Multiple-Objective Particle Swarm Optimization Algorithm for Independent Task Scheduling in Distributed Heterogeneous Computing Systems

Amit Prakash*, Karamjit Bhatia**, Raj Kumar***

*Research Scholar, Department of Computer Science, Gurukula Kangri Vishwavidyalaya, Haridwar, India
**Professor, Department of Computer Science, Gurukula Kangri Vishwavidyalaya, Haridwar, India
***Assistant Professor, Department of Computer Science, Gurukula Kangri Vishwavidyalaya, Haridwar, India

Abstract- *Task scheduling is a crucial issue in distributed (disbursed) heterogeneous processing environment and significantly influence the performance of the system. The task scheduling problem has been identified to be NP-complete in its universal frame. In this paper the task scheduling problem is investigated using multiple-objective particle (molecule) swarm optimization algorithm with crowded displacement operator (MOPSO-CD). Particle swarm optimization is a populace based meta-heuristic which mimics the convivial conduct of feathered creatures running. In this algorithm particles move in the problem's search space to achieve near optimal solutions. The performance of this algorithm is compared with non-domination sorting genetic algorithm II (NSGA-II). The proposed scheduling algorithm intends to find the near optimal solution with aim to minimize the make-span and flow time. The exploratory results demonstrate that the proposed multi-objective PSO algorithm is more productive and gives better outcomes when contrasted with those of NSGA-II.*

Keywords- *Task scheduling, Independent tasks, Meta heuristic, Particle swarm intelligence, Non-domination sorting genetic algorithm, Make-span, Flow-time.*

I. INTRODUCTION

A heterogeneous computing (HC) environment makes utilization of available group of processors associated with fast systems to undertake applications having changed computational necessities. Heterogeneous computing environment extends from various components inside a solitary PC, to a group of PCs, to a more progressed geologically circulated machine, with shifted models. To viably and effectively use the computing capacities of such frameworks, scheduling is a key issue that must be properly addressed. The scheduling method involves the ordering of task execution and its mapping onto processors so as to minimize certain objective.

In optimum scheduling process we map a pool of tasks to a set of resources to harness the computing abilities of such diverse systems so that some measure of effectiveness is optimized. Optimal mapping of a set of tasks to available processing elements in a HC framework is found to be a NP-complete problem. In view of the kind of tasks to be scheduled, scheduling issue might be classified into two categories - Scheduling tasks (meta-tasks) independently, and scheduling coordinated directed non-cyclic graphs comprising of communicating tasks having priority (precedence) relation. The present work is confined to task scheduling belonging to first category only i. e. tasks submitted by various users independently to various assets of a HC suite. Several criterions might be utilized for assessing the proficiency of a scheduling algorithm, the most vital of which are make-span and flow-time [11]. Make-span is the completion time of the last task, while flow-time is the time that gives aggregate of processing times of all tasks. In present work an attempt is made to obtain an optimum schedule that minimizes both make-span and flow time.

Particle Swarm Optimization (PSO) algorithm comes under the category of swarm intelligence and is a population based optimization technique [27]. Its fruitful application incorporates standard function optimization [22], solving permutation complications [23], and training multilayer neural systems [24]. PSO algorithm consists of a swarm of particles in which every particle (molecule) points to a potential solution to the problem. In contrast with evolutionary algorithmic approaches a swarm is like a populace, and a molecule is like a distinguishable entity (chromosome) [13]. The particles move through a multidimensional search space where the position of every molecule is attuned by its own experience and the experience of its neighbors.

The present work explores the use of multi-objective particle swarm optimization technique for

attaining optimal schedule of a task set consisting of autonomous non-dependent tasks (having no precedence relationship) on to an available pool of processing elements of varying capabilities in a distributed computing environment with aim to minimize both make-span and flow time. The performance of the proposed technique is evaluated by executing several data sets and comparing the results with those obtained for genetic algorithm (NSGA-II). The exploratory outcomes demonstrate that the proposed strategy is more productive and is relevant to HC frameworks scheduling.

The rest of the paper is structured in the following fashion. Related work is presented in section II, section III presents problem definition, section IV represents modeling heterogeneity and consistency of computing; Non-dominated Sorting Genetic Algorithm (NSGA) and NSGA-II are discussed in section V, VI represents a general principle of particle swarm optimization and section VII shows multiple objective particle swarm algorithm with crowding distance operator, section VIII presents simulation test results and section IX gives the conclusion.

II. RELATED WORK

Several heuristics strategies have been reported in literature for scheduling non-dependent tasks in distributed computing environment. These include min–max [1], Sufferage [2], min–min, max–min [3], LJFR-SJFR [4], Work-Queue [39] to name a few. The evolvement of meta heuristic has presented new avenues for problem solving and as a result of advancements in meta-heuristic optimization strategies such algorithms are observed to be efficient in taking care of schedulability related issues. The most efficient and well known among them being genetic algorithms [5], simulated annealing [6], ant searching techniques [7] and molecule swarm optimization (Salman et al) [8]. Braun et al [9] portrayed eleven heuristics and analyzed them on various sorts of heterogeneous computing environments representing the execution of GA scheduler in evaluation with others. All the above stated heuristics and meta-heuristics aimed at optimizing a single criteria i. e. minimizing the make-span of the schedule.

