Scheduling Independent Tasks in Distributed Computing Systems Using NSGA-II

Sarathambekai
{*ssi.it@psgtech.ac.in }

Assistant Professor (Selection Grade), Department of Information Technology, PSG College of Technology, Coimbatore

Abstract. Task scheduling in heterogeneous multi-processor environments is a complex and challenging issue. This problem in distributed environment is identified to be the Multi-objective Optimization Problems (MOPs), involving the simultaneous fulfillment of several objectives. The complexity of solving MOPs using traditional combinatorial methods are high and hence the optimal solutions can be achieved by using Evolutionary Algorithms. This paper presents a NSGA –II (Non-dominated Sorting Genetic Algorithm) for an efficient scheduling of tasks in a heterogeneous multiprocessorsenvironment. Most of the existing research in this area considered task scheduling with a single objective or bi-objectives only. This scheduling problem is a minimization problem with multiple objectives considering three objectives namely Makespan (Completion Time), Flow Time (Response Time) and Reliability Cost(Fault Tolerance). This method is compared with existing Weighted Sum method (WS). From comparative analysis, NSGA-II provided better performance than WS in twelve different types of heterogeneous environment.

Keywords: Scheduling, Multiple objectives, Optimization problem, NSGA, Combinatorial methods.

1 Introduction

In traditional computing systems, there will be a single processor which is responsible for satisfying all computational requirements. When the number of tasks to be serviced becomes high, most of the tasks have to wait in queues for processors. As the computational requirements grow day by day, multi-processor systems are preferred over single-processor systems. A multi-processor environment comprises of a set of homogeneous or heterogeneous processors having same or different memory and processing capabilities, connected together using different network topologies. The different tasks generated are scheduled in those processors based on availability and requirements. The common challenging issue of multi-processor scheduling can be defined as allocation of tasks on a multi-processor environment, so as tooptimize multiple criteria like completion time [1, 11].

Scheduling problems can be seen as Multi-objective Optimization Problem (MOPs). A MOP is satisfying more than one conflicting objectives simultaneously. The general objectives with respect to multi-processor scheduling may be Makespan, Reliability Cost, Flow Time, Load Balancing, Deadline Missing Time etc. Solving MOPs using traditional methods yield high complexity. Therefore using evolutionary methods for solving MOPs is preferred over traditional methods.

ICCAP 2021, December 07-08, Chennai, India

DOI 10.4108/eai.7-12-2021.2314587

The algorithm used in this paper[2, 12, 13] forscheduling problem is NSGA-II, a variant of Genetic algorithm (GA). In contrast to the weighted or aggregated sum approach, traditionally used in GAs, non-dominated sorting technique is been used, to ensure reaching Pareto-optimal solutions. Pareto-optimality is a concept in multi-objective problem solving, where the improvement of one objective should not worsen the other objectives. The solutions achieved which cannot be further improved, without worsening any of the objectives are called Pareto-optimal solutions [2, 12, 13].

The general approach for multi-processor scheduling, considers either one or two objectives. In this paper, three conflicting objectives are considered namely makespan, flow time and reliability cost, which are to be minimized. The experiment is simulated and the results are compared with existing techniques for the same constraints. The term chromosome denotes the representation of solution in GA and is interchangeable with the term solution.

The organization of the research paper asfollow: Section 2 explains Problem definition. Related works are discussed in Section 3.Section 4presents the algorithm design. Parameter setup and simulation results are elaborated in Section 5 and 6 respectively. The conclusion part of the paper is given in Section 7.

2 Problem Formulation

Distributed system is a collection of numerous processors connected with each other. If there are R tasks which are independent, Task = {$Task_1$, $Task_2$…, $Task_R$} that are to be allocated on S processors Processor = {Pr_1, Pr_2..., Pr_S}. The expected execution time of a same task running on different processors is not same because processors are heterogeneous. An ETC matrix is an R x S matrix in which Rrepresents total number of tasks and Srepresentstotal number of processors. A row in the ETC matrix indicates the estimated execution time for a given task on each machine. Similarly columns in the ETC matrix indicate that the estimated execution time of a given machine for each task.

