Ann Oper Res (2008) 159: 183–213
DOI 10.1007/s10479-007-0266-1

On the geometry, preemptions and complexity of multiprocessor and shop scheduling

Evgeny V. Shchepin · Nodari Vakhania

Published online: 1 December 2007

Abstract In this paper we study multiprocessor and open shop scheduling problems from several points of view. We explore a tight dependence of the polynomial solvability/intractability on the number of allowed preemptions. For an exhaustive interrelation, we address the geometry of problems by means of a novel graphical representation. We use the so-called preemption and machine-dependency graphs for preemptive multiprocessor and shop scheduling problems, respectively. In a natural manner, we call a scheduling problem acyclic if the corresponding graph is acyclic. There is a substantial interrelation between the structure of these graphs and the complexity of the problems. Acyclic scheduling problems are quite restrictive; at the same time, many of them still remain NP-hard. We believe that an exhaustive study of acyclic scheduling problems can lead to a better understanding and give a better insight of general scheduling problems.

We show that not only acyclic but also a special non-acyclic version of periodic job-shop scheduling can be solved in polynomial (linear) time. In that version, the corresponding machine dependency graph is allowed to have a special type of the so-called parti-colored cycles. We show that trivial extensions of this problem become NP-hard. Then we suggest a linear-time algorithm for the acyclic open-shop problem in which at most $m-2$ preemptions are allowed, where m is the number of machines. This result is also tight, as we show that if we allow one less preemption, then this strongly restricted version of the classical open-shop

E.V. Shchepin was partially supported by the program "Algebraical and combinatorial methods of mathematical cybernetics" of the Russian Academy of Sciences.
N. Vakhania was partially supported by CONACyT grant No. 48433.

E.V. Shchepin
Steklov Mathematical Institute, Gubkina 8, Moscow 117966, Russia
e-mail: scepin@mi.ras.ru

N. Vakhania (✉)
Science Faculty, State University of Morelos, Av. Universidad 1001, Cuernavaca 62210, Morelos, Mexico
e-mail: nodari@uaem.mx

N. Vakhania
Institute of Computational Mathematics, Akuri 8, Tbilisi 93, Georgia

scheduling problem becomes NP-hard. In general, we show that very simple acyclic shop scheduling problems are NP-hard. As an example, any flow-shop problem with a single job with three operations and the rest of the jobs with a single non-zero length operation is NP-hard. We suggest linear-time approximation algorithm with the worst-case performance of $\|\mathcal{M}\| + 2\|\mathcal{J}\|$ ($\|\mathcal{M}\| + \|\mathcal{J}\|$, respectively) for acyclic job-shop (open-shop, respectively), where $\|\mathcal{J}\|$ ($\|\mathcal{M}\|$, respectively) is the maximal job length (machine load, respectively). We show that no algorithm for scheduling acyclic job-shop can guarantee a better worst-case performance than $\|\mathcal{M}\| + \|\mathcal{J}\|$. We consider two special cases of the acyclic job-shop with the so-called short jobs and short operations (restricting the maximal job and operation length) and solve them optimally in linear time. We show that scheduling m identical processors with at most $m-2$ preemptions is NP-hard, whereas a venerable early linear-time algorithm by McNaughton yields $m-1$ preemptions. Another multiprocessor scheduling problem we consider is that of scheduling m unrelated processors with an additional restriction that the processing time of any job on any machine is no more than the optimal schedule makespan $C^*_{\max}$. We show that the $(2m-3)$-preemptive version of this problem is polynomially solvable, whereas the $(2m-4)$-preemptive version becomes NP-hard. For general unrelated processors, we guarantee near-optimal $(2m-3)$-preemptive schedules. The makespan of such a schedule is no more than either the corresponding non-preemptive schedule makespan or $\max\{C^*_{\max}, p_{\max}\}$, where $C^*_{\max}$ is the optimal (preemptive) schedule makespan and $p_{\max}$ is the maximal job processing time.

Keywords Algorithm · Shop scheduling · Multiprocessor scheduling · Time complexity · Preemption

1 Introduction

In a scheduling problem n *jobs* from the set $\mathcal{J} = \{J^1, \dots, J^n\}$ need to be processed by m *machines* or *processors* from the set $\mathcal{M} = \{M_1, \dots, M_m\}$. Certain restrictions on how this can be done define the set of all *feasible schedules*. One of the principle restrictions are *resource (machine) restrictions*: each machine can handle no more than one job at a time. Likewise, we can have *precedence relations* between the jobs, i.e., the job set can be partially ordered (some jobs cannot be started before the other are not completed). Both type of restrictions imply that the jobs have to be processed in a sequential manner. The precedence restrictions are traditionally represented by directed graphs, the so-called *precedence (task) graphs*.

Job preemptions might be allowed or not; correspondingly, a scheduling problem might be preemptive or non-preemptive. Quite often, a non-preemptive scheduling problem $\mathcal{P}$ is NP-hard, whereas its preemptive version $\mathcal{P}_{pmtn}$ is polynomially solvable. Let us call a scheduling problem π*-preemptive* if at most π preemptions are allowed in it, π being any non-negative integer. A natural question to ask is whether k-preemptive and l-preemptive problems have the same complexity, k and l being non-negative integers. If a 0-preemptive (non-preemptive) problem is NP-hard, does the corresponding k-preemptive problem, for $k = 1, 2, \dots$, remain NP-hard? Typically, there exists some positive integer π such that corresponding k-preemptive versions, for $k \geq \pi$, become polynomially solvable. Then what is the minimal π such that the π-preemptive version is polynomially solvable (we call such π the *critical number* of preemptions for the problem)? Traditionally, a preemptive scheduling problem implies an arbitrary number of preemptions: so a k-preemptive problem for a fixed integer $k > 0$ would not be treated as a preemptive scheduling problem (unless k is "sufficiently" large magnitude not known in advance and changing from problem to problem). So, though a k-preemptive problem ($k > 0$) is not a non-preemptive problem, it might neither be a preemptive problem in the traditional sense. It seems to be appropriate to refine the classification of scheduling problems (into preemptive or non-preemptive ones) by specifying the maximal number of preemptions which is allowed for a particular problem, i.e., consider π-preemptive scheduling problems.

A polynomial-time algorithm for a preemptive problem $\mathcal{P}_{pmtn}$ imposes a certain number of preemptions, the maximal number of which can be usually estimated. In most of the real-life problems the number of preemptions is a crucial factor which has to be made as small as possible: each preemption implies additional communication cost and may also yield a forced job migration. How small the overall number of preemptions can be made, i.e., what is the critical number of preemptions for $\mathcal{P}_{pmtn}$? This kind of question has also posed by Shachnai et al. (2002). They showed that scheduling uniform machines with at most $2m-2$ preemptions $Q/pmtn(2m-2)/C_{\max}$ is NP-hard, whereas an $O(n+m\log m)$ algorithm by Gonzalez and Sahni (1978a) yields at most $2m-1$ preemptions. Thus $2m-1$ is the critical number of preemptions for $Q/pmtn/C_{\max}$. As to scheduling identical processors $P/pmtn/C_{\max}$, we show that $P/pmtn(m-2)/C_{\max}$ is NP-hard, whereas a venerable linear-time algorithm by McNaughton (1959) yields $m-1$ preemptions. Thus $m-1$ is the critical number of preemptions for $P/pmtn/C_{\max}$.

In this paper we expose the *machine dependency graph* (*dependency graph* for short) which is a convenient form for presenting machine restrictions as follows: each node represents a unique machine and an edge (P, Q) labeled with job J indicates that J has to be scheduled (is scheduled) on both machines P and Q. Depending on a particular scheduling problem, a dependency graph may represent either a problem instance or already some distribution of jobs on machines (a *distribution* assigns jobs or their parts to machines without specifying the start times; the latter is done on the sequencing stage which completely defines a schedule). For example, in shop scheduling problems each job has to processed on different machines and hence, there is a unique machine dependency graph representing each problem instance. On the other hand, in preemptive multiprocessor scheduling problems, a job might be split into different parts assigned to different machines. In this case the machine dependency graph will represent a particular distribution of jobs on machines, which may vary from a schedule to a schedule. For this reason, we use the term *preemption graph* for distributions in multiprocessor scheduling (instead of machine dependency graph). At the same time, a shop scheduling problem already gives some distribution and the corresponding machine dependency graph represents that particular distribution. It is well-known that the structure of a precedence graph is important in the complexity analysis of scheduling problems. Quite similarly, the structure of a machine dependency (preemption) graph is important is this analysis. A distribution or a shop scheduling problem is *acyclic*, if its preemption (dependency) graph is acyclic. Acyclic problems are quite restrictive: for example, in any acyclic open-shop no two jobs may have two (non-dummy) operations on the same two machines.

Multiprocessor and open-shop scheduling problems are intimately related with acyclic distributions. An optimal schedule can be obtained in two stages. On the first stage an optimal distribution is constructed in polynomial time by linear programming. This optimal distribution has at most $m-1$ preemptions, i.e., it is acyclic (see Potts 1985 and Shchepin and Vakhania 2005a for details). Scheduling acyclic distributions turned out to be an efficient tool for the exact solution of $R/pmtn/C_{\max}$ (see Lawler and Labetoulle 1978) and an approximate solution of its non-preemptive version $R//C_{\max}$ (see Lenstra et al. 1990 and Shchepin and Vakhania 2005a). Acyclic shop scheduling problems are also interesting from the other point of view: they may represent the maximal polynomially solvable cases of the corresponding non-acyclic versions. Since acyclic graphs are easier to treat, acyclic problems turn out to be more "transparent" than their corresponding non-acyclic versions. Moreover, the study of acyclic problems may give a better insight into general (non-acyclic) scheduling problems.

One of the acyclic problem we deal with in this paper is preemptive open-shop scheduling $O/acyclic, pmtn(m-3)/C_{\max}$. General preemptive open-shop problem $O/pmtn/C_{\max}$ is well-known to be solvable in polynomial (no worse than $O(n^4)$) time, and its non-preemptive version $O//C_{\max}$ is NP-hard due to the early classical results by Gonzalez and Sahni (1976). As we show here, $(m-3)$-preemptive open-shop scheduling and even its acyclic version $O/acyclic, pmtn(m-3)/C_{\max}$ remains NP-hard. At the same time, we present a linear-time algorithm for $O/acyclic, pmtn(m-2)/C_{\max}$ showing in this way that $m-2$ is the critical number of preemptions. The early polynomial algorithm of Gonzalez and Sahni (1976) for the general preemptive open-shop imposes up to $O(n^2m)$ preemptions.

Scheduling m unrelated processors is much more complicated than that of m uniform processors and is among the heaviest NP-hard scheduling problems. We suggest a slightly restricted but easier treatable version of an unrelated processor system. The additional restriction we impose is that the processing time of any job on any machine is no more than the optimal schedule makespan $C^*_{\max}$; i.e., there is no machine which is too slow for some job. It might be assumed that the most of the unrelated machine systems, in practice, satisfy this restriction: intuitively, there is not much sense in having an extremely slow processor for some job. We call such processors *non-lazy* unrelated processors and abbreviate the problem by $R/p_{ij} \le C^*_{\max}/C_{\max}$ adopting the standard notation for scheduling problems. We show that $R/p_{ij} \le C^*_{\max}, pmtn(2m-3)/C_{\max}$ is polynomially solvable, whereas $R/p_{ij} \le C^*_{\max}, pmtn(2m-4)/C_{\max}$ becomes NP-hard. For general unrelated processors, the polynomial algorithm by Lawler and Labetoulle (1978) gives up to $4m^2-5m+2$ preemptions. We can reduce this number to $(2m-3)$ guaranteeing though a near-optimal schedule. The makespan of such a schedule is no more than either the corresponding non-preemptive schedule makespan or $\max\{C^*_{\max}, p_{\max}\}$, where $C^*_{\max}$ is the optimal (preemptive) schedule makespan and $p_{\max}$ is the maximal job processing time.

Our other results concern shop scheduling. There is a considerable list of the polynomially solvable open-shop and flow-shop scheduling problems with unit-length operations. If operation lengths are arbitrary, open-shop problem with 2 machines is solvable in linear time, whereas it becomes NP-hard if either there are 3 machines or 3 jobs Gonzalez and Sahni (1976). As already mentioned, any acyclic open-shop is NP-hard if we allow at most $m-3$ preemptions. In job-shop scheduling, if there are only two machines and two operations per job, the problem is solvable in $O(n \log n)$ time Jackson (1955). The problem can be solved in linear time with two machines and unit-length operations Hefetz and Adiri (1982). With two machines, if we allow jobs with three operations, or with three machines even if there are no more than two operations per job, the problem becomes NP-hard, see Lenstra et al. (1977) and Gonzalez and Sahni (1978b). Periodic shop scheduling, in general, is easier. For example, it is quite straightforward to solve periodic open-shop $O, periodic//C_{\max}$. Hall et al. (2002) have shown that scheduling periodic 2-machine job-shop in which each job is allowed to have 3 operations is NP-hard, but periodic job-shop problem can be solved in linear time if each job has at most two operations. We show that a wider subclass of the periodic job-shop problem can be solved in linear time. To each instance from this subclass corresponds a machine dependency graph which may contain special type of the so-called parti-colored cycles. In terms of the number of operations, this implies that for any 2 jobs with 3 or more operations there can be at most 1 couple of operations (of different jobs) have to be scheduled on the same machine. We show that the class of simplest shop instances for which this condition does not hold, becomes NP-hard.

We also suggest approximation algorithms for non-preemptive acyclic job-shop and open-shop. Our liner-time algorithm for $J/acyclic/C_{\max}$ has the worst-case performance ratio $\|\mathcal{M}\| + 2\|\mathcal{J}\|$, and the linear-time approximation algorithm for $O/acyclic/C_{\max}$ has the worst-case performance ratio $\|\mathcal{M}\| + \|\mathcal{J}\|$, where $\|\mathcal{J}\|$ is the maximal job length and $\|\mathcal{M}\|$ is the maximal machine load (both magnitudes are lower bounds on the optimal schedule makespan). We show that no algorithm for $J/acyclic/C_{\max}$ can guarantee a better worst-case performance than $\|\mathcal{M}\| + \|\mathcal{J}\|$. If we restrict the maximal job (operation, respectively) length, we can solve the acyclic job-shop problem optimally. We abbreviate the version with *short jobs* as $J/acyclic, \max_j \sum_i p_{ij} \le \frac{\max_i \sum_j p_{ij}}{m}/C_{\max}$ and with *short operations* as $J/acyclic, p_{ij} \le \frac{\max_i \sum_j p_{ij}}{2m-1}/C_{\max}$, restricting the maximal job (operation, respectively) length as indicated. We propose linear-time algorithms for the above problems.

We show that very simple classes of acyclic shop instances are NP-hard. For example, any flow-shop with a single job with 3 operations and with the rest of the jobs with a single non-zero operation is NP-hard.

