European Journal of Operational Research 235 (2014) 540–552

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A mixed integer linear programming approach to minimize the number of late jobs with and without machine availability constraints

Boris Detienne

Institut de Mathématiques de Bordeaux, Université Bordeaux 1, Team RealOpt, INRIA Bordeaux Sud-Ouest, France

A R T I C L E   I N F O

Article history:
Received 16 October 2012
Accepted 16 October 2013
Available online 30 October 2013

Keywords:
Scheduling
Integer programming
Modeling
Availability constraints
Late jobs
Exact method

A B S T R A C T

This study investigates scheduling problems that occur when the weighted number of late jobs that are subject to deterministic machine availability constraints have to be minimized. These problems can be modeled as a more general job selection problem. Cases with resumable, non-resumable, and semi-resumable jobs as well as cases without availability constraints are investigated. The proposed efficient mixed integer linear programming approach includes possible improvements to the model, notably specialized lifted knapsack cover cuts. The method proves to be competitive compared with existing dedicated methods: numerical experiments on randomly generated instances show that all 350-job instances of the test bed are closed for the well-known problem $1|r_i|\sum w_iU_i$. For all investigated problem types, 98.4% of 500-job instances can be solved to optimality within 1 hour.

1. Introduction

Even though most scheduling problems assume that a job can be processed anytime after its release date, an increasing number of research papers investigate scheduling problems that are subject to machine or job availability constraints. In this context, machines or jobs may be unavailable for processing during given time intervals. Such constraints appear (Hashemian, Diallo, & Vizvàri, 2012; Kubiak, Blazewicz, Formanowicz, Breit, & Schmidt, 2002; Mellouli, Sadfi, Chu, & Kacem, 2009, 2013; Schmidt, 2000) in the cases of joint production and maintenance management (where preventive maintenance or periodic repair tasks limit the usage time of machines), operating systems for mono and multi-processors (where high priority tasks interfere with low priority programs) or when implementing rolling time horizon approaches (where decisions taken for a first time horizon are considered as fixed for the consecutive overlapping time horizon), etc.

The problems investigated in this paper are defined by a set $I = \{J_1, \dots, J_n\}$ of n jobs. Each job J_i is characterized by a release date r_i, a due date d_i, a processing time p_i, and a weight w_i. All jobs must be processed on a single machine so that the weighted number of late jobs is minimized. A job cannot be preempted by another one: if a job starts on the machine, no other job can start until the first one completes. Additional input consists of a set of K time intervals $[B_s, F_s], s \in \{1, \dots, K\}$, when the machine is unavailable. Moreover, we add two fictitious unavailability periods 0 and $K+1$ with no loss of generality, such that $B_0 = F_0 = 0$ and $B_{K+1} = F_{K+1} = \infty$. All data are integers and deterministic.

We investigate different job behaviors in terms of unavailability constraints. First, jobs are said to be *non-resumable*, according to the classification of Ma, Chu, and Zuo (2010): if a job has started but has not completed before an unavailability period, it must be restarted from zero after the unavailability period. This problem is denoted by $1, h_k|r_i, nr-a|\sum w_iU_i$ in standard three-field notation (Graham et al., 1979). Conversely, jobs may be *resumable*: a job may start before an unavailability period and be resumed thereafter. In this case, the completion of the job is simply postponed for the duration of the unavailability periods crossed by the job. This problem is denoted by $1, h_k|r_i, r-a|\sum w_iU_i$. A third type of problem allows jobs to be resumed after an unavailability period at the expense of an additional setup time. Two variations are possible for this feature, which we refer to as *semi-resumable* jobs: according to Lee (1999), interrupted jobs must be partially restarted after an unavailability period, occupying the machine for a time which is proportional to the duration of the job that has already been processed. In this paper, we address a variation of this feature for which the additional setup time is fixed and job-dependent, analogous to the work of Graves and Lee (1999). More precisely, the setup duration is equal to $\min(\alpha_i, B_{s+1} - F_s)$, where α_i is a fixed setup time related to job J_i. The hypothesis is motivated by the following: The industrial process does not require partial job reprocessing in many real-life situations, especially in the case of preventive and planned maintenance. Instead, the machine

