European Journal of Operational Research 236 (2014) 37–50

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Scheduling with few changes

Torsten Mütze

Institute of Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland[1]

ARTICLE INFO

Article history:
Received 4 January 2013
Accepted 11 November 2013
Available online 20 November 2013

Keywords:
Assignment problem
Changeover cost
Algorithm

ABSTRACT

In this work we consider scheduling problems where a sequence of assignments from products to machines – or from tasks to operators, or from workers to resources – has to be determined, with the goal of minimizing the costs (=money, manpower, and/or time) that are incurred by the interplay between those assignments. To account for the different practical requirements (e.g. few changes between different products/tasks on the same machine/operator, few production disruptions, or few changes of the same worker between different resources), we employ different objective functions that are all based on elementary combinatorial properties of the schedule matrix. We propose simple and efficient algorithms to solve the corresponding optimization problems, and provide hardness results where such algorithms most likely do not exist.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

In many industrial production environments, considerable costs (=money, manpower, and/or time) are incurred whenever the production is disrupted, e.g. to readjust, retool or clean the machines. Undoubtedly due to their relevance in numerous practical applications, scheduling problems that aim to minimize these costs have received a lot of attention in the literature (see e.g. Allahverdi, Ng, Cheng, & Kovalyov, 2008; Błażewicz, Ecker, Pesch, Schmidt, & Węglarz, 2001; Brucker, 2007 and references therein). Another fundamental problem in combinatorial optimization is the assignment problem, where tasks are to be assigned to machines, such that the costs to complete all the tasks are minimized (see e.g. Ahuja, Magnanti, & Orlin, 1993; Çela, 2002; Cook, Cunningham, Pulleyblank, & Schrijver, 1998; Pentico, 2007).

In this work we consider a family of scheduling problems at the intersection of the two above-mentioned problems: We are interested in finding an entire sequence of assignments, and the costs are incurred (only) by the interplay between those assignments. More specifically, we are given a set of parts $\{1,2,\dots,n\}$ and a sequence of products $P_1,\dots,P_\ell$, each consisting of a subset of the parts, $P_i \subseteq \{1,2,\dots,n\}$. There are m identical machines, and in the ith step of the production, product $P_i =: \{p_1,\dots,p_k\}$ is produced by assembling the corresponding parts $p_1,\dots,p_k$. For this purpose the parts $p_1,\dots,p_k$ have to be assigned to k of the machines, and the remaining $m-k$ machines are idle. If $m \geqslant n$, then we can reserve each machine for only one particular part throughout the production and assign all occurrences of this part to this machine (this is considered an 'ideal' schedule). In practice however, we usually have $m < n$ (there could be as few as $m = \max_{i\in[\ell]}|P_i|$ machines), and a schedule will necessarily be more 'disordered', i.e., we cannot avoid assigning different parts to the same machine in subsequent time slots, or assigning the same part to different machines during the production. Later, we will formally define four different objective functions that measure this 'disorder' in a schedule (see Fig. 1 below).

To demonstrate that our model is quite versatile, we briefly mention three concrete applications in the following. These will also serve as a plausibility check when giving the formal definitions of our objective functions. Furthermore, the examples immediately suggest natural extensions of the model, some of which will be discussed in Section 1.2.4 below.

Example 1. A chemical manufacturing plant produces various chemicals (=products), each consisting of a specific set of constituents (=parts). The constituents are supplied to the mixing and reaction stage through a number of supply pipes (=machines). Whenever a constituent on a supply pipe changes during the production, this incurs costs to retool and clean the pipe.

Example 2. A pharmaceutical packaging facility bundles together different types of pills (=parts) into patient or region specific boxes (=products). The pills are stored in containers that can be hooked to the feeding holes (=machines) of the packaging unit. Costs are incurred by the manpower necessary to exchange the containers, possibly interrupting the whole production.

The last example shows that our model also captures a set of applications that are phrased in an entirely different language.

[1] This work was performed as a research project in cooperation with Supercomputing Systems AG, Zürich, Switzerland.

