ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

Implementation of Mapping Heuristic Genetic Algorithm

Amanpreet Kaur[1], Prabhjot Kaur[2]

M. Tech Research Scholar, Department of Computer Science, Guru Nanak Dev University, Amritsar, India[1]

M. Tech Research Scholar, Department of Computer Science, Guru Nanak Dev University, Amritsar, India[2]

Abstract- Mapping Heuristic APN Algorithm in which the list is ordered according to node priorities with the highest priority node. Schedule node to the processors which gives highest node. Calculate start time of node, routing table maintained for each processor, each entry of table indexed by destination processing element. Routing table will direct message from one machine to along a path with minimum communication time. Shortest path with between the processing elements are stored in routing table. Message is sent, the route from source to destination machine become busy, carrying message of certain amount of time. When message is received its route become free and this route can be used for other processor for transmission of message again. Every time a message sent and message received event is processed the routing table will be updated to profile the direction for fastest communication routes at any time. Append all ready successor nodes of n_i according to priority to the ready node list.

Keywords: DAG, multiprocessor scheduling, genetic algorithm heuristics, mapping heuristics

I. INTRODUCTION

Parallelization of an application involves the decomposition of the program into several sub tasks, analyzing the dependencies between the sub tasks and scheduling the sub tasks [1]. The assignment of tasks to the processing units and defining their execution order statically is referred to as task scheduling. Task scheduling is a NP- complete problem. Task scheduling is crucial to the performance of the application. Hence we have to find suitable heuristic technique to find "optimal" solutions. Heuristics try to find near optimal solution. For these heuristics, the program is modelled as a directed acyclic, graph, called task graph, where the nodes represent the tasks of the program and the edges are the communication.

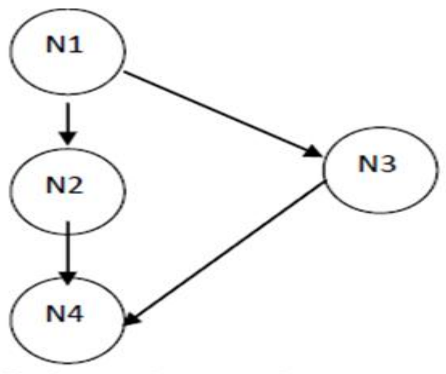
There are some assumptions for parallel system:

(1) Processor can communicate with each other through a dedicated identical communication link. (2) The task requires only one processor at a time. (3) Communication can be performed at the same time. Objective of scheduling is to map tasks onto machines and order their execution so that task precedence requirements are satisfied and there is a minimum makespan, schedule length ratio and processor utilization.

One of the best heuristic methods is Genetic Algorithm (GA). There are many researches under the topics of solving the static task scheduling using GAs in the multiprocessor systems and the distributed systems. In this paper, a novel GA is presented which has a good ability to solve the above problem using the Mapping Heuristic Approach.

II. SCHEDULING PROBLEM

Directed Acyclic Graph model is assumed for task graph. The system consists of a number of identical processors which could be limited or unlimited. DAG is a generic model of a parallel program consisting of a set of processors among which there are dependencies [4], [5]. Each process is an indivisible unit of execution it generates its output. A node has one or more input or has a one or more output to various nodes. When all input are available, the node is triggered to execute. After its execution it generates its output. A set of node $\{n_1, n_2, n_3 \dots \dots n_n\}$ are connected by a set of connected edge which are represented by (n_i, n_j) where n_i is called the Parent node and n_j is called the child node. A node without parent is called an Entry node and a node without child called an Exit node. The weight of a node, denoted by w (n_i), represents the process execution time of a process. Since each edge corresponds to a message transfer from one process to another, the weight of an edge, denoted by c (ni, nj) is equal to the message transmission time from node n_i to n_j . Thus c (n_i, n_j) becomes zero when n_i and n_j are scheduled to the same processor because intraprocessor communication time is negligible compared with the interprocessor communication time. The node and edge weights are usually obtained by estimations.



