Hybrid Multi-objective Workflow Scheduling on Utility Grids

SUNITA BANSAL[1]
CHITTARANJAN HOTA[2]

[1]Birla Institute of Technology & Science,
Dept. of Computer Science & Information Systems,
Pilani Campus, Pilani, Rajasthan, 333031, INDIA
[2]Birla Institute of Technology & Science, Pilani,
Dept. of Computer Science & Information Systems,
Hyderabad Campus, Hyderabad, AP, 500078, INDIA
[1]sunita_bansal@pilani.bits-pilani.ac.in
[2]hota@hyderabad.bits-pilani.ac.in

Abstract. Workflow scheduling is solved using heuristics and meta heuristics. Heuristics are problem-dependent techniques. Meta heuristics are general purpose method of solving different types of problem. It can be single objective or multiple objectives. This paper focuses on our proposed algorithm named as Double Hybrid NSGA-II Algorithm (DHNSGA-II) that improves up the convergence of the NSGA-II algorithm by employing Pre-selection and Memetic algorithms. DHNSGA-II does hybridization at two levels. At the first level, it uses Pre-selection operator and 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 we 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.

Keywords: NSGA-II, hybrid, ranking, workflow, scheduling.

(Received December 27th, 2013 / Accepted June 19th, 2014)

1 Introduction

Due to advances in wide-area network technologies and low cost of computing resources, Grid computing [8] has 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 services 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 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 problem is to devise new and efficient methods for resource management. Many applications in e-business and e-science is designed as workflow. As a result large number of workflow management tool has been developed [1], [7], [15] to manage the workflow execution.

In the past, researchers have developed heuristics and meta heuristics for workflow scheduling, namely List [16], Cluster [17] and GA [11] scheduling. These scheduling algorithms minimize the makespan. Now a days, user has to pay-per-use thus, he wants that his work should be complete in least time, at least cost. To solve this issue various constraint based heuristics [18] have been developed that gives a single solution but user wants a range of solutions because he has to pay. Thus, various multi objective meta heuristic has been developed in which NSGA-II [4], 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 employes non-dominated sorting without using an external memory, sharing parameter and time complexity of one iteration is $O(mN^2)$, where m is the number of objectives and N is the size of population.

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. Thus, this paper describes and compares the three versions of NSGA-II in order to increase the convergence and diversity of NSGA-II. The first version is seeded NSGA-II. Seeded NSGA-II 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 NSGA-II. Memetic NSGA-II performs neighborhood search, on the best solution that is near to optimal solution. The last version is a combination of Pre-selection and Memetic NSGA-II and denotes as DHNSGA-II. Apart from this, we have also introduced Pareto front ranking, since Pareto front contains many solutions, to pick the best optimal solution; we have ranked the solutions. Rank of solutions is computed using Technique for Order Performance by Similarity to Ideal Solution (TOPSIS) method. TOPSIS [13] algorithm ranks the solution based upon the closeness value of each alternative with reference to the ideal solution.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the problem definition. Section 4 describes DHNSGA-II, our proposal for workflow scheduling. The results are presented and discussed in Section 5. Finally, in Section 6 we conclude the work and suggest future research directions.

2 Related Works

Researchers have also developed constraint based scheduling algorithm using backtracking [18] and iterative programming [3] that minimizes the budget within deadline or vice versa. These algorithms manage to obtain better performance, but less efficient. Various multi-objective meta heuristics [18], [2] have been developed. These heuristics compute a Pareto front on two objectives using simple NSGA-II [4] and do not suggest how to select solutions from Pareto front.

