Information Sciences 181 (2011) 567–581

Contents lists available at ScienceDirect

Information Sciences

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

A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system

Yun Wen, Hua Xu [*], Jiadong Yang

State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China

A R T I C L E   I N F O

Article history:
Received 2 February 2010
Received in revised form 28 September 2010
Accepted 2 October 2010

Keywords:
Variable neighborhood search
Genetic algorithm
Hybrid metaheuristic
Memetic algorithm
Heterogeneous multiprocessor scheduling

A B S T R A C T

Effective task scheduling, which is essential for achieving high performance in a heterogeneous multiprocessor system, remains a challenging problem despite extensive studies. In this article, a heuristic-based hybrid genetic-variable neighborhood search algorithm is proposed for the minimization of makespan in the heterogeneous multiprocessor scheduling problem. The proposed algorithm distinguishes itself from many existing genetic algorithm (GA) approaches in three aspects. First, it incorporates GA with the variable neighborhood search (VNS) algorithm, a local search metaheuristic, to exploit the intrinsic structure of the solutions for guiding the exploration process of GA. Second, two novel neighborhood structures are proposed, in which problem-specific knowledge concerned with load balancing and communication reduction is utilized respectively, to improve both the search quality and efficiency of VNS. Third, the proposed algorithm restricts the use of GA to evolve the task-processor mapping solutions, while taking advantage of an upward-ranking heuristic mostly used by traditional list scheduling approaches to determine the task sequence assignment in each processor. Empirical results on benchmark task graphs of several well-known parallel applications, which have been validated by the use of non-parametric statistical tests, show that the proposed algorithm significantly outperforms several related algorithms in terms of the schedule quality. Further experiments are carried out to reveal that the proposed algorithm is able to maintain high performance within a wide range of parameter settings.

1. Introduction

As a most promising approach to meet the rising computational requirements, parallel processing approach poses a number of problems not encountered in traditional sequential processing [9,23], the most important of which is the multiprocessor scheduling issue. In general, an originally large program can be decomposed into a set of smaller tasks prior to parallel processing. These smaller tasks almost always have dependencies representing the precedence constraints, in which the results of other tasks are required before a particular task can be executed. Hence, the goal of a task scheduling algorithm is typically to schedule all the tasks on the given number of available processors so as to minimize the overall length of time required to execute the entire program, namely makespan, without violating precedence constraints. Based on various characteristics of the decomposed tasks to be scheduled and the multiprocessor system, as well as the availability of *a priori* information regarding the processing time, the multiprocessor scheduling problem can be categorized into many different classes [12,29]. In this article, only the *static* scheduling problem is addressed, in which all information needed for

* Corresponding author.
E-mail address: xuhua@mail.tsinghua.edu.cn (H. Xu).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.10.001

scheduling, including task processing times, data dependencies, and communication costs between dependent tasks, are known before program execution.

Given the NP-complete complexity for searching an optimal solution in its general form [40], the multiprocessor scheduling problem remains an open field in spite of extensive studies. Traditional scheduling research focused on the heuristic-based algorithms, an important class of which is the so-called list scheduling algorithms [29,38]. The basic idea of list scheduling consists in maintaining an ordered list of tasks by assigning priority for each task according to some greedy heuristics. Tasks are then selected in the order of their priorities and the highest-priority ready task is removed from the list to be assigned to a processor which allows the earliest start time. Traditional list scheduling algorithms usually assume a homogeneous multiprocessor system in which all processors are of the same processing ability and fully connected [2,41], while recent studies have been diverted to task scheduling for heterogeneous multiprocessor systems where the execution time of a task may vary among different processors [38]. The heuristic-based scheduling algorithms are always efficient since they narrow the search down to a very small portion of the solution space by means of greedy heuristics. However, due to the greedy nature, heuristic-based approaches are not likely to produce consistent results on a wide range of problems, especially when the complexity of the scheduling problem increases.

