European Journal of Operational Research 242 (2015) 34–44

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Complexity results for flow shop problems with synchronous movement[☆]

Stefan Waldherr, Sigrid Knust[*]

Institute of Computer Science, University of Osnabrück, Germany

ARTICLE INFO

Article history:
Received 15 December 2013
Accepted 26 September 2014
Available online 5 October 2014

Keywords:
Flow shop
Synchronous movement
Complexity
Dominating machines

ABSTRACT

In this paper we present complexity results for flow shop problems with synchronous movement which are a variant of a non-preemptive permutation flow shop. 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.

Besides the general situation we also investigate special cases involving machine dominance which means that the processing times of all jobs on a dominating machine are at least as large as the processing times of all jobs on the other machines. Especially, we study flow shops with synchronous movement for a small number of dominating machines (one or two) and different objective functions.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

A flow shop with synchronous movement 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.

Complexity of production systems with synchronous movements was first discussed by Karabati and Sayin (2003). In their work the authors consider a cyclic production environment minimizing the completion time of one production cycle and prove that the problem is $\mathcal{NP}$-hard for an arbitrary number of machines. Soylu, Kirca, and Azizoğlu (2007) present a branch-and-bound approach to minimize the makespan in flow shops with synchronous transfers. Additionally, it is claimed that the three-machine flow shop problem with synchronous movement is $\mathcal{NP}$-hard, but the proof seems to be flawed. We will present a correct $\mathcal{NP}$-hardness proof for the three-machine case in this work. In Huang and Hung (2010) as well as Huang (2011), 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 practical application of a synchronous flow shop was studied in Waldherr and Knust (2014). There, in the production process of shelf-boards at a kitchen manufacturer circular production units with eight machines incorporating synchronous movement are used.

In this work we study the complexity of flow shops with synchronous movement which we will also call "synchronous flow shops" for short. Motivated by practical applications (e.g. Waldherr and Knust (2014)) and previous work concerning classical flow shop scheduling problems, we also investigate special cases involving machine dominance. A machine is called *dominating* if the processing times of all jobs on this machine are at least as large as the processing times of all jobs on the other machines. In other words, this implies that the dominating machines dictate the pace of the flow shop with synchronous movement. For example, in the practical application in Waldherr and Knust (2014) among the eight machines two dominating ones exist. 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 (1983), Ho and Gupta (1995), and Xiang, Tang, and Cheng (2000). In Wang and Xia (2005) dominating machines in no-wait flow shops are investigated.

The remainder of this paper is organized as follows. After giving a formal description of the considered problems in Section 2, we study

[☆] This work was supported by the Deutsche Forschungsgemeinschaft, KN 512/7-1.

[*] Corresponding author. Tel.: +49 541 969 2483.
E-mail addresses: stefan.waldherr@uni-osnabrueck.de (S. Waldherr), sigrid.knust@uni-osnabrueck.de (S. Knust).

http://dx.doi.org/10.1016/j.ejor.2014.09.053
0377-2217/© 2014 Elsevier B.V. All rights reserved.

general synchronous flow shop problems in Section 3. Especially, we consider different objective functions for the two-machine case and show that the three-machine synchronous flow shop minimizing the makespan is $\mathcal{NP}$-hard. In Section 4 we consider synchronous flow shop problems with dominating machines in general. Afterwards, we present complexity results for the situations of one and two dominating machines in Sections 5 and 6, respectively. Finally, in Section 7 a summary of all results and some concluding remarks can be found.

2. Problem formulation

In this section we describe the studied problems more formally and introduce the used notations. We consider a permutation flow shop with m machines $M_1, \dots, M_m$ and n jobs where job j consists of m operations $O_{1j} \to O_{2j} \to \dots \to O_{mj}$. 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. Furthermore, the jobs are processed in the same order on all machines.

The processing is organized in synchronized so-called cycles (rounds) 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.

The time at which a job j has been processed on all machines and leaves the system is called its completion time C_j. Depending on the actual environment, for synchronous flow shops two possible definitions of completion times C_j may be considered. In the first version, a job can immediately be removed from the last machine after it is completed. In the second version, a job can only be accessed after the whole cycle has been completed, i.e. the job may have to wait until all jobs on the other machines in the corresponding cycle are finished. A flow shop of the first type can easily be transformed into an equivalent flow shop of the second type by adding another machine with processing times 0 for all jobs after the last machine. Conversely, the reverse is not necessarily true. In general, a flow shop of the second type cannot be transformed into an equivalent flow shop of the first type. In this work we always consider the second type.