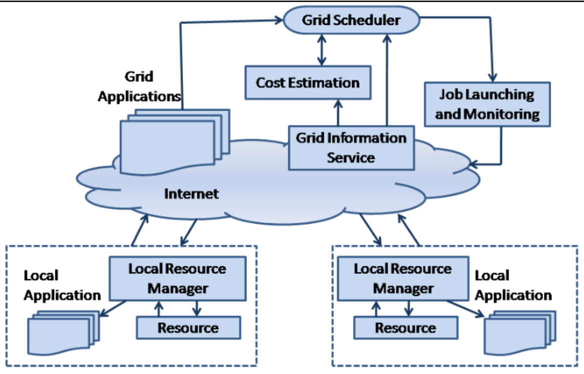


Figure 1: Grid scheduler

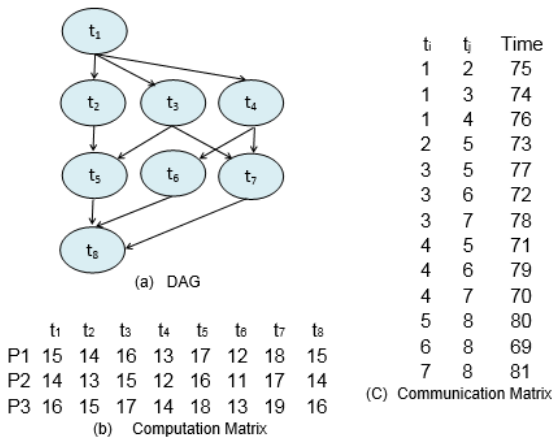
Local search can be applied on first, best or random solutions. It can be applied before genetic operators, during genetic operators or after genetic operators. It can be applied on population, offspring or both. Siarry et al. [14] have tested and suggested that local search applied on best solutions give better results. Similarly, Ishibuchi et al. [10] have tested various multi-objective local search methods on flowshop scheduling. They have found that if local search is applied on few offspring, it gives better performance.

Steady state [6] and Thread based parallel [12] version of NSGA-II are also introduced. Steady state NSGA-II does not use auxiliary population, generates single offspring, performs non-dominated sorting on offspring and population. Thus, its time complexity is more than NSGA-II. Thread based parallel NSGA-II uses multiple threads to compute the objectives of solutions. They have reported thread based parallel NSGA-II does not perform well. Garg et al. [9] have applied Reference point based NSGA-II that uses preference operator rather than crowding operator and it requires user input.

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

3 Problem Definition

This section formulates the grid resource scheduling problem into grid resource market model. It considers the utility Grid that is a collection of heterogeneous clusters and network resources. Clusters are heteroge-



	t1	t2	t3	t4	t5	t6	t7	t8
P1	15	14	16	13	17	12	18	15
P2	14	13	15	12	16	11	17	14
P3	16	15	17	14	18	13	19	16

ti	tj	Time
1	2	75
1	3	74
1	4	76
2	5	73
3	5	77
3	6	72
3	7	78
4	5	71
4	6	79
4	7	70
5	8	80
6	8	69
7	8	81

Figure 2: Workflow of 8 tasks

neous in processor architecture and pricing. Network resources have different speed and cost. We 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 are the sets 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_{ij} 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} is communication time between (t$_i$, t$_j$). Fig. 2a, 2b and 2c show the DAG of 8 tasks, computation matrix and communication matrix respectively.

DHNSGA-II scheduling algorithm minimizes makespan, computation cost and communication cost. It first determines the schedule of workflow using DHNSGA-II that encompass task ordering and cluster identification number on which task will be executed. The objective functions are defined as follows:

Makespan: The makespan is equal to completion time of a workflow. The completion time com$_{ij}$ of a task t_i on the cluster c_j is given by

$$com_{ij} = st_{ij}+x_{ij} \tag{1}$$

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

$$makespan\left(sch\right) = \max\left(com_j\right), \forall\, j = 1 \ldots m \tag{2}$$

Here, com_j is the latest completion time of cluster c_j, m is the number of clusters and sch is schedule of the workflow. Makespan of the workflow is the maximum of com_j.

Computation cost: It occurs when task is executed on a cluster. The computation cost of task t_i on cluster c_j is determined by

$$compuCost_{ij} = compuCost_j * x_{ij} \tag{3}$$

Here, $compuCost_j$ is cluster computation cost in dollars ($), and x_{ij} is computation time of task i on cluster j. Computation cost $compuCost$ of schedule sch is calculated as follows:

$$compuCost(sch) = \sum_{i=1}^{N} compuCost_{ij} \tag{4}$$

Communication cost: It occurs when a cluster transfers result of a task to other clusters. The communication cost is defined as:

$$commuCost_{ij} = commuCost_j * c_{ij} \tag{5}$$

Here, $commuCost_j$ is the communication cost and c_{ij} is the communication time between task i and j. Communication cost is not reciprocal of communication time. Communication cost $commuCost$ of schedule sch is computed as follows:

$$commuCost(sch) = \sum_{i=1}^{N} commuCost_{ij} \tag{6}$$

4 DHNSGA-II

The multi-objective optimization problem consists of a number of objectives and several equality and inequality constraints [4] as shown below:

$$f(x) = f_i\left(x\right),\ f_{i+1}\left(x\right),\ f_n(x),\ i = 1, 2, 3, \ldots.., n \tag{7}$$

$$Subjected\ to\ \ g_i\left(x\right) \geq 0, \quad i = 1, 2, 3, \ldots.., m \tag{8}$$

$$h_i\left(x\right) = 0, \quad i = 1, 2, 3, \ldots.., h \tag{9}$$

Here $f(x)$ is the decision vector representing a feasible solution, i.e. satisfying the m inequality constraints and h equality constraints; f_i is the $_i{}^{th}$ objective function to

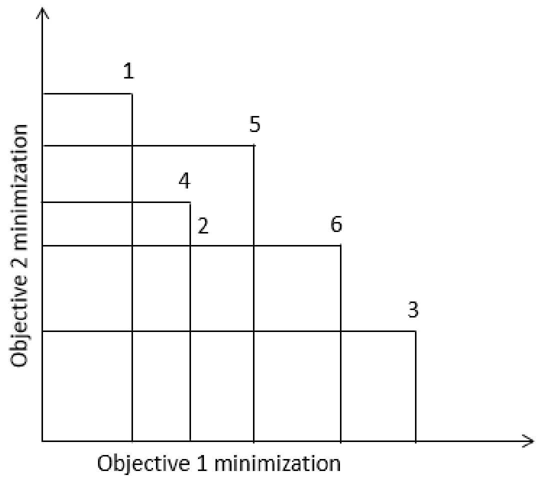


Figure 3: Unconstrained non-dominated sorting

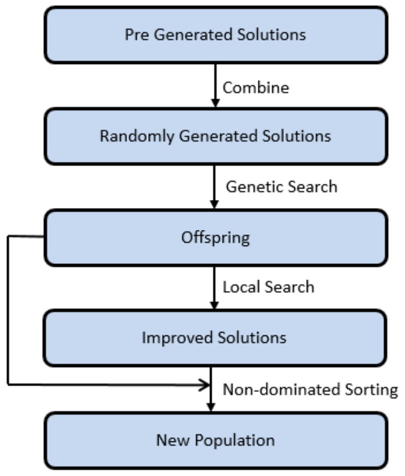


Figure 4: DHNSGA-II

be minimized, n is the number of objective functions, g_i is the inequality constraints and h_i is the equality constraint.

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 the solution y.

The dominating concept is illustrated in 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.

DHNSGA-II is an enhancement of NSGA-II where solutions are seeded and local search is applied on population. Pictorial diagram and Pseudo code of DHNSGA-II is as shown in Fig. 4 and Algorithm 1 respectively. Its, initial population p' is seeded through GA solutions (lines 3 to 4). GA employes selection, crossover and mutation operator similar to DHNSGA-II. Objective values are evaluated (line 5). In order to compute offspring p'', parents p are selected from population p' using Binary Tournament operator (line 7). Two-point crossover and insertion based mutation is applied on parents p (lines 8 to 9). After that newly generated offspring p are evaluated to determine their fitness (line 10) and added into new offspring p'' (line 11). This process is repeated until the size of offspring p'' is less than population size $popSize$. Next, simple neighborhood search is applied to newly generated offspring (line 13). Non-dominated sorting between offspring and population is performed to assign different fronts (line 14). This process is repeated until generation gen is less than number of generation N.

Algorithm 1 Pseudo code DHNSGA-II

```
1:  Input = $G(v, e, x, c)$
2:  Output = multiple schedule
3:  for $gen = 1 \to N$ do
4:      $p' \leftarrow$ seedsWithPreGernatedSolutions();
5:      computeObjectives($p'$);
6:      for $i = 1 \to popSize$ do
7:          $p \leftarrow$ selectTwoParents($p'$);
8:          $p \leftarrow$ performCrossover($p$);
9:          $p \leftarrow$ performMutation($p$);
10:         computeObjectives($p$)
11:         $p'' \leftarrow$ addOffspring($p$);
12:     end for
13:     $p'' \leftarrow$ performLocalSearch($p''$);
14:     $p' \leftarrow$ nonDominatedSorting(p', p");
15: end for
```

4.1 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:

4.1.1 Chromosome Representation

The process of representing a solution that conveys the required meaning is necessary. We 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 Ss is generated using topology sort of length v. Solution string $Ss[i] = k$ indicates that v_k is i^{th} sub task of the DAG.

4.1.2 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. We have used Binary Tournament operator [4]. 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.

4.1.3 Crossover

The different crossover operators have been developed for the GA. We 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 offsprings.

4.1.4 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. 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.

4.1.5 Pre-selection

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

4.1.6 Local Search

This algorithm selects current solution c from offspring and searches its neighborhood solution c' 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. Neighborhood search is applied on small number of solutions at random time. It computes the fitness function using equation 10 of c' that requires minimum, maximum and weight of each objective. Weights are generated randomly for each iteration such that sums of weights are one. c' replace the c in population, if it dominates c.

$$f_c = \sum_{i=1}^{m} w_i \frac{f_{in} - min_i}{max_i - min_i} \tag{10}$$

4.1.7 Evaluation

Evaluation of population is performed over three objectives that are described above. 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 chromosomes.

4.2 Time Complexity

Time complexity of DHNSGA-II is similar to NSGA-II because one iteration of NSGA-II takes $O(mN^2)$, GA one iteration time complexity is $O(N^2)$ and it runs for m objectives thus time complexity of seeded NSGA-II is $O(mN^2)$. Local search is performed on small number of solutions at less time. Thus, time taken is $O(pl)$, where p is number of solutions on which local search is performed, l is the number of times local search is performed on p, m is the number of objectives and N is the population size. Therefore, time complexity of DHNSGA-II is $O(mN^2)$.

5 Simulation and Evaluation

Simulation has been performed on Intel core, i-5, 2.4 GHz, 8GB RAM processor using jMetal [5] 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 as a workflow. Gauss elimination graph is introduced by Topcuouglu et al. [16]. Gauss Elimination algorithm finds the upper triangle of a square matrix. This application requires matrix size of m as an input, that 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 communication cost (1 to 10) and computation cost (0.1 to 0.9).

Local search operator is performed on number of solutions like (10, 10, 5), and mutation is performed on each solution (1, 2, 5) times. Remaining DHNSGA-II parameters are shown in Table 1.

To assess the search capability of the proposed algorithm, we have generated 16 Gauss elimination graphs of a particular matrix, that is combination of different average computation cost and communication ratio. For each graph, ten times communication cost and computation cost of resources are generated and local search operator is applied. Thus particular matrix graph is evaluated 160 times. We have compared the seeded NSGA-II and Memetic NSGA-II with DHNSGA-II. Seeded and Memetic NSGA-II is denoted as NSGA-II*, NSGA-II** respectively. DHNSGA-II is denoted as NSGA-II***. NSGA-II* is seeded with GA. NSGA-II** applies the local search on the best solution. Best solutions are picked out using Roulette wheel selection method. Local search is applied on number of solutions at number of times that vary at each iteration.

