Task Scheduling Techniques Based On Parallel Genetic Algorithm Approaches : A Review

Rachhpal Singh
Department of Computer Science and Applications
Khalsa College, Amritsar

ABSTRACT

In parallel computing, multifaceted tasks can be solved concurrently by a group of processors. Task scheduling is a major problem in parallel systems as execution of a task may depend on the completion of another task running at the same time. This paper reviews existing task scheduling techniques/algorithms with and without task duplication in the parallel computing domain as Genetic Algorithm based task scheduling (GATS), Master-Slave Genetic Algorithm (MSGA), Contention-aware task scheduling with task Duplication (CA-D), Node Duplication genetic algorithm (NGA) and Real time controlled Duplication Algorithm (RTCDA). The issues and challenges to the existing task scheduling algorithms in the parallel computing domain have been highlighted. The performance of these techniques is evaluated using parameters like efficiency, execution time, speedup, processor utilization, schedule length and load balancing. This paper proposes DBGA-LB (Task duplication based GA with load (task) balance) algorithm and also compares it with the existing algorithms on the basis of above mentioned parameters.

Keywords

Parallel Computing, Task Scheduling, Task Duplication, Genetic algorithm.

1. INTRODUCTION

Parallel computing is the use of two or more processors in combination to solve a single task. Parallel system can have multiple tasks where a task may depend on other tasks. To make optimal use of computing resources, one needs to efficiently schedule the tasks on the given multiprocessor computing systems. Task scheduling is of prime significance in multiprocessor parallel system. Efficient multiprocessor task scheduling is a long studied and difficult problem that continues to be a topic of considerable research (Bohler et al. 1999). In the task scheduling area, a program is represented as a directed acyclic graph, called a task graph (DAG), where the nodes represent the task and the edges represent the communications between the tasks. One of the main obstacles in obtaining high performance from message passing multicomputer systems is the inevitable communication overhead which is incurred when tasks executing on different processors exchange data. Given a task graph, duplication based scheduling can mitigate this overhead by allocating some of the tasks redundantly on more than one processor (Ahmad and Kwok, 1994, 1998). There are many algorithms that produce high quality solutions in case of homogeneous computing system (Ahmad and Kwok, 1998 and Bozdag et. Al, 2005). However scheduling in heterogeneous computing systems is far more complicated problem due to non-uniform processor speeds and communication link band width. Two of the most important classes of scheduling algorithms are list based and cluster based algorithms. List based scheduling is very popular for heterogeneous environments due to its low complexity and good quality of resulting schedules (Bajaj and Agarwal, 2004; Hagras and Janecek, 2004; Topcuoglu et. Al, 2002).

Genetic algorithm (GA) which is a meta-heuristic algorithm provides robust, stochastic solutions for numerous optimization problems (Holland, 1975; Goldberg, 1989; Bohler et. al, 1999; Sharma and Singh, 2015, Sharma M et al, 2018). A synchronous master-slave algorithm (MSGA) outperforms the sequential algorithm in case of complex and high number of generations' problem (Nourah Al-Angari and Abdullatif ALAbdullatif, 2012). To reduce the inter-processor communication in task duplication operations, tasks were scheduled using the contention-aware scheduling technique (CA-D) (Kaur and Sinnen, 2011) [10]. This approach improves the speedup of the produced schedules. In comparison to the existing deterministic scheduling techniques, Node duplication genetic algorithm (NGA) was exhibited to be efficient for minimizing cost of the execution in interprocessor traffic communication (Heidari and Chalechale, 2012). As a result, a new concurrent controlled duplication based heuristic called Real time controlled Duplication Algorithm (RTCDA) was used for scheduling tasks on heterogeneous multiprocessors (Singh and Auluck, 2011) [9]. This method does not make duplications in all the time decisions, it determines whether to duplicate or not by the deadlines of the tasks. The RTCDA utilizes the schedule openings which are created when a task is completed without duplication before time which leads to the advancement of success ratio.

This paper reviews various task scheduling techniques as the basis of with and without duplication. Some of the task scheduling algorithms like Genetic Algorithm (GATS or GA), Master-slave GA, Contention-aware task scheduling, NGA algorithm and RTCDA algorithm were analyzed and some problems were identified. DBGA-LB technique has been designed to overcome the challenging issues in above task scheduling techniques. According to this method, random task duplication is performed in certain tasks to utilize the idle time slot of the processors and thus simultaneously reducing the schedule length. This algorithm has the combination of schedule length and load or task balance as the fitness function. The load balance ensures that the number of tasks allocated to each processor is more or less equivalent to other processors. Finally, all the review techniques are compared with the DBGA-LB method for analyzing the performance. The proposed technique has outperformed the existing algorithms. The rest of the paper is organized as follows. Section 2 discusses five task scheduling algorithms like GATS/GA, MSGA, CA-D, NGA and RTCDA with and without task duplication. Section 3 reviews all the existing task scheduling techniques. In section 4 proposed algorithm DBGA-LB will be

discussed. In the section 5 performance evaluations is done and the results of proposed DBGA-LB is compared with existing approaches. In the last section 6 the overall results are concluded based on the different outputs for various techniques.

2. TASK SCHEDULING ALGORITHM

2.1 Genetic Algorithm based task scheduling (GATS or GA)

Bohler et. al, 1999 proposed improved multiprocessor task scheduling using genetic algorithm. For the effective execution of the set of assigned tasks, a new approach is encouraged with the well-known genetic approach called Heterogeneous Parallel Multiprocessor System. Heterogeneity of a processor means that the processor has different speeds or processing capabilities. The main objective is to minimize the total task finish time i.e. execution time + waiting time or idle time. A set of heterogeneous processors which are totally associated with each other through identical links and parallel applications are represented by a directed acyclic graph (DAG). Weights of vertices and edges are calculated to represent an execution duration and data communication respectively [19]. A Genetic Algorithm (GA) is used to resolve the scheduling problems like mapping, sequence of execution of the tasks, optimal configuration of the parallel system. Figure 1 shows the working procedure of a GA based task scheduling in which population strings are generated and the best population is selected by its fitness value with the objective functions[11][12][13]. Then the reproduction for the creation of a new population is carried out by cross-over and mutation operators. The crossover randomly chooses a pair of individuals among those previously selected and exchanges some parts of the information. The mutation takes an individual randomly and alters it. The task scheduling procedure using a GA is explained below:

1. Creation of population strings: Number of processors, number of tasks and population sizes are required to randomly generate initial population. Each task is randomly assigned to a processor.

2. Evaluation of Fitness Function: The fitness function separates the evaluation into two parts: **Task fitness:** All tasks are performed and scheduled in a valid order i.e. a pair of tasks is independent of the other tasks for execution. **Processor fitness:** The processor fitness component of the fitness function attempts to minimize the processing time.

3. Selection of the best string: The GA uses a selection operator and the individuals are selected according to their fitness value through the rotating roulette wheel strategy. This operator generates the next generation by selecting the best chromosomes from the parents and offspring.

4. Reproduction through cross-over and mutation: Crossover operators randomly select two parent chromosomes to produce two child (offspring) chromosomes. A mutation operation randomly selects two tasks and swaps them to reduce the idle time of the processor.

