Solving the Two-Stage Hybrid Flow Shop Problem by a Branch-and-bound Algorithm

Mohamed Haouari[1] - Lotfi Hidri[1] - Anis Gharbi[1,2]

[1] Combinatorial Optimization Research Group - ROI, Ecole Polytechnique de Tunisie, BP 743, 2078 La Marsa, Tunisia.

[2] Department of Applied Mathematics, Institut Supérieur d'Informatique, Ariana, Tunisia.

1 Introduction

A set J of n jobs has to be scheduled in a manufacturing system with two stages (machining centers) Z_1 and Z_2. Each stage Z_i $(i = 1, 2)$ has m_i identical machines in parallel (with $\max(m_1, m_2) \geq 2$). Each job j $(j = 1, ..., n)$ has to be processed first for a_j units of time by one machine of Z_1, and then for b_j units of time by one machine of Z_2. These operations must be processed without preemption. Moreover, a job cannot be processed by more than one machine at the same time and each machine processes at most one job at one time. All processing times are assumed to be deterministic and integer and all machines are ready from time zero onwards. The objective is to construct a schedule for which the maximum completion time, or makespan, is minimized. This problem, denoted by $F2(P)\,||\,C_{\max}$, is strongly $\mathcal{NP}$-hard.

The theoretical and practical importance of the $F2(P)\,||\,C_{\max}$ motivated several researchers to investigate it. In particular, most efforts have been focused on developing and analyzing heuristic algorithms with worst-case error bounds. On contrast, the literature dealing with the exact solution of $F2(P)\,||\,C_{\max}$ is surprisingly scant. Indeed, the only relevant work that we are aware is the branch-and-bound algorithm described by Gupta et al. [4] who addressed the particular case where the second stage contains a single machine and presented computational tests with up to 250 jobs. In addition, several authors developed exact procedures for the multiple-stage hybrid flow shop problem. In particular, Néron and his colleagues [6] describe an exact approach which outperforms all previous ones and report the optimal solution of small-sized instances with up to 15 jobs, 5 stages, and 3 machines in each stage.

In this paper, we present an effective branch-and bound algorithm which has been specifically designed for solving the $F2(P)\,||\,C_{\max}$ problem with an arbitrary number of machines in each stage. However, although our approach could be easily modified to handle the particular case where one of the two stages contains a single machine, we assume, for the sake of simplicity, that each stage contains at least two parallel machines (i.e. $\min(m_1, m_2) \geq 2$). A distinctive feature of our branch-and-bound is that the evaluation of terminal nodes of the search tree requires the optimal solution of a $P\,|r_j|\,C_{\max}$. However, although this problem is known to be intractable, we provide evidence that its hardness doesn't preclude its effectiveness for lower bound computation. Other features that are peculiar to our procedure include a branching strategy that is based on a representation of a $F2(P)\,||\,C_{\max}$ solution as a permutation of jobs, tight lower and upper bounding procedures, dominance rules, and procedures for adjusting heads and tails. Our algorithm has produced proved optimal solutions for a number of randomly generated instances with up to 1000 jobs.

2 An overview of the branch-and-bound algorithm

2.1 Problem representation

It is instructive to view the $F2(P)\,||\,C_{\max}$ in another way: as an identical parallel machine scheduling problem with a complex optimality criterion. To develop this interpretation, let Σ denote the set of feasible schedules of stage Z_1. Obviously, each $\sigma \in \Sigma$ induces a well-defined completion time $C_j^1(\sigma)$ for each $j \in J$. For a given $\sigma \in \Sigma$, consider the $Pm_2\,|r_j|\,C_{\max}$ that is obtained by setting for all $j \in J$ a release date $r_j = C_j^1(\sigma)$ and a processing time $p_j = b_j$. Let $\tilde{C}_{\max}(\sigma)$ denote its optimal makespan. Clearly, the $F2(P)\,||\,C_{\max}$ amounts to finding a schedule $\sigma^* \in \Sigma$ satisfying $\tilde{C}_{\max}(\sigma^*) = \min_{\sigma\in\Sigma} \tilde{C}_{\max}(\sigma)$.

A schedule $\sigma \in \Sigma$ could be represented as a permutation of the n jobs. This permutation is simply obtained by ranking the jobs according to the nondecreasing order of their starting times. Conversely, given a permutation of the n jobs $(\sigma_1, \sigma_2, ..., \sigma_n)$, the starting times of the associated schedule are computed in $O(n)$ time using the *list scheduling* rule which successively schedules the jobs $\sigma_1, \sigma_2, ..., \sigma_n$, in that order, whenever a machine becomes idle. It is noteworthy that Carlier and Néron [2] and Néron et al. [6] proposed a similar schedule representation but as a permutation of operations rather than jobs.

