Comparison of MOGA with Greedy Algorithms in Soft Real-time Task Scheduling on Heterogeneous Processors with Communication Delay

Nafiseh Sedaghat  
Department of Artificial Intelligence, Islamic Azad University Mashhad Branch  
nf_sedaghat@yahoo.com

Hamid Tabatabaee –Y  
Department of Computer Engineering, Islamic Azad University Qouchan Branch  
hamid.tabatabaee@gmail.com

M .R .Akbarzadeh –T  
Department of Electrical Engineering, Ferdowsi University of Mashhad  
akbarzadeh@ieee.org

Abstract: Scheduling of real-time tasks on a multi-processor system is an NP-hard problem. This paper aims to propose an algorithm based on multi-objective GA (MOGA) for scheduling of static soft real-time tasks on a heterogeneous multi-processor system when the real-world constraints including the precedence relationship between tasks, different arrival time for each task as well as communication delays between the processors are all considered. The objectives of the proposed scheduling algorithm are maximizing system utilization and minimizing total tardiness. Since these objectives are conflicting, the proposed method applies adaptive weight approach (AWA) where some useful information from the current population is utilized to readjust the weights for obtaining a search pressure toward a positive ideal point. In this paper, we also propose two greedy algorithms in which each algorithm aims to optimize a single objective either idle time or communication delay. The performance of the proposed MOGA is compared with the performance of the greedy algorithms on two types of DAGs, sparse and non-sparse. The results demonstrate the high efficiency of the proposed MOGA in solving real-world task scheduling problems.

Keywords: Task Scheduling, Heterogeneous Multi-processor System, Directed Acyclic Task Graph (DAG), MOGA, Greedy Algorithms.

1- Introduction

The multiprocessor task scheduling problem is a generalization of the classical scheduling problem that allows tasks (jobs) to be processed on more than one processor at a time. The utilization of parallel processing systems these days, in a vast variety of applications, is the result of numerous breakthroughs over the last two decades. The development of parallel and distributed systems has lead to there use in several applications including information processing, fluid flow, weather modelling, database systems, real-time high-speed simulation of dynamical systems, and image processing [1]. Task scheduling is the most important of these issues because inappropriate scheduling of tasks can fail to exploit the true potential of a distributed system and can offset the gains from parallelization due to excessive communication overhead or under-utilization of resources. Thus it falls to one’s scheduling strategy to produce schedules that efficiently utilize the resources of the distributed system and minimize the total execution time [2]. Real-time systems are characterized by computational activities with timing constraints and classified into two categories: hard real-time system and soft real-time system. In hard real-time system, the violation of timing constraints of a certain task should not be acceptable. The consequences of not executing a task before its deadline may lead to catastrophic consequences in certain environments i.e., in patient monitoring systems, nuclear plant control, etc. The goal of the scheduling algorithms in hard-real-time system is to meet all tasks’ deadlines, in other words, to keep the feasibility of scheduling through admission control. On the other hand, in the soft real time system (e.g. telephone switching system, image processing, etc.), in
which usefulness of results produced by a task decreases over time after the deadline expires without causing any damage to the controlled environment [3].

Most of research is concerned with the minimization of a single criterion—the makespan. However, in practice, many industries such as aircraft, electronics, semiconductors manufacturing, etc., have tradeoffs in their scheduling problems where multiple objectives need to be considered in order to optimize the overall performance of the system. Obviously, the multi-objective scheduling problems are more complex than the scheduling problems with one criterion, and it is hard to find a compromise solution because the objectives are often inconsistent, conflicting or even contradictory [4].

The multiprocessor task scheduling problem considered in this paper is based on the deterministic model, which is the execution time of tasks and the data communication time between tasks that are assigned; and, the directed acyclic task graph (DAG) that represents the precedence relations of the tasks of a parallel processing system well known as the NP-complete problem. We assume that the parallel processor system is non-preemptive; that is, task scheduling and allocation onto heterogeneous multiprocessor systems where as each processor completes the current task before the new one starts its execution, and we assume that all of tasks aren’t available at time 0 and their arrival time can be different from others. However, for a more realistic problem, we may assume that communication delays between processors are possible. When two communicating tasks are mapped to the same processor the communication delay becomes zero because the data transfer is effective; but, when mapped to different processors the communication delay is represented. The objectives of the proposed scheduling algorithm are to minimizing total tardiness and maximizing system utilization. For fitness function of GA, this paper combines Adaptive Weight Approach (AWA) [5] that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point.

The rest of the paper is organized as follows: In Section 2 we examine some related works and in section 3 we present our methods and in section 4 and 5, we discuss about the efficiency of algorithms and future works, respectively.