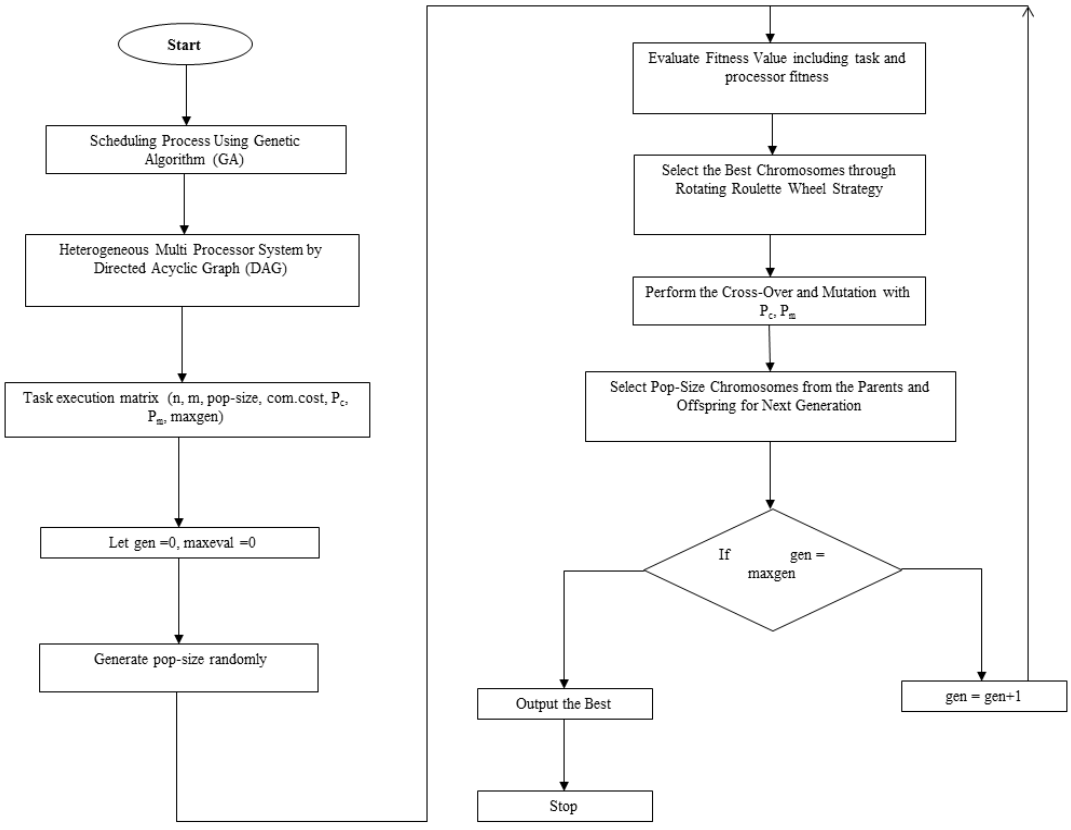

Figure 1: Genetic Algorithm based task scheduling (GATS)

2.2 Master-Slave Genetic Algorithm for task scheduling (MSGA)

The Genetic algorithm successfully schedules the task in the parallel system. However, fitness evaluation of the GA takes much more of the CPU's time and to overcome this problem the master-slave GA is proposed. Figure 2 shows the internal working principle of this algorithm. The master-salve GA scheduling will reduce the utilization of true potential parallelization. The main goal of the scheduler is to assign tasks to available processors by considering the precedence constraints between the tasks and also minimizing the overall execution time of tasks.

Nourah Al-Angari and Abdullatif ALAbdullatif, 2012, proposed MSGA, where the initial population is generated using the number of processors and number of tasks. This algorithm treats the master as the main processor and it stores the full population of chromosomes. It also assigns a certain fraction of the individuals to slave processors where the slaves evaluate the fitness value for the assigned fraction and return their values. During this stage communication is not needed because fitness of an individual is independent from the rest of the population. Theoretical analysis shows that the master-slave GA technique changes some of the genetic operators to significantly improve the overall performance.

This approach provides better performance if the parallel system's tasks never wait for all processors to finish their tasks.

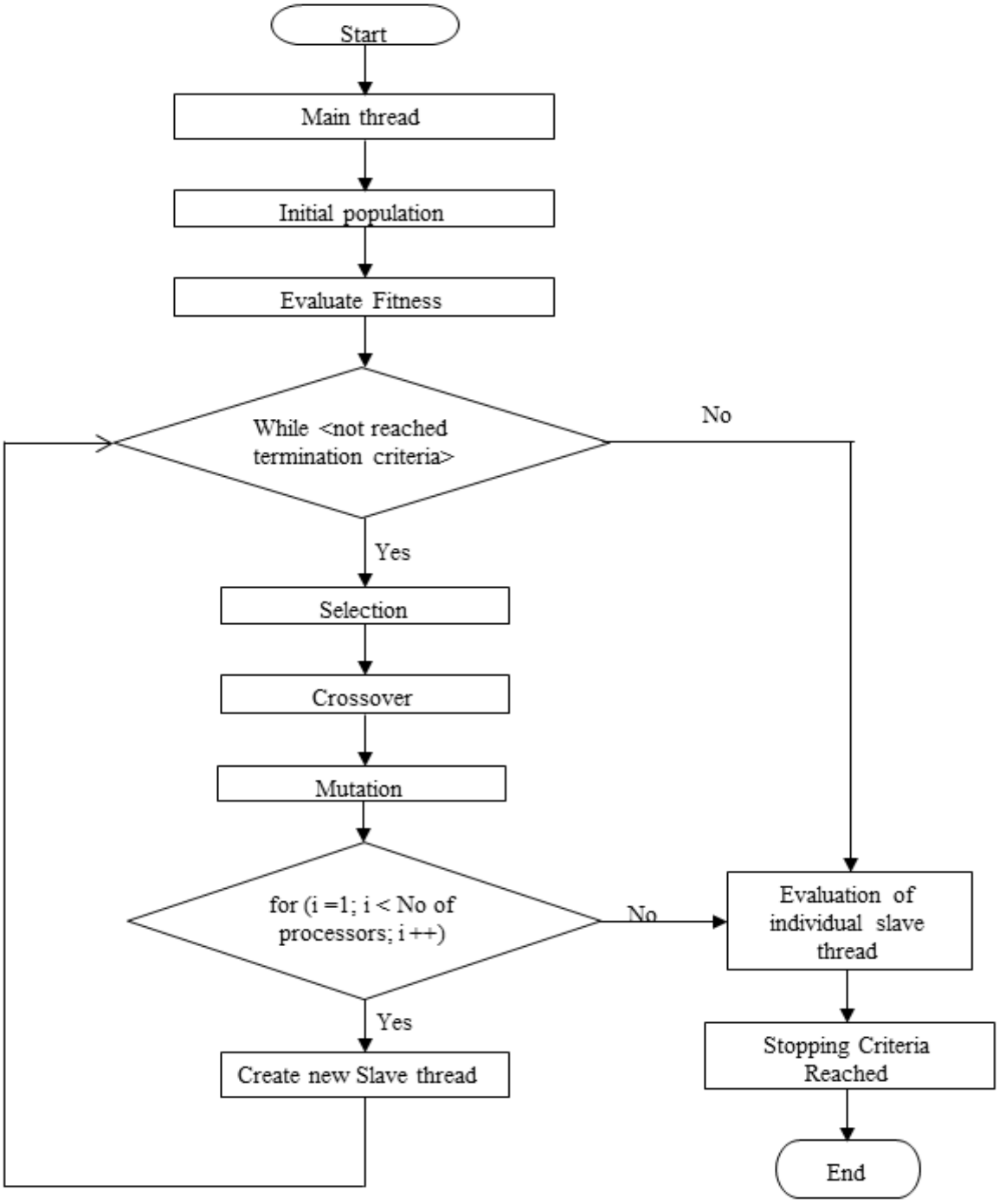

Figure 2: Master-Slave Genetic Algorithm Flow

2.3 Contention-Aware Scheduling with Task Duplication (CA-D)

Sinnen and kaur, 2011 used the contention model to schedule task scheduling with task duplication. This model consists of a set of processors P connected by the communication network TG = (P, L). This dedicated system must satisfy the following properties: local communications have zero costs and communication is performed by a communication subsystem. The main goal of the CA-D technique is scheduling tasks by maximizing the concurrency and minimizing the interprocessor communication. Figure 3 shows the working procedure of the CA-D technique. In the CA-D technique task duplication is done in two different phases. In the first phase, the nodes are ordered according to their bottom levels bl (n) and the second phase uses the "insertion technique". Each task can be scheduled between already scheduled tasks or after the finish time of processor P. Critical path nodes are identified for task duplication. In some situations, it can be more beneficial to not only duplicate the critical parent, but also consider the predecessors of the critical parent for duplication.

