ISSN 2229-6956(ONLINE)
DOI: 10.21917/ijsc.2010.0002

A FAST AND ELITIST BI-OBJECTIVE EVOLUTIONARY ALGORITHM FOR SCHEDULING INDEPENDENT TASKS ON HETEROGENEOUS SYSTEMS

G.Subashini[1] and M.C.Bhuvaneswari[2]
[1]Department of Information Technology, P.S.G College of Technology, India
E-mail: suba@ity.psgtech.ac.in
[2]Department of Electrical and Electronics Engineering, P.S.G College of Technology, India
E-mail: mcb@eee.psgtech.ac.in

Abstract
To meet the increasing computational demands, geographically distributed resources need to be logically coupled to make them work as a unified resource. In analyzing the performance of such distributed heterogeneous computing systems scheduling a set of tasks to the available set of resources for execution is highly important. Task scheduling being an NP-complete problem, use of meta-heuristics is more appropriate in obtaining optimal solutions. Schedules thus obtained can be evaluated using several criteria that may conflict with one another which require multi objective problem formulation. This paper investigates the application of an elitist Non-dominated Sorting Genetic Algorithm (NSGA-II), to efficiently schedule a set of independent tasks in a heterogeneous distributed computing system. The objectives considered in this paper include minimizing makespan and average flowtime simultaneously. The implementation of NSGA-II algorithm and Weighted-Sum Genetic Algorithm (WSGA) has been tested on benchmark instances for distributed heterogeneous systems. As NSGA-II generates a set of Pareto optimal solutions, to verify the effectiveness of NSGA-II over WSGA a fuzzy based membership value assignment method is employed to choose the best compromise solution from the obtained Pareto solution set.

Keywords:
Heterogeneous Computing, Task Scheduling, Multi-objective, NSGA-II, Pareto-optimal

1. INTRODUCTION

A heterogeneous computing (HC) environment comprises of a distributed suite of different high-performance processors, interconnected with high-speed networks to efficiently solve computationally intensive problems with diverse computational needs [1]. The applicability and strength of HC system lie in satisfying the computing needs by assigning appropriate resources. This problem of task scheduling is complex due to the high degree of resource heterogeneity and the large number of tasks submitted by different users or applications. In the HC environment considered here, the tasks are assumed to be independent, i.e., no interaction takes place between the tasks. This scenario is likely to be present, for instance, when many independent users submit their tasks to a collection of shared computational resources. Due to the importance of static scheduling in analysis and performance evaluation of certain systems, this paper considers static scheduling of tasks. High-powered computational grids [2] also may utilize static scheduling techniques to distribute resources and computational power. It is also assumed that each processor executes a single task at a time, in the order in which the tasks are assigned and no preemption of tasks takes place.

Certain heuristics that are fast, straightforward and easy to implement have been used in finding near-optimal solutions. Some of them are Sufferage [3], min-min [4], max-min [4], LJFR-SJFR [5], min-max [6], etc. From a computational complexity perspective, task scheduling is computationally hard. To improve the quality of solutions, and overcome the practical difficulty in solving large-scale optimization problems meta-heuristics have been presented for task scheduling problem. The most popular of meta-heuristic algorithms are ant colony optimization (ACO) [7], simulated annealing (SA)[8], genetic algorithm (GA)[9], and particle swarm optimization (PSO)[11],[12]. Braun etal [10] 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. Most of the available heuristics and meta-heuristics aimed at minimizing the makespan of the schedule.

Very few attempts have been made to minimize flowtime or both flowtime and makespan on HC environments. In [6] five popular heuristics for minimizing makespan and flowtime are compared. However the objectives are evaluated separately. Xhafa *et.al* [13] used Genetic Algorithm-based schedulers for computational grids. In this, multiple GA operators are implemented and compared to find the best GA scheduler for this problem. In [14] the authors also focused on Struggle Genetic Algorithms and their tuning for scheduling of independent tasks in computational grids. The authors in [15] exploited the capabilities of Cellular Memetic Algorithms (CMA) for obtaining efficient batch schedulers for grid systems. Abraham *et.al* [16] illustrated the usage of several nature inspired meta-heuristics (SA, GA, PSO, and ACO) for scheduling tasks in computational grids using single and multi-objective optimization approaches. Also Xhafa and Abraham [17] have reviewed the most important concepts from grid computing related to scheduling problems and their resolution using heuristic and meta-heuristic approaches. Several works are available in literature that deals with scheduling dependent tasks which are not cited in this paper.