E-mail address: torsten.muetze@inf.ethz.ch

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.11.011

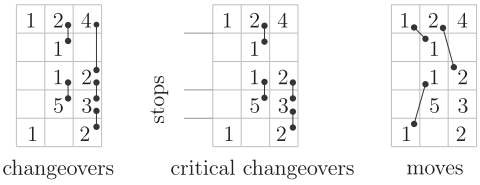

Fig. 1. The various objective functions for a schedule σ for a production with $\ell = 5$ products and $n = 5$ parts on $m = 3$ machines. The schedule σ has $N_{\text{change}}(\sigma) = 5$ changeovers (left hand side of the figure), $N_{\text{crit}}(\sigma) = 4$ critical changeovers, $N_{\text{stop}}(\sigma) = 3$ stops (middle; stops are indicated by horizontal dotted lines) and $N_{\text{move}}(\sigma) = 3$ moves (right). The entries of σ with value ε are represented by empty boxes.

Example 3. An insurance company has a number of employees (=parts) who use the company cars (=machines) for their field work. Work is organized in hourly timeslots, and during each timeslot, only a subset of employees (=products) needs a company car. Costs are incurred by the time losses and inconveniences caused whenever two different employees are assigned the same car in directly subsequent time slots (one has to prematurely hurry back for the exchange), and also whenever the same employee has to switch and adapt to a different car.

The main contribution of this work are simple and efficient algorithms for the optimization problems arising from the above-mentioned scenarios. We also provide hardness results to identify problems where such algorithms most likely do not exist. We hope to address those hard problems (which are important for some of the mentioned applications) in future work, by providing approximation algorithms with provable guarantees.

1.1. Formal problem definition

Before stating our results, we give a formal definition of the problem outlined above.

Products and machines. For any two integers r and s with $r \leqslant s$ we define $[r,s] := \{r, r+1, \ldots, s\}$ and $[r] := [1,r] = \{1,2,\ldots,r\}$.

A *production* is a pair $([n], P)$, where n is an integer and $P = (P_1, \ldots, P_\ell)$ is a sequence of non-empty subsets of $[n]$, such that $\bigcup_{i\in[\ell]} P_i = [n]$ (or equivalently, for every $p \in [n]$ there is some $i \in [\ell]$ with $p \in P_i$). As before, we refer to each of the sets P_i as a *product*, and to the elements of P_i as the *parts* of product P_i (these terms will often be used in informal discussions). Clearly, the requirement that each part appears in at least one product is just a natural way to exclude artificial complications. As before we use m to denote the number of *machines*.

Schedules. A *schedule for a production* $([n], P), P = (P_1, \ldots, P_\ell)$, *on* m *machines* is an $\ell \times m$ matrix σ with values from $[n] \cup \{\varepsilon\}$ (ε is a symbolic placeholder) with the property that for every $i \in [\ell]$ every element of P_i occurs exactly once in the ith row of σ (so the ith row contains exactly $m - |P_i|$ values equal to ε). The interpretation of an entry (i,j) with $\sigma(i,j) = p \in P_i$ is that when producing product P_i the part p is assigned to machine j, and $\sigma(i,j) = \varepsilon$ means that when producing product P_i machine j is idle. We use $\Sigma = \Sigma([n], P, m)$ to denote the set of all schedules for the production $([n], P)$ on m machines.

Clearly, a necessary (and sufficient) condition for the existence of a schedule for a given production $([n], P), P = (P_1, \ldots, P_\ell)$, on m machines is that $m \geqslant \max_{i\in[\ell]} |P_i|$.

Objective functions. We now define four objective functions that measure the 'disorder' of a schedule. The reader is invited to check that each of the objectives is meaningful for some or all of the previously mentioned applications (not all objectives are relevant for all examples, and often some objectives are more important than others). For the reader's convenience, the following definitions are illustrated in Fig. 1.

For a production $([n], P)$ and integer m, a *changeover* in a schedule $\sigma \in \Sigma([n], P, m)$ is a triple (i_1, i_2, j) with $i_1 < i_2, \sigma(i_1, j) =: p \in [n]$, $\sigma(i_2, j) =: q \in [n], p \neq q$, and $\sigma(i,j) = \varepsilon$ for all $i_1 < i < i_2$. We denote the number of changeovers in σ by $N_{\text{change}}(\sigma)$.