Task precedence graph

Basic techniques in dag scheduling: The two main attributes priorities are the t-level and b-level.

t- level: The t-level of a node n_i is the length of the longest path from an entry node to n_i . Here, the length of a path is the sum of all node and edge weights along the path. The t-level of n_i highly correlates with n_i earliest start time, denoted by T_s (n_i), which is determined after n_i is scheduled to a processor.

b- level: The b-level of a node is bounded by the length of the critical path. A Critical Path (CP) of a DAG, is the longest path from entry node to an exist node.

III. SCHEDULING ALGORITHM:

- **Unbounded Number of Clusters (UNC):** These algorithms schedule the DAG to an unbounded number of clusters. The processors are assumed to be fully-connected. The technique employed by these algorithms is also called clustering.
- **Bounded Number of Processors (BNP):** These algorithms schedule the DAG to a bounded number of processors directly. The processors are assumed to be fully-connected. Most BNP scheduling algorithms are based on the list scheduling technique.
- **Task Duplication Based (TDB):**
In duplication-based scheduling, different strategies can be employed to select ancestor nodes for duplication. Some of the algorithms duplicate only the direct predecessors whereas some other algorithms try to duplicate all possible ancestors.
- **Arbitrary Processor Network (APN):** These algorithms perform scheduling and mapping on the target architectures in which the processors are connected via a network of arbitrary topology. The algorithms in this class take into account specific architectural features such as the number of processors as well as their interconnection topology.

IV. MAPPING HEURISTIC ALGORITHM

The MH (Mapping Heuristic) Algorithm is described below:

```
i.    Compute the level of each node n_i in the task graph.
ii.   Initialize a ready node list by inserting all entry nodes in the task graph. The list is ordered according to node priorities, with the highest priority node first.
iii.  while ready node list is not empty do
iv.   n_i the first node in the list
v.    Schedule n_i to the processor which gives the smallest start time.
vi.   Append all ready successor nodes of n_i, according to their priorities, to the ready node list.
vii.  end while
```

The MH algorithm first assigns priorities by computing the levels of all nodes. A ready node list is then initialized to contain all entry nodes ordered by putting the highest priority node first. In the while loop, the first node $\mathbf{n_i}$ in the ready node list is scheduled to a processor that gives the smallest start time. In calculating the start time of node, a routing table is maintained for each processor. The table contains the information as to which path to route messages from the parent nodes to the node under consideration. The start time of $\mathbf{n_i}$ is then taken to be the larger of the message arrival time and the ready time of the processor which is defined as the finish time of the last scheduled node. After $\mathbf{n_i}$ is scheduled, all the ready successor nodes of $\mathbf{n_i}$ are appended to the ready node list.

V. GENETIC ALGORITHM

GAs are inspired by Darwin's theory about evolution- the survival of the fittest. They were proposed and developed in the 1960s by John Holland, his students and his colleagues at the University of Michigan. Genetic Algorithms are search algorithms that are based on concepts of natural selection and natural genetics. Genetic algorithm was developed to simulate some of the

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

processes observed in natural evolution, a process that operates on chromosomes. The genetic algorithm differs from other search methods in that it searches among a population of points, and works with a coding of parameter set, rather than the parameter values themselves. It also uses objective function information without any gradient information.

Algorithmically, the basic genetic algorithm (GAs) is outlined as below:

STEP I [Start] Generate random population of chromosomes, that is, suitable solutions for the problem.

STEP II [Fitness] Evaluate the fitness of each chromosome in the population.
STEP III [New Population] Create a new population by repeating following steps until new population is complete:
i. [Selection] Select two parent chromosomes from a population according to their fitness. Better the fitness, the bigger chance to be selected to be the parent.
ii. [Crossover] With a crossover probability, cross over the parents to form new offspring, that is, children. If no crossover was performed, offspring is the exact copy of parents.
iii. [Mutation] With a mutation probability, mutate new offspring at each locus.
iv. [Accepting] Place new offspring in the new population.
STEP IV [Replace] Use new generated population for a further run of the algorithm.
STEP V [Test] If the end condition is satisfied, stop, and return the best solution in current population.

STEP VI [Loop] Go to step 2.

The genetic algorithms performance is largely influenced by crossover and mutation operators.