In the case of independent tasks, makespan and flowtime are two important objectives that are to be considered in obtaining optimal solutions. Makespan is the time when an HC system finishes the last task and flow time is the sum of finalization times of all the tasks. Hence this scheduling problem is formulated as a multi-objective problem with the goal of minimizing the makespan and flowtime of the system. The recent development of Multi-Objective Evolutionary Algorithm (MOEA) algorithms is gaining popularity in several applications. This combines two major disciplines: evolutionary computation and the theoretical frameworks of multi-criteria decision making. There are two general approaches to multiple objective optimizations. One approach to solve multi- objective optimization problems is by combining the multiple objectives into a scalar cost function, ultimately making the problem single-objective prior to optimization. However, 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. These methods also have the disadvantage of requiring new runs of the optimizer every time the preferences or weights of the objectives in the multi-objective function change [18]. Most of the existing solutions use this approach to solve the multi-objective optimization problem. The second general approach is to determine an entire Pareto optimal solution set or a representative subset. Pareto optimal solution sets are often preferred to single solutions because they can be practical when considering real-life problems, since the final solution of the decision maker is always a trade-off between crucial parameters [19].

In this paper the modularity and scalability of the two algorithms NSGA-II and WSGA has been tested using the benchmark of Braun. [10], 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 processors and workload of tasks. A fuzzy based membership value is applied to the Pareto optimal set obtained with NSGA-II to determine the best solution in the set for effective comparisons.

The remainder of this paper is organized as follows. Section 2 formulates the problem. Section 3 gives a general introduction to multi-objective optimization. Description of multi-objective genetic algorithms NSGA-II and WSGA procedure for solving task scheduling problem is presented in Section 4. Experimental details and simulation results are presented in Section 5 and finally, Section 6 concludes with finishing remarks and future work.

2. PROBLEM FORMULATION

Real-world HC systems, are complex combinations of hardware, software and network components. 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 = $\{P_1, P_2, \ldots, P_m\}$ within the HC environment. As the scheduling is performed statically an estimation of the computational load of each task and the computing capacity of each resource in the HC environment is assumed to be available a priori. This formulation is practically applicable as the computing capacity and computational needs can be known from user specifications, historic data or prediction. This information can be contained in an Expected Time to Compute (ETC) matrix where each position ETC[n][m] indicates the expected time to compute task n in processor m. A row in an ETC matrix contains the ETC for a single task on each of the available processors. A simple example ETC matrix with details for 4 tasks and 2 processors is given in Table.1.

Table.1 An example ETC matrix

	Processor 1	**Processor 2**
Task 1	3	2
Task 2	2	4
Task 3	7	5
Task 4	6	7

In order to simulate various possible heterogeneous scheduling problems as realistically as possible, benchmarks of instances are generated capturing different characteristics of distributed heterogeneous systems such as consistency of computing, heterogeneity of processors and heterogeneity of tasks [3][10].

This problem is formulated to achieve minimum makespan and minimum mean flowtime. Makespan is, the time when last task is finished and flowtime includes the sum of finalization times of all the tasks. Makespan and flowtime values are in incomparable ranges, as flow-time has a higher magnitude order over makespan, and its difference increases as more tasks and processors are considered. For this reason, the mean flow-time is considered as an objective defined as follows:

$$\text{makespan}: \min_{S_i \in Sched} \{\max_{t \in Tasks} F_t\} \quad (1)$$

$$\text{flowtime}: \min_{S_i \in Sched} \{\sum_{t \in Tasks} F_t\} \quad (2)$$

$$\textit{av.flowtime : flowtime/P} \quad (3)$$

where F_t denotes the time when task t finalizes, *Sched* is the set of all possible schedules and Tasks the set of all tasks to be scheduled. P represents the total number of processors.

3. MULTI-OBJECTIVE OPTIMIZATION

A Multi-objective Optimization Problem (MOP) differs from a single-objective optimization problem as it contains several objectives to be optimized. In single objective optimization problems, the goal is to obtain the best single solution. But for multi-objective problems, with several and possibly conflicting objectives, there is no single solution but rather a set of potential solutions.

A general formulation of a MOP consists of a number of objectives with a number of inequality and equality constraints. Mathematically, the problem is written as [20]

$$\text{Minimize/Maximize} \quad f_i(x) \text{ for } i = 1, 2, \ldots, n \quad (4)$$

subject to constraints:

$$g_j(x) \le 0,\ j = 1, 2, \ldots, J,$$

$$h_k(x) \le 0,\ k = 1, 2, \ldots, K.$$

where

$f_i(x) = \{f_1(x), \ldots, f_n(x)\}$,

n = number of objectives or criteria to be optimized,

x = $\{x_1, \ldots, x_p\}$ is a vector of decision variables,

p = number of decision variables.

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. For a minimization problem, dominance is defined as follows.

A vector u = (u_1,… , u_n) is said to dominate v = (v_1,…..,v_n) if and only if u is partially less than v

$$\forall i \in \{1,....,n\}, u_i \le v_i \wedge \exists i \in \{1,....,n\}, u_i < v_i$$

A solution $x_u \in U$ is said to be Pareto-optimal if and only if there is no $x_v \in V$ for which

v= f(x_v)= (v_1, . . . , v_n) dominates u = f(x_u) = (u_1, . . . , u_n).