In many applications (cf. the above examples), a changeover (i_1, i_2, j) gets more relevant/more costly, if the difference $i_2 - i_1$ gets smaller (think of $i_2 - i_1$ as the available time in which the changeover has to be performed). In this work we distinguish changeovers (i_1, i_2, j) with $i_2 - i_1 = 1$, which we call *critical*, from changeovers with $i_2 - i_1 \geqslant 2$ (those are not critical). We denote the number of critical changeovers in σ by $N_{\text{crit}}(\sigma)$.

A critical changeover on one machine may delay or interrupt the production on all m machines (as in Examples 1 and 2 above, but not in Example 3). A *stop* in a schedule $\sigma \in \Sigma([n], P, m)$ is an index $i \in [2, \ell]$ for which there exists a critical changeover $(i-1, i, j), j \in [m]$. Note that a single stop can be caused by several critical changeovers between rows $i-1$ and i. We denote the number of stops in σ by $N_{\text{stop}}(\sigma)$.

A *move* in a schedule $\sigma \in \Sigma([n], P, m)$ is a quadruple (i_1, j_1, i_2, j_2) with $i_1 < i_2, j_1 \neq j_2, \sigma(i_1, j_1) = \sigma(i_2, j_2) =: p \in [n]$ and $p \notin P_i$ for all $i_1 < i < i_2$. We denote the number of moves in σ by $N_{\text{move}}(\sigma)$.

For the four above-mentioned quantities we define the *minimal number of changeovers, the minimal number of critical changeovers, the minimal number of stops* and the *minimal number of moves*, by setting for $* \in \{\text{change}, \text{crit}, \text{stop}, \text{move}\}$ ($*$ is used as a variable representing each of the four cases)

$$N_*^{\min} = N_*^{\min}([n], P, m) := \min_{\sigma\in\Sigma([n],P,m)} N_*(\sigma), \tag{1}$$

respectively. Later we will also consider combinations of these optimization goals. We remark here that minimizing the number of stops is equivalent to makespan minimization (assuming that every stop takes the same amount of time).

Clearly, for $m \geqslant n$ changeovers, stops and moves can be avoided completely (we may choose a different machine for each part, and assign all occurrences of a part to the corresponding machine). We are therefore only interested in the case $m < n$ in the following.

1.2. Our results

We are now ready to state our results.

1.2.1. Minimizing changeovers

Theorem 4. *There is an algorithm which computes, for any production* $([n], P), P = (P_1, \ldots, P_\ell)$, *and any integer* m *with* $\max_{i\in[\ell]} |P_i| \leqslant m < n$, *the minimal number of changeovers* $N_{\text{change}}^{\min}$ *as defined in (1) and a schedule* $\sigma \in \Sigma([n], P, m)$ *with* $N_{\text{change}}(\sigma) = N_{\text{change}}^{\min}$ *in time* $\mathcal{O}(\ell \cdot m)$.

Note that the required bounds on m in Theorem 4 are only included to avoid trivial cases as mentioned in Section 1.1.

In many applications, e.g. if m is constant, every $|P_i|$ is a positive fraction of m (recall that the P_i are all non-empty), and then the size of the input satisfies $\sum_{i\in[\ell]} |P_i| = \Theta(\ell \cdot m)$. In those cases, the claimed running time is therefore best possible. Note also that if we wish to output the computed optimal schedule σ as an entire $\ell \times m$ matrix, then already this requires time $\Theta(\ell \cdot m)$, and the given runtime bound is best possible. Nevertheless, if the input is sparse, i.e., $\sum_{i\in[\ell]} |P_i| = o(\ell \cdot m)$, and if we either decide to output an optimal schedule σ in sparse form (e.g., by returning for every $i \in [\ell]$ and every $p \in P_i$ the machine to which p is assigned) or restrict ourselves to computing only the integer $N_{\text{change}}^{\min}$ and no

Download English Version:

https://daneshyari.com/en/article/6897482

Download Persian Version:

https://daneshyari.com/article/6897482

Daneshyari.com