E-mail address: boris.detienne@u-bordeaux1.fr requires a setup operation, whose duration only depends on the job to be resumed. An example of this kind of requirement is encountered in the corrugated cardboard industry following the periodic cleaning operations of cutting devices. These setup times are implicitly included in job processing times when no maintenance activities are considered. Also, sometimes a setup operation following F_s will not be completed before B_{s+1}, and another setup time will be performed after F_{s+1}. Therefore, in many practical situations, the job will simply be deferred until F_{s+1}. According to the classification of Allahverdi, Ng, Cheng, and Kovalyov (2008), these are referred to as *non-batch sequence-independent setup times*. The problem is denoted by $1,h_k|r_i,ST_{si}|\sum w_iU_i$, of which problems $1,h_k|r_i,r-a|\sum w_iU_i$ and $1|r_i|\sum w_iU_i$ are clearly special cases.

This paper aims to provide a generic mixed integer linear programming (MILP) approach to solve problems $1,h_k|r_i,nr-a|\sum w_iU_i$ and $1,h_k|r_i,ST_{si}|\sum w_iU_i$ and their special cases $1,h_k|r_i,r-a|\sum w_iU_i$ and $1|r_i|\sum w_iU_i$. Because problem $1|r_i|\sum w_iU_i$ is strongly NP-hard (Graham et al., 1979), so are the other problems. To the best of our knowledge, to date no published work proposes an exact solution approach for cases with availability constraints, nor an efficient MILP approach for $1|r_i|\sum w_iU_i$.

Our paper is structured as follows: Section 2 contains a brief literature review for related scheduling problems. Section 3 describes the general scheduling problem (*STWP*) of selecting and scheduling jobs that are subject to time windows and group constraints on parallel machines; three MILP models are proposed. Section 4 presents valid inequalities and bound tightening results to improve these models. In particular, we use one of the MILP formulations to derive lifted knapsack cover cuts, which are also valid for the other models. We propose two specialized lifting procedures, embedding a subset of the specific constraints of *STWP* to yield stronger cuts. Section 5 describes how problems $1,h_k|r_i,nr-a|\sum w_iU_i$ and $1,h_k|r_i,ST_{si}|\sum w_iU_i$ can be converted into *STWP*. Moreover, we show that problem $1,h_k|r_i,r-a|\sum w_iU_i$ is equivalent to $1|r_i|\sum w_iU_i$, which proves some complexity results. Our method proves to be reasonably competitive compared with existing dedicated methods: in Section 6, numerical experiments on randomly generated instances show that most (98.4%) 500-job instances can be solved to optimality within one hour for all investigated problem types. For the well-known problem $1|r_i|\sum w_iU_i$, all 350-job instances are closed and only one (resp. four) 400-job instance (resp. 500-job instances) remains open. Section 7 discusses possible extensions and improvements to this work. Finally, an Appendix provides details concerning the dynamic programming algorithms used by the lifting procedures.

2. Literature review

The problem of minimizing the (weighted) number of late jobs on one machine has been studied extensively. For the general case $1|r_i|\sum w_iU_i$, (Dauzère-Pérès, 1995; Dauzère-Pérès & Sevaux, 2002) present lower bounds and heuristics and (Baptiste, Peridy, & Pinson, 2003; Péridy, Pinson, & Rivreau, 2003; Sadykov, 2008) describe algorithms to solve the problem to optimality. To our knowledge, the best known exact algorithms have been contributed by M'Hallah and Bulfin (2007) for the weighted case and by Kao, Sewell, Jacobson, and Hall (2012) for the unweighted case $1|r_i|\sum U_i$. These methods make it possible to solve instances with up to 200 and 300 jobs, respectively. Baptiste and Sadykov (2009) present a generic MILP model for total cost scheduling problems with piecewise linear objective functions. According to their numerical experiments, this very general model does not perform as well as existing dedicated approaches.