Task duplication algorithms consider the recursive duplication of the critical parent cp (n), its critical parent cp (cp (n)) and so on. Scheduling under the contention model has the need to tentatively schedule edges on the communication links in order to obtain the data ready time of a task n, i.e. the time when all incoming edges have finished communication. In order to find the best data provider, tasks are tentatively scheduled from each instance of the processor. Under the contention model, the removal of a task implies also that its in-edges can be removed from the links. Together with the insertion technique, the freed space can be used by subsequently scheduled tasks and their edges, potentially leading to shorter schedules. This algorithm checks for and removes redundant tasks after the scheduling of each task. Experimental evaluation shows that while comparing with other classical models, this algorithm speeds up the execution of tasks and also provides effective scheduling in the parallel system.

2.4 Node duplication Genetic algorithm (NGA)

Heidari and Chalechale, 2012, exhibited node duplication genetic algorithm to minimize the mean task response time. The NGA technique overcomes all the problems caused in the genetic algorithm. The intention of this scheduling is to find a way that the final time and cost of the execution have been minimized. The task (T) is divided into several non preemptive sub-tasks (t) to reduce the overall execution time. The initial population is generated using the count value of the task and the processor. The input values are initialized and fitness value is evaluated as in Figure 4. The fitness value can be evaluated for the individual task and processor. Using that fitness value the NGA technique duplicated the tasks. After that, all the chromosomes are sorted in descending order according to their fitness value. The NGA algorithm consists of eight different steps, they are explaining below:

1. Assume Generation $G_{counter} = 0$ and analyze the number of processors, Task execution matrix, communication cost and count of task.
2. Initialize Population: initial populations are created using the parameters such as the number of processors, number of tasks and size of population. Each individual consists of exactly one copy of each and every task
3. Estimate the objective and fitness function: the fitness function is calculated for the individual population.
4. Selection Operation: The Selection Operator sorts all the populations according to their fitness in the descending order.
5. Carry out crossover and mutation operations
6. Selection of population size of chromosomes from parents.
7. Generation counter G counter = G counter + 1
8. If G counter = max G counter, and exit with best solution and stop Else G counter < generation size, Go to 3

In the NGA method you must be aware of all the sub-task execution time and its precedence relation. Here the fitness function establishes to compute both the task fitness and the processor fitness. The Node duplication scheduling concept strongly recommends the multiple processors to achieve the optimal time unit of mean task response time. The task has been scheduled based on the fitness of each chromosome in heterogeneous multiprocessor parallel systems. The NGA task duplication algorithm effectively schedules the task in the parallel system and it also reduces the task duplication rate.

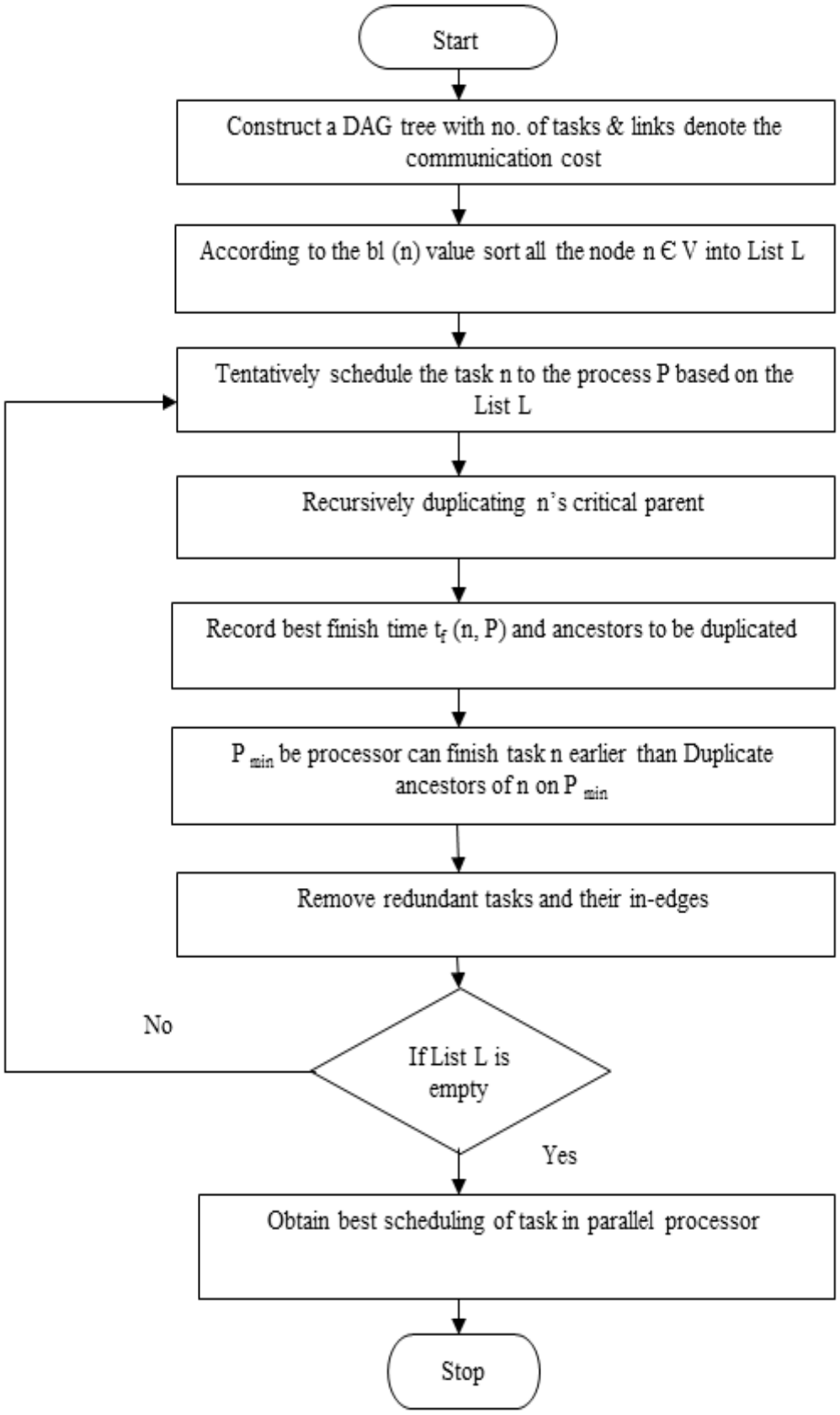

Figure 3: CA-D Algorithm Flow Chart

The performance analysis of the NGA is compared with the GA, FCFS, Priority and List scheduler with start time minimization shows that the NGA reduces the time when compared with other techniques.