The goal is to find a sequence (permutation) of the jobs such that a given objective function (e.g. the makespan $C_{\max} = \max C_j$ or the maximum lateness $L_{\max}$ involving due dates) is minimized. With each sequence a corresponding (left-shifted) schedule is associated in which each operation starts as early as possible.

We consider a small example with three machines, five jobs and the following processing times:

j	1	2	3	4	5
p_{1j}	3	1	2	5	3
p_{2j}	1	3	2	1	1
p_{3j}	1	1	5	5	1

Fig. 1 shows two feasible synchronous flow shop schedules. The vertical lines indicate the cycles and show when the jobs are transferred to the next machine. In the left part of the figure the schedule corresponding to the job sequence $(1, 2, 3, 4, 5)$ with makespan 23 is shown. Here, we can observe a long waiting period (idle time) on machine M_3 between processing jobs 2 and 3: Although job 3 has already finished processing on machine M_2 and job 2 has already been

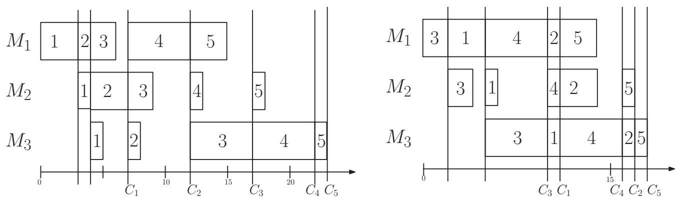

Fig. 1. A feasible and an optimal schedule for a synchronous flow shop.

processed on machine M_3, job 3 may not be transferred to machine M_3 because it has to wait until job 4 is finished on machine M_1. In the right part of the figure the schedule corresponding to the job sequence $(3, 1, 4, 2, 5)$ with makespan 18 is shown. It is optimal for this instance.

In general, a 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, where no more jobs are available for M_1).

Extending the $\alpha|\beta|\gamma$ scheduling classification scheme from Graham, Lawler, Lenstra, and Rinnooy Kan (1979), in Huang (2011) the notation "synmv" was added to the β-field in order to indicate synchronous movements. Hence, the notation $F|\text{synmv}|f$ refers to a synchronous flow shop with objective function f. If the number of machines m is fixed (i.e. not part of the input), we will write $Fm|\text{synmv}|f$.

3. General synchronous flow shops

In this section we consider general synchronous flow shops. Since a flow shop with $m = 1$ is equivalent to a classical single-machine problem, it is not of special interest. Thus, in the following we will consider synchronous flow shops with $m = 2$ or $m = 3$ machines and show that the results may be generalized for $m > 3$ machines.

As already observed in Soylu et al. (2007), the two-machine synchronous flow shop problem is closely related to the corresponding two-machine flow shop problem with no-wait or blocking constraints. It is easy to see that a feasible schedule for a synchronous flow shop also satisfies the blocking constraint and vice versa. Furthermore, a feasible no-wait schedule can be obtained from such a schedule by shifting the operations on the first machine to the right in each cycle (this does not change the completion times of the operations on the second machine). Hence, the makespan is equal in all three situations and thus an optimal schedule for either of the problems $F2|\text{no-wait}|C_{\max}$, $F2|\text{blocking}|C_{\max}$ or $F2|\text{synmv}|C_{\max}$ defines an optimal solution for the other two problems as well. All problems are equivalent to a special case of the traveling salesman problem, which can be solved in $\mathcal{O}(n \log n)$ by the algorithm of Gilmore and Gomory (1964).

On the other hand, because we defined the completion time of a job within the synchronous flow shop as the completion time of the corresponding cycle, the individual completion times of the jobs in a synchronous flow shop are not the same compared to the no-wait or the blocking situation. Hence, for other objective functions, in general we get different objective values for the three variants. Röck (1984) proved that $F2|\text{no-wait}|\sum C_j$ and $F2|\text{no-wait}|L_{\max}$ are strongly $\mathcal{NP}$-hard. Within his proof schedules without idle times on machines M_1 and M_2 are generated. If no idle times exist on the second machine, the completion times of the jobs are the same for the no-wait and the synchronous case. Thus, the results of Röck (1984) can be used to show that $F2|\text{synmv}|\sum C_j$ and $F2|\text{synmv}|L_{\max}$ are strongly $\mathcal{NP}$-hard as well.

Note that for $m \geq 3$ the described equivalence between the three concepts no-wait, blocking, and synchronous movement is no longer valid. For $F3|\text{no-wait}|C_{\max}$ and $F3|\text{blocking}|C_{\max}$ it is known that these problems are strongly $\mathcal{NP}$-hard. Soylu et al. (2007) claimed that also

Download English Version:

https://daneshyari.com/en/article/476577

Download Persian Version:

https://daneshyari.com/article/476577

Daneshyari.com