A Novel Method for Task Scheduling in Distributed Systems Using Memetic

M. Jahanshahi\(^1\), M. Gholipour\(^2\), M. S. Kordafshari\(^2\), A. M. Rahmani\(^3\)

\(^1\) Computer Engineering Department, Islamic Azad University, Central Tehran Branch, Tehran, Iran
\(^2\) Computer Engineering Department, Islamic Azad University, Qazvin Branch, Qazvin, Iran
\(^3\) Computer Engineering Department, Islamic Azad University, Science and Research Branch, Tehran, Iran
mjahanshahi@iauctb.ac.ir gholipour@qazviniau.ac.ir kordafshari@qazviniau.ac.ir rahmani@sr.iau.ac.ir

Abstract—Tasks scheduling problem is a key factor for a distributed system in order to achieve better efficiency. The problem of tasks scheduling in a distributed system can be stated as allocating tasks to the processor of each computer. The objective of this problem is minimizing makespan and communication cost, while maximizing CPU utilization. Scheduling problem is known as being NP-complete. Hence, many genetic algorithms have been proposed to search optimal solutions from entire solution space. However, these existing approaches are going to scan the entire solution space without consideration to techniques that can reduce the complexity of the optimization. In other words, the main shortcoming of these approaches is to spend much time doing scheduling, and hence, needing exhaustive time. Therefore, in this paper we use memetic algorithm to cope with this shortcoming. We apply Tabu search as local search in our proposed memetic algorithm. Extended simulation results demonstrate that the proposed method outperforms the existent GA-based method in terms of communication cost, CPU utilization and makespan.

Key word-Task scheduling; Memetic algorithm; Tabu search.

I. Introduction

With many achievements, such as technology, computer architectures, and software packages, distributed systems are used in a variety of applications. The problem of task scheduling in these systems has received a large amount of attention recently. Task scheduling in a distributed system can be stated as allocating tasks to the processors of each computer such that the optimum performance is obtained. The aim of task scheduling is minimizing makespan (job completion time) and communication cost, while maximizing CPU utilization. This problem is known as being NP-complete [20].

There are two categories for task scheduling; static and dynamic. In dynamic scheduling, schedules create during run time and no knowledge of task is in hand until it arrives. While in static scheduling, schedules are created before run time and can not change. Similarly, tasks must all be known in advance. In other words a static task scheduling algorithm schedules a set of tasks with known processing and communication characteristics on processors to optimize some performance metric, such as makespan, communication cost and CPU utilization. In this paper we focus on static scheduling.

Several methods have been proposed to solve scheduling problem. The proposed methods can be generally classified into three categories: Graph-theory-based approaches [21], mathematical models-based methods [22] and heuristic techniques [23-26]. As mentioned above the scheduling problem has been known to be NP-complete. Therefore using heuristic Techniques can solve this problem more efficiently. Three most well-known heuristics are the iterative improvement algorithms [31], the probabilistic optimization algorithms, and the constructive heuristics. In the probabilistic optimization group, GA-based methods [27-31, 2, 3] are considerable. The main distinction among them is chromosomal representation used for a schedule. However, these approaches scan the entire solution space without consideration to techniques that can reduce the complexity of the optimization. In other words their main shortcoming is to spend much time doing scheduling. This shortcoming of GA-based methods can be reduced by combing GA with another optimization technique. Hence this paper proposes a new algorithm by using memetic algorithm to cope with this shortcoming. We apply Tabu search as local search in memetic. Simulation results demonstrate that our proposed method outperforms the existent GA-based method in term of communication cost, CPU utilization and Make span.

The rest of this paper is organized as follows: In Section 2 used method as local search in the proposed memetic algorithm is presented. The proposed method comes in Section 3. Simulation results are given in Section 4. Section 5 concludes the paper.

II. Used Method As Local Search In The Proposed Memetic Algorithm

In this section we describe the method which has been used as local search in our proposed memetic algorithm:

The basic concept of Tabu Search as described in [4-11] is a meta-heuristic superimposed on another heuristic. The overall approach is to avoid entrapment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited (hence "Tabu"). Tabu search begins by marching to a local minima. To avoid retracing the steps used the method records recent moves in one or more Tabu lists. The original intent of the list was not to prevent a previous move from being repeated, but rather to insure it was not reversed.

Genetic algorithms (GA) are search methods that take their inspiration from natural selection and survival of
the fittest in the biological world [12-14]. GAs differ from more traditional optimization techniques in that they involve a search from a "population" of solutions, not from a single point. Each iteration of an GA involves a competitive selection that weeds out poor solutions. The solutions with high "fitness" are "recombined" with other solutions by swapping parts of a solution with another (crossover). After selection and crossover, there is a new population full of individuals. Some are directly copied, and others are produced by crossover. In order to ensure that the individuals are not all exactly the same, you allow for a small chance of mutation. Solutions are also "mutated" by making a small change to a single element of the solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the space for which good solutions have already been seen. Mutation is, however, vital to ensuring genetic diversity within the population.

Memetic algorithms are a form of genetic algorithm that apply a local search process to refine solutions to hard problems [23-26]. In this paper, we apply Tabu search as local search in memetic.

III. Proposed Method

In our model there are finite numbers of tasks, each having a task number and a execution time and placed in a task pool from which tasks are assigned to processors. Figure 1 shows the proposed chromosome. In this chromosome, tasks 1,2,3,4,5,6,7 and 8 are assigned to processors 2,1,2,3,3,4,2 and 1 respectively. Both tasks 1 and 3 are assigned to processor 2 but first task 1 is executed then 3.

