Engineering Applications of Artificial Intelligence 61 (2017) 35–46

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems

Mehdi Akbari[a], Hassan Rashidi[b,*], Sasan H. Alizadeh[c]

[a] Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[b] Department of Mathematics and Computer Science, Allameh Tabataba'i University, Tehran, Iran
[c] Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

ARTICLE INFO

Keywords:
Task scheduling
Heterogeneous systems
Genetic algorithm
Meta-heuristic algorithm

ABSTRACT

One of the important problems in heterogeneous computing systems is task scheduling. The task scheduling problem intends to assigns tasks to a number of processors in a manner that will optimize the overall performance of the system, i.e. minimizing execution time or maximizing parallelization in assigning the tasks to the processors. The task scheduling problem is an NP-complete and this is why the algorithms applied to this problem are heuristic or meta-heuristic by which we could reach a relatively optimal solution. This paper presents a genetic-based algorithm as a meta-heuristic method to address static task scheduling for processors in heterogeneous computing systems. The algorithm improves the performance of genetic algorithm through significant changes in its genetic functions and introduction of new operators that guarantee sample variety and consistent coverage of the whole space. Moreover, the random initial population has been replaced with some initial populations with relatively optimized solutions to lower repetitions in the genetic algorithm. The results of running this algorithm on the graphs of real-world applications and random graphs in heterogeneous computing systems with a wide range of characteristics, indicated significant improvements of efficiency of the proposed algorithm compared with other task scheduling algorithms.

1. Introduction

Heterogeneous distributed systems contain a group of varied-pace processors connected to each other through a fast network. They are employed for parallel execution of distributed applications. Task-scheduled algorithms help the processors to manage task completion via assigning tasks to available processors, thus enhancing the efficiency. They have been widely adopted in industrial and manufacturing applications (Savino et al., 2015, 2014a, 2014b, 2010). Task scheduling is an NP-complete (Ullman, 1975) and thus meta-heuristic algorithms are used to reach a relatively optimized solution, this means that to reach a solution close to the optimum solution. One of the main challenges faced by meta-heuristic algorithms such as genetic algorithms is the frequency of their repetitions for reaching relatively optimized solutions and also conflicting local optimums. Efficient genetic algorithm for task scheduling (EGA-TS) that is presenting in this paper tackles these issues via creating a new initial population function and adding improved operators to the basic genetic algorithm.

The main objective of the task scheduling algorithms is to achieve a balance between parallelization increase and communication costs decrease. EGA-TS primarily intends to minimize total execution time, assigning tasks to processors for a maximum of parallelization while minimizing communications costs in proportion to task execution costs of the processors. Lowering the repetition frequency and avoiding local optimums are also intended by this algorithm, this is achieved through minimizing repeated or similar schedules in the population. To guarantee sample variety and consistent coverage of the whole space, some significant changes are made in genetic functions and new operators are introduced. The major contributions of this study are listed below:

- In most of metaheuristic-based scheduling algorithms, there is a stage for selection and assignment of tasks to processors (Ahmad et al., 2016; Gogos et al., 2016; Kumar and Vidyarthi, 2016; Wang et al., 2016; Zhang et al., 2015). However, a new method has been developed in the proposed algorithm to display schedules in which each task is displayed in conjunction with the processor that was assigned in the stage of generating initial populations.
- In standard GA, initial population is generated randomly. Random treatment of initial population selection leaves us with unsuitable

* Corresponding author.
E-mail addresses: mehdi_akbari@hotmail.com, mehdi_akbari@pco.iaun.ac.ir (M. Akbari), Hrashi@atu.ac.ir (H. Rashidi), sasan.h.alizadeh@qiau.ac.ir (S.H. Alizadeh).

http://dx.doi.org/10.1016/j.engappai.2017.02.013
Received 24 March 2016; Received in revised form 24 February 2017; Accepted 26 February 2017
Available online 01 March 2017