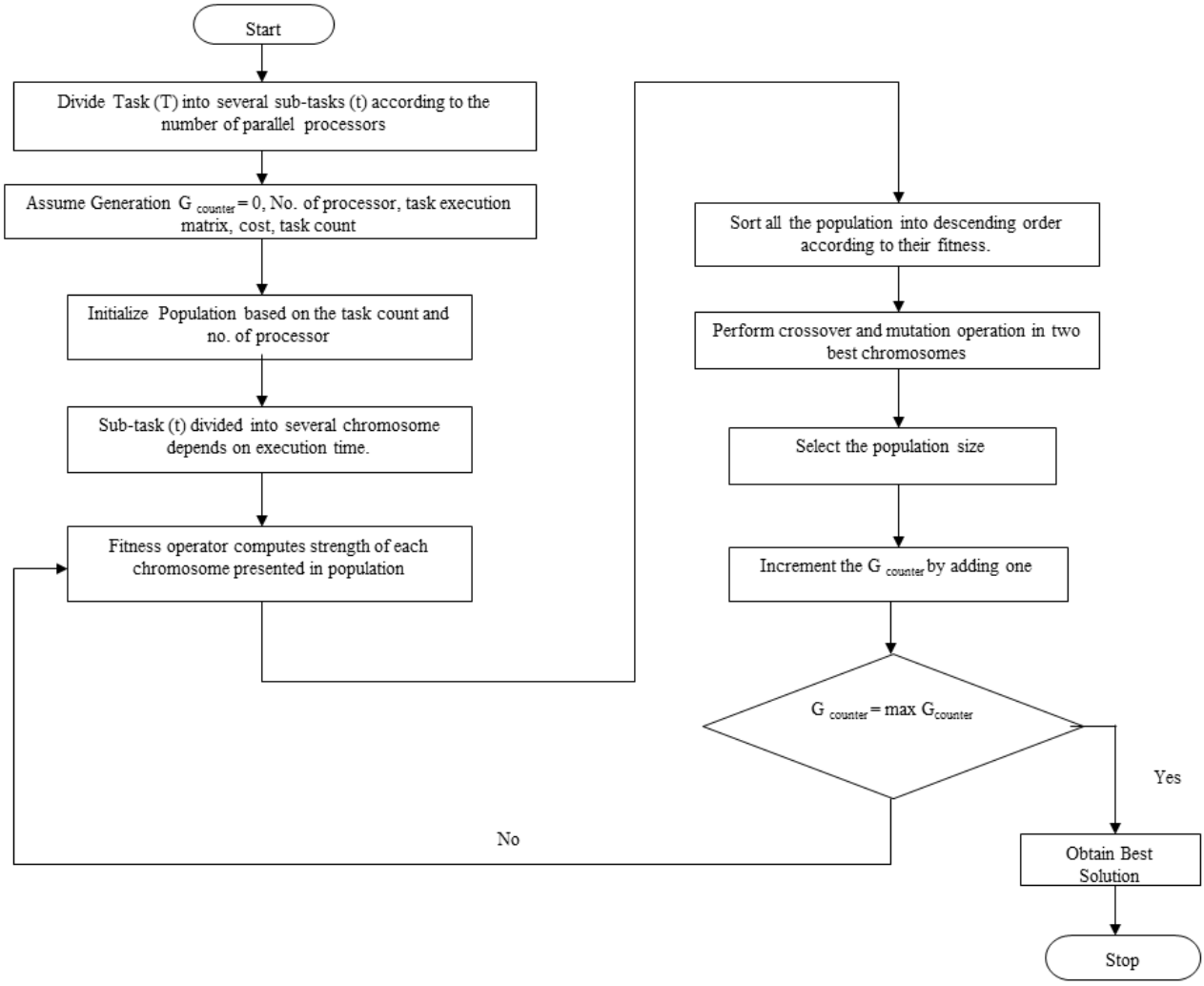

Figure 4: Node Duplication Genetic Algorithm (NGA) Flow Chart

2.5 Real-time controlled duplication based heuristic Algorithm (RTCDA)

Singh and Auluck,, 2011, worked on real time controlled duplication based heuristic technique. RTCDA scheduled periodic independent precedence-constrained tasks use schedule holes for scheduling and duplication. Duplication is not always required; it is decided by the deadlines of the tasks. RTCDA utilized schedule holes improve the success ratio and employ duplication to improve the schedulability. The scheduling algorithm allocate and schedule jobs to ensure that all the tasks meet their deadlines and to improve the schedulability. The precedence-constrained tasks are represented by a directed acyclic graph (DAG). The design of any duplication heuristic in a non real-time system has two significant steps i.e. where and how duplication is performed.

The tasks in the task set are considered for scheduling one at a time and are prioritized according to well known real-time scheduling heuristics. The two scheduling heuristics were known as earliest deadline first (EDF) and rate-monotonic (RM). The algorithm recursively attempts to duplicate the predecessors of any duplicated subtasks. As many ancestors as allowable were duplicated in a breadth-first fashion. Duplication recursively continues until no further duplication is possible. According to our observation a controlled amount of duplications could possibly improve the success ratio.

3. REVIEW OF EXISTING APPROACH

This section reviews the above stated task scheduling techniques on the basis of their scheduling approach performance. The GATS and MSGA techniques perform scheduling without any duplication of the tasks, but the NGA, RTCDA and CA-D techniques perform task scheduling by duplicating some tasks to avoid the idle time of the processor. Figure 6 shows the review techniques, scheduling procedure and their drawbacks. The drawbacks mentioned in the rectangle box are conquered by the proposed approach which is represented in the rounded rectangle box. In this technique, the Genetic Algorithm based task scheduling technique overcomes the NP-Complete problems in a parallel system. The main objective of this method reduces the execution time and increases the overall

throughput [14]. Tasks are scheduled to the processors based upon their fitness function. However, the fitness function depends on the communication delay and execution time of the tasks. This technique effectively solves the scheduling problem, but it takes much more time for computing the fitness value.

The master-slave Genetic algorithm was designed which considers one main processor as the master and all the other processors as slaves [15].