![Figure 1. An example of proposed chromosome](image)

Before we describe the proposed method it is necessary to state some definitions which have been stated in [1] as follows:

- \( T = \{ t_1, t_2, t_3, ..., t_n \} \) is the set of tasks to execute.
- \( P = \{ p_1, p_2, p_3, ..., p_m \} \) is a set of processors in the distributed system. Each processor can only execute one task at each moment, a processor completes current task before executing a new one, and a task can not be moved to another processor during execution.
- \( R \) is an \( m \times m \) matrix, where the element \( r_{uv} \) \( 1 \leq u, v \leq m \) of \( R \), is the communication delay rate between \( p_u \) and \( p_v \).
- \( H \) is an \( m \times m \) matrix, where the element \( h_{uv} \) \( 1 \leq u, v \leq m \) of \( H \), is the time required to transmit a unit of data from \( p_u \) to \( p_v \). It is obvious that \( h_{uu} = 0 \) and \( r_{uv} = 0 \).
- \( A \) is an \( n \times m \) matrix, where the element \( a_{ij} \) \( 1 \leq i \leq n \), \( 1 \leq j \leq m \) of \( A \), is the execution time of task \( t_i \) on processor \( p_j \).
- \( D \) is a linear matrix, where the element \( d_i \) \( 1 \leq i \leq n \) of \( D \), is the data volume for task \( t_i \) to be transmitted, when task \( t_i \) is to be executed on a remote processor.
- \( F \) is a linear matrix, where the element \( f_i \) \( 1 \leq i \leq n \) of \( F \), is the target processor that is selected for task \( t_i \) to be executed on.
- \( C \) is a linear matrix, where the element \( c_i \) \( 1 \leq i \leq n \) of \( C \), is the processor that the task \( t_i \) is worked on.
- The processor load for each processor is stated as follows:

\[
\text{Load}(p_i) = \sum_{j=1}^{m} a_{j,i} + \sum_{k=1}^{m} d_{k,j}
\]

- The makespan of a schedule is the maximal finishing time of all processes or maximum load.

\[
\text{makespan}(T) = \max \{ \text{Load}(p_i) \} \\
\forall 1 \leq i \leq \text{Number of Processors}
\]

- Communication cost (CC) is computed as follows:

\[
\text{CC}(T) = \sum_{i=1}^{\text{number of new processes}} (r_{c_{ij}} + h_{c_{ij}} \times d_i)
\]

- The Processor utilization for each processor and the average of processors utilization are also computed as follows:

\[
U(p_i) = \frac{\text{Load}(p_i)}{\text{max span}}
\]

\[
\text{Ave}U = (\frac{1}{\text{Number Of Processors}}) \left( \sum_{i=1}^{\text{No of processors}} U(p_i) \right)
\]

- Number of Acceptable Processor Queues (NoAPQ): We must define thresholds for light and heavy load on processors. If the tasks completion time of a processor is within the light and heavy thresholds, this processor queue will be acceptable. If it is above the heavy threshold or below the light-threshold, then it is unacceptable. But what is important is average of number of acceptable processors queues, which is achievable by:

\[
\text{AveNoAPQ} = \text{NoAPQ}/\text{Number Of Processors}
\]
A Queue associated with every processor, shows the tasks that processor has to execute. The execution order of tasks on each processor is based on queues. Finally the fitness of the chromosome (Schedule T) can be computed as follows:

\[
    \text{fitness}(T) = (\gamma \times \text{Ave}U) \times (\theta \times \text{AveNoAPQ}) \\
    \quad \times (\alpha \times \text{max span}(T)) \times (\beta \times \text{CC}(T))
\]

Which \(0 < \alpha, \beta, \gamma, \theta \leq 1\) are control parameters to control effect of each part according to special cases and their default value is one. This equation shows that a fitter solution (Schedule) has less makespan, less communication cost, higher processor utilization and higher Average number of acceptable processor queues.

Now we describe the proposed method in details. Figure 2 depicts the proposed method. We apply Tabu search as local search in our proposed memetic algorithm.

In this section, we describe the method which is used as local search in our memetic algorithm.

**Tabu search based method**

In each step of iteration a task is selected (index of the chromosome). Now two situations may occur:

- If the selected task in not in Tabu list then a new processor is assigned to it. Hence a new chromosome is created. If the created chromosome is better than the older one then it is added to population. After that the number of task (index of the chromosome) is added to the Tabu list.
- If the selected task is in the Tabu list then the above algorithm continues at most 5 times.

In our simulations length of the Tabu list is set to 3.

**IV. Simulation Results**

In this section, we evaluate the proposed method compared with [1] regarding communication cost, CPU utilization and Makespan.

**First experiment:**

In this experiment we try to increase number of tasks and compute average of all CPU utilization, communication cost and Makespan. Figures 3 through 5 depict simulation results. As it is seen in Figure 3 average of all CPU utilization in GA-based method ( reference [1] ) is less than our method.

**Second experiment:**

Similarly in this experiment the aim is computing the makespan, communication cost and CPU utilization of all processors. It can be seen in Figure 6 that makespan in GA-Based method is more than our propose method. That means if we increase population size, then the trend of makespan in GA-based method will be increased. In the other hand it can be obtain GA-based method has not scalability.
Third experiment:
Objective of this experiment is to compute the above metrics when we increase the number of generations. Figure 9 through 11 shows the simulation results. Figure 9 shows the average of CPU utilization in the methods. It can be seen with increasing the number of generation CPU utilization in GA-based method is less than our method. In another words GA-based method has not scalability.

V. Conclusion
In this paper we used the memetic algorithm for task scheduling. We applied Tabu search as local search in memetic. Extended simulation results demonstrate that our proposed method outperforms the existent GA-based method in terms of communication cost, CPU utilization and makespan.

Reference
[2] S. H. Woo, S. Yang, S. Kim and T. Han, "Task scheduling in distributed computing systems with a genetic algorithm", hpcasia, p. 301, High-


