Int. J. Open Problems Compt. Math., Vol. 4, No. 1, March 2011
ISSN 1998-6262; Copyright © ICSRS Publication, 2011
www.i-csrs.org

NSGA - II with Controlled Elitism for Scheduling Tasks in Heterogeneous Computing Systems

G. Subashini and 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

This paper presents an application of elitist Non-dominated Sorting Genetic Algorithm (NSGA-II), to efficiently schedule a set of independent tasks in a heterogeneous distributed computing system. This scheduling problem is a bi-objective problem considering two objectives. The first objective is minimization of makespan and the second one being the minimization of flowtime. As a multi-objective optimization technique, this paper makes use of Controlled-NSGA-II that modifies NSGA-II to enhance the diversity of solution sets. This algorithm uses a distribution function to control elitism and to get better diversity of individuals. The extent of elitism can be changed by fixing a user-defined parameter. Simulation results on task scheduling problem show the effectiveness of the proposed modification in comparison with NSGA-II.

Keywords: *controlled elitism ,Multi-objective, Non-dominated Sorting Genetic Algorithm (NSGA-II), Task Scheduling*

1 Introduction

Distributed computing systems have emerged as a powerful platform to perform different computationally intensive applications that have various computational requirements. The applicability and strength of such systems are derived from their ability to match computing needs to appropriate resources [1]. The problem of task scheduling is efficiently assigning user or application tasks to a set of

resources. A static task scheduling algorithm [2] can be used in such a heterogeneous system that 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. Due to the large number of resources and the large number of tasks submitted by different applications, task scheduling on heterogeneous computing (HC) systems is a large scale optimization problem. Finding optimal schedules in such a system has been shown, in general, to be NP-hard [3] and therefore the use of heuristics is one of the suitable approaches.

In the HC environment considered here, the tasks are assumed to be independent, i.e., no communications between the tasks are needed. This scenario is likely to be present, for instance, when many independent users submit their tasks to a collection of shared computational resources. The scheduling of these tasks is being performed statically. It is also assumed that each machine executes a single task at a time, in the order in which the tasks are assigned and no preemption takes place.

Some popular and efficient pure heuristics for task scheduling problem include min-min, max-min, LJFR-SJFR, min-max, Sufferage etc [4-6]. From a computational complexity perspective, task scheduling is computationally hard. In order to cope in practice with its difficulty and also, to improve the quality of solutions, meta-heuristics have been presented for task scheduling problem. The most popular of meta-heuristic algorithms are genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO) and particle swarm optimization (PSO). A hybrid ant colony optimization for scheduling in HC systems is proposed in [7]. In this, ant colony optimization is combined with local and tabu search to find shorter schedules. A simulated annealing approach for job scheduling in grids is given in [8]. Various heuristics are compared on different types of HC environments in [4] which illustrates that the GA scheduler can obtain better results in comparison with others. Most of the available methods aim at minimizing the makespan of the schedule.

Very few attempts have been made to minimize flowtime or both flowtime and makespan on HC environments. [6] 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. A genetic algorithm based schedulers for computational grids by transforming multi-objective problem into a single objective one is found in [9]. 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 has been illustrated in [10]. The most important concepts from grid computing related to scheduling problems and their resolution using heuristic and meta-heuristic approaches are reviewed in [11]. Several versions of

PSO has been implemented for the problem in literature [12,13].The authors identified different types of scheduling based on different criteria, such as static vs. dynamic environment, multi-objectivity, adaptivity, etc. [19] uses memetic algorithms to schedule tasks with communication delay between tasks.

The efficiency of scheduling algorithms can be evaluated based on different criteria the most important being makespan and flowtime. 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 ability of multi-objective evolutionary algorithms to find multiple Pareto-optimal solutions in one single run has made them attractive for solving problems with multiple and conflicting objectives. During the last decades, several multi-objective evolutionary algorithms [14] have been proposed which are aimed at finding the Pareto-optimal front and keeping diversity of individuals, in the obtained Pareto-optimal front. Recent studies have indicated that the inclusion of an elitist element can considerably improve the performance of a multi-objective evolutionary algorithm (MOEA) [15]. This resulted in a number of elitist MOEA of which the Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) has proved its performance against a set of well known difficult test functions.