The paper is divided into 5 sections. After this introductory section, we give the basics in Sect. 2. Section 3 contains our polynomial-time algorithms, and Sect. 4 contains our NP-hardness results. Our concluding remarks are given in Sect. 5. Some results from this paper were published in the proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (Shchepin and Vakhania 2005b), and in the proceedings of the 9th WSEAS International Conference on Applied Mathematics (Shchepin and Vakhania 2006).

2 Summary of basic concepts and notations

This section contains glossary of notions and notations used further in this paper. The reader may choose to have a brief look on it or skip it at all now and return to it later upon necessity. Not all the introduced notations are widely used in the literature, however we do find them convenient.

Multiprocessor and shop scheduling problems *A multiprocessor* is a triple constituted by the sets of jobs $\mathcal{J}$ and machines $\mathcal{M}$ and a *processing time function* f, a mapping from $\mathcal{J} \times \mathcal{M}$ to R^+, where the value of this function for a pair J, M is the *processing time (length)* of job J on machine M denoted by $M(J)$. A multiprocessor without any restriction on its processing time function is called a system of *unrelated processors*. In a system of *identical processors* for each P and Q from $\mathcal{M}$ and for each $J \in \mathcal{J}$, $P(J) = Q(J)$.

We will deal with three basic shop scheduling problems and will occasionally use $\mathcal{J}, \mathcal{M}$ to denote a shop scheduling instance with the set of jobs $\mathcal{J} = \{J_1, \dots, J_n\}$ and the set of machines $\mathcal{M} = \{M_1, \dots, M_m\}$. In an instance of the *job-shop* $J//C_{\max}$ each job from $\mathcal{J}$ is an ordered set of elements called *operations*. Each operation is to be scheduled on one particular machine from $\mathcal{M}$. J_i^j is the operation of job J^j to be performed on machine M_i (we shall deal with job-shops in which every job has no more than one operation to be scheduled on one particular machine). We will write $J_i^j \to J_k^j$ if J_i^j *immediately precedes* J_k^j according to the operation order in J^j. Operation J_i^j has a *processing time* or *length* p_i^j, which is the amount of time it takes on machine M_i. J_i^j is a *dummy operation* of job J^j on machine M_i if $p_i^j = 0$.

The *open-shop* $O//C_{\max}$ is a special case of the job-shop in which there is no precedence order between the operations of any job, these operations can be processed in an arbitrary order on their corresponding machines. The *flow-shop* $F//C_{\max}$ is another special case of job-shop scheduling problem in which the operation order in all jobs is the same, i.e., every job is processed by the machines in the same predetermined order.

A *restriction* $\mathcal{J}', \mathcal{M}'$ of a shop instance $\mathcal{J}, \mathcal{M}$ is another shop instance with $\mathcal{M}' \subset \mathcal{M}$ and $\mathcal{J}' \subseteq \mathcal{J}$; $\mathcal{J}, \mathcal{M}$ is an *extension* of $\mathcal{J}', \mathcal{M}'$. If $J_i^j \in J^j$ and $M_i \in \mathcal{M} \setminus \mathcal{M}'$, operation J_i^j disappears in $\mathcal{J}', \mathcal{M}'$; a job will completely disappear if all its (non-dummy) operations disappear. We will say that a shop instance $\mathcal{J}', \mathcal{M}'$ is an *elementary extension* of a shop instance $\mathcal{J}, \mathcal{M}$ if $\mathcal{J}', \mathcal{M}'$ is an extension of $\mathcal{J}, \mathcal{M}$ such that all jobs from $\mathcal{J}' \setminus \mathcal{J}$ are *elementary jobs*, that is, they consist of a single operation.

Distributions and assignments A *distribution* δ of jobs from $\mathcal{J}$ on machines from $\mathcal{M}$ is a mapping $\delta: \mathcal{J} \times \mathcal{M} \to \mathrm{R}^+$, such that $\sum_{M \in \mathcal{M}} \delta(J, M) = 1$, for all $J \in \mathcal{J}$. The *processing time (length)* of job J on M *in distribution* δ is $|J|_M^\delta = \delta(J, M)M(J)$. The *(total) processing time* of J *in* δ is $|J|^\delta = \sum_{M \in \mathcal{M}} |J|_M^\delta$. We use $|\delta|^{\max}$ for the maximal job processing time in δ and we use the traditional $p_{\max}$ for the maximal job processing time.

The *load* $|M|_\delta$ of $M \in \mathcal{M}$ in δ is $\sum_{J \in \mathcal{J}} |J|_M^\delta$. Adopting the common terminology for schedules, we shall refer to the maximal machine load in δ as the *makespan* of δ and denote it by $|\delta|_{\max}$. Note that the distribution, corresponding to each instance of a shop scheduling problem $\mathcal{J}, \mathcal{M}$ is already given by that instance; in particular, this distribution is defined by $\delta(J^j, M_i) = \frac{|J_i^j|}{|J^j|}$, and the machine load then can also be expressed as $|M_i| = \sum_{j=1}^n p_i^j$.

A distribution δ for $\mathcal{J}, \mathcal{M}$ with the minimal $|\delta|_{\max}$ is called an *optimal distribution*. A *uniform distribution (shop problem)* is one, in which all machine loads are equal. An *acyclic shop* problem is one with an acyclic distribution.

The *sequential makespan* of δ, $\|\delta\| = \max\{|\delta|_{\max}, |\delta|^{\max}\}$. It is clear that $\|\delta\|$ is a lower bound on the makespan of any feasible schedule for the corresponding open-shop problem; Gonzalez and Sahni (1976) have shown that $\|\delta\|$ is achievable in a (feasible) schedule associated with distribution δ (see also Lawler and Labetoulle 1978).

An *assignment* of jobs of $\mathcal{J}$ on machines of $\mathcal{M}$ is a binary relation on $\mathcal{J} \times \mathcal{M}$. Any distribution δ of $\mathcal{J}$ on $\mathcal{M}$ generates an assignment $\{(J, M) \mid \delta(J, M) > 0\}$. We will not use any special letter for an assignment, instead, we will use $\delta(J)$ for $\{M \in \mathcal{M} \mid \delta(J, M) > 0\}$. A distribution δ is *non-preemptive* if $\delta(J, M)$ takes value 0 or 1 for any J, M. The number of preemptions of job J in δ is $pr_J(\delta) = |\delta(J)| - 1$. $pr(\delta) = \sum_{J \in \mathcal{J}} pr_J(\delta)$ is the (total) number of preemption in δ.

Schedules A *schedule* indicates which job (operation) is in process on each machine at any time moment; if for some machine no operation for some time moment is specified, this machine is *idle* at that moment. Since a machine can process at most one job at any moment, a schedule can be seen as a mapping from $\mathcal{M} \times [0, T)$, for some $T \geq 0$, to $\mathcal{J}$, or a graph of such a mapping, i.e., a subset of the product $\mathcal{J} \times \mathcal{M} \times [0, T)$. $(J, M, t) \in \sigma$ signifies that job J is processed by machine M at the moment t in σ. T is called the *makespan* of σ and is denoted by $\|\sigma\|$. Note that for any given distribution, there are infinitely many schedules with that distribution. Conversely, with each schedule σ a distribution δ_σ defined as $\delta_\sigma(J, M) = \frac{|\sigma(J,M)|}{M(J)}$ is associated.

Besides the above finite schedules, we deal with *periodic* (infinite) schedules. A periodic schedule is defined as a pair (σ, T), where σ is an (infinite) schedule and $T \in \mathrm{R}^+$ is the *period* of σ. The period T is the minimal non-negative real number, such that: (a) at any time moment t, $(J, M, t) \in \sigma$ implies $(J, M, t + T) \in \sigma$; (b) each job J is completely processed in the time interval $[s, s + T)$, where s is the starting time of the earliest schedule operation of job J. Due to the similarity between the period of a periodic schedule and the makespan of a finite schedule, the period T of σ will be also denoted by $\|\sigma\|$.

By our convention, we use the left-interval representation, i.e., all processing intervals are left half-intervals of the form $[p,q)$ (observe that the whole time axis R^+ is also a left half-interval). A schedule *component* is the maximal time interval during which a machine processes a unique job. A schedule can be completely given by all its components. Formally, a component of a schedule σ is a triple $(J, M, [p,q))$, where J is a job, M is a machine and $[p,q)$ is a time interval which is a connectivity component in $\sigma(J,M)$. We will refer to a component $(J, M, [p,q))$ as a *J-component of σ on M* or a *(J,M)-component*.

For a pair $J, M \in \mathcal{J} \times \mathcal{M}$, $\sigma(J,M) = \{t \in \mathrm{R}^+ \mid (J,M,t) \in \sigma\}$. If σ is finite then $\sigma(J,M)$ is a union of a finite number of left half-intervals, will call such a union a *multi-interval*. A *T-periodic interval*, generated by an interval $[p,q)$ is the union of all half-intervals $\bigcup_{k=0}^{\infty}[p+kT, q+kT)$. For a periodic schedule σ, $\sigma(J,M)$ is a *$\|\sigma\|$-periodic multi-interval* that is, a union of a finite number of $\|\sigma\|$-periodic intervals. The *length* of a finite multi-interval $\mathcal{I}$, $|\mathcal{I}|$ is the sum of lengths of all its disjoint intervals. If $\mathcal{I}$ is a T-periodic multi-interval, then $|\mathcal{I}| = \lim_{k\to\infty} \frac{|\mathcal{I}\cap[0,kT)|}{k}$.

The total length of $\sigma(J,M)$, $|\sigma(J,M)|$, is the processing time of J on M *in* σ. For $M \in \mathcal{M}$ and $t \in \mathrm{R}^+$, the job-set $\sigma(M,t)$ is $\sigma(M,t) = \{J \in \mathcal{J} \mid (J,M,t) \in \sigma\}$. σ is *sequential on machine M* if the job set $\sigma(M,t)$ contains at most one element for any t, i.e., M handles at most one job at any time moment. Likewise, σ is *sequential on job J* if the machine set $\sigma(J,t) = \{M \in \mathcal{M} \mid (J,M,t) \in \sigma\}$ contains at most one element for any t, i.e., J is processed by at most one machine at any time moment. A *sequential schedule* is one which is sequential on all jobs and all machines. Some sequential schedules are represented on Figs. 7, 5a, 8 and 9.

If σ is sequential on job J, different J-components do not intersect in time and hence they are naturally ordered. Suppose $[p,q)$ and $[p',q')$ are J-components corresponding to different operations of job J. We will say that $[p',q')$ is a *continuation* of $[p,q)$ if $q \le p'$ and there is no other J-component, scheduled within the interval $[q, p')$. A sequential schedule σ on J is said to be *continuous on job J* if the continuation of every J-component $[p',q')$ is another J-component $[p,q)$ with $q = p'$ (except for the latest scheduled operation of J). Similarly, a sequential schedule σ on M is *continuous on machine M*, if for every J-component $[p,q)$, different from the last scheduled one, there is some J'-component $[p',q')$ on M, with $p' = q$. A *continuous schedule* is one which is continuous on all machines and jobs. The schedule from Fig. 7 is continuous on all 3 machines and is continuous on jobs J^1 and J^2; the schedule from Fig. 8 is continuous on all machines and on job J^0.

In general, a *feasible schedule* σ is a sequential schedule in which each job is completely processed. In other words,

(1) The job-set $\sigma(M,t) = \{J \in \mathcal{J} \mid (J,M,t) \in \sigma\}$ contains at most one element, for every $M \in \mathcal{M}$;
(2) The machine-set $\sigma(J,t) = \{M \in \mathcal{M} \mid (J,M,t) \in \sigma\}$ contains at most one element, for every $J \in \mathcal{J}$;
(3) $\sum_{M\in\mathcal{M}} \frac{|\sigma(J,M)|}{M(J)} = 1$, for every $J \in \mathcal{J}$.

Besides the above conditions, depending on a particular scheduling problem, some additional restrictions for a feasible schedule may exist. For example, in a feasible schedule σ for a job-shop (flow-shop) problem the precedence relations between the operations of each job must be respected, i.e., if $J_i^j \to J_k^j$ then the continuation of any (J^j, M_i)-component of σ is a (J^j, M_k)-component (this condition also provides that σ is sequential on each job). In addition, if no job preemptions are allowed, then the length of the (J,M)-component must be $|\sigma(J,M)|$, for any $J \in \mathcal{J}$ and $M \in \mathcal{M}$.

A feasible schedule σ with the minimal (makespan/period) $\|\sigma\|$ is *optimal*. Observe that any continuous periodic schedule for job-shop $\mathcal{J}, \mathcal{M}$ has the period $\max\{\|\mathcal{M}\|, \|\mathcal{J}\|\}$. This period/makespan is optimal for periodic as well as finite schedules, because any feasible schedule is sequential on both, machines and jobs:

Lemma 1 *For any feasible schedule* σ, $\|\sigma\| \geq \|\mathcal{M}\|$ *and* $\|\sigma\| \geq \|\mathcal{J}\|$. *Hence*, σ *is optimal if* $\|\sigma\| = \max\{\|\mathcal{M}\|, \|\mathcal{J}\|\}$.

We call a feasible schedule *tight*, if no machine is idle from time 0 to its completion time (that is, the completion time of the latest job scheduled on that machine). A tight non-preemptive schedule, associated with a distribution δ can uniquely be given by indicating a sequence of jobs for each machine. The makespan of this (not necessarily feasible) schedule is $|\delta|_{\max}$.

A boundary point of the set $\sigma(J, M)$ is called a *switching point* of J on M. At such a point, M interrupts the processing of J or (re)starts its processing. In the schedule S' on Fig. 1, $[p, q)$ is a J^*-component and p and q are switching points. Job J is *split* on machine M if $\sigma(J, M)$ is a multi-interval, i.e., it consists of two or more J-components. The *number of splittings* of job J on machine M in σ, $sp(\sigma(J, M))$ is the number or components in $\sigma(J, M)$ minus 1. $sp(\sigma(M)) = \sum_{J\in\mathcal{J}} sp(\sigma(J, M))$ is the number of splittings in σ on M; $sp(\sigma) = \sum_{M\in\mathcal{M}} sp(\sigma(M))$ is the total number of splittings in σ.

The *total number of preemptions* in σ, $pr(\sigma) = sp(\sigma) + pr(\delta_\sigma)$: a preemption in σ may come either from δ_σ, or it might be a splitting.

Schedule rotation, shifting and job insertion Given a schedule σ, we may distinguish its m parts on m different machines, and permit to "move circularly" (or rotate) these parts by some constant. We will say that σ' is obtained by a *ϕ-rotation* on machine $M \in \mathcal{M}$ from σ if $(J, M, t) \in \sigma$ iff $(J, M, (t + \phi(M)) \bmod |\sigma|) \in \sigma'$, for a real number $\phi(M)$ (ϕ is a real function). We will say that a rotation of σ is *coherent* if $\phi(M) = \phi(P)$, for every M and P from $\mathcal{M}$. We denote by σ^ϕ the schedule, obtained from σ by a coherent ϕ-rotation (here ϕ is a constant). The following basic properties are easily seen: (1) $|\sigma^\phi| = |\sigma|$; (2) associated distribution is invariant under rotations; (3) there may occur at most one additional splitting in σ_ϕ on each $M \in \mathcal{M}$; (4) if ϕ is a coherent rotation and σ is feasible, σ^ϕ is also feasible.

Instead of rotating, we may just shift completely schedule σ: for a positive number x, the *x-shifting* of σ is a schedule σ^x, such that $\sigma^x = \{(J, M, t) \mid (J, M, t - x) \in \sigma\}$. For a negative x, we define σ^x similarly with the additional condition that the starting time of an earliest scheduled job in σ is no less than $|x|$. Schedule S^x from Fig. 1 is the $x = |[r, p]|$-shifting of the schedule S from the same figure.

We will say that σ' is obtained from σ by *inserting* job J^* on machine M_0 into the time interval $[p, q)$ if for any J and M:

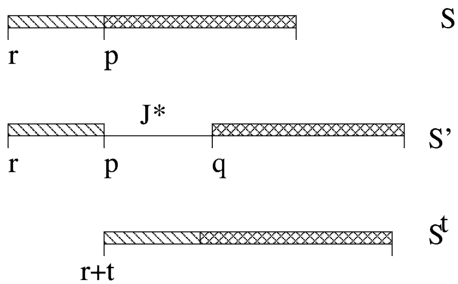


Fig. 1 Job insertion and schedule shifting

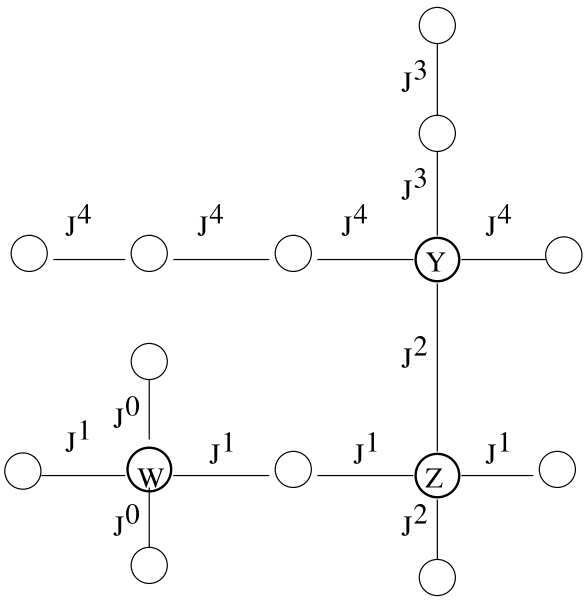
(1) $\sigma(J, M) = \sigma'(J, M)$ if $M \neq M_0$;
(2) $(J, M_0, t) \in \sigma$ iff $(J, M_0, t) \in \sigma'$, for any $t < p$;
(3) $(J, M_0, t - (q - p)) \in \sigma$ iff $(J, M_0, t) \in \sigma'$, for any $t \geq q$;
(4) $\sigma'(M_0, t) = J^*$, for any $t \in [p, q)$.

The schedule S' on Fig. 1 is obtained from schedule S by inserting job J^* into the interval $[p, q)$.

Graphical representation Recall that there is a unique node in a dependency graph G for each machine from $\mathcal{M}$, each edge in G represents a job shared by the corresponding couple of machines. For an instance of job-shop problem $\mathcal{J}, \mathcal{M}$, there is an edge (M_l, M_k) in the corresponding machine dependency graph G labeled by job J^j, iff $J_l^j \to J_k^j$. The *J-component* of G, $G[J]$ is its subgraph formed by the union of all edges of G, labeled by job J. It is easily seen that each J-component forms an acyclic path without any branching in G and that only non-elementary jobs from $\mathcal{J}$ are presented in G. On Figs. 2 and 3 dependency graphs with 5 and 6 J-components are depicted; empty nodes represent machines sharing at most one non-elementary job and the labeled nodes represent machines sharing two or more non-elementary jobs.

Recall also that preemption graphs are defined for distributions. In the *full preemption graph* of a distribution δ there is an edge (M_i, M_j) between nodes M_i and M_j labeled by job J, iff both $\delta(J, M_i)$ and $\delta(J, M_j)$ are positive. The *(reduced) preemption graph* of δ is a subgraph $G(\delta)$ of the full preemption graph, in which all redundant edges are eliminated: an edge (M_i, M_j) labeled by J is *redundant*, if there exists k, $i < k < j$, such that $\delta(J, M_k) > 0$ (according to our machine numbering in $\mathcal{M}$). Any subgraph of the full preemption graph of δ constituted by all nodes representing machines sharing some job J with all edges labeled by J, is a complete graph. The corresponding subgraph in the reduced preemption graph $G(\delta)$ is a simple path in $G(\delta)$. To different enumerations in $\mathcal{M}$ different preemption graphs correspond. However, it is easily seen that $G(\delta)$ is acyclic if and only if the preemption

Fig. 2 An acyclic dependency graph with 5 jobs



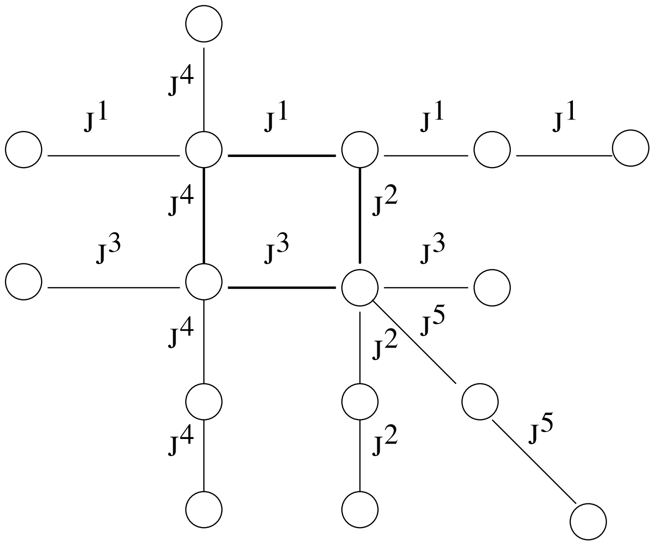


Fig. 3 An non-acyclic dependency graph with 5 jobs

graph, corresponding to any enumeration in $\mathcal{M}$ is acyclic. The following two observations are also evident. First, the number of preemptions in δ is equal to the number of edges in $G(\delta)$. Second, the preemption graph of δ is not connected iff $\mathcal{J}$ can be partitioned into subsets $\mathcal{J}_1 \cup \mathcal{J}_2 = \mathcal{J}$, such that $\delta(J_1) \cap \delta(J_2) = \emptyset$, for each $J_1 \in \mathcal{J}_1$ and $J_2 \in \mathcal{J}_2$.

Let now again G be a dependency graph of a job-shop instance. Operation order does not influence on the acyclicity of G. That is, if the dependency graph of some instance of job-shop is acyclic then the dependency graph of any other instance of job-shop obtained from the former instance by changing arbitrarily the operation order of all jobs is also acyclic. Equivalently, any job-shop instance, obtained from some acyclic open-shop instance by imposing some operation order for each job is acyclic, and vice-versa. Using this fact, we can define the machine dependency graph for an open-shop instance as that of any corresponding job-shop instance. We can associate with the machine dependency graph G representing an instance of job-shop, a *bipartite assignment graph* G^B representing the corresponding open-shop instance (see Lenstra et al. 1990). In G^B, the two sets of vertices are formed by the set of jobs $\mathcal{J}$ and set of machines of $\mathcal{M}$, respectively, and there is an edge (J^j, M_i) iff $p_i^j > 0$. To prove our claim, it will suffice to show that G is acyclic if any only if G^B is acyclic:

Lemma 2 *The bipartite assignment graph G^B is acyclic if and only if the associated machine dependency graph G is acyclic.*

Proof Each simple cycle in G is formed by a sequence of distinct machines $M_0, M_1, \ldots, M_k$, such that M_i and M_{i+1}, $i = 0, 1, \ldots, k-1$, share a job, denoted by J^i, and M_0 and M_k share a job, denoted by J^k. Note that we may have $J^i = J^j$ for some is and js, but as each J-component $G[J]$ is acyclic, at least two edges in the cycle are labeled by different jobs. If all jobs on the cycle are different, then corresponding to this cycle in G there is a simple cycle $M_0, J^0, M_1, J^1, \ldots, M_k, J^k, M_0$ in G^B. Otherwise, we find i and j such that $J^i = J^j$, and all jobs in between are different. Then $J^i, M_{i+1}, J^{i+1}, \ldots, M_j, J^j$ is a simple cycle in G^B.

Conversely, suppose $M_0, J^0, M_1, J^1, \ldots, M_k, J^k, M_0$ is a simple cycle in G^B. Now not necessarily $M_0, M_1, \ldots, M_k$ forms a cycle in G as for some is, operations J_i^i and J_{i+1}^i from the cycle may not be two successive operations in J^i. But we can form a cycle in G using a path between each M_i and M_{i+1} in G (which always exists and belongs to the J^i-component of G). □

The next observation immediately follows:

Observation 1 *All elementary extensions of any shop instance have the same dependency graph.*

Let us say that a shop instance is *finitely* (*periodically or continuously*, respectively) *solvable* if there is a polynomial-time algorithm which constructs an optimal finite (optimal periodic or continuous, respectively) schedule for all elementary extensions of this shop instance. Likewise, a shop instance is said to be *finitely* (*periodically or continuously*, respectively) *unsolvable* if the problem of constructing of an optimal finite (optimal periodic or continuous, respectively) schedule for any its elementary extension is NP-hard. A *dependency graph* is said to be *finitely* (*periodically or continuously, respectively*) solvable if there is a polynomial time algorithm, which for every shop instance with this dependency graph constructs an optimal finite (optimal periodic or continuous, respectively) schedule. Later on, we will use solvable (unsolvable) for finitely solvable (finitely unsolvable).

Observation 2 *If a shop instance has a finitely* (*periodically or continuously*, *respectively*) *solvable dependency graph then it is finitely* (*periodically or continuously*, *respectively*) *solvable*.

Proof Immediately follows from the fact that all elementary extensions of any shop instance have the same dependency graph. □

Note that the converted statement is not true: the solvability of a particular shop instance depends on job data such as operation lengths which are irrelevant in the dependency graphs.

Let us return to Figs. 2 and 3 and observe again that each J-component $G[J]$ is an acyclic path in G. Observe also that if G is acyclic then one of the ends of any $G[J]$ is a leaf. Let us call a J-component $G[J]$ a *marginal component* in G if $G[J]$ contains at most one node, associated with an edge not in $G[J]$. Thus $G[J]$ can be connected with the rest of G by at most one *common node* in G (or can be an isolated component of G). This means that job J shares at most one machine with any other (non-elementary) job from G. The components J_4, J_1 and J_0 are marginal components in the dependency graph of Fig. 2 and the machines W, Z and X are the ones with more than one elementary job.

We apply the following decomposition of an acyclic dependency graph G from Shchepin and Vakhania (2002). We find any marginal component in $G = G_0$, $G[J^0]$ and form the subgraph G_1 of G_0 by deleting all edges and nodes from the $G[J^0]$ component in G_0, except the common node of $G[J]$ in G_0. We proceed with G_1 applying the same procedure: we find a marginal component $G[J^1]$ of G_1 and form the next graph G_2 similarly. We continue until an empty graph G_k is obtained. Note that during this decomposition, former common nodes become non-common in the consequently obtained subgraphs and they are deleted. We call the sequence $G_0, G_1, \ldots, G_k$ a *collapsing* of G and the corresponding sequence of jobs $J^0, \ldots, J^{k-1}$ a *collapsing sequence of jobs*. For the dependency graph of Fig. 2, one of the possible collapsing sequences of jobs is J_0, J_1, J_2, J_3, J_4; G_1 is obtained from $G = G_0$ by deleting all nodes and edges of $G[J^0]$ except the common node W of G_0, G_2 is obtained by deleting from G_1 all nodes and edges of $G[J^1]$ except the common node Z of G_1 (W is no longer a common node in G_1), G_3 is obtained from G_2 by deleting the two edges of $G[J^2]$ (the common node Y is not deleted), G_4 is obtained from G_3 by deleting the two edges of $G[J^3]$ (node Y is again left) and G_5 is the final empty graph. Thus $G_0, G_1, \ldots, G_5$ is a possible collapsing of G. The rough estimation on the time needed for the construction of a collapsing is $O(m^2)$, but this can be done in time $O(m)$, see Shchepin and Vakhania (2002).

3 Polynomial time algorithms

3.1 Periodic job-shop

In this subsection we give a linear-time algorithm for the special subclass of the periodic job-shop scheduling problem. Later in the next section we will show that trivial extensions of this subclass become NP-hard. Each problem from the subclass is represented by machine dependency graph whose any cycle is simple parti-colored: a simple cycle in a dependency graph is *parti-colored* if all its edges have different labels. A machine dependency graph of Fig. 3 has a simple parti-colored cycle $X, J_1, Y, J_2, W, J_3, Z, J_4$. We can see from the figure that any non-elementary job may contribute with at most one edge in the cycle: while some two operations of a non-elementary job may correspond to two distinct machines from the cycle, any other operation of that job is to be scheduled on a machine which is not from the cycle. We call a job-shop problem which machine dependency graph may contain only simple parti-colored cycles a *parti-cyclic job-shop* and abbreviate the periodic version as $J, \mathit{periodic}/\mathit{parti\text{-}cyclic}/C_{\max}$.

An extension $\mathcal{J}', \mathcal{M}'$ of $\mathcal{J}, \mathcal{M}$ is said to be its *simple extension* with job I if (i) $\mathcal{J}' = \mathcal{J} \cup \{I\}$; (ii) there is no operation on any machine of $\mathcal{M}' \setminus \mathcal{M}$ of any job from $\mathcal{J}$, whereas job I has one operation on each of these machines; (iii) besides, job I has one operation on one of the machines in $\mathcal{M}$.

Lemma 3 *Let σ be a continuous finite schedule for job-shop $\mathcal{J}, \mathcal{M}$. Then there is an $O(m)$ algorithm which constructs a continuous finite schedule σ_I for a simple extension $\mathcal{J}', \mathcal{M}'$ of $\mathcal{J}, \mathcal{M}$ with job I.*

Proof Let M be the machine of $\mathcal{M}$ with an operation o of I. Since no operation of job I is scheduled on any machine of $\mathcal{M}$ except machine M, o can be scheduled on machine M at the completion time of M in σ and the resulting schedule will remain continuous (on all its jobs and all its machines). We complete the construction of σ_I by inserting job I on the rest of the machines from $\mathcal{M}' \setminus \mathcal{M}$ in the continuous manner (the operation order in I is respected, σ_I being continuous on I): we insert first the successive to o operations continuously and then, similarly, the preceding to o operations continuously in the reversed precedence order. Inserted in this way operations of job I will not overlap with any job from $\mathcal{J}$ as there is no such a job on any machine from $\mathcal{M}' \setminus \mathcal{M}$. $O(m)$ is clearly an upper bound on the running time. □

For an enumeration of jobs $J^1, J^2, \ldots, J^k$, $k \le n$ of $\mathcal{J}$, let us define the *corresponding sequence of machine subsets* $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$, as follows: $\mathcal{M}_1$ consists of all machines with an operation of job J^1; each $\mathcal{M}_i$ is $\mathcal{M}_{i-1}$ completed with all machines of $\mathcal{M} \setminus \mathcal{M}_{i-1}$ with an operation of job J^i. Clearly, for any given enumeration of jobs there is a unique corresponding sequence of machine subsets and this sequence can be obtained in time $O(nm)$.

