3

Scheduling under Unavailability Constraints to Minimize Flow-time Criteria

Imed Kacem
Institut Charles Delaunay, Université de Technologie de Troyes
France

1. Introduction

In this chapter, we consider some practical scheduling problems under unavailability constraints (breakdown periods, maintenance durations and/or setup times). Such problems can be met in different industrial environments and be associated to numerous real-life applications. This explains why many researchers have become interested in this subject. We aim to present the recent approaches proposed to solve these problems and to discuss their performances. This family of scheduling problems, addressed in this chapter, has been intensively studied (Kacem [8], Lee [17], Schmidt [24]). The studied criteria in this chapter are related to the flowtime minimization (the weighted and unweighted cases). The chapter is organized in two main parts. The first part focuses on the single machine scheduling problem (see Section 2). The second part is devoted to the parallel machine scheduling problem (see Section 3). In each part, we present the main contributions and explain their principles (complexity results, heuristic algorithms and their worstcase performance, existing approximation schemes, exact methods, branch-and-bound algorithms, dynamic programming, integer linear models, lower bounds. . .). Finally, Section 4 concludes the paper.

2. The single machine case

The minimization of the total completion time on a single machine with a fixed non-availability interval (denoted 1, $h_1|n-res|\sum C_i$), is NP-Hard according to Adiri et al. [1] and Lee and Liman [18]. Several references proposed exact and heuristic methods (a sample of these papers includes Adiri et al. [1]; Lee and Liman [18]; Sadfi et al. [21] and Breit [3]).
Numerous researchers addressed the problem of scheduling jobs and maintenance tasks together on a single machine (a sample of them includes Qi et al. [20] and Chen [4] who addressed the total flow-time minimization). Others recent references focused on the shop scheduling problems (parallel-machine, flow shop and job shop problems) and designed exact and heuristic approaches to solve them (Lee and Liman [19]; Lee [16]; Schmidt [24]; Lee [17]).
This first part of this chapter addresses the following problem. We have n jobs $\{J_1, J_2, ..., J_n\}$ to schedule on a single machine. To every job i it is associated a processing time p_i and a weight w_i. The machine is unavailable during a fixed interval $[T_1, T_2)$ and it can process at most one job at a time. We assume that all data are integers and that jobs are sorted

Source: Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,
ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

according to the WSPT rule (i.e., $\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}$). It is well-known that the WSPT order is dominant (i.e., every optimal solution is composed of two sequences such that jobs are scheduled in the WSPT order in each sequence). The objective function to minimize is the total weighted completion time (flow-time).
It is easy to verify that the studied problem (noted $\mathcal{P}$) can be solved optimally by the WSPT rule (Smith [25]) if the total processing time is less than T_1.
In the remainder of this chapter, $\varphi^*(\pi)$ represents the minimal weighted flow-time for the problem π and $\varphi\sigma(\pi)$ is the weighted flow-time of sequence σ for problem π. We also define the non-availability interval length as follows: $\Delta T = T_2 - T_1$.
Moreover, we define $(g+1)$ as the critical job in the WSPT sequence, i.e., $Q_g \leq T_1$ and $Q_{g+1} > T_1$. Finally, let Q_k and δ be the variables defined as follows:

$$Q_k = \sum_{i=1}^{k} p_i \tag{1}$$

$$\delta = T_1 - Q_g \tag{2}$$

Theorem 1 ([*1*]-[*18*]) *The problem 1,* $h_1|n - res|\sum C_i$ *is NP-Hard.*
Theorem 2 [*18*] *The problem 1,* $h_1|res|\sum C_i$ *can be optimally solved using the Shortest Remaining Processing Time rule.*
Theorem 3 [*18*] *The problem 1,* $h_1|res|\sum w_i C_i$ *is NP-Hard.*

2.1 Mixed Integer Programming (Kacem, Chu and Souissi [12])

Kacem, Chu and Souissi proved that the problem $\mathcal{P}$ can be formulated using the following mixed integer model:

$$(\mathcal{S}) : Minimize \sum_{i=1}^{n} w_i C_i$$

Subject to:

$$C_i \geq (1 - x_i)\left(T_2 + \sum_{j=1}^{i} p_j\right) - \sum_{j=1}^{i-1} x_j p_j \quad \forall 1 \leq i \leq n \tag{3}$$

$$C_i \geq \sum_{j=1}^{i} x_j p_j \quad \forall 1 \leq i \leq n \tag{4}$$

$$\sum_{j=1}^{n} x_j p_j \leq T_1 \tag{5}$$

where $x_i \in \{0, 1\}$ $\forall i \in \{1, 2, .., n\}$. Note that $x_i = 1$ if job i is scheduled before T_1 and $x_i = 0$ if job i is scheduled after T_2.

