A Hybrid Approach for Task Scheduling Problems in Parallel Systems using Task Duplication

Rachhpal Singh
PG Department of Computer Science and Applications
Khalsa College
Amritsar (Punjab) - India
rachhpal_kca@yahoo.co.in

ABSTRACT

Parallel computing is a group of computers (or) processors (or) machines carried out concurrently working on the concept that huge jobs can be divided into smaller ones often that are then executed parallel. In heterogeneous environment, machines of numerous dimensions involve in completing the job work. Task or job scheduling is the process of allocating the dependent tasks to parallel processors so that the optimized outcome will obtained. The problem is occurred while the execution of some of the jobs or tasks that depends on the completion of another set of jobs or tasks running at the same time. To reduce the complexity of such problems, the effective methodology be developed to assure the correctness and success of optimization. In this paper, task duplication based genetic algorithm with load (task) balance (DBGA-LB) is drawn and implemented. According to this method, the random task duplication is performed in certain tasks to utilize the idle time slot of the processors and thus simultaneously reducing the schedule length (maximum execution time of tasks). Our developed algorithm has the combination of schedule length and load or task balance as the fitness function. The load balance ensures the number of tasks allocated to each processor is more or less equivalent to other processors. The proposed method (DBGA-LB) gives better results and always outperforms the existing traditional algorithms or techniques.

Keywords

Parallel computing; Heterogeneous system; Task scheduling; Task duplication; Schedule length; Load balance.

1. INTRODUCTION

The parallel computing is a hopeful approach to assemble the increased need for high speed machines or resources [1]. A parallel heterogeneous system comprises of a set of nodes or processors of changeable computing power, associated by a high-speed network [2]. As the heterogeneous systems become bigger, there is a chance of processor and network link failures and this can have an unfavorable impact on applications executing on the system. In order to reduce the impact of failures on the particular application, task scheduling algorithms should be designed to optimize various NP- hard problems having various parameters like makespan, reliability, speedup, efficiency and utilization [3]. However, scheduling the jobs or tasks is one of the most challenging problems concerning parallel computing systems.

In the area of job scheduling, a program is denoted as a directed acyclic graph, called a task graph, where the nodes illustrate the tasks and the edges denotes the connections between the tasks. Scheduling under such a task graph on a set of processors for quick execution is called as NP-hard optimization problem [4]. The scheduling of task or job graph for a multiprocessor parallel system is a well-defined NP-hard problem that has established much attention and it considered as one of the major complex problems in parallel systems [5].

The goal of a scheduler is to allocate tasks to available corresponding processors by considering the preference conditions between tasks or jobs while reducing the overall execution time called as makespan. This problem becomes a tough NP-hard problem in the case of an arbitrary number of machines and task execution time. The improper scheduling will decrease or even terminate the true potential utilization of the parallel systems [1]. One critical feature in this multi processor program is the scheduling of the sub-tasks on the machines or processors of the parallel computing systems [4]. Job's scheduling or task scheduling can be stated as scheduling or allocating the tasks into an available set of machines or processors and calculating the progression of task execution at individual processor. While the task completion time simply called as makespan of the respective tasks is obtained by the performance evaluation of machines and the order of the tasks. Hence, the task scheduling comprises of three major components namely multi processor performance in parallel; task mapping to the respective processors and execution sequence of the tasks occurs in each processor. These components are independent and it can be focused carefully to achieve the optimized scheduling with respect to the quality parameters of parallel systems [6].

The NP-complete problem in task assignment should handle efficiently by developing good heuristics approach that will allocate the tasks to the corresponding processors and thus minimize the schedule length [2]. Generally, the task scheduling is mainly classified into two types: dynamic and static scheduling. In static scheduling, the characteristics of heterogeneous parallel systems such as communication, task processing periods, synchronization requirement and data dependencies are recognized before execution of tasks. In the case of dynamic scheduling under parallel systems, some

assumptions were made before execution of tasks and it decide at the time of running the process. The difficulty involves mapping a directed acyclic graph for a group of processing jobs and their data priority onto a parallel system [7].

The mapping process involves matching and scheduling of tasks and interactions is a major problem since a suitable mapping can really utilize the parallelism of the system thus achieving great speedup and high-efficiency. A well-organized scheduling of the tasks on the available processors or machines is one of the key factors for reaching good performance [2]. According to many scheduling algorithms processed in the real-time parallel computing environments, the scheduler was not successfully optimized the problems occurred during execution. This is because there are always few wrong assumptions were taken about availability of homogeneous resources [8]. Therefore, if the scheduler performs proficiently it would satisfy the precedence requirements of the task in respective processors while simultaneously minimize the schedule length [7].

In multi-processor based system; the processing capability of each processor may be different. The parallel tasks must be scheduled into the processors thus the total execution time of the task must be as less as probable [9]. If the application designed with a help of a dependency graph, the problem occurs while scheduling task of the application into the available heterogeneous systems to optimize the parameters in terms of quality [3].

In common, the task scheduling algorithms may be divided into two main classes' namely iterative and non-iterative algorithms. The main principle of the iterative algorithms is that they initiate from a starting solution and attempt to improve it. On contrary, the non-iterative algorithm only tries to reduce the starting time of the tasks in a parallel program. This is completed by scheduling the jobs into the available machines without back tracking. The task assigning algorithms may be separated into two types namely duplication based scheduling and non-duplication based scheduling [7].

The scheduling problem is solved by the meta-heuristic population-based algorithm called Genetic Algorithm (GA) [1]. To deploy this scheduling process various types of genetic algorithms are used and are scheduled by the chromosomal representation. The meta-heuristic functions are highly considered and brings the high-quality solution for the task scheduling problem while optimization. There are many pros in the genetic search, the most important and useful one is to reduce the computational time by inherent parallelism [7] and the GA is used to find the most favorable solution for this problem in less computational time of the tasks [9]. Some of the important application of genetic algorithm lies in disease diagnosis, query optimization, software engineering and task scheduling [20-25].

The processor's essential communications with the other processors are reduced by the eminent technique called task duplication. This task duplication technique, a task in a processor is replicated and executed on many processors in the network. Available different processor in the local network will process the task, and the processor will establish less communication between the processors [4]. In various duplications-based scheduling algorithms the only strategy is a selection of the task to duplicate [2]. Task duplication reduces the finish time of the total task and energy cost to reach multiple targets with genetic algorithm and ACO to gain the optimal solution quickly [10]. Fig. 1 represents the task scheduling in heterogeneous parallel systems as follows.

In the proposed approach, the Duplication Based Genetic Algorithm (DBGA) can utilize the Load Balance (LB) strategies. The purpose of adding load balance modification is that to produce the lesser schedule length and simultaneously load balance is satisfied among the processors. The load balance between the parallel processors is calculated by the ratio of the maximum execution time or schedule length to the average execution time over all processors.

