International Journal of Soft Computing 3 (3): 254-259, 2008
ISSN: 1816-9503

A New Genetic Algorithm Based on Neighborhood Search and Tabu List (GTNS) for Task Scheduling in Multiprocessor

[1]Hadi Shahamfar and [2]Sohrab khanmohamadi
[1]Department of Computer Engeneering,
Islamic Azad University Science and Research Branch, Tehran, Iran
[2]Department of Control Engeneering,
Faculty of Electrical and Computer Engeneering, Tabriz University, Tabriz, Iran

Abstract: Task scheduling is a crucial and complex problem in Multiprocessor systems and is defined NP-complete problem. Nowadays, with increasing size of programs and information. The multiprocessor Systems are widely used in parallel computing. In Many cases we can divide a huge problem into some small portions and assign these small problems to processors. This would result in a considerable reduction in the execution time of programs. Previous Algorithms have various restrictions in their assumptions such as tasks considered independent, task graph produced randomly or the zero considered communication delay. On the other hand the complexity of algorithms has been ignored. This is important since there should be a balance between the quality of solutions and execution time of algorithm. Comparison studies with realistic assumptions on scheduling algorithms have shown that some algorithms prefer quality of solutions to execution time of algorithms. This has caused them not to be applicable in realistic situations. In this study we present a new genetic algorithm that employs neighborhood search and tabu list for performing task scheduling (GTNS). This newly proposed algorithm has solved the mentioned problem, as well as having a reasonable execution time versus the makespan.

Key words: Task scheduling, genetic algorithm, tabu search, multiprocessor

INTRODUCTION

Scheduling for a set of dependent and independent task that execute parallely on a set of processors is important and computationally complex. Multiprocessor systems have widely been used in parallel computation. Appropriate scheduling for performing parallel programs to reach a higher efficiency is essential (Wu *et al*., 2004). This means that makespan must be set at the minimum level. Another objective of the scheduling is to provide precedence relationship among the tasks when allocating them to processors.

In this context there are a lot of issues that should be addressed in terms of dependent or independent tasks, task graph produced randomly, communication delay or the homogeneity or heterogenity of the multiprocessor systems (Parsa *et al*., 2007).

Various studies have been performed in this area (Kwok and Ahmad, 1999). Almost all of them have considered the easiest assumptions and have proposed a respective scheduling (Correa *et al*., 1999). However, in the realistic environment, the problems are not always compatible with such simple assumptions and all the aspects should be considered.

Reaching an efficient solution is not easy. A lot of proposed algorithms have suggested a partial efficient solution with simple assumptions and majority of them have preferred complexity of algorithm (increase execution time) to the quality of solutions. However, execution time of the algorithm is not scalable to quality of solution. In this study we proposed a scheduling algorithm for allocation of tasks to set of processors which computes the communication cost between dependant tasks when these tasks were not executed in the same processor.

This algorithm is proposed for homogenous multiprocessors since this model is more compatible with a realistic world which user frequently works with. A comparative study between nine task scheduling algorithms, Min-min, Chaining, A*, Genetic algorithms, Simulated Annealing, Tabu search, Highest Level First Known Execution Time (HLFET), Insertion Scheduling Heuristics (ISH)? Duplication Scheduling Heuristic (DSH),

Corresponding Author: Hadi Shahamfar, Islamic Azad University, Science and Research Branch, No.15 Golestan 4, Pezeshkan Street, Abressan Avenue, Tabriz, Iran

showed that genetic algorithms and tabu search have a better performance compared with the other algorithms (Jin *et al*., 2008).

This has motivated us to propose a new neighborhood search and Tabu list based genetic algorithm and compare it with a previous study (Jin *et al*., 2008) of multiprocessor scheduling. In this study, we had employed two widely used bench marks namely Gauss-Jordan elimination, LU decomposition in order to evaluate the proposed algorithm.

The importance of task scheduling in multiprocessor system has resulted in various related studies in this regard.

In a previous comparison study, Jin *et al*. (2008) has compared 9 previously mentioned algorithms, using G-j and LU benchmarks. The results of that study showed that genetic algorithm and tabu search due to their nondeterministic structure are appropriate candidates for NP-complete problems. Few genetic algorithms have been proposed for solving such problems and each suggestion has some advantages and disadvantages. The main difference between these algorithms is on chromosome encoding and generic operators. This is mainly because chromosome encodings have important effect on genetic operators. Doing a comparison between these algorithms is difficult because of 2 reasons: First Majority of algorithms are based on Authors assumptions and do not consider all aspects of the problem. Second, there exists no standard benchmark for evaluation of algorithms (Parsa *et al*., 2007). The current study a partially standard benchmark G-J and LU was used for the evaluation.

