Int. J. Advance. Soft Comput. Appl., Vol. 3, No. 1, March 2011
ISSN 2074-8523; Copyright © ICSRS Publication, 2011
www.i-csrs.org

Non Dominated Particle Swarm Optimization For Scheduling Independent Tasks On Heterogeneous Distributed Environments

G. Subashini, M.C. Bhuvaneswari

Department of Information Technology, PSG College of Technology, India
e-mail: suba@ity.psgtech.ac.in

Department of Electrical and Electronics Engineering,
PSG College of Technology, India
e-mail: mcb@eee.psgtech.ac.in

Abstract

Scheduling tasks is one of the core steps to effectively exploit the capabilities of distributed or parallel computing systems. In general, scheduling is an NP-hard problem. Most existing approaches for scheduling deal with a single objective only. This paper presents a multi-objective scheduling algorithm based on particle swarm optimization (PSO). In this paper a non-dominated sorting particle swarm optimization (NSPSO) that combines the operations of NSGA –II is used to schedule tasks in a heterogeneous environment. The approach aims at developing optimal schedules thereby minimizing two objectives, makespan and flowtime simultaneously. The experimental results indicate that NSPSO obtains good solutions on benchmark instances in comparison with a multi-objective particle swarm optimizer using a weighted approach (W-MOPSO)which is also implemented in this paper for effective comparisons.

Keywords: *Heterogeneous system, Non-dominated Sorting, Particle swarm Optimization.*

1 Introduction

Distributed computing systems have emerged as a powerful platform to perform different computationally intensive applications that have various computational

requirements. The problem of scheduling independent computational jobs in a distributed computing environment is important to exploit the different capabilities of a set of heterogeneous resources and satisfy users with high expectations for their applications. A static task scheduling algorithm [1] can be used in such a heterogeneous system which provides a variety of architectural capabilities, orchestrated to perform on application problems whose tasks have diverse execution requirements. Static scheduling may be useful for analysis of heterogeneous computing systems, to work out the effect of resource failures. Schedulers can be implemented using complex algorithmic methods that utilize the known properties of a given application and the available environment. However, finding optimal schedules in such a system has been shown, in general, to be NP-hard [2] and therefore the use of heuristics is one of the suitable approaches.

Different criteria can be used for evaluating the performance of scheduling algorithms and the most important of which are makespan and flowtime. Makespan is the time when grid finishes the latest job and flowtime is the sum of finalization times of all the jobs. An optimal schedule will be the one that optimizes the flowtime and makespan [3]. The conceptually obvious rule to minimize flowtime is to schedule Shortest Job on the Fastest Resource (SJFR). Minimizing makespan is to schedule the Longest Job on the Fastest Resource (LJFR). Minimizing flowtime asks the average job finishes quickly, at the expense of the largest job taking a long time, whereas minimizing makespan asks that no job takes too long, at the expense of most jobs taking a long time. Minimization of makespan thus results in maximization of flowtime. This requires the problem to be formulated as a multi-objective optimization problem as it deals with conflicting objectives.

PSO is a collaborative population-based search model [4–6]. This has been applied successfully to a number of problems and its use is rapidly increasing. A PSO algorithm contains a swarm of particles, each particle representing a potential solution. The particles fly through a multidimensional search space in which the position of each particle is adjusted according to its own experience and the experience of its neighbors. PSO has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration and exploitation abilities within a short calculation time. These characteristics make PSO highly viable to be used for solving also multi-objective optimization problems.

The non-dominated sorting PSO (NSPSO) of Li [7] is used in this paper due to its performances obtained against a set of well known difficult test functions. NSPSO combines the fast ranking of non-dominated solutions, crowding distance ranking and an elitist strategy of combining parent population and offspring population. The performance of NSPSO has been tested using the benchmark of Braun et al. [8], which is known to be the most difficult benchmark for static instances of the problem. It consists of instances that try to capture the high degree of heterogeneity of resources and workload of tasks.

The remainder of the paper is organized as follows: Section 2 reviews related algorithms for task scheduling problem. Section 3 presents a brief introduction of multi-objective optimization problems. The problem formulation is given in Section 4. Section 5 describes the scheduling method through W-MOPSO and NSPSO. Experimental results are reported in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Optimal mapping of independent computational tasks to available machines in a distributed computing system is a NP- hard problem as stated earlier and as such, it is a subject to various heuristic and meta-heuristic algorithms. The heuristics applied to the task scheduling problem include Sufferage [9], min-min, max-min [10], LJFR-SJFR [11], min-max [12], etc. The most popular of meta-heuristic algorithms are genetic algorithm [13], simulated annealing [15], ant colony optimization [14] and particle swarm optimization [16] . Braun *et al*. [8] described eleven heuristics and compared them on different types of HC environments which illustrates that the GA scheduler can obtain better results in comparison with others. The above referred heuristics and meta-heuristics aimed at minimizing a single criteria, the makespan of the schedule.