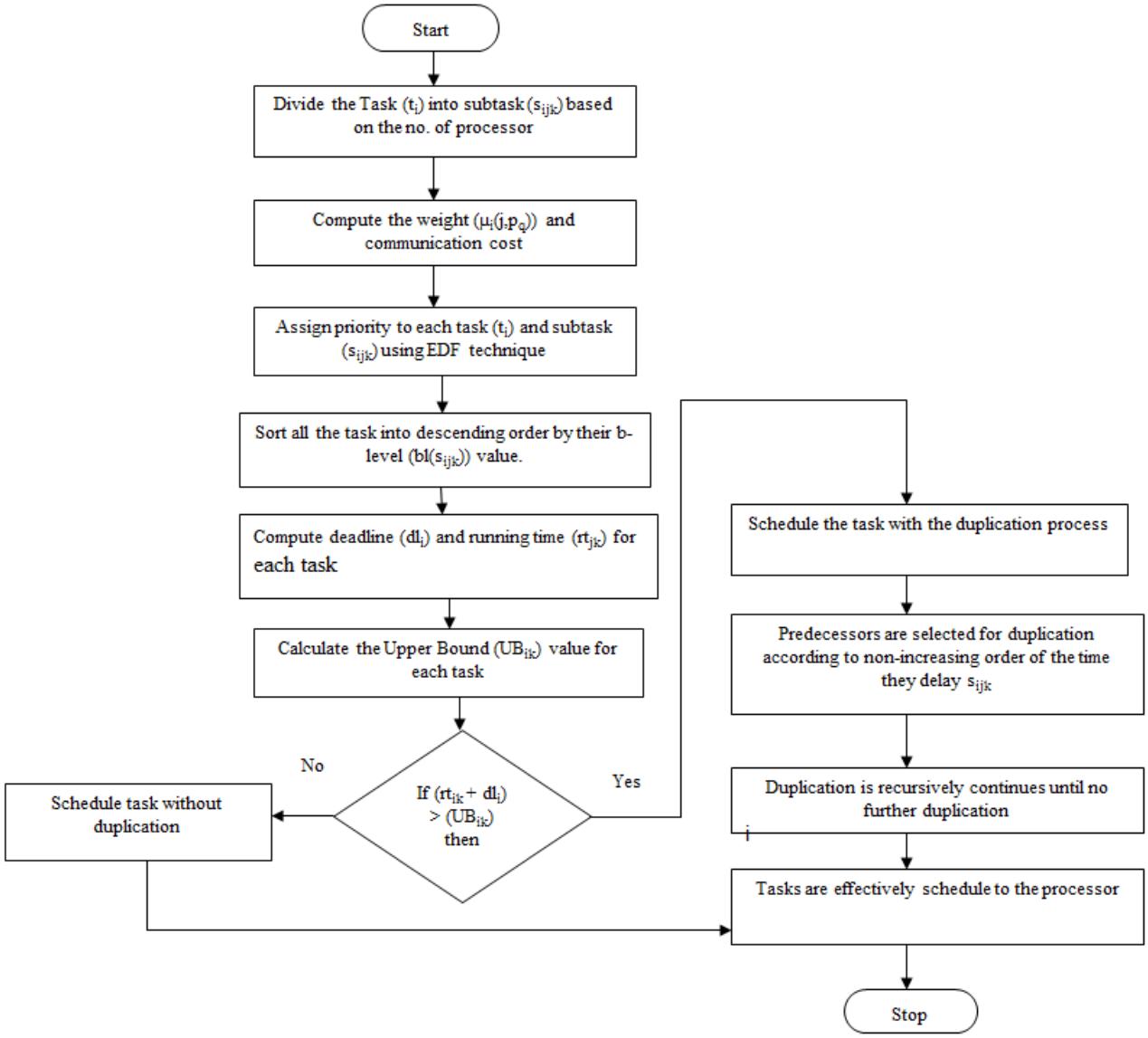

Figure 5: RTCDA Flow Chart

However, this principle does not affect the original behavior of the genetic algorithm. The master stores the full population of chromosomes whereas the slaves contain certain fraction of the chromosome that must be assigned by the master. The fitness function is evaluated in all the parallel slaves and this method reduces the computation time of fitness as in Figure 2. But this technique does not consider the communication delay parameter while scheduling tasks to the processor and also does not utilize the processor time as the processors are idle in certain situations. For better utilization of processors, the tasks are scheduled using a task duplication technique. Further the content-aware model for maximizing the concurrent execution of tasks and minimizing the communication cost by scheduling certain tasks to more than one processor was designed [16].

Furthermore, tasks under the contention model must satisfy some predefined conditions like the local communication which has zero costs, because related tasks are scheduled to the same processor. Where related tasks are scheduled to two different processors and communication is done between the communication subsystems. While comparing it with the existing approaches like contention aware list scheduling (CA-LS) and non-contention aware duplication (DCS) technique this approach speeds up the scheduling processes. This approach reduces the throughput of the scheduling process and the difficulty of modifying the tasks at the scheduling stage. In order to overcome the above mentioned drawbacks, an efficient task scheduling technique called Genetic Algorithm with Node duplication (NGA) was proposed [8]. This technique divides the original task into different subtasks based on the number of available processors. After that, based on the execution time, subtasks are scheduled to the processors using the GA algorithm. The mutation technique applied in the NGA is different from the original genetic algorithm as this technique used the partial-gene

mutation technique to reduce the idle time of the processors. Later on, a controlled duplication technique for scheduling real-time precedence tasks (RTCDA) was proposed [9] taking into consideration the drawbacks of duplication of tasks in NGA.

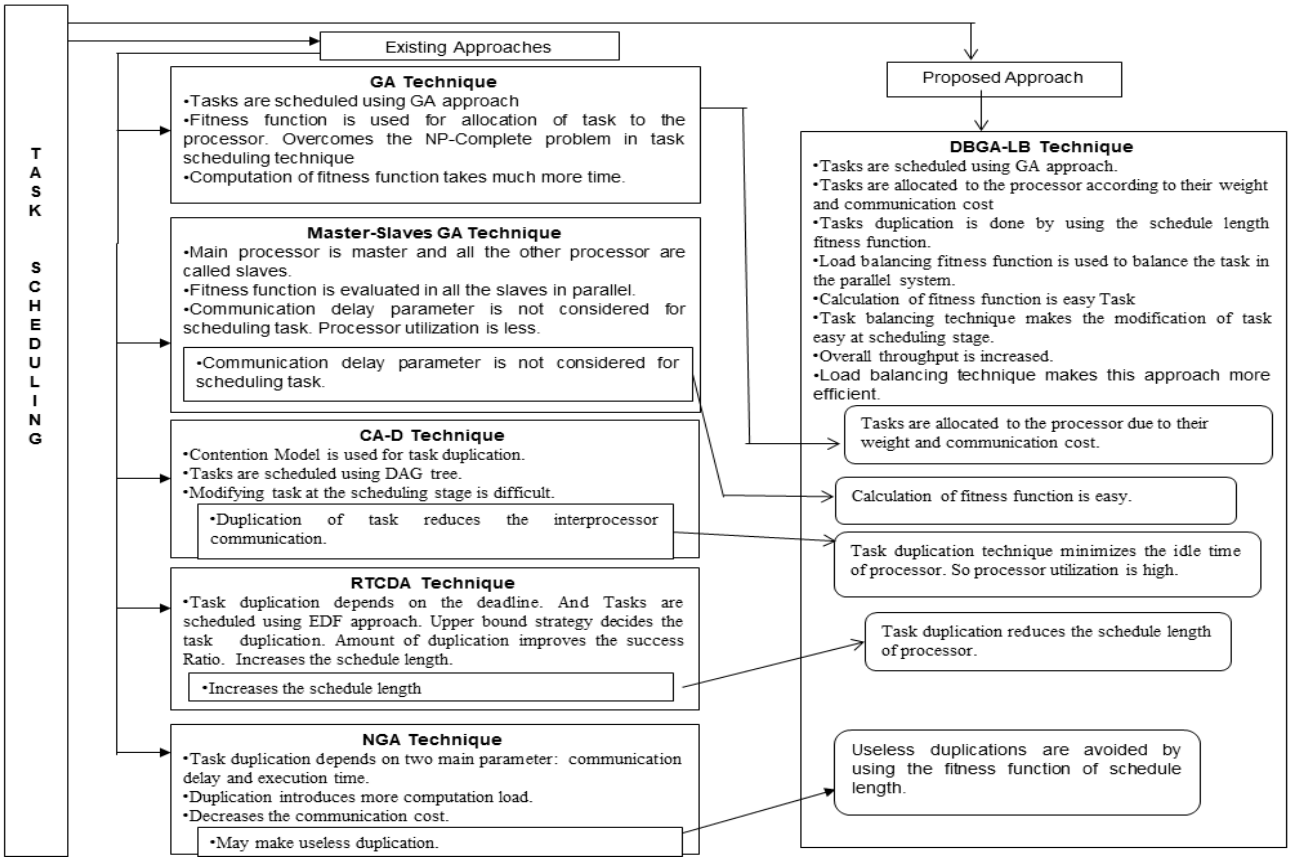

Figure 6. Architecture of the Review between Existing approaches and the proposed.