Table 1: NSGA-II Parameters

Parameter name	Value
Population size	100
Cross over rate	0.8
Mutation	0.3
Generation	500
Crossover operator	Two point crossover
Mutation operator	Insertion based mutation
Selection operator	BinaryTournament
Local selection operator	Roulette wheel
Local operator	Move and swap

5.1 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 are obtained. Hyper Volume (HV) indicator [19] is used to compute 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.

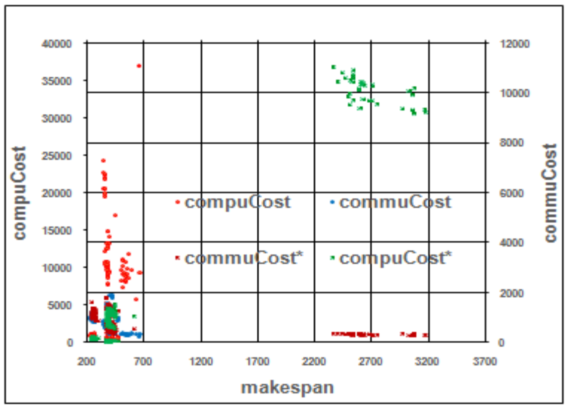


Figure 5: Comparison of NSGA-II*

5.2 Results of DHNSGA-II

To compare different multi-objective scheduling algorithms, reference solution set is obtained by combining the non-dominated solutions generated from three algorithms. Figs. 5 to 7 show the comparison between reference solution set and non-dominated solution set obtained of matrix size 16. In the figures, communication cost (blue color) and computation cost (orange color) show the reference solution set and, communication cost (red color) and computation cost (green color) show the solution set of different algorithms. Reference solutions have least communication cost, computation cost and makespan. From the figures it is clearly visible that NSGA-II*** Fig. 7 has least computation cost, communication cost and makespan. NSGA-II*** has more number of solutions around the reference point while other solutions are scattered. This is because its initial population is seeded and local search is applied. NSGA-II*** has at least 20% less makespan, communication cost and computation cost than other algorithms. From the figures it can be also observed that NSGA-II* performs better than NSGA-II**, because solutions are seeded. Thus, we observe that the pre-selection operator plays a greater role than the local search. NSGA-II* has 15% less objective value than NSGA-II**.

Inter Quartile Range (IQR) value (It measures a variability of data by ignoring outliers of different quartiles) of HV of different size of matrices is summarized in Table 2. Since the lesser value of IQR is better, from the table it can be observed that NSGA-II*** obtains least value than other algorithms. NSGA-II* per-

