Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 178 (2007) 677–685

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

On no-wait and no-idle flow shops with makespan criterion

Pawel Jan Kalczynski, Jerzy Kamburowski [*]

Department of Information, Operations and Technology Management, College of Business Administration, MS #103, The University of Toledo, Toledo, OH 43606-3390, USA

Received 26 January 2005; accepted 25 January 2006
Available online 17 April 2006

Abstract

The paper deals with the NP-hard problems of minimizing the makespan in m-machine no-wait and no-idle permutation flow shops. We identify networks whose longest path lengths represent the makespans. These networks reveal the *duality* between the two problems, and show graphical explanations of the fact that under no-wait and no-idle conditions the makespan can be a decreasing function of some job processing times. Moreover, they also lead to a natural reduction of the no-wait flow shop problem to the traveling salesman problem, some lower bounds on the shortest makespan, and new efficiently solvable special cases.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Scheduling; Flow shops; Makespan; No-wait condition; No-idle condition

1. Introduction

A set of n jobs available at time zero has to be processed in a shop with m machines $M_1, M_2, \ldots, M_m$. Each job is processed first on M_1, next on M_2, and so on, and lastly on M_m. No machine can process more than one job at a time, no job preemption is allowed, all setup times are included into the job processing times, and there is unlimited storage between the machines. The problem, commonly referred to as $Fm||C_{\max}$, is to determine a schedule that minimizes the completion time of the last job on M_m, also known as the makespan. When the schedule must have the same job sequence on every machine, the corresponding permutation problem is denoted by $Fm|\text{prmu}|C_{\max}$.

The paper deals with minimizing the makespan in permutation flow shops under no-wait and no-idle conditions. When each job must be processed from the start to finish without any interruption between the machines, the problem $Fm|\text{no-wait}|C_{\max}$ is defined. The no-wait condition secures that any no-wait schedule must be a permutation schedule, and thus $Fm|\text{no-wait}|C_{\max}$ and $Fm|\text{prmu, no-wait}|C_{\max}$ are the same [9]. On the other hand, the no-idle permutation problem $Fm|\text{prmu, no-idle}|C_{\max}$ is formulated when each machine

[*] Corresponding author. Tel.: +1 419 530 2258; fax: +1 419 530 7744.
E-mail address: Jerzy.Kamburowski@utoledo.edu (J. Kamburowski).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.01.036

must process the jobs without any idle time. As in the case of $Fm||C_{\max}$, there are instances of $Fm|\text{no-idle}|C_{\max}$ with $m \geqslant 4$ for which the restriction to permutation schedules can be costly [11].

Piehler [14] was the first to show that $Fm|\text{no-wait}|C_{\max}$ reduces to an instance of the traveling salesman problem (TSP); see also [16,21]. When $m = 2$, this instance becomes solvable by the algorithm of Gilmore–Gomory [7]; see [16]. Adiri and Pohoryles [3] observed that $F2|\text{prmu, no-idle}|C_{\max}$ and $F2|\text{prmu}|C_{\max}$ are equivalent, and thus both problems can be solved by Johnson's [10] algorithm. Röck [17] and Baptiste and Lee [5] proved that $F3|\text{no-wait}|C_{\max}$ and $F3|\text{prmu, no-idle}|C_{\max}$ are strongly NP-hard. Numerous practical examples of job scheduling in no-wait environments are summarized in the review paper of Hall and Sriskandarajah [9]. Several applications under the no-idle condition are reported by Saadani et al. [18], and in some of their references.

It is well known that the makespan of a given job sequence in $Fm|\text{prmu}|C_{\max}$ can be represented as the length of a critical (longest) path in a network; see e.g. [15, p. 131]. To the best of our knowledge, similar network representations are unknown in the case of $Fm|\text{no-wait}|C_{\max}$ and $Fm|\text{prmu, no-idle}|C_{\max}$.

In this paper we present the two *missed* network representations. They allowed us to reveal a *duality* relationship that exists between $Fm|\text{no-wait}|C_{\max}$ and $Fm|\text{prmu, no-idle}|C_{\max}$, and explain clearly an observed *anomaly* in flow shop scheduling under the no-wait and no-idle conditions. This virtual anomaly is manifested in the possible reduction in the makespan as a consequence of prolonging the processing of a job on an intermediate machine; see [1,12]. Our network representations also lead to a natural reduction of $Fm|\text{no-wait}|C_{\max}$ to TSP, some lower bounds on the shortest makespans, and new efficiently solvable special cases.

2. Two-machine flow shops

Suppose a set of jobs, $\{1,2,\ldots,n\}$, available at time zero has to be processed in a flow shop with two machines A and B in series. Let a_k and b_k be the processing times of job k on A and B, respectively, and let $C_{\max}(\pi;A,B)$, $C_{\max}(\pi;\text{no-wait},A,B)$, and $C_{\max}(\pi;\text{no-idle},A,B)$ be the makespans of a job sequence $\pi = (\pi(1),\pi(2),\ldots,\pi(n))$ in $F2|\text{prmu}|C_{\max}$, $F2|\text{no-wait}|C_{\max}$, and $F2|\text{prmu, no-idle}|C_{\max}$.