Some cases demonstrate the endeavors made in optimizing multiple goals while scheduling non-dependent tasks on heterogeneous situations. Liu et al [10] made an attempt to attain minimum make-span and flow time. Izakian et al [11] investigated five heuristics for limiting make-span and flow-time on heterogeneous conditions with different qualities of both machines and tasks. However, they evaluated both objectives independently. In [12], authors demonstrated the utilization of a few nature inspired meta-heuristics (SA, GA, PSO, and ACO) for scheduling tasks in matrices utilizing uni-objective and multi-objective optimization approaches. Variations of fuzzy molecule swarm algorithm for limiting make-span and flow-time are presented by Liu [10] and Izakian [13]. GA-based schedulers are proposed in [14] [15]. In [16] authors presented a Genetic Algorithm based schedule with an insight to balance load dynamically in distributed systems. Christos Gogos, Christos Valouxis et al. utilize Penalty Based (PB) computation [18] [19] and mathematical programming based approach of Column Pricing [18] to take care of such issue. In this work two new heuristics named as list suffrage algorithm and Tenacious Penalty based algorithm are proposed. These techniques merge multiple goals into a scalar cost function, thus transforming the multi-objective issue to single-objective issue before performing optimization. G. Subashini and M. C. Bhuvaneswari [20] made an endeavor by identifying the task scheduling problem as a true multi-objective optimization problem. They applied Non-dominated Sorting Genetic Algorithm-II, and Non-dominated Sorting Particle Swarm Optimization algorithms to address the problem. However, in this study multi-objective particle swarm optimization algorithm did not employ the crowded distance comparison operator as proposed in [21].

Multiple objective molecule swarm improvement has been proposed by a few researchers [25] [26] [30]. This approach has shown to produce better results when contrasted with other evolutionary strategies for multiple objective optimizations [38]. NSGA-II has been effectively implemented in a few applications [35] [36] [37]. Multi-objective particle swarm optimization has been applied in several standard function optimization problems [21] [25]. The proposed work is an attempt to explore the utility of multiple objective swarm improvement algorithm propelled by [21] [30] to tackle non-dependent task scheduling problem in heterogeneous distributed computing environment.

III. PROBLEM DEFINITION

Computing resources like a solitary PC, a group of PCs, or a supercomputer makes up a HC environment. Let $\tau = \{\tau_1, \tau_2, \ldots, \tau_k\}$ signifies the set of tasks submitted to the resources in a particular time interval. The tasks are assumed to be non-dependent and autonomous (having no inter task dependencies) and in between acquisition of the resources is not permitted. It is also assumed that tasks are not allowed to change the resources they have been assigned to. Let PE = $\{PE_1, PE_2, \ldots, PE_p\}$ denotes the set of processing elements (machines) in a HC environment where the tasks are supposed to be executed. Every

machine utilizes the First Come First Served approach for executing the tasks. Each machine in HC suite knows about the expected execution time of each task. A k × p, Estimated Execution Time (EET) matrix model, where k and p represent the number of tasks and processing elements in distributed computing system, is utilized to represent the time of executing a particular task on a particular processing element. Each row of the EET matrix includes the assessed execution time for a given task on every processing element and each column of the EET matrix comprises of the evaluated execution time of a given processing element for each task. Thus, for an arbitrary task τ_j and an arbitrary processing element PE_i, EET(τ_j, PE_i) is the estimated execution time of τ_j on PE_i. In EET matrix model the standard presumption is made that the processing limit of every task, an estimation or forecast of the computational requirements of each task and the machine accessibility time (prepared time) of every resource is known *a priori*. The principal goal of the scheduler is to limit make-span and flow time.

The specification of the make-span may be effectively expressed by utilizing the use of the EET matrix adaptation. Here, make-span can be represented by means of the completion times of the tasks on processing elements. Let completion [I] = [CT[1] , CT[2] , …, CT[p]] be a vector of completion times of all machines accessible for a given clump of tasks. The completion time of machine I, indicated by completion[i] or CT[i], depicts an aggregate time needed for reloading the machine I after finalizing the previously assigned task(s) and completing the newly assigned task(s) on it. Thus

$$CT[i] = mat_i \text{ or } ready_i + \sum_{j \in \tau(i)} EET[j][i] \quad (1)$$

where, mat_i is the machine availability time of processing element i or $ready_i$ is the ready time of processing element i.
τ(i) is the group of tasks assigned to processing element i.

The make-span is the maximal finish (completion) time and can be expressed as

$$\text{Make-span} = \max_{i \in PE_i} CT[i] \quad (2)$$

and the flow-time is defined as the sum of the completion times of all the tasks in the batch of tasks.

$$\text{Flow-time} = \min_{S \in schedules} \sum_{j \in k} F_j \quad (3)$$

Flow-time is generally regarded as a service optimizing criteria as it articulates the response time to the submitted task execution requests submitted by the HC users. In terms of EET matrix model the flow-time can be measured as a workflow of a sequence of tasks submitted to a given machine i. It is given as

$$\text{Flow-time [i]} = ready_i + \sum_{j \in sorted\,(i)} EET[j][i] \quad (4)$$

where, $sorted\,[i]$ is the group of tasks assigned to the machine i sorted in ascending order by the corresponding EET values.