Different criteria can be used for evaluating the efficacy of scheduling algorithms. Few attempts have been made to optimize multiple criteria. [12] investigates the efficacy of five popular heuristics for minimizing makespan and flowtime on HC environments with various characteristics of both machines and tasks. However the objectives are evaluated separately here. Xhafa *et al*. [17] used Genetic Algorithm-based schedulers for computational grids and most of GA operators are implemented and compared to find the best GA scheduler for this problem. Abraham *et al*. [18] illustrated the usage of several nature inspired meta-heuristics (SA, GA, PSO, and ACO) for scheduling jobs in computational grids using single and multi-objective optimization approaches. Abraham *et al*. [19] used a fuzzy particle swarm optimization and Izakian *et al*. [20] used a discrete version of particle swarm optimization for scheduling problem. These methods combine the multiple objectives into a scalar cost function, ultimately making the problem single-objective prior to optimization. In practice, it can be very difficult to precisely and accurately select these weights as small perturbations in the weights can lead to very different solutions.

Hence, in this paper NSPSO that adapts the approach of determining an entire Pareto optimal solution set or a representative subset is implemented for the problem. Pareto optimal solution sets are often preferred to single solutions when considering real-life problems, since the final solution of the decision maker is always a trade-off between crucial parameters [21].

3 Multi-Objective Optimization

Many optimization problems in the world involve the optimization of several objectives at the same time. Generally, these functions are non-commensurable and often conflicting objectives. Multi-objective optimization with such conflicting objective functions gives rise to a set of optimal solutions, instead of one optimal solution. The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all objective functions. These optimal solutions are known as Pareto-optimal solutions. In general, the multi-objective minimization problem can be formulated as

$$\text{Minimize } z = (f1(x), f2(x) \ldots\ldots fm(x)) \quad \text{subject to } x \in X \tag{1}$$

where $x = [x_1, x_2, \ldots, x_n]$ is the vector of decision variables, $f_i : \Re^n \rightarrow \Re$, i = 1, 2,….m , are the objective functions and $X \subset \Re^n$ is the feasible region in the decision space. A solution $x \in X$ is said to dominate another solution $y \in X$ (denoted as $x \prec y$) if the following two conditions are satisfied:

$$\begin{aligned} &\forall i \in \{1,2,.....m\},\ f_i(x) \leq f_i(y) \\ &\exists i \in \{1,2,.....m\},\ f_i(x) < f_i(y) \end{aligned} \tag{2}$$

If there is no solution which dominates $x \in X$, x is said to be a Pareto Optimal Solution. The set of all elements of the search space that are not dominated by any other element is called the Pareto Optimal Front of the multi-objective problem which represents the best possible solution with respect to the contradictory objectives. A multi-objective optimization problem is solved, when its complete Pareto Optimal Solution is found.

4 Problem Formulation

A distributed computing system is composed of computing resources where these resources can be a single PC, a cluster of workstations or a supercomputer. Let $T = \{T_1, T_2, \ldots, T_n\}$ denote the set of tasks that are independent of each other to be scheduled on m processors $P = \{P_1, P_2, \ldots, P_m\}$. It is also assumed that the tasks are non pre-emptive. As the scheduling is performed statically the time required to perform each task can be estimated. The required time for executing a task in a processor is contained in an Expected Time to Compute (ETC) matrix. An ETC matrix is a *n x m* matrix, where each position ETC[n][m] indicates the expected time to compute task n in processor m. One row of the ETC matrix contains the estimated execution time for a given task on each machine. Similarly one column of the ETC matrix consists of the estimated execution time of a given machine for each task.