The first constraint (3) determines the completion time C_i of job i if it is performed after T_2. Constraint (4) gives this completion time if job i is performed before T_1. Finally, constraint (5) represents the workload constraint for processing jobs before the fixed non-availability interval.

2.2 Branch-and-bound procedures ([12]- [13]-[21])

The first branch-and-bound algorithm was proposed by Sadfi et al. [22] for solving the unweighted case (w_i = 1 for every job i). The algorithm is based on the SRPT lower bound and the MSPT heuristic proposed by Sadfi et al. [21]. As it is mentioned before, the problem consists to find two subsets: the subsets of jobs to be scheduled before and after the non-availability interval. Each subset respects the SPT order. Therefore, the branching scheme is based on considering the two possibilities of assignment for every job.

Kacem and Chu [13] and Kacem et al. [12] considered the weighted case. Similarly, the problem is also reduced to determine if every job has to be scheduled before or after the unavailability period. Obviously, in the optimal solution, the subset of jobs scheduled before T_1 and the subset of jobs scheduled after T_2 are performed in the WSPT order. Consequently, every node is represented by the following elements:

- the number of scheduled jobs denoted by k,
- a partial assignment vector: $PA = \{a_1, a_2, ..., a_k\}$ with $a_i \in \{0, 1\}\ \forall i \leq k$ and $a_i = 1$ if job i is performed before T_1 and $a_i = 0$ otherwise,
- a lower bound LB formulated in Equation 11.

The upper bound UB is obtained by taking the best result yielded by some heuristics (described later in this chapter). At each new branching step, one explore two possibilities; the first one is to perform job (k + 1) before T_1 (a_{k+1} = 1) and the second possibility is to schedule it after T_2 (a_{k+1} = 0). If the lower bound is greater than the current upper bound, then the corresponding node is removed.

In the remainder of this subsection, we present the major results (i.e., the lower bounds) proposed in the above branch-and-bound algorithm. The heuristics used in such an algorithm will be described later in this section.

Theorem 4 (*Wang et al.* [*26*], *Lee* [*16*]) *Let $\mathcal{P}'$ denote the resumable scenario of problem $\mathcal{P}$. Therefore, the following relation holds: $w_{g+1}\Delta T \geq \varphi_{WSRPT}(\mathcal{P}') - \varphi^*(\mathcal{P})$ where WSRPT (Weighted Shortest Remaining Processing Time) is the rule that consists in scheduling jobs according to the WSPT order under the resumable scenario.*

Example 1 *We consider the following four-job instance: p_1 = 2; w_1 = 4; p_2 = 3; w_2 = 5; p_3 = 2; w_3 = 3; p_4 = 1; w_4 = 1; T_1 = 6; ΔT = 2. Given this instance, we have: g + 1 = 3. Figure* 1 *shows the schedules obtained by using the WSPT and the WSRPT rules.*

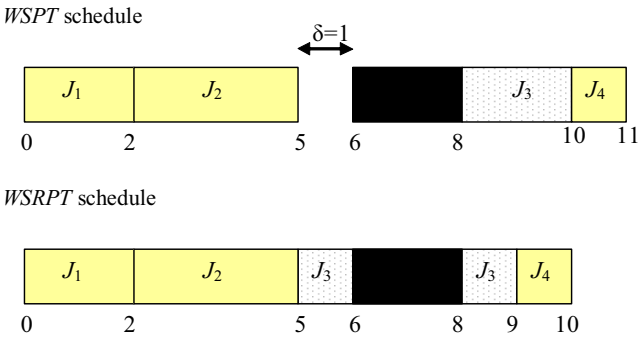


Figure 1. Illustration of the rules *WSPT* and *WSRPT*

From Theorem 4, we can show the following proposition.
Proposition 1 ([*26*], [*16*]) *Let*

$$lb_1 = \sum_{i=1}^{g+1} w_i Q_i + \sum_{i=g+2}^{n} w_i \left(Q_i + \Delta T\right) \tag{6}$$

