The two-machine flowshop total completion time problem: A branch-and-bound based on Network-flow formulation

Boris Detienne, Ruslan Sadykov, Shunji Tanaka

MISTA 2015

The two-machine flowshop total completion time problem: A branch-and-bound based on Network-flow formulation

Boris Detienne · Ruslan Sadykov · Shunji Tanaka

1 Introduction

We consider the problem of scheduling a set $J = \{1, \ldots, n\}$ of jobs in a two-machine flowshop with the objective to minimize the sum of the 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 the 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 $F2||\sum C_j$. It is known to be NP-hard in the strong sense [6]. It has been shown by Conway et al. [3] that there exists at least one optimal solution where both machines have the same sequence of the jobs. Thus, we may restrict the search to permutation schedules only.

The problem $F2||\sum C_j$ has been studied in the literature for many years. Akkan and Karabati [1] suggest a network flow formulation for the problem. They use a Lagrangian relaxation to obtain a lower bound which is used inside a branch-and-bound algorithm. This algorithm 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). In this work, we propose an improved branch-and-bound algorithm for the problem $F2||\sum C_j$ based on their work. To obtain stronger dual bounds, we use a network which is larger than the one used in [1]. 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 the dual bound only once in the root, and then recompute the bound in linear time at every node of the enumeration tree. Thus, tens

Boris Detienne
INRIA team RealOpt and Mathematics Institute, Bordeaux University, Talence, France
E-mail: boris.detienne@math.u-bordeaux1.fr

Ruslan Sadykov
INRIA team RealOpt and Mathematics Institute, Bordeaux University, Talence, France
E-mail: ruslan.sadykov@inria.fr

Shunji Tanaka
Institute for Liberal Arts and Sciences, Kyoto University, Japan
E-mail: tanaka@kuee.kyoto-u.ac.jp

of millions of nodes can be checked in a reasonable time. Using the proposed algorithm, we were able to solve all instances of the problem $F2||\sum C_j$ with up to 100 jobs with large processing times.

2 Network formulation

In the following, $[k]$ denotes the index of the job in position k. The completion times $C^m_{[k]}$ of the job in position k, $k \in J$, on machines $m = 1, 2$ can be computed as $C^1_{[k]} = C^1_{[k-1]} + p^1_{[k]}$ and $C^2_{[k]} = \max\{C^1_{[k]}, C^2_{[k-1]}\} + p^2_{[k]}$

In [1], the authors introduce the notion of time lag between the processing of a same job on both machines to write an assignment model and a network flow model for the problem. This kind of models is also called *waiting time-based* [models] in [7]. 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 : $L^c_k = C^2_{[k]} - C^1_{[k]} = \max\{0, L^c_{k-1} - p^1_{[k]}\} + p^2_{[k]}$. In order to design a convenient network model, the objective function can be expressed as:

$$\sum_k C^2_{[k]} = \sum_k (C^1_{[k]} + L^c_k) = \sum_k \left((n-k+1)p^1_{[k]} + L^c_k\right) \tag{1}$$

Our model is based on a transshipment type network $G(V, A)$, which extends the one proposed in [1]:

- Each node $v_{k,l,i} \in V$ of the network is associated with one position k in the sequence, and the start of job i when the completion-to-completion lag of the previous job is l. Two dummy nodes $(0, 0, \emptyset)$ and $(n+1, \emptyset, \emptyset)$ are added, representing the start and the end of the schedule, respectively.
- Each arc $(v_{k,l_1,i_1}, v_{k+1,l_2,i_2}) \in A$ from node v_{k,l_1,i_1} to node v_{k+1,l_2,i_2} is associated with the processing of job i_1 in position k, when the completion-to-completion lag of the previous job is equal to l_1, so that job i_1 ends with a completion-to-completion lag equal to l_2 and is immediately followed by job i_2. According to the expression of the objective function given by (1), the cost of using the arc is $c(v_{k,l_1,i_1}, v_{k+1,l_2,i_2}, j) = (n-k+1)p^1_{i_1} + l_2$.