An optimal schedule is one that optimizes flowtime and makespan that is defined as

$$\text{makespan: } \min_{S_i \in Sched} \{\max_{j \in Jobs} F_j\} \quad (3)$$

$$\text{flowtime : } \min_{S_i \in Sched} \{\sum_{j \in Jobs} F_j\} \quad (4)$$

where *Sched* is the set of all possible schedules, *Jobs* stands for the set of all tasks and F_j represents the time in which job *j* finalizes. To formulate the objective, $C_{i,j}(i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, m\})$ is defined as the completion time for finishing the task T_i on processor P_j and *Wi* ($i \in 1, 2, \ldots, m$) is the previous workload of P*i*, then $\sum(Ci + Wi)$ is the time required for processor P*i* to complete the tasks assigned to it. Hence makespan and flowtime can be evaluated as

$$\text{makespan : } \min_{i \in 1 \ldots m} \{\sum C_i + W_i\} \quad (5)$$

$$\text{flowtime : } \sum_{i=1}^{m} C_i \quad (6)$$

Minimizing makespan aims to execute the whole meta-task as fast as possible while minimizing flowtime aims to utilize the computing environment efficiently. Conceptually minimizing flowtime is to schedule Shortest Job on the Fastest Resource (SJFR) and minimizing makespan is to schedule the Longest Job on the Fastest Resource (LJFR). Minimization of makespan thus results in maximization of flowtime, thus making the problem multi-objective.

5 Multi–Objective Particle Swarm Optimization for Task Scheduling

PSO is relatively a recent stochastic heuristic introduced by Eberhart and Kennedy [4]. it is based on the analogy of swarm of bird and school of fish [4]. A swarm consisting of a population of birds/fishes flocks synchronously, changes direction suddenly, scatters and regroups iteratively, and finally perches on a target, in order to escape from enemies or search for food. The PSO mimics the interesting behavior and serves as a function optimizer. It keeps a population of individuals which are potential solutions to the optimization problem. By taking advantage of individual cognition and social interaction, the swarm improves the solutions iteratively and eventually converges to the optimal solution.

As stated earlier, there are two approaches to solve the MOP. One approach is the classical weighted-sum approach where the objective function is formulated as a weighted sum of the objectives. But the problem lies in the correct selection of the weights or utility functions to characterize the decision-makers preferences. The second approach called Pareto-optimal solution have no unique or perfect solution,

but a set of non-dominated, alternative solutions, known as the Pareto-optimal set. The two methods of solving the task scheduling problem are described below.

5.1 Weighted Multi-objective Particle Swarm Optimization (W-MOPSO)

5.1.1 Particle Representation

One of the key issues in designing a successful PSO algorithm is the representation step, i.e. finding a suitable mapping between problem solution and PSO particle. The representation of the particle vector should be compact and simple For the considered problem, the i^{th} particle is represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{ir})^T$ where pi,j indicates the index of the allocated processor for the jth module. The PSO swarm consists of N particles and the initial swarm is generated at random. The swarm particles iteratively improve their solution quality based on personal cognition and social interaction by the particle movement formula.

5.1.2 Particle Evaluation

The fitness function measures to what extent the particle solution S satisfies the objective of the optimization problem. The function *fit*(*S*) is a sum of two objectives, the makespan of schedule *S* and mean flowtime of schedule *S* as given by Equation (5) and (6). The mean flowtime of the schedule is the flowtime divided by number of processors *p*. The mean flowtime is taken in order to keep both the objectives in approximately the same magnitude. The influence of makespan and flowtime in *fit*(*S*) is parameterized by the weights, where weights w_1 and w_2 are chosen such that $w_2=1-w_1$ and $w_1+w_2=1$. Hence particle evaluation is done as given by Equation (7)

$$\text{fit (S)} = w_1 . \text{ makespan} + w_2 . \text{ mean flowtime} \quad (7)$$

5.1.3 Particle Movement

The particle position is updated during each iteration based on two types of experiences: personal best and global best experiences. This is a positive feedback process which increases the probability for targeting the optimal solution. The personal best experience, denoted by $pbest_i$, is the experienced position by particle P_i which receives the highest fitness value during flying. gbest represents the best particle found in the entire population each generation. At each iteration, the particle P_i modifies its velocity v_{ij} and position p_{ij} through each dimension j by referring to $pbest_i$ and the swarm's best experience gbest using Equation (8) and (9)

$$V_{ij} = WV_{ij} + c1rand1(\,)\,(\,pbest_i - p_{ij}\,) + c2rand2(\,)\,(gbest - p_{ij}\,) \quad (8)$$

$$P_{ij} = P_{ij} + V_{ij} \quad (9)$$

where c1 and c2 are the cognitive and interaction coefficients, rand1 and rand2 are random real numbers drawn from U(0, 1). The inertia weight, W is a user-specified parameter that controls the momentum of the particle. A larger inertia weight pressures towards global exploration while a smaller inertia weight pressures toward fine-tuning the current search area. The following weighting function is usually utilized:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} * iter \quad (10)$$

where,

- wmax: initial weight,
- wmin: final weight,
- itermax : maximum iteration number,
- iter: current iteration number.