The quantity lb_1 is a lower bound on the optimal weighted flow-time for problem $\mathcal{P}$.
Theorem 5 (Kacem, Chu and Souissi [*12*]) *Let*

$$lb_2 = \sum_{i=1}^{g+1} w_i Q_i + \sum_{i=g+2}^{n} w_i \left(Q_i + \Delta T\right) + w_{g+1} \frac{\Delta T}{p_{g+1}} \left(p_{g+1} - \delta\right) \tag{7}$$

The quantity lb_2 is a lower bound on the optimal weighted flow-time for problem $\mathcal{P}$ and it dominates lb_1.
Theorem 6 (Kacem and Chu [*13*]) *For every instance of $\mathcal{P}$, the lower bound lb_2 is greater than lb_0 (lb_0 denotes the weighted flow-time value obtained by solving the relaxation of the linear model by assuming that $x_i \in [0, 1]$).*
In order to improve the lower bound lb_2, Kacem and Chu proposed to use the fact that job $(g+1)$ must be scheduled before or after the non-availability interval (i.e., either $C_{g+1} \geq T_2 + p_{g+1}$ or $C_{g+1} \leq T_1$ must hold). By applying a clever lagrangian relaxation, a stronger lower bound lb_3 has been proposed:
Theorem 7 (Kacem and Chu [*13*]) *Let*

$$lb_3 = lb_2 + \min\left(\lambda_1^* \left(\delta + \Delta T \frac{\delta}{p_{g+1}}\right), \lambda_2^* \left(p_{g+1} - \delta\right)\left(1 + \frac{\Delta T}{p_{g+1}}\right)\right) \tag{8}$$

with $\lambda_1^* = \begin{cases} w_{g+1} - \frac{p_{g+1}}{p_{g+2}} w_{g+2} & \text{if job } (g+2) \text{ exists} \\ w_{g+1} & \text{otherwise} \end{cases}$
and $\lambda_2^* = \frac{p_{g+1}}{p_g} w_g - w_{g+1}$.

The quantity lb_3 is a lower bound on the optimal weighted flow-time for problem $\mathcal{P}$ and it dominates lb_2.

Another possible improvement can be carried out using the splitting principle (introduced by Belouadah et al. [2] and used by other authors [27] for solving flow-time minimization problems). The splitting consists in subdividing jobs into pieces so that the new problem can be solved exactly. Therefore, one divide every job i into n_i pieces, such that each piece (i, k) has a processing time p_i^k and a weight w_i^k ($\forall 1 \leq k \leq n_i$), with $p_i = \sum_{k=1}^{n_i} p_i^k$ and $w_i = \sum_{k=1}^{n_i} w_i^k$.

Using the splitting principle, Kacem and Chu established the following theorem.

Theorem 8 (Kacem and Chu [*13*]) *Index z_1 denotes the job such that $Q_{z_1} > T_1 - p_{g+1}$ and $Q_{z_1-1} \leq T_1 - p_{g+1}$ and index z_2 denotes the job such that $\sum_{i=g+2}^{z_2-1} p_i \leq \delta$ and $\sum_{i=g+2}^{z_2} p_i > \delta$. We also define $\delta_1 = T_1 - p_{g+1} - Q_{z_1-1}$ and $\delta_2 = \delta - \sum_{i=g+2}^{z_2-1} p_i$. Therefore, the quantity $lb_4 = \min(\gamma_1, \gamma_2)$ is a lower bound on the optimal weighted flow-time for $\mathcal{P}$ and it dominates lb_3, where*

$$\begin{aligned}\gamma_1 &= \sum_{i=1}^{z_1-1} w_i Q_i + \sum_{i=z_1+1}^{g} w_i \left(\Delta T + p_{g+1} + Q_i\right) + \frac{w_{z_1}}{p_{z_1}} \delta_1 \left(T_1 - p_{g+1}\right) \\ &\quad + w_{g+1} T_1 + \sum_{i=g+2}^{n} w_i \left(\Delta T + Q_i\right) + \frac{w_{z_1}}{p_{z_1}} \left(p_{z_1} - \delta_1\right) \left(T_2 + p_{z_1}\right)\end{aligned} \tag{9}$$

and

$$\begin{aligned}\gamma_2 &= \sum_{i=1}^{g} w_i Q_i + \sum_{i=g+2}^{z_2-1} w_i \left(Q_i - p_{g+1}\right) + w_{z_2} \frac{\delta_2}{p_{z_2}} T_1 + w_{g+1} \left(T_2 + p_{g+1}\right) \\ &\quad + w_{z_2} \frac{p_{z_2} - \delta_2}{p_{z_2}} \left(T_2 + p_{g+1} + p_{z_2}\right) + \sum_{i=z_2+1}^{n} w_i \left(\Delta T + Q_i\right)\end{aligned} \tag{10}$$

By using another decomposition, Kacem and Chu have proposed another complementary lower bound:

Theorem 9 (*Kacem, Chu and Souissi* [*12*]) *Let*

$$lb_5 = lb_2 + \Delta T \left(w_{g+1} \frac{\delta}{p_{g+1}} - \left\lfloor w_{g+1} \frac{\delta}{p_{g+1}} \right\rfloor \right).$$

The quantity lb_5 is a lower bound on the optimal weighted flow-time for problem $\mathcal{P}$ and it dominates lb_2.

In conclusion, these last two lower bounds (lb_4 and lb_5) are usually greater than the other bounds for every instance. These lower bounds have a complexity time of $O(n)$ (since jobs are indexed according to the WSPT order). For this reason, Kacem and Chu used all of them (lb_4 and lb_5) as complementary lower bounds. The lower bound LB used in their branch-and-bound algorithm is defined as follows:

$$LB = \max\left\{lb_4, lb_5\right\} \tag{11}$$

2.3 Approximation algorithms

2.3.1 Heuristics and worst-case analysis

The problem (1, $h_1|n-res|\sum w_iC_i$) was studied by Kacem and Chu [11] under the non-resumable scenario. They showed that both WSPT[1] and MWSPT[2] rules have a tight worst-case performance ratio of 3 under some conditions. Kellerer and Strusevich [14] proposed a 4-approximation by converting the resumable solution of Wang et al. [26] into a feasible solution for the non-resumable scenario. Kacem proposed a 2-approximation algorithm which can be implemented in $O(n^2)$ time [10]. Kellerer and Strusevich proposed also an FPTAS (Fully Polynomial Time Approximation Scheme) with $O(n^4/\epsilon^2)$ time complexity [14].

WSPT and MWSPT These heuristics were proposed by Kacem and Chu [11]. MWSPT heuristic consists of two steps. In the first step, we schedule jobs according to the WSPT order (g is the last job scheduled before T_1). In the second step, we insert job i before T_1 if $p_i \leq \delta$ (we test this possibility for each job $i \in \{g+2, g+3, ..., n\}$ and after every insertion, we set $\delta \leftarrow \delta - p_i$).

To illustrate this heuristic, we consider the four-job instance presented in Example 1. Figure 2 shows the schedules obtained by using the WSPT and the MWSPT rules. Thus, it can be established that: $\varphi_{WSPT}(\mathcal{P})= 74$ and $\varphi_{MWSPT}(\mathcal{P})= 69$.

Remark 1 *The MWSPT rule can be implemented in O* (n log (n)) *time.*

Theorem 10 (Kacem and Chu [11]) *WSPT and MWSPT have a tight worst-case performance bound of* 3 *if* $\Delta t \leq \max_{1\leq i\leq g+1} p_i$. *Otherwise, this bound can be arbitrarily large.*

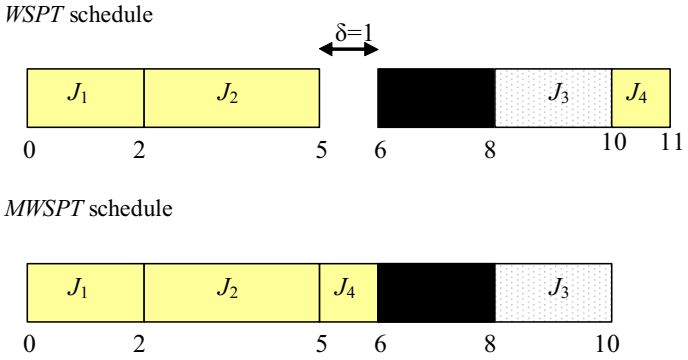


Figure 2. Illustration of MWSPT

MSPT: the weighted and the unweighted cases The weighted case of this heuristic can be described as follows (Kacem and Chu [13]). First, we schedule jobs according to the WSPT order (g is the last job scheduled before T_1). In the second step, we try to improve the WSPT solution by testing an exchange of jobs i and j if possible, where i =1,…, g and j = g+1,…, n. The best exchange is considered as the obtained solution.

