A Survey Of Genetic Algorithm Based Load Balancing Techniques For Task Graph Scheduling Problem In Distributed Environment

Ravreet Kaur, Gurvinder Singh

Abstract: Evolutionary strategies based on the principle of genetic programming are being used to achieve sub-optimal solutions for various NP-Complete problems. One of the most sought after NP-Complete problem is Task graph scheduling i.e. optimally execute the schedule of tasks on available parallel and distributed environment so as to achieve efficient utilization of available resources. Task scheduling is defined as a multi-objective combinatorial optimization problem, with the aim to achieve reduced completion time and effective load balance on the available resources. Various algorithms have been proposed by various authors to achieve the above mentioned goal with the help of various heuristics based on genetic algorithms. All the proposed algorithms by different researchers have been individually reported to be efficient in some certain restricted environment parameters with certain limitations; offering very preliminary improvement on the state of art of one single type of environment. Designers face difficulty in choosing the optimal algorithm for the generalized environment. This paper will introduce basic characteristics of generalized distributed environment along with a survey of all the proposed algorithms that perform load balancing using Genetic Algorithm. Hence, an extensive study has been done with the aim to identify the gaps in existing literature.

Index Terms: Distributed Environment, Genetic Algorithm, Heuristics, Load Balancing, NP-Complete Problem, Task Graph Scheduling, Sub-Optimal Solution.

1 INTRODUCTION

Presently, large – scale systems and high – performance computing is the desire of the community working for applications, directed towards realization of solutions which can be termed as intelligent systems [1,2,3]. Distributed systems are providing the framework for fulfilment of above stated goals. In distributed systems, crucial characteristic is absence of an absolute global control unit. The participating units can be heterogeneous in nature. The decentralized architecture, helps in effective distribution and sharing of resources. This further calls for designing proficient policies that decide on the question that how often various units agree on to share resources amongst each other. But, as these systems are open, which might lead to unreliable structures. With this comes up the need to follow certain fault – tolerant and reliable solutions. Also, another limitation of these systems is network structure under consideration. It impacts the systems performance. The network structure greatly influences the communication between neighboring as well as non – neighboring units. NP-Complete problems have always been of great interest to researchers and scientists because of its challenging nature [6]. Classifying the same in a particular category and then proposing various approaches to solve the same using various heuristics have interested a lot. Task scheduling is a very well known and most sought after NP-Complete problem. The important objective to be achieved is maximum and efficient utilization of various available resources. Various applications like web services and scientific applications are implemented in distributed environment.

Genetic Algorithms (GA) have been recognized to be an efficient heuristic for achieving high quality solutions for various combinatorial optimization problems. One such problem being solved using GA is task scheduling [13, 15, 16, 20, 21, 22, 24, 27, 33, 34, 42, 52]. Heuristic based solutions provide sub – optimal results for Task Graph Scheduling. Various solutions exist for multiprocessor task scheduling like Highest Level First Execution Time (HLFET) [7], Modified Critical Path (MCP) [8], Duplication Scheduling Heuristic (DSH) [9], Linear Clustering (LC) [10] and Dynamic Critical Path (DCP) [11] are a few to mention here, which belong to different classes based on the heuristics used as shown in Table 1 below. The algorithms mentioned above work well only for a particular environment and not generalized environment. Hence, arises the need for a better solution that not only works in any given situation but also adapts itself to the changes in state as and when required. GAs stand out amongst various other heuristics because of there inherent parallelism which can be further exploited to further maximize required objectives. Therefore, many studies have endeavored to achieve the above mentioned objectives in distributed environment [13, 15, 16, 20, 21, 22, 24, 27, 33, 34, 42, 52].

- *Ravreet kaur is currently pursuing post-doctoral degree program in Deptt. Computer Science & Engineering in Guru Nanak Dev University, India, E-mail: ravreet@yahoo.com*
- *Gurvinder Singh is currently working as Professor in Deptt. of Computer Science, Guru Nanak Dev University, India. E-mail: gsbawa71@yahoo.com*

Table 1: Scheduling algorithms based on various heuristics for homogeneous environment

S.No.	Algorithm	Environment	Number of Processors	Heuristic
1.	HLFET	Homogeneous	Bounded	List Scheduling based
2.	MCP	Homogeneous	Bounded	List Scheduling based
3.	DSH	Homogeneous	Unbounded	Task Duplication based
4.	LC	Homogeneous	Unbounded	Critical Path based
5.	DCP	Homogeneous	Unbounded	Critical Path based

The goal of this survey is to provide a timely remark on the current research aimed at improving and enhancing the already achieved objectives of task graph scheduling algorithm studies performing load balance using GAs as a meta-heuristic. This paper provides a basic framework for proposed methodology to be used in future.

2 TASK GRAPH SCHEDULING PROBLEM

Broadly all problems can be classified into two categories: Data intensive and Computation intensive. Scheduling of a parallel program lies in the class of computation intensive problem. Generally, scheduling is categorized to be either static or dynamic. Static scheduling problem solves the tasks scheduling of those tasks whose processing times, communication costs, data dependencies and synchronization requirement are well known before performing scheduling. Whereas, Dynamic scheduling problem solves the tasks scheduling of those tasks whose processing times, communication costs, data dependencies and synchronization requirement may not be well known before performing scheduling. Further, Task scheduling algorithms may be broadly divided in two main classes: Greedy and Non-greedy (iterative) algorithms [6]. Greedy algorithms solve the problem and achieve local optimal solutions assuming the same to be global optimal solution. But, Non-Greedy algorithms improve the local optimal solution so as to achieve global optimal solution. A partial taxonomy of task graph scheduling algorithms has been reviewed and discussed in detail by Kwok and Ahmad [6].

Task graph scheduling problem is defined as problem of scheduling a weighted directed acyclic graph (DAG), onto a set processors as available so as to minimize the completion time. Various other parameters are also to be considered so as to measure the performance of available environment using the algorithms as defined by Kwok and Ahmad [6]. Task graph is considered to be a DAG. The DAG is a generic model that represents any parallel program by mapping a set of tasks in an acyclic graph and also represents dependencies among the tasks.