At each iteration, the PSO flies each particle through the solution space using Equations (8) and (9). The particles learn through the personal cognition ($pbest_i$) and the social interaction (gbest) .They explore new areas with the random multipliers (rand1 and rand2) to escape from the barrier of the local optimality. When the algorithm is terminated with a given maximum number of iterations, the best experienced position by the entire swarm is reported as the final solution. The pseudo code of PSO algorithm for task scheduling in grid computing system is given as follows,

```
begin
    Initialize population randomly;
    Initialize each particle position vector and velocity vector;
    Initialize parameters;
    Evaluate each particle using combined fitness and find the personal best
    and the global best;
    repeat
        for each particle i=1,…,N do
```

```
            Update each particle's velocity and position;
            Evaluate each particle and update the personal best and the global
            best;
        end
    until termination
end
```

5.2 Non-dominated Sorting Particle Swarm Optimization (NSPSO)

NSPSO extends the basic form of PSO by making a better use of particles' personal bests and offspring for effective non-domination comparisons. The problem with the basic form PSO is that dominance comparisons are not fully utilized in the process of updating the personal best of each particle. To overcome this problem and increase the sharing level between particles in the swarm, NSPSO combine the entire population of N pbest and N of these particles' offspring to form a temporary population of $2N$ particles. Then, domination comparisons among all the $2N$ individuals are carried out. By comparing the combined $2N$ particles for non-domination relationships, the entire population is sorted into different non-domination fronts as used in NSGA II. Each individual in each front is assigned a rank based on front in which they belong to. Individuals in the first front are given a fitness value of 1 and individuals in second are assigned a rank of 2 and so on. In addition to the rank, a new parameter called crowding distance is calculated for each individual to ensure the best distribution of the non-dominated solutions. The crowding distance is a measure of how close an individual is to its neighbors. The global best $gbest_i$ for the i^{th} particle P_i is selected randomly from the top part of the first front. N particles are selected based on fitness and the crowding distance to play the role of pbest. Equation (8) uses the above information to calculate the new updated velocity for each particle in the next iteration step. Equation (9) updates each particle's position in the search space.

The steps of basic NSPSO algorithm is summarized below

1. Initialize the population P_t. Each particle in the swarm is initialized randomly within the specified limits. The initial velocity for each particle is set to zero. The personal best position $Pbest_i$, is set to X_i .

2. Evaluate each particle in the population,

3. Apply non-dominated sorting on the particles.

4. Calculate crowding distance of each particle.

5. Sort the solutions based on crowding distance.

6. Select randomly gbest for each particle from a specified top part (e.g. top 5%) of the first front *F*1;

7. Calculate the new velocity v_{t+1} based on Equation (8) and new position P_{t+1} from Equation (9) using the determined gbest and pbest.

8. Create a new population of size 2N by combining the new position and their pbest, $Pt_{+1} \cup Pbest_{t}$.

9. Apply non-dominated sorting on 2N particles and calculate the crowding distance for each particle.

10. Generate a new set of N solutions by selecting solutions from non-dominated fronts F1, F2 and so on using the crowding distance. The N solutions form the pbest for the next iteration.

11. Go to step 6 till the termination criteria is met.

6 Experimental Results And Discussion

In this section, the proposed NSPSO algorithm based on Pareto-optimal approach is implemented. To assess the performance W-MOPSO algorithm that uses a weighted approach is also implemented. Both the algorithms were implemented in Linux using C to analyze their comparative performances. The experimental parameter settings of the competing algorithms are set as follows.

Table 1:Parameters for W-MOPSO and NSPSO

Population size	Number of generation	W_{max}	W_{min}	c1	c2
100	1000	0.9	0.4	2	2

In addition to the above parameters W-MOPSO uses weights w1 and w2 in Equation (7) set to 0.5 and 0.5 respectively.

This setting considers both objectives equally important.

Since both the algorithms are stochastic based, each independent run of the same algorithm on a particular problem instance may yield a different result. To make a fair comparison of the algorithms each experiment was repeated 10 times with different random seeds and the average of the results are reported in the case of W-MOPSO and the best solutions are considered with respect to NSPSO.