This paper implements a modified version of NSGA-II for scheduling tasks in HC environment. A distribution function is used to limit the extent of elitism. This has been tested on several benchmark instances of task scheduling problem. The experimental results with controlled elitism show that the proposed algorithm has much better diversity thereby having a better performance over NSGA-II.

The remainder of this paper is organized as follows. The problem formulation is given in Section 2. Section 3 gives implementation of controlled NSGA-II for task scheduling problem. Experimental details and simulation results are presented in Section 4 and Section 5 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 to be scheduled on the HC system. It is assumed that the tasks are independent of each other with no intertask data dependencies and preemption is not allowed. At the time of the arrival of tasks m processors $P = \{P_1, P_2,..., P_m\}$ are available within the HC environment. To model the problem estimation or prediction of the computational load of each task, the computing capacity of each resource, and an estimation of the prior load of each one of the resources is required. This is the ETC – Expected Time to Compute matrix model [4]. Having the computing capacity of the resources and the workload of the tasks, an Expected Time to

Compute matrix ETC can be built, where each position ETC[n][m] indicates the expected time to compute task n in processor m. The entries ETC[n][m] could be computed by dividing the workload of task n by the computing capacity of processor m. The ETC model allows to quite easily introduce possible inconsistencies in the HC system and different levels of heterogeneity.

To formulate the objective $C_{i,j}$ (i ϵ{1, 2, ……..,n}, j ϵ {1, 2,……., m}) is defined as the completion time for finishing the task i on processor j and W_i ($i \epsilon$ 1,2, . . .,m) is the previous workload of P_i, Equation (1) shows the time required for P_i to complete the jobs include in it.

$$\sum(Ci+Wi) \tag{1}$$

Then makespan and flowtime can be defined as :

$$makespan \ : \ \max_{i \epsilon 1 \dots m} \{\sum C_i + W_i\} \tag{2}$$

$$flowtime \ : \ \sum_{i=1}^{m} C_i \tag{3}$$

Minimizing makespan aims to execute the whole set of tasks as fast as possible while minimizing flow time aims to utilize the computing environment efficiently. The goal of the scheduler is to minimize makespan and flowtime simultaneously.

3 NSGA-II for Task Scheduling

The NSGA-II algorithm and its detailed implementation procedure can be found in [16,17]. This algorithm has been demonstrated as one of the most efficient algorithms for multi-objective optimization on a number of benchmark problems. A brief description of NSGA-II is as follows:

NSGA-II uses non-dominated sorting for fitness assignments. All individuals not dominated by any other individuals, are assigned front number 1. All individuals only dominated by individuals in front number 1 are assigned front number 2, and so on. Selection is made, using tournament between two individuals. This selects an individual with the lowest front number if the two individuals are from different fronts. If the individuals are from the same front, then the individual with the highest crowding distance is selected if they are from the same front. A higher fitness value is assigned to individuals located on a sparsely populated part of the front. There are *N* parents and in every iteration *N* new offsprings are generated. Both parents and offspring compete with each other for inclusion in the next iteration.

NSGA-II uses Simulated Binary Crossover (SBX) and Polynomial mutation described as follows [17].

3.1 Simulated Binary Crossover

The SBX operator operates on two parent solutions and creates two offspring. The difference between offspring and parent depends on crossover index η_c. The crossover index 'η_c' is any non-negative real number. A large value of 'η_c' gives a higher probability for creating solutions closer to the parent and a small value of 'η_c' allows distant solutions to be selected as offspring. The two offspring created are symmetric about the parent solutions. Also, for a fixed 'η_c' the offspring have a spread which is proportional to that of the parent solutions. It has two properties: (a) the difference between corresponding decision variables of the created offspring is proportional to the difference between corresponding decision variables of the parent solutions; (b) offspring having decision variables nearer to those of the parent solutions are more likely to be selected.