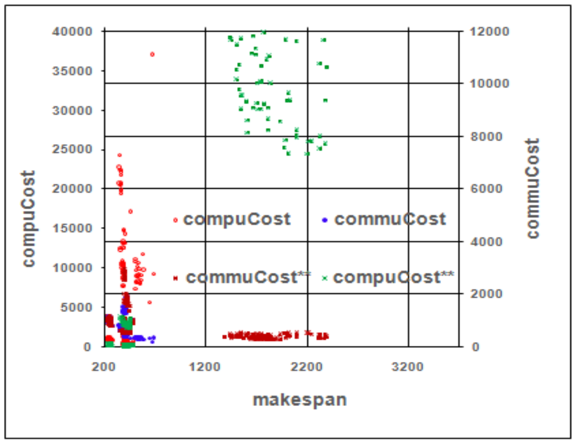


Figure 6: Comparison of NSGA-II**

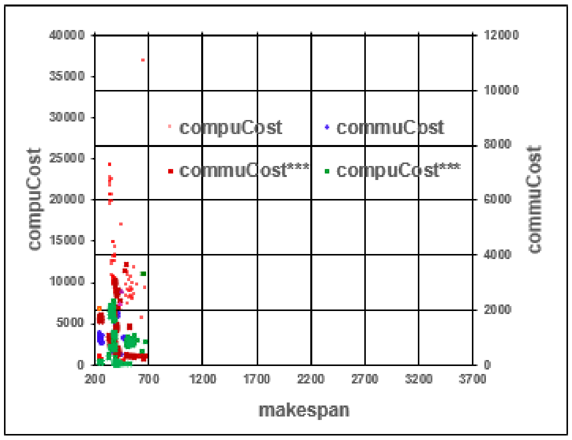


Figure 7: Comparison of NSGA-II***

forms better than NSGA-II**. NSGA-II*** overall average (all matrices) HV value is 0.11 lesser than NSGA-II*. Similarly, NSGA-II* overall average (all matrices) value 0.06 is lesser than NSGA-II**.

Table 2: HV of Matrices

Matrix	NSGA-II*	NSGA-II**	NSGA-II***
8	0.46	0.47	0.32
32	0.38	0.40	0.21
16	0.32	0.34	0.22
64	0.25	0.42	0.23

5.3 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 this work, a comprehensive approach has been adopted to rank the non-dominated solutions. Non-dominated solutions are ranked using TOPSIS algorithm. It gives the rank of a solution based on relative closeness with positive (maximum) and negative (minimum) values of a separation matrix. Separation matrix which is an N-dimensional Euclidean distance of each alternative from the positive solution and negative solutions is computed as follows:

$$\mathrm{R}_i^+ = \sqrt{\sum_{i=1}^{n} \left(x_{ij} - x_i^+\right)^2}, \quad i = 1,..,m \tag{11}$$

$$\mathrm{R}_i^- = \sqrt{\sum_{i=1}^{n} \left(x_{ij} - x_i^-\right)^2}, \quad i = 1,..,m \tag{12}$$

Here, m is the number of objectives, n is the number of solutions, x_{ij} is normalized value of each solution's objective, x_i^+ and x_i^- are the maximum and minimum values of each objective.

The relative closeness of each solution i with ideal solution R_i^- and R_i^+ is measured as shown in equation (13).

$$C_i^+ = \frac{R_i^-}{R_i^- + R_i^+} \quad 0 \le c_i^+ \le 1; i = 1, \ldots, n \tag{13}$$

The solution that has the least value of C_i^+ is considered as the best alternative.

5.4 Results of Ranking Algorithm

The developed approach has been tested on different matrices. Top five solutions of each algorithm are selected by giving equal weight (0.33, 0.33, 0.34). After that ratio of makespan and total cost (communication cost and computation cost) is computed. Matrix of size 16 and 32, top five solutions, objective functions ratio, of each algorithm is tabulated in Table 3 and 4. It can be observed that NSGA-II*** obtained less ratio than other two algorithms and NSGA-II* has less value than NSGA-II**.

6 Discussion

In the present study, workflow scheduling problem is analyzed and solutions are proposed for utility Grid. The proposed scheduling algorithm minimizes the computation cost, makespan and communication cost. The multi-objective problem of workflow scheduling is solved using seeded NSGA-II, Memetic NSGA-II, and

Table 3: Ratio of Makespan and Total Cost of Matrix 16

No.	NSGA-II*	NSGA-II**	NSGA-II***
1	11.24	13.96	3.82
2	12.31	13.80	4.23
3	11.22	12.80	4.35
4	11.71	13.58	4.38
5	11.47	11.67	4.14

Table 4: Ratio of Makespan and Total Cost of Matrix 32

No.	NSGA-II*	NSGA-II**	NSGA-II***
1	17.90	23.77	8.14
2	17.81	20.64	8.17
3	17.85	21.88	7.99
4	18.09	20.87	7.90
5	18.68	23.19	8.20

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. 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 the current study neighborhood search is used as local search. In future, authors plan to apply other local search techniques like archived multi-objective based simulated annealing to improve the results.