It was observed in [3], that for every job sequence π,

$$C_{\max}(\pi;A,B) = C_{\max}(\pi;\text{no-idle},A,B) = \max_{k=1,2,\ldots,n}\left[\sum_{j=1}^{k} a_{\pi(j)} + \sum_{j=k}^{n} b_{\pi(j)}\right].$$

Thus, both $F2|\text{prmu}|C_{\max}$ and $F2|\text{prmu, no-idle}|C_{\max}$ can be solved in $O(n\log n)$ time by Johnson's algorithm.

Since

$$C_{\max}(\pi;\text{no-wait},A,B) = a_{\pi(1)} + \sum_{j=1}^{n-1}\max(a_{\pi(j+1)},b_{\pi(j)}) + b_{\pi(n)},$$

by adding an artificial job 0 with $a_0 = b_0 = 0$, $F2|\text{no-wait}|C_{\max}$ reduces to TSP on the $(n+1)\times(n+1)$ distance matrix defined by

$$D_{jk} = \max(a_k,b_j) \quad \text{for } 0 \leqslant j,k \leqslant n.$$

If $0 \to j_1 \to j_2 \to \cdots \to j_n \to 0$ is the shortest tour, the job sequence $(j_1,j_2,\ldots,j_n)$ is optimal. The TSP on the above matrix can be solved in $O(n\log n)$ time by the GG algorithm of Gilmore and Gomory [7]; see also [20].

3. *m*-Machine flow shops

3.1. Network representations of the makespans

Let p_{ij} be the processing time of job j on machine M_i for $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$. Let $C_{\max}(\pi;\text{no-wait})$ and $C_{\max}(\pi;\text{no-idle})$ denote the makespans of a job sequence $\pi = (\pi(1),\pi(2),\ldots,\pi(n))$ in $Fm|\text{no-wait}|C_{\max}$ and $Fm|\text{prmu, no-idle}|C_{\max}$.

The following result shows network representations of $C_{\max}(\pi;\text{no-wait})$ and $C_{\max}(\pi;\text{no-idle})$.

Theorem 1. *The makespans* $C_{\max}(\pi;\text{no-wait})$ *and* $C_{\max}(\pi;\text{no-idle})$, *where for simplicity* $\pi = (1,2,\ldots,n)$, *are the lengths of critical paths in the networks shown in Figs.* 1*a and b*, *and* 2*a and b*, *respectively.*

Proof. The no-wait and no-idle conditions can be modeled by a technique, developed by Elmaghraby and Kamburowski [6], for representing generalized precedence relations in activity networks. Using this technique, $C_{\max}(\pi;\text{no-wait})$ is the critical path length in the network of Fig. 1a. Its solid arcs represent the processing of the jobs, while its dashed arcs are dummy. The network resembles an *activity-on-arc* type CPM network with the only difference that two anti-parallel arcs weighted by p_{ij} and $-p_{ij}$ model the processing of any intermediate job j on every intermediate machine M_i. The existence of such arcs secures the no-wait condition. (There is no need to add the arcs with negative weights for the jobs processed on M_1 and M_m, as well as for the first and last jobs; there exists a critical path that does not pass through such arcs.) The makespan $C_{\max}(\pi;\text{no-wait})$ is also the critical path length in the *activity-on-node* type network of Fig. 1b; the nodes are numbered by the corresponding processing times. In order to verify this, it suffices to check that the networks in Figs. 1a and b are equivalent because there is a one-to-one correspondence between their simple paths, and the related paths have the same length. Similarly, $C_{\max}(\pi;\text{no-idle})$ is the critical path length in the network of Fig. 2a, and the networks in Figs. 2a and b are equivalent. □

Let $C_{\max}((j,k))$ be the makespan of a two-job subsequence (j,k) in $Fm|\text{prmu}|C_{\max}$, and let $C_{\max}(\pi;M_i,M_{i+1})$ be the makespan of π in the flow shop with two machines M_i and M_{i+1} in series. From Theorem 1 one obtains the following simple representations of $C_{\max}(\pi;\text{no-wait})$ and $C_{\max}(\pi;\text{no-idle})$.

