Production of Prefabricated Wall Elements: Flow Shop with Multi-Task Flexibility

Gaia Nicosia[1], Andrea Pacifici[2], Ulrich Pferschy[3], Cecilia Salvatore[2]

[1] Università degli studi Roma Tre, Roma, Italy
gaia.nicosia@uniroma3.it
[2] Università di Roma Tor Vergata, Roma, Italy
{andrea.pacifici}{cecilia.salvatore}@uniroma2.it
[3] University of Graz, Graz, Austria
ulrich.pferschy@uni-graz.at

Abstract

We address a special flow shop scheduling problem arising in a production line of a company building prefabricated house walls. Each job requires the execution of a number of operations, in a strict order, by machines placed along a line. Some of these operations can be executed indifferently at one stage or the next one along the flow line. Both the assignment of operations and the scheduling of the jobs has to be decided in order to minimize the time needed to complete all the jobs. This problem may be viewed as a permutation flow shop with special additional blocking constraints and multi-task flexibility, viz. no buffer is available between consecutive machines in the line and certain machines at one stage can process a number operations of two other adjacent stages in the system.

In our study, we present four different Mixed Integer Programs and the outcome of computational tests which simulate realistic scenarios of the production environments. Experiments are designed in order to assess the performance of the models both in terms of computing time and effectiveness. Preliminary results show that embedding the operations-to-machine assignment in the scheduling decision variables guarantees satisfactory results when dealing with instances significant for the motivating application.

Keywords : *Scheduling, Mixed Integer Linear Programming, Multi-Task Flexibility.*

1 Introduction

The optimization problem addressed in this work is inspired by a material handling problem arising at the production line of a company building prefabricated house walls. The production line of the company is currently organized so that the house walls (the jobs) pass through five different work stations where several processing steps (operations) are carried out. The sequence of processing steps for the preparation of distinct walls present only small differences, which depend on the facilities placed on the wall itself. Hence, the stations are visited by each wall in a same sequence. In the following we will refer to the i-th visited station as machine M_i, $i = 1, 2, \ldots, 5$. Moreover, due to the large size and weight of the bulky walls, no intermediate storage is possible between the individual stations. As a consequence, the so-called blocking constraints occur: a wall cannot move to the next work station if this is busy thus blocking the upstream machines.

A peculiar characteristic of this problem is that some of the operations that have to be executed on each wall may be undertaken by any of two consecutive machines of the flow line. A graphic representation of this situation is depicted in Figure 1 which depicts possible assignments (indicated by dotted arrows) of the operations on the corresponding processing

machines. Jobs are comprised of a set of operations to be executed in a strict order. In the illustration, with no loss of generality, we aggregated operations that must be performed on a specific machine (namely, operations 1, 2, 10, 15, and 16, to be executed on M_1, M_2, M_3, M_4, and M_5, respectively). The other operations can be assigned to one of two consecutive machines: For instance, operations 11–14 can be assigned to machine M_3 as well as to M_4.

Therefore, the problem consists in simultaneously finding (i) the assignment of each operation to machines and (ii) the sequencing of the jobs so that makespan is minimized.

Problems in which, in a flow line, certain operations can be assigned to at least two consecutive stages/machines are called scheduling problems with *multi-task flexibility* or *inter-stage flexibility*, i.e., flexible operation-to-machine assignment [1, 3, 4].

In [4] the authors address a permutation flowshop with m machines and $q > m$ operations in which each job has different tasks to be processed that need to be assigned to the workstations and, as in our case, such assignment is restricted by the order of operations. The authors propose a heuristic algorithm to solve the problem.

Other papers, such as [2, 3], provide complexity characterization of flowshop scheduling problems with multi-task flexibility on 2 or 3 machines.

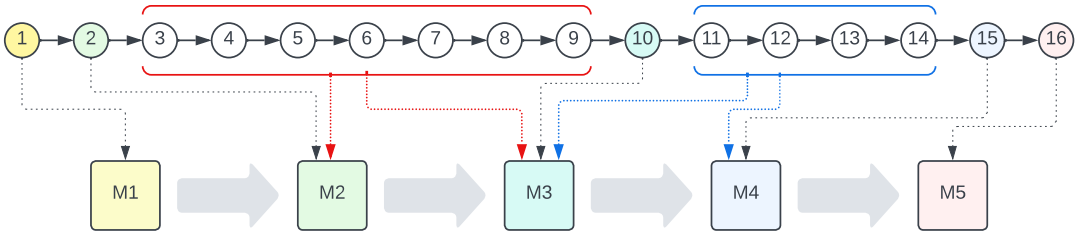

FIG. 1: Feasible operation-to-machine assignments for a single job.