Concerning only the case of deterministic machine availability constraints (when unavailability periods are perfectly known in advance), (Ma et al., 2010) gathers more than 90 research papers. Interestingly, only one paper mentioning the minimization of the number of late jobs is referenced in this survey: (Lee, 1996) shows that $1,h_1|r-a|\sum U_i$ is polynomial and that Moore–Hodgson's algorithm (Moore, 1968) leads to an absolute error of 1 to solve the problem $1,h_1|nr-a|\sum w_iU_i$. Another survey (Schmidt, 2000) cites (Lawler & Martel, 1989), who studies the minimization of the number of late jobs on two machines whose speeds can vary (and possibly be null) over time if preemption is allowed. Chen (2009), who studies a special case of $1,h_k|nr-a|\sum U_i$ in which unavailability intervals correspond to periodic maintenance, develops a branch-and-bound to optimally solve instances with up to 32 jobs. Zhong, Ou, and Wang (2014) studies the Order Acceptance and Scheduling problem with such constraints, which is an extension of the problem.

Semi-resumable jobs with non-batch sequence-independent setup times were studied in Graves and Lee (1999), where the authors consider both job and maintenance planning for problems $1|ST_{si}|L_{max}$ and $1|ST_{si}|\sum w_iC_i$. The study provides complexity results and algorithms for some polynomial and ordinary NP-hard special cases. Liu and Edwin Cheng (2002, 2004) investigate problems $1|ST_{si},pmnt,r_i|D_{max}$ and $1|ST_{si},pmnt,r_i|\sum w_iC_i$, where a fixed setup time occurs after each job preemption. A special case of STWP is studied in Sadykov and Wolsey (2006), where the authors combine MILP and constraint programming techniques to solve $R|r_i,d_i|\sum c_{ij}$, for which jobs have to be assigned to machines so that the assignment cost is minimized and all jobs are processed within their respective time window. The authors report success for instances with up to nine machines and 54 jobs.

3. The problem of scheduling jobs with time windows

This section describes the problem of selecting jobs with time windows on parallel processors (STWP), which is solved with the help of an efficient mixed integer linear program.

3.1. Definition

An instance of STWP is defined by the following data:

- A set $I=\{J_1,\dots,J_{n_I}\}$ of n_I jobs.
- A partition G of I into n_G disjoint groups: $G=\{G_1,\dots,G_{n_G}\}$.
- A set $M=\{M_1,\dots,M_{n_M}\}$ of n_M machines.
- For each job $J_i\in I$, the following integers are given: a release date r_i, a deadline $\bar{d}_i$, a processing cost w_i, a processing time p_i, and a processing machine m_i.

A feasible solution of this instance of STWP satisfies the following constraints: For each group $G_g, g\in\{1,\dots,n_G\}$, exactly one job in G_g must be processed. Processing job J_i, $i\in\{1,\dots,n_I\}$ generates a cost w_i. If it is selected, job J_i must be processed on machine M_{m_i} without preemption within its processing time window $[r_i,\bar{d}_i]$. The objective is to minimize the total processing cost.

This problem clearly generalizes the problem of minimizing the number of tardy jobs on parallel machines ($P\|\sum U_i$); it is NP-hard in the strong sense (Garey & Johnson, 1979) when the number of machines is arbitrary.

3.2. Characterizing feasible job selections

This section introduces a dominance property for optimal schedules of *STWP*. It is a generalization of a result proposed in Detienne, Dauzère-Pérès, and Yugma (2011), where the authors present MILP models to minimize a regular step cost function of job completion times on parallel machines. A similar principle