VI. PROPOSED ALGORITHM

The actual number of generation needed depends upon the number of task and population size. Fewer the number of task and the larger the population size, the fewer the number of generation required for convergent solution.

Crossover & Mutation:_ Crossover the first genetic operation performs on the genomes. Crossover examines the current solution in order to find better ones. The crossover between two dominant parents chosen by the selection gives higher probability of producing offspring having traits. All population are randomly paired for crossover. Crossover site or location where crossover paired. where the value is small for all nodes than the value of for all nodes after the site. A valid site must be one task before and after the site. Otherwise, either none or all tasks would be swapped and the resultant genome would be different. Following crossover operator mutation is performed:

Randomly select task swapped with another randomly selected task both within the same genome. After breeding a new generation of scheduling the fitness of each genome in the population is computed by calling fitness. Finish Time of genome first computed and store array. Task Precedence between processor must be enforced to consider the time complete all of the tasks scheduled in genome. Finish time for all genomes in population are known the fitness of genome is calculated.

Fitness function: The fitness function interprets the chromosome in term of physical representation. The fitness function must measure the quality of the chromosome in the population. Main objective to minimize the finishing time of schedule, the fitness value of the schedule will be maximum finishing time observed in a population, minus the finsh time of schedule.

Fitval [] = (maximum finish time in population – Finish time schedule)/ maximum finish time.

The fitness function of GA is generally the objective function that requires to be optimized. The fitness function has a higher value when the fitness of the chromosome is better than others. The fitness function introduces a criterion for selection of chromosomes.

Selection: This selection operator is intended to improve the average quality of the population by giving the high quality chromosome a better chance to get copied into the next generation. Selection pressure characterizes the selection schemes. It is defined as the ratio of the probability of selection of the best chromosome in the population to that of an average chromosome. In this Roulette wheel selection is used. Roulette wheel Selection is known as Fitness Proportionate selection. In roulette wheel individuals are selected with a probability that is directly proportion to their fitness value. The probabilities of selecting a parent can be seen as spinning a roulette wheel the size of the segment for each parent being proportional to its fitness. Those with the largest fitness have more probability of being chosen.

Reproduce: creates a new population of genomes by selection from the current population using weighted random number based on the fitness values. Thus genomes with higher fitness value is have a better chance of serving to next generation.

ALGORITHM

1. Initialize ready node list by inserting all entry node in the task graph. The list is ordered according to node priorities with thw highest priority node.
2. While ready node list is not empty do
3. n_i first node in the list.
4. Call Generate Schedule
5. Repeat step 6 to 9 generations number of times.
6. Call reproduce
7. Call crossover
8. Call mutation
9. Compute the fitness value of each chromosome store fitness value.
10. Compute fitness value. Store result in fitness value select chromosome with best fitness.
11. Append all ready successor node of n_i. According to priority to the ready node list.
12. End while.

GENERATE SCHEDULE

1. Schedule n_i to the processors which gives highest node.
2. Calculate start time of node, Routing table maintain for each processor.
3. Each entry of table indexed by destination processing element.
4. Routing table will direct message from one machine to along a path with minimum communication time.
5. Shortest path with between the processing elements are stored in routing table.
6. Message is sent, the route from source to destination machine become busy, carrying message of certain amount of time.
7. When message is received its route become free and this route can be used for other processor for transmission of message again.
8. Every time a message sent and message received event is processed the routing table will be updated to profile the direction for fastest communication routes at any time.

CROSSOVER:

1. Perform the crossover operation of two chromosome pair A and B.
2. C1: Select crossover Generate a random number crossover between 0 and the maximum node of task graph.
3. C2: Do C3 for each processor Pi in chromosome A and B.
4. C3: Find node n_i in the processor Pi that has Highest priority node and nj is the node following n_i, where $c = n_i < n_j$ highest for i.
5. C4: Do C5 for each processor Pi in chromosome A and B.
6. C5: Using the crossover site selected in C3 exchange the part of chromosome A and B.

MUTATION:

1. Mutate a chromosome to form new chromosome.
2. M1: Randomly Pick Node n_i.
3. M2: Randomly Pick Node n_j Form a new chromosome by exchange the two nodes n_i and n_j in the chromosome.

