Exact Method For Hybrid Flowshop With Batching Machines And Tasks Compatibilities

Adrien Bellanger, Ammar Oulamara

Exact Method For Hybrid Flowshop With Batching Machines And Tasks Compatibilities

Bellanger, A. and Oulamara, A.

LORIA - INRIA Nancy Grand Est, Ecole des Mines de Nancy
Parc de Saurupt, 54042 Nancy, France.
e-mails : Adrien.Bellanger@loria.fr Ammar.Oulamara@loria.fr

Keywords: Branch and bound, hybrid flowshop, batching machines, tasks compatibilities.

1 Introduction

In tire manufacturing industry, tire production is commonly represented as a two-stage process. First and second stages represent tire building and tire curing process, respectively. Tire building is made on one machine, all components (sidewalls and tread) are assembled on round drums in order to build radial tires. The assembled product is called a *green tire* or *uncured tire*. Tire curing is an high-temperature and high-pressure batch operation, in which a pair of green tires is placed into a mold at a specified temperature. Two kinds of tire can be cured together if they share a same value of total curing duration. The mold cannot be opened until the curing reaction is completed for the both green tires on the same mold.

This industrial problem is modeled as a two-stage hybrid flowshop problem where n jobs are to be processed first at stage 1 and then at stage 2. The first stage consists of m_1 parallel machines and the second stage contains m_2 parallel batching machines. Each job j ($j = 1, \dots, n$) has processing time p_j at stage 1 and processing time q_j lies in interval $[a_j, b_j]$ at stage 2. At stage 2 jobs are processed in batch, and all jobs of the same batch start and finish together and have to be compatible. A pair of job (i, j) is compatible if they share a similar processing time at second stage, i.e. $[a_i, b_i] \cap [a_j, b_j] \neq \emptyset$, therefore the processing time of a batch k is determined as the maximal initial endpoint a_j of jobs contained in batch k. The objective is to find a schedule which minimizing the completion time of the latest batch. This problem is noted $FH2(m_1, m_2)|p\_batch(II), G_P = INT, k < n|C_{max}$ by the standard notation introduced by Graham, (1979).

Several studies dealing with exact approach of hybrid flowshop were proposed in the literature (Carlier, (2000), Moursli, (2000), Houari, (2006)). A survey of different solving approach for hybrid flowshop is published by Kis, (2004).

In this paper we present a Branch & Bound method for a two-stage hybrid flowshop in order to minimize the makespan.

2 Branch and bound method

In our branch and bound method, we consider each stage of hybrid flowshop separately. In other words, the first step of branch and bound considers the scheduling problem of parallel machines and the second step treats the scheduling problem of parallel batching machines with release dates r_j, where each terminal node of the branch and bound tree represents a solution of the problem $PB(m_2)|r_j|C_{max}$.

Let Σ be the set of feasible schedules at the first stage. Each $\sigma \in \Sigma$ induces a completion time $C_j^1(\sigma)$ for each job $j \in J$. Then for a given $\sigma \in \Sigma$, one considers the problem $PB(m_2)|r_j, G_P = INT, k < n|C_{max}$ obtained by setting $r_j = C_j^1(\sigma)$, $\forall j \in J$. Let $C_{max}(\sigma)$ be the optimal makespan value of problem $PB(m_2)|r_j, G_P = INT, k < n|C_{max}$ for a given $\sigma \in \Sigma$, then clearly solving the hybrid flowshop problem is equivalent to finding a schedule $\sigma^* \in \Sigma$ which is satisfying the following equation $\tilde{C}_{max}(\sigma^*) = \min_{\sigma \in \Sigma} \tilde{C}_{max}(\sigma)$.

2.1 First step

In the first step of the branch and bound procedure, we generate a set Σ of schedules for the first stage of parallel machines problem. The set Σ should contains all non dominated schedules. Each schedule σ of Σ is represented by a list $\sigma = (\sigma_1, \ldots, \sigma_n)$, and conversely each list represents an unique schedule. In fact, for a given list, a schedule is obtained by assigning the first task to the first available machine, then removing this task from the list and assigning the next task of the list to the new available machine and so on.

Consider the branch and bound tree, the root node N_0 is set to an empty list. A node N_l^1 of level l corresponds to the partial schedule $\sigma(N_l) = (\sigma_1, \ldots, \sigma_l)$ of l jobs at the first stage. Each Node of level n corresponds to a complete schedule at the first stage. In order to reduce the number of nodes, we develop different lower bounds, dominance rules and an adjustment procedure called FAP (Feasibility and Adjustment Procedure) which was introduced by Gharbi, (2002).