6.1 Simulation Model

To assess the comparative performances of the algorithms, the simulation model in [10] based on expected time to compute (ETC) matrix for 512 tasks and 16 processors is used. To realistically simulate possible heterogeneous environments, different types of ETC matrix according to three metrics: task heterogeneity, machine heterogeneity and consistency are simulated.. The task heterogeneity is defined as the amount of variance possible among the execution times of the jobs with two possible values low and high. Machine heterogeneity is the variation of the running time of a particular job across all the processors, which can be high and low. To capture other possible features of real scheduling problems, three different ETC consistencies namely consistent, inconsistent and semi-consistent are used. An ETC matrix is considered consistent if a processor P_i executes task t faster than processor P_j , then P_i executes all the jobs faster than P_j . Inconsistency means that a processor is faster for some jobs and slower for some others. An ETC matrix is considered semi-consistent if it contains a consistent sub-matrix. A semi consistent ETC matrix is characterized by an inconsistent matrix which has a consistent sub-matrix of a predefined size.

Thus 12 distinct types of ETC matrix can be generated considering the different combinations. The matrices used here a randomly generated as described. Initially a $m \times 1$ baseline column vector B is generated by repeatedly selecting m uniform random floating point values between 1 and ϕ_b, the upper bound on values in B. Then the ETC matrix is constructed by taking each value B(i) in B and multiplying it by a uniform random number $x_r^{i,k}$ which has an upper bound of ϕ_r. Each row in the ETC matrix is then given by $B(i) \times x_r^{i,k}$. The vector B is not used in the actual matrix. This process is repeated for each row until the $m \times n$ matrix is full. Therefore, any given value in the ETC matrix is within the range $(1, \phi_b , \phi_r)$. Different task and machine heterogeneities described above are modeled by using different baseline values. High task heterogeneity was represented by ϕ_b =3000 and low task heterogeneity used ϕ_b=100. High machine heterogeneity was represented by ϕ_r=1000 and low machine heterogeneity was modeled using ϕ_r =10. To model a consistent matrix each row in the matrix was sorted independently, with processor P_1 always being the fastest, and P_m being the slowest. Inconsistent matrices are left in the random state in which they are generated. Semi-consistent matrices are generated by extracting the row elements {0, 2, 4 . . .} of each row i, sorting them and then replacing in order, while the elements {1, 3, 5, . . .} are left in their original order, this means that the even columns are consistent while the odd columns are inconsistent.

6.2 Comparative Performances

Both W-MOPSO and NSPSO algorithms are applied on all 12 problem instances and the results plotted in Fig 1- 4. The plots show the non-dominated solutions in Rank1 using NSPSO and the optimal solution in the case of W-MOPSO. Both makespan and mean flowtime are measured in same time units and scaled to ten thousands of units in the plotted results. In the results the different problem instances are identified according to the following scheme: u- x- yy-zz, where

u means uniform distribution

x denotes the type of consistency (c–consistent, i–inconsistent and s means semi-consistent).

yy indicates the heterogeneity of the jobs (hi–high, and lo–low).

zz indicates the heterogeneity of the resources (hi–high, and lo–low).

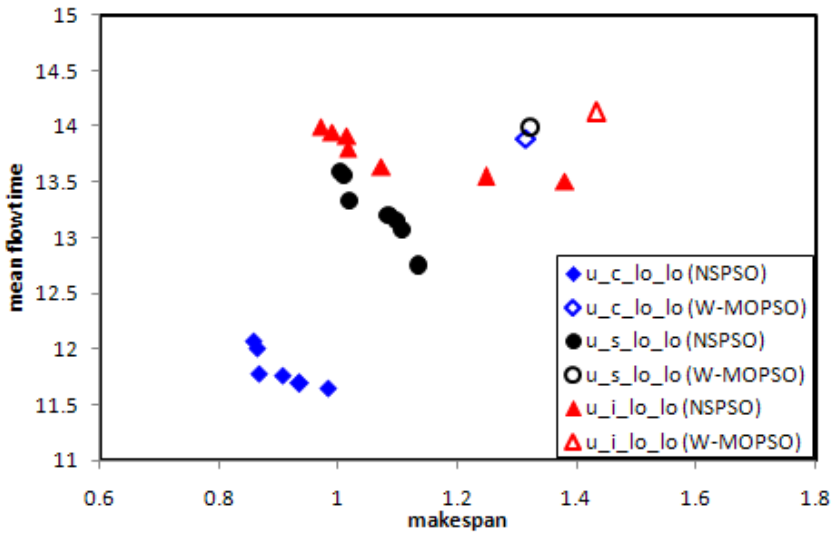

Fig.1: Performance comparison of NSPSO and W-MOPSO for low task, low machine heterogeneity

