Proceedings of the 9th WSEAS International Conference on Applied Mathematics, Istanbul, Turkey, May 27-29, 2006 (pp323-331)

On Machine Dependency in Shop Scheduling

Evgeny Shchepin* Nodari Vakhania†

Abstract

One of the main restrictions in scheduling problems are the machine (resource) restrictions: each machine can perform at most one job at a time. We explore machine dependencies is shop scheduling problems representing them as graphs. Our study shows that the structure of machine dependency graphs is important in the complexity analysis of shop scheduling problems. We call acyclic a shop scheduling problem with the acyclic machine dependency graph. Here we first consider the periodic job-shop scheduling problem and present a linear-time algorithm for its subclass which machine dependency graphs are allowed to have only simple so-called parti-colored cycles. This result is tight as the trivial extension of the problem becomes NP-hard. We show that the acyclic preemptive open-shop problem with at most $m-2$ preemptions can be solved in linear time. This result is also tight as the same problem with at most $m-3$ preemptions is NP-hard. We also show that very simple acyclic shop scheduling problems are NP-hard; for example, any flow-shop with a single job with three operations and the rest of the jobs with a single non-zero length operation is NP-hard.

Key words: algorithm, shop scheduling, NP-hardness, machine dependency graph

1 Introduction

In this paper we explore machine dependencies is shop scheduling problems representing them as graphs. Our study shows that the structure of these graphs is important in the complexity analysis of shop scheduling problems. We call acyclic a shop scheduling problem with an acyclic machine dependency graph. We come across the polynomially solvable cases of several shop scheduling problems: periodic job-shop scheduling, preemptive open-shop scheduling and acyclic job-shop scheduling problems. We also complete known complexity results on shop scheduling showing that very simple acyclic shop scheduling problems are NP-hard.

In a scheduling problem *n jobs* need to be processed by *m machines*. 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*. A *machine dependency graph* (*dependency graph* for

*Steklov Math. Inst., 117966, Gubkina 8, Moscow, Russia E-mail: scepin@mian.ras.su. Partially supported by CONACYT grant 32728E and Russian Foundation of Basic Researches grant 99-01-00009

†Science Faculty, State University of Morelos, Av. Universidad 1001, Cuernavaca 62210, Morelos, Mexico. Inst. of Computational Math., Akuri 8, Tbilisi 93, Georgia. E-mail: nodari@servm.fc.uaem.mx

short) represents machine restrictions as follows: each node represents a unique machine and if there is an edge (M, P) labelled with job J, then job J has to be scheduled (or is scheduled) on both machines M and P. Depending on a particular scheduling problem, a machine dependency graph may represent a problem instance or already some distribution of jobs on machines (a schedule characteristic). For example, in shop scheduling problems each job J consists of a finite number of operations and for each operation there is given particular machine on which it has to be scheduled. Hence, there is a unique machine dependency graph representing each problem instance (whereas there are infinitely many instances represented by this graph). On the other hand, in preemptive multiprocessor scheduling problems, though a job is not initially split into operations, it can be split into different parts and these parts can be assigned to different machines. In this case the machine dependency graph will represent a particular distribution of jobs on machines and may vary from a schedule to a schedule. (We refer the reader to Shchepin & Vakhania [8] and [10] for more details.) Similarly as the structure of a precedence graph is important in the complexity analysis of many scheduling problems, the structure of a machine dependency graph is also important is this analysis. We shall call a shop scheduling problem *acyclic* if to each its instance an acyclic machine dependency graph corresponds. Here we focus on shop scheduling problems, and consider *periodic* and (traditional) non-periodic, or as we also call *finite* shop scheduling problems.

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 & Sahni [1]. It was recently shown that open-shop problem remains NP-hard if it is acyclic and at most $m-3$ preemptions are allowed Shchepin & Vakhania [11]. 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 [5]. The problem can be solved in time linear in the total number of operations with two machines and unit-length operations Hefetz & Adiri [4]. 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 Lenstra et al. [7] and Gonzalez & Sahni [2].