Remark 2 *MSPT has a time complexity of O* (n^3).

To illustrate this improved heuristic, we use the same example. For this example we have:

---

[1] WSPT: Weighted Shortest Processing Time

[2] MWSPT: Modified Weighted Shortest Processing Time

g+ 1 = 3. Therefore, four possible exchanges have to be distinguished: (J_1 and J_3), (J_1 and J_4), (J_2 and J_3) and (J_2 and J_4). Figure 3 depicts the solutions corresponding to these exchanges. By computing the corresponding weighted flow-time, we obtain $\varphi_{MSPT}(\mathcal{P})=\varphi_{WSPT}(\mathcal{P})$.
The weighted version of this heuristic has been used by Kacem and Chu in their branch-and-bound algorithm [13]. For the unweighted case (w_i = 1), Sadfi et al. studied the worst-case performance of the *MSPT* heuristic and established the following theorem:
Theorem 11 (Sadfi et al. [21]) *MSPT has a tight worst-case performance bound of* 20/17 *when* w_i=1 *for every job i.*
Recently, Breit improved the result obtained by Sadfi et al. and proposed a better worst-case performance bound for the unweighted case [3].

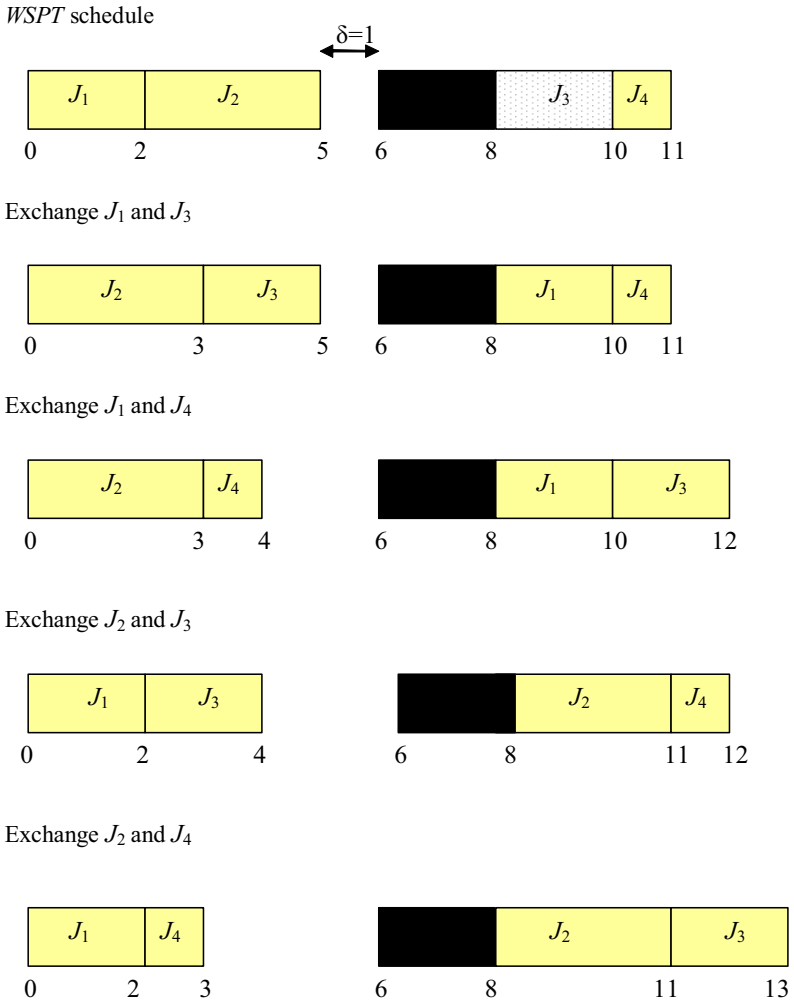


Figure 3. Illustration of *MSPT* for the weighted case

Critical job-based heuristic (*HS*) [**10**] This heuristic represents an extension of the one proposed by Wang et al. [26] for the resumable scenario. It is based on the following algorithm (Kacem [**10**]):

i. Let $l = 0$ and $\mathcal{G}_l = \emptyset$.