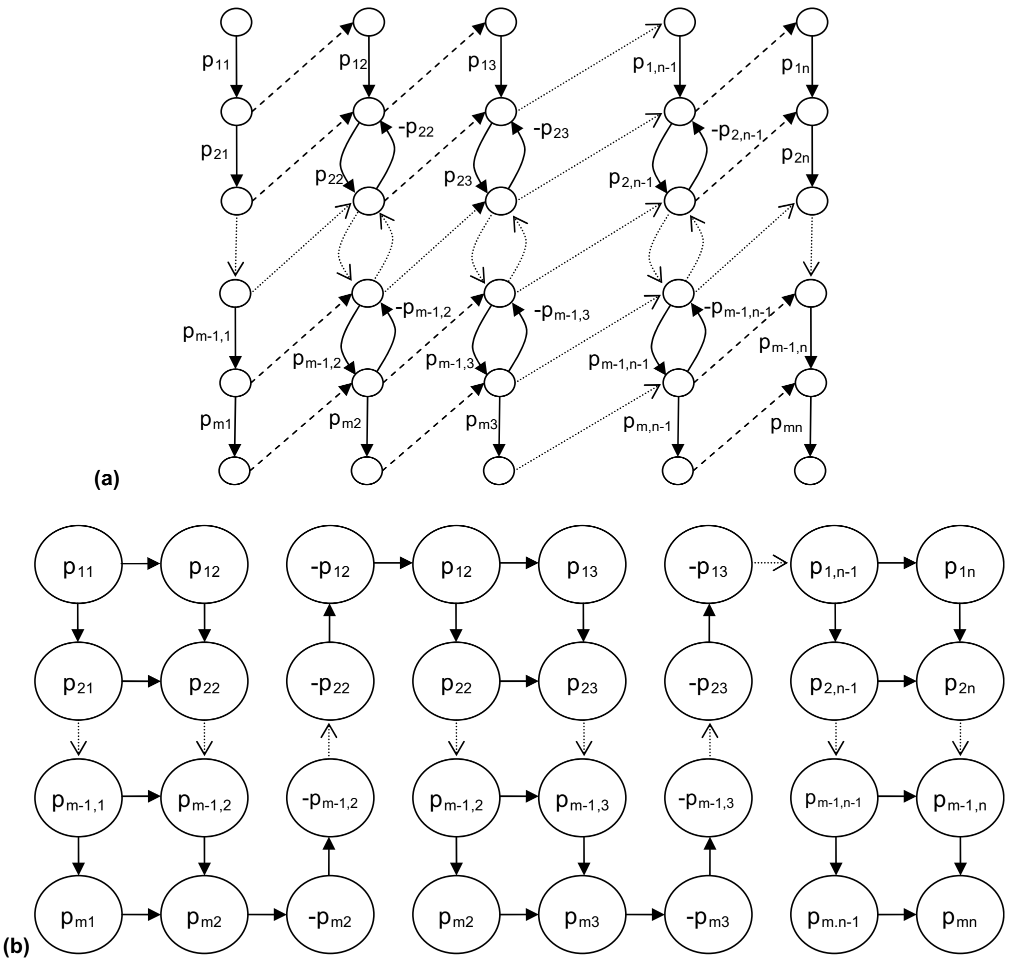

Fig. 1. Equivalent networks for computing the makespan $C_{\max}(\pi;\text{no-wait})$ for $\pi = (1,2,\ldots,n)$.

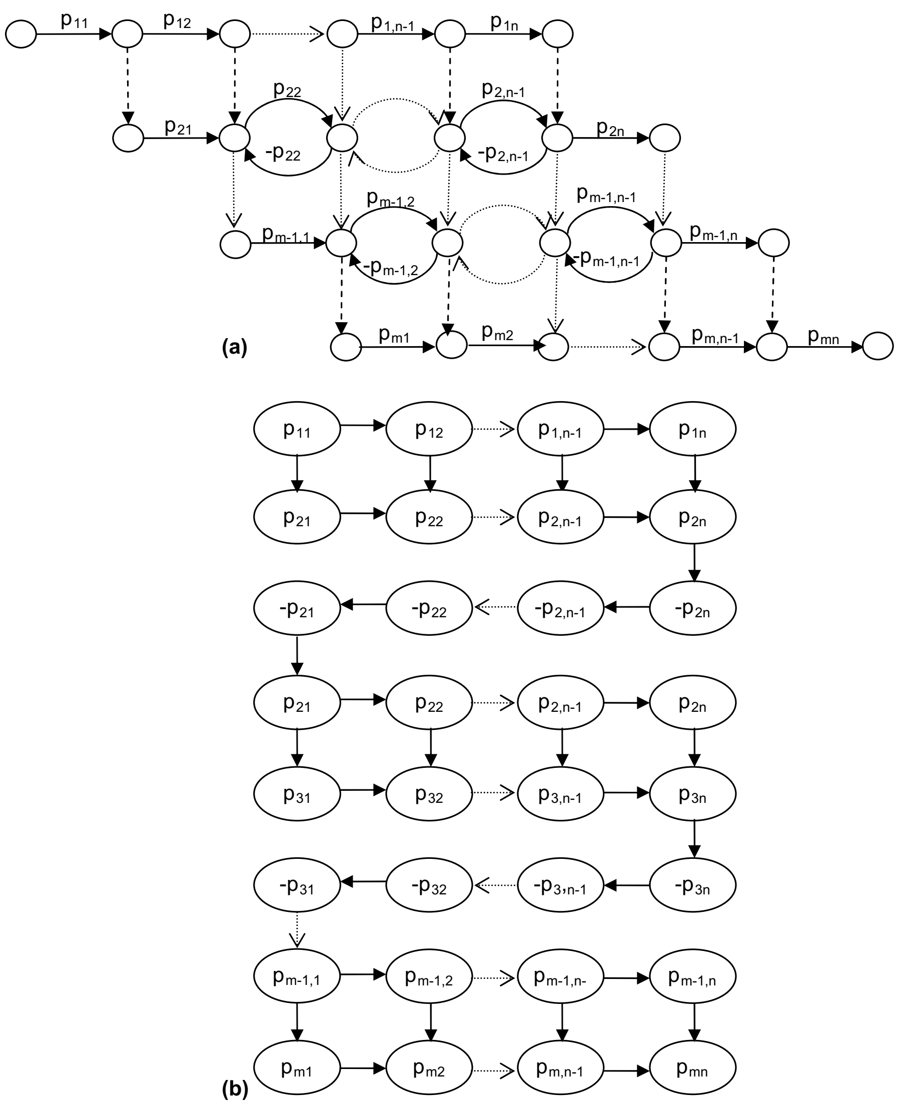

Fig. 2. Equivalent networks for computing the makespan $C_{\max}(\pi;\text{no-idle})$ for $\pi = (1,2,\ldots,n)$.

Corollary 1. *For every job sequence* π,

(i) $C_{\max}(\pi;\textit{no-wait}) = \sum_{j=1}^{n-1} C_{\max}((\pi(j),\pi(j+1))) - \sum_{i=1}^{m}\sum_{j=2}^{n-1} p_{i,\pi(j)}$;

(ii) [12] $C_{\max}(\pi;\textit{no-idle}) = \sum_{i=1}^{m-1} C_{\max}(\pi;M_i,M_{i+1}) - \sum_{i=2}^{m-1}\sum_{j=1}^{n} p_{ij}$.

Note that the makespan $C_{\max}(\pi;\text{no-wait})$ has been computed so far by rather complex recursive formulas developed by Reddi and Ramamoorthy [16].

It could be added here that when the arcs with negative weights are deleted from the network of Fig. 1a, the critical path length in the resulting *activity-on-arc* CPM network is $C_{\max}(\pi;\text{block})$, that is, the makespan of π computed for the problem, referred to as $Fm|\text{block}|C_{\max}$, under the so-called blocking condition; see e.g. [15, p. 143]. On the other hand, when the arcs with negative weights are deleted from the network of Fig. 2a, the longest path length in the resulting network does not seem to have any known interpretation. This length equals the makespan of π under the condition that a job completed on a machine must remain on it as long as this machine is idle and waits for other jobs. Since all machines are kept *busy*, we propose to call the corresponding problem, which so far is only a theoretical one, $Fm|\text{prmu,busy}|C_{\max}$.

From the networks of Figs. 1 and 2, as well as Corollary 1, it is evident that a certain *duality* relationship exists between $Fm|\text{no-wait}|C_{\max}$ and $Fm|\text{prmu, no-idle}|C_{\max}$. (A similar duality relationship exists between $Fm|\text{block}|C_{\max}$ and $Fm|\text{prmu,busy}|C_{\max}$.) Clearly, for a given job sequence $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, define n artificial machines in series $\overline{M}_{\pi(1)}, \overline{M}_{\pi(2)}, \ldots, \overline{M}_{\pi(n)}$. Define also a sequence $J = (J_1, J_2, \ldots, J_m)$ of m artificial jobs, and assume that p_{ij} is the processing time of job J_i on machine $\overline{M}_{\pi(j)}$. Then $C_{\max}(\pi;\text{no-wait}) = C_{\max}(J;\text{no-idle}, \overline{M}_{\pi(1)} - \overline{M}_{\pi(n)})$, and $C_{\max}(\pi;\text{no-idle}) = C_{\max}(J;\text{no-wait}, \overline{M}_{\pi(1)} - \overline{M}_{\pi(n)})$, where $C_{\max}(J;\text{no-idle}, \overline{M}_{\pi(1)} - \overline{M}_{\pi(n)})$ and $C_{\max}(J;\text{no-wait}, \overline{M}_{\pi(1)} - \overline{M}_{\pi(n)})$ are the makespans of J computed under the no-idle and no-wait conditions on the ordered machines $\overline{M}_{\pi(1)}, \overline{M}_{\pi(2)}, \ldots, \overline{M}_{\pi(n)}$.

3.2. Reductions of $Fm|\text{no-wait}|C_{max}$ to TSP

From Corollary 1(i) we have

$$C_{\max}(\pi;\text{no-wait}) = \sum_{i=1}^{m} p_{i,\pi(1)} + \sum_{j=1}^{n-1} C_{\max}((\pi(j), \pi(j+1))) + \sum_{i=1}^{m} p_{i,\pi(n)} - \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}.$$

Since $\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}$ is a constant, by adding an artificial job 0 with $p_{i0} = 0$ for $i = 1, 2, \ldots, m$, $Fm|\text{no-wait}|C_{\max}$ reduces to TSP on the $(n+1)\times(n+1)$ distance matrix defined by

$$d_{jk} = C_{\max}((j,k)) = C_{\max}((j,k);\text{no-wait}).$$

The authors of [14,16,21] found an equivalent reduction to TSP with the distance matrix

$$D_{jk} = \max_{1 \leqslant i \leqslant m}\left[\sum_{h=1}^{i} p_{hj} - \sum_{h=1}^{i-1} p_{hk}\right],$$

which is commonly presented in the scheduling literature; see e.g. [1,9,15].