Periodic shop scheduling is easer. For example, while open-shop problem $O//C_{\max}$ is known to be NP-hard, it is quite straightforward to solve its periodic version $O, periodic//C_{\max}$. Scheduling periodic 2-machine job shop in which each job is allowed to have 3 operations is already NP-hard, but periodic job-shop problem can be solved in linear time if each job has at most two operations Hall et al. [3]. As we show here, a wider subclass of periodic job-shop problem can be solved in linear time. To each instance from this subclass, a machine dependency graph which may contain only special type of simple cycles corresponds; we call these cycles parti-colored and we call the corresponding scheduling problem *parti-cyclic job-shop*, abbreviated as $J, periodic/parti-cyclic/C_{\max}$. 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, is NP-hard. Preemptive open-shop scheduling problem is known to be solvable in polynomial (in the worst case in $O(n^4)$) time by the early algorithm of Gonzales & Sahni [1] imposing up to $O(n^2m)$ preemptions. We propose liner-time algorithm for the acyclic version $O/acyclic, pmtn/C_{max}$ of this problem imposing at most $m-2$ preemptions. As already noted, this problem becomes NP-hard if the number of preemptions is restricted to $m-3$. We also 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. Due to the space limitation, some of the proofs are omitted (they are available in the complete version).

2 Glossary of basic concepts and notations

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 $\mathcal{J} = \{J_1, ..., J_n\}$ and the set of machines $\mathcal{M} = \{M_1, \ldots, M_m\}$. 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 a 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 once scheduled on machine M_i. J_i^j is a *dummy operation* of job J^j on machine M_i if $p_i^j = 0$; further on we will use "operation" exclusively for a non-dummy operation.

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 a 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. 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.

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*. Since a machine at any moment can process at most one job, a schedule can be seen as a mapping from $\mathcal{M} \times \mathrm{R}^+$ to $\mathcal{J}$, or a graph of such a mapping, i.e., a subset of the product $\mathcal{J} \times \mathcal{M} \times \mathrm{R}^+$. For a given schedule σ, $(J, M, t) \in \sigma$ signifies that in σ job J is processed by machine M at the moment t. For a given pair $(J, M) \in \mathcal{J} \times \mathcal{M}$, $\sigma(J, M)$ is the set of all time moments at which machine M processes job J, i.e., $\sigma(J, M) = \{t \in \mathrm{R}^+ \mid (J, M, t) \in \sigma\}$. As already noted, for each $M \in \mathcal{M}$ and $t \in \mathrm{R}^+$, the job set $\sigma(M, t)$ defined as $\sigma(M, t) = \{J \in \mathcal{J} \mid (J, M, t) \in \sigma\}$ contains at most one element, i.e. σ is *sequential on machine* M. Likewise, σ is *sequential on job* J if J is processed by at most one machine at any time moment t (different operations of J do not overlap in time on different machines), i.e., the machine set $\sigma(J, t) = \{M \in \mathcal{M} \mid (J, M, t) \in \sigma\}$ contains at most one element. A *sequential schedule* is a one which is sequential on all jobs (and all machines).

We consider two sorts of schedules, *periodic* (infinite) schedules and *non-periodic* (finite) schedules. A finite schedule σ is a subset of $\mathcal{J} \times \mathcal{M} \times [0, T)$, for some $T \in \mathrm{R}^+$. The minimal such T for σ is called the *makespan* of σ and is denoted by $\|\sigma\|$. 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 $\|\sigma\|$.

If σ is sequential on job J, different J-components do not intersect in time. Therefore, 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 $q \leq p'$ and there is no other J-component, scheduled within the interval $[q, p')$, equivalently, there is

no J-switching point between q and p'. We will say that σ is *continuous on job* J if it is sequential on J and 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, σ is *continuous on machine* M, if it is sequential on M and 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 a one which is continuous on all machines and jobs.

A (finite or periodic) schedule σ is called *feasible* for a job-shop $\mathcal{J}, \mathcal{M}$ if it satisfies the following conditions: (1) It is sequential (on jobs of $\mathcal{J}$ and machines of $\mathcal{M}$); (2) $|\sigma(J^j, M_i)| = |J_i^j|$ for all $i \leq m$ and $j \leq n$; (3) $J_i^j \rightarrow J_k^j$ implies that the continuation of any (J^j, M_i)-component of σ is a (J^j, M_k)-component; (4) For any job J and any machine M the length of the (J, M)-component is $|\sigma(J, M)|$. (2) provides that operation J_i^j is completely processed. (3) provides that the precedence relations between the operations of each job are respected (this condition also provides that σ is sequential on each job). (4) provides that operations are processed without any preemptions.

A feasible schedule σ with the minimal (makespan/period) $\|\sigma\|$ is called *optimal*. The following lemma holds because any feasible schedule is sequential on both, machines and jobs:

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

For a positive number x, the *x-shifting* of a schedule σ is a schedule σ^x, such that $\sigma^x = \{(J, M, t) \mid (J, M, t - x) \in \sigma\}$. For a negative x, σ^x is defined similarly with the additional condition that the starting time of an earliest scheduled job in σ is to be no less than $|x|$.