IV. MODELING HETEROGENEITY AND CONSISTENCY IN DCE

The distributed computing model can depict distinctive degrees of heterogeneity in disbursed figuring condition through consistency of computing. Consistency of computing alludes to the intelligence among execution times got by a machine with those acquired by whatever remains of machines for an arrangement of tasks. In this way three sorts of consistency of computing in HC condition can be characterized utilizing properties of EET matrices: consistent, inconsistent, semi-consistent [9]. Matrices are said to be consistent if at whatever point a machine m_j executes any assignment t_j quicker than machine m_k, then machine m_i of type m_j executes all tasks speedier than machine m_k. consistent matrices were produced by sorting each row of the EET matrix independently, with machine m_0 continually being the speediest and machine $m_{(m-1)}$ the slowest.

Interestingly inconsistent or unreliable matrices portray the circumstance where machine m_i might be speedier than the machine m_k for a few jobs and slower for others. These matrices are left in an unordered, arbitrary state in which they were created (i.e. no consistency is upheld). Semi-consistent, mostly consistent matrices, are inconsistent

matrices that incorporate a consistent sub-matrix. For semi-consistent matrices, the row components in column position{0, 2, 4, …..} of row i are extracted, sorted and supplanted all together, while the row components in column
positions {1, 3, 5, …..} remain unordered (i.e. the even columns are consistent and odd columns are all in all inconsistent).

V. NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA)

Non-dominated Sorting Genetic Procedure [30] motivated GA in light of the idea of Non-dominated sorting of populace into Pareto ideal fronts. It is utilized as a part of multi-goal improvement issues and is an example of developmental algorithms. NSGAs basic goal is to enhance the versatile fitness of

a populace of competitor answers for the problem by sorting it into Pareto ideal fronts. The calculation is roused by a transformative procedure and utilizations developmental genetic administrators of crowded tournament selection, crossover, and mutation. In this calculation the populace is sorted in view of the request of Pareto predominance. Every subgroup part is subjected to a likeness testing inside a Pareto front. The resultant gatherings and further comparability measures are utilized to advance assorted diversity of arrangements among various fronts. Established NSGA and the upgraded and adjusted NSGA-II [17] are two sorts of NSGA. Established NSGA has been for the most part scrutinized for its high computational complexity, absence of elitism and utilization of a predefined ideal parameter for sharing fitness σ share. The following paragraphs give the description of NSGA-II **[**17**]** [31] **[**33**]**.

A. NSGA – II

An altered and redesigned adaptation of NSGA is called NSGA-II. It utilizes a superior and quick non-domination sorting, consolidates idea of elitism, and the populace fitness, require not to be shared utilizing a sharing fitness parameter. The elitism instrument of the algorithm advances that best non-dominated arrangements of the parent and child populace are proliferated to the people to come. Amid elitism great arrangements discovered early are never lost unless a superior arrangement is found to supplant them. The close Pareto ideal arrangement of the last front gives diverse answers for the scheduling issue.

1) Quick Non-dominated Sorting

For the most part non-domination sorting technique is the primary segment of a multi-objective evolutionary algorithm. It yields high computational complexity. So the utilization of a quick and computationally proficient non-domination sorting method is exceptionally significant to the accomplishment of Multi Objective Evolutionary Algorithm (MOEA). NSGA-II utilizes a quick and computationally viable non-dominated sorting strategy. In non-dominated sorting approach, utilized as a part of NSGA-II, the populace is sorted in light of non-dominance. The populace is initialized and sorted in view of non-domination to be classified in various Pareto ideal fronts. The main front being totally dominated in the present populace, the people of the second front are just dominated by the people of the principal front and the people in the third front are being overwhelmed by those of the first and second front and the sorting into fronts goes on. The populace in each front is positioned utilizing fitness values. Individuals from the principal front are appointed rank 1. What's more, the individuals from the second and consequent fronts are doled out the rank 2, 3 and the positioning proceeds.

A second parameter known as crowded displacement measurement is assessed for every person of the front. Crowded displacement estimation indicates how closely related an individual is to its neighbors. Crowded displacement estimation are utilized to keep up the better differences (diversity) in the populace.

The non-domination sorting methodology utilized as a part of NSGA-II is quick when contrasted with different MOEAs. NSGA-II has been tuned in a manner that it is computationally effective and the non-dominated sorting method is fast.

For a populace of size T and the quantity of target objectives D, the quick non-dominated sorting methodology is characterized as described here. For every part t of the populace, two qualities are figured.

a. Domination count n_t, i.e. the number of members (solutions) which dominate the individual t, and

b. The set S_t which provides solutions which the distinct t dominates.

All member solutions in the first front will get $n_t = 0$, then for every member q in S_t, we reduce the domination count by one. Continuing in this way if for every member solution the domination count becomes zero, then we place it into a separate list and the second front is discovered. The process is continued until all fronts are discovered.

The final complexity of the fast non-domination procedure is $O(DT^2)$, whereas the complexity of normal non-dominated sorting used in classical NSGA is $O(DT^3)$.

2) Fitness Assignment and Positioning Strategy

Every part arrangement of the populace is allocated a rank in view of fitness incentive in non-domination sorting methodology. For individuals from a similar front crowded displacement measurement is also assessed.

3) Diversity Preserving Mechanism

The NSGA-II converges the arrangement into Pareto ideal front. Other than convergence, differences of populace in the front should be kept up. Differing qualities in the front demonstrates a decent spread of arrangements along the Pareto ideal front. The

traditional NSGA utilized a sharing parameter (niched parameter) which keeps up the craved differences of part arrangements, yet the utilization of sharing parameter makes the calculation awkward and furthermore builds the reliance of the calculation on the estimation of the sharing (niching) parameter picked. In NSGA-II the utilization of crowded displacement separation calculation dispenses with the above issues to some degree.