This technique duplicates tasks based on two parameters such as dynamic upper bound value and absolute deadline of scheduled tasks. If the upper bound value is greater than the absolute deadline, then tasks are duplicated and scheduled using the earliest deadline first technique otherwise tasks are not duplicated. The RTCDA achieves the effective task duplication, but it increases the schedule length. This paper entirely considers the above stated problem and overcomes those problems by proposing the Duplication Based Genetic Algorithm with Load Balance (DBGA-LB). The detail explanation of this approach is given in the next section.

4. Duplication Based Genetic Algorithm with Load Balance (DBGA-LB)

The DBGA-LB algorithm has the characteristics of identifying the critical tasks and redundantly assigning it 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 five major components and is shown in Figure 7. These are:

(i) Production of the input having weight and communication cost of each task t_i randomly to be allocated in different parallel processors p_j.

(ii) Initialization of the chromosome structure up to population size N by applying duplication of some tasks from the tasks list randomly to schedule it to available processors without duplicating it in the same processor.

(iii) The fitness function plays the vital role to select the appropriate chromosomes with the help of an objective function which optimizes the schedule length and load balance.

(iv) Certain number of chromosomes or individuals are selected having best fitness values among the initial population.

(v) Finally, the genetic operations like crossover and mutation are applied to maintain diversity among the chromosomes. These procedures can be carried out until the specified number of generations G is reached [18].

The detailed explanation of each step is given below:

Initial Population: It initializes the population with distinct parameters as schedule length and idle time. To overcome the issues in task allocation, this mechanism performs the duplication of some tasks in the task list randomly. The chromosome structure can be formed by continuing the task duplication to be scheduled in available machines without allocation of replicated tasks in the same processor itself. This process of constructing the chromosomes expands until the population size N.

Computing Fitness Function: The two fitness functions are used for total fitness, (i) Schedule length and (ii) Load balance. The total fitness is computed as:

$$Total\ Fitness\ = {}^{1}/_{(F_1 + F_2)}$$

Where, F_1 denotes the schedule length fitness and F_2 denotes the load balancing fitness. These are computed as:

(i). Schedule Length (F_1): Obtain the chromosomes of best fitness values 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.

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

In the above equation, $E\_Time\ (t_i)$ represents the ending time of latest task running on the processors p_j.

(ii). Load Balance (F_2): It is calculated by the proportion of the schedule length (i.e. maximum execution time) to the average ending time of tasks in all the processors.

$$Task\ Balance\ = Schedule\ Length \Big/ Average\ Execution\ time\ of\ p_j$$

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

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

The load balance fitness can mime the individuals having lesser load balance values to provide the optimized results when compared to the existing methods.

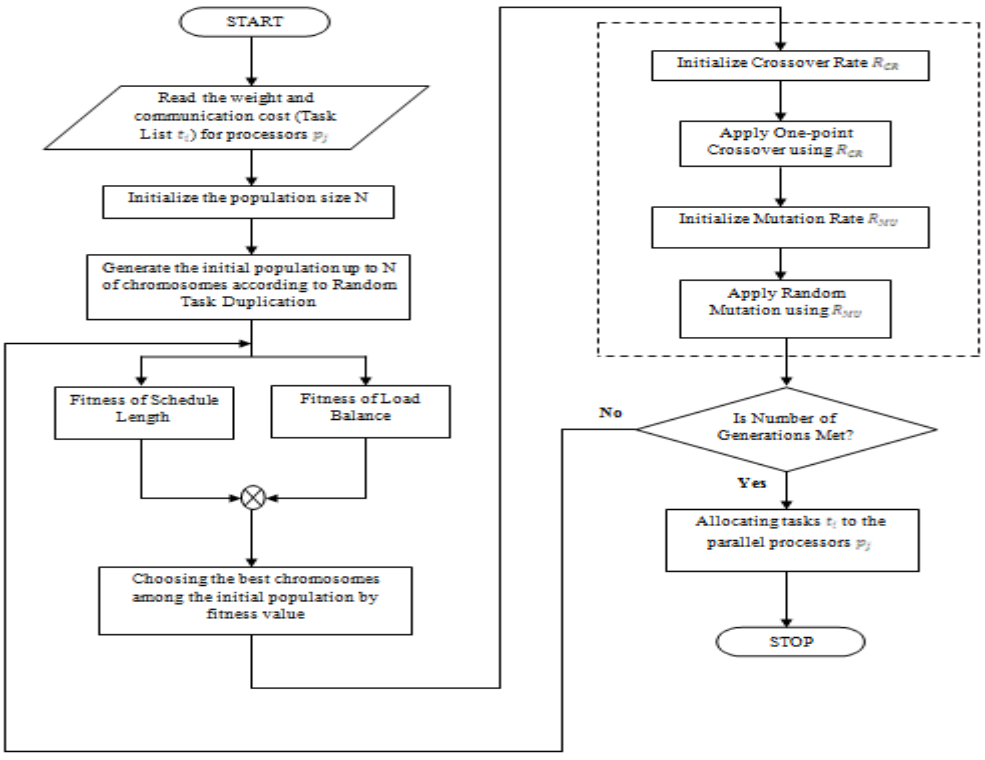

Figure 7: DBGA-LB algorithm Flow Chart

Selection of Chromosomes: The individuals (tasks) are filtered and picked out based on higher fitness values from the original population. Before going into the next iteration of population, the chosen chromosomes undergo genetic operations to produce the offspring chromosomes in the further generations of the population.

Genetic Operators: The schedule length and load balance is the objective function to be concentrated for generating better results than the traditional approaches. These operations are performed in order to maintain the diversity or variations among the individuals. The probability of crossover and mutation (R_{CR},R_{MU}) can be initialized in every generation of population.

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 DBGA-LB algorithm, the probability of crossover R_{CR} can be between 0.5 and 0.9 to receive good results.

Mutation: Mutation is commonly carried out in the genetic algorithm for eliminating the task scheduling problems which occur 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 0.1 and 0.5.

Termination of Task Scheduling: In the 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.

5. PERFORMANCE EVALUATION

This section analyzes the performance of various task scheduling techniques for parallel systems with and without task duplication operations. The comparisons between algorithms are taken according to the fitness function of load balance and schedule length. Table 1 shows the comparison of the DBGA-LB technique with all the algorithms discussed in section 2 on the basis of the following parameters like efficiency, speedup, execution time, processor utilization, schedule length and load balancing. These parameters are implemented for the given scheduling algorithms and their results are shown graphically.

.

Table1: Performance Evaluation Table

Performance Measures	**GATS**	**MSGA**	**CA-D**	**NGA**	**RTCDA**	**DBGA-LB**
Efficiency	Low	High	Low	Medium	Medium	***High***
Execution time	High	Low	Low	Low	High	***Low***
Processor Utilization	Low	Low	Medium	High	High	***High***
Scalability	Low	High	Low	Low	Low	***High***
Schedule length	High	High	High	High	High	***Low***
Speedup	Low	Medium	High	High	Medium	***High***
Task Duplication	No	No	Yes	Yes	Yes	***Yes***

Efficiency: It is defined as the fraction of time that is usefully employed by the processors for a given task [21]. It means how the resources of the parallel system are being utilized. The Figure 8 shows the efficiency of various task scheduling approaches and it also clearly shows that DBGA-LB technique has better efficiency.