Lemma 4 *A continuous finite schedule σ for an instance of acyclic job shop $\mathcal{J}, \mathcal{M}$ can be constructed in time $O(nm)$.*

Proof Let G be the machine dependency graph of our job-shop instance, $J^1, J^2, \dots, J^k$ be the collapsing sequence of jobs of G (Sect. 3), $\mathcal{J}_i = \{J^k, J^{k-1}, \dots, J^{k-i+1}\}$ ($i = 1, 2, \dots, k$), and let $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_k$ be the sequence of machine subsets corresponding to $J^1, J^2, \dots, J^k$. Our algorithm for the construction of σ is simple. First construct a continuous schedule σ_1 for $\mathcal{J}^1, \mathcal{M}_1$ (this is easy since there is only one job in $\mathcal{J}^1$). The job shop $\mathcal{J}^{i+1}, \mathcal{M}_{i+1}$ is a simple extension of $\mathcal{J}^i, \mathcal{M}_i$, for all $i = 1, 2, \dots, k-1$. Hence, for each $i = 2, 3, \dots, k-1$, we can extend the continuous schedule σ_i for $\mathcal{J}^i, \mathcal{M}_i$ to the continuous schedule σ_{i+1} for $\mathcal{J}_{i+1}, \mathcal{M}_{i+1}$ by Lemma 3. The construction of the collapsing sequence and the corresponding sequence of machine subsets, respectively, takes time $O(m)$ and $O(nm)$, respectively. Since the construction of each extension takes time $O(m)$ (Lemma 3) and $k \le n$, the overall time complexity is $O(nm)$. □

Lemma 5 *From a finite continuous schedule σ, a periodic continuous schedule σ^P for job-shop $\mathcal{J}, \mathcal{M}$ with the optimal period $T = \max\{\|\mathcal{J}\|, \|\mathcal{M}\|\}$ can be obtained in time $O(nm)$.*

Proof We obtain σ by Lemma 4, and we build the destiny periodic schedule σ^P by the *periodic extension* of σ, $\sigma^P = \bigcup_{k=1}^{\infty} \sigma^{kT}$, where σ^{kT} is the kT-shifting of σ. It is not difficult to see that σ^P is a periodic sequential schedule. Indeed, let S and C be the starting and completion time, respectively of $M \in \mathcal{M}$ in σ. Since σ is continuous, $C - S \le T$ and hence for any k, the kT-shifting of the interval $[S, C)$ will be disjoint from the $(k+1)T$-shifting. Analogously, σ^P is sequential on any job in $\mathcal{J}$. Finally, we observe that σ^P has the optimal period $T = \max\{\|\mathcal{J}\|, \|\mathcal{M}\|\}$ (see Lemma 1). □

We already have an $O(nm)$ algorithm for $J, periodic/acyclic/C_{\max}$. Indeed, we construct a finite continuous schedule σ for all non-elementary jobs of our job-shop by Lemma 4. Then we insert all elementary jobs obtaining another continuous schedule. Finally, we periodically extend the latter continuous schedule by Lemma 5.

Theorem 1 *There is an $O(nm)$ algorithm for $J, periodic/acyclic/C_{\max}$.*

Now we generalize this result for the periodic parti-cyclic job-shop using the following lemmas.

Lemma 6 *A dependency graph G is continuously solvable in linear-time if it is a simple parti-colored cycle.*

Proof Let $\mathcal{J}, \mathcal{M}$ be a job-shop instance with the dependency graph G, and let $\mathcal{J}'$ and $\mathcal{M}'$ be the set of jobs and machines presented in G (elementary jobs from $\mathcal{J} \setminus \mathcal{J}'$ and their corresponding machines from $\mathcal{M} \setminus \mathcal{M}'$ are not presented in G). Observe that each non-elementary job from $\mathcal{J}'$ is associated with a single edge of G, representing two unique operations of this job to be scheduled on two distinct machines of $\mathcal{M}$. Since G is a parti-colored cycle, for each machine $M \in \mathcal{M}'$, there are only two operations (of two different jobs of $\mathcal{J}'$) have to be performed on M. Let T be the maximal operation length in $\mathcal{J}'$. Then we schedule one of the above operations to be completed at time T, starting the other operation at time T, depending on the precedence order. In particular, let J_l^j and J_k^j, with $J_l^j \to J_k^j$, be operations of a $J^j \in \mathcal{J}'$. Then we schedule operation J_l^j to be finished at time

T on machine $M_l \in \mathcal{M}'$ and we start operation J_k^j at time T on machine $M_k \in \mathcal{M}'$. By this construction, the resulting schedule is continuous on all machines of $\mathcal{M}'$ and all jobs of $\mathcal{J}'$. We extend this continuous schedule to another continuous schedule by adding a single operation of each elementary job from $\mathcal{J} \setminus \mathcal{J}'$ without creating any machine idle time. □

Lemma 7 *Let G be a dependency graph, partitioned into subgraphs G_1 and G_2 with a single connecting edge E. Then G is continuously solvable if both, G_1 and G_2 are continuously solvable.*

Proof Let $\mathcal{J}, \mathcal{M}$ be a job-shop with the dependency graph G. We enumerate the set $\mathcal{M}$ in such a way that the first p machines belong to G_1 and the last $m - p$ machines belong to G_2, with $E = (M_p, M_{p+1})$. Let $\mathcal{M}_1 = \{M_1, \ldots, M_p, M_{p+1}\}$ and $\mathcal{M}_2 = \{M_{p+1}, \ldots, M_m\}$, and J^j be the job, shared by machines M_p and M_{p+1} with $J_p^j \to J_{p+1}^j$. Suppose $\mathcal{J}^i, \mathcal{M}_i$ is the restriction of $\mathcal{J}, \mathcal{M}$ on $\mathcal{M}_i$, and σ_i is a continuous schedule for $\mathcal{J}^i, \mathcal{M}_i$, $i = 1, 2$. Let f be the completion time of J^j in σ_1 and s be the starting time of J^j in σ_2. Further, let σ_1^s and σ_2^f, respectively, be the s-shifting of s_1 and the f-shifting of s_2, respectively. Then it is easily seen that $\sigma = \sigma_1^s \cup \sigma_2^f$ is a feasible continuous schedule for $\mathcal{J}, \mathcal{M}$. □

Theorem 2 *There is an $O(nm)$ algorithm for $J, periodic/parti-cyclic/C_{\max}$.*

Proof The proof is analogous to the proof of Theorem 1 with the additional use of Lemmas 6 and 7. Here is a sketch. First, we use a *semi-collapsing* $G_k \subset G_{k-1} \subset \cdots \subset G_0 = G$ of G defined analogously as a collapsing with the only difference that each $G_i \setminus G_{i+1}$ is either a marginal component (as in collapsing) or it is a parti-colored cycle of G (a semi-collapsing of G can be constructed in linear time, quite similarly as a collapsing). As in Lemma 4, we iteratively apply our semi-collapsing and generate the resulting continuous schedule. We construct the initial partial schedule as in Lemma 4 if G_k is acyclic, or we apply Lemma 6 if G_k is a parti-colored cycle. Iteratively, we apply Lemma 3 if $G_i \setminus G_{i+1}$ is a marginal component. If $G_i \setminus G_{i+1}$ is a parti-colored cycle, then we first construct a continuous schedule for the cycle applying Lemma 6; then we apply Lemma 7 to unify the latter continuous schedule with the (already constructed) continuous schedule for G_{i+1}. As in Theorem 1, the T-periodic extension of σ ($T = \max\{\|\mathcal{M}\|, \|\mathcal{J}\|\}$) is an optimal feasible periodic schedule for $\mathcal{J}, \mathcal{M}$. □

3.2 Preemptive open-shop

In this section we show that acyclic open-shop problem with up to $m - 2$ preemptions $O/acyclic, pmtn(m-2)/C_{\max}$ can be efficiently solved. Later in the next section we prove that the problem with one less preemption becomes NP-hard.

Theorem 3 *$O/acyclic, pmtn(m-2)/C_{\max}$ can be solved in time $O(nm)$.*

Proof First we generate a job-shop instance, corresponding to our open-shop instance by imposing any operation order in each job. Applying Theorem 1, we construct a periodic schedule σ^P with the optimal period $\|\sigma^P\| = \max\{\|\mathcal{J}\|, \|\mathcal{M}\|\}$ for this job-shop instance. σ^P is a periodic extension of a continuous finite schedule. Hence, each J^j is continuous in σ^P. Consider any switching point τ, such that J_i^j completes at time τ on machine M_i and (its immediate successor) operation J_k^j starts at the same time on machine M_k.

To obtain our destiny schedule σ, we coherently rotate a finite segment of σ^P. In particular, let σ be the coherent $(-\tau)$-rotation of the finite schedule $\sigma^P \cap \mathcal{J} \times \mathcal{M} \times [\tau, \tau + \|\sigma^P\|)$. Observe that after the rotation, J_i^j (J_k^j, respectively) will be the last (the first, respectively) scheduled operation on M_i (M_k, respectively), and J_k^j will start at time 0 in σ. Since τ is a switching point for machines M_i and M_k in σ^P, there will be no preemption on these machines in σ. At the same time, in the worst case, there will occur a single preemption on each of the other $m-2$ machines at time 0 (such a preemption will occur on machine $M \in \mathcal{M}$ if τ was not a switching point for the job processed in σ^P at moment τ on M: the part of the corresponding operation scheduled after time τ in σ^P will be scheduled from time 0 on M and the other part will be scheduled the last on M and will be completed right at the moment $\|\sigma\|$ in σ). Thus σ has at most $m-2$ preemptions and σ is optimal as its makespan is T (Lemma 1). □

3.3 Non-preemptive job-shop and open-shop

In this subsection we propose linear-time approximation algorithm for acyclic job-shop with the worst-case performance of $\|\mathcal{M}\| + 2\|\mathcal{J}\|$ and linear-time approximation algorithm for acyclic open-shop with the worst-case performance of $\|\mathcal{M}\| + \|\mathcal{J}\|$; then we show that the bound $\|\mathcal{M}\| + \|\mathcal{J}\|$ is tight for $J/acyclic/C_{\max}$, i.e., there always exist acyclic job-shop instances for which $\|\mathcal{M}\| + \|\mathcal{J}\|$ is the optimal makespan, hence no algorithm can guarantee a better approximation for this problem. Finally, we suggest two exact linear-time algorithms for special case of acyclic job-shop restricting the maximal operation and job lengths.

3.3.1 Approximation algorithms for acyclic job-shop and open-shop

Theorem 4 *A feasible schedule σ with $\|\sigma\| \le \|\mathcal{M}\| + 2\|\mathcal{J}\| \le 3C_{\max}$ for $J/acyclic/C_{\max}$ can be obtained in time $O(nm)$.*

Proof For our job-shop $\mathcal{J}, \mathcal{M}$, let $J^1, \ldots, J^l$ be the collapsing sequence of jobs for our job-shop and let $\mathcal{M}_1, \ldots, \mathcal{M}_l$ be the corresponding sequence of machine subsets (see Sect. 3.1). We describe below the algorithm which builds our destiny schedule σ.

Initial step. Initially we construct schedule s_1 for the restriction of $\mathcal{J}, \mathcal{M}$ on $\mathcal{M}_1$ as follows. Schedule the first in the precedence order operation of J^1 on machine M_1 so that to terminate it at the moment $\|\mathcal{M}\| + \|\mathcal{J}\|$. Schedule all other operations on M_1 starting from the moment $\|\mathcal{J}\|$ continuously without idle times. By this construction, the last of these operations will be completed no later than operation J_1^1 was started. So all operations on M_1 will be processed within the time interval $[\|\mathcal{J}\|, \|\mathcal{J}\| + \|\mathcal{M}\|)$. Schedule each succeeding to J_1^1 operation of job J^1 on its corresponding machine providing the continuity of σ_1 on J^1. Then the completion time of the latest scheduled operation of J^1 will be no more than $\|\mathcal{M}\| + 2\|\mathcal{J}\| - p_1^1$ (see Fig. 4).

General step. At each consequent iteration, we already have a schedule σ_k defined on the restriction of our job-shop on $\mathcal{M}_k$ and we extend it to $\mathcal{M}_{k+1}$. If there is no operation of J^{k+1} to be scheduled on any machine of $\mathcal{M}_k$, we define the schedule on $\mathcal{M}_{k+1} \setminus \mathcal{M}_k$ similarly as in the Initial step. Suppose M_q is the unique machine from $\mathcal{M}_k$ sharing job J^{k+1}. Operation J_q^{k+1} is scheduled at time $\|\mathcal{J}\|$ on M_q (Fig. 4). We shall distinguish two cases for scheduling the other operations of job J^{k+1}: the first case deals with the operations of job J^{k+1} preceding operation J_q^{k+1}, and the second case deals with the operations of J^{k+1}, succeeding J_q^{k+1}.