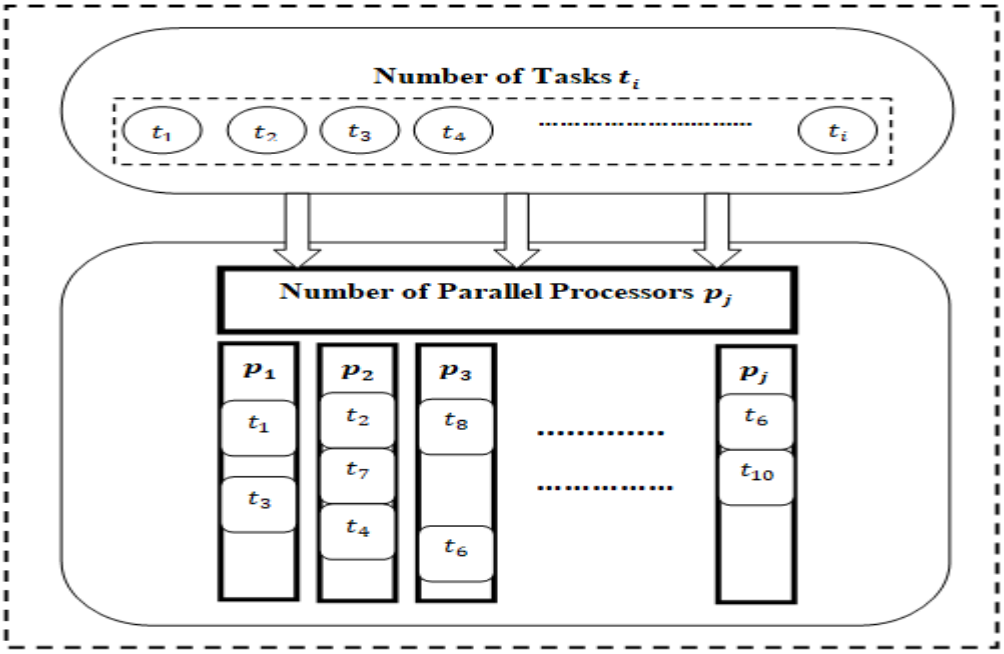

Fig. 1: Task Scheduling or Allocation in Heterogeneous Multiprocessor Parallel Systems

Finally, the proposed DBGA-LB approach can be compared with standard genetic algorithm and First Come First Serve (FCFS) scheduling algorithm in terms of schedule length, load balance, speed, processor utilization and efficiency.

The paper is structured as follows: The related works are specified in Section 2. The proposed Duplication Based Genetic Algorithm with Load Balance (DBGA-LB) approach and its procedures are derived in Section 3. The performance evaluation and results are categorized in Section 4. Finally, Section 5 concludes the paper.

2 RELATED WORKS

A number of existing methods are being addressed by various researchers in the past; in this section, some of the methods associated to task scheduling in parallel systems are provided.

Amrit Agrawal et.al [2] represented the task scheduling in heterogeneous parallel and multiprocessor distributed computing environment was a challenging problem in practical. The applications which were identified by parallel tasks can be represented by directed-acyclic graphs (DAGs). The scheduling referred to the assignment of these parallel tasks on a set of surrounded heterogeneous processors connected by high-speed networks. Since task scheduling for allocation was an NP-complete problem and as a replacement for finding an accurate solution, some scheduling algorithms were developed based on heuristics having prime objective to reduce the overall execution time or schedule length. In this paper, the overall execution time (schedule length) of the job is minimized by using task duplication on top.

He Kun et. al. [8] presented an idea to duplicate tasks on multiple machines so that the results of the duplicated tasks are available on multiple machines to trade computation time for communication time and proposed a novel Task Duplication based Clustering Algorithm (TDCA) to improve the schedule performance by utilizing duplication task more thoroughly. TDCA improves parameter calculation, task duplication, and task merging. The analysis and experiments are based on randomly generated graphs with various characteristics, including DAG depth and width, communication-computing cost ration, and variant computation power of processors and improved the schedule makespan of task duplication-based algorithms.

Fatma A. Omara et.al [7] have represented that the scheduling and mapping of precedence-constrained task graph to processors and was the most critical NP-complete problem in parallel computing systems. A number of genetic algorithms have been developed to resolve this problem. However, these algorithms are massive, as they try to examine the whole solution space without taking into account how to decrease the difficulty of the optimization procedure. In this method, two genetic algorithms have been developed and implemented. The developed algorithms with a number of heuristic principles added to increase the performance. According to the first genetic algorithm, two fitness functions have applied one after the other. The former fitness was for minimizing the total execution time and the latter one was concerned with the load balance satisfaction. The second genetic algorithm was based on a task duplication technique to prevent the communication overhead. According to the combined results, it has been found that this algorithm always better than the traditional algorithms in the case of scheduling parameters.
Jagpreet Singh et. al. [11] proposed a Contention-aware, Energy Efficient, Duplication based Mixed Integer Programming (CEEDMIP) formulation for scheduling task graphs on heterogeneous multiprocessors, interconnected in a distributed system or a network on chip architecture. The effect of duplication was studied with respect to minimizing: the makespan, the total energy for processing tasks and messages on processors and network resources respectively and the tardiness of tasks with respect to their deadlines. Optimizing the use of duplication with MIP provided both energy efficiency and performance by reducing the communication energy consumption and the communication latency. The contention awareness gave more accurate estimation of the energy consumption and also proposed a corner case that allows the scheduling of a parent task copy after a copy of the child task which may lead to efficient schedules. It has been observed that the proposed MIP with a clustering based heuristic provide scalability and gave 10-30 percent improvement in energy with improved makespan and accuracy when compared with other duplication based energy aware algorithms.

Oliver Sinnen et.al [4] proposed a contention-aware task duplication scheduling algorithm. They investigated the essentials for task duplication in the contention model and presented an algorithm that was based on modern techniques found in task duplication and contention-aware algorithms. Task duplication is a technique that has been employed to reduce or avoid to inter-processor communication. Some tasks are duplicated on the processors to create the information locally and avoid the message among processors. This contention model introduced the contention awareness into task scheduling by allocating the ends of the task graph to the relations of the communication network. It is necessary that scheduling under such a model performed even more from task duplication. However, there was no such algorithm has been proposed as it is not slight to duplicate tasks over the contention model.

Singh Rachhpal [17] proposed a performance effective genetic algorithm approach to schedule parallel tasks on the heterogeneous multiprocessor system in cloud computing environment. Here to optimize the job scheduling issue, hybrid metaheuristics with job duplication strategy is proposed that uses the features of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Variable Neighborhood Search (VNS). To simulate this a cloud-based model is designed by considering well-known Fast Fourier Transformation (FFT) problem using Directed Acyclic Graph (DAG). Further, the job duplication can reduce the inter-processor communication. Extensive experiments show that the proposed technique outperforms over the other well-known scheduling approaches.

Sasmita Kumari Nayak et.al [14] dealt with the problem of dynamic task scheduling in the grid of multi-processors. Initially, this method illustrated task scheduling as a complex problem and after optimized with a new hybrid algorithm. The proposed algorithm comprises the merits of bacterial foraging optimization and genetic algorithm method. The simulation results showed that it gave good results than other algorithms.

P. Chitra et.al [3] represented the task scheduling problem in heterogeneous systems is a multi-objective optimization problem. They found that the multi-evolutionary algorithms are well suited for solving multi objective task scheduling problem. The two main objectives namely, makespan and reliability were considered. The presentation of this method can be improved by using local search. Simple neighborhood search algorithm is used as the limited search algorithm. The weighted-sum based approach for solving the multi-objective optimization problem with its hybrid version is compared. Then the two multi objective evolutionary algorithms namely SPEA2 and NSGA-II are comparing with each other in the wholesome and hybrid edition. The simulations confirmed that Hybrid NSGA-II was best suitable for resolving the task scheduling problem.

