Computers & Operations Research 74 (2016) 42–52

Contents lists available at ScienceDirect

Computers & Operations Research

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

Solution algorithms for synchronous flow shop problems with two dominating machines ☆

Matthias Kampmeyer, Sigrid Knust *, Stefan Waldherr

Institute of Computer Science, University of Osnabrück, Germany

ARTICLE INFO

Article history:
Received 29 May 2015
Received in revised form
1 December 2015
Accepted 8 April 2016
Available online 12 April 2016

Keywords:
Flow shop
Synchronous movement
Dominating machines
Vehicle routing

ABSTRACT

In this paper, we present solution algorithms for synchronous flow shop problems with two dominating machines. In such an environment, jobs have to be moved from one machine to the next by an unpaced synchronous transportation system, which implies that the processing is organized in synchronized cycles. This means that in each cycle the current jobs start at the same time on the corresponding machines and after processing have to wait until the last job is finished. Afterwards, all jobs are moved to the next machine simultaneously. Motivated by a practical application, we investigate the special case of two dominating machines where the processing times of all jobs on these two machines are at least as large as the processing times of all jobs on the other machines and hence always determine the cycle times. After formulating the considered problem as a special vehicle routing problem, we propose mixed integer linear programming formulations and a tabu search algorithm. Finally, we present computational results for randomly generated data and show the efficiency of the approaches.

1. Introduction

A flow shop with synchronous movement ("synchronous flow shop" for short) is a variant of a non-preemptive permutation flow shop where transfers of jobs from one machine to the next take place at the same time. Processing of a job on the next machine may only start after the current jobs on all machines are finished, i.e., after the maximal processing time of the jobs that are currently processed. If the processing time of a job on a certain machine is smaller than this maximum, the corresponding machine is idle until the job may be transferred to the next machine. In contrast, in a classical flow shop the transfer of jobs is asynchronous: Jobs may be transferred to the next machine as soon as their processing on the current machine is completed and processing on the next machine immediately starts as soon as this machine is available.

Synchronous flow shops were first discussed by Kouvelis and Karabati [14]. The authors present a mixed integer programming formulation and discuss approaches for a cyclic production environment in which a set of jobs is produced repeatedly. Further, they proved that the synchronous flow shop problem is $\mathcal{NP}$-hard for an arbitrary number of machines. This result was strengthened by Waldherr and Knust [23] showing that the synchronous flow shop problem is already $\mathcal{NP}$-hard for three machines. Soylu et al. [17] present a branch-and-bound approach as well as several heuristics to minimize the makespan in synchronous flow shops. In Huang [10], rotating production units with synchronous movement and a loading/unloading (L/U) station are considered. In this framework, a job enters the production unit at the L/U station and is then processed on all machines before returning to the L/U station where it is unloaded. A polynomial algorithm to minimize the makespan for a production unit with two machines and constant product-independent removal times is presented. Furthermore, dynamic programming approaches for the case with non-constant removal times and two or three machines are proposed.

Our work is further motivated by a practical application studied in Waldherr and Knust [22]. There, in the production process of shelf-boards at a kitchen manufacturer circular production units with eight machines incorporating synchronous movement are used. For these production units the concept of machine dominance is important. A subset of machines is called dominating if the processing times of all jobs on these machine are at least as large as the processing times of all jobs on the other machines. In other words, this means that the dominating machines dictate the pace of the flow shop with synchronous movement. In the practical application from [22] two of the eight machines are dominating. Classical flow shop problems with dominating machines are well studied in literature. For example, efficiently solvable cases can be found in Monma and Rinnooy Kan [16], Ho and Gupta

☆This work was supported by the Deutsche Forschungsgemeinschaft, KN 512/7-1
* Corresponding author.
E-mail addresses: matthias.kampmeyer@uni-osnabrueck.de (M. Kampmeyer), sigrid.knust@uni-osnabrueck.de (S. Knust), stefan.waldherr@uni-osnabrueck.de (S. Waldherr).

http://dx.doi.org/10.1016/j.cor.2016.04.010
0305-0548/

[9], Xiang et al. [25]. In Wang and Xia [24] dominating machines in no-wait flow shops are investigated. Complexity results for synchronous flow shop problems with dominating machines can be found in [23].

In this paper, we show that synchronous flow shop problems with two dominating machines are closely related to vehicle routing problems with unit demands of the customers and special arc costs. For a general survey on vehicle problems see Toth and Vigo [18], problems with unit demands were, for example, considered in Campos et al. [2] and Angel et al. [1]. The special arc costs in our application are the same as those considered by Gilmore and Gomory [8] for the traveling salesman problem. This polynomially solvable case of the TSP has received a lot of attention in the literature, cf. Chandrasekaran [3], Kabadi and Baki [12], Kabadi [11], Kao and Sanghi [13], and Vairaktarakis [19,20].

The remainder of this paper is organized as follows. After giving a formal description of the considered problem in Section 2, in Section 3 we show how the problem can be formulated as a special vehicle routing problem. In Section 4 we propose different mixed integer linear programming formulations, in Section 5 a tabu search algorithm is described. Computational results can be found in Section 6. Finally, conclusions are presented in Section 7.

2. Problem formulation

In this section, we describe the studied problem more formally and introduce the used notations. We consider a permutation flow shop with m machines $M_1, \ldots, M_m$ and n jobs $N = \{1, \ldots, n\}$ where job j consists of m operations $O_{1j} \rightarrow O_{2j} \rightarrow \ldots \rightarrow O_{mj}$. Operation O_{ij} has to be processed without preemption on machine M_i for p_{ij} time units. In a feasible schedule each machine processes at most one operation at any time, each job is processed on at most one machine at any time, and the jobs are processed in the predefined order.

