European Journal of Operational Research 252 (2016) 750–760

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation

Boris Detienne[a,b,*], Ruslan Sadykov[a,b], Shunji Tanaka[c]

[a] Institute of Mathematics, University of Bordeaux, France
[b] Inria Bordeaux – Sud-Ouest, France
[c] Institute for Liberal Arts and Sciences, Department of Electrical Engineering, Kyoto University, Japan

ARTICLE INFO

Article history:
Received 10 June 2015
Accepted 3 February 2016
Available online 11 February 2016

Keywords:
Scheduling
Flowshop
Branch-and-bound
Lagrangian relaxation

ABSTRACT

We consider the flowshop problem on two machines with sequence-independent setup times to minimize total completion time. Large scale network flow formulations of the problem are suggested together with strong Lagrangian bounds based on these formulations. To cope with their size, filtering procedures are developed. To solve the problem to optimality, we embed the Lagrangian bounds into two branch-and-bound algorithms. The best algorithm is able to solve all 100-job instances of our testbed with setup times and all 140-job instances without setup times, thus significantly outperforming the best algorithms in the literature.

1. Introduction

Problem description. We consider the problem of scheduling a set of jobs $J = \{1, \dots, n\}$ in a two-machine flowshop with the objective of minimizing the sum of completion times of jobs. The jobs are available at time zero and they should be processed first on machine 1, and then on machine 2. Each machine can process at most one job at a time. Let p_j^m denote the processing time of job j on machine m, where $m = 1, 2$. All processing times are integer. Preemption of the processing of the jobs in not allowed on either machine. Let C_j^m denote the completion time of job j on machine m. According to the scheduling classification, the problem is denoted by $F2||\Sigma C_j$. It is known to be NP-hard in the strong sense (Garey, Johnson, & Sethi, 1976). It has been shown by Conway, Maxwell, and Miller (1967) that there exists at least one optimal solution where both machines have the same sequence of jobs. Thus, we may restrict the search to permutation schedules only.

In addition to this classic two-machine flowshop problem, we also consider its extension in which every job should be set up on each machine before being processed. Let s_j^m denote the setup time of job j on machine m. Setup of a job on machine 2 and processing of the same job on machine 1 can be performed in parallel. Note that setup times do not depend on the job processed just before job j, i.e. the setup times are *sequence independent*. This generalisation of the problem has been treated previously by Gharbi, Ladhari, Msakni, and Serairi (2013). It can be denoted as $F2|ST_{si}|\Sigma C_j$ in the scheduling classification. The set of permutation schedules remains dominant for this generalization as indicated in Allahverdi, Gupta, and Aldowaisan (1999).

Literature review. The problem $F2||\Sigma C_j$ has been studied in the literature for many years. First lower bounds and the branch-and-bound algorithms based on them were proposed by Ignall and Schrage (1965), Ahmadi and Bagchi (1990), and Della Croce, Narayan, and Tadei (1996). In these papers, instances with 10, 15, and 30 jobs, respectively, have been solved to optimality. Note that the processing times of jobs here are quite short and do not exceed 20 time units (even 10 in Della Croce et al. (1996)).

Several Lagrangian relaxation-based lower bounds have been proposed for the problem. van de Velde (1990) relaxed precedence constraints between operations of the same job to obtain a lower bound. The Lagrangian relaxation subproblem is difficult in the case where the schedule is restricted to be a permutation one. However, this subproblem becomes polynomially solvable when we further restrict Lagrangian multipliers to a same value. This restriction makes the Lagrangian bound weaker but computable in a short time. A perturbation procedure is used to improve this bound which consists in modifying the Lagrangian multipliers so as not to break the optimality of the current solution. Instances with up to 20 jobs have been solved to optimality using this bound. Hoogeveen and van de Velde (1995) have improved this lower bound by adding slack variables to the precedence constraints and by transforming these constraints to

* Corresponding author. Tel.: +33 5 40 00 21 43.
E-mail address: boris.detienne@math.u-bordeaux.fr, boris.detienne@u-bordeaux1.fr, boris.detienne@gmail.com (B. Detienne).

http://dx.doi.org/10.1016/j.ejor.2016.02.003
0377-2217/© 2016 Elsevier B.V. All rights reserved.

equalities. However, this improved lower bounds has not been tested inside an enumeration algorithm. Della Croce, Ghirardi, and Tadei (2002) used the same Lagrangian lower bound, but improved the perturbation procedure used by van de Velde (1990). They also introduced some dominance relations to reduce the enumeration in the branch-and-bound algorithms. Instances with up to 45 jobs with short processing times (up to 10) and up to 30 jobs with long processing times have been solved to optimality.

A positional (assignment) formulation for the problem $F2||\Sigma C_j$ has been proposed independently by Akkan and Karabati (2004) and Hoogeveen, van Norden, and van de Velde (2006). In both works, the authors use the notion of waiting time of a job before its processing starts on the second machine. The formulation has $O(n^2)$ variables and $O(n)$ constraints. In Hoogeveen et al. (2006), it was shown experimentally that the lower bound one can obtain by solving the linear relaxation of the positional formulation is stronger than any other bound proposed previously in the literature. It was also shown that any Lagrangian relaxation does not improve this bound. In Akkan and Karabati (2004), a network flow formulation for the problem was also suggested. In this network, each node corresponds to a position in the schedule and the waiting time of the job on this position. The network was then reduced by finding bounds on waiting times of jobs on different positions. To find a lower bound and design a branch-and-bound, the Lagrangian relaxation is used, in which job occurrence constraints are relaxed. Here the subproblem is the shortest path problem, and the Lagrangian dual problem is solved using a subgradient method. The branch-and-bound algorithm which uses this Lagrangian relaxation is able to solve instances with up to 60 jobs with small processing times (up to 10) and up to 45 jobs with large processing times (up to 100).