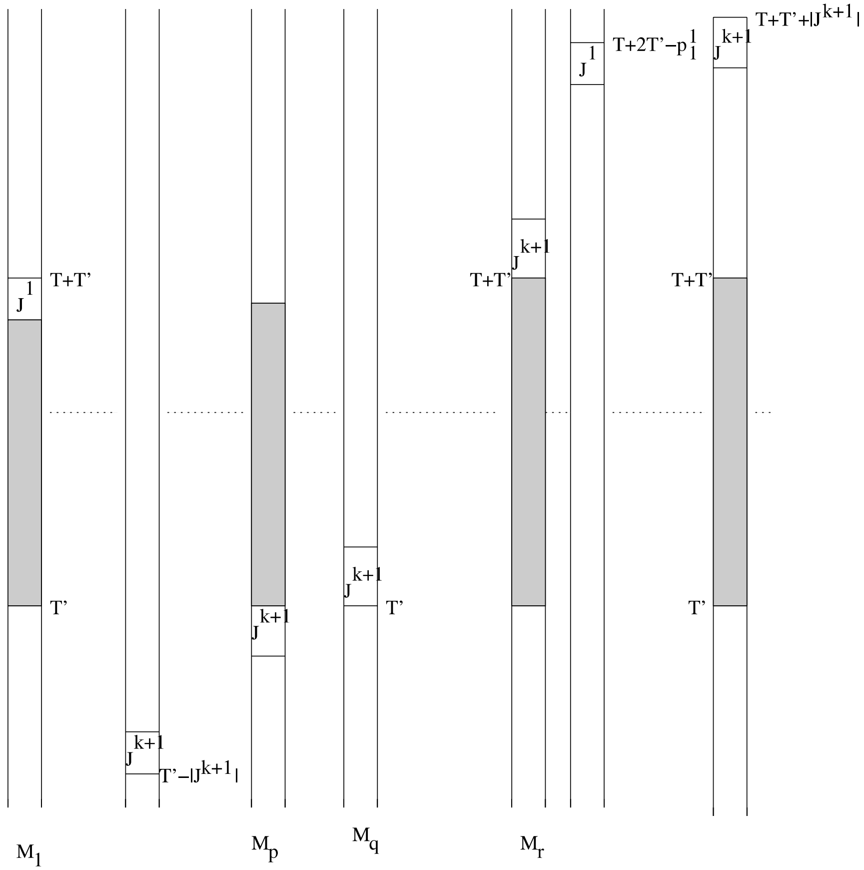
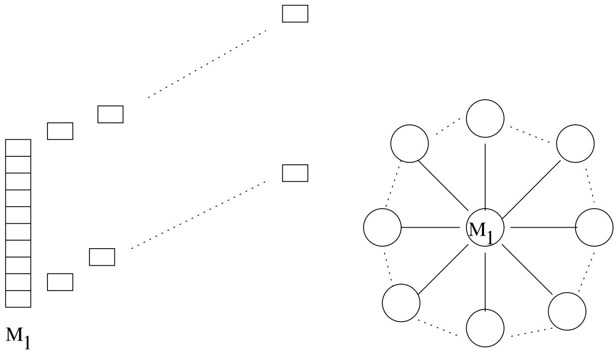


Fig. 4 Approximation algorithm for acyclic job-shop

Case 1. Suppose $J_p^{k+1} \to J_q^{k+1}$, for some p. The schedule on machine M_p is constructed as follows: J_p^{k+1} is scheduled the first so that it terminates at the moment $\|\mathcal{J}\|$ and all other operations are scheduled continuously in an arbitrary order on M_p (Fig. 4). The operations of J^{k+1}, preceding J_q^{k+1} are scheduled similarly as operation J_p^{k+1}, in a continuous manner: each such an operation, say J_i^{k+1}, will be scheduled the first on M_i inside the time interval $(\|\mathcal{J}\| - |J^{k+1}|, \|\mathcal{J}\|)$ and the rest of the jobs on M_i will be scheduled continuously starting from the moment $\|\mathcal{J}\|$.

Case 2. Suppose $J_q^{k+1} \to J_r^{k+1}$, for some r. The schedule on machine M_r is constructed as follows. J_r^{k+1} is scheduled last on M_r starting at the moment $\|\mathcal{M}\| + \|\mathcal{J}\|$ and other operations on M_r are scheduled continuously, in an arbitrary order, starting from the moment $\|\mathcal{J}\|$ (Fig. 4). The operations of J^{k+1}, succeeding J_q^{k+1} are scheduled similarly as operation J_r^{k+1}, in a continuous manner: each such an operation, say J_i^{k+1}, will be scheduled the last on machine M_i inside the time interval $(\|\mathcal{J}\| + \|\mathcal{M}\|, \|\mathcal{J}\| + \|\mathcal{M}\| + |J^{k+1}|)$ and the rest of the jobs on M_i will be scheduled continuously starting from the moment $\|\mathcal{J}\|$.

Fig. 5 (a) A 2-approximation job-shop instance. (b) An acyclic dependency graph for the job-shop instance of Example 1



(a) (b)

By the construction, all operations in each σ_k will be started no earlier than at time 0 and will be completed no later than at time $\|\mathcal{J}\| + \|\mathcal{M}\| + \|\mathcal{J}\|$ (Fig. 4) and $\sigma_l = \sigma$ is a feasible schedule for $\mathcal{J}, \mathcal{M}$ with the makespan of at most $\|\mathcal{M}\| + 2\|\mathcal{J}\|$. □

The 3-approximation algorithm from the above proof can be easily adopted to a 2-approximation algorithm for $O/acyclic/C_{\max}$. In the case of open-shop, we may always treat the operation J_p^{k+1} as the last operation of job J^{k+1} and we can schedule continuously all other operations of J^{k+1} before that operation. Then no operation will be completed after time $\|\mathcal{M}\| + \|\mathcal{J}\|$ and we obtain a 2-approximation algorithm:

Theorem 5 *A feasible schedule* σ *with* $\|\sigma\| \leq \||\mathcal{J}|| + |\mathcal{M}\| \leq 2C_{\max}$ *for* $O/acyclic/C_{\max}$ *can be obtained in time* $O(nm)$.

The final example of this subsection shows that the bound $\|\mathcal{M}\| + \|\mathcal{J}\|$ is tight for $J/acyclic/C_{\max}$:

Theorem 6 *For any* $\varepsilon > 0$ *there exists an instance of an acyclic job-shop* $\mathcal{J}, \mathcal{M}$, *such that for any its feasible finite schedule* σ, $\|\sigma\| > \|\mathcal{M}\| + \|\mathcal{J}\| - \varepsilon$.

Proof Let μ be a natural number such that $1/\mu < \varepsilon$. We will construct a job-shop with $\mu^2 - \mu + 1$ machines and μ jobs. The set of machines consists of one distinguished machine M_1, and $\mu^2 - \mu$ other machines enumerated as M_{ij}, $1 \leq i \leq \mu$ and $1 < j \leq \mu$. The length of all operations is $1/\mu$ and the first (in the precedence order) operation of every job is to be scheduled on machine M_1. The rest of the operations of each job J^i are to be scheduled on machines $M_{i2}, M_{i3}, \ldots, M_{i\mu}$ in this order. As we can see from Fig. 5b, the dependency graph of the defined job-shop is acyclic. $\|\mathcal{J}\| = \|\mathcal{M}\| = 1$ and the makespan of the optimal schedule is $\|\mathcal{J}\| + \|\mathcal{M}\| - 1/\mu > \|\mathcal{J}\| + \|\mathcal{M}\| - \varepsilon$ (see Fig. 5a). □

3.3.2 *Scheduling job-shop with short jobs*

We use the following notations in the rest of this section. $T = \|\mathcal{M}\|$ (which is a lower bound on the optimal makespan), $[J_i^j]$ is the number of operations of job J^j preceding operation J_i^J, $|J^j|_i$ is the total summary length of all operations of job J^j preceding operation J_i^j

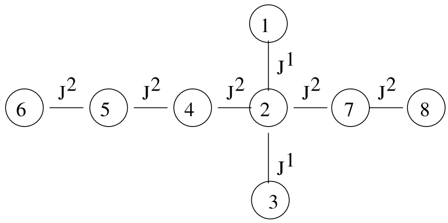


Fig. 6 Machine enumeration in the proof of Theorem 8

and $[J^j]$ is the total number of operations of job J^j minus 1. Let us say that an instance of job-shop $\mathcal{J}, \mathcal{M}$ is one with *short jobs* if $m\|\mathcal{J}\| \leq T$.

Theorem 7 *There is an $O(nm)$ algorithm, which for any acyclic job-shop $\mathcal{J}, \mathcal{M}$ with short jobs constructs an optimal schedule σ with $\|\sigma\| = T$.*

First we describe the algorithm and then give the proof.

Algorithm As in our previous algorithms, we start with the construction of a collapsing sequence of jobs $J^1, \dots, J^\nu$ of the dependency graph G of $\mathcal{J}, \mathcal{M}$ and the corresponding sequence of machine sets $\mathcal{M}_1, \dots, \mathcal{M}_\nu = \mathcal{M}$. Before we schedule each machine one-by-one, we enumerate first the machines of $\mathcal{M}_1$, then the machines of $\mathcal{M}_2 \setminus \mathcal{M}_1$, and so on, finally we enumerate the machines of $\mathcal{M}_\nu \setminus \mathcal{M}_{\nu-1}$ as follows. For $i = 1, \dots, \nu$, if no operation of J^i is to be scheduled on any machine of $\mathcal{M}_{i-1}$, then the machine numbering inside $\mathcal{M}_i \setminus \mathcal{M}_{i-1}$ is according to the operation order in J^i (this rule, in particular, applies to $\mathcal{M}_1$). Otherwise, there is a unique operation J^i_p to be assigned to some machine of $\mathcal{M}_{i-1}$. In this case, we first number all machines of $\mathcal{M}_i \setminus \mathcal{M}_{i-1}$ corresponding to the operations of J^i, preceding J^i_p according to the order, inverse to the operation order in J^i; then we number all machines with operations succeeding J^i_p according to the operation order in J^i. Note that this enumeration takes $O(m)$ time. (In Fig. 6 is depicted a dependency graph with two marginal components corresponding to jobs J^1 and J^2, where for the job sequence J^1, J^2, $\mathcal{M}_1 = \{1, 2, 3\}$ and $\mathcal{M}_2 = \{1, \dots, 8\}$, where $\mathcal{M}_2 \setminus \mathcal{M}_1 = \{4, 5, 6, 7, 8\}$.)

We will have ν outer iterations: on iteration 1 we schedule machine M_1, then machine M_2 and so on, lastly on iteration ν we schedule machine M_ν. We partition non-elementary jobs, to be assigned to each M_i, in two groups: in the first group (a) are included the jobs which have not been yet scheduled on any M_l, $l < i$; the second group (b) is the complement of group (a). Let the *estimated starting time* of an operation $J^j_i \in J^j$ be defined as $t(J^j_i) = [J^j_i]T/m + |J^j|_i$. The scheduling of each M_i goes into 3 steps; Step 1 deals with the elementary jobs, Step 2 deals with the jobs of group (a) and Step 3 deals with the jobs of group (b):

Step 1. Starting from time 0, schedule continuously in any order all elementary jobs to be assigned to M_i.

Step 2. Schedule the jobs of category (a) in the non-decreasing order of their estimated starting times as follows: if no operation is processed on M_i at time $t(J^j_i)$ or $t(J^j_i)$ is a switching point of M_i, then insert J^j_i in the interval $[t(J^j_i), t(J^j_i) + p^j_i)$; otherwise, insert J^j_i at the completion time of the above operation.

Step 3. Since the dependency graph G is acyclic, there is a unique job J^j already scheduled on some machine M_l, $l < i$, whereas there is no other non-elementary job to be scheduled on machine M_i. (On Fig. 6, $J^2 = J^j$, M_l can be any of the machines $2, 4, 5, 7$ whereas M_i can be any of the machines $4, 5, 6, 7, 8$; the possible pairs (M_l, M_i) are (2,4), (4,5), (5,6) and (2,7), (7,8).) Due to our machine numbering, the operation J_l^j either immediately precedes or immediately succeeds the operation J_i^j, consider each case separately.

Case 1. $J_i^j \to J_l^j$. Let s be the starting time of J_l^j. If $s - p_i^j$ is a switching point of M_i or there is no operation processed by M_i at the moment $s - p_i^j$, then insert J_i^j in the interval $[s - p_i^j, s)$. If there exists such an operation, then insert J_i^j just before it.

Case 2. $J_l^j \to J_i^j$. Let f be the completion time of J_l^j. If $f + p_i^j$ is a switching point of M_i or there is no operation processed by M_i at the moment $f + p_i^j$, then insert J^j on M_i in the interval $[f, f + p_i^j)$. If there exists such an operation, then insert J_i^j just after it. This completes the description of the algorithm.

Proof To prove the theorem, we will show that all operations on each M_i are processed within the time interval $[0, T]$. At Step 1, the elementary jobs are continuously scheduled starting from time 0. Since no insertion of any non-elementary job causes an idle time before the elementary jobs, the completion time of any elementary job will not exceed the load time of corresponding machine and hence T. Similarly, the our claim holds if M_i has an idle time. As we have seen, no idle time can occur at Step 1. Let us check our schedule on M_i after Step 2. Let J^j be the job, scheduled right after the last idle interval on M_i. Operation J_i^j is scheduled at its estimated starting time and all operations, scheduled after J_i^j belong to non-elementary jobs; let k be the number of these jobs. J_i^j completes at the moment $[J_i^j]T/m + |J^j|_i + p_i^j \le [J_i^j]T/m + |J^j| \le ([J_i^j]+1)T/m$ (the last inequality hods as we have an instance with short jobs). Hence, the last operation on M_i will complete no later than $([J_i^j] + 1 + k)T/m$. Further, since G is acyclic, $\sum_j [J^j] \le m - 1$, where index j runs through all non-elementary jobs. Then $[J_i^j] + k \le m - 1$ which in turn implies that the completion time of M_i is no more than T.

Now consider the Step 3 which produces the final schedule on M_i. At this step we insert only one additional job J^j, already have been scheduled on M_l. If J^j is inserted no later than at the completion time of the last job, scheduled on Step 2, then it will increase the completion time of M_i by no more than T/m. But since $[J_i^j] + k < m - 1$, the completion time of M_i will be no more than T. Consider the last possibility when job J^j is the latest scheduled job after some idle interval on Step 3. Then it was started exactly at the completion time of its direct predecessor operation. Let J_p^j be the earliest scheduled operation of job J^j. This operation was originally scheduled to be completed no later than at time $t(J_p^j) + T/m + |J_p^j|$. After i insertions before this operation, the completion time of J_p^j will be no more than $t(J_p^j) + (i + 1)T/m + p_p^j = ([J_p^j] + i + 1)T/m + p_p^j + |J^j|_p$. To estimate the completion time of the next scheduled operation, we add to the above magnitude the length of that operation and T/m, which is the maximal possible delay which the algorithm may induce. Then the completion time of the operation J_r^j, directly preceding J_i^j, will be no more than $([J_r^j] + i + 1)T/m + p_r^j + |J^j|_r$. Since J_i^j starts right at the completion time of J_r^j, we get that its completion time is no more than $([J_r^j] + i + 1)T/m + p_r^j + |J^j|_r + p_i^j \le ([J_r^j]+i+1)T/m+|J^j| \le ([J_r^j]+i+2)T/m$. But $[J_r^j] < [J^j]$, hence $([J_r^j]+i+2)T/m \le [J^j] + i + 1$. Now $[J^j] + i \le m - 1$ implies our claim.

Finally, we show that all operations are scheduled no earlier than at time 0 by our algorithm. Let J_r^j be the earliest scheduled operation of J^j. Then its starting time is at least its estimated starting time $t(J_r^j) = [J_r^j]T/m + |J^j|_r$. The preceding operation J_{r-1}^j will be scheduled no earlier than at time $t(J_r^j) - p_{r-1}^j - T/m = [J_r^j]T/m + |J^j|_r - p_{r-1}^j - T/m = ([J_r^j] - 1)T/m + |J^j|_{r-1}| = [J_{r-1}^j]T/m + |J^j|_{r-1}| = t(J_{r-1}^j)$ and the starting time of this operation is also no less than its estimated starting time. In general, the starting time of all operations, scheduled before J_r^j on M_r is no less than their estimated starting times. But the estimated starting time of the earliest scheduled operation on M_i is 0 and hence all operations are scheduled at non-negative time moments. □

3.3.3 Scheduling job-shop with short operations

We will say that a job-shop instance $\mathcal{J}, \mathcal{M}$ is one with *short operations* if for any operation J_i^j, $(2m-1)p_i^j \leq T$.