The problem is framed under the following assumptions:

- The execution time values are known in advance.
- Tasks are considered as non-preemptive.
- A processor can serve a single task at a time.
- At a time, every task is scheduled to a single processor.

The work considered in this paper is the multi-objective optimization problem of allocating independent tasks on a set of available multiple processors in the distributed systemsin order to optimize Makespan, Reliability Cost and Flow Time simultaneously.

The common standard optimization criterion is minimization of completion time (makespan) which is the total execution time of all tasks. Consider that $C_{i,j}$(i ε{1,2,...,R}, j ε {1,2,...,S}) is the execution time for performing i^{th} task in j^{th} processor and W_j (j ε {1,2,...,S}) is the previous workload of P_j. Then Makespan can be calculatedusing the Eq. (1)

$$Makespan = max\{\sum C_{i,j} + W_j\}, j\varepsilon\{1, 2 \dots S\} \quad (1)$$

Reliability Cost (RC) is factor that defines the predicted rate of failure with respect to each processor that it is executing the tasks. Processor failures are considered to be independent in this paper, and it uses a Poisson process with a constant failure rate. Since the tasks are

independent, failure in communication links between the processors are not considered. The RC of a task $\mathbf{T_i}$on a processor $\mathbf{P_j}$ is the product of $\mathbf{P_j}$'s failure rate (PFR) λj and$\mathbf{T_i}$'s execution time on **j**. The Reliability Cost can be defined in Eq. (2), where $\mathbf{pr}(\mathbf{T_i}) = \mathbf{j}$represents that task $\mathbf{T_i}$is allocated to $\mathbf{P_j}$

$$ReliabityCost = \sum_{j=1}^{S} \sum_{pr(T_i)=j} \lambda_j C_{ij}(T_i) \quad (2)$$

Flow time is manipulated as the summation of the finishing times of all tasks. It is used to measure the quality of service of the computing system. This can be defined in Eq. (3), where $F_{i,j}$ is the finishing time of task$T_i$on a processor P_j

$$FlowTime = \sum_{j=1}^{S} \sum F_{i,j} \quad (3)$$

3 Related Works

Multi-objective scheduling problems are NP-hard generally. That is these scheduling problems are non-deterministic and it requires some polynomial time to solve. There have been many researches going on in such scheduling problems worldwide. Kalyanmoy Deb [3] proposed a fast non-dominated sorting approach. The existing genetic algorithms had a complexity of $O(MN^3)$ where M representstotal number of objectives and N represents the size of the population. This high computational complexity is due to the comparisons made in non-dominated sorting. The proposed approach with $O(MN^2)$ computational complexity, can give optimal solution and better results when compared to other Evolutionary Algorithms such as PAES and SPEA.

The modified version of NSGA-II [2, 14] was recommended for task scheduling considering Makespan and Flow Time and compared it with NSGA-II. From the results, the author found that the algorithm with controlled elitism preservesuniformly distributed solutions in the obtained non-dominated front. RituVermaet al., [4] implemented a GA for optimizing completion time in scheduling independent tasks multi-processor systems in a static environment. Rio et al.,[5] proposed a modified version of NSGA-II algorithm with lower runtime complexity. This algorithm used a faster non-dominated sorting procedure that sorts individuals based on each of the objectives, one after the other, till all objectives are considered, finding a position value for each individual. H.Izakian, A. Abraham [6] analyzed the well-known heuristics algorithms in scheduling of independent tasks on distributed systems based on makespan and flow Time and also suggested that using a min-max heuristic provided better results than other heuristics.

NSGA-II [7, 15] algorithm for solving multiobjective optimization problems with the performance measures task execution time, and the task transfer time in static environments was discussed. The author proved NSGA II provided better results than the multi-objective PSO.Kamaljit Kaur et al., [8] developed a new variant of genetic algorithm named Heuristics based Genetic Algorithm in homogeneous parallel system for static tasks scheduling. The author considered the minimization of completion time and increased the throughput of the system.