2.2 Branching scheme

Since each feasible schedule could be represented as a permutation of n jobs, we adopted the following branching scheme. Each node N_l of level l of the search tree corresponds to a partial permutation (i.e. schedule) $\sigma(N_l) = (\sigma_1, \sigma_2, ..., \sigma_l)$ of l jobs. Therefore, the corresponding set of unscheduled jobs is $\bar{J}(N_l) = J \setminus \{\sigma_1, \sigma_2, ..., \sigma_l\}$. Obviously, the root node N_0 corresponds to the empty permutation. Each node N_l has $n-l = \left|\bar{J}(N_l)\right|$ descendants. Each of these descendants corresponds to a partial permutation $(\sigma_1, \sigma_2, ..., \sigma_l, j_0)$ where $j_0 \in \bar{J}(N_l)$. In this way, a node at level $n-1$ corresponds to a well defined schedule of the first stage. In our branch-and-bound algorithm, we adopted the depth-first strategy.

2.3 Upper bounds

Two heuristics were implemented for delivering an upper bound on the optimal makespan. The first one is only used at the root node for generating an initial upper bound. It has been designed in the same vein as the celebrated Shifting Bottleneck Procedure [1]. Basically, it consists in alternatively solving a parallel machine problem on stage Z_1 and on stage Z_2 until a termination condition holds. In our implementation, the parallel machine problems are solved using the branch-and-bound algorithm described in [3]. By interchanging the roles of stages Z_1 and Z_2, a second upper bound is computed in a similar way. We take the best of the two derived solutions. The second heuristic, which is a very fast priority-rule based heuristic, is called at each node of the search tree.

2.4 Lower bounds

2.4.1 Lower bounds that are computed at the root node

- Let $C_{\max}^i$ $(i = 1, 2)$ denote the optimal makespan of the parallel machine problem obtained by relaxing the constraint that each machine of stage i can process at most one job at a time. Hence, a valid lower bound on the optimal makespan of the $F2(P)\,||\,C_{\max}$ is $LB_1 = \max(C_{\max}^1, C_{\max}^2)$. The computation of LB_1 requires the optimal solution of two $\mathcal{NP}$-hard problems : a $Pm_1|q_j|C_{\max}$ and a $Pm_2|r_j|C_{\max}$. In our implementation, LB_1 is obtained as a by-product of the heuristic that is used in the root node.

- Following Haouari and M'Hallah [5], the total idle time in stage Z_2 for scheduling a subset S of jobs is at least equal to $I_2(S)$ which corresponds to the minimum sum of completion times, on

stage Z_1, of the m_2 jobs of S whose processing times are the shortest. Clearly, $I_2(S)$ can be obtained by applying the *Shortest Processing Time*-rule. Hence, a valid lower bound is $LB^2_{SPT} = \max_{S\subseteq J}\left\{\left\lceil (I_2(S)+\sum_{j\in S} b_j)/m_2\right\rceil\right\}$. By using the symmetry of the hybrid flow shop problem, we get the lower bound $LB^1_{SPT} = \max_{S\subseteq J}\left\{\left\lceil (I_1(S)+\sum_{j\in S} a_j)/m_1\right\rceil\right\}$. It is shown that the computation of $LB_2 = \max(LB^1_{SPT}, LB^2_{SPT})$ amounts to solving a longest path problem in a digraph. Therefore, LB_2 can be computed in $O(n^2 \max(m_1, m_2))$ time.

2.4.2 Lower bounds that are computed at non-root nodes

Assume that at a given node $N \neq N_0$ of the search tree, a set J_S of jobs have been already scheduled and define $\bar{J} = J \setminus J_S$. Each job $j \in J_S$ has a well defined completion time on stage Z_1 which is denoted by C_{1j}. Also, each machine M_i $(i = 1, ..., m_1)$ of the first stage has an availability time τ_i on which it becomes ready for processing jobs from $\bar{J}$. We assume that $\tau_1 \leq \tau_2 \leq ... \leq \tau_{m_1}$.