3.3. Anomalies in no-wait and no-idle scheduling

Given a job sequence π and its makespan, the corresponding schedules are defined by the job start times. Therefore, let s_{ij} denote the start time of job j on machine M_i. Although the makespan representations of Figs. 1a and 2a have cycles, all s_{ij} can be found in $O(mn)$ time, which is demonstrated below for $\pi = (1, 2, \ldots, n)$ and $Fm|\text{no-wait}|C_{\max}$.

Algorithm 1 (*Finding a no-wait schedule for a given job sequence*). Let $s_{i0} = p_{i0} = s_{0j} = p_{0j} = 0$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, and

```
For $j = 1, 2, \ldots, n$ do
    set $s_{ij} = \max(s_{i-1,j} + p_{i-1,j}, s_{i+1,j-1})$ for $i = 1, 2, \ldots, m-1$, and $s_{mj} = \max(s_{m-1,j} + p_{m-1,j}, s_{m,j-1} + p_{m,j-1})$;
    set $s_{ij} =: \max(s_{ij}, s_{i+1,j} - p_{ij})$ for $i = m-1, m-2, \ldots, 1$;
Next $j$.
```

Since the problem $Fm|\text{prmu, no-idle}|C_{\max}$ was found to be dual to $Fm|\text{no-wait}|C_{\max}$, the computations of s_{ij} for a no-idle permutation schedule are left as an exercise.

The existence of the arcs with negative weights in the networks of Figs. 1a and 2a explains clearly the known fact that the makespan can be reduced by *slowing down* the processing of some jobs [1,12].

Example 1. Consider a three-machine no-wait flow shop with three jobs defined by $p_{11} = 1$, $p_{21} = 1$, $p_{31} = 5$, $p_{12} = 2$, $p_{22} = 2$, $p_{32} = 2$, $p_{13} = 3$, $p_{23} = 3$, and $p_{33} = 1$. The makespan of $\pi = (1, 2, 3)$ is $C_{\max}(\pi;$

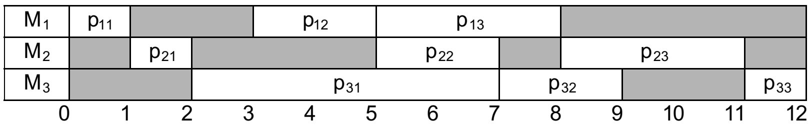

Fig. 3. Gantt's charts of the no-wait schedule for Example 1.

no-wait) $= p_{11} + p_{21} + p_{31} - p_{22} + p_{13} + p_{23} + p_{33} = 12$; see Fig. 3. If p_{22} increases from 2 to 3, the makespan decreases from 12 to 11.

Example 2. Consider a three-machine no-idle permutation flow shop with three jobs defined by $p_{11} = 1$, $p_{21} = 2$, $p_{31} = 4$, $p_{12} = 3$, $p_{22} = 2$, $p_{32} = 2$, $p_{13} = 3$, $p_{23} = 2$, and $p_{33} = 1$. The makespan of $\pi = (1,2,3)$ is $C_{\max}(\pi;\text{no-idle}) = p_{11} + p_{12} + p_{13} - p_{22} + p_{31} + p_{32} + p_{33} = 12$; see Fig. 4. If p_{22} increases from 2 to 3, the makespan decreases from 12 to 11.

The anomaly in no-wait scheduling was used by Abadi et al. [1] in a heuristic that can be regarded as that for solving $Fm|\text{block}|C_{\max}$. Since $Fm|\text{no-wait}|C_{\max}$ reduces to TSP, they proposed to find a no-wait schedule by a *good* TSP heuristic first. Then attempts are made to shorten this schedule by some increases in the job processing times. Therefore, assume that the times p_{ij} are allowed to be increased, and let $C_{\max}(\pi;\text{no-wait}, x_{ij})$ be the makespan of π with the increased processing times x_{ij}. It was proven in [1] that for any π, there exist $x_{ij} \geqslant p_{ij}$ such that $C_{\max}(\pi;\text{no-wait}, x_{ij}) = C_{\max}(\pi;\text{block})$. The authors of [1] find the optimal x_{ij} by solving an instance of the minimum cost flow problem defined on a directed network with O(mn) nodes and O(mn) arcs. Fig. 1a reveals that the optimal x_{ij} can be determined by the following O(mn) algorithm presented below for $\pi = (1,2,\ldots,n)$. It finds the earliest and latest start times e_{ij} and ℓ_{ij} of job j on machine M_i under the blocking condition, and use them to define x_{ij}.

Algorithm 2 (*Finding a no-wait schedule with slowing down*)

```
Step 1. (Finding the earliest blocking schedule)
        Let $e_{i0} = p_{i0} = e_{0j} = p_{0j} = 0$ for $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$, and
    For $j = 1,2,\ldots,n$ do
        set $e_{ij} = \max(e_{i-1,j} + p_{i-1,j},\ e_{i+1,j-1})$ for $i = 1,2,\ldots,m-1$, and $e_{mj} = \max(e_{m-1,j} + p_{m-1,j},\ e_{m,j-1} + p_{m,j-1})$;
    Next $j$.
Step 2. (Finding the latest blocking schedule)
        Let $\ell_{i,n+1} = \ell_{0j} = e_{mn} + p_{mn}$ for $i = 0,1,\ldots,m$ and $j = 1,2,\ldots,n$, and
    For $j = n, n-1,\ldots,1$ do
        set $\ell_{mj} = \ell_{m,j+1} - p_{mj}$ and $\ell_{ij} = \min(\ell_{i+1,j} - p_{ij}, \ell_{i-1,j+1})$ for $i = m-1, m-2,\ldots,1$;
    Next $j$.
Step 3. For $j = 1,2,\ldots,n$, let $M_{i_j}$ be a machine such that $e_{i_j} = \ell_{i_j}$, and define
    $x_{mj} = p_{mj}$, $x_{ij} = \ell_{i+1,j} - \ell_{ij}$ if $i < i_j$, and $x_{ij} = e_{i+1,j} - e_{ij}$ if $i \geqslant i_j$ and $i \neq m$.
```

Theorem 2. *Given* $\pi = (1,2,\ldots,n)$, *Algorithm 3 finds the increased processing times* x_{ij} *such that* $C_{\max}(\pi;\textit{no-wait}, x_{ij}) = C_{\max}(\pi;\textit{block})$ *and* $\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$ *is minimum.*

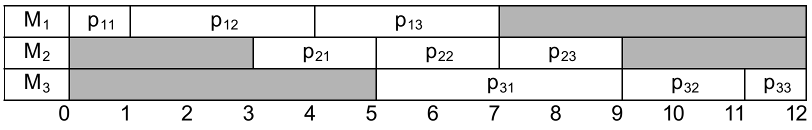

Fig. 4. Gantt's charts of the no-idle schedule for Example 2.

Proof. To secure the makespan $C_{\max}(\pi;\text{block}) = e_{mn} + p_{mn}$ and the minimum $\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}$, it is necessary to process every job j in the interval $[\ell_{1j}, e_{mj} + p_{mj}]$. The definition of x_{ij} in Step 3 leads to $C_{\max}(\pi;\text{no-wait}, x_{ij}) = C_{\max}(\pi;\text{block})$ and $\sum_{i=1}^{m}x_{ij} = e_{mj} + p_{mj} - \ell_{1j}$ for $j = 1, 2, \ldots, n$. □

Reconsider the instance of Example 1. Algorithm 2 finds $x_{22} = 3$ and leaves the remaining times unchanged. This leads to $C_{\max}(\pi;\text{no-wait}, x_{ij}) = 11$, and we also have $C_{\max}(\pi;\text{block}) = 11$.

A similar *slowing-down* algorithm can be proposed for finding $x_{ij} \geqslant p_{ij}$ such that $C_{\max}(\pi;\text{no-idle}, x_{ij}) = C_{\max}(\pi;\text{busy})$ and $\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}$ is minimum. When applied on the instance of Example 2, it would find $x_{22} = 3$ to yield $C_{\max}(\pi;\text{no-idle}, x_{ij}) = 11$, and match $C_{\max}(\pi;\text{busy}) = 11$.

3.4. Lower bounds and efficiently solvable special cases

In the first part of this subsection we consider $Fm|\text{no-wait}|C_{\max}$. The makespan representation shown in Corollary 1(i) allows us to define a polynomially-found sequence π_{hi} and the corresponding lower bound L_{hi} on the shortest no-wait makespan for every pair of machines (M_h, M_i) such that $h < i$.

Algorithm 3 (*Finding π_{hi} and L_{hi}*)

Step 1. Define two artificial machines A and B in series with the times $a_k = \sum_{g=h}^{i-1}p_{gk}$ and $b_k = \sum_{g=h+1}^{i}p_{gk}$ for $k = 1, 2, \ldots, n$.

Step 2. For every pair of jobs (r, s), define a dummy job 0 with $a_0 = \sum_{g=h}^{i-1}p_{gs}$ and $b_0 = \sum_{g=h+1}^{i}p_{gr}$, apply the GG algorithm on the $(n-1)\times(n-1)$ distance matrix $D_{jk} = \max(a_k, b_j)$, where $j, k \in \{0, 1, \ldots, n\}$-$\{r, s\}$, to find the $(n-2)$-element sequence $\pi(r, s)$, and compute

$$C(r,s) = \sum_{g=1}^{h-1}p_{gr} + C_{\max}((r, \pi(r,s), s);\text{no-wait}, A, B) + \sum_{g=i+1}^{m}p_{gs}.$$

Step 3. Find r^* and s^* that minimize $C(r, s)$, and let $\pi_{hi} = (r^*, \pi(r^*, s^*), s^*)$ and $L_{hi} = C(r^*, s^*)$.

Since the computations of all a_k and b_k require O(mn) time, and the GG algorithm has to be repeated n^2 times when $1 < h < i < m$, the overall complexity of Algorithm 3 is $O(\max(m, n^2\log n)n)$. It reduces to $O(\max(m, n\log n)n)$ when $h = 1$ or $i = m$, and $O(\max(m, \log n)n)$ when $h = 1$ and $i = m$.