MyungryunYoo, Mitsuo Gen [9] designed a new scheduling algorithm called multi-objective hybrid GAfor allocating real-time tasks on heterogeneous multi-processorssystem. The author co-operatively used the GA and the simulated annealing for optimizing the delay and

completion time. Peng-Yeng Yin [10] has developed a Hybrid PSO which optimizes the Reliability of the multiprocessor systems.The HPSO embeddedwith local search heuristic for improving the diversity and convergence of the algorithm.Most of the works are concentrating on two objectives simultaneously for various task scheduling problems in distributed systems. A very few research works have been implemented using three objectives. This paper presents tri-objective optimization for scheduling tasks in multi-processor environments using NSGA-II.

4 Design

This work presented in this paper uses the NSGA-II, a variant of GA. The steps followed in NSGA-II are

- Population Initialization

Population is the set of all feasible solutions that can be arrived at, for a given problem. The first step of the algorithm is to initialize the population. Population initialization is providing all the initial possible solutions on which the genetic operations are to be performed.

- Objective Function Evaluation

The objective functions are to be evaluated to search the best solutions. The objective functions are with respect to Makespan, Flow Time and Reliability Cost.

- Variation

New offspring are generated by variation. Variation is an operation that causes some changes to the parent chromosomes to yield new chromosomes. Variation can be performed using crossover and mutation.

- Non-dominated Sorting

After the offspring are been created, the total solutions become 2 x N consisting both the N parents and the newly created N offspring. The solutions are to be filtered to a set of best N solutions. Non-dominated sorting is performed for this purpose.

- Calculating Crowding Distance

Many solutions may be of same rank and there may be a tie in choosing the solutions for next generations. Crowding distance is used to choose the best solution in case of a tie.

These steps are applied in iterative phases. That is, after calculating crowding distance and forming new generation, steps from objective evaluation through crowding distance calculation continues. This iterative process gets over after termination conditions, such as optimal solution or certain number of generations, is achieved. The algorithm is described in Fig.1. The phases are explained in detailed below:

4.1 Population Initialization

This is a scheduling problem consisting of solutions that are schedules of tasks that are to be allocated to the processors. Therefore populations are represented using permutations. In Permutation-based encoding, the solution is a collection of numbers that indicates a position in a sequence. Thistype ensures that no task is allocated to more than one processor.

4.2 Tri-Objective Calculation

The objectives are evaluated in order to find whether the chromosome is eligible for further process. Three objective functions are been used, which are Makespan - the completion time of all tasks as in Eq. (1) ; Flow Time - total execution time with respect to each task as in Eq. (2); Reliability Cost - factor that defines the predicted rate of failure with respect to each processor as in Eq. (3).