Haouari and Kharbeche (2013) proposed valid inequalities for the positional formulation. They experimentally showed that the dual bound of the linear relaxation of the positional formulation is improved when these inequalities are added. However, these improved dual bounds were not embedded in any exact algorithm for the solution of the problem.

Hoogeveen and Kawaguchi (1999) considered several special cases of the problem $F2||\Sigma C_j$. They proposed approximation algorithms for the general case of the problem as well as for a special case which they proved to be NP-hard. Three special cases of the problem were proved to be polynomially solvable.

There were two most important recent contributions to the problem $F2|ST_{si}|\Sigma C_j$ with sequence-independent setup times. Allahverdi (2000) proposed two dominance relations and a branch-and-bound algorithm based on them. With this algorithm, all instances with up to 20 jobs with large processing and setup times (up to 100) were solved to optimality.

Gharbi et al. (2013) proposed several dual bounds for the problem $F2|ST_{si}|\Sigma C_j$. Some of the suggested lower bounding procedures are similar to those used for the problem without setup times. One lower bound is based on solving the linear relaxation of a positional formulation. Another lower bound is based on Lagrangian relaxation similar to one used in van de Velde (1990). Best exact algorithms based on the proposed dual bounds allowed the authors to solve all instances with up to 30 jobs and the majority of instances with 35 jobs with large processing and setup times (up to 100).

Our contribution. In this work, we propose improved branch-and-bound algorithms for the problem $F2||\Sigma C_j$ as well as for its extension $F2|ST_{si}|\Sigma C_j$. Our approach is based on the network flow formulation from (Akkan & Karabati, 2004). To obtain stronger dual bounds, we use a larger network than the one used in Akkan and Karabati (2004). Different dominance rules and filtering techniques are exploited in order to cope with the size of the network. The structure of the network allows us to compute an expensive Lagrangian dual bound only once at the root node, and then recompute the bound in linear time at every node of the enumeration tree. Thus, millions of nodes can be checked in a reasonable time. Using the proposed algorithm, we are able to solve all instances of both problems $F2||\Sigma C_j$ and $F2|ST_{si}|\Sigma C_j$ with up to 100 jobs with large processing times.

The outline of the paper is as follows. In Section 2, we give the classic assignment MIP formulation of the problem. Different dominance rules, which are both from the literature and new ones, are described in Section 3. In Section 4, we present network flow formulations for the problem, as well as a subgradient algorithm with embedded filtering procedures to obtain Lagrangian dual bounds. Two improved branch-and-bound algorithms for the problem are suggested in Section 5. Results of computational experiments with these algorithms are given in Section 6. Finally, in Section 7, conclusions are drawn.

2. Mixed-integer linear programming formulation

This section introduces a mixed-integer linear programming formulation for the problem $F2|ST_{si}|\Sigma C_j$, which generalizes the positional formulation proposed in Akkan and Karabati (2004).

Note that the setup time of any job on machine 1 can be integrated into its processing time on machine 1. This follows from the fact that there exists an optimal schedule in which machine 1 process jobs without idle time. So, without loss of generality, for all $j \in J$, we can set $s^1_j = 0$, and adjust appropriately processing times p^1_j.

In the following, $[k]$ denotes the index of the job in position k. Assuming the convention that $C^1_{[0]} = C^2_{[0]} = 0$, the completion times $C^m_{[k]}$ of the job in position k, $k \in J$, on machines $m = 1, 2$ can be computed as:

$$C^1_{[k]} = C^1_{[k-1]} + p^1_{[k]}. \tag{1}$$

$$C^2_{[k]} = \max\left\{C^1_{[k]}, C^2_{[k-1]} + s^2_{[k]}\right\} + p^2_{[k]}. \tag{2}$$

In Akkan and Karabati (2004), the authors introduced the notion of time lag between the processing of the same job on both machines to write a positional model and a network flow model for the problem. This kind of models is also called *waiting time-based* models in Gharbi et al. (2013).

The completion-to-completion lag L^c_k of the job in position k, $k \in J$ is defined as the time elapsed between the completion of the job on machine 1 and its completion on machine 2:

$$\begin{aligned} L^c_k &= C^2_{[k]} - C^1_{[k]} \\ &= \max\left\{0, L^c_{k-1} + s^2_{[k]} - p^1_{[k]}\right\} + p^2_{[k]}. \end{aligned}$$

The completion-to-start lag L^s_k of the job in position k, $k \in J$, is defined as the time elapsed between the completion of the job on machine 1 and its start on machine 2:

$$L^s_k = L^c_k - p^2_{[k]} = \max\left\{0, L^s_{k-1} + p^2_{[k-1]} + s^2_{[k]} - p^1_{[k]}\right\}.$$

In order to write a convenient MILP model, the objective function is rewritten as:

$$\begin{aligned} \sum_k C^2_{[k]} &= \sum_k (C^1_{[k]} + L^c_k) \\ &= \sum_k \left((n-k+1)p^1_{[k]} + L^s_k + p^2_{[k]}\right). \end{aligned}$$

Let a binary variable x_{jk}, $j, k \in J$, determine whether job j is processed in position k in the schedule. Let a continuous variable L^s_k, $k \in J$, represent the completion-to-start lag of the job in position k