2- Related Work

In the case of Parallel machine scheduling, there are many literatures surrounding the multi-objective problem. The use of Holland’s genetic algorithms (GAs) in scheduling, which apply evolutionary strategies to allow for the fast exploration of the search space of schedules, allows good solutions to be found quickly and for the scheduler to be applied to more general problems. E. Kim et al. [7], consider a deterministic scheduling problem where multiple jobs with s-precedence relations are processed on multiple identical parallel machines. The objective is to minimize the total completion time. The s-precedence relation between two jobs $i$ and $j$ represents the situation where job $j$ is constrained from processing until job $i$ starts processing, which is different from the standard definition of a precedence relation where $j$ cannot start until $i$ completes. X. Kong et al. [8], incorporate a miscellaneous population-based search technique, Particle Swarm Optimization (PSO), with list scheduling and develop an alternative PSO algorithm for multiprocessor tasks scheduling. The framework of the hybrid PSO algorithm for the multiprocessor scheduling is developed according to the permutation-based solution representation to find optimal task-machine pairs. The merit of the hybrid method is to keep improving the potential solution continuously through learning experience from the searching knowledge of one particle itself or the best of all particles in the swarm in the interest of minimizing the scheduling length. Hwang et al. [1], addresses the challenge of multiprocessor task scheduling parallel programs, represented as directed acyclic task graph (DAG), for execution on multiprocessors with communication costs. They use genetic algorithm for solving this problem and design the new encoding mechanism with a multi-functional chromosome that uses the priority representation—the so-called priority-based multi-chromosome (PMC). Yoo [5], proposes a new scheduling algorithm for soft real-time tasks using MOGA on multiprocessors system. It is assumed that tasks have
precedence relations among them and are executed on homogeneous multiprocessor environment. The objective of the proposed scheduling algorithm is to minimize the total tardiness and total number of processors used. Yoo et al. [3], propose a new scheduling algorithm for real-time tasks using multi-objective hybrid genetic algorithm (MOHGA) on heterogeneous multiprocessor environment. In solution algorithms, the genetic algorithm (GA) and the simulated annealing (SA) are cooperatively used. In this method, the convergence of GA is improved by introducing the probability of SA as the criterion for acceptance of new trial solution. S. Zhou et al. (2006) [9], propose a new genetic-anneal algorithm (GAA) with task duplication. It uses a simulated annealing scheme to alleviate the selection pressure of the genetic algorithm, and enhance the global search of the genetic algorithm.

3- Proposed Method

In this paper we examine Performance of genetic algorithm and compare its performance with two greedy algorithms. The criteria has selected, are system utilization and total tardiness. System utilization is an important factor in multiprocessor systems; because we want to maximize it always. On other side, in the soft real time systems, tardiness is another important factor; because we want to perform tasks, as soon as possible; for this reason, we want to reduce total tardiness of tasks. In the follows, we present structure of genetic algorithm used and also two greedy algorithms.

3-1- Multi-objective Genetic Algorithm

A GA is a meta-heuristic search technique which allows for large solution spaces to be partially searched in polynomial time, by applying evolutionary techniques from nature [6]. GAs use historical information to exploit the best solutions from previous searches, known as generations, along with random mutations to explore new regions of the solution space. In general a GA repeats three steps (selection, crossover, and random mutations) as shown by the pseudo code in Fig. 1. Selection according to fitness is a source of exploitation, and crossover and mutations promote exploration. A generation of a GA contains a population of individuals, each of which correspond to a possible solution from the search space [2]. Genetic algorithm requires a coding scheme that can represent all legal solutions to the optimization problem. Any possible solution is uniquely represented by a particular chromosome (or string), and chromosomes are manipulated in various ways by applying three genetic operators until the termination condition is met. In order for this manipulation to proceed in the correct direction, a method of prescribing a quality value (or fitness) to each solution string, which is called fitness function, is also required. In this section, we present genetic algorithm in details.

![Fig. 1 Pseudo code for genetic algorithm](image)

3-1-1 CODING SCHEME

Normal binary encoding does not work very well for this problem as the strings may become too long in order to incorporate all the information that is needed [10]. Therefore, the strings are encoded using decimal numbers. Each individual in the population represents a possible schedule. Fig. 2 shows the encoding used. Each character is a mapping between a task and processor. Each character contains the unique identification number of a task, with \( S \) being used to delimit different processor queues, where \( P_i \) is processor. Thus the number of characters is \( N + M - 1 \), where \( N \) is the number of tasks in the batch, and \( M \) is the number of processors [2].

![Fig. 2 An individual in GA](image)
3-1-2 Fitness Function

A fitness function attaches a value to each individual in the population, which indicates the goodness of the schedule. Most of literatures in this field consider total tardiness and makespan time as objectives. However, in this paper we consider utilization of multiprocessor system and total tardiness of tasks, as objectives. Utilization ratio for multiprocessor systems is an important factor, because we want to use total capacity of processors. Also, total tardiness is an important factor for real time scheduling; because we want to complete tasks before their deadlines as much as possible. For unifying these objectives, we minimize the inverse of utilization and total tardiness. The total tardiness, defined as

$$\tau = \sum_{i=1}^{n} \max\{0,(y_i - d_i)\},$$

where $y_i$ is the completion time and $d_i$ is the deadline of task $T_i$, respectively.