Definition 1 (DAG model) [6, 58]

Given a DAG, G = <V;E>, where V is the set of nodes and E is set of directed edges. Each node in DAG i.e. n_i represents a task which in turn is a set of instructions that must be executed sequentially without preemption in the same processor. The computation cost of task is the weight of node denoted by $w(n_i)$. The nodes are connected via edges i.e. $\forall<n_i ; n_j> \varepsilon E$ indicates the communication cost of edge denoted by $c(n_i , n_j)$, if task n_i and n_j are executed on different processors. Communication – to – Computation - Ratio(CCR) of a parallel program is:

$$\textbf{CCR} = E \Big/ \sum_{V=1} w(ni)$$

A node with no parent is called an entry node and a node with no child is called an exit node. Hence, scheduling algorithm aims to achieve minimum completion time for the given DAG by satisfying all the constraints represented in the same. An example of a DAG is represented in Figure 1.

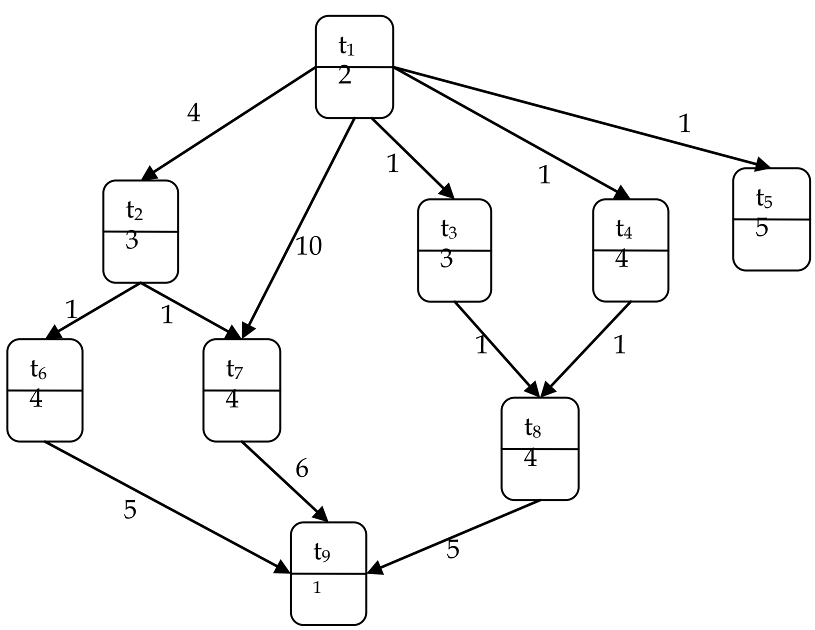

Figure 1: Directed Acyclic Graph [6]

Definition 2 (Task Scheduling in distributed systems) [58]
Given a distributed system, S = <N; E>, where N is the set of nodes and each node owns a different set of resources, and ∀ $<n_i; n_j>$ ε E indicates the existence of a network link or communication path between node n_i and n_j. The set of resources in node n_i is assumed to be R_{ni}, and the set of resources required by task t is assumed to be R_t. If task t arrives at the network, the task scheduling in S can be defined as the mapping of task t to a set of nodes, N_t, which can satisfy the following situations:

1) The resource requirements of t can be satisfied.
2) The predefined objective can be achieved by the task execution of N_t, such as minimizing the execution time or any other objective under consideration.
3) The nodes in N_t can execute the allocated tasks under the constraint of the network structure.

2.1. Issues in Task Graph Scheduling Problem
Task graph scheduling in distributed systems is applied on the available resources in the network. The resources are allocated to a unit based on some decision parameters, giving rise to different decisions for different set of problems. But, various issues arise with the use of distributed systems. The task at hand, might require more than one unit to negotiate with each other for sharing of resources. Also, there will be a need to implement task – friendly, resource – access policies. Unreliability of network can hugely impact the desired result of task. There can be a rise of conflict between tasks, while sharing critical resources. Lastly, network structure can influence the performance of tasks significantly. Hence, there is a need to keep note of various issues like

1. Load Balance
2. Resource Optimization
3. Reliability
4. Relationship between processing units
5. Network structures under consideration.

This paper will focus on the critical issue of load balancing being satisfied by the existing solutions under study. Also, the issues identified above suggest that with the existence of dynamic parameters of systems, solutions proposed for the problem of task graph scheduling should be adaptive i.e. system adapts itself to perform task graph scheduling with the available resources by choosing the appropriate solution algorithm for the same, also focusing towards achievement of a system that is reliable as well as fault - tolerant.

3 LOAD BALANCING

3.1 Introduction
Task graph scheduling is a multi-objective problem. Load Balancing is one of the main objective to be achieved efficiently as well as effectively by not compromising on the other objectives. The problem can be defined in simple words as the task of allocating the resources in a way that they are utilized in a fair manner and the execution times are minimized with reduced communication overhead.

Definition 3 (Load Balancing with Task Graph Scheduling in distributed systems) [58]
Given a distributed systems, S = <N; E>, where N is the set of nodes, $\forall n_i \varepsilon N$, the team of tasks that queue for resource r_k of node n_i can be denoted as Q_{ik}. The size of Q_{ik} is s_{ik} and the processing capability of n_i is v_j; the original probability of node n_i to receive tasks (which need k type resources) is $P_i(k)$, which can be calculated by considering the resources of n_i. Load balancing should be performed when s_{ik} is too large, which is implemented by discounting the probability of n_i's receiving new tasks, $P_i(k)$, to a revised probability, $Dp_i(k)$: $Dp_i(k) = c(s_{ik} = v_i) * P_i(k)$; where c is an attenuation function, 0 <= c <= 1; the value of $c(s_{ik} = v_i)$, decreases monotonically from 1 to 0 as $S_{ik} = V_i$ increases.
Load Balancing problem can be defined in simple words as a sequence of following steps [16, 47]:

1. Load Measurement – Decides which unit is lightly loaded or heavily loaded. Certain threshold value is used to make a unit fall under one of the categories stated above.
2. Information exchange – Resources need to exchange information regarding their loads so that lightly loaded units can be made to help out the heavily loaded units.
3. Load Balancing decision by deciding on new work distribution – should be a global optimal decision that leads to a balanced system.
4. Task migration – system decides when to migrate and where to migrate, keeping the migration costs as low as possible

Load balancing along with task scheduling in distributed systems provokes issues like:

1. No assurance of global optimal results because of decentralized architecture
2. Accessibility of required resources on a unit as well as its neighboring or sharing units
3. No guarantee of reliability due to openness of the system
4. Coordination policies for heterogeneous units
5. Influence of network structures on transmission or exchange of information amongst various units

A very brief and straightforward taxonomy of load balancing algorithms can be represented as shown in Figure 2 below:

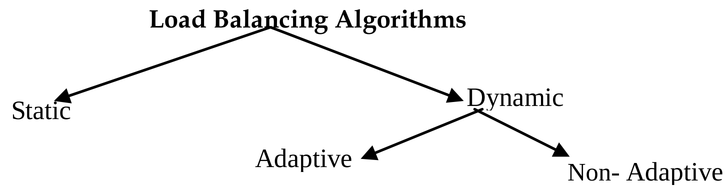

Figure 2: Partial Taxonomy of Load Balancing Algorithms

3.2 Existing Literature under survey

Sr. No.	Research Paper	Objectives	Data Sets	Environment	Metrics	Research Gaps identified	Category
1	Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor Computing [1999][20]	Comparison with previously proposed List Scheduling heuristics based schedulers	DAG with Maximum number of tasks considered = 25	Loosely Coupled Heterogeneous Multiprocessor Systems	Cost and MakeSpan	Generalization of the proposed algorithm and further exploration of the same for solving larger size problems	Load Balancing is not considered
2	Observations on using Genetic Algorithms for dynamic Load Balancing [2001] [24]	Comparison of centralized GA scheduler with First Fit Heuristic	Independent set of fixed size tasks used	Homogeneous Multiprocessor Systems	Total Completion Time and Average Processor Utilization	Results based on assumptions like constant communication cost and dedicated processors, are far away from real – time scenarios	Dynamic load balancing but on static environment
3	Dynamic Task Scheduling using Genetic Algorithm for Heterogeneous Distributed Computing [2005] [27]	Comparison with batch – mode and immediate – mode schedulers	Randomly generated independent set of tasks upto 10,000	Heterogeneous Multiprocessor Systems	Total execution time[MakeSpan] and Efficiency [Processor Utilization]	Testing on a generalized distributed environment is pending	Random approach for load balancing, which may further lead to worse results
4	A Genetic Algorithm for Process Scheduling in distributed operating systems considering load balancing [2007] [30]	Proposed a Genetic Algorithm based solution	Independent set of Tasks considered with the added constraint that they are non - preemptive	Distributed Systems with no proper specifications	Multi-objectives: minimize makespan and communication cost, maximize avg. processor utilization and load balance	Comparisons with the existing literature can be performed based on certain benchmarks for validation of the proposed algorithm	Random approach for load balancing, which may further lead to worse results
5	A Genetic Algorithm based Dynamic Load Balancing Scheme for Heterogeneous Distributed Systems [2008] [33]	Proposed a Genetic Algorithm based solution that is compared only with standard genetic algorithm	Independent set of tasks with known execution times prior considered	Heterogeneous multiprocessor system with two classes of processors considered, with one class having double the speed of other class	Total Completion Time and Average Processor Utilization	All genetic algorithm parameters kept constant which leads to limited performance of genetic algorithm; Various challenges of Heterogeneous environment like reliability, network structures, scalability to be researched upon elaborately	Dynamic load balancing but on static environment

6	A fast hybrid genetic algorithm in heterogeneous computing environment [2009] [37]	Hybrid genetic algorithm based solution compared with standard GA	Used one foxed DAG of height 6 and width 10.	Heterogeneous Multiprocessor System	Total execution time	Comparisons with the existing literature can be performed based on certain benchmarks for validation of the proposed algorithm	Load Balancing is not considered
7	An Efficient Dynamic Load Balancing Scheme for Heterogeneous Processing System [2009] [36]	Comparison with weighted least connection algorithm (WLCA) and Genetic Algorithm	Tested on 3 groups of high-performance computer resources	Heterogeneous Multiprocessor System	Cost of running time and average time of response delay	Network structure constraints not taken under consideration	Partial load balancing performed
8	Multi-Heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system [2010] [42]	Comparison with MM, MX, LLM, LLX, MMC, MXC, LLMC, LLXC, PN	Indivisible, independent, set of tasks	150 non – dedicated heterogeneous multiprocessor system	MakeSpan	Elaborate work required for correct run – time estimations for various tasks.	Load Balancing is not considered
9	A Genetic Algorithm for mapping tasks in Heterogeneous Computing System [2011] [44]	Comparison with Standard GA	Precedence constrained tasks	No proper specification given	MakeSpan	3-step mutation could be further altered to determine optimal probabilities i.e Adaptive mutation probabilities to be tested	Non – adaptive load balancing performed
10	Dynamic Task Scheduling algorithm with load balancing for heterogeneous computing system [2012] [47]	Compared proposed CBHD with HEFT and Triplet Clustering algorithms	Only executed for a pre-defined DAG with 10 nodes and 11 nodes	Heterogeneous Multiprocessor System	MakeSpan, Load Balance, Processor Utilization	Comparison analysis needs to be elaborated	Non – adaptive load balancing performed
11	An Improved Genetic Algorithm for Load Balance in Multiprocessor Systems [2012] [50]	Critical Path based	Randomly generated DAG of 400 tasks which are non – preemptive in nature	Only homogeneous environment	Schedule length and Load Blance	No comparison analysis done for validation of proposed solution	Non – adaptive load balancing performed
12	Characterization of the Iterative Application of makespan heuristics on non-makespan machines in a Heterogeneous Parallel and Distributed Environment [2012] [51]	Comparison of algorithms using an Iterative approach	Set of 4 independent set of tasks	Heterogeneous Multiprocessor System	MakeSpan	Negative: Could not improve makespan of the non-makespan machines	Load Balancing is not considered
13	Dynamic Task Scheduling Modeling in Unstructured Heterogeneous Multiprocessor Systems [2014] [54]	Comparison with static and time-varying environment evolutionary algorithm	Random DAGs	Heterogeneous Multiprocessor System	Computation time, Communication time	Not considered time-varying task arrival and also adaptation to real – time scheduling constraints	Load Balancing is not considered
14	A uniform approach for	Introduced library libwater-	Self programmed	Heterogeneous Multiprocessor	Execution time and SpeedUp	Use and check if it could become a research	Load Balancing is