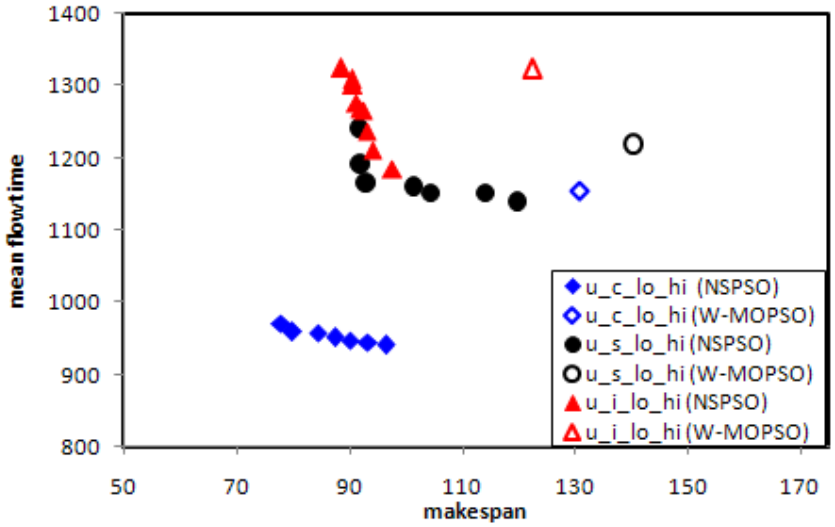

Fig. 2: Performance comparison of NSPSO and W-MOPSO for low task, high machine heterogeneity

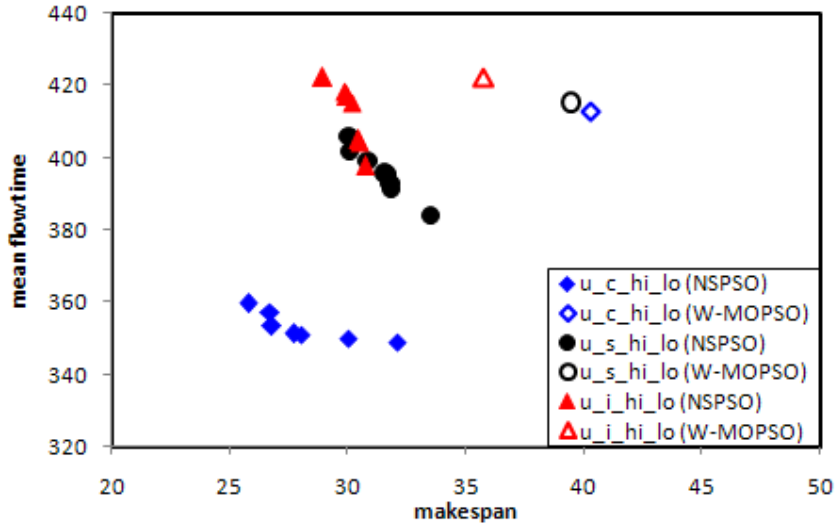

Fig. 3: Performance comparison of NSPSO and W-MOPSO for high task, low machine heterogeneity

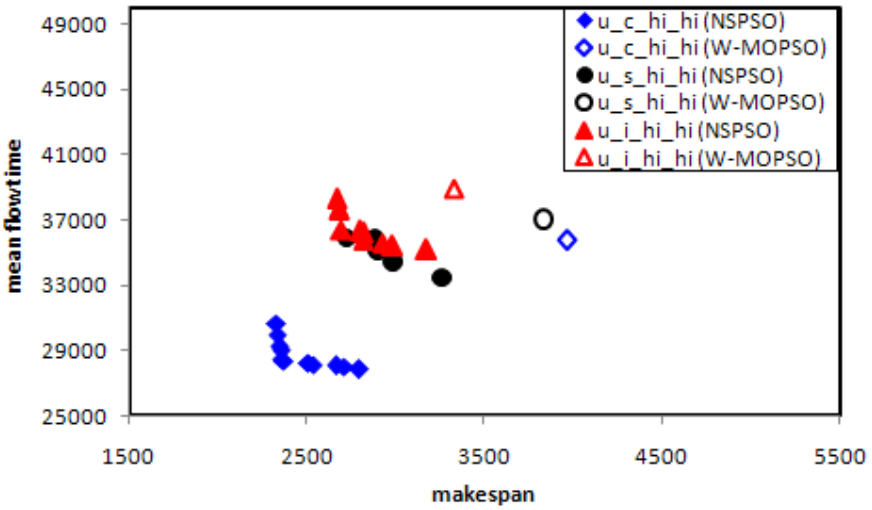

Fig.4: Performance comparison of NSPSO and W-MOPSO for high task, high machine heterogeneity

