Expert Systems With Applications 60 (2016) 234–248

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems

Mehdi Akbari[a,*], Hassan Rashidi[b]

[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

A R T I C L E I N F O

Article history:
Received 20 January 2016
Revised 7 May 2016
Accepted 8 May 2016
Available online 10 May 2016

Keywords:
Task scheduling
Heterogeneous systems
Cuckoo optimization algorithm
Meta-heuristic algorithm
Makespan

A B S T R A C T

To handle scheduling of tasks on heterogeneous systems, an algorithm is proposed to reduce execution time while allowing for maximum parallelization. The algorithm is based on multi-objective scheduling cuckoo optimization algorithm (MOSCOA). In this algorithm, each cuckoo represents a scheduling solution in which the ordering of tasks and processors allocated to them are considered. In addition, the operators of cuckoo optimization algorithm means laying and immigration are defined so that it is usable for scheduling scenario of the directed acyclic graph of the problem. This algorithm adapts cuckoo optimization algorithm operators to create proper scheduling in each stage. This ensures avoiding local optima while allowing for global search within the problem space for accelerating the finding of a global optimum and delivering a relatively optimized scheduling with the least number of repetitions. Moving toward global optima is done through a target immigration operator in this algorithm and schedules in each repetition are pushed toward optimized schedules to secure global optima. The results of MOSCOA implementation on a large number of random graphs and real-world application graphs with a wide range characteristics show MOSCOA superiority over the previous task scheduling algorithms.

1. Introduction

To make balance between maximizing parallelization and minimizing communication delay, heterogeneous systems face a critical problem in task scheduling. This problem is a max-min problem with parallelism. The parallelization in this problem concerned with simultaneous task allocation to a higher number of processors, which increases communication cost especially when communication delay is high. Meanwhile, task execution on some specific processors reduces efficiency and therefore, the main objective of the task scheduling algorithms is to make a balance between parallelization increase and communication costs decrease.

The static task scheduling problem in which this is no changes in situation, is an NP-hard problem (Ullman, 1975). To solve this problem, the proposed algorithms employ heuristic methods to achieve a relatively optimal solution. The main quandary of heuristic methods is to get stuck in local optima. To scape of this quandary, some meta-heuristic methods are used. These methods utilize detection functions in search strategies (Kwok & Ahmad, 2005; Wu, Shu, & Gu, 1997) that results in reducing the possibility of engaging with local optima as well as decreasing search space.

One of the difficulties of meta-heuristic methods such as genetic algorithms is to determine the number of repetitions of these algorithms to achieve a relatively optimal solution. Cuckoo optimization algorithm achieves the solution at a lower number of repetitions compared with some meta-heuristic algorithms in this regard (Rajabioun, 2011). In addition, the number of repetitions for achieving an appropriate solution does not affect execution time used for next times since the program is compiled once and ran for several times. Faster convergence and local search capability along with global search are among the factors which justify the application of cuckoo optimization algorithm.

The main objectives of MOSCOA are to minimize total execution time and processor allocation to tasks so that maximum parallelization in simultaneous performance of processors is achieved while the ratio of communication costs between processors to task execution costs is minimized. In MOSCOA, first new schedule is created in laying stage with regard to the radius of defined modifications for each cuckoo. Then these schedules immigrate toward optimal schedules in generated population. In this algorithm, creating new schedules is done in a way that precedence relations between tasks are maintained in order to have a correct schedule. The number of repeated schedules or those close to each other in

* Corresponding author. Fax: +983142291016.
E-mail addresses: mehdi_akbari@hotmail.com, mehdi_akbari@pco.iaun.ac.ir (M. Akbari), Hrashi@atu.ac.ir (H. Rashidi).

http://dx.doi.org/10.1016/j.eswa.2016.05.014
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

generated population is minimized at each stage to maintain the variety of samples. The three 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, Liew, Munir, Ang, & Khan, 2016; Gogos, et al., 2016; N. Kumar & Vidyarthi, 2016; Wang, Wang, Liu, & Guo, 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 randomly in the stage of generating initial populations.
- Most algorithms that employ CS as the metaheuristic-based method for handling scheduling problem use Le'vy flights for conducting searches within the problem space (Mandal & Acharyya, 2015; Navimipour & Milani, 2015; H. Wang et al., 2016), while in the proposed algorithm immigration and laying operators are tasked with this.
- There is often a repair stage in some algorithms that employ laying and immigration operators (Shahdi-Pashaki, Teymourian, Kayvanfar, Komaki, & Sajadi, 2015). This stage is tasked with verifying the integrity of subsequences of the schedules which have been created by operators and repairing them in case of need. These operators are defined in MOSCOA to skip repair stage.

The rest of this paper is organized as follows. In Section 2, the related works on the scheduling algorithms on heterogeneous systems is reviewed. In Section 3, the task scheduling problem is described and its model is formulated. In Section 4, we present MOSCOA framework for the task scheduling to achieve maximizing parallelization and minimizing the makespan based on cuckoo optimization algorithm. In Section 5, we evaluate the time and space complexities. In Section 6, the results and analyses of simulations to validate the algorithm are given. In Section 7 downsides of MOSCOA algorithm is described. Finally, a conclusion and an overview of future work are presented in Section 8.

2. Related works

The major previous researches in task allocation problem at compile time in Heterogeneous Systems focused on static task scheduling algorithms. These algorithms can be classified as heuristic and meta-heuristic algorithms (Bansal, Kumar, & Singh, 2003; Kwok & Ahmad, 1996; Manudhane & Wadhe, 2013; Topcuoglu, Hariri, & Wu, 2002). The Heuristic algorithms are divided into list-based heuristics, clustering heuristics and duplication heuristics. Fig. 1 shows this classification together with the proposed algorithms samples for each category.

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. The 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). Heterogeneous earliest finish time 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 couple 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. *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 critical path on a processor 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 & Ahmad, 1996). Therefore, an effective scheduling list-based algorithm requires an appropriate scheduling of tasks placed on critical path (Daoud & 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 in different processors (Lin, Lin, Lin, Hsiung, & Shih, 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, Mishra, Mishra, & Tripathi, 2012).

The Meta-heuristic algorithms usually make up an important solution of global optimization algorithms. 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 methods in a way that can achieve solutions and innovations beyond what is normally accessible in local optimum search. There are two types of local and global optima while dealing with optimization problems. The local optima are the best solution found in an area of solution spaces but not necessarily the best for the whole problem space. In contrast, the global optima are the best solution to the whole problem space. Finding the global optima in most real-life problems is extremely difficult and therefore satisfactory and good-enough solutions are often accepted. Meta-heuristic algorithms are employed to achieve these targets. The reasons for using meta-heuristic algorithms could be put in three categories of simplicity, flexibility and ergodicity which come as below:

- ***Simplicity***: Most meta-heuristic algorithms are simple, easy to implement and relatively less complex. The primary part of the algorithm could be written in 100 lines in programming languages.
- ***Flexibility***: These algorithms are flexible as they are simple to cover a wide range of optimization problems which can't be handled by classic algorithms.
- ***Ergodicity***: Meta-heuristic algorithms have high degrees of ergodicity which means they can search multi-modal search spaces with sufficient variety and avoiding local optima at the same time. Ergodicity is often the result of randomized techniques derived from natural systems such as crossover and mutation or statistical models such Random walks or Le'vy flights (Yang, Cui, Xiao, Gandomi, & Karamanoglu, 2013).

There are various methods to solve task scheduling problems based on meta-heuristic methods. The most famous ones are: