Abstract: Heuristic based strategies have always been of interest for researchers to achieve sub-optimal solutions for various NP-Complete problems. Human evolution based methods have been an inspiration for research since ages. One of the many evolutionary strategies based on the principle of genetic algorithm have been able to provide much sought after 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 a multi-objective combinatorial optimization problem, with key parameters being 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 like list scheduling, task duplication and critical path based. The algorithms proposed by various authors like Highest Level First Execution Time (HLFET), Modified Critical Path (MCP), Duplication Scheduling Heuristic (DSH), Linear Clustering (LC) and Dynamic Critical Path (DCP), belonging to each heuristic mentioned before will be taken under study. Previously these algorithms 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 identify the gaps in existing literature that forms the base of every research focusing in the direction of improvement of task graph scheduling algorithms. Further, this paper will propose a hybrid meta-heuristic i.e. Genetic Algorithm based task graph scheduling solution and perform a comparative study of aforementioned algorithms.

Keywords: Directed Acyclic Graphs, Task Graph Scheduling, List Scheduling, Task Duplication, Critical Path, Genetic Algorithm.

I. 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 fulfillment of above stated goals. These systems have distributed computational units connected and organized in some form of a network [15,27,32]. The units are connected in an absolute and unambiguous manner. The units are self – sufficient but can also work jointly for the fulfillment of objectives like optimal resource usage and sharing, concurrency, scalability, fault – tolerance and transparency [37,47]. 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. It resolves the problem of scheduling a task graph of a parallel program on the available distributed environment. The important objective to be achieved is maximum and efficient utilization of various available resources [12,13,20]. Various applications like web services and scientific applications are implemented in distributed environment [14,15,16]. 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 [10,11,13,18,22,30,31,32,34]. 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) [48], Modified Critical Path (MCP) [49], Duplication Scheduling Heuristic (DSH) [50], Linear Clustering (LC) [51] and Dynamic Critical Path (DCP) [17] are a few to mention here, which belong to different classes based on the heuristics 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 [10, 11, 13, 18, 22, 30, 31, 32, 34].

Revised Manuscript Received on August 19, 2019.

* Correspondence Author
Ravreet Kaur, Department of Computer Science & Engineering, UIET, Panjab University, Chandigarh, India.
Gurvinder Singh, Department of Computer Science, Guru Nanak Dev University, Amritsar, India.
This paper proposes a solution based on Genetic Algorithm, Intelligent Genetic Algorithm (IGA), which will use the results of above mentioned heuristics also to conclude the best sub-optimal solution possible.

Table I. Various Heuristics Based Algorithms Proposed By Various Authors for Homogeneous Environment Processors

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Heuristic</th>
<th>Principle applied</th>
<th>Algorithms Under Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>List Scheduling</td>
<td>Priority given to largest task achieved using certain sorting mechanism</td>
<td>HLFET , MCP</td>
</tr>
<tr>
<td>2.</td>
<td>Task Duplication</td>
<td>Aims to optimize communication time overhead</td>
<td>DSH , LC</td>
</tr>
<tr>
<td>3.</td>
<td>Critical Path</td>
<td>Priority given to tasks having maximum effect on completion time</td>
<td>DCP</td>
</tr>
</tbody>
</table>

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 an insight into the major algorithms being considered as base of every research directed towards achieving sub-optimal solution for task graph scheduling problem. Also, these algorithms will be compared with the proposed solution i.e. IGA, further providing a basic methodology to be used in future for homogeneous environment.

II. 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. Therefore, some assumptions might be made and some criteria might be calculated or known while task scheduling execution is in process. 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]. Mostly a system model considered will consist of fully connected or partially connected homogeneous or heterogeneous processors. Further 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, 47]

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. \( \forall n_i, n_j \in E \), 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:

\[
CCR = \frac{\sum_{i=1}^{E} c(n_i,n_j)}{E} \frac{\sum_{i=1}^{V} w(n_i)}{V}
\]

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.
Fig. 1. Directed Acyclic Graph

A. 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. Completion Time
2. Load Balance
3. Resource Optimization
4. Reliability
5. Relationship between processing units

This paper will focus on the critical issues of completion time and resource utilization being satisfied by the existing solutions and proposed solution 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.

III. EXISTING DAG BASED TASK GRAPH ALGORITHMS BASED ON ABOVE MENTIONED HEURISTICS UNDER STUDY

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Research paper and Authors</th>
<th>Year</th>
<th>Algorithm</th>
<th>Environment</th>
<th>Heuristic</th>
<th>Parameters Improved</th>
<th>Research Gaps identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A Comparison of List Scheduling for Parallel Processing Systems [T.L. Adam et.al.]</td>
<td>1974</td>
<td>HLFET</td>
<td>Homogeneous</td>
<td>List Scheduling</td>
<td>Complete execution time</td>
<td>Inappropriate use of available processors</td>
</tr>
<tr>
<td>2</td>
<td>Duplication Scheduling Heuristics (DSH): A New Precedence Task Scheduler for Parallel Processor Systems [B. Kruatrachue et.al.]</td>
<td>1987</td>
<td>DSH</td>
<td>Homogeneous</td>
<td>Task Duplication</td>
<td>Complete execution time, Average Processor Utilization</td>
<td>Increased Communication Costs</td>
</tr>
</tbody>
</table>
IV. PROPOSED ALGORITHM: INTELLIGENT GENETIC ALGORITHM (IGA)

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 Fig. 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
}

Fig. 2. Control Abstract for a Basic Genetic Algorithm

A. 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. The method used in IGA is as follows:

![Fig. 3. Method 1](image)

**Fig. 3. Method 1**

- Processor ID to which the corresponding task is assigned

![Fig. 4. Method 2](image)

**Fig. 4. Method 2**

**B. 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.

IGA would use all 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.

C. 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 and balance of all metrics like MakeSpan and Load Balance. The strength of fitness function would be affected by problem representation used. It can vary from a very simple function to a complex function. IGA 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 = TEFT() X LBF()

where, TEFT() is the total execution time factor defined as [1/MakeSpan]

LBF() is the Load Balance Factor based on Average Processor Utilization

D. 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[]. Hence, IGA would use pareto ranking system for the selection process.

E. 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,
Design and Analysis of Multi - Heuristic Based Solution for Task Graph Scheduling Problem

then new generation is made by performing crossover of all offsprings 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. IGA 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 strings. Further the probability of crossover will be decided by performing experiments over the range from 0 to 1 and best will be chosen accordingly.

F. 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. IGA 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 [1]. 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.

G. 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.

H. 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 are 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.

V. SIMULATION

The proposed solution i.e. IGA have been tested on well known Gaussian – Elimination benchmark parallel program. Different sizes of Gaussian – Elimination task graphs have considered to compare the proposed solution with existing benchmark task graph algorithms. IGA have been compared with well known task graph scheduling algorithms such as Highest Level First Execution Time (HLFET), Modified Critical Path (MCP), Duplication Scheduling Heuristic (DSH), Linear Clustering (LC) and Dynamic Critical Path (DCP). The experiments were performed on Intel Core i5 processor @ 2.5GHz with 8 GB 1600 MHz DDR3 RAM. The above mentioned algorithms have been implemented in JAVA Programming language. Gaussian – Elimination task graphs of different sizes have been used i.e. from 8 to 26. Further, to increase the data set size, communication to computation ration (CCR) values of 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75 and 2.00 have been used.

VI. PERFORMANCE EVALUATION AND COMPARISON

The performance of proposed IGA is compared with the previously discussed scheduling algorithms, namely HLFET, MCP, DSH, LC and DCP based on performance metrics as defined below.

1. MakeSpan = Total time taken by a DAG to complete execution
2. %Efficiency = ((Average Processor Utilization/Number of Processors) *100)
3. %Improvement = ((MakeSpan/Time taken on uniprocessor machine)*100)
including the communication delays to reduce the completion time and to increase the throughput of the system. The proposed IGA is different from the Basic GA as it uses a good initial population to search a large solution space in an intelligent way in order to find the best possible solution within an acceptable CPU time. The proposed method found a sub-optimal solution for assigning the tasks to the homogeneous parallel multiprocessor system. Performance analysis of the proposed GA, HLFET, MCP, DSH, LC, and DCP scheduling algorithm shows that proposed IGA gives better result for task graph scheduling in distributed environment.

VIII. 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.

The drawbacks identified in the studies undertaken point towards the need to devise task scheduling along with load balancing for distributed systems that 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.

REFERENCES

Design and Analysis of Multi - Heuristic Based Solution for Task Graph Scheduling Problem


53. 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.


AUTHORS PROFILE

Ravreet Kaur, currently working as Assistant Professor, Deptt. of CSE, UIET, Panjab University, India and pursuing Ph. D. from Deptt. of CSE, GNDU, Amritsar, India. Research areas of interest are Parallel and Distributed Computing.

Gurvinder Singh currently working as Professor, Deptt. of Computer Science, Guru Nanak Dev University, Amritsar, India. Research areas of interest are Parallel and Distributed Computing and Data Sciences.