	programming distributed heterogeneous computing systems [2014] [55]	extension to OpenCL programming model	applications	System		platform to investigate performance aspects of heterogeneous and distributed HPC architectures	not considered
15	A New Genetic Algorithm for Release – Time Aware Divisible – Load Balancing [2014] [53]	Proposed algorithm for global optimization technique to perform release-time aware divisible-load scheduling	Randomly generated tasks	Parallel and Distributed system	Makespan	Very limited approach for comparison of proposed Genetic Algorithm with the Standard Genetic Algorithm; no proper specification of environment given	Static Load Balancing
16	Load Balancing in Fault-Tolerant Real –Time Systems for Periodic Task Scheduling [2015] [56]	Improved existing R-Batch algorithm for fault-tolerant systems	Independent set of tasks	Homogeneous Multiprocessor System	Load Mean Square error	Analysis can be further extended to real time environment	Static Load Balancing
17	Load Balancing in Distributed Systems: A survey [2015] [57]	Brief survey of load balancing techniques along with the theoretical use of Game Theory	-NA-	P2P Cluster Heterogeneous environment	Coupling, Scheduling, Physical Location, Scalability	Single-objective approach has been followed, rejecting the various trade-offs that come up with this approach	-NA-

4 GENETIC ALGORITHM WITH THE CAPABILITY OF LOAD BALANCING FOR TASK GRAPH SCHEDULING

Nature has been an inspiration for research since ages. Presently human race is working towards incorporation of natural life principles for various scientific problems. Most of the work is inspired to inculcate natural human evolution process to achieve the maximum success rate possible. Genetic algorithms (GAs) are an invention of this inspiration. GA works by using the human evolution process of reproduction for searching the best possible solution of the given choices. It is a searching technique which works towards achieving sub - optimal solution by exploring and exploiting the complete solution space. Basic GA works by representing the given problem in the form of some strings, which can be used as an initial solution space. Further the solution space is explored using selection criteria based on well defined fitness function to choose two best possible solutions from the initial defined solution space. Lastly the chosen solutions are exploited using some crossover and mutation methods to generate an offspring that is fitter than the chosen best solutions. Thus, algorithm works towards achieving optimal solution using the available sub-optimal solutions. A control abstract for a Basic GA is defined as follows in Figure 3:

```
Algorithm BasicGA()
//Define a problem representation based on the problem parameters at hand.
{
    Initialize_Population();    //Create some initial solution space
    Fitness_function();         // Used to choose best out of possible solutions
    While (X=0 and X<= number of generations; X++;)
    {

      Parent_String1();         //Chosen based on defined fitness function and selection criteria
      Parent_String2();         //Chosen based on defined fitness function and selection criteria
      NewString();              //Generates Offspring using available exploitative functions
      {
        Crossover_NewString(); // Parent Strings used to generate best possible Offspring
        Mutation_NewString();  //Offspring manipulated to achieve more efficient solution
      }
    }
    Sub_Optimal_Solution();    //Returns last generated NewString
}
```

Figure 3: Control Abstract for a Basic Genetic Algorithm

4.1 Problem Representation

The initial planning stage for the designing of Genetic Algorithm consists of forming a representation for the problem to be solved considering all of its essential aspects. Focused approach should be used so that it provides power to GA in achieving the results it aims for. An efficient problem representation can be achieved by considering the problem, its constraints and objectives in an effective combination. The proposed methodology will generate problem representation based on the input parameters of the problem i.e. set of tasks and processors. The representation is done by using linear data structures i.e. an array or linked list.

4.2 Initial Population

The generation of initial Population is possible in various ways. It is chosen based on problem space and problem representation. Most important aspects to be fulfilled by this parameter are recognized to be as follows:

1. Have the ability to represent all possible mappings
2. Application of genetic operators on all generated mappings should be possible

Most commonly used method is to generate all random chromosomes. Another method is to use some existing heuristics to generate the same. One should pay attention to the extra time consumed by the heuristic for the generation of initial population. Therefore, to reduce time a small percentage of population should be generated initially and then further the whole problem space can be explored with the help of other parameters like crossover and mutation. Further some methods should be used to validate the size being considered. Its initial population size should be large enough to cover all important aspects of problem under consideration as well as small enough to save on time parameter. Also, there is no general rule that applies to selection of initial population size as no known size exists, which itself proves that size doesn't affect solution. Hence, the conclusion drawn from previous observations with reference to initial population size is that optimal solution found through testing shouldn't be too small or too large. The proposed methodology would use all the the heuristics mentioned in table 1 to generate an initial population within GA scheduler. The hypothesis is applied here that using more heuristics will result in an improved initial population, however this implies the need for further research but that is beyond the scope of this paper. The complete population is generated by applying random permutation of these heuristics. Employing heuristics for generating initial population as compared to a randomly generated initial population, provides a certain meaningful and reasonable initial population.

4.3 Fitness Function

The fitness function of a genetic algorithm measures the fitness of a chromosome based on the metrics of problem at hand. The literature studied till date point out that for scheduling based problems the basic fitness criterion considered is MakeSpan, which is defined as the total execution time of a schedule. Task graph scheduling falls under the class of multi objective problem. Hence, there is a drawback of considering only single objective as fitness criterion, one might end up ignoring other important issues like load balancing for the optimal usage of all available resources. Therefore, fitness function should be a combination of all metrics. The strength of fitness function would be affected by problem representation used. It can vary from a very simple function to a complex function. The proposed methodology would use fitness function based on the objective of our problem i.e. reduced total execution time along with a well balanced load distribution for efficient utilization of all the resources. Hence, fitness function should be a combination of both the parameters, as follows:

Fitness Function = TETF() X LBF()

where,

TEFT() is the total execution time factor defined based on MakeSpan

LBF() is the Load Balance Factor based on utilization of all available processors

4.4 Selection

Problem space under exploration is very large, with the property that parent and offspring both have equal chance of competing for survival. Winner of competition is the one with highest fitness value i.e. the one with highest probability than others. The existing selection methods are based on rank criteria as it eliminates the scaling problem of direct fitness based approaches. Most commonly used existing selection methods are:

1. Pareto ranking system: selects the fittest chromosome out of all the existing chromosomes in population, by assigning them rank based on fitness function value, with the condition that the highest rank is assigned to chromosome having best fitness value.
2. Roulette wheel selection: selects chromosome randomly but surety of selection of fittest chromosome is made by assigning it more probability of selection as compared to others.

Mostly pareto ranking system is used as it has been proved to be faster in convergence as compared to roulette wheel selection.

4.5 Crossover

Crossover operator is used for the purpose of exploration of the problem space. This technique takes two or more parent chromosomes and applies existing crossover techniques to produce new child chromosomes. Some or all existing solutions are made to perform crossover to achieve new possible solutions. Crossover probability parameter takes care of that how often crossover is to be performed. One can say if crossover probability is 0, then new generation is exact copy of previous generation and if crossover probability is 1, then new generation is made by performing crossover of all off springs in previous generations. There are various techniques being used to perform crossover like One point, Two point, Three point, Partially Matched CrossOver, Ordered CrossOver, Linear Ordered CrossOver, Cycle Crossover, Uniform, Half uniform or Cut & Splice. The problem, mostly encountered is that solutions obtained after performing crossover might be infeasible so certain measures are to be taken so that only feasible solutions considered. But, this might result in loss of some potential solutions which can be achieved through crossover of infeasible solutions. Some crossover techniques produce one child or two children; no logical explanation exists for the same. Also, metrics under consideration are not observed while performing crossover. Hence, proposed crossover operator should take schedules in account with respect to task scheduling problem. The proposed methodology would use cycle crossover method as it helps in eliminating cycles i.e. infeasible schedules. This method validates the child strings generated. The method works on two randomly chosen parent chromosomes selected from the population. Let them be parent1 and parent2. A certain index is chosen randomly in both the parents and is marked as visited. Then the value of parent2 index is noted and parent1 is traversed to search for the same value. When this value is found, for that particular index values in parent1 and parent2 that index value is marked as visited. This process continues unless a certain value in parent1 is visited twice. This step identifies the cycle. Hence, now the visited values are crossed over to produce new child springs. Further the probability of crossover will be decided by performing experiments over the range from 0 to 1 and best will be chosen accordingly.

4.6 Mutation

Mutation operator like crossover operator, is used for the purpose of exploration of the problem space. It attempts to find new points in search space so that population diversity can be maintained. This technique takes one chromosome and applies existing mutation techniques to produce new child chromosome. Mutation probability parameter takes care of that how often mutation is to be performed, based on similar principle as applicable to crossover operator. There are various techniques being used to perform mutation like Flip – bit, Boundary, Uniform, Non – uniform or Gaussian. Mutation techniques hugely contributes towards widening of search space. But, probability of mutation factor is usually kept small enough to avoid generation of completely infeasible solution sets. Similar to crossover, in case of mutation also, a note should be kept of various metrics of problem under consideration. The proposed methodology would use random mutation to utilize the exploitative feature of GAs. Also, along with mutation operation a rebalancing heuristic will be applied as used by many researchers [42]. Rebalancing heuristic will perform the job of a load balancer. Further the probability of mutation will be decided by performing experiments over the range from 0 to 1 and best will be chosen accordingly.

4.7 Other Operators

It needs to be pointed out that other than crossover and mutation, few other operators have also evolved into genetic algorithms for more effective exploration and exploitation of problem space. Elitism operator states that best solutions be considered without any change for next generation. This may give rise to a faster convergence, but it needs to make sure that solution is globally optimal. Certain solutions work by generating groups in population, working individually. Literature studies also state that there might be a need of repair algorithm after crossover and mutation. All the additional mechanisms point towards the problem of achieving local optimal solution. Hence, all methods should be used while keeping note of that problem space is explored well before stopping for optimal solution.

4.8 Ending Criterion

Lastly, the end criterion in genetic algorithm should be chosen so as to achieve globally optimal result in limited number of generations with the constraint of limited run time. The consecutive generations should be constantly overviewed to check if fitness parameter stays constant. If this property is met, then also execution should be halted. The criteria is considered to avoid infinite running of genetic algorithm. The proposed methodology would constantly overview consecutive generations to check for positive growth of fitness parameter. If it starts decreasing, then execution would be stalled.