The processing is organized in synchronized cycles since jobs have to be moved from one machine to the next by an unpaced synchronous transportation system. This means that in a cycle all current jobs start at the same time on the corresponding machines. Then, all jobs are processed and have to wait until the last one is finished. Afterwards, all jobs are moved to the next machine simultaneously. The job processed on the last machine M_m leaves the system, a new job (if available) is put on the first machine M_1. As a consequence, the processing time of a cycle is determined by the maximum processing time of the operations contained in it. Furthermore, only permutation schedules are feasible, i.e., the jobs have to be processed in the same order on all machines.

Let C_j be the completion time of job j, i.e., the time where j has been processed on all machines and leaves the system. We assume that a job can only be accessed after the whole cycle has been completed, i.e., the job has to wait until all jobs on the other machines in the corresponding cycle are finished. Thus, the completion time C_j of a job j is defined as the time when the corresponding cycle of its last operation O_{mj} is finished (i.e., the next cycle starts). The goal is to find a sequence of the jobs such that the makespan $C_{\max} = \max C_j$ is minimized. With each sequence a corresponding (left-shifted) schedule is associated in which each operation starts as early as possible (cf. Waldherr and Knust [23]).

As an example consider a flow shop with $n=7$ jobs and $m=5$ machines. In Fig. 1, a feasible schedule corresponding to the sequence $\pi = (1, 2, 3, 4, 5, 6, 7)$ and the completion times C_j are shown. The schedule consists of $n + m - 1$ cycles, which are divided into a starting phase ($m - 1$ cycles, until jobs are present on each machine), a standard phase ($n - m + 1$ cycles, as described above), and a final phase ($m - 1$ cycles, no more jobs are available for M_1).

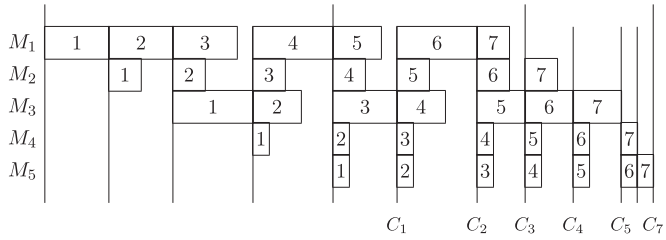

Fig. 1. Feasible synchronous flow shop schedule for the sequence $\pi = (1, 2, 3, 4, 5, 6, 7)$ and $m=5$ machines.

For a given sequence π the makespan can be calculated as follows:

$$C_{\max}(\pi) = \sum_{\lambda=1}^{m-1} \max_{\mu=1,\ldots,\lambda} \{p_{\mu\pi_{\lambda-\mu+1}}\} + \sum_{\lambda=m}^{n-m+1} \max_{\mu=1,\ldots,m} \{p_{\mu\pi_{\lambda-\mu+1}}\} + \sum_{\lambda=2}^{m} \max_{\mu=\lambda,\ldots,m} \{p_{\mu\pi_{n-\mu+\lambda}}\}. \quad (1)$$

Motivated by the practical application, we assume that the processing times of the cycles are only determined by a subset of so-called dominating machines, i.e., these machines dictate the pace of the synchronous movement. A subset $D \subset \{1, \ldots, m\}$ is called dominating if the processing times on the other machines are always not larger than those on the dominating machines, i.e.

$$\min_{i \in D} \min_{j \in N} p_{ij} \geq \max_{i \notin D} \max_{j \in N} p_{ij}.$$

For example, in Fig. 1 the two machines M_1 and M_3 are dominating.

Additionally, we assume that on the non-dominating machines the jobs have job-independent processing times $p_{ij} = p_i$ (which are all smaller than the processing times on the dominating machines). This assumption is motivated by the fact that the non-dominating machines resemble work processes like insertion or removal of work pieces that have the same processing time regardless of the actual job. In this case, the required time in cycles where no job is processed on a dominating machine is constant and independent of the job sequence. Thus, we can simply ignore these cycles and add a corresponding constant to all completion times. Since this constant is sequence-independent and adding a constant to the makespan objective does not change an optimal sequence, in this situation we may even assume that all processing times on the non-dominating machines are equal to zero (cf. Waldherr and Knust [23]).

In [23] it was also shown that in the case of arbitrary processing times on the non-dominating machines it is possible to iterate over all potential sequences of the first $i_1 - 1$ and the last $m - i_2$ jobs and then to solve a reduced problem with job-independent processing times for the remaining non-fixed jobs in between. Therefore, the algorithms of this paper can also be extended to cases where the processing times on the dominating machines are not job-independent. Since there are $O(n^{m-i_2+i_1-1})$ possibilities to fix the job sequences in the first $i_1 - 1$ and the last $m - i_2$ cycles, this additional iteration can be done in polynomial time if the number of machines m is fixed (i.e., not part of the input).

According to [23] our problem can be classified as $F|synmv, dom(i_1, i_2), p_{ij}^{ndom} = p_i|C_{\max}$ or equivalently $F|synmv, dom(i_1, i_2), p_{ij}^{ndom} = 0|C_{\max}$ where i_1,i_2 are the indices of the two dominating machines. In [23] it was shown that problem $F|synmv, dom(i, i+1), p_{ij}^{ndom} = p_i|C_{\max}$ with two adjacent dominating machines is equivalent to the two-machine no-wait flow shop problem $F2|no-wait|C_{\max}$ where the two machines in the no-wait problem correspond to the two dominating machines M_i and M_{i+1} of the synchronous flow shop. This problem can be solved in