REPRODUCE:

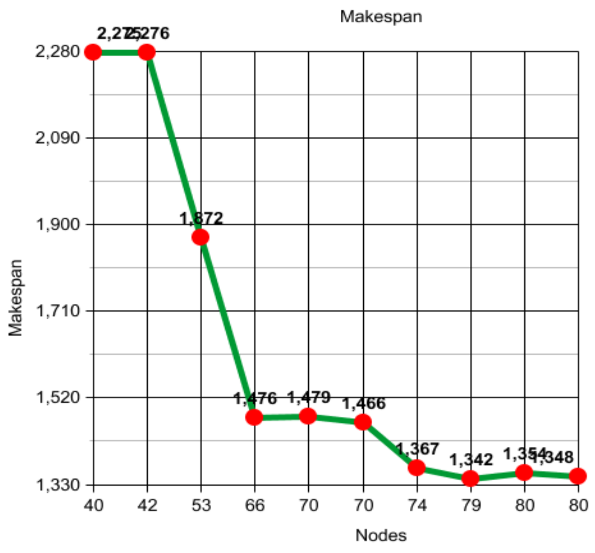
1. Population of chromosome Pop and generate new population NEWPOP.
2. R1: Construct roulette wheel: Let Fitness sum be the sum of all the fitness value of chromosome in POP. Form fitness_ sum slot and assign chromosome to occupy number of slots according to their fitness value of the chromosome.
3. R2: Select the first chromosome in POP with highest fitness value. Add this chromosome to NEWPOP.
4. R3: Let NEWPOP be the number of chromosome in POP. Repeat R4 NEWPOP -1 time.
5. R4: Generate Random number between 1 and fitness_sum. Use it to index b into the slots to find the corresponding chromosome. Add the chromosome to NEWPOP.

VII. EXPERIMENTAL RESULT

Makespan time graph

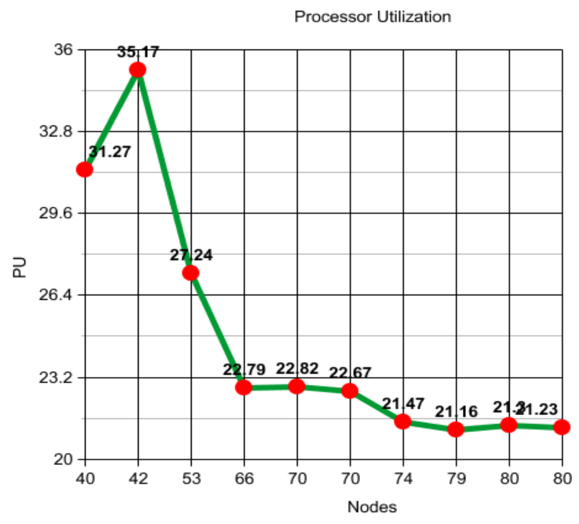
ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013



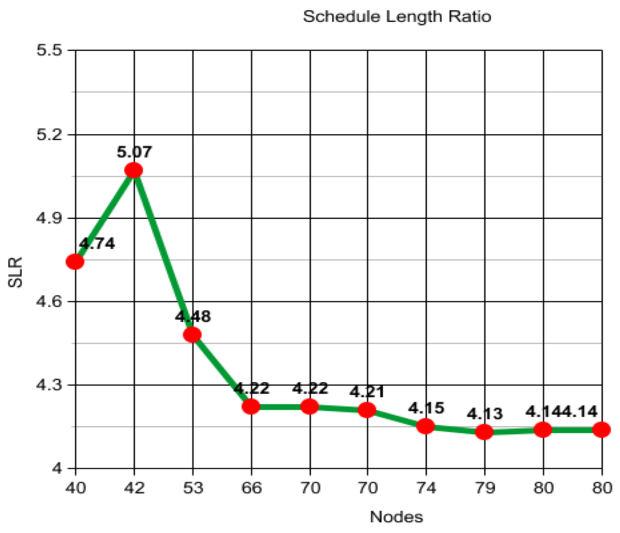
Makespan is defined as the completion time of the algorithm. Lesser the makespan less time to execute the algorithm more efficient is the algorithm. Makespan is calculated by measuring the finishing time of the exit task by the algorithm. As Graph shows at node 40 makespan per generation is 2275 then as nodes increases means highest node 80 nodes. in list then its makespan time decreases to 1348.