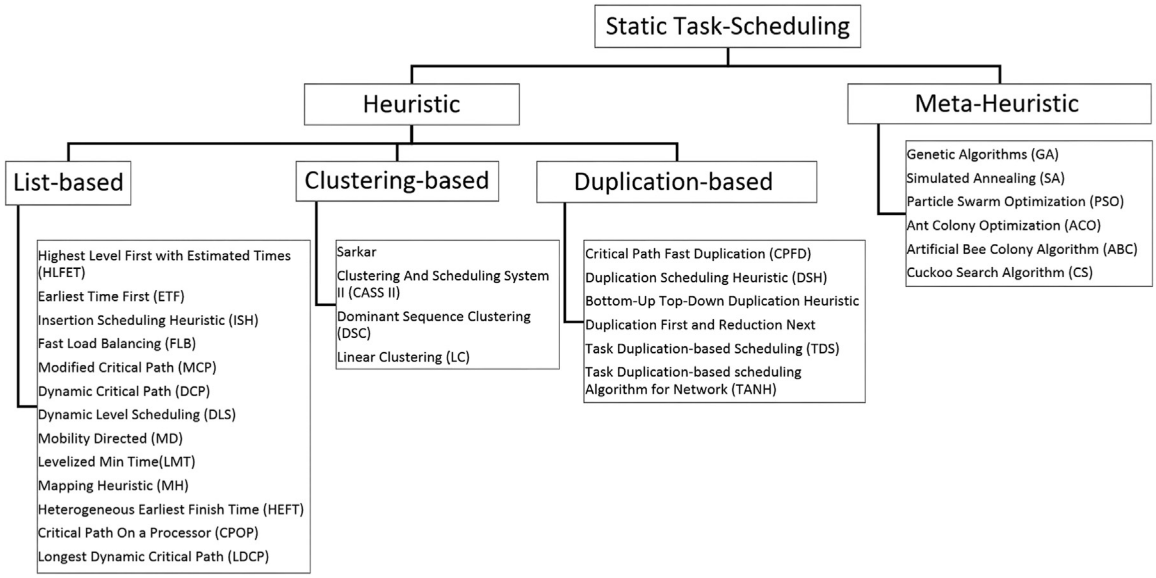

Fig. 1. Classification of static task scheduling algorithms at compile time.

search space. In suitable search space, the samples are close to the optimal solution. Therefore we should generate initial populations that allow us to reach relatively optimized solutions in reasonable runtime.

- The proposed algorithm presents the new operator which is called Inversion. This operator promotes diversity among the samples that will be produced in each generation. By using this new operator the number of repeated schedules in generated population is minimized.

The remaining parts of this paper is structured as follows. Section 2 presents the related works and then we proceed to explain application task graphs along task scheduling model in Section 3. Section 4 is dedicated to the improved genetic algorithm and drawing comparisons with some criteria. In Section 5 we evaluate the time and space complexities. Section 6 presents the results of the comparisons done and the simulations made. Conclusions are presented in the last section.

2. Related works

Static task scheduling algorithms are categorized as heuristic and meta-heuristic (Bansal et al., 2003; Kwok and Ahmad, 1996; Manudhane and Wadhe, 2013; Topcuoglu et al., 2002). Heuristic algorithms are in turn divided into list-based, clustering and duplication methods (Topcuoglu et al., 2002).

In list-based heuristics, a priority is assigned to each task and the tasks are listed in order of their preferences. In these kinds of heuristics, the task selection for processing is done in order of preference and a task with a higher priority is assigned to processor earlier. Heterogeneous earliest finish time (HEFT) and critical path on a processor (CPOP) are the most important examples of list based heuristics (Topcuoglu et al., 2002).

The HEFT algorithm is designed for a given number of heterogeneous processors and includes two stages: at the first stage, task scheduling priority is calculated and at the second stage, processor selection stage, tasks are analyzed in order of priority and assigned to a processor providing the shortest finish time. In this algorithm, priority is determined by a pair of parameters known as *tlevel* and *blevel*. In tasks graph, *tlevel* of each node denotes the longest path weight from the start node to the desired node. In another aspect, *tlevel* of each node represents the nearest possible start time of that node and *blevel* of each node shows the longest pass weight of that node to the exit node. If this algorithm uses *tlevel* parameter to determine the priorities it is called HEFT-T and if *blevel* parameter is applied it is called HEFT-B.

In CPOP algorithm, the total sum of *blevel* and *tlevel* of each node is regarded as task priority and then tasks are selected in order of priority and if placed on critical path, are allocated to the processor which minimizes total task calculation costs on critical path and if not, they are assigned to the processor which makes the nearest complete time possible for it. Critical path refers to the path from the input node to the output node that has the highest total value of calculation cost and communication costs of edges (Kwok and Ahmad, 1996). Therefore, an effective scheduling list-based algorithm requires an appropriate scheduling of tasks placed on critical path (Daoud and Kharma, 2011). The length of critical path is equal to *blevel* and *tlevel* sum of input task. Thus, each task whose *tlevel* and *blevel* equals to the sum of *blevel* and *tlevel* of input task lies on critical path (Topcuoglu et al., 2002).

Duplication heuristic method reduces runtime by applying task duplication to different processors (Lin et al., 2013). In this method, communication time between processors is reduced by executing tasks on more than one processor. It avoids the transmission of the results from a specific task to the next one (because both are executed on a single processor) and therefore it reduces communication costs. In parallel and distributed systems, clustering heuristic method is an appropriate solution to reduce graph communication delay. Communication delay is reduced in this method since tasks which have high relations with each other are put together in a cluster and are assigned to a single processor (Mishra et al., 2012). Fig. 1 shows this classification together with the proposed algorithms samples for each category (Topcuoglu et al., 2002).

Heuristic based algorithms are not distinctly possible to produce constant results for a wide range of problems, mostly when the complexity of the task scheduling problem increases. Contrary to the meta-heuristic algorithms, a heuristic algorithm uses a combinatory process in the search for solutions, which is less efficient and generates much higher computational cost than the meta-heuristic algorithms. For this reason, balance between makespan and speed of convergence is required.

The Meta-heuristic algorithms usually make up an important solution for global optimization problems. Totally heuristic means to find and discover by error and trial. "Meta-heuristic" can be applied to strategies with a higher level that have modified and guided heuristic