We will say that a schedule σ' is obtained from another schedule σ by *inserting* job J^* on machine M_0 into the time interval $[p, q)$ if for any J and M:
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 $t \in [p, q)$.

3 Machine Dependency Graph

Recall that a (machine) dependency graph G represents machine dependencies (nodes representing the machines and edges representing the jobs shared by the corresponding couple of machines). For an instance of the job-shop scheduling problem $\mathcal{J}, \mathcal{M}$, there is an edge (M_l, M_k) in this graph labelled by job J^j, iff $J_l^j \rightarrow J_k^j$. The *J-component* of G, $G[J]$ is its subgraph formed by the union of all edges of G, labelled 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 (i.e. associated with an edge of G).

A shop scheduling problem is said to be *acyclic* if the dependency graph of its any instance is acyclic. We will see now that operation order does not influence on the acyclicity of G. More precisely, 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 in all jobs is also acyclic. Equivalently, any job-shop instance, obtained from some acyclic open-shop instance by imposing some operation order in 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.

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

graph.

A shop instance is said to be *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.*

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. We shall apply the following decomposition of an acyclic dependency graph G. 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 $G_0, G_1, ..., G_k$ a *collapsing* of G and the corresponding sequence of jobs $J^0, \ldots, J^{k-1}$ a *collapsing sequence of jobs*. The brutal 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 & Vakhania [9]).

4 Scheduling Periodic job-shop and preemptive open-shop

Let us call a simple cycle in a dependency graph *parti-colored* if all its edges have different labels. 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*. We abbreviate the periodic parti-cyclic job-shop problem as $J, periodic/parti-cyclic/C_{\max}$.

An extension $\mathcal{J}', \mathcal{M}'$ of $\mathcal{J}, \mathcal{M}$ is said to be its *simple extension* with job I if:
1. $\mathcal{J}' = \mathcal{J} \cup \{I\}$;
2. there is no operation on any machine of $\mathcal{M}' \setminus \mathcal{M}$ of any job from $\mathcal{J}$;
3. job I has one operation on each machine of $\mathcal{M}' \setminus \mathcal{M}$ and it has one operation on exactly one machine of $\mathcal{M}$. The following lemma will be iteratively applied in our basic algorithm.

Lemma 2 *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 the simple extension $\mathcal{J}', \mathcal{M}'$ of $\mathcal{J}, \mathcal{M}$ with job I.*

For an enumeration of jobs $J^1, J^2, \ldots J^k$, $k \leq n$ 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 $O(nm)$. We can prove the following result using the collapsing sequence of jobs of G:

Lemma 3 *A continuous finite schedule σ for an instance of acyclic job shop $\mathcal{J}, \mathcal{M}$ can be constructed in time $O(nm)$.*

The following result can be proved by applying the periodic extension of σ:

Lemma 4 *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 $O(nm)$.*

Now we already have an $O(nm)$ algorithm for the acyclic periodic job-shop $J, periodic/acyclic/C_{\max}$. Indeed, we construct a finite continuous schedule σ for all non-elementary jobs of our job-shop by Lemma 3. Then we insert all elementary jobs and obtain another continuous schedule. Finally, we periodically extend the latter continuous schedule by Lemma 4:

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

Next, we extend this result for the periodic parti-cyclic job-shop using the following lemmas:

Lemma 5 *A dependency graph G is continuously solvable in linear-time if it is a simple parti-colored cycle.*

Lemma 6 *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.*

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

We have earlier mentioned that $O/acyclic, pmtn(m-3)/C_{max}$ is NP-hard. We will see now that the version of the same problem with 1 less preemption, $O/acyclic, pmtn(m-2)/C_{max}$ can be efficiently solved.

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 non-elementary job J^j is continuous in σ^P. Consider a switching point τ in σ^P, 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. τ is a switching point on both machines M_i and M_k.

To obtain our destiny schedule σ, we rotate a finite segment of σ^P. In particular, let σ be the $(-\tau)$-shifting of the finite schedule $\sigma^P \bigcap \mathcal{J} \times \mathcal{M} \times [\tau, \tau + \|\sigma^P\|)$. 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. (There will occur a preemption on machine M in σ if τ is not a switching point for the job, which operation is processed in σ^P by machine M at moment τ: the part of this operation scheduled after time τ in σ^P will be scheduled the first 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.) Hence, σ will have at most $m-2$ preemptions and since the makespan of σ is T, it is optimal (Lemma 1).$\diamondsuit$

5 NP-hardness of simple acyclic shop problems

First we show that trivial extensions of $J, periodic/parti-cyclic/C_{\max}$ become NP-hard. Recall 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:

Theorem 4 *Let $FS(3)$ be a flow-shop instance with three machines M_1, M_2 and M_3, and two jobs J^1 and J^2. $p_1^1 = p_1^2 = 2$, 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. $FS(3)$ 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:
$\sigma(J^1, M_1) = [0, 2)$ and $\sigma(J^2, M_1) = [2, 4)$;
$\sigma(J^1, M_2) = [2, 3)$ and $\sigma(J^2, M_2) = [4, 5)$;
$\sigma(J^1, M_3) = [3, 4)$ and $\sigma(J^2, M_3) = [5, 6)$;
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).

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.$\diamondsuit$

Lemma 7 *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 $C \leq \sum_i x_i$ be an arbitrary instance of KNAPSACK. In our scheduling instance we have 3 machines M_1, M_2 and M_2 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 $p_3^2 = p_1^0 + p_2^0$; finally, k jobs $J^3, \ldots, J^{k+2}$ with $p_2^i = x_i$, $i = 3, ..., k + 2$ 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 $\sum\limits_{i \in X'} x_i = C$.$\diamondsuit$

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 8 *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 $|||[\sigma]|| \leq ||\sigma||$. 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 $|||[\sigma]||| \leq ||\sigma||$, $\|\sigma'\| + 1 \leq \|\sigma\|$. 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 $\|\sigma'\| + 1 \leq \|\sigma\|$, $\|\sigma^k\| < \|\sigma\|$ and we came to a contradiction.$\diamondsuit$

The next result immediately follows from Lemmas 7 and 8:

Theorem 5 *Any flow-shop problem with at least 1 job with at least 3 operations is (finitely) unsolvable.*

We complete this section by giving 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):

Theorem 6 *Consider the following flow-shop instance $FS(7)$ with three machines M_1, M_2 and M_3 and seven jobs J^i, $i = 1, 2, \ldots, 7$. The processing order of each job coincides with the machine numbering and all operations of all these jobs have length 1. The problem of construction of an optimal finite schedule for any elementary extension of $FS(7)$ with short jobs is NP-hard.*

6 Further research

We believe that approximation algorithms for $J//C_{max}$ can be built with their worst-case performance depending on the total number of cycles in the machine dependency graphs. A further investigation of the solvability conditions is of the interest. For example, can there be found sufficient and necessary conditions for (periodic and finite) solvability of dependency graphs in job-shop?

References

[1] Gonzalez T. and S. Sahni, "Open Shop Scheduling to Minimize Finish time", *Journal of the ACM* 23, 665-679 (1976)

[2] Gonzalez T. and S. Sahni, "Flow-Shop and Job-Shop schedules: complexity and approximations", *Oper. Research* 26, 36-52 (1978)

[3] Hall N.G., T.E. Lee and M.E. Posner, "The complexity of cyclic shop scheduling problems", *Journal of Schedulung* 5, 307-327 (2002)

[4] Hefetz N. and I. Adiri. An efficient optimal algorithm for two-machines unit-time job-shop schedule-length problem. *Math. Oper. Res.*, 7, 354-360 (1982)

[5] Jackson J. R., "Scheduling a production line to minimize maximum tardiness", *Research Report* 43, Management Science Research Project, University of California, Los Angeles (1955)

[6] Lenstra J.K., Shmoys D. B. and Tardos E. "Approximation algorithms for scheduling unrelated parallel machines" *Mathematical Programming*, 46, 259-271 (1990)

[7] Lenstra J.K., Rinnooy Kan A. H. G., and Brucker P. "Algorithms for scheduling unrelated parallel machines" *Ann. Discr. Math.*, 1, 343-362 (1977)

[8] Shchepin E. and Vakhania N., "Task distributions on multiprocessor systems", In: *Lecture Notes in Computer Science (IFIP TCS)* 1872, p.112-125, Springer, New York (2000)

[9] Shchepin E. and N. Vakhania, "Little-preemptive scheduling on unrelated processors", *Journal of Mathematical Modelling and Algorithms* 1, 43-56 (2002)

[10] Shchepin E. and N. Vakhania. "An optimal rounding gives a better approximation for scheduling unrelated machines". *Operations Research Letters* 33, p.127-133 (2005)

[11] Shchepin E. and N. Vakhania. "New tight NP-hardness of preemptive multiprocessor and open-shop scheduling". *Proceedings of 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications MISTA 2005*, p. 606-629 (2005)