3.2 Polynomial mutation

This operator has higher probability of creating a solution near to the parent than the probability of creating one distant from it. The shape of the probability distribution is directly controlled by an external parameter η_m and the distribution remains unchanged throughout the iterations.

3.3 Controlled elitism

Elitism is an important issue to ensure diversity of individuals and get a better convergence. In NSGA-II, elite solutions are emphasized on two occasions, i.e., once in the tournament selection operation and again during the elite preserving operation. This will cause a rapid deletion of solutions belonging to the non-elitist fronts. Though the crowding tournament operator will ensure diversity along the current non-dominated front, lateral diversity will be lost. Thus, to ensure better convergence, an algorithm may need diversity in both aspects – along and lateral to the Pareto-optimal front.

It is important to restrict the maximum number of currently elitist fronts adaptively and make solutions of non-elitist fronts involved in new population. For this purpose a predefined distribution of number of individuals in each front is maintained. A geometric distribution given by Eq.(4) is used.

$$N_i = rN_{i-1} \tag{4}$$

where N_i is maximum number of allowed individuals in the *i*th front; r(<1) reduction rate. Although the parameter *r* is user defined, it is adaptive.

Selection of solutions for the next iteration is as follows. The combined 2N population of parent and offspring is sorted for non-domination. If the number of non-dominated fronts in the 2N population is K, then according to the geometric distribution, the maximum number of individuals allowed in the i-th front (i=1,2,…….,K) in the new population of size N is

$$N_i = N\frac{1-r}{1-r^K}\, r^{i-1} \tag{5}$$

Since r<1, the allowable number of individuals in front one is the highest. Thereafter, each front is allowed to have an exponentially reducing number of solutions. Other distributions such as arithmetic distribution, harmonic distribution are also possible. It is clear that the new population under the modified algorithm is more diverse than that under NSGA II.

Although Equation (4) denotes the maximum number of individuals Ni in each front, there may not exist exactly N_i individuals in such fronts. This is resolved using the following procedure. First the number of individuals in the first front is counted. Assuming there are N_1^t individuals, if $N_1^t > N_1$ then N1 solutions are chosen using the crowded tournament selection. Hence N1 solutions residing in the less crowded region are selected. Otherwise, if $N_1^t \leq N_1$, all N_1^t solutions are chosen and the number of remaining slots $X_1 = N_1 - N_1^t$ is counted. The maximum number of individuals in the second front is now increased to $N_2 + X_1$. Thereafter, the actual number of solutions N_2^t in the second front is counted and compared with N_2 as above. This procedure is repeated till N individuals are selected.

However, there could be some case that after all 2*N* solutions are processed, the size of new population cannot reach N. The new population still has some space needed to be filled, especially when *r* is large. In such case, filling the population again continues with the remainder individuals from the first rank, continue to other ranks, and include them until the size of new population reaches to *N*.

4 Test Results

The proposed NSGA-II algorithm with controlled elitism for scheduling tasks was implemented in C programming language on a PC with Pentium dual processor running under Linux environment. To measure the effectiveness and viability of NSGA-II algorithm with controlled elitism results are compared with simple NSGA-II algorithm.

4.1 Test Problems

To assess the comparative performances of the algorithms, the simulation model in [4] 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 an m×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) × $x_r^{i,k}$. The vector B is not used in the actual matrix. This process is repeated for each row until the m × n matrix is full. Therefore, any given value in the ETC matrix is within the range (1, ϕ_b , ϕ_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.

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).

4.2 Performance Measure

For all the test instances with NSGA-II and NSGA-II with controlled elitism the parameters used are given in Table 1.