The scheduling problem can be seen as the problem of finding a minimum cost flow of value 1 (a path) from the source node to the sink node, going through exactly one arc associated with each job.

Network reduction By dualizing these job occurence constraints, we obtain a Lagrangian lower bound. Given a vector of Lagrange multipliers, this bound can be computed by solving a simple shortest path problem in G. Using the same idea, a lower bound of the length of a feasible path that passes through a node (resp. an arc) is computed and the node (resp. the arc) is removed from the network if it is greater than an upper bound of the path length. This lower bound can be computed by applying dynamic programming in both forward and backward manners when the shortest path problem is solved [8]. More reductions are obtained by reinforcing the job assignment constraint in the shortest path problem for one job at a time [5]. Moreover, several dominance rules from [10, 4, 2] are used to remove some arcs in the graph.

3 Branch-and-bound algorithm

The set of possible job sequences is explored, by enumerating the set of feasible (with respect to the job occurence constraint) paths in graph G. We proceed from the left to the right in the graph. We perform a Depth-First-Search. In a preprocessing stage, an upper bound is computed using a dynasearch procedure [9], and graph G is reduced using a subgradient procedure inside which the network reduction procedures are applied. For each job j and node v of G, we compute the cost of a shortest path from v to the sink node going through exactly one arc representing j, as well as the cost of one going through no arc representing j. In order to evaluate a partial sequence σ represented by a path ending at node v, we compute a lower bound of the cost of extending σ into a feasible schedule. For each job j, we derive in constant time a lower bound in which the job assignment constraint is enforced for j. A job is a candidate for the next job in the sequence only if there is a corresponding arc in G and the resulting subsequence is not dominated according to several dominance rules, coming from the literature or extending some of them. Candidate jobs are processed in non-decreasing order of the distance from the corresponding terminal node to the sink.

4 Numerical results

The branch-and-bound algorithm solves to optimality all instances of our test bed, composed of randomly generated instances with up to 100 jobs with up to 100-unit long processing times (as in [1]). The hardest instance is solved in 7759 seconds, while all the others are solved in less than one hour on a laptop equipped with a 2.7GHz processor and 4GB RAM. The average computing time for 100-job instances is 502.6 seconds, and the average size of the search tree is 128.8 millions of nodes.

References

1. C. Akkan and S. Karabati. The two-machine flowshop total completion time problem: Improved lower bounds and a branch-and-bound algorithm. *European Journal of Operational Research*, 159(2):420–429, December 2004.
2. B.W. Cadambi and Y.S. Sathe. Two-machine flowshop scheduling to minimize mean flow time. *Opsearch*, 30(1):35–41, 1993.
3. R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, MA, 1967.
4. F. Della Croce, V. Narayan, and R. Tadei. The two-machine total completion time flow shop problem. *European Journal of Operational Research*, 90(2):227 – 237, 1996.
5. B. Detienne, S. Dauzère-Pérès, and C. Yugma. An exact approach for scheduling jobs with regular step cost functions on a single machine. *Computers & Operations Research*, 39(5):1033–1043, 2012.
6. M. R. Garey, D. S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.
7. A. Gharbi, T. Ladhari, M. K. Msakni, and M. Serairi. The two-machine flowshop scheduling problem with sequence-independent setup times: New lower bounding strategies. *European Journal of Operational Research*, 231(1):69–78, November 2013.
8. T. Ibaraki and Y. Nakamura. A dynamic programming method for single machine scheduling. *European Journal of Operational Research*, 76(1):72–82, July 1994.
9. S. Tanaka. An extension of the dynasearch to the two-machine permutation flowshop scheduling problem. In *Proceedings of the 2010 International Symposium on Flexible Automation*, page 6, 2010.
10. S.L. van de Velde. Minimizing the sum of the job completion times in the two-machine flow shop by lagrangian relaxation. *Annals of Operations Research*, pages 257–268, 1990.