	0.9926	0.9646	10
8	195	14	0.960	0.9273	0.9852	0.9661	4
8	201	14	0.960	0.9227	0.9852	0.9657	5
4	305	21	0.992	0.8422	0.9777	0.9750	1

$$R^{-}_{i} = \sqrt{\sum_{i=1}^{n} \left(x_{ij} - x^{-}_{i} \right)^2} \; i = 1 \ldots m \tag{21}$$

3. Calculate relative closeness of each resource to the ideal point as follows:

4. $$C^{+}_{i} = \frac{R^{-}_{i}}{R^{-}_{i} + R^{+}_{i}} = 0 \le c^{+}_{i} \le 1; i = 1 \ldots m \tag{22}$$

5. Rank the resources based on the magnitude of closeness $C(i)^{+}$
6: Rank the schedule based on the magnitude of close-ness.

The weighted normalized matrix is found using Eq. 17 (line-1). Positive and negative ideal solution of each attribute is computed using Eq. 18-19, respectively (line-2). Separation measures from PIS and NIS using Eq. 20-21 is computed (line-3). Relative closeness of separation measures is calculated using Eq. 22 (line-4). After that solutions are ranked based on closeness with one (line-5).

RESULTS OF RANKING ALGORITHM

Table 2 describes the working of the weight determination process. The first three columns show nondominated feasible solutions obtained of matrix size 8 that include the computation cost, makespan time and network cost (column 1-3). Minimum and maximum values of each attribute are bold face (column 1-3). Computation cost, makespan and network cost weights are determined as 0.54, 0.09 and 0.36, respectively. Columns 4-6 show weighted normalized matrix of all solutions. Relative closeness value c_i of each solution is displayed in column 7. Column 8 shows the rank of solutions. It is observed that s10, s6, s5, s8, s9 occupied positions 1, 2, 3, 4, 5 due to their c_i values, i.e., 0.9750, 0.9684, 0.9676, 0.9661, 0.9657 as higher the c_i value, better is the alternative.

The developed approach has been applied to different number of tasks. Top 5 solutions of each case are tabulated in Table 3. The number of non-dominated solutions obtained for all cases vary from 100-2000. The best values for computation cost, makespan time and network cost are boldface among the five top-ranking solutions. The solution may be selected from the top-ranking solutions depending upon the priorities of the user and market scenario.

Table 3: NSGA-II*** top-5 solution

Matrix	CompuCost	Makespan	NetCost	Closeness	Rank
8	1003	101	140	0.6500	5
	308	113	150	0.8100	1
	211	163	110	0.6600	4
	96	174	6	0.6900	2
	94	174	8	0.6800	3
16	268	265	294	0.8351	4
	270	263	315	0.8350	5
	272	262	228	0.8362	2
	273	260	309	0.8357	3
	284	251	246	0.8372	1
32	129	600	1901	0.9392	2
	129	601	1923	0.9389	3
	129	608	2008	0.9368	5
	147	542	2000	0.9439	1
	149	563	1999	0.9374	4
64	7327	1308	11410	0.9361	2
	7343	1310	11303	0.9358	4
	7351	1306	11417	0.9358	3
	7357	1299	11328	0.9363	1
	7360	1308	11231	0.9358	5

CONCLUSION

In the present study, workflow scheduling problem is solved over the utility grid. To the best of the knowledge, this study is also the first attempt to develop the Double Hybrid, Multi-objective, Non-Dominated Sorting Genetic algorithm that is combination of Memtic and Pre-Selection algorithms. The objective of scheduling algorithm are minimizing the computation cost, makespan time and network cost. The multi-objective problem of workflow scheduling is solved using seeded NSGA-II, Memetic NSGA-II and DHNSGA-II. The reference solution set is obtained by combining the Pareto front of all the algorithms. Then, spread and convergence of algorithms are measured along with reference solution set. From the results it is noted that DHNSGA-II gives the satisfactory performance. The DHNSGA-II is further analyzed using TOPSIS to rank the solutions built upon their distance from the best solution and worst solution in this research.

An entropy concept is also used for computing the weights for attributes which is based upon the information theory. Considering the ranking of the solutions, the decision manager may choose a suitable candidate among the top-ranking solutions to justify the objectives defined by the management along with the present market scenario. In future, researchers plan to apply the developed algorithms on grid projects.

REFERENCES

Almond, J. and D. Snelling, 1999. UNICORE: Uniform access to supercomputing as an element of electronic commerce. Future Gener. Comp. Syst., 15: 539-548.

Bajaj, R. and D.P. Agrawal, 2004. Improving scheduling of tasks in a heterogeneous environment. IEEE Trans. Parallel Dist. Syst., 15: 107-118.

Bandyopadhyay, S., S. Saha, U. Maulik and K. Deb, 2008. A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Trans. Evolutionary Comput., 12: 269-283.

Chen, W.N. and J. Zhang, 2009. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans. Syst. Man Cybernet. Part C: Appl. Rev., 39: 29-43.