Table 1 : Test parameters

Parameters	Values (type)
Population size	100

Number of iteration	1000
Pc, crossover probability	0.8
Pm, mutation probability	0.02
Cross over index η_c	2 (SBX crossover)
Mutation index η_m	20(Polynomial mutation)
r, controlled elitism	0.65(geometric distribution)

Since the diversity among optimized solutions is an important matter in multi-objective optimization, a measure based on the consecutive distances among the solutions of the best non-dominated front in the final population is used. The obtained set of the first non-dominated solutions are compared with a uniform distribution and the deviation is computed as follows [18]

$$\Delta = \sum_{i=1}^{|F_1|} \frac{|d_i - d|}{|F_i|} \quad (6)$$

In the above equation, *di* is the Euclidean distance between two consecutive solutions in the first non-dominated front of the final population in the objective function space. The parameter *d* is the average of these distances.

The deviation measure Δ of these consecutive distances is calculated for each run. An average of these deviations over 5 runs is calculated as the measure ($\bar{\Delta}$) for comparing the two algorithms. Thus, it is clear that an algorithm having a smaller $\bar{\Delta}$is better, in terms of its ability to widely spread solutions in the obtained front. Table 2 shows the deviation from a uniform spread ($\bar{\Delta}$) in 5 independent runs obtained using NSGA-II and NSGA-II with controlled elitism.

The entire non-dominated front found by NSGA-II and NSGA-II with controlled elitism is given for four instances out of the 12 instances considered in Fig 1-4. This helps in better understanding of how the solutions are spread over the non-dominated front.

Since flowtime has a higher magnitude over makespan, its difference increases as more jobs and processors are considered. For this reason, the value of mean flowtime, flowtime/number of machines, is used to evaluate flowtime. The values of makespan and mean flowtime are measured in same time units. The values obtained for the non-dominated solutions are plotted in ten thousands of units in Fig 1-4.

Table 2 : Comparison of NSGA –II and NSGA-II with controlled elitism

Instances	NSGA-II	NSGA-II –CE

	($\bar{\Delta}$)	($\bar{\Delta}$)
u_c_lo_lo	0.000053	*0.000036*
u_c_lo_hi	0.0146	*0.0092*
u_c_hi_lo	0.0034	*0.0016*
u_c_hi_hi	0.5761	*0.3897*
u_s_lo_lo	.00012	*.00009*
u_s_lo_hi	.0065	*.0063*
u_s_hi_lo	.0031	*.0029*
u_s_hi_hi	.5489	*0.4867*
i_s_lo_lo	.0002	*.00007*
i_s_lo_hi	.5532	*.0278*
i_s_hi_lo	.0072	*.0045*
i_s_hi_hi	.9769	*.4167*