4) Density Estimation - Crowding Displacement Assignment

Figure the normal separation of two focuses on either side of the point along each of the objective in order to get a gauge of the thickness of arrangements encompassing a specific arrangement in the populace. Crowded displacement is allotted front savvy and contrasting the crowded measurement between two people in various fronts is good for nothing. Crowded displacement measurement helps in acquiring uniform dissemination.

The fundamental thought behind the crowded displacement calculation is finding the Euclidean separation between individual in a front in light of their objective function values in the m dimensional hyperspace. The people in the limit are constantly chosen since they have unbounded crowded displacement value.

5) Crowded Displacement Based Sorting

Crowded displacement based comparison is utilized to control the procedure of choice at the different phases of the algorithm towards a consistently spread-out Pareto optimum front. Expect that each individual i in the populace has two properties:

- Non-domination rank (i rank)
- Crowded Distance (i distance)

Between two people i and j, the person with lower rank will be selected (i.e. irank < jrank) or if both individual has a place with a similar front then their crowded distance is looked at, and individual with more prominent crowded distance i.e. an individual situated in a lesser crowded district is chosen.

6) Elitist Method

The most influential part of NSGA-II is its elitist method where the best non-dominated solutions of the parent and child population are moved through to the next generation.

VI. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a stochastic populace based inquiry strategy enlivened by the social conduct of creatures [27] [28] [41] , for example, winged creatures and fish. This algorithm was initially presented by James Kennedy and Russell Eberhart [27] in 1995. In PSO, every particle, called a molecule, flies through the multidimensional search space and alters its position as per its own involvement and the experience of its neighbors. A molecule can fly either quick and a long way from the best positions to investigate obscure zones (global inquiry), or gradually and near a specific position (fine tune) to discover
better outcomes. PSO is very easy to execute and has few control parameters. Equations 5 and 6 are the two essential redesign guidelines of standard PSO

$$v_k \leftarrow w v_k + c_1 r_1 (P_k - x_k) + c_2 r_2 (P_g - x_k) \quad (5)$$

$$x_k \leftarrow x_k + v_k, \quad (6)$$

where, v_k and x_k are velocity and position vectors of particle k, respectively, P_k is the best local position found by particle k, and P_g is the best global position found in the whole population. The two parameters c_1 and c_2 are positive constants, called learning factors; c_1 presents how much a particle is attracted to its best position, and c_2 is the same for the global position. Values of these two parameters vary depending on the nature of the problem but they are usually considered to be equal to 2.0. w is the inertia weight and controls the amount of freedom of the particles to explore. It has been shown, e.g. in [32], that PSO performs better when w decays from 0.9 to 0.4 over time. r_1 and r_2 are uniform random variables providing the stochastic aspect of the algorithm [41].

Algorithm 1 : Pseudo Code for Particle Swarm Optimization (Continuous Numbers)

Initialize population (position and velocity of particles)

repeat

calculate all particles

for all particle k do

if current position of particle k, x_k, produces the best fitness in

its history then

$P_k \leftarrow x_k$

If fitness of x_k is the best fitness in global then

$P_g \leftarrow x_k$

end if

end if

end for

update velocity and position of particles according to the

Equations 5 and 6

until termination criteria are met

In this manner in a PSO strategy, all particles are started haphazardly and assessed to register the fitness together with finding the individual best (estimation of every molecule) and the worldwide best (estimation of molecule in the whole swarm) after that a loop begins to locate an ideal arrangement. In the loop, first the particles' speed is overhauled by the individual and worldwide bests and every molecule's position is upgraded by the present speed. Essentially two models of PSO calculations, to be specific the Global Best (gbest) and Local (Nearby) Best (lbest) PSO, have been created which vary in the span of their neighborhoods [40]. The worldwide best PSO (or gbest PSO) is a strategy where the position of every molecule is impacted by the best fit molecule in the whole swarm. It utilizes a star social topology where the social data acquired from all particles in the whole swarm is utilized.

The nearby best PSO (or lbest PSO) approach allows each molecule to be affected by the best fit molecule surfed in its neighborhood, and it reflects a ring social topology [40]. Here this social data traded with in the neighborhood of the molecule, indicating adjacent learning of the environment. These two models of PSO calculation are generally relevant in the vast majority of the streamlining issues. The gbest PSO specifically have been connected in clustering issue, job-shop scheduling issue, and single machine aggregate weighted lateness issue.

PSO can take care of assortment of optimization problems, particularly in the field of multi-dimensional persistent space improvement problem. Communicating the particle's position is difficult, thus it is limited in the utilization of Combinatorial Optimization Problems [COP]. Numerous current investigates are for the most part in the utilization of PSO to COP. The primary COPs tended to by PSO are: scheduling issues, for example, job shop planning and flow shop scheduling and routing problems, for example, Traveling Salesman issue and Vehicle Routing issue

Scheduling: PSO in hybridization with Hill Climbing (HC) calculation as a local search technique has been connected for comprehending assignment of task in dispersed frameworks [42]. PSO is likewise connected for taking care of task assignment issue by Salman et al (2002) [8]. A few creators have likewise executed the flow shop scheduling with restricted supports. They proposed a hybrid PSO for illuminating the flow shop planning, where permutation encoding is utilized for representing the particles.