4.9 Proposed Methodology Framework: Flowchart

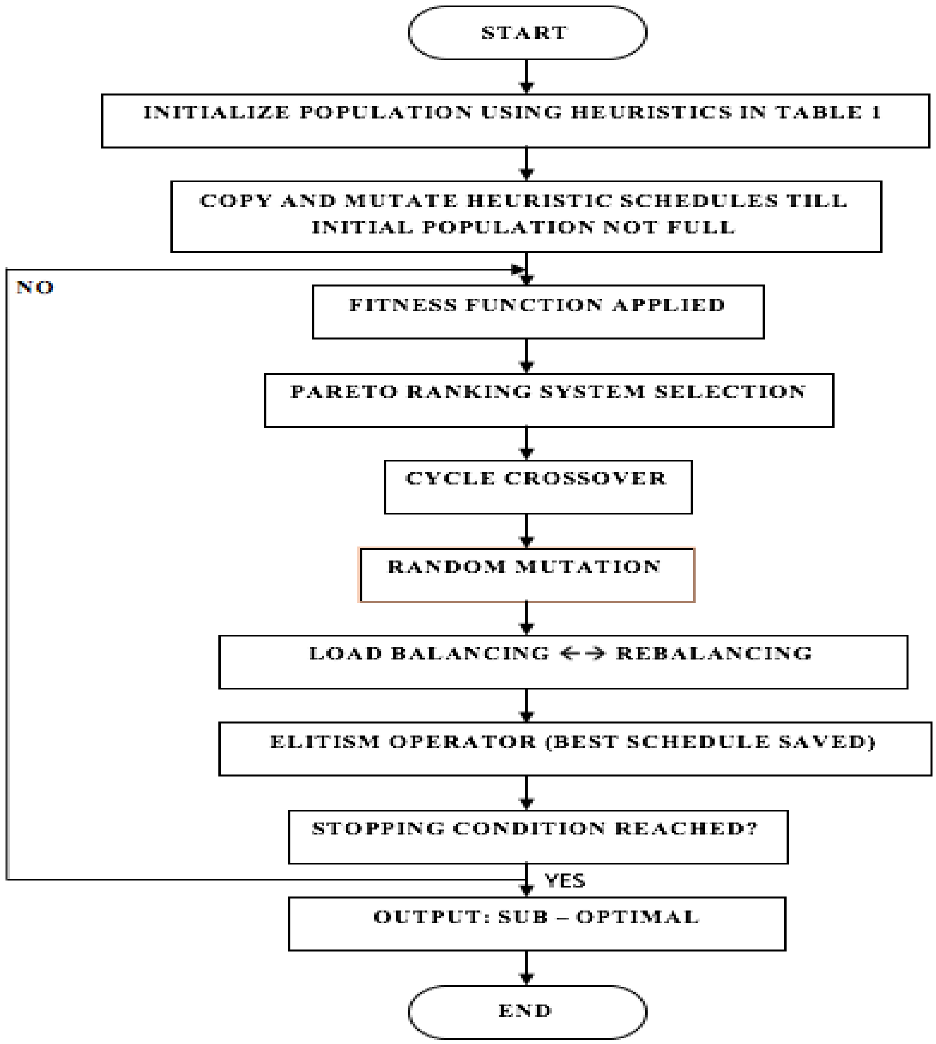

Figure 4: Flow of tasks to be performed by the heuristic (Genetic Algorithm) under consideration

5 CONCLUSION

Distributed systems contribute towards accomplishment of large – scale systems, where various units exist without the presence of any global control unit. These units can operate autonomously as well as cooperate with each other for sharing of resources if desired by the tasks executing on the systems. The systems can be made reliable by introducing some trust/reputation mechanisms. Heterogeneous nodes exist in decentralized architecture. This architecture maps the dynamic situation closely. The network structure should be a combination of physical as well as virtual interaction characteristics. Further, task scheduling along with load balancing for distributed systems should achieve global optimal results using genetic algorithm. Various issues with respect to load balancing still needs to be explored by devising some hybrid approach. On the basis of this survey, it is observed that a hybrid solution using genetic algorithm may achieve better results.

6 FUTURE OBJECTIVES

Some challenges in existing studies on the issue of task allocation along with load balance on distributed systems using genetic algorithm heuristics may be:

1. Reliable and fault – tolerant distributed systems: by introducing some policies that can use previous behavior of units to include then under trusting or non – trusting units
2. Environment is yet to be standardized for conducting such studies.
3. Network structure favoring the system performance: communication between various units should not be constrained by the under laying network structure
4. Adaptive Task scheduler that performs load balancing as well: by adapting to the changes in the information of

various units and make decisions by including the current knowledge as well.

5. Measure processor load efficiently: policy is to be designed that could apply uniformly on the complete environment.
6. Task migration decision as well as migration cost: Decision making is done on the information collected in previous point, more complete the information and more efficient the decision will be.
7. Genetic parameters should be chosen by considering the task as well as system constraints: so that the problem space is explored and exploited well. Proper selection of genetic parameters is still an open issue.

This paper has provided a brief survey of the task graph scheduling solutions using genetic algorithm that perform load balancing in distributed systems. The drawbacks identified in the studies undertaken for survey, point towards the need for a solution that will work towards removal of the highlighted drawbacks. It is proposed that a prototype will be implemented keeping in the mind the observations gained through the survey. The prototype will help in assessing the impact of including the above stated properties on the simulation development; by performing its comparison to the existing solutions.

7 References

[1] D.G. Feitelson and L. Rudolph, "Parallel Job Scheduling: Issues and Approaches", Proceedings of Job Scheduling Strategies for Parallel Processing, JSSPP, Springer, Berlin, Heidelberg, 1995.

[2] S. J. Chapin, W. Cirne, D.G. Feitelson, J. Patton, J. Scott, T. Leutenegger, U. Schwiegelshohn, W. Smith and D. Talby, "Benchmarks and Standards for the Evaluation of Parallel Job Schedulers", Proceedings of Job Scheduling Strategies for Parallel Processing, JSSPP, Springer, Berlin, Heidelberg, 1999.

[3] D.G. Feitelson and L. Rudolph, "Metrics and benchmarking for parallel job scheduling", Proceedings of Job Scheduling Strategies for Parallel Processing, JSSPP, Springer, Berlin, Heidelberg, 1998.

[4] D.G. Feitelson and L. Rudolph, Schwiegelshohn U. "Parallel Job Scheduling — A Status Report", Proceedings of Job Scheduling Strategies for Parallel Processing, JSSPP, Springer, Berlin, Heidelberg, 2004.

[5] D.G. Feitelson and A. Mu'alem, "On the definition of "on-line" in job scheduling problems", Proceedings of Journal SIGACT News, volume no. 36, 2005, pp. 122-131.

[6] Y.K. Kwok and I. Ahmad, "Benchmarking the Task Graph Scheduling Algorithms", Proceedings of First Merged International Parallel Pocessing Symposium, Parallel and Distributed Processing Conference, pp. 531-537, 1998.

[7] T.L. Adam, K. Chandy and J. Dickson, "A Comparison of List Scheduling for Parallel Processing Systems," Proceedings of Communications of the ACM, Volume 17, Issue no. 12, Dec. 1974, pp.685-690.

[8] M. Wu and D. D. Gajski, "Hypertool: A programming aid for message-passing systems," Proceedings of IEEE Transactions Parallel and Distributed Systems, volume 1, July 1990, pp. 330–343.