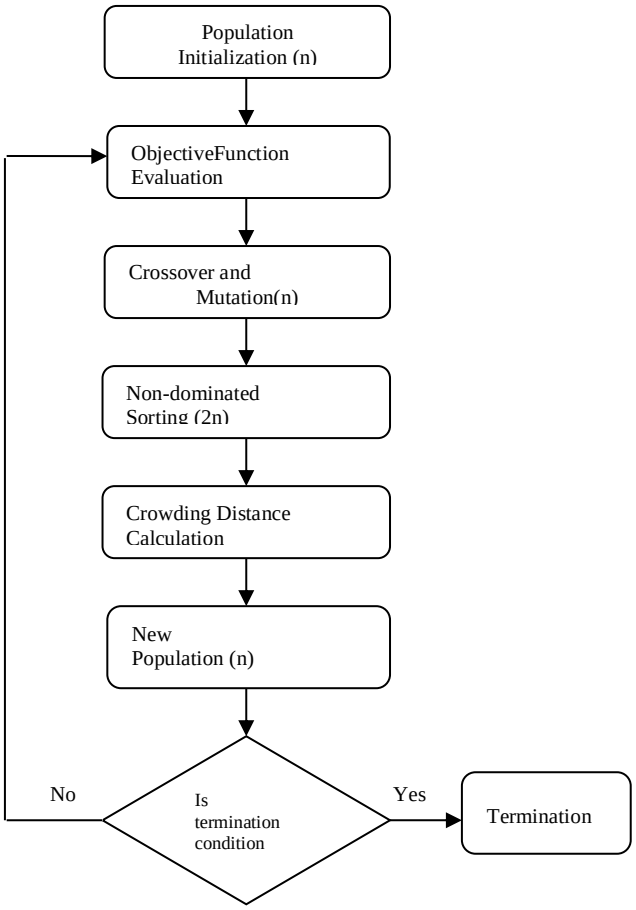

Figure 1 Flowchart of NSGA II for task scheduling problem

4.3 Variation

Variation is a method to produce offspring from one or two parents. Variation can be performed by either crossover or mutation. Combining crossover and mutation can bring better results as mutation helps in maintaining diversity.

Crossover is recombining the parents to get new offspring. It needs two parents and generates two or more offspring. Crossover method and crossover points are to be chosen. C1 crossoveris the modified standard single point crossover that is used in this paper. Mutation is performed to preserve diversity of the algorithm and to avoid premature convergence. Shift mutation has been used in this paper. In shift mutation, the task at select point is removed and the tasks till the insert point are shifted by one position towards the select point. The task removed is placed at the insert point, where select point and insert point are randomly selected positions.

4.4 Non-dominated Sorting

After variation, N numbers of offspring are created from N number of parents. Therefore the number of solutions will become 2xN. Out of this, best N solutions are chosen by Non-dominated sorting for moving to the next generation.

In contrast to the traditional weighted sum method, non-dominated sorting ensures no objective is completely dominating any other objectives. Non-dominated sorting helps us to achieve Pareto-optimal solutions. The solutions are ranked based on the number of solutions it dominates and the number of solutions which dominates it. The N solutions with best ranks get a chance to survive to the next generation.

4.5 Calculating Crowding Distance

Among the 2N solutions, the best N solutions become the parents in next generation. After non-dominated sorting, two or more solutions may be in same rank. In sorting out the best N solutions, there may be a tie between two or more solutions that reside in the same rank. The crowding distance (CD) is done by arranging the set of chromosomesin ascending order based on objective values. The CD value of a particular chromosome is calculated by taking the average distance of its two adjacent chromosomes. The boundary solutions have the lowest and highest objective values. These are set to infinite crowding distance values so that they constantly get selected. This procedure is performed for each objective function. To find out the best solution in such cases, crowding distance of those solutions are calculated and the one having the least crowding distance is awarded the next step.

The result of the NSGA-II algorithm is compared with the weighted sum method. The procedure to calculate the weighted sum is explained as follows:

The weighted sum method follows the same algorithm except the non-dominated sorting. Instead of non-dominated sorting, the solutions based on three objectives are consolidated giving weights to each objective. Then the objective function will become a single value. The calculation of weights W_1, W_2 and W_3 for three objectives is performed using Eq. (4), Eq. (5) and Eq. (6) respectively.

$$w_1^i(t) = random(\lambda)/\lambda \quad (4)$$

$$w_2^i(t) = (1 - w_1^i(t)) * random(\lambda)/\lambda \quad (5)$$

$$w_3^i(t) = 1 - w_1^i(t) - w_2^i(t) \quad (6)$$

Here w_1, w_2, w_3 are the respective weights of the three objectives. The function random (λ) produces a uniformly distributed random number between 0 and λ, i = 1... λ¸ Where λ is the population size, and t is the iteration index. The fitness function is calculated using the formula as in Eq. (7).

$$\text{Minimize } ws^i(t) = w_1^i(t) * Ms + w_2^i(t) * RC + w_3^i(t) * FT \quad (7)$$

Where Ms is the makespan value, RC is the Reliability Cost value and FT is the Flow Time value and wsis the fitness value based on aggregated sum method of the chromosome.

