Operations Research Letters 43 (2015) 317–322

Contents lists available at ScienceDirect

Operations Research Letters

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

Minimizing the number of switch instances on a flexible machine in polynomial time

David Adjiashvili [*], Sandro Bosio, Kevin Zemmer

Institute for Operations Research, ETH Zürich, Switzerland

ARTICLE INFO

Article history:
Received 26 August 2014
Received in revised form
1 April 2015
Accepted 1 April 2015
Available online 12 April 2015

Keywords:
Scheduling
Tool switching
Combinatorial algorithm
Manufacturing
Flexible machine

ABSTRACT

We revisit the tool switching problem on a flexible manufacturing machine. We present a polynomial algorithm for the problem of finding a switching plan that minimizes the number of tool switch instances on the machine, given a fixed job sequence. We prove tight hardness results for the variable sequence case with the same objective function, as well as a new objective function naturally arising in multi-feeder mailroom inserting systems.

1. Introduction

A *flexible manufacturing machine* consists of a single production line with C independent *slots*, each able to contain a single *tool* out of a set of M different tools. A *job* corresponds to a product that needs to be processed by the machine, and it is characterized by a subset of required tools that need to be mounted on the slots of the machine. While we always assume that there are enough slots to produce any single job in the given batch, typically the total number of tools is much larger than the number of slots. Hence to complete a batch of jobs, it is often necessary to switch the tools mounted onto the machine at different times of the production. This gives rise to several optimization problems concerning the tool switches. Formally, an instance of the tool switching problem consists of the number C of slots in the machine, a collection $I = \{1, \ldots, M\}$ of tools, and a collection $J = \{1, \ldots, N\}$ of jobs. Each job $j \in J$ is associated with its required set of tools $I_j \subseteq I$. Additionally, each job j has a *production time* $d_j \in \mathbb{Z}_{>0}$, corresponding to the *consecutive* number of time units required to complete its production, and every tool $i \in I$ has a *setup time* $b_i \in \mathbb{Z}_{>0}$, that is the number of consecutive time units that a slot needs to be idle in order to load a tool on it. We also use $J_i \subseteq J$ to denote the set of jobs requiring tool i.

Given a *production batch*, the goal is to decide the production order of the jobs as well as, for each job, the assignment of its tools to the slots. In some cases the job sequence is fixed by the machine operator, and only the tool assignment has to be decided. The latter type of problems, which we denote by *fixed-sequence* optimization, are the main focus of this paper. For this class of problems, we assume that the numbering of the jobs corresponds to their fixed sequence. For the general variable-sequence case we prove some complexity results, and we defer the required definitions to the corresponding section. We represent a solution to a fixed-sequence optimization problem by a *plan* $\mathbf{F} \in \{0, \ldots, C\}^{M\times N}$ where $\mathbf{F}_{ij} = k$ if tool i is assigned to slot k for job j, or $\mathbf{F}_{ij} = 0$ if $i \notin I_j$. Since each slot can only contain a single tool at any time, we have that $\mathbf{F}_{ij} \neq \mathbf{F}_{lj}$ for any $j \in J$ and $i, l \in I_j$. The solution to some optimization problems considered here consists of a plan and some additional information, computable only from the plan. We elaborate on this later on.

With a slight abuse of notation, we denote by J both the job set $\{1, \ldots, N\}$ and the job sequence $(1, \ldots, N)$. We similarly denote by $S \subseteq J$ both a job subset and a *job subsequence*, i.e., a sequence $S = (a, \ldots, e)$ of consecutive integers with $1 \leqslant a \leqslant e \leqslant m$. Also, in the fixed-sequence case statements like "last", "first", "later" and "sooner" will correspond to the order given by J. We say that a pair (i, j) is *valid* if $j \in J$ and $i \in I_j$. Given a plan $\mathbf{F}$ and a valid pair (i, j), let

$$\mathrm{PrevJob}(i, j) := \max\left\{q \in J : q < j,\ \mathbf{F}_{lq} = \mathbf{F}_{ij} \text{ for some } l \in I\right\}$$

and $\mathrm{PrevTool}(i, j) := l$, where l is such that $\mathbf{F}_{l\,\mathrm{PrevJob}(i,j)} = \mathbf{F}_{ij}$. Since each slot can only contain a single tool at each time, $\mathrm{PrevTool}(i, j)$ is well-defined. Informally, $\mathrm{PrevJob}(i, j)$ denotes the last job before