[9] B. Kruatrachue and T.G. Lewis, "Duplication Scheduling Heuristics (DSH): A New Precedence Task Scheduler for Parallel Processor Systems,", Proceedings of Technical Report, Oregon State University, Corvallis, OR 97331, 1987.

[10] S.J. Kim and J.C. Browne, "A General Approach to Mapping of Parallel Computation upon Multiprocessor Architectures", Proceedings of Internationa1 Conference on Parallel Processing, volume 11,1988, pp. 1-8.

[11] Y.K. Kwok and I. Ahmad, "Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors", Proceedings of IEEE Transactions Parallel and Distributed Systems, vol. 7, no. 5, May 1996, pp. 506-521.

[12] I. Ahmad and Y.K. Kwak, "A New Approach to Scheduling Parallel Programs using Task Duplication", Proceedings of the International Conference on Parallel Processing , volume 2, 1994, pp. 47-51.

[13] E. S. H. Hou, N. Ansari and H. Ren, "A Genetic Algorithm for Multiprocessor Scheduling", Proceedings of IEEE Transactions Parallel and Distributed Systems, volume 5, number 2, Feb 1994, pp. 113-120.

[14] Srinivas M., Patnaik L.M., "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms", Proceedings of IEEE Transactions on Systems, Man, and Cybernetics, Volume 24 , Issue 4, April 1994, pp. 656-667.

[15] A.L. Corcoran and R.L. Wainwright, "A Parallel Island Model Genetic Algorithm for the Multiprocessor Scheduling Problem", Proceedings of ACM symposium on Applied computing, Phoenix, Arizona, USA, March 06-08, 1994, pp.483-487.

[16] M.K. Dhodhi, I. Ahmad and I. Ahmad, "A Multiprocessor Scheduling Scheme Using Problem – Space Genetic Algorithms", Proceedings of IEEE International Conference Evolutionary Computation, 1995, pp. 214-219.

[17] G.W. Greenwood, C. Lang and S. Hurley, "Scheduling Tasks in Real – Time Systems Using Evolutionary Strategies", Proceedings of 3rd Workshop Parallel Distributed Real-Time Systems, 1995, pp. 195-196.

[18] I. Ahmad and M.K. Dhodhi, "Multiprocessor Scheduling in a Genetic paradigm", Proceedings of Journal Parallel Computing, Volume 22, Issue 3, March 1996, pp. 395-406.

[19] W. Ying and L. Bin, "Job-Shop Scheduling using Genetic Algorithm", Proceedings of Systems, Man, and Cybernetics, IEEE International Conference on 4-17 Oct. 1996, Beijing, China.

[20] M. Grajcar, "Genetic List Scheduling Algorithm for Scheduling and Allocation on a Loosely Coupled Heterogeneous Multiprocessor System", Proceedings of Design Automation Conference (DAC), volume 17, June 1999, pp. 280-285.

[21] Y.H. Lee and C. Chen, "A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems", Proceedings of 6th International Conference Systems and Applications, 1999, pp. 382-387.

[22] S. M. Alaoui, O. Frieder and T. El-Ghazawi, "A Parallel Genetic Algorithm for task Mapping an Parallel Machines", Proceedings of Symposium on Parallel and Distributed Processing, April 1999, pp. 201-209.

[23] R.C. Correa and A. Ferreiera, "Scheduling Multiprocessor Tasks with Genetic Algorithms", Proceedings of IEEE Transactions Parallel and Distributed Systems, volume 10, number 8, 1999, pp. 825-837.

[24] A.Y. Zomaya and Y. Hwei, "Observations on using Genetic Algorithms for Dynamic Load Balancing", Proceedings of Journal IEEE Transactions on Parallel and Distributed Systems, Volume 12 Issue 9, September 2001, pp. 899-911.

[25] S.S. Seiden, "Preemptive Multiprocessor Scheduling with Rejection", Proceedings of Theoretical Computer Science, Volume 262, Issues 1–2, 6 July 2001 , pp. 437-458.

[26] A.S. Wu, H. Yu, S. Jin, K.C. Lin and G. Schiavone, "An Incremental Genetic Algorithm Approach to Multiprocessor Scheduling", Proceedings of IEEE Transactions on Parallel and Distributed Systems, Volume 15, Issue 9, September 2004, pp. 824-834.

[27] A.J. Page and T.J. Naughton., "Dynamic Task scheduling using Genetic Algorithms for Heterogeneous Distributed computing", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), April 2005.

[28] M. Omar, A. Baharum and Y.A. Hasan, "A Job-Shop Scheduling Problem (JSSP) Using Genetic Algorithm (GA)", Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications University Sains Malaysia, Penang, June 13–15, 2006.

[29] M. Li, B. Yu and M. Qi, "PGGA: A Predictable and Grouped Genetic Algorithm for Job Scheduling", Proceedings of Journal Future Generation Computer Systems - Parallel input/output management techniques (PIOMT) in cluster and grid computing, Volume 22 Issue 5, April 2006, pp. 588-599.

[30] M. Nikravan and M.H. Kashani, "A genetic Algorithm for Process Scheduling in Distributed Operating Systems Considering Load Balancing", Proceedings of 21 European Conference on Modelling and Simulation, Prague, Czech Republic (2007) 645-650.

[31] S. Jin, G. Schiavone and D. Turgut, "A Performance Study of Multiprocessor Task Scheduling Algorithms", Proceedings of The Journal of Supercomputing, Volume 43, Issue 1, January 2008, pp. 77–97.

[32] R. Hwang, M. Gen and H. Katayama, "A Comparison of Multiprocessor Task Scheduling algorithms with Communication Costs", Proceedings of Journal Computers & OR, volume 35, 2008, pp. 976-993.

[33] B. Sahoo, S. Mohapatara and S.K. Jena, "A Genetic Algorithm Based Dynamic Load Balancing Scheme for Heterogeneous Distributed Systems", Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2008, Las Vegas, Nevada, USA, July 14-17, 2008.

[34] F.A. Omara and M.M. Arafa, "Genetic Algorithms for Task Scheduling Problem" Proceedings of Foundations of Computational Intelligence, Volume 3, Studies in Computational Intelligence, volume 203, Springer, Berlin, Heidelberg, 2009.