5 Experimental Setup

The experiment is simulated in C language in Linux platform considering 10 processors and 30 tasks for thousands of generations. Table 1 gives the population size, crossover and mutation details.

Table 1 Test Parameters

Parameters	Values
Size of the population	50
Iteration Count	1000
crossover and mutation probability	0.5

The execution time of the tasks is represented using the Expected Time to Compute (ETC) matrix. The processors considered here are heterogeneous in two ways i) Machine heterogeneity [1] which is a value that represents the variation in execution time of a task in different processors ii) Task heterogeneity [1] which represents the variation in execution time different tasks in a processor. Consistency of the processors is also considered. The processor may be consistent, inconsistent or semi-consistent. ETC for each task is derived from various heuristics with respect to consistency, machine and task heterogeneities.

The ETC matrix values for inconsistent type are completely random in NSGA-II. The consistent and semi-consistent ETCs are obtained from the inconsistent using same method for NSGA-II and weighted sum method. For consistent ETC, the values of the matrices are arranged in ascending order based on the first objective for all columns. For semi-consistent ETC, the values for certain columns are arranged in ascending order based on the first objective and the remaining columns are left as before.

Therefore there are twelve cases of ETC matrices which are used for experimentation.

6 Implementation Results

The algorithm is been simulated for thousand generations and the results are plotted. The Figure 2 to Figure 8 represents the best solutions collected from weighted sum and NSGA-II at thousandth generation. The three axes denote the three objectives. In these figures, ns refer to non-dominated sorting and ws refer to weighted sum method.

The results are prepared based on task and machine heterogeneities of the processors. Figure 2 presents the solutions of low task and low machine heterogeneity for weighted sum and NSGA-II with respect to inconsistent ETC.

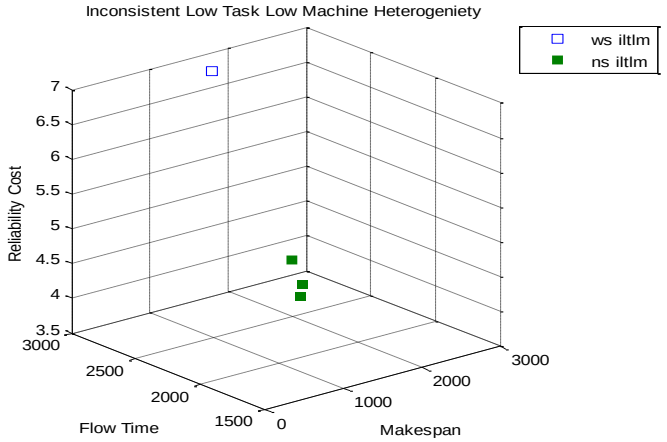

Figure 2 NSGA-II and weighted sum solutions inconsistent low task and low processor heterogeneity

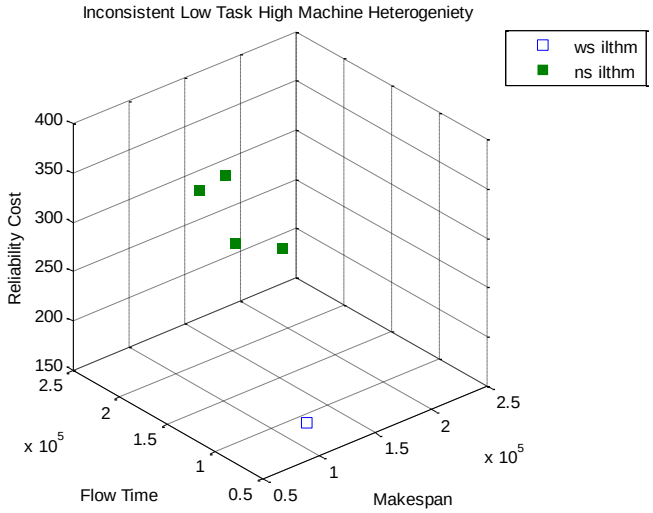

Figure 3 NSGA-II and weighted sum solutions inconsistent low task and high processor heterogeneity

Figure 3 presents the solutions of low task and high processor heterogeneity for weighted sum and NSGA-II with respect to inconsistent ETC.