In the present work this method is used due to its suitability for solving scheduling problem as it has fewer control parameters, better convergence rate, and significant improved computational efficiency. These factors have motivated the researchers to apply multiple objective particle swarm optimization in independent task scheduling problem [20]. Above factors have also motivated us to use the global best model of multiple objective particle swarm optimization with crowded displacement operator to solve independent task scheduling in heterogeneous computing environment.

A. *Particle's Encoding*

Following scheme of particle's encoding has been proposed in [20]. A particle tells about a conceivable arrangement in the populace and dimensionality n represents n tasks The Smallest Position Value (SPV) control is utilized first to discover a change in comparing to the persistent (continuous) position X_i . For the n tasks and m processors scheduling issue, every particle represents a probable solution. The position vector $X_k^i = [X_1^i , X_2^i , ..., X_n^i]$ has a continuous set of values. In light of the SPV run, the persistent position vector can be changed to an arrangement of discrete permutations $S_k^i = [S_1^i, S_2^i , ..., S_n^i]$. Thus, the operation vector $r_k^i = [r_1^i, r_2^i, ..., r_n^i]$ is shown by the following rule: $r_k^i = S_k^i \bmod m$, where m is used to represents the number of processors. This sequence represents the computing processor number for n tasks.

VII. MULTIPLE-OBJECTIVE PARTICLE SWARM OPTIMIZATION WITH CROWDING DISTANCE OPERATOR (MOPSO-CD)

In the present work, an approach is proposed that develops the Particle Swarm Optimization (PSO) calculation to handle with multi-goal optimization

issues by consolidating the system of crowded displacement separation procedure into the calculation of PSO, particularly on global best selection and in the erasure technique for an outside archive (chronicle) of non-dominated arrangements. The crowded displacement separation instrument together with a mutation administrator keeps up the assorted qualities of non-dominated arrangements in the outer archive. The algorithm suggested in [21] [29] is rephrased for Multiple Objective Particle Swarm Optimization with Crowded Displacement Operator and presented below:

Algorithm 2: MOPSO-CD Algorithm

1. A loop is started for 1 up to the size of the population Q.
   a. Initially POP[i] is generated arbitrarily (POP is the population of particles).
   b. Velocity is set to zero, Vel[i] =0.
   c. Based on above initialization POP[i] is calculated.
   d. Again personal best position of every particle is initialized and set to POP[i], PERbests[i] = POP[i].
   e. Update the global best position (GLbest) with the best particle's position found in POP[i].
2. End of loop.
3. An iterative count t=0 is initialized.
4. Non-domination oriented solutions that are found in population POP are stored in outer archive A. (A is the external archive set that stores non-dominated solutions found in population POP).
5. A repetitive loop is started to do the following :
   a. Calculate the crowded displacement values of every non-dominated member solution in archive set A using algorithm 2.1.
   b. The non-dominated solutions of set A are stored in decreasing crowded displacement values.
   c. A loop is started for 1 up to the size of population, Q.
      i. Arbitrarily choose the global best instructor for POP[i] from predefined top portion (e.g. top 10%) of sorted archive A, thereafter store its position to GLbest.
      ii. Velocity is updated according to the following.
      Vel[i] = ω x Vel[i] + r_1 x (PERbests[i] - POP[i]) + r_2 x (A[GLbest] - POP[i])
      (ω is the inertia weight = 0.4)
      (r_1 and r_2 are random numbers in the range [0,1])
      (PERbests[i] is the personal best position that the particle *i* has reached)
      (A[GLbest] is the global best guide for each non-dominated solution)
      iii. Calculate the new position of POP[i] : POP[i] = POP[i] + Vel[i]
      iv. On the off chance that POP[i] goes past the limits, then it is reintegrated by having the choice variable take the estimation of its relating lower or upper limit and its speed is decreased by - 1 with the goal that it looks the other way.
      v. On the off chance that (t < (MAXT * PMUT), then perform change (mutation) on POP[i] . (MAXT is the maximum number of iterations)

      (PMUT is the probability of mutation)
      vi. Calculate POP[i]
   d. End of loop.
   e. Embed all new non-dominated arrangement in P into A on the off chance that they are not commanded (dominated) by any of the put away arrangements. Every single dominated arrangement in the archive set A by the new arrangements is expelled from the chronicle (archive). On the off chance that the archive is full, the answer for be supplanted is controlled by the accompanying strides:
      i. Compute the crowded displacement estimations of each non-dominated arrangement in the chronicle (external archive) A,
      ii. Sort the non - dominated arrangements in A in descending crowded displacement separation values,
      iii. Randomly select a molecule from a predefined base segment (e.g. bring down 10%) which include the most crowded particles in the archive then supplant it with the new arrangement.
   f. Update the individual best arrangement of every molecule in POP. On the off chance that the current PERbests dominates the position of molecule in memory, the particles (molecules) position is upgraded utilizing PERbests[i] = POP[i]
   g. Increment iteration counter t

6. Until most extreme number of iterations is come to.

A. Crowding Displacement Based Operator:

Crowded displacement is ascertained by first sorting the set of solutions in climbing objective function values. The crowded displacement separation estimation of a specific (member solution) arrangement is the average displacement separation of its two neighboring arrangements. The limiting arrangements which have the most reduced and most noteworthy objective function values are given an

interminable crowded displacement values with the goal that they are constantly chosen. This procedure is accomplished for every objective value. The last crowded displacement separation estimation of an answer is registered by adding the whole individual crowded displacement values in every objective function.