Nourah Al-Angari et.al [1] illustrated that task scheduling was the most challenging problem in the parallel computing resulting in the inappropriate scheduling. It reduced or even aborts the utilization of the parallelization. Hence, they proposed synchronous master-slave algorithm to minimize a task scheduling problems. Genetic algorithm was successfully applied to solve that problem. The evaluation process of fitness function took considerable time, which affects the GA performance. The proposed algorithm outperforms the sequential algorithm in case of the complex and high number of generation problem.

Arash Ghorbannia Delavar et.al [19] proposed a hybrid meta-heuristic mechanism based on Genetic Algorithm (GMSW) in order to find an appropriate solution for mapping the requests on resources. The proposed method was trying to acquire the reply quickly with some goal-oriented operations. The makespan and reliability were the two problems to be optimized by considering a good distribution of workload on resources. The experimental results indicated that the GMSW improves the results than the other traditional algorithms, with the increasing number of tasks in graph simulation for the respective objective functions.

Pelin Alcan et.al [16] proposed a genetic algorithm based machine code for reducing the execution times in non-identical task scheduling problem. It has shown considerable advantages in resolving the optimization problems fit for practical application. Two different arithmetical examples revealed the proposed genetic algorithm was efficient and robust for large scale identical parallel machine scheduling problem for decreasing the makespan. Moreover, triangular fuzzy processing times were used in order to adapt the GA to non-identical parallel machine scheduling problem.

Probir Roy et.al [9] represented that finding the best solution for scheduling the tasks among the processors is NP-complete; that is, it will take more time to obtain the optimal solution. For that case, they proposed a heuristic for task scheduling based on genetic algorithm in multiprocessor systems by choosing the eligible processor on educated guess. The comparison results found that this new heuristic based genetic algorithm takes less calculation time to meet the suboptimal solution.

Ranjit Rajak [18] presented a scheduling algorithm which was known as job scheduling based on Breath First Search (TSB) to avoid the NP- Complete problem. The TSB was queue based mechanism to schedule parallel tasks in the homogeneous multiprocessor system. Its performance was evaluated in comparison with Dynamic Level Scheduling, Highest Level First with Estimate Time algorithm and Modified Critical Path algorithm in terms of efficiency, speed, load balance and optimized execution time.

Xu et. al. [13] described a good scheduling multiple priority queues *genetic algorithm* (MPQGA) that efficiently assigned a priority to each subtask depending on the resources needed to minimize makespan and exploit the advantages of both evolutionary-based and heuristic-based algorithms while avoiding their drawbacks for the scenario of *directed acyclic graph* (DAG) scheduling.

Tariq et. al. [12] proposed a new heuristic based on DAG is proposed for task scheduling on tasks order, average communication cost and best available processor/resource. To achieve the efficiency tasks were assigned to best suited processor while mini mizing communication cost. Directed Acyclic Graph (DAG) was also used to solve the issue of performance in distributed networks and the experimental study showed the proposed approach gave promising results by comparing the schedule time, efficiency and schedule length with other well-known algorithms for task scheduling.

Akbari et. al. [15] presented a genetic-based algorithm as a meta-heuristic method to address static task scheduling for processors in heterogeneous computing systems and improved the performance of GA through significant changes in its genetic functions and introduction of new operators that guarantee sample variety and consistent coverage of the whole space. The results of running this algorithm on the graphs of real-world applications and random graphs in heterogeneous computing systems with a wide range of characteristics, indicated significant improvements of efficiency of the proposed algorithm compared with other task scheduling algorithms.

3 THE PROPOSED APPROACH

There are several scheduling algorithms exist in allocating the tasks to heterogeneous parallel systems. Even with an efficient scheduling algorithm, some processors might be inoperative or idle throughout the execution of the parallel task allocation because the tasks assigned to them might be waiting to collect some data from the tasks running in other processors.

According to the proposed algorithm, a good schedule based on task duplication has been proposed. This proposed algorithm called the Duplication Based Genetic Algorithm with Load Balance (DBGA-LB) that employs genetic algorithm for solving the scheduling problem and applying load balance alteration to assure the efficient load balance between the parallel multiprocessor systems. The DBGA-LB algorithm has the characteristics of identify the critical tasks and redundantly assigned to idle time slots of a waiting processor. This mechanism utilizes the idle time between the parallel processors to reduce the schedule length of the processors.

The basic structure of the proposed DBGA-LB algorithm has been categorized into six major components. They are (i) Produce the input having weight and communication cost of each task t_i randomly to be allocated in different parallel processors p_j. (ii) Initialize the chromosome structure up to population size N by applying duplication of some tasks from the tasks list randomly to schedule in available processors without duplicating it in same processor. (iii) The fitness function plays the vital role to select the appropriate chromosomes with the help of objective function. Here the objective function to be optimized is schedule length and load balance respectively. (iv) Select the certain number of chromosomes or individuals having best fitness value among the initial population. (v) Finally, the genetic operations like crossover and mutation is applied to maintain diversity among the chromosomes. These procedures can be carried out until the specified number of generations G is reached. Fig. 2 illustrates the basic structure and Fig. 3 shows the flow chart of proposed DBGA-LB algorithm is as follows. Here in the basic structure from the start of the input generation, by taking initial population and evaluation of the fitness function or fitness value. Further selection procedure be illustrated with crossover and mutation operators and at end termination criteria be set.

3.1 BASIC STRUCTURE

Step 1: Input Generation

Step 1.1: Generate the task list t_i of weight and communication cost for processors p_j

Step 1.2: Acquire the task list as an input for DBGA-LB approach

Step 2: Initial Population

Step 2.1: Obtain the chromosome structure by duplicating certain tasks randomly to available processors p_j without doing it in same processor

Step 2.2: Initialize the population size N

Step 2.3: Create the initial population of chromosomes up to N

Step 3: Evaluation of Fitness

Step 3.1: Estimate the fitness of schedule length among the population

Step 3.2: Compute the load balance fitness function

Step 3.3: Accumulating both fitness value of N chromosomes for selection

Step 4: Selection Process

Step 4.1: Initialize the size of chromosomes s selected among the initial population

Step 4.2: Choosing the best chromosomes up to s with respect to fitness

Step 5: Genetic Operators

Step 5.1: Apply One-point Crossover using R_{CR}

Step 5.2: Perform Random Mutation using R_{MU}

Step 6: Termination

Step 6.1: Allocate tasks t_i to parallel processors p_j, if generations G is achieved

Step 6.2: Go to Step 3 until the termination criteria G is met

Fig. 2: Basic Structure of the proposed DBGA-LB algorithm

3.3 DUPLICATION BASED GENETIC ALGORITHM WITH LOAD BALANCE

Initially, the proposed DBGA-LB approach for scheduling parallel tasks needs the input task list of multiple tasks that have the communication cost and their respective weights. The weight of the task denotes the computation time. The communication cost of a

task illustrates the weight over the edge between two tasks in Directed Acyclic Graph (DAG). This cost incurred when the respective two tasks are allocated in different processor and it becomes zero, if both tasks are scheduled to same processor or machine.