A lot of algorithms do not have an appropriate chromosome encoding. Further more they use simple genetic operators for producing generations which do not provide genetic diversity in the genetic population. Therefore, a string encoding was used in the proposed algorithm.

TASK SCEDULING PROBLEM

In this study, we formulate the scheduling problem (Dhodhi and Ahmad, 1995). Let $P = \{P_i : i=1, \ldots, m\}$ be a set of m homogeneous fully connected processors and let the application program be modeled by directed acycle graph $T = \{T_j : j = 1, \ldots, n\}$ of n tasks. For any 2 tasks i, $j \in T$, $i<j$ means that task j cannot be scheduled until i has been complete, i is a predecessor of j and j is a successor of i. Weights associated with the nodes represent the computation cost and the weights associated with edge represent the communication cost.An example of a Directed Acyclic Graph (DAG) consisting of 11 tasks shown in Fig. 1. The multiprocessor sceduling is to assign

```
Step1:
    Create initial population
    Produce a quarter of the population rabdomly
    Produce the rest of the population as follows:
        Find the earliest start time (EST) for each task
        Identify all the tasks on the critical path
        Sort the task according to their EST in a linear list
        Repeat
            Select randomly a task amont the tasks ready to schedule at the beginning of the linear list
            If the selected task resides on the critical path then
                Assign the task to the processor on which the previous task on the critical path resides
            Else
                Assign the task to the processor including a task with highest interconnections with the task
            Remove the selected task from the linear list
        Until the linear list of the task is empty
Step2:
    While termination criteria not satisfied do
        For each choromosome in current population do
            Calculate its fitness value
        Create intermediate generation as follow:
            Add the fittest choromosome to the intermediate population
            Repeat
                For all chromosoms that fitnesses >2/3 avvrage fitness
                    Apply all crossover for this cromosome considering nmaxmoves and calculate fitness for these chromosome
                    Replace the best cromosome in the population and put it parrents to tabu list
                Apply tournament selection to selected two cromosome
                Apply crossover opperator
                Calculate fitness value
                Apply mutation and calculate fitnesses of thes cromosom
            Until the intermediate population size is commplet
        Copy the intermadiate population over current population
```

Fig. 1: Structure of proposed genetic algorithm

the set of tasks T onto the set of processor P in such a way that precedence constrain are maintained and to determine the start and finish time of each task with the objective to minimize the completetion time. We assume that the communication system is contention free and it permits the overlap of communication with computation. Task execution is started only after all the data have been recived from its predecessors. The communication links are full duplex.duplication of same task is not allowed. Communication is zero when two tasks are assigned to the same processor, otherwise they incure the communication cost equal to the edge weight.

NEW GENETIC ALGORITHM BASED ON NEIGHBORHOOD SEARCH AND TABU LIST (GTNS)

In this study we propose a new genetic algorithm based on neighborhood search and Tabu list. In this Algorithm in order to accelerate evolutionary process and to reach an efficient Solution with a reasonable execution time, several techniques are used: Firstly the use of an initial population in which chromosome are produced in the basis of the earliest start time of each node (task) (Parsa *et al*., 2007). Secondly, using neighborhood search (Tian and Sannonmiya, 2000) for those parents whose fitness value is more than 75% of mean population fitness. This is because the possibility of finding the best solution from doing genetic operators on these parents is higher than that of rest of population. Therefore, this algorithm performs all the crossovers with considering an nmaxmoves constant. Thirdly, a tabu list (Thesen, 1998) is used for preventing the repetition of the parents in the next generation whose children have been searched in the previous stage.

This prevention is because performing operation on such chromosomes will not produce children whose fitness is better than current generation.

Furthermore, in order to perform a more precise evaluation of fitness value of chromosomes, another parameter other than makespan i.e waiting time has been used. The coincident use of these parameters evaluates fitness of chromosomes more precisely. Another important factor in scheduling algorithms is the execution time of algorithm which should be scalable to solution quality. We have addressed this objective by restricting searches to those solutions which do not result in the considered value.

Structure of proposed genetic algorithm is shown in Fig. 1.