ii. Let $\pi\ (i, l)$ be the ith job in $J - \mathcal{G}_l$ according to the WSPT order. Construct a schedule $\sigma_l = \langle \pi\ (1, l)\ ,\ \pi\ (2, l),\ ...,\ \pi\ (g\ (l)\ ,\ l),\ \mathcal{G}_l,\ \pi\ (g(l) + 1, l),\ ...,\ \pi\ (n - |\mathcal{G}_l|\ ,\ l)\ \rangle$ such that $\sum_{i\in\mathcal{G}_l} p_i + \sum_{i=1}^{g(l)} p_{\pi(i,l)} \leq T_1$ and $\sum_{i\in\mathcal{G}_l} p_i + \sum_{i=1}^{g(l)+1} p_{\pi(i,l)} > T_1$ where jobs in $\mathcal{G}_l$ are sequenced according to the WSPT order.

iii. If $\sum_{i\in\mathcal{G}_l} p_i + p_{\pi(g(l)+1,l)} \leq T_1$, then: $\mathcal{G}_{l+1} = \{\pi\,(g\,(l) + 1, l)\} \cup \mathcal{G}_l;\ l = l + 1$; go to step (ii). Otherwise, go to step (iv).

iv. $\varphi_{HS}\,(\mathcal{P}) = \min_{0\leq h\leq l} \{\varphi_{\sigma_h}\,(\mathcal{P})\}$.

Remark 3 *HS can be implemented in O* (n^2) *time.*

We consider the previous example to illustrate *HS*. Figure 4 shows the sequences σ^h ($0 \leq h \leq l$) generated by the algorithm. For this instance, we have $l = 2$ and $\varphi_{HS}\,(\mathcal{P}) = \varphi_{WSPT}\,(\mathcal{P})$.

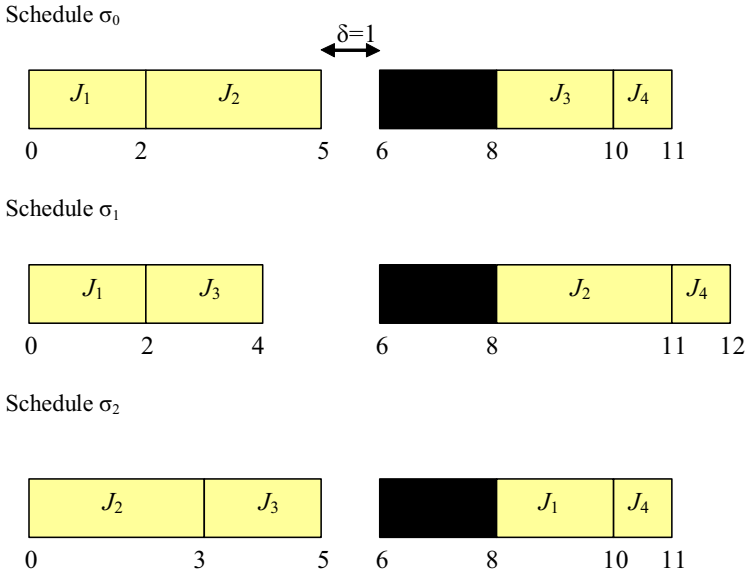


Figure 4. Illustration of heuristic *HS*

Theorem 12 (Kacem [10]) *Heuristic HS is a* 2*-approximation algorithm for problem S and its worst-case performance ratio is tight.*

2.3.2 Dynamic programming and FPTAS

The problem can be optimally solved by applying the following dynamic programming algorithm *AS*, which is a weak version of the one proposed by Kacem et al [12]. This algorithm generates iteratively some sets of states. At every iteration *k*, a set $\mathcal{Z}_k$ composed of states is generated ($1 \leq k \leq n$). Each state $[t, f]$ in $\mathcal{Z}_k$ can be associated to a feasible schedule for the first *k* jobs.