The utilization of system, defined as,

$$\phi = \frac{\sum_{i=1}^{n} U_i}{N},$$

where $U_i$ is the utilization of each processor and $N$ is the number of processors. The utilization of each processor, is calculated as,

$$U_j = \frac{\sum_{i=1}^{k} C_{i}}{\max_{i \in \{1, k\}} y_{i}},$$

where $C_i$ is the computation time of task $i$th on processor $j$th. $k$ is the number of tasks allocate on processor $j$th. $y_i$ is the completion time of task $i$th on processor $j$th.

We combine these conflicting objectives with adaptive weight approach (AWA) [5] that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point.

3-1-4 Stopping Conditions

The GA will evolve the population until one or more stopping conditions are met. The best individual is selected after each generation and if it doesn’t improve for 100 generations, the GA stops evolving. The maximum number of generations is set at 300 because the quality of the schedules returned with more than that number does not justify the increased computation cost.

3-2 Greedy Algorithms

Algorithms for optimization problems typically go through a sequence of steps, with a set of choices at each step. A greedy algorithm always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution. Greedy algorithms do not always yield optimal solutions, but for many problems they do[12]. In this Section we present two greedy algorithms used for comparison. In the follows, we describe each algorithm with details.

The first algorithm is based on minimizing idle time of each processor; In other words, it maximizes the utilization of each processor. With minimizing the idle time of processors, span is decreased and in that follows, tardiness that is related to span time, is decreased. For this purpose, we allocate each task to processor that has more idle time. Second algorithm is based on minimizing communication delay between processors. In order to achieve this goal, we identify tasks that communication cost between them, is large; then allocate them to same processor. In allocating tasks to processors, we try to achieve best load balancing state in system. Therefore, large communication delay in multiprocessor system is deleted and span time is decreased; in that follows, tardiness is decreased. Since we have considered the load balancing in system, utilization of system is increased.
1. Simulation and Experimental Results

The scheduling algorithms described in Sect. 2 has been implemented and applied to simulated data. Two different experiments have been performed to demonstrate the effectiveness of the genetic algorithm to scheduling problems with sparse and non-sparse DAG. We compare GA scheduler to two greedy schedulers, and evaluate the results using two different but related metrics, total tardiness and utilization. Total tardiness is the sum of differences between end time and deadline time of tasks. Efficiency is the percentage of the time that processors actually spend processing rather than communicating or idling.

A representative set of heterogeneous computing task benchmarks does not exist as yet. In order to evaluate our algorithms, we consider two kinds of DAGs: Sparse DAG and non sparse DAG. Each DAG has 25 tasks and communication cost between tasks, has generated randomly based on normal distribution. We experiment these DAGs with 5 heterogeneous processors. For generating arrival time and computational time for tasks,
we use Poisson and normal distribution, respectively. Deadline times have generated based on formulation as [13] with lax=10. Fig. 3 and Fig. 4 show sparse and non-sparse DAGs used in experiments. Characteristics of GA are: population size: 40, crossover probability: 0.8, mutation probability: 0.3. Fig. 5-8 show experimental results. In these figures, horizontal axis is arrival rate of tasks and vertical axis in figure 5 and 7 is total tardiness and in figures 6 and 8 is utilization percentage of multiprocessor system.

Fig. 5 and Fig. 6, show the sparse DAG results. As Fig. 5 shows, GA has lowest total tardiness and also as in Fig. 6 shows, best utilization. First greedy algorithm is better than second algorithm in total tardiness but its utilization is lowest. Fig. 7 and Fig. 8, show the non-sparse DAG results. As Fig. 6 shows, GA has lowest total tardiness and also as in Fig. 8 shows, best utilization. In non-sparse DAG, second greedy algorithm is better than first algorithm in total tardiness (they are close together). In utilization criterion, second greedy algorithm is better than first algorithm because in this algorithm we reduce communication cost between processors and it leads the busy time of processors are increased and in follows utilization of each processor and finally utilization of all processors increased. In sparse and non-sparse DAG, GA has best performance, but it is noticeable that the computation time of GA is more than greedy algorithms. Computation time for all of algorithms, for sparse DAG is less than Non-sparse DAG; because when DAG is non-sparse, their numbers of edges are more than sparse DAG and this leads to more computation time for scheduling.

2. Conclusion and Future works

We have addressed the problem of determining the optimal scheduling of soft real-time applications in heterogeneous multiprocessor systems such that system Utilization maximized and total tardiness is minimized. This requires the simultaneous optimization of the two criteria. For fitness function of GA, this paper combines Adaptive Weight Approach (AWA) that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point. We also proposed two greedy algorithms that each of them, focuses to optimize one objective that effects on system utilization and total tardiness, directly.

In our simulation experiments, two kinds of DAGs are assumed (sparse DAG and non-sparse DAG) and their parameters generated, randomly. We compare the results of GA with greedy algorithms. The experimental results show that GA performed as well as or better than other algorithms but computation time for GA is more than greedy algorithms. In the future, we plan to consider the dynamic task scheduling problem, which network topology and processors computational power and communicational links between processors change with time; moreover, preemption and duplication is allowed.

References