Pareto-optimal solutions are also called efficient, non-dominated, and non-inferior solutions. The set of all non-dominated solution is known as the non-dominated set of the problem. The elements in the Pareto set has the property that it is impossible to further reduce any of the objective functions, without increasing, at least, one of the other objective functions. The ability to handle complex problems, involving several features reveals the potential effectiveness of GA in optimization problems.

4. MULTI–OBJECTIVE GENETIC ALGORITHM FOR TASK SCHEDULING

Being a population based approach; GA is well suited to solve MOPs. The ability of GA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems. The crossover operator of GA exploits structures of good solutions with respect to different objectives to create new non-dominated solutions in unexplored parts of the Pareto front.

4.1 WEIGHTED SUM GENETIC ALGORITHM (WSGA)

The initial population P_0 is generated randomly to contain predefined number of chromosomes (schedules) represented as a string of length equal to the number of tasks. The value corresponding to each position *i* in the string represent the processor to which task *i* is allocated. Considering the availability of 7 tasks and 3 processors, a schedule and the task allocation can be represented as depicted in Fig. 1.

1	3	2	1	2	1	3

Processor 1	Task 1	Task 4	Task 6
Processor 2	Task 3	Task 5	
Processor 3	Task 2	Task 7	

Fig.1. Schedule representation

In this method, the makespan given in (1), the average flow-time in (3) are weighted according to their relative importance and the weighted functions are added together to produce the objective function as

*Min f= w1*makespan +w2*av.flowtime* (5)

where weights w_1 and w_2 are chosen such that the sum of weights will be equal to one. As the aim is to minimize the combined objective function, the individual that has a lower fitness value will be the best one.

This problem uses a Binary Tournament Selection to select good individuals for reproduction. Selection is done by making two randomly selected individuals to participate in the tournament and choose the best among them. This scheme probabilistically duplicates some chromosomes and deletes others, where better mappings have a higher probability of being duplicated in the next generation. Investigations prove that this is comparatively better than other selection methods suitable for the problem [13].

Genetic algorithm uses two operators to generate new solutions from existing ones: crossover and mutation. Crossover obtains descendants of better quality that will feed the next generation and enable the search to explore new regions of solution space not explored yet. This is achieved by selecting individuals from the parental generation and interchanging their genes to obtain new individuals. A single point crossover is used in this problem with a high probability.

Mutation alters one ore more gene values in an individual from its initial value preventing the population from stagnating at any local optima. As movement of tasks between processors is considered to be effective, the swap mutation operator used here interchanges the allocation of two tasks assigned to two different processors.

This completes one generation of the GA. All the individuals available at the end of the first iteration will be treated as parents for the second iteration. This procedure is repeated for the number of iterations as given by the user.

4.2 NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA– II)

Pareto-based fitness assignment was first proposed by Goldberg [21], the idea being to assign equal probability of reproduction to all non-dominated individuals in the population. NSGA-II uses a fast non-dominated sorting procedure with complexity O ($k\mu^2$), [23]. A, a combined ranking and crowding distance approach is used to ensure the diversity of the individuals within the population. An initial population N is generated. From this N offsprings are generated using selection, crossover and mutation operations. Both the parent and offspring individuals are combined to form a population of 2N individuals and sorted based on non-domination into various fronts. In addition a new parameter called crowding distance is calculated for each individual. The crowding distance is a measure of how close an individual is to its neighbors. Each individual is assigned a fitness based on front in which they belong to. Individuals in first front with a larger crowding distance is assigned with a minimum fitness and one in the higher front with less crowding distance is given a higher fitness value. Based on fitness values determined, N individuals are selected having less fitness values.

Implementation of NSGA-II requires the determination of some fundamental issues. After initializing the population the following schemes are employed [22], [23].

4.2.1 Non-Dominated Sort:

Non-dominated sorting is used to classify the population to identify the solutions for the next generation. This allows a global non-domination check among the offspring and parent solutions. The procedure for sorting is implemented as given below

Step 1: For each solution p in population N.

Step 2: For each solution q in population N.

Step 3: If q not equal to p

Compare p and q for all 'm' objectives

Step 4: If for any q, p is dominated by q mark solution p as dominated.

The solutions, not marked, are called non-dominated solutions, which form the first non-dominated front in the population. The process is repeated with the remaining solutions for other higher non-domination fronts until all the populations are classified into different fronts.