[*] Corresponding author.
E-mail addresses: david.adjiashvili@ifor.math.ethz.ch (D. Adjiashvili), sandro.bosio@ifor.math.ethz.ch (S. Bosio), kevin.zemmer@ifor.math.ethz.ch (K. Zemmer).

http://dx.doi.org/10.1016/j.orl.2015.04.001

job j during which slot $\mathbf{F}_{ij}$ is used, and $\text{PrevTool}(i,j)$ denotes the tool that is present in slot $\mathbf{F}_{ij}$ for the production of job $\text{PrevJob}(i,j)$. If no job uses slot $\mathbf{F}_{ij}$ before job j we set $\text{PrevTool}(i,j) = \otimes$ and $\text{PrevJob}(i,j) = 0$.

Let us define next the objective functions that are treated in this paper. We say that plan $\mathbf{F}$ requires a *tool switch* (or, simply, a switch) for tool i at job j, if (i,j) is a valid pair and $\text{PrevTool}(i,j) \neq i$. Informally, a tool switch is required if the slot $\mathbf{F}_{ij}$ was either never used before processing job j, or the tool that slot $\mathbf{F}_{ij}$ contained previous to job j was different from i. Given a plan $\mathbf{F}$, a slot k is said to be *idle* at job j if there is no tool i such that $\mathbf{F}_{ij} = k$. Note that idle slots do not need to be empty, i.e., they can still have tools loaded in them. In fact, we implicitly assume that tools are never unloaded from a slot unless a new tool is loaded in the same slot. The *idle time* of slot $\mathbf{F}_{ij}$ is defined as $\sum_{\ell \in S_{ij}} d_\ell$ where $S_{ij} = \{\ell \in \mathbb{N} \mid \text{PrevJob}(i,j) < \ell < j\}$, i.e. it is the total time that the slot $\mathbf{F}_{ij}$ was not used before the production of job j. In particular, the idle time of slot k at job j is zero if $\mathbf{F}_{i(j-1)} = k$ for some i. A switch for (i,j) results in a *setup violation* if the total idle time of the slot $\mathbf{F}_{ij}$ before processing job j is less than b_i, the setup time of tool i. Setup violations have to be resolved by *machine stops*, which are usually not desirable because they amount to additional idle time. How to determine an optimal stop program for a given plan is described in Section 2.1. The *split degree* of tool i is defined as $\left|\left\{k \mid \exists (i,j) \text{ valid s.t. } \mathbf{F}_{ij} = k\right\}\right|$. In other words, it is the number of distinct slots onto which tool i is mounted in plan $\mathbf{F}$, counting each slot only once even if a tool is loaded onto the slot for multiple jobs. The split degree of a plan is the sum of the split degrees of all tools. Observe that the split degree of a plan $\mathbf{F}$ is invariant under permutations of the columns, which correspond to alteration of the job sequence. Hence, the split degree of a plan is independent of the job sequence, but only depends on the collection of tool to slot assignments of all jobs.

Finally, the *tool switch minimization, the machine stop minimization* and the *split minimization* problems correspond to the problem of finding a plan $\mathbf{F}$ with a minimum number of tool switches, machine stops, and split degree, respectively. We note that the split degree of a plan is independent of the job sequence and the setup and production times. In contrast, the stop minimization problem critically depends on all these parameters, and comprises the main focus of this paper. We describe our motivation for studying these objective functions later on.