Figure 4 presents the solutions generated at thousandth generation of high task, high processor heterogeneity for weighted sum and NSGA-II with respect to inconsistent and ETC.

The solutions from NSGA-II show signs of Pareto-optimality and lesser objective function values than those of weighted sum method.

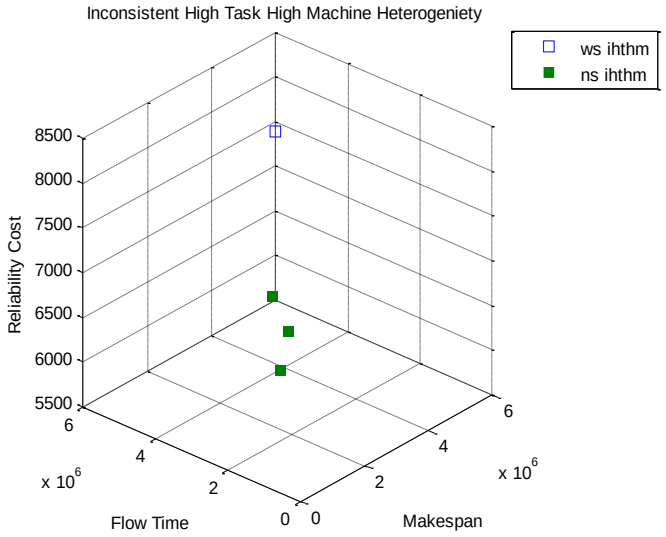

Figure 4 NSGA-II and weighted sum solutions inconsistent high task and high processor heterogeneity

Figures 5 and 6 presents the solutions generated at thousandth generation of low task, low processor heterogeneity and low task, high processor heterogeneity for weighted sum and NSGA-II with respect to consistent, ETC, respectively.

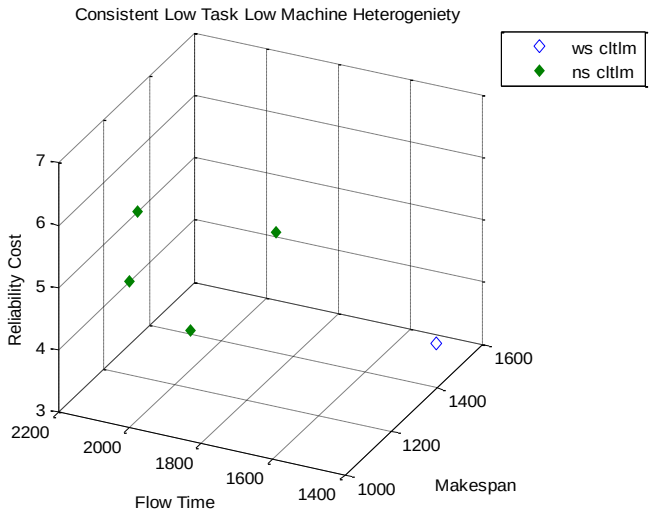

Figure 5 NSGA-II and weighted sum solutions consistent low task and low processor heterogeneity

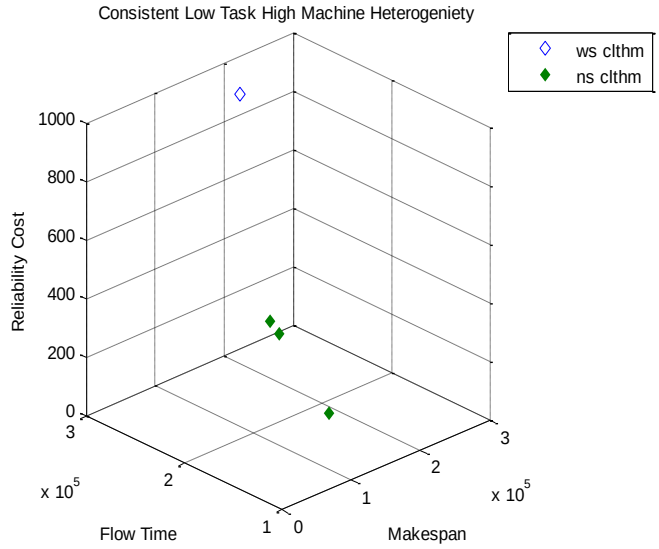

Figure 6 NSGA-II and weighted sum solutions consistent low task and high processor heterogeneity