4.2.2 Crowding Distance:

The basic idea behind the crowing distance calculation is the determination of Euclidian distance between each individual in a front based on their *m* objectives in the m dimensional space. All the individuals in the population are assigned a crowding distance value as the individuals are selected based on rank and crowding distance. Crowding distance is assigned front wise as below

Step 1: For all j individuals in each front F_k, initialize the distance to be zero, $F_k(d_j) = 0$, where j corresponds to the jth individual in front F_k.

Step 2: For each objective function m, sort the individuals in front F_k based on objective m, I = sort(f_k,m).

Step 3: For m=1,2,…………,M assign a large distance to the boundary solutions or $I(d_1)^m = \infty$ and $I(d_n)^m = \infty$.

Step 4: For j = 2 to (n – 1)

$$I(d_j)^m = \frac{I(d_j) + I(j+1)f_m - I(j-1)f_m}{f_m^{max} - f_m^{min}}$$

$I(j)f_m$ is the value of the mth objective function of the jth individual in I .

4.2.3 Selection:

The selection of solutions is performed using a Crowded Tournament Selection Operator given as below:

A solution x wins the tournament with another solution y if any of the following conditions are true.

1. If solution x has a better rank, i.e., $r_x < r_y$.

2. If x and y hold the same rank then solution xhas a better crowding distance than solution y, that is $r_x = r_y$ and $d_x > d_y$.

The first condition makes sure that the chosen solution lies on a better non dominated front. The second condition resolves the tie of both the solutions being on the same non dominated front by deciding on their crowded distance. The one residing in the less crowded area wins, that is, it has a larger crowding distance.

4.2.4 Genetic Operators:

Genetic algorithm uses two operators to generate new solutions from existing ones: crossover and mutation. Single point crossover and swap mutation is used by NSGA-II in this paper against the Simulated Binary Crossover and Polynomial Mutation that is normally implemented.

5. RESULTS AND DISCUSSIONS

In order to evaluate NSGA-II algorithm and WSGA and compare their performances a simulation model used in [10] is developed to generate several types of instances.

5.1 SIMULATION RESULTS

The simulation model in [10] is based on expected time to compute (ETC) matrix for 512 tasks and 16 processors. The instances of the benchmark are classified into 12 different types of ETC matrices according to the three following metrics: task heterogeneity, processor heterogeneity, and consistency. In ETC matrix, task heterogeneity is defined as the amount of variance possible among the execution times of the tasks. Processor heterogeneity represents the possible variation of the running time of a particular task across all the processors, both of which can be classified into low or high heterogeneity.

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 tasks faster than p_j . Inconsistency means that a processor is faster for some tasks 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. All instances consist of 512 tasks and 16 processors and are labeled *u- x- yy-zz* that represents the following

u means uniform distribution (used in generating the matrix).

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

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

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

The results of all heuristic techniques are sensitive to the parameters. Hence, in setting the parameters, it is required to do extensive simulations to find suitable values for various parameters. The best parameters for the NSGA-II which are selected through test simulation runs are given in Table 2.

Table.2. Best Parameter values

Parameters	**WSGA**	**NSGA-II**
Population size 100	100	100
Number of iterations	100	100
Pc, crossover probability	0.8	0.8
Pm, mutation probability	0.02	0.02
Crossover type	Single point	Single point
Mutation type	swap	swap
Selection type	Binary Tournament	crowded Tournament
Weights used	w1=0.75 ,w2=0.25	-

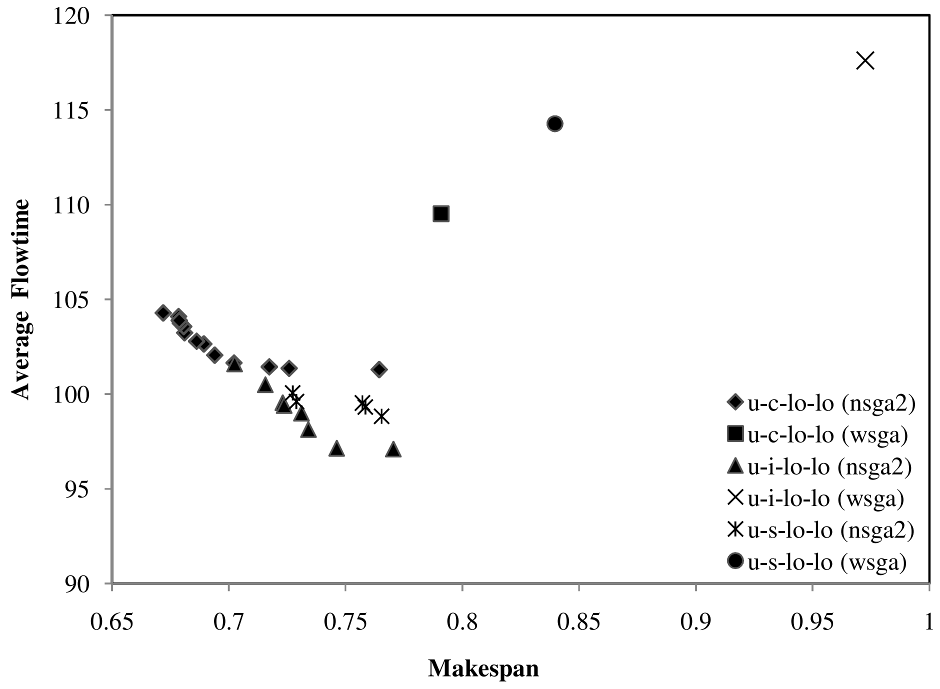

Fig.2. Pareto-optimal solutions obtained with NSGAII and WSGA for low task, low processor heterogeneity