- Clearly, a valid relaxation is a $P\,|r_j|\,C_{\max}$ which is defined on the second stage and where each job j has a release date r_j such that $r_j = C_{1j}$ if $j \in J_S$ and $r_j = a_j + \tau_1$ otherwise. For a given subset $S \subseteq J$, define $\bar{r}_k(S)$ as the k^{th} smallest release date of S. The value $LB_3 = \max_{S\subseteq J}\left\{\left\lceil (\sum_{k=1}^{m_2} \bar{r}_k(S) + \sum_{j\in S} b_j)/m_2\right\rceil\right\}$ is a valid lower bound on the subproblem corresponding to node N. LB_3 can be computed in $O(n \log m_2)$ time.
- A relaxation of the hybrid flow shop problem is derived by setting for each job $j \in \bar{J}$ a tail $q_j = b_j$. In this way, we define on stage Z_1 a parallel machine problem with machine availability times and tails $(P, NC_{inc}\,|q_j|\,C_{\max})$. It is worth noting that due to availability times, some machines may not process any job in any optimal solution. Let UB denote an upper bound on the optimal makespan of the $P, NC_{inc}\,|q_j|\,C_{\max}$. Gharbi and Haouari [3] prove that the number of machines m that are processing in an optimal schedule satisfies $m_l(\bar{J}) \leq m \leq m_u(\bar{J})$, where $m_l(\bar{J}) = \left\lceil \sum_{j\in\bar{J}} a_j/(UB - \tau_1 - \min_{j\in\bar{J}} q_j)\right\rceil$ and $m_u(\bar{J})$ is the smallest k $(k = 1, ..., m_1 - 1)$ satisfying $\tau_{k+1} + \min_{j\in\bar{J}}(a_j + q_j) > UB$. Assume that the jobs of $\bar{J}$ are assigned to exactly m machines of stage Z_1, then a valid lower bound is $LB_4(m) = \max_{S\subseteq\bar{J}}\left\{\left\lceil (\sum_{i=1}^{m}\tau_i + \sum_{j\in S} a_j + \sum_{k=1}^{m} \bar{q}_k(S))/m\right\rceil\right\}$ where $\bar{q}_k(S)$ is defined as the k^{th} smallest tail of S. Provided that the jobs are sorted according to non decreasing tails, the lower bound $LB_4 = \min_{m_l(\bar{J})\leq m\leq m_u(\bar{J})} LB_4(m)$ can be computed in $O(m_1 n \log m_1)$ time.

2.5 Further enhancements

We extended the dominance rules as well as the so-called Feasibility and Adjustment Procedure (which were successfully used for the parallel machine problem [3]) to deal with the $F2(P)\,||\,C_{\max}$. The latter procedure aims at adjusting the heads and tails, and checking the feasibility of a nonpreemptive schedule. Also, in order to take advantage of the symmetry of the $F2(P)\,||\,C_{\max}$, we propose a cyclic implementation of our branch-and-bound algorithm. It consists in iteratively solving the original problem and its symmetric. The process continues until a solution is proved optimal or there is no improvement of neither the lower nor the upper bound.

3 Preliminary computational results

We evaluated the performance of our branch-and-bound algorithm on 3 sets of instances. The first set contains a diversified mix of shop and size configurations. For the second set, the workloads at the two stages tend to be well balanced while for the third set, the workloads are mostly unbalanced. The number

of jobs n is taken equal to 10, 20, 30, 40, 50, 100, 150, 200, 500, 750, and 1000. The algorithm was coded in C on a Pentium IV 2.8 GHz Personal Computer with 1 GB RAM. A CPU time limit was set equal to 600 seconds.

We found that the proposed algorithm can solve large scale instances within moderate CPU time. Indeed, it produced proven optimal solutions for 94% of the instances (3290 out of 3500). Most of the unsolved instances are of very large scale (48% are the 1000-job ones). For $n \leq 500$, the average computation time was no larger than 5 minutes. Furthermore, the average gap of the unsolved instances is strictly less than 0.28% for $n \geq 200$.

We observed that the hardest instances are those where the workloads in the two stages are balanced. Also, the problems get harder as the number of machines increases. On contrary, when the workloads are unbalanced, the problems are much easier to solve. For this problem class, branching was only required for very large instances ($n \geq 500$). Surprisingly, we found that solving a medium-sized balanced instances ($n =$ 20 or 30) could be more challenging than solving large-sized ones.

References

[1] Adams J, Balas E, Zawack D (1988) The shifting bottleneck procedure for job shop scheduling. Management Science 34:391-401.

[2] Carlier J, Néron E (2000) An exact method for solving the multiprocessor flowshop. RAIRO-Operations Research 34:1-25.

[3] Gharbi A, Haouari M (2005) Optimal Parallel Machines Scheduling with Availability Constraints. Discrete Applied Mathematics, in press.

[4] Gupta JND, Hariri AMA, Potts CN (1997) Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. Annals of Operations Research 69:171-191.

[5] Haouari M, M'Hallah R (1997) Heuristic algorithms for the two-stage hybrid flowshop problem. Operations Research Letters 21:43-53.

[6] Néron E, Baptiste Ph, Gupta JND (2001) Solving hybrid flow shop problem using the energetic reasoning and global operations. Omega 29:501-511.