Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 36 (2009) 698–710

computers & operations research

www.elsevier.com/locate/cor

Scheduling of parallel machines to minimize total completion time subject to s-precedence constraints

Eun-Seok Kim[a], Chang-Sup Sung[a,*], Ik-Sun Lee[b]

[a]Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea

[b]School of Business, Dong–A University, Saha-gu, Pusan 604-714, Republic of Korea

Available online 4 November 2007

Abstract

This paper considers 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. The s-precedence relation has wide applicability in the real world such as first-come-first-served processing systems.

The problem is shown to be intractable, for which a heuristic procedure is derived. Numerical experiments are conducted to show that the derived heuristic provides effective solutions.

Keywords: Parallel machine scheduling; Precedence constraints; Total completion time; Heuristic

1. Introduction

The s-precedence constraints, introduced by Kim and Posner [1], require that one or more tasks may have to be started before another task is allowed to start its processing. It is different from standard precedence constraints in which one or more tasks may have to complete processing before another task is allowed to start. Standard precedence constraints are realistic when tasks are operations of a specific job, as might be the case for a job in a flow shop. There are numerous situations where standard precedence constraints occur in practice. One example is that the shell of a computer has to be built before the hardware can be installed. However, standard precedence constraints are less meaningful when the tasks are unrelated jobs. Except for single machine problems, it is hard to find realistic examples where one unrelated job should be constrained to start no sooner than another finishes.

A more common situation when there are precedence constraints between unrelated jobs is where one job is constrained to start no sooner than another job starts. This type of constraint occurs in first-come-first-served environments. In this environment, suppose job 1 arrives before job 2. Then, while job 2 cannot start processing before job 1, job 2 can complete before job 1. These types of restrictions are defined to be *s-precedence* constraints. Formally, s-precedence constrains the processing order of jobs. If job *i* precedes job *j*, then job *j* cannot start processing before

* Corresponding author. Tel.: +82 42 869 3131; fax: +82 42 869 3110.
E-mail address: cssung@kaist.ac.kr (C.-S. Sung).

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.10.025

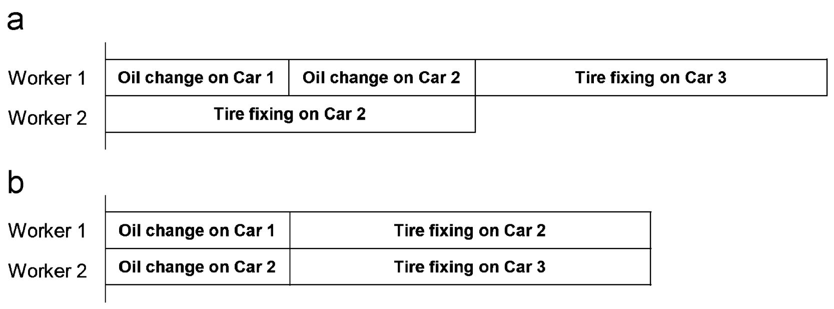

Fig. 1. Two feasible schedules (a) A feasible schedule (b) Another feasible schedule.

job i starts. Observe that it is not possible to model this type of environment with standard scheduling restrictions. Examples of these types of constraints include most queueing systems where earlier arriving jobs start processing first. Applications occur in many production and service industries, including telecommunications, health care, and bank service.

The proposed concept of precedence constraints can be illustrated by an automotive repair shop where each car has several broken parts that need to be fixed. Each mechanic is capable of fixing up all broken parts, and several mechanics can work on different parts of the same car at the same time. With respect to the arrival order of cars, cars that arrive first must start its all repair processes before cars that arrive later. For example, suppose that two workers are employed to provide their services in parallel and three cars currently wait for repair service. The three cars have arrived in sequence to get individual repair service on "oil change", "oil change and tire fixing", and "tire fixing", respectively. Suppose that the repair times required on the service "oil change" and "tire fixing" take 1 and 2 h, respectively. The associated two feasible s-precedence schedules are illustrated as in Fig. 1. As shown in Fig. 1, "oil change" of first car starts before "oil change" and "tire fixing" of second car starts, and "oil change" and "tire fixing" of second car starts before "tire fixing" of third car starts.

As a reference, Kim and Posner [1] have considered the parallel machine scheduling problem of minimizing makespan subject to s-precedence constraints. They have showed that the problem is strongly *NP*-hard, and introduced a list scheduling heuristic that, whenever a machine becomes available, schedules the job maximizing the sum of processing times of its successors along any path. Based on the structures of the s-precedence constraint graph, attainable worst case bounds on the relative error have been developed.

As in the literature, some studies have been made on scheduling problems of minimizing total completion time subject to standard precedence constraints. For single machine problems, Lawler [2] has proved that the problem with arbitrary precedence constraints is *NP*-hard. A polynomial-time algorithm has been known when the precedence graph is a forest [3]. Adolphson and Hu [4] have showed that such a problem is equivalent to the optimal linear ordering problem, and showed that Horn's algorithm can be implemented in running time $O(n \log n)$. The case of arbitrary precedence has been treated by Sidney [5]. He has suggested the concept of ρ-maximal initial sets, which can be used to decompose the arbitrary precedence constraints of this problem. However, he also has indicated that there is no efficient method of finding these ρ-maximal initial sets. Chekuri and Motwani [6] have suggested 2-approximation algorithm based on solving minimum cut, and Chudak and Hochbaum [7] have suggested 2-approximation algorithm using modified linear programming (LP) relaxations.

For multi-machine problems, the problem is strongly *NP*-hard even when the precedence graph is a set of chains [8]. Based on the well-known A* algorithm proposed in the field of artificial intelligence [9], Chang and Hsu [10] have developed branch-and-bound approach in order to efficiently find an optimal schedule for arbitrary precedence graph. Dror et al. [11] have distinguished between *strong* and *weak* precedence constraints. In a strong chain precedence constraint, the jobs of a chain have to be done on one machine with no job from another chain inserted between them, whereas a weak chain allows for such insertions. They have proved that two machines problem with strong chains can be solved in polynomial time. Hall et al. [12] have proposed 7-approximation algorithm using LP relaxations when each job has individual release date. Ramachandra and Elmaghraby [13] have provided a binary integer program (BIP) and a dynamic program (DP) which has computational limits around 25 jobs with mean job processing time of 10. They have also introduced a genetic algorithm (GA) procedure for larger size problems. Queyranne and Schulz [14]