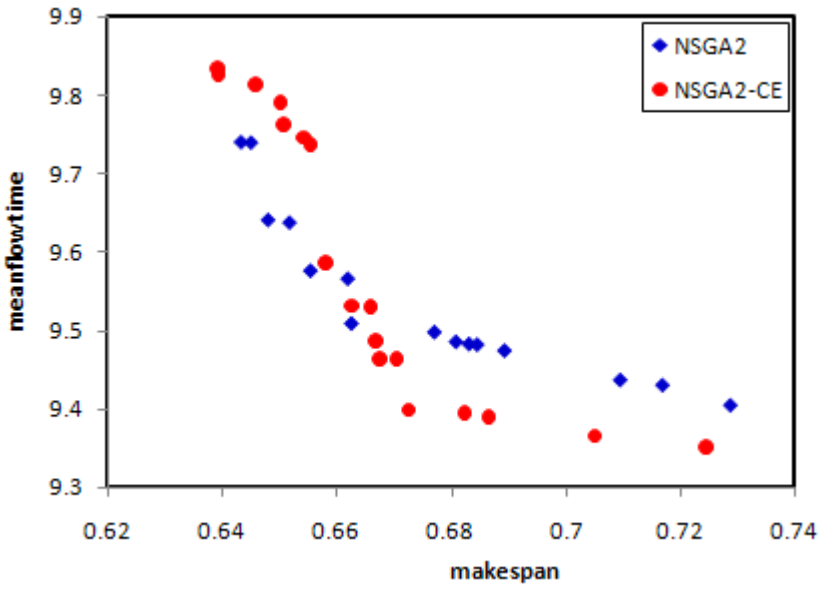

Fig 1. Non-dominated solutions obtained on instance u_c_lo_lo

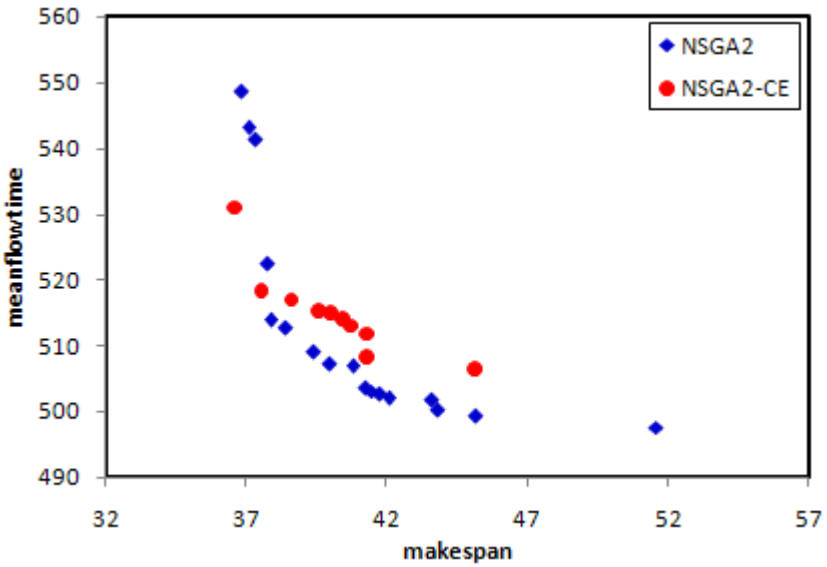

Fig 2. Non-dominated solutions obtained on instance u_c_lo_hi

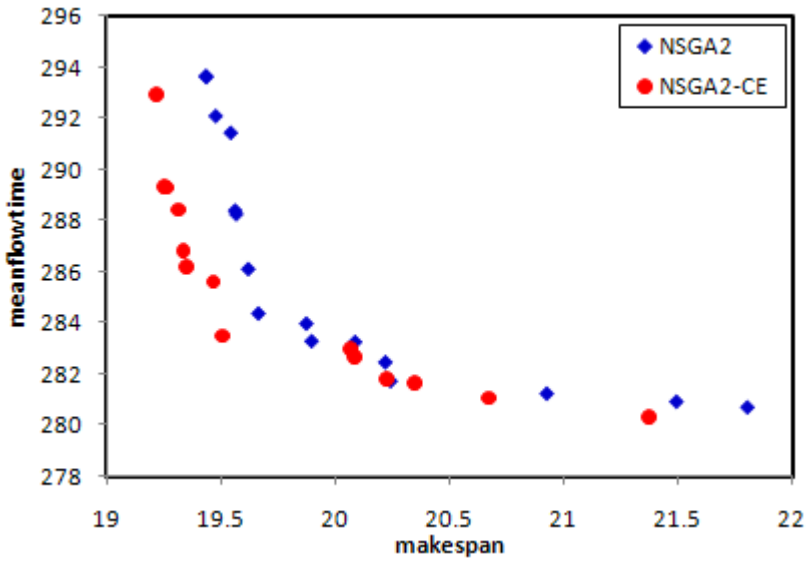

Fig 3. Non-dominated solutions obtained on instance u_c_hi_lo

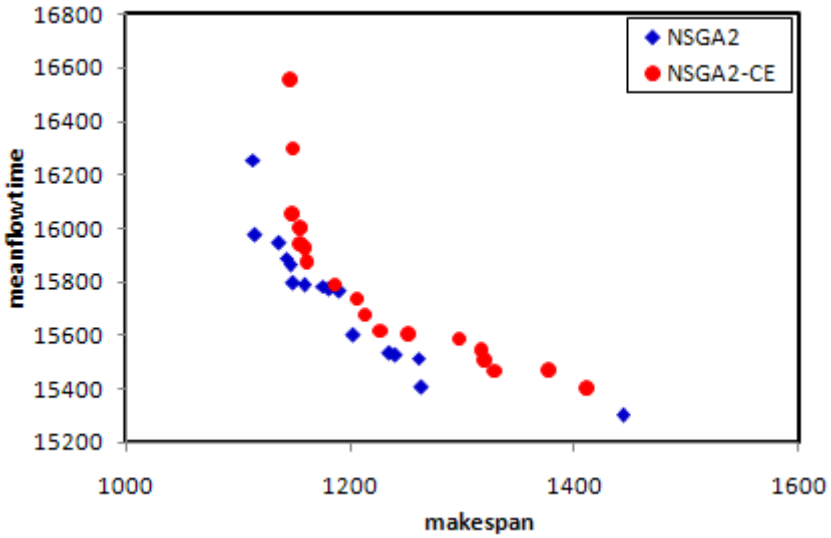

Fig 4. Non-dominated solutions obtained on instance u_c_hi_hi

Further in NSGA-II with controlled elitism it can be seen that the number of fronts are more when compared to NSGA-II which shows that the algorithm preserves good lateral diversity as shown in Fig 5. It is also found that still thousands of iterations are required to have more number of solutions in the first front in NSGA-II.

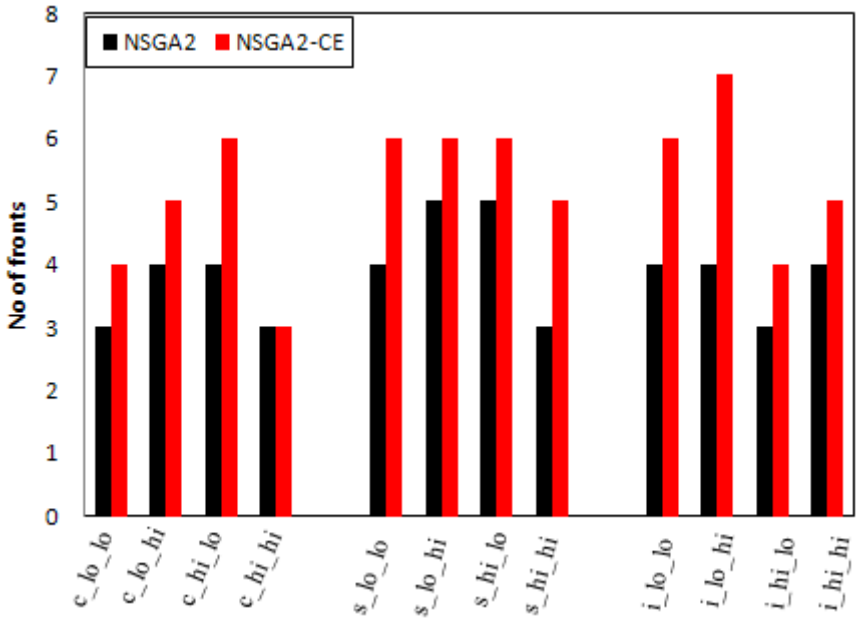

Fig 5. Number of fronts in NSGA-II and NSGA-II with controlled elitism

5 Conclusion

In this paper a modified version of NSGA-II is implemented for task scheduling and compared with NSGA-II. It is found that the quality of schedules achieved by both the algorithms is almost same. Comparing the performance of both the algorithms it is seen that NSGA-II with controlled elitism maintains a uniform spread of solutions in the obtained non-dominated front. Even though NSGA-II uses crowding tournament selection to ensure diversity it does not preserve diversity lateral to the Pareto–optimal front. This is achieved by controlling elitism in NSGA-II. Since the parameter *r* controls the extent of exploration, investigation can be extended in choosing a value of *r* where the algorithm could perform still better on the task scheduling problem. This concept of controlled elitism could also be tried on other algorithms for this problem to study its performance.

6 Open Problem

Traditional methods for determining optimal schedules do not provide exact solutions and requires an exhaustive search if the distributed system is large. Many research works using heuristic techniques are being undertaken to solve this scheduling problem. This problem also requires several criteria to be satisfied while scheduling tasks on heterogeneous environments. This paper proposes a static scheduling technique whereas there are problems available that deal with dynamic scheduling. Moreover this paper considers tasks that are independent of each other whereas the problem can be analyzed for tasks which depend on other tasks. The GA parameters applied for the problem also may be analyzed for better performance.

References

[1] S. Ali, T. D. Braun, H. J. Siegel, and A. A. Maciejewski, *"Heterogeneous computing",* Encyclopedia of Distributed Computing, Kluwer Academic, (2001).
[2] A. Abdelmageed Elsadek, B. Earl Wells, "A Heuristic model for task allocation in heterogeneous distributed computing systems," *The International Journal of Computers and Their Applications*, Vol. 6, No. 1, (1999).
[3] M.R. Garey and D.Johnson, *Computers and Intractability: A Guide to the theory of NP-Completeness*. Freeman and Company, San Francisco,(1979).
[4] 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*, 61(6), (2001).
[5] H.Izakian, A. Abraham and V. Snasel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments", *In*

Proceedings of International Joint Conference on Computational Sciences and Optimization 1,8-12, (2009).
[6] E.U. Munir, Jian-Zhong Li, Sheng-Fei Shi, Q. Rasool, " Performance Analysis of Task Scheduling Heuristics in Grid" *In: ICMLC '07: Proceedings of the International Conference on Machine Learning and Cybernetics*, Volume 6, Issue 19-22, (2007) ,pp. 3093 – 3098.
[7] 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).
[8] 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.
[9] Javier Carretero, Fatos Xhafa and Ajith Abraham, "Genetic Algorithm Based Schedulers for Grid Computing Systems," *International Journal of Innovative Computing, Information and Control* 3, 6, (2007).
[10] 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.
[11] F. Xhafa, A. Abraham, "Meta-heuristics for grid scheduling problems". *Studies in Computational Intelligence*, Springer Verlag: Heidelberg, Germany, (2008), pp. 1–37.
[12] 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 4252, Springer (2006), pp. 500-507.
[13] 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.
[14] C.A.C. Coello, *"A comprehensive survey of evolutionary-based multiobjective optimization techniques,*" Knowledge and Information Systems, 1, (1999), pp. 269–308.
[15] R.C. Purshouse, and P.J. Fleming, "Elitism Sharing and Ranking Choices in Evolutionary Multi-Criterion Optimisation," Research Report No.815, Department of Automatic Control and System Engineering, University of Sheffield, (2002).
[16] Kalyanmoy Deb, *"Multi-objective Optimization using Evolutionary Algorithms,"* John Wiley and Sons Ltd, 2002.

[17] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan , "A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, 6 (2002), pp.182–197.
[18] K.Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Proceedings of the Parallel Problem Solving from Nature VI conference*, Paris, (2000), pp. 849-858.
[19] S.Padmavathi, S.Mercy Shalinie and R.Abhilaash, "A Memetic Algorithm Based Task Scheduling considering Communication Cost on Cluster of Workstations," *International Journal of Advances in Soft Compuing and its Appications*, Vol. 2, No. 2, (2010),pp 174-190