Chromosome encoding: In this algorithm, a new string encoding for chromosomes is used which employs the

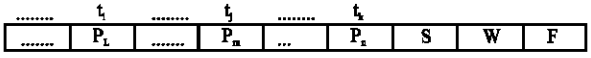

Fig. 2: Chromosome encoding

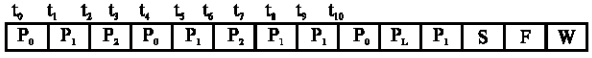

Fig. 3: An example for graph Fig. 1

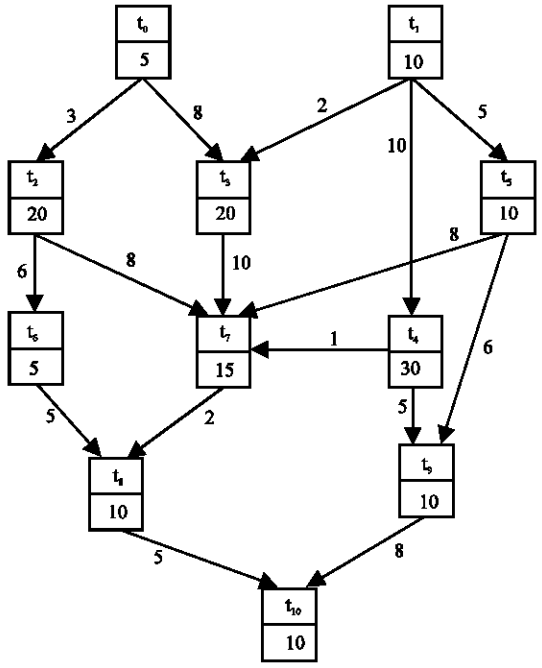

Fig. 4: A directed acycle graph

advantage of clear global precedence of each task. In the previous chromosome encodings, only the precedence of the tasks that are executed on the same processor were clear while the global precedence of tasks was not achievable and an additional string should have been used for achieving the global precedence.

The new string encoding for chromosome is demonstrated in Fig. 2.

Each element of array indicates the allocation of a task to the corresponding processor. One advantage of such encoding is the constant length of the chromosome during the genetic operation. This chromosome encoding provides the possibility of a more precise Computation of chromosome fitness value by incorporating two fields, S and W which indicate makespan and waiting time, respectively.

F represents the fitness value which is achieved through S and W fields.

For instance, Fig. 3 demonstrates an examplitory chromosome for task graph demonstrated in Fig. 4.

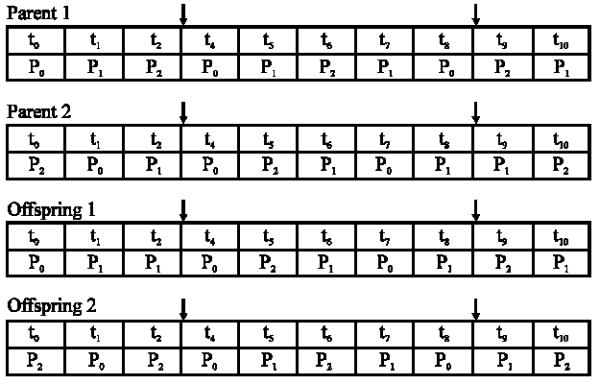

Fig. 5: Performing crossover

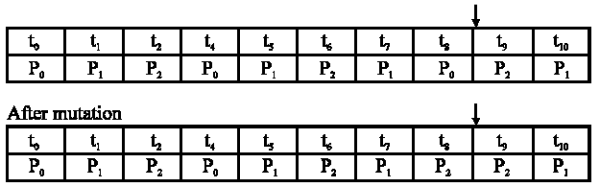

Fig. 6: Performing mutation

Production of initial population: In order to produce initial population, first, a sequential queue containing the number of tasks is made. This queue is organized on the Basis of ascending order of Earliest Start Time (EST) of each task. In this algorithm the nodes on the critical path as well its length are calculated. For each node a reference count which initially equal the number of its parents is considered. Starting from the initial point of the queue, a task whose parents has finished execution is chosen randomly and allocated to a processor. Consequently the reference belonging to all children of the chosen task is reduced by one mark. Thus the reference counts of the ready tasks equal to zero.

This cycle is repeated unit all the tasks are scheduled. It is noteworthy that the processor allocated to each task is not selected on a random basis. In this algorithm the task belonging to critical path is allocated to processor that the previous task on the critical path is operated on.

If the task does not belong to critical path it is allocated to a processor which the presented task on it has the maximum communication cost with. In case if there were several equal communication cost, one of them were chosen randomly. This algorithm tries to minimize the communication and produce a better solution. However, if all the members of the initial population are produced with this algorithm, there will be the possibility of premature convergence and repeated identical solutions. Therefore, it would be better to produce some initial populations randomly.

Genetic operators: The operators of selection, crossover and mutation are described as follow:

Selection operator: The selection operator used in this algorithm is the tournament algorithm.