[35] S.R. Vijyalakshmi and G. Padmavathi, "Multiprocessor Scheduling For Tasks with Priority Using GA", Proceedings of International Journal of Computer Science and Information Security, IJCSIS, USA, Vol. 6, No. 3, December 2009, pp. 093-100.

[36] X. Tong and W. Shu, "An efficient Dynamic Load Balancing Scheme for Heterogeneous Processing System", Proceedings of Computational Intelligence and National Computing, 2009, pp. 319 – 322.

[37] Z. Jiang and S. Feng, "A Fast Hybrid genetic algorithm in heterogeneous computing environment", Proceedings of Fifth International Conference on Natural Computation, vol. 4, August 2009, pp. 71–75.

[38] H. Lotfii, A. Broumandania and S. Lotfi, "Task Graph Scheduling in Multiprocessor Systems Using a Coarse Grained Genetic Algorithm", Proceedings of Computer Technology and Development (ICCTD), 2nd International Conference, November 2010.

[39] S. Rashtbar, A. Isazadeh and L.M. Khanly, "A New Hybrid Approach for Multiprocessor System Scheduling with Genetic Algorithm and Tabu Search(HGTS)", Proceedings of Information Sciences and Interaction Sciences (ICIS) 2010 3rd International Conference, 2010, pp. 626-631.

[40] S. Gupta, G. Agarwal and V. Kumar, "Task Scheduling in Multiprocessor System Using Genetic Algorithm", Proceedings of IEEE 2nd International Conference Machine Learning Computation (ICMLC'10), 2010, pp. 267-271.

[41] S.R. Vijyalakshmi and G. Padmavathi, "A Performance Study of GA and LSH in Multiprocessor Job Scheduling", Proceedings of International Journal of Computer Science Issues IJCSI, Volume 7, Issue No. 1, January 2010, pp. 37-42.

[42] A.J. Page, T.M. Keane and T.J. Naughton, "Multi – Heuristic dynamic task allocation using Genetic Algorithms in a Heterogeneous Distributed System", Proceedings of Journal of Parallel and Distributed Computing Volume 70, Issue 7, July 2010, pp. 758-766.

[43] M.I. Daoud and N. Kharma, "A Hybrid – genetic Algorithm for Task Scheduling in Heterogeneous processor networks", Proceedings of Journal of Parallel and Distributed Computing, Volume 71 Issue 11, November 2011, pp. 1518-1531.

[44] A. Alexandrescue, I. Agvariloaei and C. Mitica, "A genetic algorithm for Mapping Tasks in heterogeneous Computing Systems", Proceedings of System Theory, Control, and Computing (ICSTCC), 15th International Conference, 2011, pp. 1-6.

[45] J. Singh and G. Singh, "Improved Task Scheduling on Parallel System Using Genetic Algorithms", Proceedings

of International Journal of Computer Applications, Volume 39, Issue No. 17, February, 2012, pp. 17-22.

[46] S.K. Nayak, S.K. Padhay and S.P. Panigrahi, "A novel algorithm for dynamic task scheduling", Proceedings of Future Generation Computer Systems, Volume 28, Issue 5, May 2012 , pp. 709-717.

[47] D.M. Abdelkader and F. Omara, "Dynamic Task Scheduling algorithm with load balancing for heterogeneous computing system", Proceedings of Egyptian Informatics Journal, Volume 13, Issue 2, July 2012 , pp. 135-145.

[48] M. Etinski, J. Corbalan, J. Labarta and M. Valero, "Parallel Job scheduling for Power Constrained HPC Systems", Proceedings of Parallel Computing, volume 38 Issue 12, December 2012, pp. 615-630.

[49] N. Arora, "Analysis and Performance Comparison of Algorithms for Scheduling Directed Task Graphs to Parallel Processors", Proceedings of International journal of emerging trends in Engineering and development, volume 4 (2), 2012, pp. 793-802.

[50] B. Jiang, LI R., LI R. and Han D., "An Improved Genetic Algorithm for Load Balance in Multiprocessor Systems", Proceedings of 14th International Conference on Advanced Communication Technology (ICACT), 2012, pp. 870-874.

[51] L.D. Briceno, H. J. Siegel, A. A. Maciejewski and M. Oltikar, "Characterization of the Iterative application of Makespan Heuristics on non-Makespan machines in a heterogeneous parallel and distributed environment", Proceedings of The Journal of Supercomputing, Volume 62, Issue 1, October 2012, pp. 461–485.

[52] B. Jiang, R. Li, R. Li and D. Han, "Improved Genetic Algorithm for Load Balance in Multiprocessor Systems", Proceedings of 14th International Conference Advanced Communication Technology (ICACT), PyeongChang, South Korea, 2012.

[53] M. Wang, X. Wang, K. Meng and Y. Wang, "A New Genetic Algorithm for Release – Time Aware Divisible – Load Balancing", Proceedings of International Journal of Pattern Recognition and Artificial Intelligence, Volume 27, Issue 07, November 2013.

[54] H. Tabatabaee, M.R. Akbarzadeh-T and N. Pariz, "Dynamic Task Scheduling Modeling in Unstructured Heterogeneous Multiprocessor Systems", Proceedings of Journal of Zhejiang University SCIENCE C, Volume 15, Issue 6, June 2014, pp 423–434.

[55] I. Grasso, S. Pellegrini, B. Cosenza and T. Fahringer, "A uniform approach for programming distributed heterogeneous computing systems", Proceedings of Journal of Parallel and Distributed Computing, Volume 74, Issue 12, December 2014, pp. 3228-3239.

[56] Divya Jain and Sushil Chandra Jain, "Load Balancing in Fault-Tolerant Real –Time Systems for Periodic Task Scheduling", Proceedings of International Conference Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 2015.

[57] A.S. Hanamakkanavar and V.S. Handur, "Load Balancing in Distributed Systems: A survey", Proceedings of International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE), Volume 14, Issue 2, April 2015.

[58] Yichuan Jiang, "A Survey of Task Allocation and Load Balancing in Distributed Systems", Proceedings of IEEE Transactions on Parallel and Distributed Systems, Volume 27, Issue 2, February 2016.