Theorem 8 *There is an $O(nm)$ algorithm, which for any acyclic job-shop $\mathcal{J}, \mathcal{M}$ with short operations constructs an optimal schedule σ with $\|\sigma\| = T$.*

We first describe the algorithm and then give the proof.

Algorithm We again start with the construction of a collapsing sequence of jobs $J^1, \dots, J^\nu$ and the corresponding sequence of machine sets $\mathcal{M}_1, \dots, \mathcal{M}_\nu = \mathcal{M}$. Let $M_1, \dots M_p, \dots, M_k$ be all machines of $\mathcal{M}_j \setminus \mathcal{M}_{j-1}$ numbered according to the processing order of the operations of J^j, M_p being the unique machine from $\mathcal{M}_{j-1}$. We schedule job J^j at iteration j with k embedded iterations for scheduling each machine from $\mathcal{M}_j \setminus \mathcal{M}_{j-1}$. These k iterations are split into three parts: first are scheduled machines $M_1, \dots, M_{k-1}$, then machine M_p and then machines $M_{p+1}, \dots, M_k$:

Part 1. For $i = 1$ to $p-1$ do {schedule machines $M_1, \dots, M_{p-1}$}.

- (i) Determine the (possibly empty) subsequence of $J^1, \dots, J^\nu$, such that the first (in the precedence order) operation of each job from this subsequence is to be assigned to M_i; starting from time 0, schedule these jobs continuously on M_i in the order of their appearance in the subsequence.
- (ii) Continue by scheduling all elementary jobs on M_i (in an arbitrary order) continuously without leaving any idle time on M_i.
- (iii) If $i \neq 1$, insert J_i^j at the completion time of its immediate predecessor operation (already scheduled iteration $i-1$) if at that time M_i is idle; otherwise, insert J_i^j at the completion time of the operation, processed by M_i at that time.

Part 2. {Schedule J_p^j if $p > 1$ (if $p = 1$, J_1^j is already scheduled on M_p at step (i)).}
If $p > 1$ then schedule J_p^j last on machine M_p: schedule J_p^j at the completion time of its direct predecessor-operation if this time is no less than the completion time of M_p; otherwise, schedule J_p^j at the completion time of M_p.
Part 3. Schedule machines $M_{p+1}, \dots, M_k$ as the first $p-1$ machines.

Proof To prove that the above described algorithm gives a schedule σ with the makespan T, it is sufficient to show that the latest scheduled operation of any job completes no later than T. Since the algorithm does not imply idle intervals before elementary jobs, the completion time of any elementary job will not exceed the load time of the corresponding machine and hence T. Next we deal with the non-elementary jobs. From $\sum_{k=1}^{j}[J^k] \leq m-1$ we obtain $(2\sum_{k=1}^{j}[J^k] + 1)T/(2m-1) \leq T$.

We will use the induction to prove that the completion time of a non-elementary job J^j is no more than $(2\sum_{k=1}^{j}[J^k]+1)T/(2m-1)$. We prove the base for J^1. J^1_1 is scheduled at time 0 and hence completes no later than at $T/(2m-1)$ (which is the maximal operation length); J^1_2 starts no later than $2T/(2m-1)$ and completes no later than at $3T/(2m-1)$. Thus the completion time of ith operation is no more than $(2i-1)T/(2m-1)$. Now the completion time of J^1 does not exceed T since the total number of its operations is no more than m.

Now suppose that our claim holds for J^{j-1}. If there is no operation of J^j scheduled on a machine of $\mathcal{M}_{j-1}$, then we can estimate the completion time of J^j similarly as for J^1. Otherwise, let M_p be the machine from $\mathcal{M}_{j-1}$ with an operation of J^j, and let $M_1,\dots,M_p,\dots,M_k$ be the sequence of all machines with operations of J^j, enumerated according to the operation order in J^j. The starting time of J^j_1 does not exceed $(j-1)T/(2m-1)$, because before J^j_1 there might be scheduled at most $j-1$ operations (the first operations of the previous $j-1$ jobs of the collapsing sequence of jobs). If $p=1$, then all the rest of the operations of J^j are scheduled in Part 3: the completion time of an operation does not exceed the completion time of its direct predecessor operation plus $2T/(2m-1)$. Then the completion time of J^j is no more than $((j-1)+2[J^j])T/(2m-1)$. Since $[J^k]\ge 1$ for any k, $(j-1)+2[J^j]\le(2\sum_{k=1}^{j}[J^k]+1)$.

Now assume that $p>1$. Then the completion time t of the operation, directly preceding J^j_p is no more than $((j-1)+2(p-2))T/(2m-1)$. There are two possibilities for scheduling J^j_p. With the first possibility the starting time of J^j_p is t. In this case the completion time of J^j is no more than $((j-1)+2(p-2)+1+2(k-p))T/(2m-1)=(j+2k-4)T/(2m-1)$ which is less than $((2\sum_{k=1}^{j}[J^k]+1)(T/(2m-1))$. With the second possibility, J^j_p is scheduled right after the completion of an operation of some other job. By the induction hypothesis, this completion time is no more than $(2\sum_{k=1}^{j-1}[J^k]+1)T/(2m-1)$. Hence, the completion time of J^j is no more than $(2\sum_{k=1}^{j-1}[J^k]+1)T/(2m-1)+(2(k-p)+1)T/(2m-1)$. But since $p>1$, $(2(k-p)+1)\le 2(k-1)=2[J^j]$. □

3.4 Preemptive scheduling of (non-lazy) unrelated processors

Theorem 9 *$R/p_{ij}\le C^*_{\max}, pmtn(2m-3)/C_{\max}$ is polynomially solvable.*

Proof On the first stage, we obtain an optimal distribution δ by solving the linear program:

$$\begin{aligned}
&\text{minimize} && D_{\text{opt}},\\
&\text{subject to} && \sum_{i=1}^{n} x_{ij}t_{ij}\le D_{\text{opt}},\quad j=1,\dots,m,\\
& && \sum_{j=1}^{m} x_{ij}=1,\quad i=1,\dots,n,\\
& && x_{ij}\ge 0,\quad i=1,\dots,n,\ j=1,\dots,m,
\end{aligned}$$

where we let $t_{ij}=M_j(J_i)$ and $x_{ij}=\delta(J_i,M_j)$. δ is acyclic (see Potts 1985), hence its preemption graph G_δ is acyclic. Let σ_δ be tight schedule, associated with δ (it can be easily obtained in linear time). We have $|\sigma_\delta|=|\delta|_{\max}$. Since δ is acyclic, σ_δ has at most $m-1$ preemptions. However, not necessarily σ_δ is sequential on all jobs. We construct the desired sequential schedule with the distribution δ rotating iteratively σ_δ (at the expense of creating at most $m-2$ additional preemptions).

Let $G_0 \supset G_1 \supset \cdots \supset G_{m-1}$ be a collapsing of G_δ ($G_0 = G_\delta$), and let σ_k stand for the schedule of iteration k. Since G_{m-1} consists of a single machine, $\sigma_{m-1} = \sigma_\delta$ is sequential for G_{m-1}, i.e., for the single machine in G_{m-1}. Suppose that we have already constructed a rotated schedule σ_{k+1}, which is sequential for the machines in G_{k+1}. Now we define a rotation which results a sequential schedule σ_k for G_k. Let (M, M') be the unique edge from $G_k \setminus G_{k+1}$, M being the machine from $G_k \setminus G_{k+1}$ and M' being the corresponding machine from G_{k+1}, and let J be the job associated with (M, M'). Suppose M'' is the machine in σ_{k+1} on which job J scheduled the earliest. We coherently rotate σ_{k+1} so that in the resultant schedule σ'_{k+1}, the starting time of job J on M'' becomes 0. Note that σ'_{k+1} remains sequential for the machines in G_{k+1}. Let t be the completion time of J on the machine, on which it is scheduled the last in σ'_{k+1}. Then we rotate σ'_{k+1}, now only on machine M in such a way that the starting time of J on M becomes t. Since J is the only job shared by machine M and the machines of G_{k+1}, the resultant rotated schedule σ_k is sequential already for all machines in G_k. This completes the inductive step in the case G_δ is a tree. If it is a forest, we merely apply the above procedure separately to each component of G_δ. Since no job is shared by the machines from different components, the resultant sequential schedule σ is easily obtained by simply joining the sequenced machines of different components.

To complete our proof, we need two additional observations.

First, it easily follows from the above construction that the makespan of the schedule of each iteration, including that of the last iteration $\sigma = \sigma_0$ is the same as that of σ_δ which, in turn, equals to the sequential makespan of δ. Since δ is optimal, $|\delta|_{\max} \le C^*_{\max}$. Besides for each $J \in \mathcal{J}$, because of our imposed restriction $|J|^\delta = \sum_{M\in\mathcal{M}} \delta(J, M)M(J) \le \sum_{M\in\mathcal{M}} \delta(J, M)p_{\max} = p_{\max} \le C^*_{\max}$. Hence, $\|\delta\| \le C^*_{\max}$ and so σ has the optimal makespan.

Second, since δ is acyclic, σ_δ has at most $m - 1$ preemptions. We may have at most one additional split in σ on each of the machines as a result of our rotations. Hence in the worst-case, we will have $2m - 1$ preemptions in σ, i.e., there will be both, a split and a preempted job on each of the machines. We can eliminate two of these splits by a further rotation (similarly as in the proof of Theorem 3). Indeed, let J be any preempted job. Since σ is sequential on J, there are machines M and P, such that the completion time t of J on M is equal to the starting time of J on P. We coherently rotate σ in such a way that t becomes 0. Then neither M nor P will have a split job. Thus the total number of preemptions in our final schedule is $2m - 3$. □

If $p_{ij} \le C^*_{\max}$ does not hold, our distribution δ may assign some job $J \in \mathcal{J}$ to some (slow) machines(s) from $\mathcal{M}$ so that the total processing time of J in δ is already more than $C^*_{\max}$; i.e., the sequential makespan of δ is more than the optimal preemptive schedule makespan. Two observations follow. First, we have the bound $\max\{C^*_{\max}, p_{\max}\}$ for general unrelated processors. Second, a better bound cannot be obtained using distribution δ from the proof of Theorem 9. Let $C^0_{\max}$ be the optimal non-preemptive schedule (equivalently, distribution) makespan. Observe that in an optimal non-preemptive distribution (schedule) no job on any of the machines can take more than $C^0_{\max}$. Then to obtain the bound $C^0_{\max}$, we can impose this additional restriction in our distribution δ. In particular, whenever $\delta(J, M) > 0$ we require that $M(J) \le C^0_{\max}$, and also $|d|_{\max} \le C^0_{\max}$. Using linear programming with binary search, it is possible to find among all distributions with the above property one with the minimal makespan in polynomial time (the reader may look for the details in Lenstra et al. 1990 and Shchepin and Vakhania 2005a). The distribution has at most $m - 1$ preemptions and its sequential makespan is $C^0_{\max}$. Then we can apply the proof of Theorem 9 to obtain a sequential schedule with at most $2m - 3$ preemptions and with the makespan no more than $C^0_{\max}$. Thus we have the following result:

Theorem 10 *There is a polynomial time algorithm, which for any instance of* $R/pmtn/C_{\max}$ *constructs a feasible schedule* σ *with* $pr(\sigma) \le 2m - 3$ *and with the makespan* $\|\sigma\| \le \min\{\max\{C^*_{\max}, p_{\max}\}, C^0_{\max}\}$.

4 NP-hardness results

4.1 NP-hardness of a trivial extension of $J, periodic/parti\text{-}cyclic/C_{\max}$

In this section we show that trivial extensions of $J, periodic/parti\text{-}cyclic/C_{\max}$ become NP-hard. Recall from Sect. 3.1 that in the parti-cyclic job-shop we can have no job with three or more operations on any cycle in the dependency graph G. We shall prove that if we have two jobs with three operations on a cycle in G, then even periodic flow-shop with only two possible operation lengths 1 or 2 is periodically unsolvable:

Example 1 We define $FS(3)$ to be a flow-shop instance with three machines M_1, M_2 and M_3, and two jobs J^1 and J^2. $p^1_1 = p^2_1 = 2$, while other operations of J^1 and J^2 have the length 1 (there is no dummy operation in J^1 or in J^2). The processing order of each job is M_1, M_2, M_3.

Theorem 11 $FS(3)$ *is periodically unsolvable.*

Proof We use the reduction from PARTITION. Let $X = \{x_1, x_2, \ldots, x_k\}$ be an instance of PARTITION with $S = \sum_{i=1}^k x_i$. We define an elementary extension $FS(3, X)$ of $FS(3)$ as follows: we add k partition jobs $J^3, \ldots, J^{k+2}$ on machine M_2 with $p_2^{i+2} = 2x_i/S$ (with the total length of 2). We show that the problem of construction of a feasible periodic schedule with the optimal period 4 for $FS(3, X)$ is equivalent to the construction of a partition for X.

Suppose first that $\sum_{i=1}^l x_i = S/2$ is a partition of X (for the notation simplicity, we renumber the partition elements respectively). Then we define a periodic schedule σ with the period 4 by specifying the processing intervals of all jobs as follows:

$$\begin{aligned} \sigma(J^1, M_1) &= [0, 2) \quad \text{and} \quad \sigma(J^2, M_1) = [2, 4), \\ \sigma(J^1, M_2) &= [2, 3) \quad \text{and} \quad \sigma(J^2, M_2) = [4, 5), \\ \sigma(J^1, M_3) &= [3, 4) \quad \text{and} \quad \sigma(J^2, M_3) = [5, 6). \end{aligned}$$

On Fig. 7 is depicted the resulting schedule, dashed regions represent gaps and dark regions represent the partition jobs. The first l partition jobs are continuously scheduled from moment 3 on M_2, they exactly fill in the interval $[3, 4)$; other partition jobs are continuously scheduled from moment 5 and exactly fill in the interval $[5, 6)$; all jobs are then scheduled periodically with the period 4. The constructed schedule with the period 4 is optimal since 4 is the load time of M_1 (Lemma 1).