Processor utilization graph



Processor utilization is the most important aspect of determining the performance of algorithm. At 40 nodes having processor utilization approx 31 % but as nodes increases to 80 processor utilization decreases because with increasing nodes there are more work has to done by single processors so processors utilization is decreases approx 22%.

Schedule length ratio



Schedule length ratio is defined as the ratio DAG. The lesser the values of SLR the more efficient is the algorithm, but the SLR cannot be less than the Critical Path values.

Scheduled Length Ratio = Makespan / Critical path

Due to this effect as per number of node are increases its SLR ratio should be decreases and above graph shown same as explained if node is 40 then it has higher SLR 4.74 and as nodes increase to 80 then it varies down and having smallest SLR 4.14.

VIII. CONCLUSION

Proposed algorithm using a novel encoding scheme, an effective initial population generation strategy and computationally efficient genetic search operator. In experimental study almost having linear result. As nodes are increasing its makespan time decreases. Due to flexible value with increases nodes processor utilization also decreases. As makespan time decreases the schedule length also decreases when nodes increases. As applying genetic to find an optimal result minimize parameters at sum instance near to goal. The current researches can consider further elaboration by applying various techniques such as improving computation time, incorporating network topology and communication traffic. Another avenue of further research is to extend the applicability of proposed algorithm. While targeted to be used in APN algorithm may be extended to handle BNP and UNC scheduling as well. Some efficient algorithmic techniques for scheduling messages to link need to sought lest the time complexity of the algorithm. The proposed algorithm has shown an encourage performance. Further improvement may be possible if we can determine an optimal set of control parameters including crossover rate, mutation rate, population size, number of generation and number of parallel processors used. However finding an optimal parameters set for particular genetic algorithm is up till now an open research problem.

REFERENCES

1. Yu-Kwong Kwok, "Benchmarking and Comparison of the Task Graph Scheduling Algorithms", Journal of Parallel and Distributed Computing 59, pp.381-422, 1999.
2. L. D. Davis (Ed.), The Handbook of Genetic Algorithms, New York, Van Nostrand Reinhold, 1991.
3. G. C. Sih and E. A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection Constrained Heterogeneous Processor Architectures", IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 2, Feb. 1993, pp. 75-87.
4. F. Gruau, G. R. Joubert, F. J. Peters, D. Trystram and D. J. Evans, "The Mixed Parallel Genetic Algorithm", Parallel Computing: Trends And Applications (Proc. of the International Conference ParCo' 93), 1994, pp. 521-524.
5. Yu-Kwong Kwok and Ishfaq Ahmad, "A Parallel Genetic-Search-Based Algorithm for Scheduling Arbitrary Task Graphs to Multiprocessors".
6. Nafiseh Sedaghat, Hamid Tabatabaee-Yazdi, Mohammad-R. Akbarzadeh-T, "Pareto Front Based Realistic Soft Real-Time Task Scheduling with Multi-objective Genetic Algorithm on Arbitrary Heterogeneous Multiprocessor System", Journal of Internet Technology Volume 12 (2011) No.1
7. Jasbir Singh and Gurvinder Singh, "Improved Task Scheduling on Parallel System using Genetic Algorithm", International Journal of Computer Applications (0975 – 8887) vol 39, no.17, Feb 2012.
8. Kamaljit Kaur, Amit Chhabra and Gurvinder Singh, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security (IJCSS), Volume (4): Issue (2)

BIOGRAPHIES

Amanpreet Kaur received the B. Tech degree in Computer Science and Engineering from DAVIET Jalandhar in 2011 and M. Tech degree in Software System from Guru Nanak Dev University Amritsar in 2013 respectively.

Prabhjot Kaur received the B. Tech degree in Computer Science and Engineering from Guru Nanak Dev Engineering College Ludhiana in 2011 and M. Tech degree in Computer Science and Engineering from Guru Nanak Dev University Amritsar in 2013 respectively.