Lower bounds Consider the node N_l, let J be the set of scheduled jobs and let $\bar{J}$ be the set of remaining jobs. Denote by $t_1, \ldots, t_{m1}$ the availability time on which machines $1, \ldots, m_1$ become ready for processing jobs of $\bar{J}$. The first lower bound is a job-based lower bound, i.e. $LB_1 = \max_{i \in \bar{J}}(t_k + p_i + a_i)$ where $t_k = \min\{t_i | i = 1, \ldots, m_1\}$. The second lower bound is a machine-based lower bound, i.e. $LB_2 = t_k + p_k + \frac{C_{max}^{FBCLPT}(\bar{J})}{m_2}$ where $p_k = \min\{p_j | j \in \bar{J}\}$ and $C_{max}^{FBCLPT}(\bar{J})$ is the minimal total load on batching machines of unscheduled jobs $\bar{J}$, obtained by using the FBCLPT algorithm, Oulamara, (2007).

Upper bounds To help the search in our branch and bound algorithm, the makespan obtained from the heuristic based on Johnson rules (Johnson, (1954)) that we have developped in Bellanger, (2007), is used as an initial upper bound, and when a better schedule is obtained the value of the upper bound is updated.

Dominance rules In order to speeding up the branch and bound algorithm, we propose dominance rules to remove dominated nodes from the set of candidate nodes to be branched. In our branching scheme, we use the following rules on both stages,

- If two jobs j_k and j_{k+1} have equal processing times on both stages, then job j_{k+1} is not a candidate to be add.
- If previous machine starts by task j_k, then current machine has to start by a task j_l with $l > k$.

2.2 Second step

For a given leaf node N' of the first step of the branch and bound, a complete schedule of the jobs is obtained and for each job we have its completion time at first stage, which can be considered as a release date for the second stage. The second step of the branch and bound procedure considers the problem $PB(m_2)|r_j, G_P = INT, k < n|C_{max}$. The node N' is considered as the root of the second step of the tree and each node describes a list of batches. In other words, a node N_l^2 of level l of the step two, describes a list of batches constructed by adding batch $B_{N_l^2}$ of unscheduled jobs, thus all combinaison of compatible jobs are considered. To limit the size of the tree we use similar lower bounds described for the first step of the branch and bound. Specific dominance rules which considering batch properties are also used to limit the size of the tree, which are given in the following,

- If there is an available job, which could be added to the current batch B without any change on the processing time of B, B is not candidate to extend the list of batches.
- Let $cb_j = \max\{t_k, r_j\} + a_j$ be the smallest completion time of a candiate batch $\bar{B}$ composed by an unscheduled jobs and contains an unscheduled job j, if an other batch B has a release date greatest than cb_i, then B is not candidate to extend the list of batches.

3 Preliminary computational results

In order to evaluate the performance of this branch and bound algorithm, we carried out series of preliminary experiments. The algorithm is coded in C++ language, and runs on an Intel Pentium M 1,5 GHz and 512 MB RAM. The processing times of jobs at the first stage are generated from an uniform distribution $[5, 100]$, and at the second stage the initial endpoints a_j of the processing time intervals are generated from an uniform distribution $[5, 100]$, where the terminal endpoints b_j are given as $b_j = a_j + 0.05 \times a_j$. We tested instances with 5 machines at each stage and the capacity of batch is equal to 2. The preliminary experiments show that instances with 12 tasks are solved easily in a few seconds. More experiments should be proposed in the final presentation of this paper.

References

1. A. Bellanger, et A. Oulamara. (2007) Flowshop hybride avec machines à traitement par batch et compatibilité entre les tâches. *FRANCORO V / ROADEF'07*, Livre des articles : 21-35.
2. J. Carlier and E. Néron. (2000) An exact method for solving the multiprocessor flowshop. *RAIRO-Oper Res.* 78 : 146-161.
3. A. Gharbi, M. Haouari. (2002) Minimizing makespan on parallel machines subject to release dates and delivery times. *J. Scheduling* 5:329-355.
4. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rimooy Kan. (1979) Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, 5 : 287-326.
5. M. Haouari, A.Gharbi. (2006) Optimal scheduling of a two-stage hybrid flow shop. *Math. Meth. Oper. Res.* 64 : 107-124.
6. T. Kis, et E. Pesch. (2005) A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *European Journal of Operational Research* 164 : 592-608.
7. O. Mousli and Y. Pichot. (2000) A branch and bound algorithm for the hybrid flowshop. *Int. J. Product. Econ.* 64: 113-125.
8. A. Oulamara, and Gerd Finke, A. Kamgaing Kuiten. (2007) Flowshop Scheduling problem with Batching Machine and Task Compatibilities. *Computers & Operations Research*, In press,