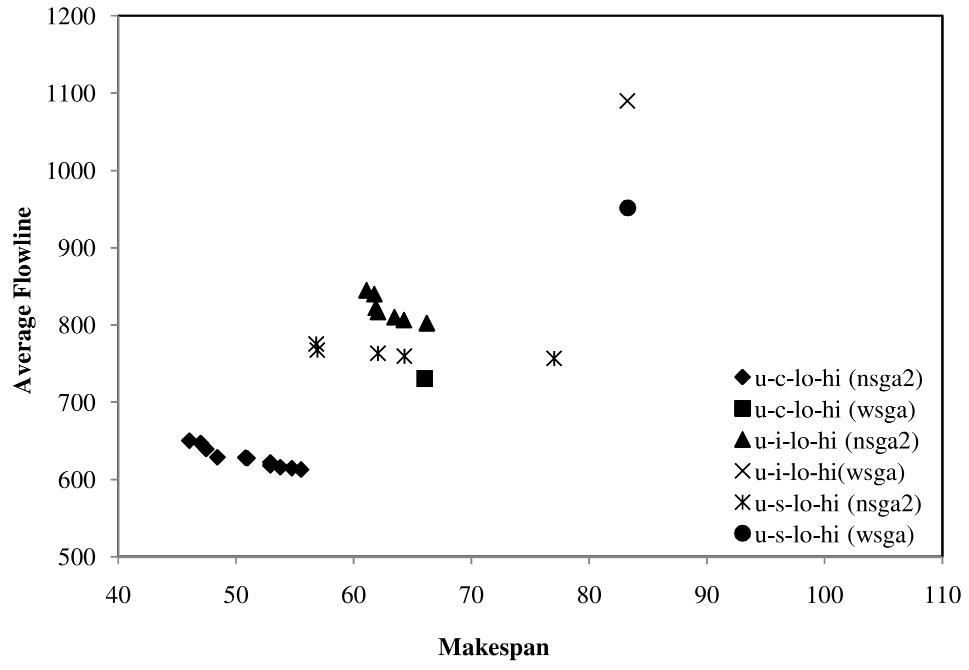

Fig.3. Pareto-optimal solutions obtained with NSGAII and WSGA for low task, high processor heterogeneity

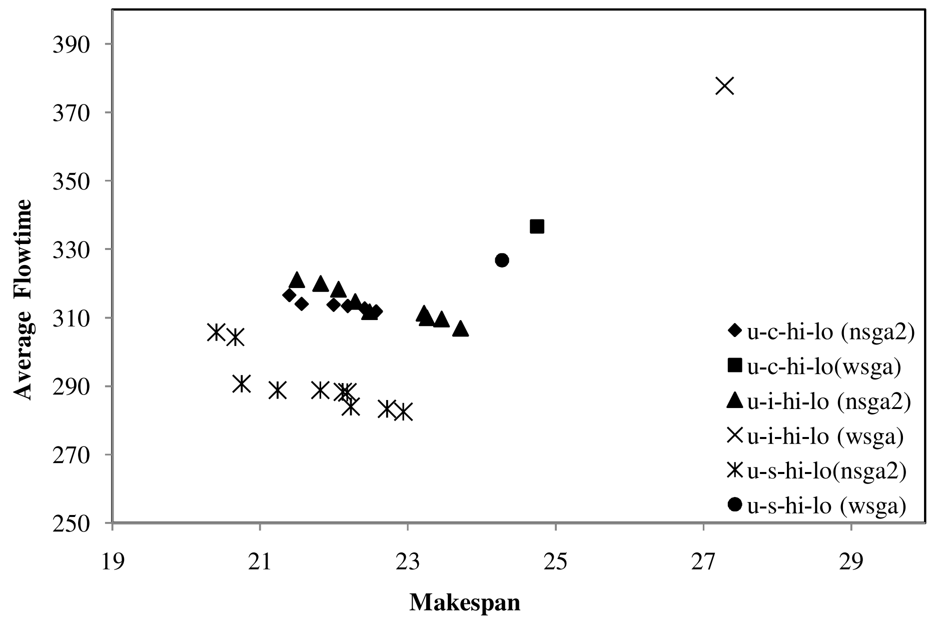

Fig.4. Pareto-optimal solutions obtained with NSGAII and WSGA for high task, low processor heterogeneity

