Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 994–1009
computers & operations research
www.elsevier.com/locate/cor

Scheduling two parallel machines with a single server: the general case

Amir H. Abdekhodaee[a], Andrew Wirth[b,*], Heng-Soon Gan[a]

[a] *Operations Research Group, CSIRO Mathematics and Information Sciences, Private Bag 10, Clayton South MDC 3169, Australia*

[b] *Department of Mechanical and Manufacturing Engineering, The University of Melbourne, Parkville, VIC 3010, Australia*

Available online 17 September 2004

Abstract

This paper considers the problem of scheduling two-operation non-preemptable jobs on two identical semi-automatic machines. A single server is available to carry out the first (or setup) operation. The second operation is executed automatically, without the server. The general problem of makespan minimization is NP-hard in the strong sense. In earlier work, we showed that the equal total length problem is polynomial time and we also provided efficient and effective solutions for the special cases of equal setup and equal processing times. Most of the cases analyzed thus far have fallen into the category of regular problems. In this paper we build on this earlier work to deal with the general case. Various approaches will be considered. One may reduce the problem to a regular one by amalgamating jobs, or we may apply the earlier heuristics to (possibly regular) job clusters. Alternately we may apply a greedy heuristic, a metaheuristic such as a genetic algorithm or the well known Gilmore–Gomory algorithm to solve the general problem. We report on the performance of these various methods.

Keywords: Parallel machines; Single server; Setup

1. Introduction

This paper considers the general problem, $P2, S1|p_i, s_i|C_{\max}$, Brucker et al. [1], of scheduling two-operation non-preemptable jobs on two identical semi-automatic machines. A single server is available

* Corresponding author. Tel.: +61-3-8344-4852; fax: +61-3-9347-8784.
E-mail addresses: amir.abdekhodaee@csiro.au (A.H. Abdekhodaee), wirth@mame.mu.oz.au (A. Wirth), hsgan@mame.mu.oz.au (H.-S. Gan).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.08.013

to carry out the first (or setup) operation. The server can handle at most one job at a time. The second operation is executed automatically, and on the same machine, without the server. The processing operation must be carried out immediately after the setup. Our objective is to minimize makespan. Let p_i and s_i denote the processing and setup times, respectively, of job i. It is well known, Brucker et al. [1], that $P2, S1|p_i, s_i|C_{\max}$ is unary NP-hard. So is $P2, S1|s_i = s|C_{\max}$. Brucker et al. [1] contains a full discussion of the computational complexity of these and many related problems. They show, for example, that $P2, S1|p_i = p|C_{\max}$ is binary NP-hard, but on the other hand they prove that $P, S1|p_i = p, r_i, s_i = s|C_{\max}$, where r_i denote setup release dates, is polynomial time. Brucker et al. [1] also consider various other scenarios and objective functions. Not surprisingly, all but the simplest cases are NP-hard. One special case which is polynomial time is $P2, S1|p_i + s_i = a|C_{\max}$, see Abdekhodaee and Wirth [2] for further details.

Most of the literature on the general problem has dealt with issues of computational complexity. Computational results appear to be restricted to a few papers. Namely Koulamas [3],who studied the closely related problem of minimizing *total idle time* (excluding the last inevitable idle time). Also Abdekhodaee and Wirth [2] and Abdekhodaee et al. [4], who considered the *regular* case, where $p_i - p_j \leqslant s_j$ for all i, j and the equal processing and equal setup times cases.

In this paper, we build on this earlier work to deal with the general case. Various approaches will be considered. One may reduce the problem to a regular one by merging jobs, or we may partition the jobs into (possibly regular)subsets, apply the earlier heuristics and then sequence these partial solutions using the well known Gilmore–Gomory algorithm. We may also apply a greedy heuristic or a metaheuristic such as a genetic algorithm to solve the general problem.

The structure of this paper is as follows: first we review some of the results of Abdekhodaee and Wirth [2] and Abdekhodaee et al. [4] and discuss the regularity condition and its schematic representation in the processing time—setup time plane. Then we briefly consider how the general problem may be reduced to a (possibly) regular one by amalgamating or clustering jobs.

A range of heuristics, based on the above considerations as well as two versions of a greedy procedure and a genetic algorithm are presented. We then introduce a version of the Gilmore–Gomory algorithm appropriate to this problem. We compare the performance of these various approaches for a range of different scenarios and draw some conclusions.

2. The processing—setup time plane

We briefly recall some definitions from Abdekhodaee and Wirth [2] and a result from Abdekhodaee et al. [4].

Assume that we have n jobs with setup times s_i and processing times p_i for $i = 1, \ldots, n$. Let t_i be the start time of job i and c_i its completion time. So $c_i = t_i + s_i + p_i$. Denote the length of job i by $a_i = s_i + p_i$. Fig. 1 illustrates this.

As stated earlier processing does not require the server. We also introduce the convention that uppercase terms shall refer to the set of jobs after it has been scheduled, thus C_i shall refer to the completion time of the ith scheduled job and S_1 is the setup time of the job that is scheduled first. We assume that no job is unnecessarily delayed (Fig. 2). We say a set of jobs is *regular* if $p_i \leqslant a_j$ for all i, j. If the jobs, sorted by setup start times, is processed alternately on the two machines we say the processing or schedule is *alternating*. We assume, without loss of generality, that the first job is started on machine one and that