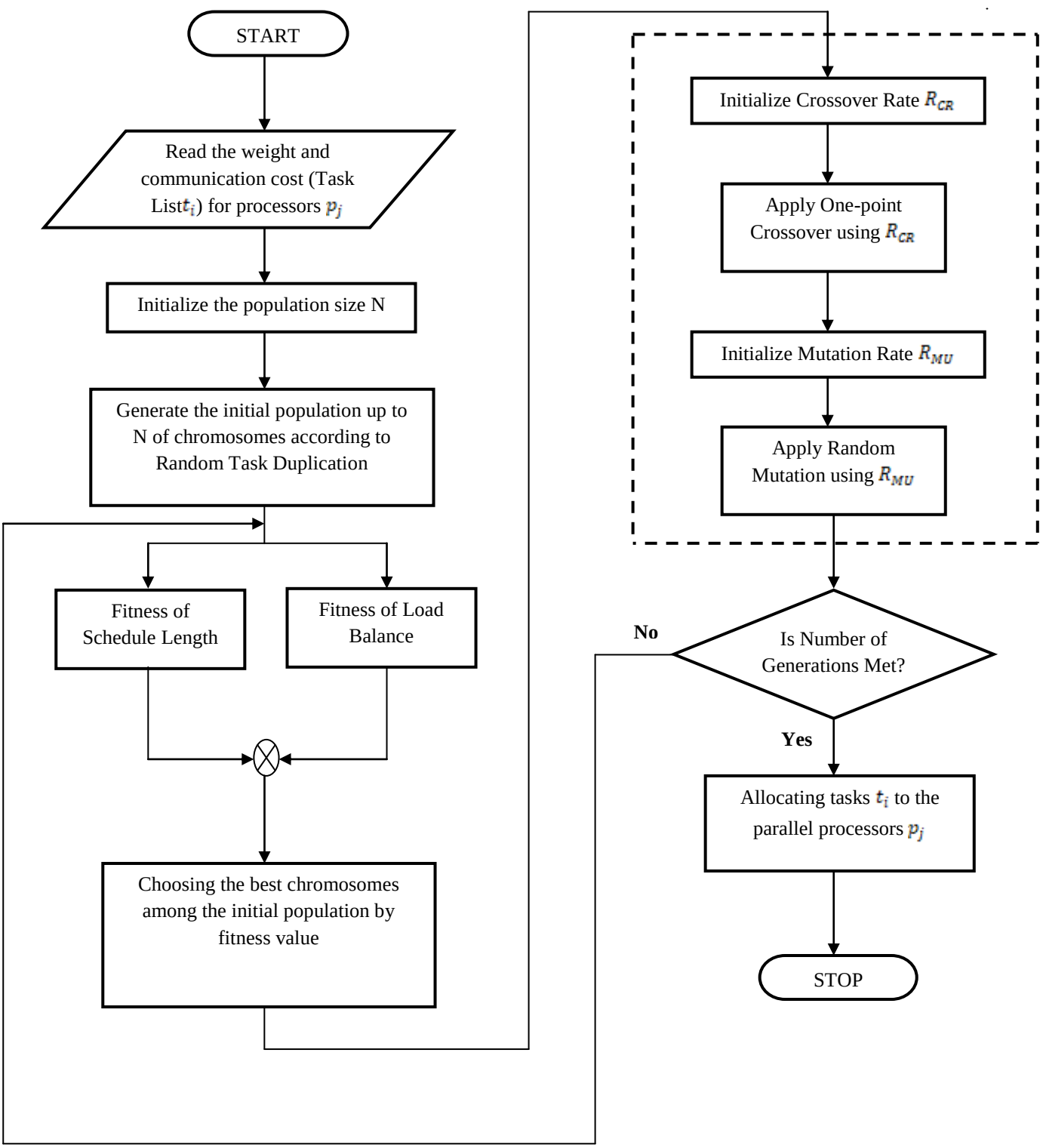

Fig. 3: Flowchart of proposed DBGA-LB algorithm

The representation of weight and communication cost are w (t_i,p_j) and c (t_i,p_j) respectively. Fig. 4 denotes the list of notations used as follows.

3.3.1 Initial Population

In contrary to standard genetic algorithm, the proposed DBGA-LB algorithm initializes their population with a distinct manner according to the problem specification. The parameters which can be focused in parallel systems are schedule length and idle time. To overcome these issues in task allocation, the proposed mechanism performed the duplication of some tasks among the task list randomly. The chromosome structure can be formed by continuing the task duplication to be scheduled in available machines without allocate the replicated task in same processor itself.

Nomenclature

t_i	i number of tasks
p_j	j number of processors
$w(t_i, p_j)$	Weight of task i allocated to processor j
c(t_i, p_j)	Communication cost for task i allocated to processor j
$R\_Time\ (p_j)$	Ready time of processor j
$S\_Time\ (t_i)$	Starting time of task i
$E\_Time\ (t_i)$	Ending time of task i
N	Size of initial population
G	Number of generations or iterations
R_{CR}	Rate of crossover
R_{MU}	Rate of mutation

Fig. 4: Nomenclature of notations used in proposed DBGA-LB approach

This process of constructing the chromosomes is expanding until the population size N. The sample chromosome structure that undergoes task duplication is depicted in Fig. 5.

$$(t_1, p_3), (t_5, p_1), (t_3, p_2), (t_2, p_1), (t_4, p_3), (t_1, p_2), (t_3, p_1)$$

Fig. 5: Example of Task-Duplication Chromosome (Tasks=5, Processors=3)

3.3.2 Computing Fitness Function

The fitness calculation of each chromosome in the initial population is the major process carried out in genetic algorithm. It contributes dominantly in the field of task scheduling in parallel systems because it executes according to the function to be optimized. The formula for calculating the total fitness of the proposed DBGA-LB approach is as follows:

$$Total\ Fitness = \frac{1}{(F_1 + F_2)} \quad (1)$$

Here, there is two fitness function used F_1 and F_2 one after the other to extract the best results. They are (i) Schedule length and (ii) Load balance.

3.3.2.1 Schedule Length (F_1)

The main objective of this function is to obtain the chromosomes of best fitness value in terms of schedule length. In this case, the individuals or chromosomes are selected according to lower schedule length (or) maximum ending time of tasks among the available processors. It can be defined as follows.

$$Schedule\ Length = max\ \{E\_Time\ (t_i)\} \quad (2)$$

In the above equation, $E\_Time\ (\ t_i\ )$ represents the ending time of latest task running on the processors p_j. Therefore, this first fitness processes their action by finding the chromosomes of reduced schedule length between the initial populations as carried by task duplication.

3.3.2.2 Load Balance (F_2)

The load or task balance is the second fitness function used to satisfy load balance between parallel machines. Several approaches produced the optimized schedule length, but the task balance between heterogeneous processors might not be satisfied in some of them. To overcome these effects, the proposed DBGA-LB algorithm can take load balance modification to obtain the less schedule length and simultaneously, the balance of load (tasks) is also satisfied.
It is calculated by the proportion of the schedule length (i.e. maximum execution time) to the average ending time of tasks over all the processors.

$$Task\ Balance\ = \ Schedule\ Length \Big/ Average\ Execution\ time\ of\ p_j \qquad (3)$$

The average execution time of processors is defined as the ratio of sum of ending time of tasks running on the processors to the number of parallel processors as follows.

$$Average\ Execution\ time\ of\ p_j = \frac{\sum_{j=1}^{No.of.Processors} Execution\ time\ [No.of.Processors]}{No.of.Processors} \qquad (4)$$

The load balance fitness can mine the individuals having lesser load balance value to provide the optimized results while compared to the existing methods.