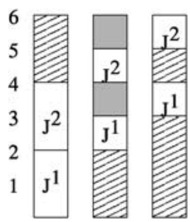


Fig. 7 Flow-shop problem with 2 non-elementary jobs

In the other way, suppose we have a feasible schedule σ for $FS(3, X)$ with the period 4 and t is the completion time of J_1^1 on M_1. Note that σ has to be continuous on J^1 and J^2 because the length of these jobs is 4. Besides, σ has to be continuous on M_1 because its load time is also 4. It follows that t must be the starting time of J_2^1 on M_2 and at the same time it must be the starting time of J_1^2 on M_1. Then the completion time of J_1^2 on M_1 is $t+2$, which is also the starting time of J_2^2. The schedule has to be continuous on M_2 as well because its load time is 4. Hence, in the interval $[t+1, t+2)$ between second operations of J^1 and J^2 must be continuously scheduled partition jobs to fill in completely this interval of length 1. This gives a solution to the PARTITION and the lemma is proved. □

4.2 NP-hardness of $P/pmtn(m-2)/C_{\max}$

In this subsection we show that $P/pmtn(m-2)/C_{\max}$ is NP-hard. We use the reduction from the NP-complete PARTITION problem for the decision version of $P/pmtn(m-2)/C_{\max}$. In the PARTITION problem we are given a finite set of integer numbers $C = \{c_1, c_2, \dots, c_n\}$ with $S = \sum_{i=1}^{n} c_i$.[1] This decision problem gives a "yes" answer iff there exists a subset of C which sums up to $S/2$. Given an arbitrary instance of a PARTITION, let us define our scheduling instance with $n + 2m + 2$ jobs with the total length of $2m + \frac{1}{2^m}$ as follows.

There are m pairs of the so-called *big* jobs denoted by $B_i^{\pm}$, $i = 1, \dots, m$ with $|B_i^+| = 1 + 1/2^i$, and $|B_i^-| = 1 - 1/2^i$. So the total length of all big jobs is $2m$.

There are two *median* jobs denoted by D and D', with $|D| = \frac{1}{2^m} - \frac{5}{m2^{m+2}}$ and $|D'| = \frac{3}{m2^{m+2}}$. So total length of the two median jobs is $\frac{1}{2^m} - \frac{1}{m2^{m+1}}$.

There are n *small* jobs C_i with $|C_i| = \frac{c_i}{mS2^{m+1}}$. So the total length of small jobs is $\frac{1}{m2^{m+1}}$.

This transformation is polynomial as the number of jobs is bounded by the polynomial in n and m, and all magnitudes can be represented in binary encoding in $O(m)$ bits.

Now we prove that there exists a feasible schedule with less than $m-1$ preemptions and with the optimal makespan $2 + \frac{1}{m2^m}$ iff there exists a solution to our PARTITION. In one direction, suppose $\sum_{i=1}^{k} c_i = S/2$, for some $k < n$, i.e. we have a solution to the PARTITION. Then we define a tight schedule σ with the makespan $2 + \frac{1}{m2^m}$ as follows. The job sequence on M_1 is: $B_1^-, B_1^+, D', C_1, \dots, C_k$; so the completion time of M_1 is $2 + \frac{3}{m2^{m+2}} + \frac{1}{m2^{m+2}} = 2 + \frac{1}{m2^m}$ (the load of M_1). The job sequence on M_2 is: $B_2^-, D, C_{k+1}, \dots, C_n, B_2^+$, where we have only a part of D with the length $\frac{3}{m2^{m+2}} = \frac{3}{4}\frac{1}{m2^m}$ (providing that the load of M_2 is $2 + \frac{1}{m2^m}$). The rest of D is divided into equal parts of the length $\frac{1}{m2^m}$ and is distributed on the machines M_i, $i > 2$. The schedule on M_i, $i > 2$ is tight and is generated by the sequence B^-, D, B^+.

To see that σ is feasible, we need to check the sequentiality of σ on D, which is the only preempted job in σ. The completion time of D on M_i is no more than $1 - \frac{1}{2^i} + \frac{1}{m2^m} \le 1 - \frac{1}{2^i} + \frac{1}{m2^{m+1}} \le 1 - \frac{1}{2^i} + \frac{1}{2^{i+1}} = 1 - \frac{1}{2^{i+1}}$, and its starting time on M_{i+1} is $1 - \frac{1}{2^{i+1}}$. Hence, σ is sequential and has the makespan $2 + \frac{1}{m2^m}$. This completes the proof in one direction. We need the following lemma for the other direction:

Lemma 8 *If there exists a uniform distribution δ of jobs of $\mathcal{J}$ on m identical processors from $\mathcal{M}$ with less than $m-1$ preemptions, then there is a subset $\mathcal{J}'$ of $\mathcal{J}$ with the total length of $k|\delta|_{\max}$, for some natural $k < m$.*

[1] In this and the following subsection n stands exclusively for the number of elements in PARTITION; in the rest of the paper n denotes the number of jobs.

Proof Since $pr(\delta) < m-1$, $G(\delta)$ has less than $m-1$ edges and hence any its connected component contains $k < m$ machines (we may have two or more such components). Let $\mathcal{J}'$ be the set of all jobs distributed on machines in any of the connected components. Since δ is uniform, the total length of these jobs is $k|\delta|_{\max}$. □

Assume now σ is a tight $(m-2)$-preemptive schedule for our scheduling instance. A distribution, generated by σ also has no more than $m-2$ preemptions and the load on all machines in this distribution is $2+\frac{1}{m2^m}$. We will prove that this distribution already gives a solution to our instance of PARTITION.

Suppose $\mathcal{J}'$ is a subset of $\mathcal{J}$ with the total length of $|\mathcal{J}'| = 2k + k\frac{1}{m2^m}$ (see Lemma 8). First we note that $\mathcal{J}'$ contains exactly $2k$ big jobs. Indeed, $\mathcal{J}'$ cannot contain more than $2k$ big jobs, because the total length of the smallest $2k+1$ big jobs is $2k+1-\sum_{i=1}^{k+1}\frac{1}{2^i} = 2k+\frac{1}{2^{k+1}} \geq 2k+\frac{1}{2^m} > 2k+k\frac{1}{m2^m} = |\mathcal{J}'|$. At the same time, the total length of the longest $2k-1$ big jobs together with all median and small jobs is less than $2k$. Further, the total length B' of all big jobs from $\mathcal{J}'$ is $2k$. Indeed if $B'-2k$ is not 0, it must have an absolute value of at least $\frac{1}{2^m}$. If $B' < 2k$, then B' plus the total length of all non-big jobs (which is no more than $\frac{1}{2^m}$) will be no more than $2k$. This contradicts our assumption that $|\mathcal{J}'| = 2k + k\frac{1}{m2^m}$. Similarly, if $B' > 2k$, then the above magnitude is at least $2k+\frac{1}{2^m} > 2k+k\frac{1}{m2^m} = |\mathcal{J}'|$, which again is a contradiction. Thus, $B' = 2k$.

It follows that the total length of the big jobs from $\mathcal{J}^c$ is $2(m-k)$, $\mathcal{J}^c$ being the complement of $\mathcal{J}'$ in $\mathcal{J}$, and hence without loss of generality, we can assume that $k \leq m/2$. In this case, D cannot belong to $\mathcal{J}'$, because $|D| > k\frac{1}{m2^m}$. On the other hand, D' must belong to $\mathcal{J}'$. Indeed, denote by C' the total length of all small jobs from $\mathcal{J}'$. If $\mathcal{J}'$ does not contain a median job, then $|\mathcal{J}'| = B' + C' \leq 2k + \frac{1}{m2^{m+1}}$, which contradicts our conjecture that $|\mathcal{J}'| = 2k + k\frac{1}{m2^m}$. Therefore, $D' \in \mathcal{J}'$. In this case $|\mathcal{J}'| = B' + C' + |D'|$. This implies that $k\frac{1}{m2^m} = C' + \frac{3}{m2^{m+2}}$. But since $C' \leq \frac{1}{m2^{m+1}}$, the only possible value of k is 1. Then $C' = \frac{1}{m2^m} - \frac{3}{2^{m+2}m} = \frac{1}{m2^{m+2}}$, which is $S/2$ and we have obtained a solution to the PARTITION. We have proved this section's main result:

Theorem 12 *$P/pmtn(m-2)/C_{\max}$ is NP-hard.*

4.3 The NP-hardness of $O/acyclic, pmtn(m-3)/C_{\max}$

In this section we prove that acyclic open-shop with at most $m-3$ preemptions is NP-hard:

Theorem 13 *$O/acyclic, pmtn(m-3)/C_{\max}$ is NP-hard.*

Below we describe the reduction from the PARTITION problem. We transform the PARTITION problem to $O/pmtn/C_{\max}$. Let $C = \{c_1, c_2, \dots, c_n\}$ and $S/2$, $S = \sum_{i=1}^{n} c_i$, form again an arbitrary instance of a PARTITION. We define our open shop instance $O(C,m)$ as follows (assuming without loss of generality that $m \geq 3$). $O(C,m)$ deals with $1 + n(m-2) + 2m - 2 = (n+2)m - 2n - 1$ jobs and m machines $\{M_i\}_{i=1}^{m}$. We divide the set of jobs into the following three categories:

(1) There is one *common* job I to be distributed on all machines. The processing requirement of I on M_i, $i = 1, 2, \dots, m$, is 1. Hence, the total processing time of I is m.
(2) The jobs from the second category are called *partition jobs*. We introduce n partition jobs for each of the machines, except the first one M_1 and the last one M_m (we call these machines *extremal*, and the rest of machines *intermediate*). The jth, $j \leq n$, partition

job to be distributed on machine M_i $(1 < i < m)$ is denoted by $P_{i,j}$. The processing time of $P_{i,j}$ is $\frac{c_j}{S^{2i}}$. Note that the total processing time of all partition jobs on each M_i $(1 < i < m)$ is equal to $\frac{1}{S^{2i-1}}$, which is a magnitude, strictly less than 1.

(3) The jobs from the third category are called *fixers* and are denoted as $F_1, F_2^+, F_2^-, F_3^+, F_3^-, \ldots, F_{m-1}^+, F_{m-1}^-, F_m$. So there is only one fixer on machines M_1 and M_m, and there are two fixers on any intermediate machine. The processing time of F_1 and F_m is $m-1$. The processing time of the first fixer for machine M_i, F_i^+, $1 < i < m$, is equal to $(i-1) - \frac{1}{2S^{2i-1}}$, and the processing time of the second fixer F_i^- is $(m-i) - \frac{1}{2S^{2i-1}}$.

The operations which we have not explicitly defined are dummy ones providing that $O(C,m)$ is acyclic. Our transformation is obviously polynomial (similarly as in the previous subsection). The load of each machine in $O(C,m)$ is m and any feasible schedule for $O(C,m)$ with the makespan m is tight and hence optimal. Given a solution $\sum_{i\le k} c_i = S/2$ to our PARTITION instance (we renumber elements in PARTITION correspondingly), a feasible schedule for $O(C,m)$ without any splitting and with the optimal makespan m can easily be built. On M_1 first is scheduled the fixer F_1 and then the common job I. On the last machine, first I is scheduled and then F_m. On each intermediate machine M_i, jobs are scheduled in the following order: first the fixer F_i^+, then all $P_{i,j}$, $j \le k$ (in any order), then job I followed by the rest of the partition jobs $P_{i,j}$, $j > k$, and finally the second fixer F_i^-.

In the opposite direction, a feasible schedule σ with at most $m-3$ preemptions for our open-shop yields a solution to PARTITION. Indeed, since there are at most $m-3$ splittings, no job on at least one intermediate machine M_i is split. An important observation is that, without loss of generality, it might be assumed that the common job I is scheduled within the interval $[i-1, i]$ on M_i. Suppose this is not the case. If I is processed out of interval $[1, m]$ then σ is not tight and hence optimal. At the same time, any part of I scheduled out of the interval $[i-1, i]$ will yield at least two splits on one of the other machines. Indeed, there must be a machine M_j on which job I is split, and hence at least one of the fixers will be split on the same machine. We apply the same reasoning now for machine M_j and find another machine with two splits. We can continue in the same manner finding machines with at least two splits. But this contradicts our assumption that there are at most $m-3$ splits.

Thus we can assume that job I is processed within the interval $[i-1, i]$ in σ. Since we have no split on M_i, the fixers F_i^+ and F_i^- must be scheduled before and after job I. We are left with the intervals of the same length $\frac{1}{2S^{2i-1}}$ before and after job I within which the partition jobs must be scheduled. Since no partition job can be split, these jobs must be partitioned into two subsets with the same total length. This solves the PARTITION problem and proves Theorem 13.

4.4 NP-hardness of simple acyclic shop problems

In this section we show that very simple classes of acyclic shop scheduling problems are NP-hard.

Lemma 9 *There is an unsolvable flow-shop instance with a single job J^0 with 3 operations.*

Proof We use the reduction from KNAPSACK. Let $X = \{x_1, \ldots, x_k\}$ and $C \le \sum_i x_i$ be an arbitrary instance of KNAPSACK. In our scheduling instance we have 3 machines M_1, M_2 and M_3 and all jobs have to be processed in this order. Job J^0 is such that $p_1^0 = C$ and $p_1^0 + p_3^0 = \sum_i x_i$. We consider the following elementary extension with $k+2$ jobs of this flow-shop instance. Job J^1 is added to M_1 with $p_1^1 = p_2^0 + p_3^0$; job J^2 is added to M_3 with

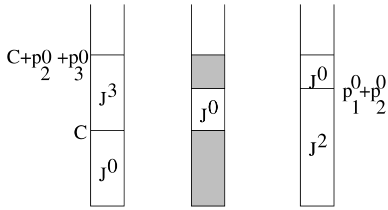


Fig. 8 Unsolvable acyclic job-shop instance with a single non-elementary job

$p_3^2 = p_1^0 + p_2^0$; finally, k jobs $J^3, \dots, J^{k+2}$ with $p_2^i = x_i$, $i = 3, \dots, k+2$ are added to M_2. The rest of operations of all jobs are dummy. It is clear that the problem of constructing of a feasible schedule with the optimal makespan $p_1^0 + p_2^0 + p_3^0$ is equivalent to finding a subset X' of X with $\sum_{i \in X'} x_i = C$ (see Fig. 8 on which dark regions represent partition jobs). □

Given a schedule σ, let us denote by $[\sigma]$ the schedule, which components are defined as $\{(M, J, [[p], [q])\}$, for each component $(M, J, [p, q))$ of σ ($[x]$ is the integral part of x).

Lemma 10 *Any subgraph G' of a solvable dependency graph G is also solvable.*

Proof Let $\mathcal{J}', \mathcal{M}'$ and $\mathcal{J}, \mathcal{M}$ be job-shop instances with dependency graphs G' and G, respectively. Since operation lengths are irrelevant in dependency graphs, without loss of generality, we can assume that the operation lengths in $\mathcal{J}'$ are integers and that the total length of all operations from $\mathcal{J} \setminus \mathcal{J}'$ is strictly less than 1.