From the plots it is observed that the schedules obtained by NSPSO algorithm results in minimal makespan and minimal flowtime when compared with the best schedule obtained through W-MOPSO for the same number of iterations. This proves that the Pareto-optimal approach of finding solutions to multi-objective task scheduling problem is more effective and produces better solutions. As the problem tested comprises of 512 tasks, each particle has 512 positions and requires more number of iterations to generate the optimal solutions. It is seen from the results plotted that only a maximum of 15 solutions were found to be non-dominating among the initial population of 100 solutions even after 1000 generations. To obtain more number of solutions in the non-dominated set of rank1 it is required to run the algorithm still few thousand generations. It is found that the optimal solution obtained for all types of heterogeneous systems obtained by W-MOPSO lie away from the non-dominated solutions. In inconsistent cases, it is seen that the solution obtained by W-MOPSO is comparatively closer than the other two types.

6.3 Best Compromise Solution

The Pareto optimal set obtained by applying NSPSO comprises of solutions that satisfy different goals to some extent [22]. Hence for effective comparison it is practical to choose one solution from the obtained Pareto set. A Fuzzy-based approach is applied to select the best compromise solution from the obtained Pareto set. This is due to the fuzzy or imprecise nature goals of each objective function [23]. Hence, the membership functions are introduced to represents the goals of each objective function. In this , a simple linear membership function was

considered for each of the objective functions. The membership function is defined as follows [23]

$$\mu_k = \begin{cases} 1, & f_k \le f_{\min} \\ \dfrac{f_k^{\max} - f_k}{f_k^{\max} - f_k^{\min}}, & f_k^{\min} < f_k < f_k^{\max} \\ 0, & f_k \ge f_{\max} \end{cases} \tag{10}$$

where $f_k^{\max}$ and $f_k^{\min}$ are the maximum and minimum values of the k-th objective function, among all nondominated solutions respectively.

For each non-dominated solution i, the normalized membership function μ^i is calculated

$$\mu^i = \frac{\sum_{k=1}^{n} \mu_k^i}{\sum_{i=1}^{m} \sum_{k=1}^{n} \mu_k^i} \tag{11}$$

where *n* is the number of objectives functions and *m* is the number of non-dominated solutions. The function μ^i can be considered as a membership function of non-dominated solutions in a fuzzy set, where the solution having the maximum membership in the fuzzy set is considered as the best compromise solution.

Table 2 compares the results of the best solution obtained by NSPSO using fuzzy based approach with W-MOPSO. The tabulated results verify that NSPSO exhibits better performance for all heterogeneous systems resulting in good schedules with minimum values of both makespan and flowtime.

Table 2: Makespan and mean flow time values for the best solution obtained by NSPSO in comparison with W-MOPSO

Problem Instance	W-MOPSO		Best compromise solution with NSPSO	
	Make span	Mean Flow time	Make span	Mean Flow time
u_c_lolo	13147.8146	138927.7324	*8664.8048*	*117821.7894*
u_c_lohi	1306429.9203	11539135.76835	*800174.51875*	*9590558.19116*
u_c_hilo	402567.68741	4127891.428444	*277143.87913*	*3512357.84960*
u_c_hihi	39642454.101	357292914.4200	*23572329.554*	*284455898.971*
u_s_lolo	13183.350319	140097.056526	*10181.234019*	*133332.872718*
u_s_lohi	1401040.7629	12185136.86652	*929886.22269*	*11646285.2373*
u_s_hilo	394100.64126	4151139.102559	*301078.13824*	*4017087.20455*
u_s_hihi	38206507.901	370121079.5219	*29851120.200*	*344021357.343*
u_i_lolo	14310.789926	141295.647419	*10722.189847*	*136339.872525*
u_i_lohi	1223733.8337	13077839.61080	*941143.95154*	*12098941.5217*
u_i_hilo	357708.37182	4222794.014857	*307548.62870*	*3976545.55130*
u_i_hihi	33067834.545	385235242.3981	*26921663.456*	*363928422.755*

7 Conclusion

In distributed computing systems, qualified assignment of tasks among processors is an important step for efficient utilization of resources and execution of the tasks. In this paper, the application of multi-objective Non-dominated Sorting Particle Swarm Optimizer (NSPSO) intends to find schedules for independent tasks minimizing the makespan and flowtime simultaneously. The study also reveals the quality of schedules in comparison to a weighted PSO that simultaneously minimizes both the objectives for several benchmark problems. It is found that solving multi-objective task scheduling using a Pareto-optimal approach is more effective in determining optimal solutions. However, further work could be carried out with the NSPSO algorithm investigating different methods of updating position and velocities of the particle. Investigations also can be extended to considering several forms of HC scheduling, such as scheduling jobs with precedence constraints or in dynamic environments.