Definition 1. We say that machine M_g is

(i) [13] dominated by M_{g-1}, written $M_{g-1} \geqslant M_g$, if $p_{g-1,k} \geqslant p_{gj}$ for all $j \neq k$;
(ii) weakly dominated by M_{g-1} and M_{g+1}, written $(M_{g-1}, M_{g+1}) \geqslant {}_w M_g$, if $\max(p_{g-1,k} - p_{gj}, p_{g+1,j} - p_{gk}) \geqslant 0$ for all $j \neq k$.

Note that $M_{g-1} \geqslant M_g$ implies $(M_{g-1}, M_{g+1}) \geqslant {}_w M_g$.

Theorem 3. *For every job sequence π and $0 \leqslant h < i \leqslant m$, $C_{\max}(\pi;\text{no-wait}) \geqslant L_{hi}$. If $M_1 \geqslant M_2 \geqslant \cdots \geqslant M_h$, $(M_{g-1}, M_{g+1}) \geqslant {}_w M_g$ for $g = h+1, h+2, \ldots, i-1$, and $M_i \geqslant M_{i+1} \geqslant \cdots \geqslant M_m$, then π_{hi} is optimal for $Fm|\text{no-wait}|C_{\max}$.*

Proof. Observe first that the sequence π_{hi} minimizes

$$L_{hi}(\pi) = \sum_{g=1}^{h-1}p_{g,\pi(1)} + a_{\pi(1)} + \sum_{j=1}^{n-1}\max(a_{\pi(j+1)}, b_{\pi(j)}) + b_{\pi(n)} + \sum_{g=i+1}^{m}p_{g,\pi(n)}.$$

Since for every π and $j = 1, 2, \ldots, n-1$,

$$C_{\max}(\pi(j), \pi(j+1)) \geqslant \sum_{g=1}^{h}p_{g,\pi(j)} + \max(a_{\pi(j+1)}, b_{\pi(j)}) + \sum_{g=i}^{m}p_{g,\pi(j+1)}, \tag{1}$$

from Corollary 1(i) and the definition of L_{hi}, we have $C_{\max}(\pi;\text{no-wait}) \geqslant L_{hi}(\pi) \geqslant L_{hi}$.

For the machine dominance relations specified in the theorem, we have the equality in (1). Hence, $C_{\max}(\pi;\text{no-wait}) = L_{hi}(\pi)$ for every π, and since π_{hi} minimizes $L_{hi}(\pi)$, $C_{\max}(\pi_{hi};\text{no-wait}) = L_{hi}$. □

In the remainder of this subsection we consider the problem $Fm|\text{prmu, no-idle}|C_{\max}$. Let π_i be a sequence found by Johnson's algorithm applied on machines M_i and M_{i+1}, $i = 1,2,\ldots,m-1$. The makespan representation given in Corollary 1(ii) leads to the following result.

Theorem 4. *For every job sequence* π, $C_{\max}(\pi;\text{no-idle}) \geqslant L = \sum_{i=1}^{m-1} C_{\max}(\pi_i; M_i, M_{i+1}) - \sum_{i=2}^{m-1}\sum_{j=1}^{n} p_{ij}$. *If* $[\min(p_{hk}, p_{h+1,j}) - \min(p_{hj}, p_{h+1,k})]\,[\min(p_{ik}, p_{i+1,j}) - \min(p_{ij}, p_{i+1,k})] \geqslant 0$ *for all jobs* $j \neq k$ *and machines* M_h *and* M_i *such that* $1 \leqslant h < i < m$, *then the lower bound L is tight and* $Fm|\text{prmu, no-idle}|C_{\max}$ *polynomially solvable.*

Proof. The lower bound L was presented in [12]; its computations require $O(mn\log n)$ time. Recall that Johnson's algorithm applied on times a_k and b_k finds one of the so-called Johnson's sequences that satisfy the rule: j precedes k if $\min(a_k, b_j) > \min(a_j, b_k)$. Under the condition specified in the theorem, the sequence π_h may differ from π_i only in the ordering of jobs j and k for which $[\min(p_{hk}, p_{h+1,j}) - \min(p_{hj}, p_{h+1,k})][\min(p_{ik}, p_{i+1,j}) - \min(p_{ij}, p_{i+1,k})] = 0$. By breaking these ties successively [2,19], all π_i can be transformed into a Johnson's sequence π^* that is optimal for all (M_i, M_{i+1}) and satisfies $C_{\max}(\pi^*;\text{no-idle}) = L$. □

It is noteworthy that, as opposed to $Fm|\text{prmu, no-idle}|C_{\max}$, the optimality of π^* for all (M_i, M_{i+1}) does not guarantee the optimality of π^* in $Fm|\text{prmu}|C_{\max}$, see [2,13]. The efficiently solvable special case of $Fm|\text{prmu, no-idle}|C_{\max}$ presented in Theorem 4 corrects the erroneous case claimed in Theorem 4(iii) of [12].