Chitra, P., P. Venkatesh and R. Rajaram, 2011. Comparison of evolutionary computation algorithms for solving bi-objective task scheduling problem on heterogeneous distributed computing systems. Sadhana, 36: 167-180.

Chunlin, L. and L. Layuan, 2006. A distributed multiple dimensional QoS constrained resource scheduling optimization policy in computational grid. J. Comput. Syst. Sci., 72: 706-726.

Deb, K., 2007. Multi-objective Optimization Using Evolutionary Algorithms. Wiley, New York.

Deelman, E., G. Singh, M. Su, J. Blythe and Y. Gil *et al*., 2005. *Pegasus*: A framework for mapping complex scientific workflows onto distributed systems. Sci. Program. J., 13: 219-237.

Durillo, J.J. and A.J. Nebro, 2011. Metal: A java framework for multi-objective optimization. Adv. Eng. Software, 42: 760-771.

Durillo, J.J., A.J. Nebro, F. Luna and E. Alba, 2009. On the effect of the steady-state selection scheme in multiobjective genetic algorithms. Proceedings of the 5th International Conference Evolutionary Multi-Criterion Optimization, April 7-10, 2009, Nantes, France, pp: 183-197.

Fahringer, T., A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto and H.L. Truong, 2005. Askalon: A tool set for cluster and grid computing. J. Concurrency Computation: Pract. Exp., 17: 143-169.

Foster, I. and C. Kesselman, 1999. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco, CA, USA..

Garg, R. and A.K. Singh, 2011. Multi-objective optimization to workflow grid scheduling using reference point based evolutionary algorithm. Int. J. Comput. Appl. Technol., 22: 44-49.

Garg, S.K., P. Konugurthi and R. Buyya, 2008. A linear programming driven genetic algorithm for meta-scheduling on utility grids. Proceedings of the 16th International Conference on Advanced Computing and Communication, December 14-17, 2008, Chennai, India, pp: 19-26.

Ishibuchi, H., Y. Hitotsuyanagi, Y. Wakamatsu and Y. Nojima, 2010. How to choose solutions for local search in multi-objective combinatorial memetic algorithms. Proceedings of the 11th International Conference on Parallel Problem Solving from Nature, Part I, September 11-15, 2010, Krakow, Poland, pp: 516-525.

Kang, O.H. and D.P. Agrawal, 2000. S3MP: A task duplication based scalable scheduling algorithm for symmetric multiprocessors. Proceedings of the 14th International Parallel and Distributed Processing Symposium, May 1-5, 2000, Cancun, pp: 451-456.

Menasce, D.A. and E. Casalicchio, 2004. A framework for resource allocation in grid computing. Proceedings of the 12th Annual International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, October 5-7, 2004, Volendam, The Netherlands, pp: 259-267.

Nebro, A.J., J.J. Durillo, C.A. Coello, F. Luna and E. Alba, 2008. A study of convergence speed in multi-objective metaheuristics. Proceedings of the 10th International Conference on Parallel Problem Solving from Nature, September 13-17, 2008, Dortmund, Germany, pp: 763-772.

Pandey, S., L. Wu, S. Guru and R. Buyya, 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, April 20-23, 2010, Perth WA, Australia, pp: 400-407.

Sen, P. and J. Yang, 1998. Multiple Criteria Decision Support in Engineering Design. Springer-Verlag, Berlin, Heidelberg.

Siarry, P. and Z. Michalewicz, 2008. Advances in Metaheuristics for Hard Optimization. Springer-Verlag, Berlin, Heidelberg.

Thain, D., T. Tannenbaum and M. Livny, 2005. Distributed computing in practice: The condor experience. J. Concurrency Computation: Pract. Exp., 17: 323-356.

Topcuouglu, H., S. Hariri and M. Wu, 2002. Performance effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst., 13: 260-274.

Wieczorek, M., S. Podlipnig, R. Prodan and T. Fahringer, 2008. Bi-criteria scheduling of scientific workflows for the grid. Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid, May 19-22, 2008, Lyon, pp: 9-16.

Yu, J., 2007. QoS-based scheduling of workflows on global grids. Ph.D. Thesis, Department of Computer Science and Software Engineering, The University of Melbourne, Australia.

Zeleny, M., 1982. Multiple Criteria Decision Making. McGrawHill Book Company, New York, pp: 84-95.

Zhang, H., C.L. Gu, L.W. Gu and Y. Zhang, 2011. The evaluation of tourism destination competitiveness by topsis information entropy-A case in the Yangtze River Delta of China. Tourism Manage., 32: 443-451.

Zitzler, E. and L. Thiele, 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Trans. Evolutionary Comput., 3: 257-271.

Zomaya, A.Y. and Y.H. Tan, 2001. The observations on using genetic algorithms for dynamic load balancing. IEEE Trans. Parallel Distributed Syst., 12: 899-911.