References

[1] A. Abdelmageed Elsadek, B. Earl Wells,"A Heuristic model for task allocation in heterogeneous distributed computing systems", *International Journal of Computers and Their Applications*, Vol. 6, No. 1, (1999).

[2] M.R. Garey and D. Johnson ," *Computers and Intractability: A Guide to the theory of NP-Completeness*", Freeman and Company, San Francisco, (1979).

[3] A. Abraham, H. Liu, W. Zhang, T.G. Chang, "Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm", Springer-Verlag Berlin Heidelberg , (2006), pp. 500-507.

[4] James Kennedy, Russell Eberhart, "Particle Swarm Optimization", *Proc. IEEE International Conference on Neural Networks*, Vol.4, (1995), pp.1942-1948.

[5] J. Kennedy and Russell C. Eberhart, 'Swarm Intelligence", Morgan-Kaufmann, (2000), pp 337-342

[6] Y. Shi, and R. Eberhart , "Parameter Selection in Particle Swarm Optimization, Evolutionary Programming VII", *Proceedings of Evolutionary Programming*, (1998), pp. 591-600

[7] X. Li, "A Non-dominated Sorting Particle Swarm Optimizer for Multi-objective Optimization", in *Proceeding of Genetic and Evolutionary Computation Conference* 2003 (GECCO'03), Chicago, USA, (2003).

[8] H.J. Braun et al, "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,No.6, (2001).

[9] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, and R.F. Freund. "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", *Journal of Parallel and Distributed Computing*, Vol.59, No.2, (1999), pp.107–131.

[10] R.F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling resources in multiuser, heterogeneous, computing environments with SmartNet", in *7th IEEE Heterogeneous Computing Workshop (HCW '98),* (1998), pp. 184-199.

[11] A. Abraham, R. Buyya, and B. Nath," Nature's heuristics for scheduling jobs on computational grids", In *The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000*), India, (2000).

[12] H.Izakian , A. Abraham and V. Snasel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments", *Proceedings of International Joint Conference on Computational Sciences and Optimization, (*2009), pp.8-12.

[13] J. Page and J. Naughton,"Framework for task scheduling in heterogeneous distributed computing using genetic algorithms", *Artificial Intelligence Review*, (2005), Vol. 24, pp.415–429.

[14] G. Ritchie and J. Levine, "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).

[15] A. Yarkhan , J. Dongarra , "Experiments with scheduling using simulated annealing in a grid environment", In *Proceedings of the 3rd International Workshop on Grid Computing (GRID2002)*, Baltimore, MD, USA, November 18, (2002), pp. 232–242.

[16] I. Salman, I, S. Ahmad, Al-Madani, "Particle swarm optimization for task assignment problem", Microprocessors and Microsystems Vol. 26,(2002), pp.363-371.

[17] Javier Carretero, Fatos Xhafa and Ajith Abraham, "Genetic Algorithm Based Schedulers for Grid Computing Systems", *International Journal of Innovative Computing, Information and Control, Vol.* 3, No. 6, (2007).

[18] A. Abraham, H . Liu, C. Grosan, F. Xhafa, " Nature inspired meta-heuristics for grid scheduling: single and multi-objective optimization approaches", *Studies in Computational Intelligence*, Springer Verlag: Heidelberg, Germany, (2008), pp. 247–272.

[19] A. Abraham, H. Liu, W. Zhang and T.G. Chang, "Job Scheduling on Computational Grids Using Fuzzy Particle Swarm Algorithm", *10th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, B. Gabrys et al. (Eds.): Part II, Lecture Notes on Artificial Intelligence ,Springer, 4252, (2006), pp.500-507.

[20] H. Izakian, B. Tork Ladani, K. Zamanifar, A. Abraham , "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.

[21] V. Chankong, Y.Haimes," Multi-objective Decision Making Theory and Methodology, North-Holland, New York, (1983).

[22] T. Niimura, T. Nakashima, "Multi-objective tradeoff analysis of deregulated electricity transactions", *International Journal of Electrical Power & Energy Systems* ,25,(2003), pp. 179–185.

[23] J.S. Dhillon, S.C. Parti, D.P. Kothari, "Stochastic economic emission load dispatch", Electric Power Systems Research ,26,(1993) pp. 179–186.