The crowded displacement operator is used mainly to maintain the diversity of solutions in each of the resolved Pareto optimal fronts. Its main advantage is that it can locate the optimum solution with in each front very quickly.

Algorithm 2.1: Crowded Displacement (distance) Operator

1. Get the number of non-monopolized results in the outer repository (archive)
   a. n = | Sol |
2. Initialize displacement
   a. FOR k=0 TO MAX

Parameters	Values
Population size	100
Maximum iteration	40
Inertia Weight (W)	0.4
Acceleration Coefficient (C1)	2
Acceleration Coefficient (C2)	2
Maximum Velocity (Vmax)	[10, 90]

   b. Sol[k].distance = 0
3. Calculate the crowded displacement value of every solution
   a. For each objective m
   b. Sort using each objective value
      Sol = sort(Sol, m)
   c. For k=1 to (n-1)
   d. Sol[k].distance = Sol[k].distance + (Sol[k+1] .m –Sol[k-1].m)
   e. The maximum distance to the boundary points so that they are always selected
      Sol [0].distance = Sol[n].distance = maximum distance

VIII. SIMULATION TEST RESULTS

Simulation test runs were carried out for performance evaluation of the proposed method. In EET matrix framework, the measure of difference amongst the execution times of tasks for a given machine is characterized as task heterogeneity. Machine heterogeneity reflects the variety that is conceivable among the execution times for a given task over every one of the machines. Simulation study is based on the simulation test runs carried out several times utilizing the benchmark problem instance of the undertaken problem as given in [9]. Instance consisted of 512 tasks and 16 machines and is labeled as uf-xo-yy-zz as follows:

- uf depicts uniform distribution used in generating the matrices.
- xo depicts the type of inconsistency; co represents consistent, in represents inconsistent, and p represents partially-consistent or semi-consistent.
- yy signifies the heterogeneity of the tasks; hi represents high and lo represents low.
- zz signifies the heterogeneity of the machines; where hi represents high and lo represents low.

Simulation run was also carried out several times for problem instance of 64 tasks to be assigned on 8 machines. In our test based on benchmark problem instance, the underlying populace for the looked at techniques is produced utilizing two scenarios – (a) arbitrarily created particles from a uniform appropriation, and (b) one molecule utilizing the min-min heuristic (that can accomplish a decent lessening in make-span) and the others are arbitrary arrangements. NSGA-II and MOPSO-CD parameters were tuned to obtain scenario for fair study. List of parameters, used in simulation study, along with their values/ range are given in Table 1.

Table 1: Parameters for Simulation Study

Standard Deviation of the proposed method along with NSGA-II is shown in Figure 1 which is obtained by running 10 independent runs of each algorithm on benchmark problem instance. The simulation for this is carried out in MATLAB R2012 environment using benchmark problem instance of EET matrix of size 512 x 16. Table 2 and Table 3 show the make-span and flow-time values of EET matrices of size 512 x 16, respectively. The study in which NSGA-II suggested in [20] is compared with the proposed method shows that the proposed method MOPSO-CD is viable and effective in all 12 instances of the problem.

Table 2: Make-span Values of EET Matrix

Instance	NSGA-II	MOPSO-CD
uf-co-hi-hi	7.8921E+06	7.8678E+06
uf-co-hi-lo	1.6163E+05	1.5143E+05
uf-co-lo-hi	2.7648E+05	2.6463E+05
uf-co-lo-lo	5.297 E+03	5.194E+03
uf-in-hi-hi	3.4962E+06	3.4605E+06
uf-in-hi-lo	8.1715E+05	8.1615E+05
uf-in-lo-hi	1.1270E+05	1.1217E+05
uf-in-lo-lo	2.636E+03	2.423E+03
uf-pa-hi-hi	4.5713E+06	4.4806E+06
uf-pa-hi-lo	1.0485E+05	1.0478E+05
uf-pa-lo-hi	1.5397E+05	1.5393E+05
uf-pa-lo-lo	3.449E+03	3.423E+03

Table 3: Flow-time Values of EET Matrix

Instance	NSGA-II	MOPSO-CD
uf-co-hi-hi	105123172	104322164
uf-co-hi-lo	2372403	2278675
uf-co-lo-hi	3612434	3576535
uf-co-lo-lo	78945	78168
uf-in-hi-hi	43655443	43415667
uf-in-hi-lo	11131834	11024564
uf-in-lo-hi	1478862	1446619
uf-in-lo-lo	37608	36981
uf-pa-hi-hi	62435712	62234510
uf-pa-hi-lo	1453702	1383892
uf-pa-lo-hi	1985104	1945102
uf-pa-lo-lo	49789	49562

IX. CONCLUSION

Statically timetabling of autonomous jobs in heterogeneous processing condition discovers value in numerous applications. In the proposed paper, the scheduling problem of independent tasks in heterogeneous computing environment is investigated using Multiple-Objective Particle Swarm Optimization with Crowded Displacement operator (MOPSO-CD) in order to limit both make-span and flow-time. The performance of the proposed method is contrasted with Multiple-Objective Non-dominated Sorting Genetic Algorithm-II. The trial comes about uncover the nature of schedules nearly for all benchmark issue examples. Subsequently, MOPSO-CD can be utilized to discover better schedules fulfilling various goals and it appears to be encouraging way to deal with planning autonomous assignments in HC condition. Additionally, methodologies might be connected for considering different types of HC scheduling, for example, scheduling jobs with priority limitations or inside element condition. Commonly used search algorithms for example 'Hill climbing' and 'Simulated Annealing' continuously move towards the solutions that have an improved fitness function value and they search the problem space in an arbitrarily manner. Likewise, MOPSO move towards stochastic search in the problems space to find improved solutions thus generating an ideal Pareto optimal front.