3.3.3 Selection of Chromosomes

The chromosomes or individuals among the initial population need to be selected to perform the genetic operations like crossover and mutation. To achieve this action, the proposed DBGA-LB approach can filter and pick out the individuals according to higher fitness values from the original population until the particular number of chromosomes. Before going it into the next iteration of population, the chosen chromosomes will undergo genetic operations to produce the child (offspring) chromosomes to presence in the further generations of population.

3.3.4 Genetic Operators

There are two genetic operators who contribute extremely in the genetic algorithm to optimize the objective function. They are (i) crossover and (ii) mutation. In the proposed approach, the schedule length and load balance is the objective function to be concentrate for generating the better results than the traditional approaches. These operations are performed in order to maintain the diversity or variations among the individuals while the generation of same chromosomes occurs. The probability of crossover and mutation (R_{CR},R_{MU}) can be initialized in every generation of population.

3.3.4.1 Crossover

Crossover is the process of altering the chromosome structure in selected individuals among the initial population using the crossover rate (probability) R_{CR}. In the proposed DBGA-LB algorithm, the probability of crossover R_{CR} can be defined between 0.5 and 0.9 to receive the good results.

There are several methods exist to carry the crossover mechanism. In the proposed method, one-point crossover is performing to restrict the production of same individuals among them. In one-point crossover, the crossover point will be selected randomly between 1 and number of tasks in chromosomes that take part in crossover. After this point of crossover is chosen among the individuals, the part of the chromosome after the crossover point is interchanged between them to produce the child (offspring) chromosomes.

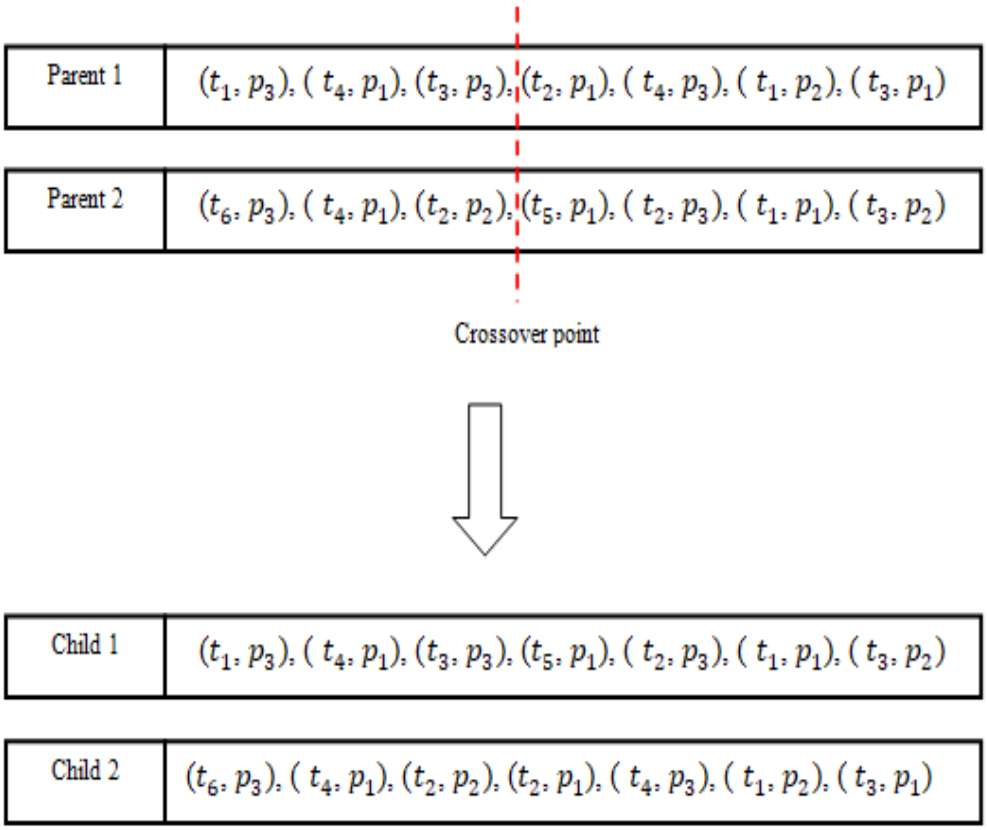

Fig. 6: Crossover (one-point) operation

In the above Fig. 6 the one-point crossover can be illustrated with respect to the crossover point chosen in random.

3.3.4.2 Mutation

Mutation is commonly carried out in genetic algorithm for eliminating the task scheduling problems occurred in parallel and connected systems. The mutation probability R_{MU} indicates the probability of the task-processor pair in the chromosomes to be changed. The probability of mutation is basically between 0.001 and 0.005. The point at which the mutation takes place is called mutation point, and it can be chosen randomly in an individual. The mutation process illustrates in Fig. 7. In the above Fig. 7, tasks 3 alter its allocated machine from p_2 to p_3 with a result of mutation. Similarly, the mutation can happen in respective generations of chromosomes with rate R_{MU}.

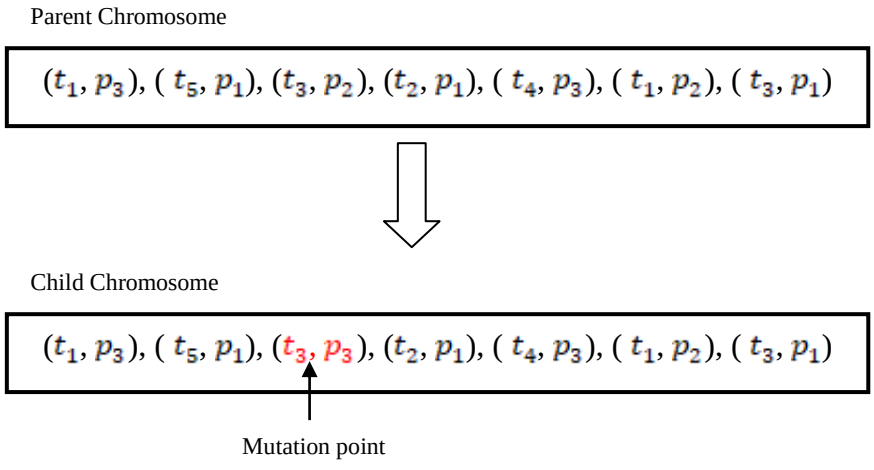

Fig. 7: Process of Random Mutation

3.3.5 Termination of Task Scheduling

The termination is the process of ending the genetic search for obtaining the optimum solutions. According to the standard genetic algorithm, the common criteria for predicting the termination conditions are (i) fixed number of iterations and (ii) obtained minimum objective function.

In the proposed DBGA-LB algorithm, the chosen termination is the fixed number of generations to gather the optimized results. If the condition of termination is satisfied, then the scheduled tasks are allocated to their corresponding machines or processors as defined by the duplication based load balancing genetic algorithm. Otherwise, the searching of genetic solution procedure can take place from the execution of fitness evaluation until the termination criterion has been reached.

4 PERFORMANCE EVALUATION

4.1 Problem Evaluation

The proposed Genetic Algorithm with task duplication in parallel computing is simulated in NetBeans IDE (Java) Environment. The proposed task duplication algorithm is applied using tasks communication cost and tasks weight which are taken randomly generated tasks communication cost as well as tasks weight in task table. This work also considered the comparisons between proposed GA with task duplication, GA without task duplication and FCFS algorithm without task duplication. In the simulated task duplication algorithm, the first is duplicated and then the reliability of task duplication can enhance by duplication of more tasks dynamically in order to reduce the schedule length and load balancing of the parallel processors.

4.2 The developed GA with Task Duplication

The proposed GA with task duplication algorithm, GA without task duplication algorithm and FCFS without task duplication algorithm have been implemented and compared. The proposed algorithm dynamically allocated the tasks to the processors according to the fitness functions. By default, the first task is duplicated in each and every available processor and also other tasks can be duplicated as per user reliability. The tasks in the processor are allocated by tasks weight and its communication cost between the adjacent tasks.

The proposed method finds the ideal time of each processor that running parallel and duplicates the tasks in ideal time of the processor in order to reduce the schedule length and load balancing. The comparisons between algorithms are taken according to the fitness function of load balance and schedule length.

The Table 1 explains the comparison between proposed GA with task duplication, GA without task duplication and FCFS without task duplication on load balance, schedule length, speedup, utilization and efficiency. The table clearly shows the proposed algorithms proficiency about task duplication method in parallel processor.

The ultimate aim of this research work is to reduce the schedule length by duplication of task in parallel processor. The Fig. 8 shows the comparison of schedule length between the proposed algorithm by varying duplication tasks with GA without task duplication and FCFS without task duplication. The bar graph shows the proficient reduction of schedule length that compared with dynamic tasks and dynamic duplicated tasks.

Table 1: Comparison between proposed algorithms, GA and FCFS algorithms with 3 processors

Parameters Analyzed with Three Processors		GA with Task Duplication			GA without Task Duplication	FCFS
		3	4	5		
For 10 Tasks	Efficiency	0.2115	0.1842	0.30952	0.1181	0.1936
	Load Balance	1.0206	1.0327	1.125	1.06363	1.7429
	Schedule Length	33.0	21.0	39.0	78.0	165.0
	Speed Up	0.6346	0.5526	0.9285	0.3545	0.5809
	Utilization	0.9797	0.9682	0.8888	0.9401	0.5737
For 15 Tasks	Efficiency	0.3205	0.4492	0.36111	0.1238	0.1441
	Load Balance	1.0765	1.1204	1.0893	1.1442	1.2972
	Schedule Length	75.0	62.0	65.0	130.0	288.0
	Speed Up	0.9615	1.3478	1.08333	0.3714	0.4324
	Utilization	0.9288	0.8924	0.9179	0.8974	0.7708
F	Efficiency	0.4942	0.5	0.5384	0.1247	0.1399

	Load Balance	1.0886	1.0	1.0	1.1229	1.2598
	Schedule Length	86.0	66.0	42.0	140.0	223.0
	Speed Up	1.4827	1.5	1.6153	0.3743	0.4199
	Utilization	0.9186	1.0	1.0	0.8904	0.7937

Fig. 9 shows the Load Balancing with varying values between number of tasks and number of processors. The proposed algorithm also designed for load balancing, the tasks are allocated to the parallel processor are equally balanced for the effective utilization of the processor. The above bar graph shows the tasks are balanced proficiently using proposed algorithm.

The comparison between the proposed algorithm with GA without task duplication and FCFS without task duplication with respect to the speedup with different number of

processor has been calculated. The speedup of the process is defined as:

$$Speedup = \frac{S_L}{Avg(P_T)} \quad (5)$$

where S_L is the schedule length of the parallel processors and $Avg(P_T)$ is the average processors time.

The speedup of the parallel processors increases when the numbers of tasks were duplicated and the proposed algorithm brings quicker task allocation.

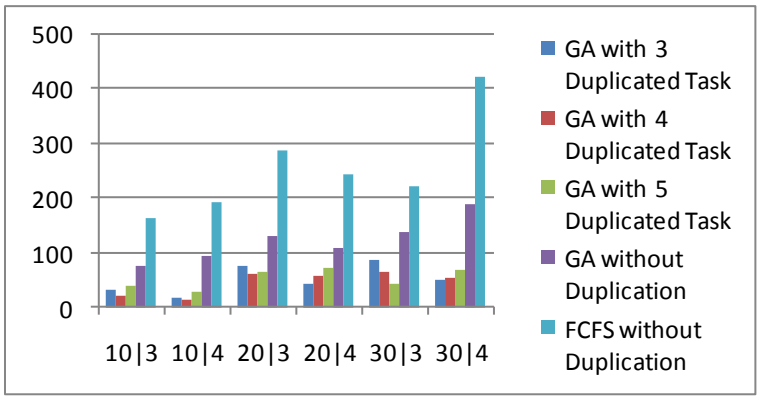

Fig. 8: Schedule Length with varying tasks and processors number (No. of Task | No. of Processor)

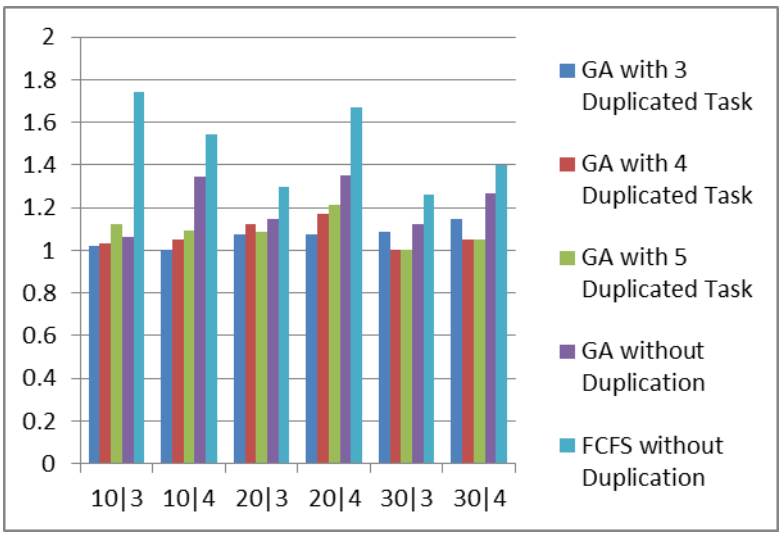

Fig. 9: Load Balance with varying tasks and processors number (No. of Task | No. of Processor)

The bar graph in Fig. 10 shows the speedup of the parallel processor that increases when the number of tasks gets duplicated. The GA without task duplication and FCFS without task duplication shows less speed when compare to the proposed task duplication algorithm.

The Fig. 10 shows the goodness of the proposed algorithm with three processors, dynamic task duplication from one to five and with various numbers of tasks to schedule in parallel processor.