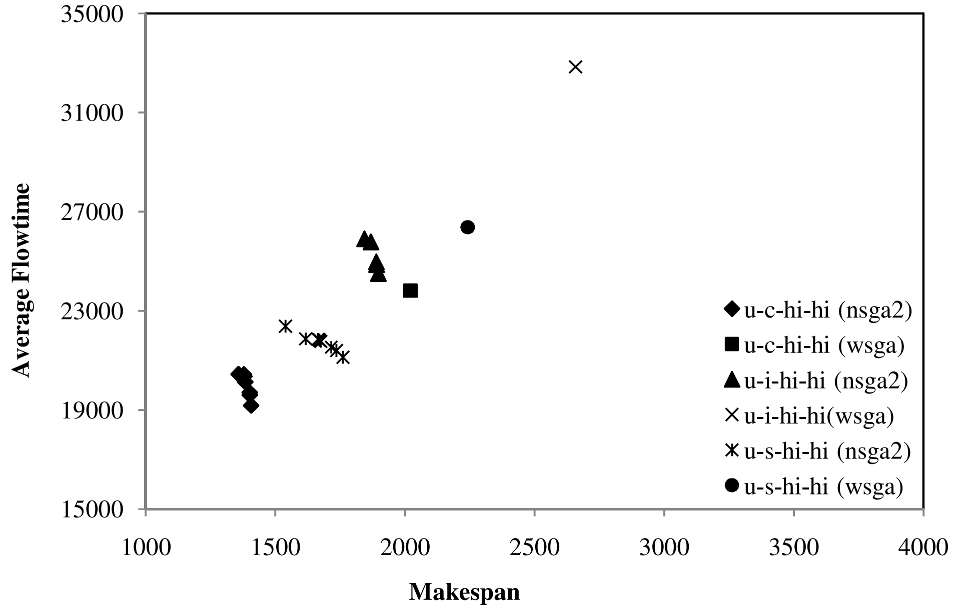

Fig.5. Pareto-optimal solutions obtained with NSGAII and WSGA for high task, high processor heterogeneity

NSGA-II and WSGA algorithms were simulated and the results taken with the best parameter settings are plotted. The algorithms are run on the same instance and under the same configuration 10 times and the best obtained results are plotted. The values of makespan and mean flowtime are measured in same time units. The values obtained for the pareto-optimal solutions are plotted in ten thousands of units in Figures 2-5. From the plots it can be seen that the schedules obtained by implementing NSGA-II algorithm has minimal makespan and minimal flowtime when compared with the best schedule obtained through WSGA which lies far away as seen in the plots. This proves that the Pareto-optimal approach of finding solution to multi-objective task scheduling problem is more effective and produces better solutions. The results are plotted by running both the algorithms for hundred generations.

The number of Pareto optimal solutions in Front 1 obtained using NSGA-II increases as the number of iterations increase and the solutions in the various fronts come close to one another as the algorithm is run for several iterations. Fig 6-9 shows the Pareto optimal solutions obtained with NSGA-II by varying the number of iterations for a u_c_hilo.0 instance type considering an initial population of 50 chromosomes and the above specified parameter settings. Fig. 6 and 7 shows the non-dominated solutions in both first and second front. As the solutions in adjacent fronts lie closer at higher iterations the solutions in Front 1 alone are plotted in Fig. 8 and 9. The increase in number of optimal solutions are seen in all the plots as the number of iterations increase. Similar results were obtained on other instances which indicates that more number of optimal solutions are generated as the algorithm is run many times thereby avoiding local optima. From the results obtained it can also be seen that increasing the number of iterations result in better solutions from the fitness values.