Cross over operator: In the proposed algorithm for performing combination of two chromosomes, the two point crossover method is used. This operator is performed on the basis of crossover rate (denoted by P_c). On this operator two point of chromosome is selected randomly. The intervals between these 2 points are exchanged while the sequence of tasks remains intact. It is noteworthy that the selected points on each chromosome should be in identical position compared with the other. This is necessary so that the crossover operator can produce valid chromosomes. Figure 5 shows an example of random crossover.

Mutation operator: The mutation rate (denoted by P_m) provides variation and possiblity of avoiding local optimum. If a task is selected to be mutated, then its processor number will be randomly changed. This will result in production of a valid chromosome. Figure 6 shows an example of mutation.

Fitness function: In order to achieve a more precise fitness function, waiting time as well as makespan is considered. Waiting time focuses on the time in which the task could be executed but is delayed due to scheduling policy. This waiting is either because of precedence of task graph, the way of allocating tasks to processors, or the communication delay between tasks. In other words, no task can be executed unless all the ancestor tasks are executed. If 2 communicatory tasks do not executed on the same processor, then the communication cost has to be spent.

If the related chromosome involve fitnesses of F_1, F_2, …, F_m and each one share a portion of 100% as C_1, C_2, …, C_m then general fitness can be considered as Eq. 1

$$\text{Fitness} = \sum_{i=1}^{m} C_i F_i \qquad (1)$$

In this study 2 fitesses are used in each chromosome, which can be extended without losing generality. The makespan and waiting time are S and W, respectively. According to the importance of S and W in scheduling problem, one can assign a value to C_1 for S and C_2 for

W. For example if share portion of S is 70% and share portion of W is 30% then general fitness can be calculated as Eq. 2.

$$F = 0.7^* S+0.3^* W \quad (2)$$

RESULTS

In this study, the proposed algorithm is compared with a previous algorithm (Jin *et al*., 2008) in which both have considered communication cost.

This has been taken into account since most genetic algorithms have been evaluated on random graphs that have not considered communication cost. We have used G-J, LU benchmarks to evaluate the proposed algorithm. In our implementation the number of processors is assumed to be four. The number of tasks we have chosen for G-J and LU algorithm is shown in Table 1.

The proposed algorithm is implemented on Matlab programming language. The achieved makespan are shown in Fig. 7.

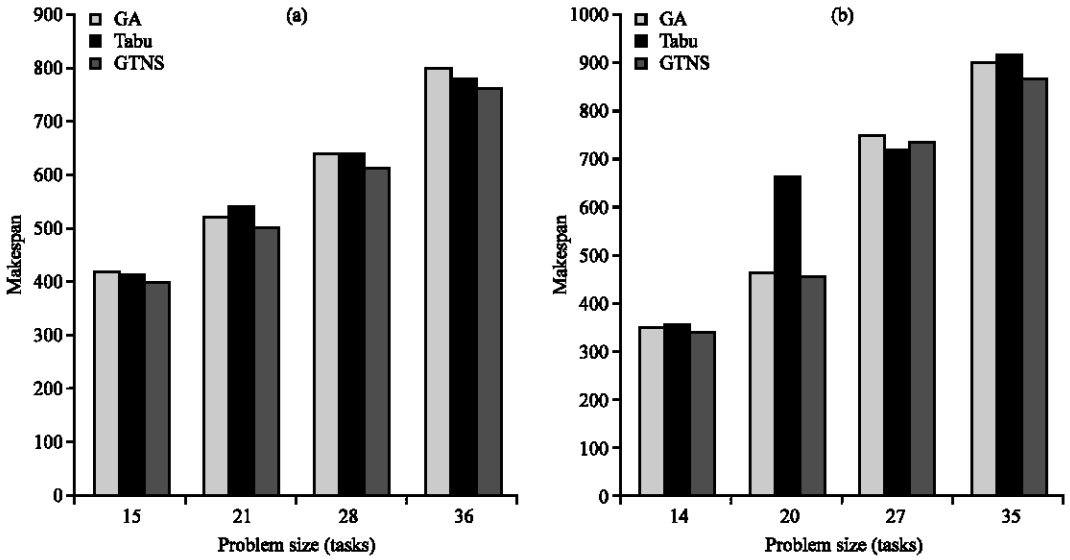

Fig. 7: Makespan for G-J elemination (a) and LU factorization (b), problem for variable task sizes