2 Mathematical Programs

In this study we compare four Mixed Integer linear Programs (MIP) for our special flow shop scheduling problem. We are using two standard ways to represent the sequencing of the jobs that make use of different set of variables: In the first one, each job is assigned to a "position" in the sequence strict order. The second one models precedence relations between job pairs. We refer to those models as positional- (π) or precedence- (α) variables models.

The first two models address multi-task flexibility by using variables (x) that explicitly model the decision of performing one operation on a specific machine. Hereafter, we detail one of the models in Section 2.1 and, in the remainder of the paper, we limit ourselves to briefly sketching the main distinct features of the three remaining models w.r.t. the first one.

Hereafter, $\mathcal{M}$ is the set of m $(= 5)$ machines in the flow line; $\mathcal{J}$ is the set of n jobs: Each job j consists of an ordered sequence $\mathcal{O}_j$ of q operations. The i-th operation of job $j \in \mathcal{J}$ is denoted by $j(i)$. Set $M_{j(i)}$ contains (at most two consecutive) machines that can process $j(i)$. The processing time of $j(i)$ on machine k is $p^k_{j(i)}$. Moreover, $\mathcal{H} = 1, \ldots, n$ is the set of positions in the sequence.

2.1 Positional Variables Model

We consider the following sets of binary variables: π_{jh} indicates whether job j is processed as the h-th job of the sequence (hereafter, the job "in position" h) while x_{ijk} if operation i of job j is assigned to machine k to be processed. We also use two sets of continuous variables: s_{hk} is the starting time of the job in position h on machine k and P_{hk} represents the total processing

time of the operations of job in position h assigned to machine k. The above variables are defined for all $h \in \mathcal{H}$, $i \in \mathcal{O}(j)$, $j \in \mathcal{J}$, $k \in \mathcal{M}$.

The objective function, to be minimized, is the makespan $C_{\max} = s_{nm} + P_{nm}$, that is the completion time of the last job on the last machine. In order to guarantee that exactly one job is placed in each position and that each job is in a position we use standard assignment constraints $\sum_{h \in \mathcal{H}} \pi_{jh} = 1$ for all $j \in \mathcal{J}$ and $\sum_{j \in J} \pi_{jh} = 1$ for $h \in \mathcal{H}$. We also ensure that each operation is executed on (exactly) one machine with $\sum_{k \in M_{j(i)}} x_{ijk} = 1$ for all operations $i \in \mathcal{O}_j, j \in \mathcal{J}$.

We need to impose that the sequence of operations on the machines is respected: this is enforced by the constraints $x_{i+1jk} + x_{ij\hat{k}} \leq 1$, defined for all $i \in \mathcal{O}_j$, $j \in \mathcal{J}$, $k \in M_{j(i+1)}$, $\hat{k} \in M_{j(i)}$ with $k < \hat{k}$.

The following sets of additional constraints are discussed hereafter.

$$s_{hk+1} \geq s_{hk} + P_{hk} \qquad k \in \mathcal{M} \setminus \{m\}, h \in \mathcal{H} \tag{1}$$

$$s_{h+1k} \geq s_{hk} + P_{hk} \qquad k \in \mathcal{M}, h \in \mathcal{H} \setminus \{n\} \tag{2}$$

$$s_{hk} \geq s_{h-1k+1} \qquad k \in \mathcal{M} \setminus \{m\}, h \in \mathcal{H} \setminus \{1\} \tag{3}$$

$$P_{hk} \geq \sum_{j(i) \,:\, k \in M_{j(i)}} p^k_{j(i)} x_{ijk} + - B(1 - \pi_{jh}) \qquad h \in \mathcal{H}, k \in \mathcal{M}, j \in \mathcal{J} \tag{4}$$

Inequalities (1) and (2) establish the correct sequence of jobs w.r.t. machines. The blocking constraints are set with equations (3) imposing that the job in position h may start being processed on machine k only if the preceding job in position $h-1$ has moved on and started its processing on machine $k+1$. Finally, the latter set of constraints (4), in which we may choose the "large" constant $B = \max_{j,k}\{\sum_{j(i):k \in M_{j(i)}} p^k_{j(i)}\}$, returns the correct quantities for the processing time of the job placed in position h on machine k. Binary variables π_{jh} and x_{ijk} are defined for all $i \in \mathcal{O}_j$, $j \in \mathcal{J}$, $h \in \mathcal{H}$, and $k \in \mathcal{M}$, whereas continuous nonnegative variables s_{hk} and P_{hk} for $k \in \mathcal{M}$ and $h \in \mathcal{H}$.