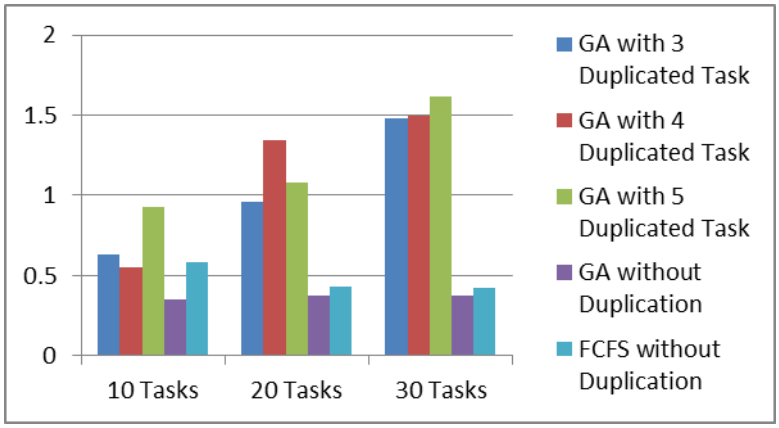

Fig. 10: Comparison on Speedup with 3 processors

The Table 2 delivers the comparison between proposed GA with task duplication, GA without task duplication and FCFS without task duplication on load balance, schedule length, speedup, utilization and efficiency. The table comparisons are taken throughout with four processors with dynamic number of tasks and duplicated task. The operation done by inputting 10 tasks with four processors having GA with task duplication with three, four and five duplicated tasks. Further same procedure or experiment be done by taking 15 tasks on same four processors by considering GA with task duplication by inputting three, four and five duplicated tasks so that a good comparisons takes place in between GA without task duplication and first come first technique. Also same procedure was applied by considering 20 tasks with same four processors having three, four and five duplicated tasks compared to the FCFS mechanism and GA without task duplication approach. Well-designed experiment and algorithmic approach shows the results in Table 2 so that a good decision can takes place.

The comparison between the algorithms with respect to the efficiency of the parallel processors has been carried out. The efficiency of the parallel processor is calculated as:

$$Efficiency = \frac{Speedup}{Total\ number\ of\ processor} \quad (6)$$

The efficiency formula derives the speedup of the processor with the all parallel processor is taken into account. As soon as efficiency results are better in the case of GA with task duplication as compared to the GA without task duplication and third scheduling mechanism first come first serve, the a decision for right choice was taken to apply the proposed algorithm in a better way.

Table 2: Comparison between proposed algorithm, GA and FCFS algorithms with 3 processors

Parameters Analyzed with Four Processors		GA with Task Duplication			GA without Task Duplication	FCFS
		3	4	5		
For 10 Tasks	**Efficiency**	0.1136	0.1052	0.1136	0.0842	0.0965
	Load Balance	1.0	1.049	1.0909	1.3476	1.544
	Schedule Length	20.0	16.0	30.0	94.0	193.0
	Speed Up	0.4545	0.4210	0.4545	0.3369	0.386
	Utilization	1.0	0.9531	0.9166	0.7420	0.6476
For 15 Tasks	**Efficiency**	0.1791	0.1785	0.1764	0.0843	0.1045
	Load Balance	1.075	1.1707	1.2100	1.3496	1.6723

	Schedule Length	43.0	60.0	72.0	110.0	245.0
	Speed Up	0.7166	0.7142	0.7058	0.3374	0.4180
	Utilization	0.9302	0.8541	0.8263	0.7409	0.5979
For 20 Tasks	**Efficiency**	0.2840	0.1964	0.2083	0.0791	0.0875
	Load Balance	1.1494	1.0526	1.0526	1.2666	1.4003
	Schedule Length	50.0	55.0	70.0	190.0	425.0
	Speed Up	1.1363	0.78571	0.8333	0.3166	0.3500
	Utilization	0.87	0.95	0.95	0.7894	0.7141

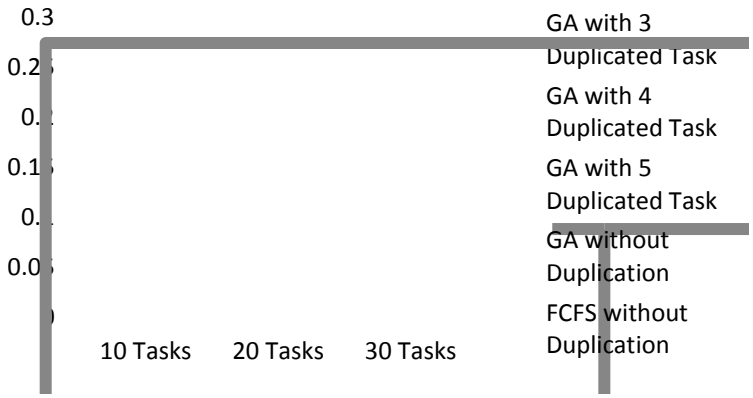

Fig. 11: Comparison on Efficiency with 4 processors

The efficiency of the parallel processors is high using proposed GA with task duplication algorithm when compared with GA without task duplication and FCFS without task duplication algorithms. The efficiency is highly improved when there is increase in the number of tasks duplicated as shown in the Fig. 11 and finally, the utilization of the parallel processor can be defined as:

$$Utilization = \frac{\sum FEP_{P_i}}{Total\ number\ of\ processor} \quad (7)$$

where FEP_{P_i} denotes for each processors scheduling time and FEP_{P_i} is calculated as:

$$FEP_{P_i} = \frac{P_i^{th} S_L}{S_L} \quad (8)$$

where $P_i^{th} S_L$ denotes i^{th} processors schedule length and S_L denotes the processors maximum schedule length among all parallel processors. The utilization of machines with tasks is highly improved when there is increase in the number of tasks duplicated as shown in the Fig. 12.

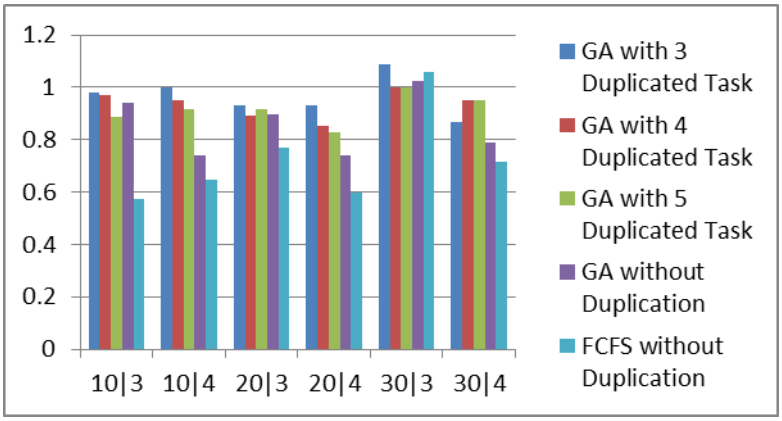

Fig. 12: Comparison on Utilization (No. of Tasks | No. of Processors)

The tasks are effectively utilizes the parallel processors in the proposed GA with task duplication algorithm rather than GA without task duplication and FCFS without task duplication. The overall comparison of proposed GA with task duplication algorithm with other GA without task duplication and FCFS without task duplication algorithms shows that task duplication leads to good balanced task in all processor, execution of speedup is geared, proficient utilization of parallel processor, efficiency of the proposed algorithm is greatly emerged and the schedule length of the tasks are ultimately reduced as shown in Fig. 13.