Figures 7 and 8 present the solutions generated at thousandth generation of low task, high processor heterogeneity and high task, high processor heterogeneity for weighted sum and NSGA-II with respect to consistent, ETC, respectively.

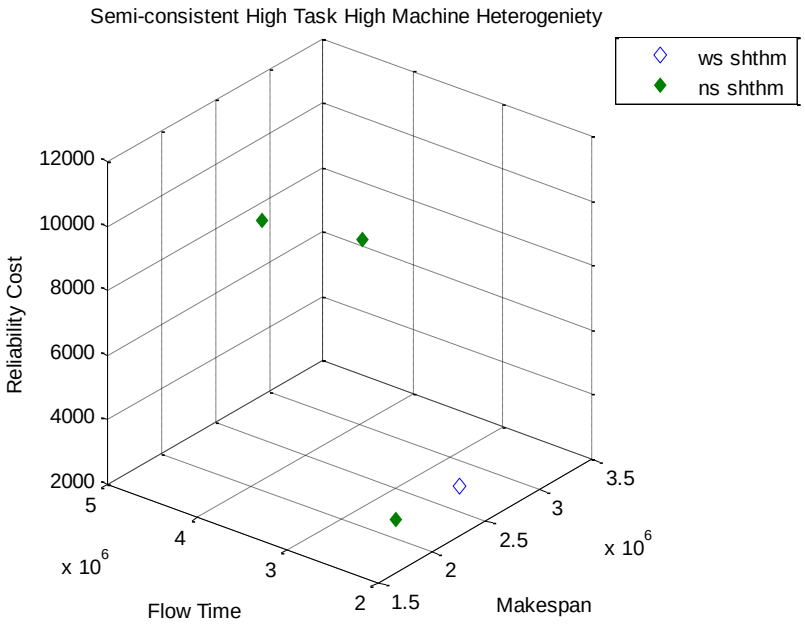

Fig 7 NSGA-II and weighted sum solutions semi-consistent high task and high processor heterogeneity

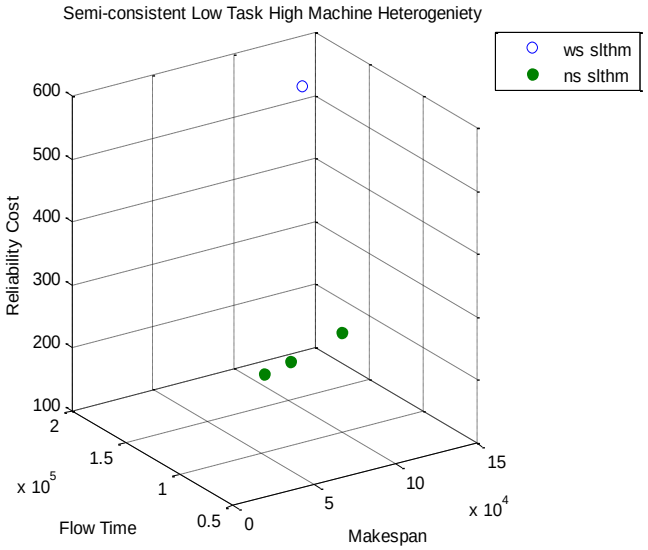

Figure 8 NSGA-II and weighted sum solutions semi-consistent low task and high processor heterogeneity

From the figure 7 and figure 8, it is inferred that the solutions from NSGA-II show signs of Pareto-optimality and lesser objective function values than those of weighted sum method.