1.1. Contribution

Our main contribution is a first polynomial algorithm for the machine stop minimization problem. In the special case where *every* tool switch requires a machine stop, our algorithm finds a plan minimizing the number of *switch instances*, i.e. the number of jobs which require some switch. The algorithm and its analysis is presented in Section 2. For the variable-sequence variants of the machine stop minimization problem and the split minimization problem, we show tight NP-hardness results in Section 4. Concretely, we show that both problems are NP-hard for any $C \geqslant 2$. Finally, we show in Section 3 that the fixed-sequence case and for a fixed *stop program*, i.e. a fixed set of times in which machine stops are to be performed, it is possible to find in polynomial time a plan that requires stops at most at these times and minimizes the number of required tool switches. The latter result allows us to solve some bi-objective variants of the fixed-sequence problem. In particular, we can show lexicographic minimization of stops and tool switches is possible for the well-studied case of *uniform instances*, where all production and setup times are equal to one. This result is the only one involving the tool switches objective in this paper.

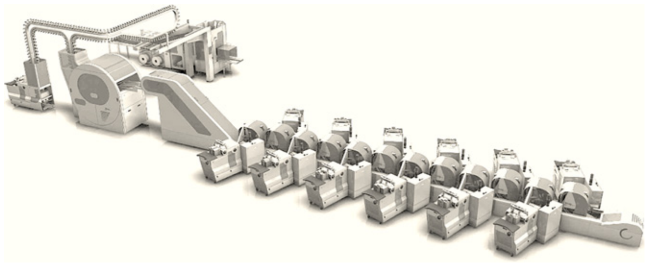

Fig. 1. Inserting machine with $C = 12$ feeders, six on each side. Image courtesy of Ferag AG, Switzerland.

1.2. Motivation

The motivation for our work stems from a collaboration with a company in the mailing industry. In this context tools and jobs are associated with *advertisement inserts* and *folders* (e.g., newspapers), respectively. Fig. 1 illustrates an inserting machine with 12 tool slots, corresponding to insert *feeders*. While the tool switch minimization and stop minimization problems are relevant not only for mailroom insert planning, the split minimization problem defined here is more specific to this application. Concretely, in order to use an insert on any feeder during the production, it is often necessary to prepare a large pallet of this insert near the feeder before the production starts. If an insert is used on more than one feeder, multiple pallets have to be prepared for this insert. Thus split degree of a plan corresponds to the number of pallets that need to be prepared. As this preparation work can be very time consuming and the space for the pallets can be limited, minimizing the total number of pallets that need to be prepared can become an important objective.

1.3. Related work

A special case of the tool switch minimization problem was first introduced by Belady [1] in the context of virtual management in computers. Tang and Denardo [12] were the first to give a polynomial algorithm for the fixed-sequence tool switching minimization problem. Their elegant algorithm uses the *keep tool needed soonest* heuristic, which they proved to be optimal for this problem. The authors provide some computational results for a natural IP formulation of the variable-sequence variant of the problem. In a follow-up paper Tang and Denardo [13] consider the problem of minimizing the number of switch instances, which is a special case of the stop minimization problem. The authors develop a branch-and-bound procedure for the variable-sequence case, and prove its NP-hardness. Crama et al. [2] use the theory of totally unimodular matrices to provide a polynomial algorithm for a more general variant of the tool switching minimization problem, in which switches are weighted by their loading time. The authors also prove that the variable-sequence problem is NP-hard. Privault and Finke [10] reduce the fixed-sequence tool switching minimization problem to a minimum-cost flow problem. Song and Hwang [11] proposed a polynomial algorithm for the problem when at most D tools can be switched at any given time, for some $D \in \mathbb{Z}_{>0}$. Crama et al. [3] show that the fixed-sequence tool switching minimization problem becomes NP-hard when tools have variable size, and when they are *physical*, i.e., they occupy consecutive slots in the magazine. For a fixed magazine size, the authors provide a polynomial time algorithm. NP-hardness of the variable-sequence tool switch instances minimization problem was proved by Crama and Oerlemans [4] for $C \geqslant 3$. The authors also provide a column generation algorithm, and analyze it empirically. Crama and van de Klundert [5] analyzed the approximation guarantees of several heuristics for the variable-sequence variants of the tool switching and the tool