References

[1] Almond, J. and Snelling, D. Unicore: Uniform access to supercomputing as an element of electronic commerce. volume 15, pages 539–548. Future Generation Computer Systems, NH-Elsevier, 1999.

[2] Chitra, P., Venkatesh, P., and Rajaram, R. Comparison of evolutionary computation algorithms for solving bi-objective task scheduling problem. volume 36, pages 167–180. Sadhana, Indian Academy of Science, 2011.

[3] Chunlin, L. and Layuan, L. A distributed multiple dimensional qos constrained resource scheduling optimization policy in computational grid. volume 72, pages 706–726. Journal of Computer and System Sciences, 2006.

[4] Deb, K. *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, New York, 2007.

[5] Durillo, J. J. and Nebro, A. J. jmetal: A java framework for multi-objective optimization, advances in engineering software. volume 42, pages 760–771. Elsevier, 2011.

[6] Durillo, J. J., Nebro, A. J., Luna, F., and Alba, E. On the effect of the steady-state selection scheme in multi-objective genetic algorithms. pages 183–197. 5th International Conference Evolutionary Multi-Criterion Optimization, Nantes, France, 2009.

[7] Fahringer, T., Jugravu, A., Pllana, S., Prodan, R., Junior, C. S., and Truong, H. Askalon: A tool set for cluster and grid computing. volume 17, pages 143–169. Journal Concurrency and Computation: Practice & Experience- Grid Performance, 2005.

[8] Foster, I. and Kesselman, C. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, USA, 1999.

[9] Garg, R. and Singh, A. K. Multi-objective optimization to workflow grid scheduling using reference point based evolutionary algorithm. volume 22, pages 44–49. International Journal of Computer Applications, 2011.

[10] Ishibuchi, H., Hitotsuyanagi, Y., Wakamatsu, Y., and Nojima, Y. How to choose solutions for local search in multiobjective combinatorial memetic algorithms. pages 516–525. Lecture Notes in Computer Science Springer, Berlin, Parallel Problem Solving from Nature - PPSN XI, part I, 2010.

[11] Kang, O. and Agrawal, D. P. A task duplication based scalable scheduling algorithm for symmetric multiprocessors. pages 451–456. 14th International Parallel and Distributed Processing Symposium, 2000.

[12] Nebro, A. J., Durillo, J. J., Coello, C. A., Luna, F., and Alba, E. A study of convergence speed in multi-objective metaheuristics. pages 763–772. 10th International Conference Parallel Problem Solving from Nature, Dortmund, Germany, 2008.

[13] Sen, P. and Yang, J. *Multiple Criteria Decision Support in Engineering Design*. Springer-Verlag, Berlin Heidelberg, New York, 1998.

[14] Siarry, P. and Michalewicz, Z. *Advances in Metaheuristics for Hard Optimization*. Springer-Verlag Berlin Heidelberg,New York, 2008.

[15] Thain, D., Tannenbaum, T., and Livny, M. Distributed computing in practice: The condor experience. volume 17, pages 323–356. Journal Concurrency and Computation: Practice and Experience, 2005.

[16] Topcuouglu, H., Hariri, S., and Wu, M. Performance-effective and low-complexity task scheduling for heterogeneous computing. volume 13, pages 260–274. IEEE Transaction on Parallel and Distributed Systems, 2002.

[17] Wieczorek, M., Podlipnig, S., Prodan, R., and Fahringer, T. Bi-criteria scheduling of scientific workflows for the grid. pages 9–16. Eighth IEEE International Symposium on Cluster Computing and the Grid,Lyon, 2008.

[18] Yu, J. *QoS-based Scheduling of Workflows on Global Grids*. PhD thesis, Department of Computer Science and Software Engineering, The University of Melbourne, Australia., 2007.

[19] Zitzler, E. and Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. volume 3, pages 257–271. IEEE Transactions on Evolutionary Computation, 1999.