Conclusion

An efficient scheduling of tasks in a multiprocessor system using NSGA-II is presented in this paper. The experimental results proved that the NSGA-II with non-dominated sorting technique provided better results than weighted sum method in all the twelve types of heterogeneity with respect to tasks and machines. The algorithm provided solutions with objective values that are much minimized with respect to multiple objectives such as makespan, flow time and reliability cost simultaneously. Results are presented in the graph shown that the Pareto-optimality by the curves formed by the solutions of NSGA-II. This shows no objective is given priority over the others and the solutions are Pareto-optimal. The work can be further extended to test with large number of tasks with processors and also for dependent tasks, which takes the problem to a different context.

References

[1] M.Maheswaran, S.Ali and H.J.Siegal, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", Heterogeneous Computing Workshop, 1999

[2] G.Subashini and M.C.Bhuvaneswari, "NSGA - II with controlled elitism for scheduling tasks in heterogeneous computing systems", International Journal for Open Problems Computing, Volume 4, 2011

[3] Kalyanmoy Deb, "A fast and elitist Multiobjective Genetic Algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, volume 6, Issue 2, 2002

[4] RituVerma, SunitaDhingra, "Genetic Algorithm for multiprocessor task scheduling", International Journal of Computer Science and Management Studies, Volume 11, Issue 2, 2011

[5] Rio G.L. D'Souza, K.ChandraSekaran and A. Kandasamy, "Improved NSGA-II based on a novel ranking scheme", International Journal Of Computing, Volume 2, Issue 2, 2010

[6] H.Izakian, A.Abraham, Snasel, "Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments", International Joint Conference on Computational Sciences and Optimization, 2009

[7] Sun Yijie, Shen Gongzhang, "Improved NSGA-II Multi-objective Genetic Algorithm based on Hybridization-encouraged mechanism", Elsevier Chinese Journal of Aeronautics, Volume 21, Issue 6, 2008

[8] Kamaljit Kaur, Amit Chhabra and Gurvinder Singh, "Heuristics based Genetic Algorithm for scheduling static tasks in homogeneous parallel system", International Journal of Computer Science and Security, Volume 4, Issue 2, 2009

[9] MyungryunYoo and Mitsuo Gen, "Scheduling Algorithm for real-time tasks using Multiobjective Hybrid Genetic Algorithm in heterogeneous multiprocessors system", Elsevier Journal of Computers & Operations Research, 2007

[10] Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, Yi-Te Wang, "Task allocation for maximizing reliability of a distributed system using Hybrid Particle Swarm Optimization", Elsevier Journal of Systems and Software, 2007

[11] G.E. Weiqing, Cui Yanru, “Task-scheduling Algorithm based on Improved Genetic Algorithm in Cloud Computing Environment”, Recent Advances in Electrical & Electronic Engineering, Volume 14 , Issue 1 , pp. 13 – 19, 2021

[12] S.Velliangiri, P.Karthikeyan, V. M.ArulXavier,, D.Baswaraj, “Hybrid electro search with genetic algorithm for task scheduling in cloud computing”, Ain Shams Engineering Journal, Volume 12, Issue 1, pp. 631-639,2021

[13] Gilberto Rivera, Luis Cisneros, Patricia Sánchez-Solís, Nelson Rangel-Valdez, Jorge Rodas-Osollo,“Genetic Algorithm for Scheduling Optimization Considering Heterogeneous Containers: A Real-World Case Study”,MDPI, pp 1-16, 2020 doi:10.3390/axioms9010027

[14] M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. J. Ryan and K. -K. R. Choo, "An Automated Task Scheduling Model using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems," in IEEE Transactions on Cloud Computing, 2020, doi: 10.1109/TCC.2020.3032386.

[15] HaniehGhorashi, MeghdadMirabi, “An Effective Task Scheduling Framework for Cloud Computing using NSGA-II”, Volume 6, Issue 3, pp. 151-160, 2020