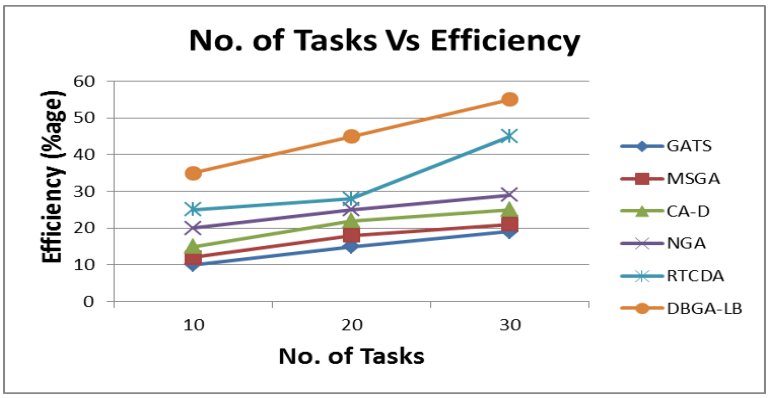

Figure 8: Comparison of DBGA-LB with GATS, MSGA, CA-D, NGA and RTCDA scheduling techniques' efficiency by varying number of tasks

Execution time: It is the amount of time consumed in execution of an algorithm for a given input on the parallel computer. For a better algorithm, execution time must be small.

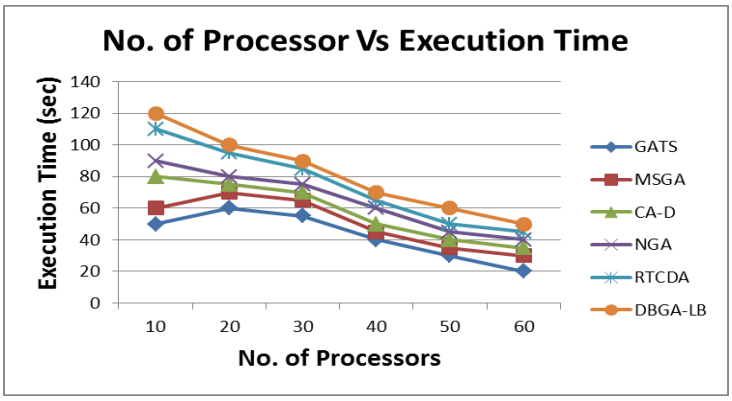

Figure 9: Comparison of DBGA-LB with GATS, MSGA, CA-D, NGA and RTCDA scheduling techniques' Execution Time by varying number of processors

As per the above Figure 9, the DBGA-LB algorithm has a lower execution time, so it has a better performance than the other existing approaches.

Processor Utilization: It is another important factor to measure the performance of an algorithm. A maximum number of processors must be utilized in case of a high performance system. Figure 10 clearly shows that the DBGA-LB technique has a high processor utilization than the other existing approaches such as RTCDA and GA.

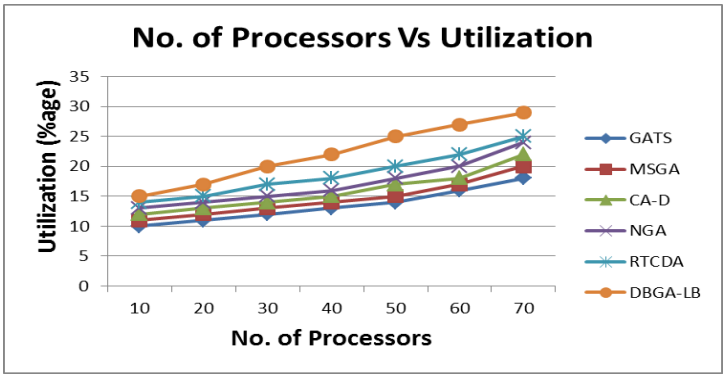

Figure 10: Comparison of DBGA-LB with GATS, MSGA, CA-D, NGA, and RTCDA scheduling techniques' Processor utilization by varying number of processors

Schedule Length: It denotes the maximum ending time of tasks on the available processors [19]. For effective task scheduling, the technique must reduce the schedule length parameter. In Figure 11 the scheduled length of DBGA-LB technique is evaluated by varying the number of duplicated tasks the result with FCFS and DBGA-LB without Duplication method is also compared. It shows that the DBGA-LB technique reduces the scheduled length. It is measured in time unit second.

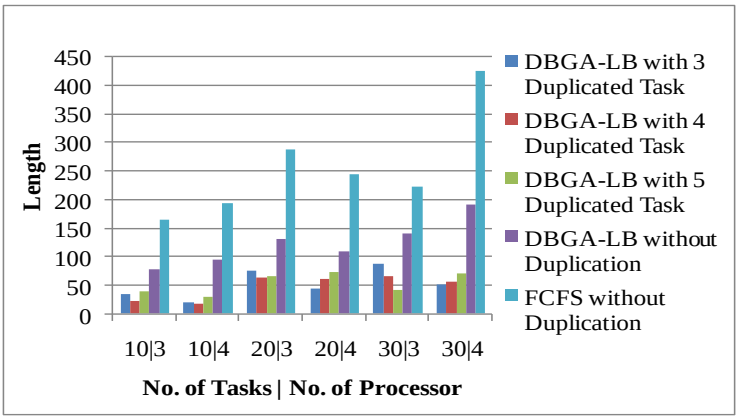

Figure 11: Comparison of DBGA-LB with duplication, DBGA-LB without duplication and FCFS without duplication with Schedule Length parameter

Speedup: It is defined as the ratio of the execution time in the single processor to that of the multiprocessor. The purpose of a parallel computer is to speed up the computer's processing capability, so for a better algorithm the speedup value must be high. The Figure 12 shows the speedup parameter of the task scheduling technique.

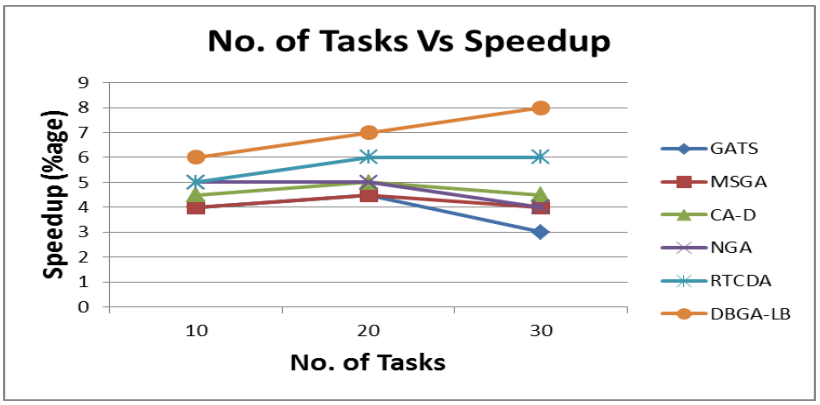

Figure 12: Comparison of DBGA-LB with GATS, MSGA, CA-D, NGA and RTCDA scheduling techniques Speedup by varying the number of Tasks.