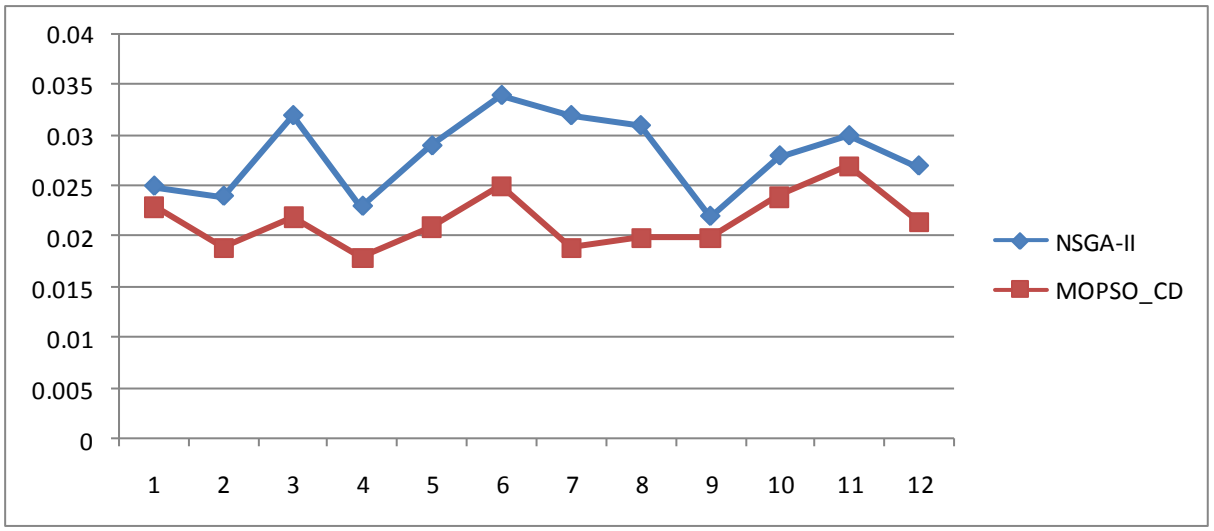

Fig. 1: Standard Deviation (Y-axis) of Two Methods for 12 Instances (X-axis)

REFERENCES

[1] Munir E U, Li J-Z, Shi S-F and Rasool Q Performance Analysis of Task Scheduling Heuristics in Grid, In: ICMLC'07: Proceedings of the International Conference on Machine Learning and Cybernetics,6: 3093–3098, 2007.

[2] Maheswaran M, Ali S, Siegel H J, Hensgen D and Freund R F, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, J. Parallel and Distributed Computing 59: 107–131, 1999.

[3] Freund R F, Gherrity M, Ambrosius S, Campbell M, Halderman M, Hensgen D, Keith D E, Kidd T, Kussow M, Lima J D, Mirabile F, Moore L, Rust B and Siegel H J, Scheduling resources in multiuser, heterogeneous, computing environments with SmartNet, In: 7th IEEE Heterogeneous Computing Workshop, 184–199, 1998.

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

[5] Page A and Naughton J, Framework for task scheduling in heterogeneous distributed computing using genetic algorithms, Artificial Intelligence Rev. 24: 415–429, 2005.

[6] Yarkhan A and Dongarra J, Experiments with scheduling using simulated annealing in a grid environment, In: Proceedings of the 3rd International Workshop on Grid Computing (GRID2002), 232–242, 2002.

[7] Ritchie G and Levine L , A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments, In: 23rd Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG, 2004.

[8] Salman A, Ahmad I and Al-Madani S, Particle swarm optimization for task assignment problem, Microprocessors and Microsystems 26(8): 363–371, 2002.

[9] Braun T D, Siegel H J, Beck N, Boloni L L, Maheswaran M, Reuther A I, Robertson J P, Theys M D and Yao B, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, J. Parallel and Distributed Computing 61: 810–837, 2001.

[10] Liu H, Abraham A and Hassanien A, Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, Future Generation Computer Systems 26 (8) : 1336–1343, 2010.

[11] Izakian H, Abraham A and Snasel V, Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments, IEEE Control Systems Magazine 1: 8–12, 2009.

[12] Abraham A, Liu H, Grosan C and Xhafa F, Nature inspired meta-heuristics for grid scheduling: single and multi-objective optimization approaches. Studies in computational intelligence, Berlin Heidelberg: Springer Verlag, 247–272, 2008.

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

[14] Xhafa F, Carretero J and Abraham A, Genetic algorithm based schedulers for grid computing systems, Int. J. Innovative Computing, Information and Control 3: 1053–1071, 2007.

[15] V. Di Martino and M. Mililotti. Sub optimal scheduling in a grid using genetic algorithms. Parallel Computing, 30:553–565, 2004.

[16] A.Y. Zomaya and Y.H. Teh. Observations on using genetic algorithms for dynamic load-balancing. IEEE Transactions On Parallel and Distributed Systems, 12(9):899–911, 2001.

[17] Deb K, Pratap A, Agarwal S. and Meyarivan T, A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II, Kanpur Genetic Algorithms Laboratory Report No-200001 , Indian Institute of Technology, Kanpur, 2000.

[18] Christos Gogos, Christos Valouxis, Panayiotis Alefragis, George Goulas, Nikolaos Voros, Efthymios Housos, Scheduling independent tasks on heterogeneous processors using heuristics and Column Pricing, Future Generation Computer Systems 60, 48–66, 2016.

[19] A.K. Chaturvedi, R. Sahu, New heuristic for scheduling of independent tasks in computational grid, Int. J. Grid Distributed Computing. 4 (3), 25–36, 2011.

[20] G Subashini and M C Bhuvaneswari, Comparison of multi-objective evolutionary approaches for task scheduling in

distributed computing systems, Sadhana, Vol. 37, Part 6, pp. 675–694. Indian Academy of Sciences, December 2012.

[21] Carlo R. Raquel, Prospero C. Naval, Jr., An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization, ACM, GECCO' 05, June 25-29, 2005.

[22] Angeline, P.J. Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In Proceedings of the Seventh Annual Conference on Evolutionary Programming, San Diego, pp. 601–610, 1998.

[23] Salerno, J. Using the particle swarm optimization technique to train a recurrent neural model. In Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, Newport Beach, CA, USA, pp. 45–49, November 3-8, 1997.

[24] Eberhart, R.C.; Shi, Y. Evolving artificial neural networks. In Proceedings of the International Conference on Neural Networks and Brain, Beijing, P.R. China, pp. 5–13, October 27-30, 1998.

[25] Coello Coello, C. A. and Salazar Lechuga, M., MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In Proceedings of the Congress on Evolutionay Computation (CEC'02), volume 1, pages 1051–1056, Honolulu, HI. IEEE Press, 2002.

[26] Coello Coello, C. A., Toscano Pulido, G., and Salazar Lechuga, M. Handling Multiple Objectives With Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation, 8(3):256–279, 2004.

[27] Kennedy, J. and Eberhart, R. C. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, pages 1942–1948, Piscataway, New Jersey. IEEE Service Center, 1995.

[28] Kennedy, J. and Eberhart, R. C. Swarm Intelligence. Morgan Kaufmann, San Francisco, California, 2001.

[29] C. Tsou, S. Chang, and P. Lai., Using Crowding Distance to Improve Multi-Objective PSO with Local Search, Swarm Intelligence , Focus on Ant and Particle Swarm Optimization, Edited by Felix T.S. Chan and Manoj Kumar Tiwari, 2007.

[30] K. Deb and N. Srinivas, Multi-objective optimization using non-dominated sorting in genetic algorithms. Evolutionary Computation, pages 221–248, 1994.

[31] Shreeram Kushwaha, Multiobjective optimization of cluster measures in Microarray Cancer data using Genetic Algorithm Based Fuzzy Clustering, Bachelor of Technology in Computer Science & Engineering Thesis, National Institute of Technology Rourkela, 2012-13.

[32] Y. Shi and R. Eberhart, Empirical study of particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999), pages 1945–1950, 1999.

[33] Seshadri, A., A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II (1st ed.). New Jersey: New Jersey Institute of Technology. Retrieved from https://web.njit.edu/~horacio/Math451H/download/Seshadri_NSGA-II.pdf, 2006.

[34] M. T. Jensen., Reducing the run-time complexity of multiobjective eas: The nsga-ii and other algorithms. Trans. Evol. Comp, 7(5) : 503 -515. ISSN 1089 - 778X. doi: 10.1109/TEVC. 2003. 817234. URL http://dx/doi.org/10.1109/TEVC.2003.817234, October 2003.

[35] Hossein Ghiasi, Damiano Pasini and Larry Lessard, A non-dominated sorting hybrid algorithm for multi-objective optimization of engineering problems, Vol. 43, No. 1, 39–59, Engineering Optimization, January 2011.

[36] Zengqiang Jiang, Le Zuo, Mingcheng E, Study on Multi-objective Flexible Job-shop Scheduling Problem considering Energy Consumption, 7(3), 589-604, Journal of Industrial Engineering and Management, 2014.

[37] Kalyanmoy Deb, Udaya Bhaskara Rao N., and S. Karthik, Dynamic Multi-Objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-Thermal Power Scheduling, Kanpur Genetic Algorithms Laboratory (KanGAL) Report Number 2006008, Indian Institute of Technology Kanpur, 2005.

[38] Adriana Cortes Godınez, Luis Ernesto Mancilla Espinosa, EfrenMezura Montes, An Experimental Comparison of Multi-Objective Algorithms: NSGA-II and OMOPSO, Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010, Oct. 2010,.

[39] Hagerup T., Allocating Independent Tasks to Parallel Processors: An Experimental Study. Journal of Parallel and Distributed Computing, 47, pp. 185-197, 1997.

[40] Satyobroto Talukder, Mathematical Modeling and Applications of Particle Swarm Optimization, Master of Science Thesis, School of Engineering, Blekinge Institute of Technology, Sweden, February 2011.

[41] Mahmood Rahmani, Particle swarm optimization of artificial neural networks for autonomous robots, Master of Science in Complex Adaptive Systems Thesis, Department of Applied Physics, Chalmers University of Technology, Sweden, 2008.

[42] Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, Yi-Te Wang, A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems, Computer Standards & Interfaces , Vol.28, pp. 441-450, 2006.