In attempts to obtain schedules of better quality, many well-known metaheuristics, including genetic algorithm (GA) [1,6,8,21,33,37,42], artificial immune system (AIS) [44], ant colony optimization (ACO) [5], particle swarm optimization (PSO) [34], simulated annealing (SA) [26], tabu search (TS) [36], and variable neighborhood search (VNS) [32], have been adopted. Generally, metaheuristics can be divided into trajectory methods (also named local search heuristics) and population-based methods. Population-based methods deal with a set of solutions in every iteration of the algorithm while trajectory methods only deal with a single solution [3]. As one of the most studied population-based methods, GA shows robust performance with various scheduling problems, for it has a powerful global exploration ability of concurrently tracking a set of solutions. Plenty of empirical results demonstrate that GA-based methods always outperform traditional heuristic-based scheduling algorithms regarding the schedule quality [6,21,42]. However, GA usually takes more computing efforts to locate the optimal in the region of convergence [42], owing to the lack of local search ability. On the other hand, the trajectory method, such as VNS [10], has shown its potential in exploiting the promising regions in the search space with high quality solutions. Nevertheless, it is still prone to premature convergence traps due to the limited exploration ability. Thus, it's a natural choice to consider the hybridization of metaheuristics, also named memetic algorithm (MA) in some literatures [27], which has recently been applied to solve scheduling problems [4].

The approach proposed in this article is largely inspired by the fact that each type of scheduling technique has its own strengths and weaknesses while appearing that they are complementary to each other. This paper attempts to present a novel heuristic-based hybrid genetic-variable neighborhood search (GVNS) algorithm for solving the heterogeneous multiprocessor scheduling problem, which improves upon the standard GA in three aspects. First, the proposed algorithm incorporates GA and VNS to obtain a balance between the exploration and exploitation of the search space, which is crucial to the success of metaheuristics. Next, two common scheduling strategies, load balancing and communication reduction, are utilized in our proposed neighborhood structures in VNS, which further improves the local search efficiency. Finally, unlike many existing GA approaches which directly use GA to evolve the priorities of the tasks that in turn determine the final schedule solution, the proposed GVNS restricts the use of GA and VNS to find optimal task-processor mapping while leaving the task sequence assignment to an upward-ranking heuristic originally used in HEFT [38], a list scheduling algorithm.

In the next section, a formal statement of the studied heterogeneous multiprocessor scheduling problem is provided. Section 3 introduces some existing related researches on the multiprocessor scheduling problem. Section 4 presents the various features of the proposed GVNS. The benchmark problems, parameter settings, experimental results and analysis are explained in Section 5. Finally, conclusions and suggestions for future research are drawn in Section 6.

2. Problem formulation

The static multiprocessor scheduling problem is typically given by two inputs: a multiprocessor system in which tasks can be solved and a parallel program to be computed. Generally, the target multiprocessor system is composed of a network of processors, each of which has a local memory so that inter-processor communications rely solely on message-passing [29]. In this paper, the studied multiprocessor system model is assumed to be with the following characteristics: (1) heterogeneous; (2) non-preemptive; (3) fully connected network; (4) communication links with uniform bandwidth; (5) task duplication is NOT allowed; (6) each processor has an independent I/O unit that allows for communication and computation to be performed simultaneously.

Meanwhile, the parallel program is typically decomposed into smaller tasks with precedence constraints, which are described as a directed acyclic graph (DAG). In general, a DAG $G = (V,E)$ consists of a set V of v nodes and a set E of e edges. A node in the DAG represents a decomposed task, which must be executed sequentially without preemption in the same processor. Hereafter, the terms *node* and *task* will be used interchangeably. The computation cost of a particular task $n_i \in V$ on the processor p_j is denoted as $w_{i,j}$. Each edge $e_{i,j} \in E$ represents precedence constraints between task n_i and n_j, which means that the result of task n_i has to be transmitted to task n_j before task n_j starts its execution. Each edge $e_{i,j} \in E$ is associated with a nonnegative weight $c_{i,j}$ representing the communication cost between the interdependent tasks n_i and n_j. Note that the real communication cost is equal to zero when the interdependent pair of tasks are assigned to the same processor. The source