Load Balancing: To balance the load between the processor, a load balance technique is used in the parallel system. Several approaches produce optimized schedule length, but the task balance between heterogeneous processors might not be satisfied in some of them. To overcome these effects, the DBGA-LB algorithm can take load balance modifications to obtain the least schedule length and the balance of load is also satisfied. Figure 13 demonstrates the load balancing value of the DBGA-LB technique by varying number of tasks and number of processors. And also it compares the techniques such as FCFS and DBGA-LB without the duplication technique in order to prove that the DBGA-LB technique achieves the balance load condition in scheduling the task.

.

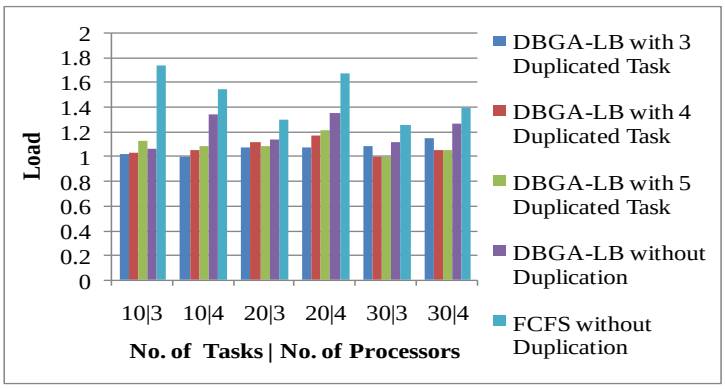

Figure 13: Comparison of DBGA-LB with duplication and DBGA-LB without duplication and FCFS without duplication with Load Balance parameter

The Figure 14 shows the overall comparison of the DBGA-LB algorithm with other GA algorithms with and without task duplication. It also demonstrates that task duplication leads to the good balanced task in all processors, gearing up of execution of speedup, proficient utilization of parallel processors; efficiency of the proposed algorithm is greatly emerged and the schedule lengths of the tasks are ultimately reduced to 10%.

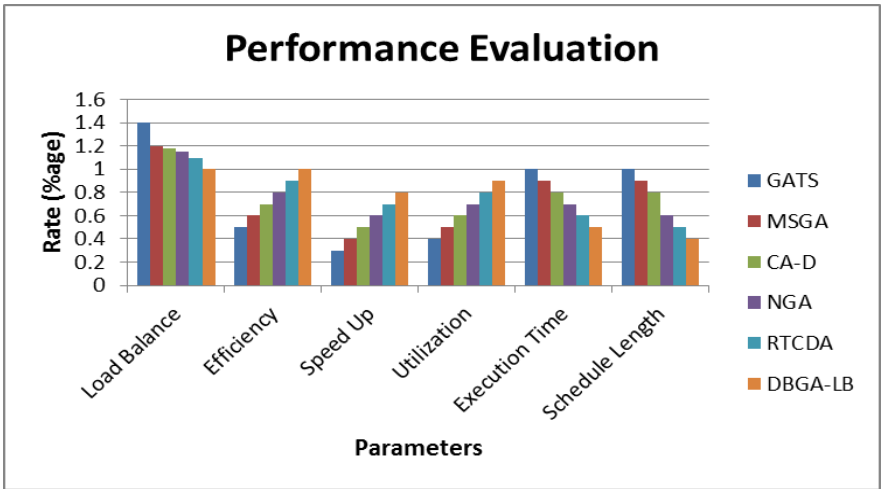

Figure 14: Performance comparison of the DBGA-LB technique with GATS, MSGA, CA-D, NGA and RTCDA

6. CONCLUSION

The scheduling algorithms employed in the parallel systems aim to reduce the execution time of jobs. In the analysis carried out in this paper, it is concluded that task scheduling using duplication techniques produce the effective scheduling results in the parallel systems. Furthermore, the DBGA-LB algorithm proves to be better as its overall execution time is less, and speedup is better than the other existing algorithms.

7. REFERENCES

[1] Ishfaq Ahmad and Yu-Kwang Kwok, “A New Approach to Scheduling Parallel Programs Using Task Duplication”, 1994
[2] Avi Ziv and Jehoshua Bruck, “Performance Optimization of Checkpointing Schemes With Task Duplication”, 1995.
[3] Koichi Asakura, Bing Shao and Toyohide Watanabe, “A Task Duplication Based Scheduling Algorithm for Avoiding Useless Duplication”, 1998
[4] Chan-IK Park and Tae-Young Choe, “An Optimal Scheduling Algorithm Based on Task Duplication”, April 2002.
[5] Doruk Bozdag, Fusun Ozguner, Eylem Ekici, Umit Catalyurek, “A Task Duplication Based Scheduling Algorithm Using Partial Schedules”, 2005.
[6] Doruk Bozdag, Umit Catalyurek and Fusun Ozguner, “A Task Duplication Based Bottom-Up Scheduling Algorithm for Heterogeneous Environments”, 2006.
[7] Wei-Minglin and Qiuyan Gu, “An Efficient Clustering-Based Task Scheduling Algorithm for Parallel Programs with Task Duplication”, 2007
[8] Jaspal Singh and Harsharanpal Singh, “Efficient Tasks scheduling for heterogeneous multiprocessor using Genetic algorithm with Node duplication”, 2011
[9] Jagpreet Singh and Nitin Auluck, “Controlled duplication for scheduling real-time precedence tasks on heterogeneous multiprocessors”, 2011
[10] Oliver Sinnen, Andrea To and Manpreet Kaur, “Contention-Aware Scheduling with Task Duplication”, 2011
[11] 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.
[12] Sharma, Manik, et al. "Stochastic Analysis of DSS Queries for a Distributed Database Design." International Journal of Computer Applications 83.5 (2013).
[13] Sharma, Manik. "Role and Working of Genetic Algorithm in Computer Science." International Journal of Computer Applications and Information Technology (IJCAIT) 2.1 (2013).
[14] Nourah Al-Angari and Abdullatif Alabdullatif, “Multiprocessor Scheduling Using Parallel Genetic Algorithm”, 2012
[15] Ranjit Rajak, “A Novel Approach for Task Scheduling in Multiprocessor System”, 2012.
[16] Samantha Ranaweera and Dharma P. Agrawal, “A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems”, 2000
[17] Probir Roy, Md. Mejbah Ul Alam and Nishita Das, “Heuristic Based Task Scheduling in Multiprocessor Systems with Genetic Algorithm By Choosing the Eligible Processor”,2012
[18] Chih-Hsueh Yang, PeiZong Lee, and Yeh-Ching Chung, “Improving Static Task Scheduling in Heterogeneous and Homogeneous Computing Systems”, 2012
[19] Lin Huang and Michael J. Oudshoorn, “ATME: A Parallel Programming Environment for Applications with Conditional Task Attributes”, 1998.
[20] Ishfaq Ahmad, Member, and Yu-Kwang Kwok, “On Exploiting Task Duplication in Parallel Program Scheduling”, 1998.
[21] Zhiao Shia, Jack J. Dongarraa, “Scheduling workflow applications on processors with different capabilities”, FGCS, 2006.

[22] Sagar A. Tamhane and Mohan Kumar, "Task Scheduling on Heterogeneous Devices in Parallel Pervasive Systems (P^2S)", Springer-High Performance Computing, 2008.
[23] Rashmi Bajaj, Agrawal D.P, "Improving scheduling of tasks in a heterogeneous environment", IEEE-Parallel and Distributed Computing, 2004.
[24] 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.
[25] 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).
[26] 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.