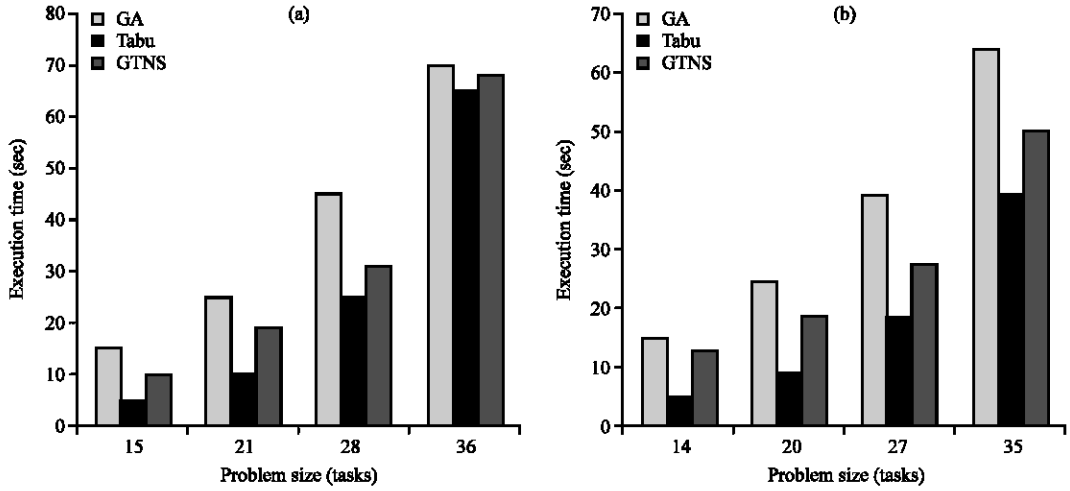

Fig. 8: Execution time for G-J elemination (a) and LU factorization (b), problem for variable task sizes

Table 1: Experimental setup

Problem	No. of tasks	Consumption time	Communication time
Gauss-Jordan elimination	15, 21, 28, 36	40 sec task^{-1}	100 sec
LU factorization	14, 20, 27, 35	10 sec bottom layer task, plus 10 sec for every layer	80 sec

One important point in most scheduling algorithms is the execution time of the algorithm which should be reasonable. Comparative results of execution time in GTNS and the previous algorithms are shown in Fig. 8.

CONCLUSION

Task graph scheduling is an NP-hard problem, therefore nondeterministic approaches e.g., genetic algorithms, tabu list and neighborhood search are applicable to this context. Therefore, in this study a new genetic algorithm (GTNS) based on neighborhood Search and Tabu list is proposed. The GTNS algorithm is capable of achieving an appropriate scheduling while spending less execution time since there should be a balance between solution space and execution time of algorithm. The GTNS algorithm is based on reduction of communication cost between processors and neighborhood search for parents whose fitness are more than 75% of mean current population fitness. Furthermore, the Tabu list is used for avoiding repetition of the parents whose children have been already searched in previous generation.

Finally, for achieving a more precise fitness calculation, an additional parameter except makespan i.e. waiting time, is also used. Experimental result from implementation of the GTNS indicated that better solutions with less execution time are feasible by a combination of Tabu search and genetic algorithm.

REFERENCES

Correa, R., A. Ferreira and P. Rebereyend, 1999. Scheduling multiprocessor tasks with genetic algorithms. IEEE. Trans. Parallel and Distrib. Sys., 10: 825-837.

Dhodhi, M.K. and I. Ahmad, 1995. A Multiprocessor Scheduling Scheme Using Problem-Space Genetic Algorithms. Proceeding of IEEE International Conference on Evolutionary Computation

Jin, S., G. Schiavone and D. Turgut, 2008. A performance study of multiprocessor task scheduling algorithms. J. Supercomput, 43: 77-97

Kwok, Y.K. and I. Ahmad, 1999. Benchmarking and Comparison of the Task Graph Scheduling Algorithms. J. Parallel Distrib. Comput., 59: 381-422.

Parsa, S., S. Lotfi and N. Lotfi, 2007. An Evolutionary Approach to Task Graph Scheduling. Lecture Notes in Computer Science, 4431: 111-119.

Thesen, A., 1998. Design and evaluation of tabu search algorithms for multiprocessor scheduling. J. Huristic, 4: 141-160.

Tian, Y. and N. Sannomiya, 2000. A tabu search with a new neighborhood search technique applied to flow shop schedulig problems. In: Proc. 39th IEEE. Conf. Decision and Control, 5: 4606-4611.

Wu, A.S., H. Yu, Sh. Jin, K. Ch. Lin and G. Schiavone, 2004. An Incremental Genetic Algorithm Approach to Multiprocessor Scheduling. IEEE. Trans. Parallel and Distrib. Sys., 15: 824-834.