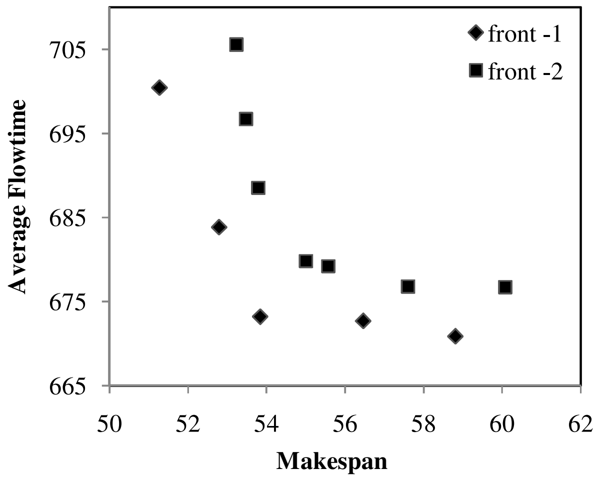

Fig.6. Pareto-optimal solutions in NSGA-II after 100 generations

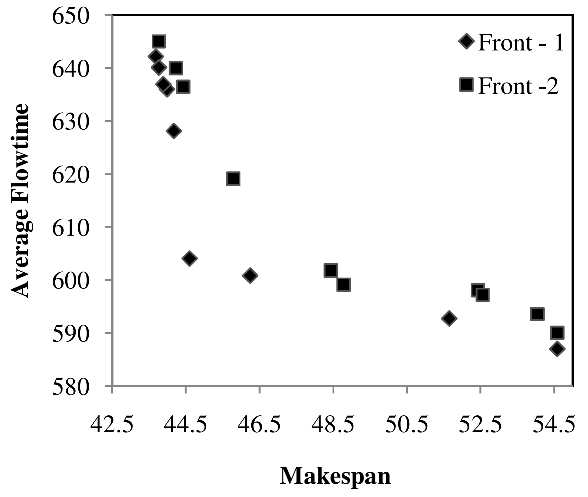

Fig.7. Pareto-optimal solutions in NSGA-II after 200 generations

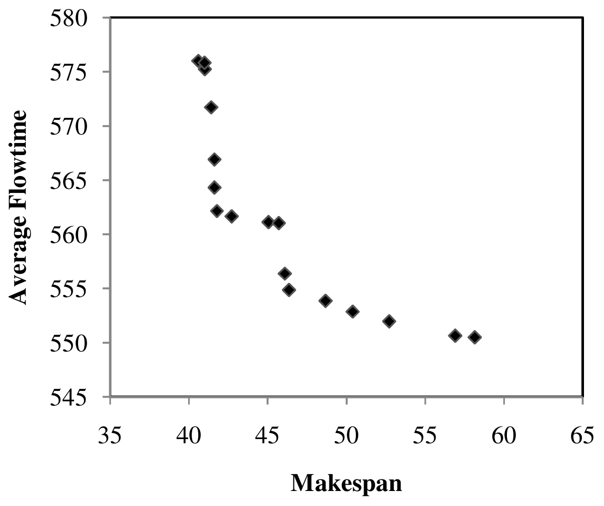

Fig.8. Pareto-optimal solutions in NSGA-II after 300 generations in Front 1

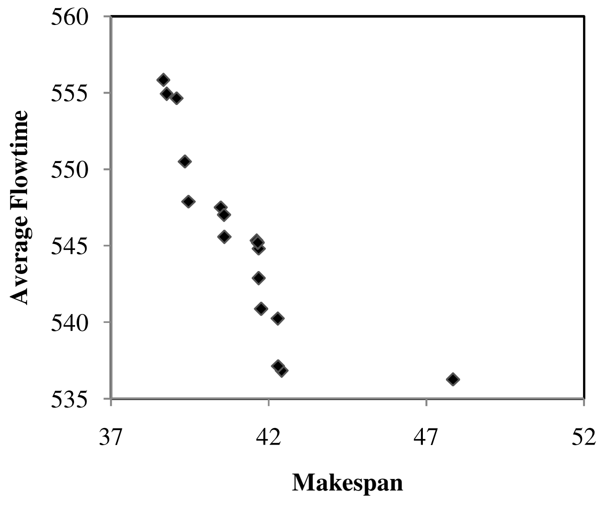

Fig.9. Pareto-optimal solutions in NSGA-II after 400 generations in in NSGA-II Front1

5.2 FUZZY BASED APPROACH

A Fuzzy-based approach is applied to select the best compromise solution from the obtained Pareto set. The k-th objective function of a solution in a Pareto set f_k is represented by a membership function μ_k defined as [24]

$$\mu_k = \begin{cases} 1, & f_k \le f_{\min} \\ \dfrac{f_k^{\max} - f_k}{f_k^{\max} - f_k^{\min}}, & f_k^{\min} < f_k < f_k^{\max} \\ 0, & f_k \ge f_{\max} \end{cases} \quad (4)$$

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

For each solution i , the membership function μ^i is calculated as:

$$\mu_i = \frac{\sum_{k=1}^{n} \mu_k^i}{\sum_{i=1}^{m} \sum_{k=1}^{n} \mu_k^i} \qquad (5)$$

where n is the number of objectives functions and m is the number of solutions.

The solution having the maximum value of μ^i is the best compromise solution. Table 3 shows the makespan and mean flowtime values for the best compromise solution obtained by NSGA-II that can be effectively compared with that of WSGA which proves NSGA-II outperforms WSGA.