2.2 Precedence Variables Model

In this standard MIP model, a solution schedule is defined by a set of binary variables α_{ij}, each associated to a pair of jobs i, j, controlling the precedence relation between these two jobs. In addition, we use job starting-time variables $s_j \geq 0$, for all $j \in \mathcal{J}$ and an additional $C_{\max}$ variable to record the makespan. The above assignment variables x_{ijk} and the corresponding constraints ensuring that each operation is assigned to one machine are still used in this model.

2.3 Models with Implicit Assignment

In our particular setting, the number of possible assignments is limited: Respectively, 7 and 4 operations can be assigned to M_2 or M_3 and M_3 or M_4 (see Figure 1) yielding to a set of feasible assignment modalities (hereafter denoted as $\mathcal{A}$) having cardinality equal to 40. On these grounds, and due to unsatisfactory results of some preliminary tests with the above two models, we design an additional pair of MIP models in which the operation-to-machine assignment is taken into account *implicitly*, as we discuss in the following section.

In these models, we look at the job sequences for each possible assignment modality $l \in \mathcal{A}$. The latter set $\mathcal{A}$ is computed in a preprocessing phase together with all the processing times p_{jkl} of each job $j \in \mathcal{J}$ on all machines $k \in \mathcal{M}$, under all assignment modes $l \in \mathcal{A}$.

Similar to the MIPs introduced in Section 2.1 and 2.2, we use $|\mathcal{A}|$ binary variables for each job pair: π_{jhl} indicates whether job j is processed in the h-th position when the assignment modality is l. Similarly, in the model with precedence variables, we need to define $n^2|\mathcal{A}|$ binary variables α_{ijl} indicating if job i precedes job j under the assignment modality l.

In both the above models, it is necessary to select one among the $|\mathcal{A}|$ possible assignments. Note that, in general, these models require an exponential number of variables, as the number of feasible operation-to-machine assignments is $O(|\mathcal{M}|^q)$.

3 Preliminary Computational Experiments

Preliminary tests have been executed on a standard PC with 16GB RAM, 2.80GHz CPU, and the Gurobi 10.0.1 solver. At the moment the experiments consist of four classes with 10, 15, 20, and 25 jobs, each comprised of 10 randomly generated instances.

Our results illustrate the difficulty of the problem. Within a time limit of half an hour per instance, already with 15 jobs some of the four models are not able to find an optimal solution for all instances. Indeed, the precedence-variables models are less effective compared to those with positional-variables. It is also evident that the two implicit-assignment models outperform the corresponding models adopting explicit assignment variables. The positional-variables model with implicit assignment is the clear winner, as it closes the gap on all instances up to 20 jobs and on 6 out of 10 instances with 25 jobs. The relative gaps from the best lower bounds do not exceed 3.5% for this model (and, in any case, it stays below 4.5% for all four models in all the tests).

Current experiments, concerning the best model only, aim at testing a procedure pairing the MIP with a preprocessing heuristic that selects—based on the instance—a suitable subset of the assignment modes in order to reduce the size of the model itself, so to hopefully improve its performance.

References

[1] Mohamed-Naceur Azaiez, Anis Gharbi, Imed Kacem, Yosra Makhlouf, and Malek Masmoudi. Two-stage no-wait hybrid flow shop with inter-stage flexibility for operating room scheduling. *Computers & Industrial Engineering*, 168:108040, 2022.

[2] Federico Della Croce, Fabio Salassa, and Vincent T'kindt. Exact solution of the two-machine flow shop problem with three operations. *Computers & Operations Research*, 138:105595, 2022.

[3] Danial Khorasanian and Ghasem Moslehi. Two-machine flow shop scheduling problem with blocking, multi-task flexibility of the first machine, and preemption. *Computers & Operations Research*, 79:94–108, 2017.

[4] Alex J. Ruiz-Torres, Johnny C. Ho, and José H. Ablanedo-Rosas. Makespan and workstation utilization minimization in a flowshop with operations flexibility. *Omega*, 39(3):273–282, 2011.