Let σ be an optimal schedule for $\mathcal{J}, \mathcal{M}$. Then obviously, $[\sigma]$ is a feasible schedule for $\mathcal{J}', \mathcal{M}'$ with $\|[\sigma]\| \le \|\sigma\|$. We claim that $[\sigma]$ is also optimal. Assume that σ' is an optimal schedule for $\mathcal{J}', \mathcal{M}'$ with $\|\sigma'\| < \|[\sigma]\|$. Since all operation lengths in σ' are integers and $\|[\sigma]\| \le \|\sigma\|$, $\|\sigma'\| + 1 \le \|\sigma\|$. We will come to a contradiction by extending σ' to a feasible schedule for $\mathcal{J}, \mathcal{M}$ with the makespan, less than σ. This schedule is constructed step-by-step, at each step a single job from $\mathcal{J} \setminus \mathcal{J}'$ is inserted; we denote by σ^i the schedule obtained after the ith insertion, $\sigma^0 = \sigma'$. Suppose J_i^j is an operation of job $J^j \in \mathcal{J} \setminus \mathcal{J}'$ inserted at step i in σ^{i-1}. Let t be the completion time of the latest predecessor-operation of J_i^j already scheduled in σ^{i-1} ($t = 0$ if there is no such operation). σ^i is obtained from σ^{i-1} by inserting J_i^j at time t and shifting all operations, scheduled after t in σ^{i-1} by p_i^j. It is easily seen that σ^k, $k = |\mathcal{J} \setminus \mathcal{J}'|$ is a feasible schedule for $\mathcal{J}, \mathcal{M}$. Besides, $\|\sigma^k\| < \|\sigma'\| + 1$, as the overall shifting in σ^k does not exceed the summary length of all operations in $\mathcal{J} \setminus \mathcal{J}'$ which is strictly less than 1. But since $\|\sigma'\| + 1 \le \|\sigma\|$, $\|\sigma^k\| < \|\sigma\|$ and we came to a contradiction. □

The next result immediately follows from Lemmas 9 and 10:

Theorem 14 *Any flow-shop problem with at least* 1 *job with at least* 3 *operations is* (*finitely*) *unsolvable.*

Our second example is a flow-shop instance with 7 jobs which is finitely unsolvable with respect to the extensions with short jobs (an analogous example with short operations can be similarly constructed and proved).

Example 2 We define $FS(7)$ to be the following flow-shop instance with three machines M_i ($i = 1, 2, 3$) and seven jobs J^i ($i = 1, 2, \dots, 7$). The processing order of each job coincides with the machine numbering and all operations of all these jobs have length 1.

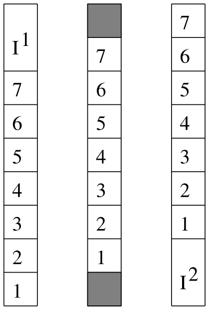


Fig. 9 NP-hard flow-shop problem for elementary extensions with short jobs

Theorem 15 *The problem of construction of an optimal finite schedule for any elementary extension of $FS(7)$ with short jobs is NP-hard.*

Proof We again use PARTITION $X = \{x_1, \dots, x_k\}$ and $S = \sum_i x_i$. Let us consider an elementary extension $FS(7, X)$ of $FS(7)$: we add two jobs I^1 and I^2 with the length 2 on machines M_1 and M_3, respectively, and k partition jobs $P_1, \dots, P_k$ on machine M_2 with $|P_i| = x_i/S$, where we let $S = 2$. Since the load time of all machines in $FS(7, X)$ is 9, and the length of any job does not exceed 3, we have a flow-shop instance with short jobs.

The construction of a feasible schedule with the optimal makespan 9 is equivalent to finding a solution to the PARTITION. Indeed, suppose we have a partition $\sum_{i=1}^{l} x_i = S/2$. Then we easily define a feasible schedule with makespan 9 by continuously scheduling the jobs on M_1, M_2 and M_3 in the following order (see Fig. 9 on which dark regions represent partition jobs):

$$\begin{aligned}
&J^1, J^2, J^3, \dots, J^7, I^1 \quad \text{on } M_1,\\
&P_1, P_2, \dots, P_l, J^1, J^2, \dots, J^7, P_{l+1}, \dots, P_k \quad \text{on } M_2,\\
&I^2, J^1, J^2, \dots, J^7 \quad \text{on } M_3.
\end{aligned}$$

In the other direction, suppose there is a feasible schedule for $FS(7, X)$ with the makespan 9. Without loss of generality, assume that jobs $J^1, \dots, J^7$ are scheduled in this same order machine on M_1. Then the starting time of J_1^7 on M_1 cannot be less than 6. But it cannot be more than 6, because $|J^7| = 3$ and this job has to be completed by time 9. Hence, our schedule on M_1 is continuous and has the following operation order: $J^1, J^2, J^3, \dots, J^7, I^1$.

On machine M_3, the operation J_3^7 cannot be completed before time 9 and hence the latest scheduled operation on machine M_3 has to be J_3^7. Similarly, the preceding operation has to be J_3^6, and so on. Thus the only possible ordering of operations on M_3 is $I^2, J^1, J^2, J^3, \dots, J^7$.

Now, on machine M_2, J_1 has to be started at time 1, J^2 at time 2, and so on, J^7 has to be started at time 7. These jobs are to be scheduled continuously in the time interval $[2, 7)$. Hence, the partition operations must be divided into two non-intersecting subsets with the overall length of 1 in each subset: operations from one subset are to be scheduled in the interval $[0, 1)$ and operations from the second subset are to be scheduled in the interval $[7, 8)$. This gives a solution to our PARTITION instance. □

4.5 NP-hardness of $R/p_{ij} \le C^*_{\max}, pmtn(2m-4)/C_{\max}$

In this section we use the earlier NP-hardness result for $O/acyclic, pmtn(m-3)/C_{\max}$ and show that $R/p_{ij} \le C^*_{\max}, pmtn(2m-4)/C_{\max}$ is NP-hard. First we prove the following auxiliary lemma:

Lemma 11 *Two distributions δ and δ' are equal if they have the same acyclic assignment and the load on all machines in both δ and δ' is the same.*

Proof Assume δ and δ' are two arbitrary distributions satisfying the lemma. The proof is by the induction on the number of machines. Suppose the lemma is proved for distributions with $<m$ machines.

Let δ and δ' be two acyclic distributions with m machines from $\mathcal{M}$. Since δ and δ' have the same acyclic preemption graph, there is a node of degree one N in this graph. Let I be the job, corresponding to the (only) edge of M. Note that the rest of the jobs, distributed in δ and δ' on M have no preemption, i.e., they are assigned completely to M. Therefore, the length of all these jobs in δ and δ' is the same. But since the load of M in δ and δ' is the same, the length of the portion of I on M in both δ and δ', must be also the same. Hence $\delta(J, M) = \delta'(J, M)$ for all J. To apply the induction hypothesis, we define distributions δ_0 and δ'_0 on $\mathcal{M} \setminus M$ in the following way: $\delta_0(J, M) = \delta(J, M)$ if $J \neq I$ and $\delta(J, M) > 0$, $\delta_0(I, M) = \delta(I, M)\frac{1}{1-\delta(I,M)}$; $\delta'_0(J, M) = \delta'(J, M)$ if $J \neq I$ and $\delta'(J, M) > 0$, $\delta'_0(I, M) = \delta'(I, M)\frac{1}{1-\delta'(I,M)}$. Distributions δ_0 and δ'_0 coincide the by induction hypothesis. This implies that $\delta = \delta'$. □

Given a multiprocessor $\mathcal{J}, \mathcal{M}$, let us say that a distribution δ on $\mathcal{J}, \mathcal{M}$ *generates* an open shop O on $\mathcal{J}, \mathcal{M}$, if $|J^i_j| = \delta(J^j, M_i)M_i(J^j)$.

Theorem 16 *For any uniform acyclic open shop O on $\mathcal{J}, \mathcal{M}$, there is a processing time function f and a distribution δ for the multiprocessor $\mathcal{J}, \mathcal{M}$ with this processing function, such that δ generates O and δ is a unique optimal distribution for $\mathcal{J}, \mathcal{M}$.*

Proof First we define f as follows: $M_i(J^j) = |J^j| + \varepsilon$ if J^j_i is dummy, and $M_i(J^j) = |J^j|$ otherwise, where ε is a positive real number. Now we define the distribution δ as $\delta(J^j, M_i) = |J^j_i|/|J^j|$. Note that if $|J^j_i| > 0$, then $\delta(J^j, M_i)M_i(J^j) = M_i(J^j)\frac{|J^j_i|}{|J^j|} = |J_j|\frac{|J^j_i|}{|J^j|} = |J^j_i|$. Similarly, $|J^j_i| = 0$ implies $\delta(J_j, M_i) = 0$. Hence, $(M_i)_\delta(J^j) = |J^j_i|$ holds for all i, j and δ generates O. Since O is uniform, δ is also uniform and the total processing time of δ is $m|\delta|_{\max}$. Besides, every job in δ is distributed on its fastest machine (i.e. the machine where this job can be processed in the minimal time). This implies that δ has the minimal possible total processing time, and since δ is uniform, it is optimal.

It remains to show that δ is unique. Assume δ' is another optimal distribution such that δ' generates O. The total processing time in δ' is no more than $m|\delta'|_{\max}$, but the latter by our assumption is no more than $m|\delta|_{\max}$; now since $m|\delta|_{\max}$ is the minimal possible total processing time, the total processing time in δ' is to be equal to $m|\delta|_{\max}$. Hence, $m|\delta'|_{\max} = m|\delta|_{\max}$ and δ' is also uniform, i.e., loads on all machines in both, δ and δ' are equal. At the same time, δ' has to distribute each job on the fastest machine. But the processing time function is defined in such a way that a machine M is fastest for a job J iff $\delta(J, M) > 0$. Hence, δ' and δ have the same acyclic assignment and our claim follows from Lemma 11. □

Theorem 17 *$R/p_{ij} \le C^*_{\max}, pmtn(2m-4)/C_{\max}$ is NP-hard.*

Proof We prove that the corresponding decision problem is NP-complete. Recall that the open-shop problem $O(C,m)$ of Sect. 4 is acyclic and uniform. We apply Theorem 16 with $\varepsilon \le 1$ to $O(C,m)$ to construct a processing time function f for multiprocessor $\mathcal{J}, \mathcal{M}$, and a distribution δ, such that δ generate $O(C,m)$. Note that $|\delta|_{\max} = m$ and $p_{\max} = m$, hence the multiprocessor $\mathcal{J}, \mathcal{M}$, defined by the processing time function f, is non-lazy (i.e., $p_{\max} \le C_{\max}$).

It is not difficult to see that our decision problem, "does there exist a feasible schedule σ for the multiprocessor $\mathcal{J}, \mathcal{M}$ with $||\sigma|| \le m$ and with $pr(\sigma) \le 2m-4$?", has a "yes" answer if and only if there exists a tight schedule for $O(C,m)$ with the makespan, not exceeding m and with less than $m-2$ splittings. Indeed, if σ is such a schedule for $O(C,m)$, then σ is a feasible schedule for the multiprocessor $\mathcal{J}, \mathcal{M}$ with less than $(m-1)+(m-2) = 2m-3$ preemptions.

In the other direction, suppose σ is a feasible schedule for $\mathcal{J}, \mathcal{M}$ with $\|\sigma\| \le m$ and $pr(\sigma) < 2m-3$. Then the makespan of any distribution associated with σ is m and it is optimal. Since there is only one such a distribution (Theorem 16), it is precisely the distribution'δ. δ has exactly $m-1$ preemptions. It follows that σ is a feasible schedule for $O(C,m)$ with the number of splittings $pr(\sigma)-(m-1) < m-2$. □

5 Further research

We have seen that acyclic scheduling problems, though very restrictive in nature, remain hard; allowing preemptions do help. The iterative application of the collapsing of dependency (preemption) graphs was intensively used for sequencing stages. It might be possible to extend this approach for scheduling problems with non-acyclic graphs by exploiting more general structures than the collapsing. For example, instead of a single edge between two neighboring elements in a collapsing, two or more edges might be allowed, which would require more sophisticated sequencing tools for incorporating two or more jobs on a single iteration.

We have exploited acyclic distributions obtained by linear programming which are preemptive. An optimal sequencing of these distributions is not a trivial task: an optimal distribution not necessarily yields an optimal schedule, as the total length of some job(s) in our distribution may exceed the optimal schedule makespan. We believe that a closer study of these distributions is possible. Firstly, there may exist conditions which provide distributions without such long jobs or guarantee their limited number. This might be natural to expect; roughly, an optimal distribution cannot allocate many jobs to inefficient processors. Secondly, it might be possible to convert an optimal distribution with long jobs to another optimal distribution without such jobs. A straightforward scheme would redistribute long jobs to more efficient processors, while some jobs distributed to these processors would move to other "sufficiently efficient" processors so that they would not turn to long jobs.

In the other direction, it seems to us that approximation algorithms for shop scheduling problems with their worst-case performance depending on the number of cycles in the machine dependency graphs should exist. Further study of the solvability/unsolvability conditions is also of a prior interest.

References

Gonzalez, T., & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM*, *23*, 665–679.

Gonzalez, T., & Sahni, S. (1978a). Preemptive scheduling of uniform processor systems. *Journal of the ACM*, *25*, 92–101.

Gonzalez, T., & Sahni, S. (1978b). Flow-Shop and Job-Shop schedules: complexity and approximations. *Operations Research*, *26*, 36–52.

Hall, N. G., Lee, T. E., & Posner, M. E. (2002). The complexity of cyclic shop scheduling problems. *Journal of Scheduling*, *5*, 307–327.

Hefetz, N., & Adiri, I. (1982). An efficient optimal algorithm for two-machines unit-time job-shop schedule-length problem. *Mathematics of Operations Research*, *7*, 354–360.

Jackson, J. R. (1955). *Scheduling a production line to minimize maximum tardiness* (Research Report No. 43). Management Science Research Project, University of California, Los Angeles.

Lawler, E. L., & Labetoulle, J. (1978). On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the ACM*, *25*, 612–619.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Algorithms for scheduling unrelated parallel machines. *Annals of Discrete Mathematics*, *1*, 343–362.

Lenstra, J. K., Shmoys, D. B., & Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, *46*, 259–271.

McNaughton (1959). Scheduling with deadlines and loss functions. *Management Science*, *6*, 1–12.

Potts, C. N. (1985). Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics*, *10*, 155–164.

Shachnai, H., Tamir, T., & Woeginger, G. (2002). Minimizing makespan and preemption costs on a system of uniform machines. In *Proc. 10th European symposium on algorithms*.

Shchepin, E., & Vakhania, N. (2002). Little-preemptive scheduling on unrelated processors. *Journal of Mathematical Modeling and Algorithms*, *1*, 43–56.

Shchepin, E., & Vakhania, N. (2005a). An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, *33*, 127–133.

Shchepin, E., & Vakhania, N. (2005b). New tight NP-hardness of preemptive multiprocessor and open-shop scheduling. In *Proceedings of 2nd multidisciplinary international conference on scheduling: Theory and applications MISTA 2005* (pp. 606–629).

Shchepin, E., & Vakhania, N. (2006). On machine dependency in shop scheduling. In *Proceedings of the 9th WSEAS international conference on applied mathematics* (pp. 323–331).