6. CONCLUSION

Statically scheduling independent tasks in heterogeneous computing environments finds usefulness in predictive analyses, impact studies, and post-mortem analyses. In this paper, a multi-objective Non-dominated Sorting Genetic algorithm is applied to find schedules for independent tasks minimizing the makespan and flowtime simultaneously. The experimental results reveal the quality of schedules in comparison to a weighted GA for all benchmark problem instances. Therefore NSGA-II can be used to find better schedules satisfying multiple objectives and it seems a promising approach to scheduling independent tasks in HC environments. Investigations also can be extended to considering several forms of HC scheduling, such as scheduling tasks with precedence constraints or in dynamic environments.

Table.3 Makespan and mean flow time values for the best solution obtained by NSGA-II in comparison with WSGA

Problem Instance	**WSGA**		**NSGA-II with fuzzy**	
	Make span	**Mean Flow time**	**Make span**	**Mean Flow time**
u_c_lolo.0	7908.439487	109527.247525	**7022.690479**	**101646.392356**
u_c_lohi.0	660288.423171	7303868.317070	**484360.508061**	**6287677.584696**
u_c_hilo.0	247499.153235	3366470.633626	**215624.789783**	**3140585.963835**
u_c_hihi.0	20213891.13913	238095476.71669	**14071196.16461**	**191815426.49902**
u_s_lolo.0	8396.812439	114276.135706	**7288.960389**	**99618.841779**
u_s_lohi.0	832827.545638	9513825.056911	**643263.153080**	**7594550.134683**
u_s_hilo.0	242699.484059	3268130.573917	**207484.642142**	**2907533.239039**
u_s_hihi.0	22424274.66160	263806051.47077	**16179189.27053**	**218631980.08911**
u_i_lolo.0	9725.625921	117621.278907	**7462.832029**	**97147.631759**
u_i_lohi.0	832621.177800	10900565.040223	**620779.034615**	**8168155.371800**
u_i_hilo.0	272862.101494	3778086.185060	**224802.512895**	**3117567.422151**
u_i_hihi.0	26582765.68203	328476497.93762	**18981587.86093**	**244977496.32083**

REFERENCES

[1] S. Ali, T. D. Braun, H. J. Siegel, and A. A. Maciejewski, 2001, Heterogeneous computing, Encyclopedia of Distributed Computing, Kluwer Academic.

[2] Foster and C. Kesselman, 1998, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufman, New York.

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

[4] 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, 1998, Scheduling resources in multiuser, heterogeneous, computing environments with SmartNet, in 7th IEEE Heterogeneous Computing Workshop (HCW '98),' pp. 184 – 199.

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

[6] H.Izakian , A. Abraham and V. Snasel, 2009, Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments. *In Proceedings of International Joint Conference on Computational Sciences and Optimization,* Vol.1, pp.8-12.

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

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

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

[10] H.J. Braun *et.al.*, 2001, A comparison of eleven static heuristics for mapping a class of independent tasks onto

heterogeneous distributed computing systems, *Journal of Parallel and Distributed Computing,* Vol.61, No.6, pp.810-837.

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

[12] H. Izakian, B. Tork Ladani, K. Zamanifar, 2009, A. Abraham , A novel particle swarm optimization approach for grid task scheduling, In *Proceedings of the Third International Conference on Information Systems, Technology and Management*, Springer: Heidelberg, Germany, pp. 100–110.

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

[14] F. Xhafa, B. Duran, A. Abraham, K. Dahal, 2008, Tuning struggle strategy in genetic algorithms for scheduling in computational grids. *Neural Network World,* pp. 209–225.

[15] F. Xhafa, E. Alba, B. Dorronsoro, B.Duran, A. Abraham, 2008, Efficient batch task scheduling in grids using cellular memetic algorithms. *Studies in Computational Intelligence*, Springer Verlag: Heidelberg, Germany, pp. 273–299.

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

[17] F. Xhafa, A. Abraham, 2008, Meta-heuristics for grid scheduling problems. *Studies in Computational Intelligence*, Springer Verlag: Heidelberg, Germany, pp.1–37.

[18] C.A.C. Coello, 1999, A comprehensive survey of evolutionary-based multiobjective optimization techniques, Knowledge and Information Systems Vol.1, pp. 269–308.

[19] V. Chankong, Y.Haimes, 1983, Multiobjective Decision Making Theory and Methodology, North-Holland, New York.

[20] S.S.Rao, 1991, Optimization Theory and Application, Wiley Eastern Limited, New Delhi.

[21] D.E.Goldberg, 1989, Genetic Algorithms in Search, Optimization and Processor Learning, Addison-Wesley.

[22] Kalyanmoy Deb, Multi-objective Optimization using Evolutionary Algorithms, John Wiley and Sons Ltd, 2002.

[23] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan , 2002, A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation Vol.6 (2002), pp.182–197.

[24] M.A. Abido, 2006, Multiobjective evolutionary algorithms for electric power dispatch problem, IEEE Trans. Evol. Comp. Vol.10, pp. 315–329.