4. Final remarks

The technique we adapted to model the makespans in no-wait and no-idle flow shops can also be implemented for some hybrid flow shops. To illustrate, consider the problem $Fm|\text{block}(1,2),\text{no-wait}|C_{\max}$, that is, there is no storage between M_1 and M_2, and the no-wait condition must be respected by the remaining machines. When the arcs with negative weights $-p_{2j}$ are deleted from Fig. 1a, the critical path length in the resulting network is the corresponding makespan. In particular, this shows that both $F3|\text{block}(1,2),\text{no-wait}(2,3)|C_{\max}$ and $F3|\text{no-wait}(1,2),\text{block}(2,3)|C_{\max}$ reduce to $F3|\text{block}|C_{\max}$, and hence they are strongly NP-hard; see Lemmas 2 and 3 in [9].

The lower bound L falls beyond the idea of a constructive heuristic for solving $Fm|\text{prmu, no-idle}|C_{\max}$ [11] that was shown to outperform significantly earlier heuristics. We strongly believe that the sequences π_{hi} (in particular π_{1m}) can be used to develop an efficient $Fm|\text{no-wait}|C_{\max}$ constructive heuristic whose solution will become a good initial sequence for metaheuristics; see e.g. [4,8]. We also believe that the lower bounds L and L_{hi} will find applications in developing effective branch and bound algorithms.

We identified duality relations that exist between $Fm|\text{no-wait}|C_{\max}$ and $Fm|\text{prmu, no-idle}|C_{\max}$, and $Fm|\text{block}|C_{\max}$ and $Fm|\text{prmu,busy}|C_{\max}$. Future research is necessary to better explore the observed dualities. We hope that the new theoretical problem $Fm|\text{prmu,busy}|C_{\max}$ will find validations in real world flow shop scheduling.

Acknowledgements

The authors are grateful to three anonymous referees for their valuable suggestions and comments.

References

[1] I.N.L. Abadi, N.G. Hall, C. Sriskandarajah, Minimizing cycle time in a blocking flowshop, Operations Research 48 (2000) 177–180.
[2] N.R. Achuthan, A special case of the ($n/m/F/F$max) problem, Opsearch 14 (1977) 71–87.
[3] I. Adiri, D. Pohoryles, Flow-shop/no-idle or no-wait scheduling to minimise the sum of completion times, Naval Research Logistics Quarterly 29 (1982) 495–504.
[4] A. Allahverdi, T. Aldowaisan, No-wait flowshops with bicriteria of makespan and maximum lateness, European Journal of Operational Research 152 (2004) 132–147.

[5] P. Baptiste, K.H. Lee, A branch and bound algorithm for the $F|\text{no-idle}|C_{\max}$, in: International Conference on Industrial Engineering and Production Management (IEPM'97), Lyon, 1997.
[6] S. Elmaghraby, J. Kamburowski, The analysis of activity networks under generalized precedence relations (GPRs), Management Science 38 (1992) 1245–1263.
[7] P.C. Gilmore, R.E. Gomory, Sequencing a one state variable machine: A solvable case of the traveling salesman problem, Operations Research 12 (1964) 655–679.
[8] J. Grabowski, J. Pempera, Some local search algorithms for no-wait flow-shop problem with makespan criterion, Computers and Operations Research 32 (2005) 2197–2212.
[9] N.G. Hall, C. Sriskandarajah, A survey of machine scheduling problems with blocking and no-wait in process, Operations Research 44 (1996) 510–515.
[10] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quarterly 1 (1954) 61–68.
[11] P.J. Kalczynski, J. Kamburowski, A heuristic for minimizing the makespan in no-idle flow shops, Computers and Industrial Engineering 49 (2005) 146–154.
[12] J. Kamburowski, More on three-machine no-idle flow shops, Computers and Industrial Engineering 46 (2004) 461–466.
[13] C.L. Monma, A.H.G. Rinnoy-Kan, A concise survey of efficiently solvable special cases of the permutation flow-shop problem, RAIRO Recherche Operationelle 17 (1983) 105–119.
[14] J. Piehler, Ein Beitrag Zum Reinhenfolgeproblem, Unternehmensforschung 4 (1960) 138–142.
[15] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice Hall, Upper Saddle, NJ, 2002.
[16] S.S. Reddi, C.V. Ramamoorthy, On the flowshop sequencing problem with no-wait in process, Operational Research Quarterly 23 (1972) 323–331.
[17] H. Röck, The three machine no-wait flowshop problem is NP-complete, Journal of the Association for Computing Machinery 31 (1984) 336–345.
[18] H. Saadani, A. Guinet, M. Moalla, Three stage no-idle flow-shops, Computers and Industrial Engineering 44 (2003) 425–434.
[19] W. Szwarc, Optimal two-machine orderings in the $3 \times n$ flow-shop problem, Operations Research 25 (1977) 70–77.
[20] G. Vairaktarakis, Simple algorithms for Gilmore–Gomory's traveling salesman and related problems, Journal of Scheduling 6 (2003) 499–520.
[21] D.A. Wismer, Solution of the flowshop scheduling problem with no intermediate queue, Operations Research 20 (1972) 689–697.