Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 35 (2008) 2018–2033

computers & operations research

www.elsevier.com/locate/cor

Heuristics for minimizing maximum lateness on a single machine with family-dependent set-up times

Reha Uzsoy*, Juan Diego Velásquez

Laboratory for Extended Enterprises at Purdue, School of Industrial Engineering, Grissom Hall, 315 N. Grant Street, West Lafayette, IN 47907-2023, USA

Available online 4 December 2006

Abstract

We address the problem of scheduling a single machine subject to family-dependent set-up times in order to minimize maximum lateness. We present a number of local improvement heuristics based on the work of previous researchers, a rolling horizon heuristic, and an incomplete dynamic programming heuristic. Extensive computational experiments on randomly generated test problems compare the performance of these heuristics. The rolling horizon procedures perform particularly well but require their parameters to be set based on problem characteristics to obtain their best performance.

Keywords: Scheduling; Single machine; Setup times; Heuristics; Dynamic programming

1. Introduction

The shop-floor scheduling activity, which tries to allocate available production to jobs over time, is an important element in achieving a good tradeoff between the operational objectives of high machine utilization, low work-in-process (WIP) inventory, high customer satisfaction and short delivery lead times. However, maintaining a good tradeoff between these objectives becomes significantly more difficult when the production equipment requires significant set-up times. While the long-run solution, as advocated by just-in-time manufacturing [1], is to redesign the production process so that set-up times are reduced or eliminated, in the short-term set-up times are a fact of life in many environments. Hence, effective scheduling procedures for machines with set-up times fulfill a useful, common industrial need.

In this paper, we consider a single-machine processing jobs one at a time, where all jobs are available simultaneously. Preemption is not allowed. Jobs are classified into families based on machine and process requirements. When the machine processes two jobs from the same family one after the other, no set-up time is required. However, when jobs from different families are processed consecutively, a set-up time is incurred. Equipment of this type is widely encountered in industry. An example described in detail by Ashby and Uzsoy [2] consists of a turning center where rods of different diameters are processed to make effectors for hydraulic and pneumatic cylinders. Rods of the same diameter can be processed consecutively with negligible set-up time, as this requires only downloading a part program from memory. However, in order to process rods of a different diameter from that currently on the machine, the jaws of the machine must be disassembled and adjusted to the right diameter, causing a significant set-up time. This turning

* Corresponding author. Tel.: +1 765 494 5400; fax: +1 765 494 5448.
E-mail address: uzsoy@ecn.purdue.edu (R. Uzsoy).

0305-0548/$ - see front matter
doi:10.1016/j.cor.2006.10.003

center was the bottleneck station of the entire system, motivating the development of effective scheduling algorithms for this individual machine.

In this work we take the position that the function of shop-floor scheduling is to ensure that production on the shop-floor adheres as closely as possible to the master production schedule, since the personnel making shop-floor decisions do not have information necessary to make high-level capacity allocation decisions such as which customer's orders have priority over others. Hence we assume, following common industrial practice, that each job on the shop-floor has been assigned a completion date by the planning system, and that the scheduling system should try to minimize some function of deviation from these due dates. This motivates our formulation of the shop-floor scheduling problem as that of minimizing maximum lateness (L_{max}), since this will avoid making some jobs early at the expense of others being extremely late. More discussion of this motivation is given in Ovacik and Uzsoy [3]. Extensive computational experiments [3] have shown that schedules that perform well for L_{max} also perform well with respect to total tardiness and cycle time. Hence, we focus on the L_{max} measure in this paper.

Thus, the problem addressed in this paper can be formally stated as follows: we are given n jobs that are simultaneously available for scheduling. The jobs can be partitioned into m families based on their set-up requirements, such that each family i contains n_i jobs and

$$\sum_{i=1}^{m} n_i = n.$$

We shall denote the jth job of family i by job (ij), which has processing time p_{ij} and due date d_{ij}. If two jobs of the same family are processed consecutively, no set-up time is incurred. On the other hand, if a job of family k is processed after a job of family i, a set-up time of s_k time units defined by the family of the new family being set up is incurred. The objective is to minimize the maximum lateness over all jobs in the schedule, given by

$$L_{max} = \max_{1 \leqslant j \leqslant n_i, 1 \leqslant i \leqslant m} \{C_{ij} - d_{ij}\},$$

where C_{ij} denotes the completion time of job j of family i in the schedule in question. Note that since L_{max} is a regular performance measure that is monotonically non-decreasing in the job completion times C_{ij}, there exists an optimal schedule with no inserted idle time.

This problem has been studied by a number of researchers [4,5], and exact solutions are known to be hard to obtain in CPU times that are short enough to permit the use of the resulting algorithms on the shop floor. Hence, we focus on developing effective heuristics that obtain approximate solutions in modest CPU times.

In the following section we discuss previous related work in the area of the single-machine scheduling. A description of the proposed heuristics is presented in Section 3. In Section 4, the experimental design and results are presented. We conclude the paper with a summary and some directions for future research.

2. Previous related work

The problem of minimizing L_{max} on a single machine without set-up times has been extensively explored. The simplest case of the problem, denoted by $1//L_{max}$ in the standard three-field notation [6], is easy to solve using the earliest due date (EDD) rule [7] in $O(n \log n)$ time, where n is the number of jobs. The problem with arbitrary precedence constraints ($1/prec/L_{max}$) is solved using Lawler's algorithm [8] in $O(n^2)$ time. Since there are no set-up times present in these problems, we shall drop the family subscript i in referring to job parameters, retaining the job index j.

The inclusion of non-simultaneous release times r_j causes the problem ($1/r_j/L_{max}$) to become NP-hard in the strong sense [9]. Several researchers have explored branch-and-bound algorithms for the $1/r_j/L_{max}$ problem. Carlier [10] uses the EDD algorithm to identify a critical job and its respective critical set at each node of the search tree; two subsets are identified based on whether the critical job precedes or follows every job in the critical set. Similar to the branch-and-bound method of McMahon and Florian [11], this procedure has the unusual trait of obtaining a complete solution at each node of the tree. Other methods for $1/r_j/L_{max}$ have been developed by Kise and Uno [12], Kise et al. [13], Potts [14], and Hall and Shmoys [15].

Ignoring set-up time considerations can lead to excessive amounts of time spent in changeovers, resulting in a significant loss of capacity; therefore, the problem of minimizing L_{max} with the presence of set-up times is of importance.