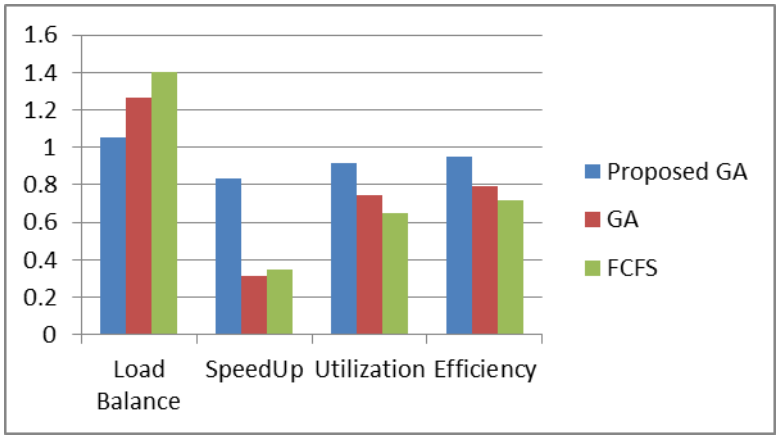

Fig. 13: Comparison with 20 tasks, 5 duplicated tasks and 4 processors

5 CONCLUSION

In this paper, an execution of duplication based genetic algorithm with load balance (DBGA-LB) approach for task scheduling in parallel heterogeneous resources has been presented. This can be allocated efficiently and addressed the problems occurred during task assignment in practical. The proposed method works according to task duplication genetic algorithm with inclusion of balancing the tasks (load balance) across the processors in parallel. The communication cost and weight of the tasks can be generated randomly as an input. The DBGA-LB algorithm obtained the duplication of tasks scheduled randomly in the available processors without being scheduled in the same processor itself as the initial population. The fitness function performed iterative until a fixed number of generations to pick the best schedule of chromosome with respect to schedule length and load balance. The genetic operations like crossover and mutation takes place as usual to maintain the diversity among the chromosomes. A comparative study between the traditional scheduling strategies like First Come First Serve (FCFS) and Genetic algorithm (GA) with our proposed DBGA-LB method has been presented. The experimental result shows that our proposed algorithm outperforms the other approaches in terms of schedule length, load balance, speedup and efficiency.

REFERENCES

1. Nourah Al-Angari, Abdul latif AL, 2012, Multiprocessor Scheduling Using Parallel Genetic Algorithm, International Journal of Computer Science, Issues 1, Vol. 9.
2. Amrit Agrawal, Pranay Chaudhuri, 2011, An Algorithm for Task Scheduling in Heterogeneous Distributed Systems Using Task Duplication, International Journal of Grid and High Performance Computing, Vol. 3, No. 1, pp 89-97.

3. P. Chitra, R. Rajaram, P. Venkatesh, 2011, Application and comparison of hybrid evolutionary multi-objective optimization algorithms for solving task scheduling problem on heterogeneous systems, Applied Soft Computing, Vol.11, pp 2725–2734.
4. Oliver Sinnen, Andrea To, Manpreet Kaur, 2011, Contention-aware scheduling with task duplication, Journal of Parallel Distributed Computing, Vol. 71, pp 77–86.
5. Mostafa R. Mohamed, Medhat H. A. Awadalla, 2011, Hybrid Algorithm for Multiprocessor Task Scheduling, International Journal of Computer Science Issues, Vol. 8.
6. Singh, Rachhpal. 2016, An optimized task duplication based scheduling in parallel system." *Int. J. Intell. Syst. Appl.(IJISA)*8, no. 8: 26-37.
7. Fatma A. Omara, Mona M. Arafa, 2010, Genetic algorithms for task scheduling problem, Journal of Parallel Distributed Computing, Vol. 70, pp 13-22.
8. He, Kun, Xiaozhu Meng, Zhizhou Pan, Ling Yuan, and Pan Zhou, 2019, A Novel Task-Duplication Based Clustering Algorithm for Heterogeneous Computing Environments, *IEEE Transactions on Parallel and Distributed Systems* 30, no. 1: 2-14.
9. Probir Roy, Md. Mejbah Ul Alam, Nishita Das, 2012, Heuristic Based Task Scheduling In Multiprocessor Systems With Genetic Algorithm By Choosing The Eligible Processor, International Journal of Distributed and Parallel Systems, Vol.3, No.4.
10. Zhao, Jianfeng, and Hongze Qiu, 2013, Genetic algorithm and ant colony algorithm based Energy-Efficient Task Scheduling, *IEEE (ICIST)*, pp. 946-950.
11. Singh, Jagpreet, Sandeep Betha, Bhargav Mangipudi, and Nitin Auluck, 2015, Contention aware energy efficient scheduling on heterogeneous multiprocessors, *IEEE Transactions on Parallel and Distributed Systems* 26, no. 5: 1251-1264.
12. Tariq, Rehan, Farhan Aadil, Muhammad Faizan Malik, Sadia Ejaz, Muhammad Umair Khan, and Muhammad Fahad Khan, 2018, Directed Acyclic Graph Based Task Scheduling Algorithm for Heterogeneous Systems, In *Proceedings of SAI Intelligent Systems Conference*, pp. 936-947. Springer.
13. Xu, Yuming, Kenli Li, Jingtong Hu, and Keqin Li, 2014, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, *Information Sciences* 270: 255-287.
14. Sasmita Kumari Nayak, Sasmita Kumari Padhy, Siba Prasada Panigrahi, 2012, A novel algorithm for dynamic task scheduling, Future Generation Computer Systems, Vol. 28, pp 709–717.
15. Akbari, Mehdi, Hassan Rashidi, and Sasan H. Alizadeh, 2017, An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems, *Engineering Applications of Artificial Intelligence*, 61: 35-46.
16. Pelin Alcan, Huseyin Bas Uligil, 2012, A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem, Advances in Engineering Software, Vol. 45, pp 272–280.
17. Singh, Rachhpal., 2019, Hybrid Metaheuristic Based Scheduling with Job Duplication for Cloud Data Centers., In *Harmony Search and Nature Inspired Optimization Algorithms*, pp. 989-997. Springer, Singapore.
18. Ranjit Rajak, 2012, A Novel Approach for Task Scheduling in Multiprocessor System", International Journal of Computer Applications, Vol. 44.
19. Arash Ghorbannia Delavar, Yalda Aryan, 2012, A Goal-Oriented Workflow Scheduling in Heterogeneous Distributed Systems", International Journal of Computer Applications, Vol. 52.
20. Sharma, Manik, Gurvinder Singh, Rajinder Singh, and Gurdev Singh. "Analysis of DSS Queries using Entropy based Restricted Genetic Algorithm." Appl. Math 9, no. 5 (2015): 2599-2609.
21. Sharma, Manik. "Role and Working of Genetic Algorithm in Computer Science." International Journal of Computer Applications and Information Technology (IJCAIT) 2.1 (2013).
22. Sharma, Manik, et al. "Design and comparative analysis of DSS queries in distributed environment." 2013 International Computer Science and Engineering Conference (ICSEC). IEEE, 2013.
23. Sharma, M., G. Singh, and R. Singh. "Clinical decision support system query optimizer using hybrid Firefly and controlled Genetic Algorithm." Journal of King Saud University-Computer and Information Sciences (2018).
24. Kaur, Prableen, and Manik Sharma. "A Survey on Using Nature Inspired Computing for Fatal Disease Diagnosis."International Journal of Information System Modeling and Design (IJISMD) 8.2 (2017): 70-91.
25. Samriti Sharma. "Applications of Genetic Algorithm in Software Engineering, Distributed Computing and Machine Learning". International Journal of Computer Applications and Information Technology (IJCAIT).9.2:208-212.