Variable t denotes the completion time of the last job scheduled before T_1 and f is the total weighted flow-time of the corresponding schedule. This algorithm can be described as follows:

Algorithm *AS*

i. Set $\mathcal{Z}_1 = \{[0, w_1(T_2 + p_1)] , [p_1, w_1p_1]\}$.

ii. For $k \in \{2, 3, ..., n\}$,

For every state $[t, f]$ in $\mathcal{Z}_{k-1}$:

1) Put $\left[t, f + w_k(T_2 + \sum_{i=1}^{k} p_i - t)\right]$ in $\mathcal{Z}_k$

2) Put $\left[t + p_k, f + w_k\,(t + p_k)\right]$ in $\mathcal{Z}_k$ if $t + p_k \leq T_1$

Remove $\mathcal{Z}_{k-1}$

iii. $\varphi^*(\mathcal{P}) = \min_{[t,f] \in \mathcal{Z}_n} \{f\}$.

Let UB'' be an upper bound on the optimal weighted flow-time for problem $(\mathcal{P})$. If we add the restriction that for every state $[t, f]$ the relation $f \leq UB''$ must hold, then the running time of *AS* can be bounded by nT_1UB'' (by keeping only one vector for each state). Indeed, t and f are integers and at each step k, we have to create at most T_1UB″ states to construct $\mathcal{Z}_k$. Moreover, the complexity of *AS* is proportional to $\sum_{k=1}^{n} |\mathcal{Z}_k|$.

However, this complexity can be reduced to $O(nT_1)$ as it was done by Kacem et al [12], by choosing at each iteration k and for every t the state $[t, f]$ with the smallest value of f.

In the remainder of this chapter, algorithm *AS* denotes the weak version of the dynamic programming algorithm by taking $UB'' = \varphi_{HS}(\mathcal{P})$, where *HS* is the heuristic proposed by Kacem [10].

The algorithm starts by computing the upper bound yielded by algorithm *HS*.

In the second step of our FPTAS, we modify the execution of algorithm *AS* in order to reduce the running time. The main idea is to remove a special part of the states generated by the algorithm. Therefore, the modified algorithm *AS*′ becomes faster and yields an approximate solution instead of the optimal schedule.

The approach of *modifying the execution* of an exact algorithm to design FPTAS, was initially proposed by Ibarra and Kim for solving the knapsack problem [7]. It is noteworthy that during the last decades numerous scheduling problems have been addressed by applying such an approach (a sample of these papers includes Gens and Levner [6], Kacem [8], Sahni [23], Kovalyov and Kubiak [15], Kellerer and Strusevich [14] and Woeginger [28]-[29]).

Given an arbitrary $\varepsilon > 0$, we define $LB' = \frac{\varphi_{HS}(\mathcal{P})}{2}$, $m_1 = \left\lceil \frac{4n}{\varepsilon} \right\rceil$, $m_2 = \left\lceil \frac{2}{\varepsilon} \right\rceil$, $\delta'_1 = \frac{\varphi_{HS}(\mathcal{P})}{m_1}$ and $\delta'_2 = \frac{T_1}{m_2}$. We split the interval $[0, \varphi_{HS}(\mathcal{P})]$ into m_1 equal subintervals $I_r^1 = [(r-1)\delta'_1, r\delta'_1]_{1 \leq r \leq m_1}$ of length δ'_1. We also split the interval $[0, T_1]$ into m_2 equal subintervals $I_s^2 = [(s-1)\delta'_2, s\delta'_2]_{1 \leq s \leq m_2}$ of length δ'_2. The algorithm AS'_ε generates reduced sets $\mathcal{Z}_k^{\#}$ instead of sets $\mathcal{Z}_k$. Also, it uses artificially an additional variable w^+ for every state, which denotes the sum of weights of jobs scheduled after T_2 for the corresponding state. It can be described as follows:

Algorithm AS'_ε

i. Set $\mathcal{Z}_1^{\#} = \{[0, w_1(T_2 + p_1), w_1] , [p_1, w_1p_1, 0]\}.$,

ii. For $k \in \{2, 3, ..., n\}$,

For every state $[t, f, w^+]$ in $\mathcal{Z}_{k-1}^{\#}$:

1) Put $\left[t, f + w_k(T_2 + \sum_{i=1}^{k} p_i - t), w^+ + w_k\right]$ in $\mathcal{Z}_k^{\#}$

2) Put $\left[t + p_k, f + w_k\left(t + p_k\right), w^+\right]$ in $\mathcal{Z}_k^{\#}$ if $t + p_k \leq T_1$

Remove $\mathcal{Z}_{k-1}^{\#}$

Let $[t, f, w^+]_{r,s}$ be the state in $\mathcal{Z}_k^{\#}$ such that $f \in I_r^1$ and $t \in I_s^2$ with the smallest possible t (ties are broken by choosing the sate of the smallest f). Set $\mathcal{Z}_k^{\#} = \left\{[t, f, w^+]_{r,s} \,|\, 1 \leq r \leq m_1, 1 \leq s \leq m_2\right\}$.

iii. $\varphi_{AS'_\varepsilon}(\mathcal{P}) = \min_{[t,f,w^+] \in \mathcal{Z}_n^{\#}} \{f\}$.

The worst-case analysis of this FPTAS is based on the comparison of the execution of algorithms *AS* and *AS'*$_\varepsilon$. In particular, we focus on the comparison of the states generated by each of the two algorithms. We can remark that the main action of algorithm *AS'*$_\varepsilon$ consists in reducing the cardinal of the state subsets by splitting $[0, \varphi_{HS}(\mathcal{P})] \times [0, T_1]$ into $m_1 m_2$ boxes $I_r^1 \times I_s^2$ and by replacing all the vectors of $\mathcal{Z}_k$ belonging to $I_r^1 \times I_s^2$ by a single "approximate" state with the smallest *t*.

Theorem 13 (Kacem [9]) *Given an arbitrary* $\varepsilon > 0$, *algorithm AS' can be implemented in O* (n^2/ε^2) time and *it yields an output* $\varphi_{AS'_\varepsilon}(\mathcal{P})$ such that: $\varphi_{AS'_\varepsilon}(\mathcal{P}) / \varphi^*(\mathcal{P}) \leq 1 + \varepsilon$.

From Theorem 13, algorithm *AS'*$_\varepsilon$ is an FPTAS for the problem 1, $h_1|n - res| \sum w_i C_i$.

Remark 4 *The approach of Woeginger* [28]-[29] *can also be applied to obtain FPTAS for this problem. However, this needs an implementation in O* $(|I|^3 n^3/\varepsilon^3)$, where $|I|$ is the input size.

3. The two-parallel machine case

This problem for the unweighted case was studied by Lee and Liman [19]. They proved that the problem is NP-complete and provided a pseudo-polynomial dynamic programming algorithm to solve it. They also proposed a heuristic that has a worst case performance ratio of 3/2.

The problem is to schedule *n* jobs on two-parallel machines, with the aim of minimizing the total weighted completion time. Every job *i* has a processing time p_i and a weight w_i. The first machine is available for a specified period of time [0, T_1] (i.e., after T_1 it can no longer process any job). Every machine can process at most one job at a time. With no loss of generality, we consider that all data are integers and that jobs are indexed according to the *WSPT* rule: $\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}$. Due to the dominance of the *WSPT* order, an optimal solution is composed of two sequences (one sequence for each machine) of jobs scheduled in non-decreasing order of their indexes (Smith [25]). In the remainder of the paper, $(\mathcal{P})$ denotes the studied problem, $\varphi^*(Q)$ denotes the minimal weighted sum of the completion times for problem *Q* and $\varphi_S(Q)$ is the weighted sum of the completion times of schedule *S* for problem *Q*.

3.1 The unweighted case

In this subsection, we consider the unweighted case of the problem, i.e., for every job *i*, we have $w_i = 1$. Hence, the *WSPT* order becomes: $p_1 \leq p_2 \leq \ldots \leq p_n$.

In this case, we can easily remark the following property.

Proposition 2 (Kacem [9]) *If* $\sum_{i=1}^{n} p_i \leq 2T_1$, *then problem* $(\mathcal{P})$ *can be optimally solved in O*(*n*log (*n*)) *time*.

Based on the result of Proposition 2, we only consider the case where $\sum_{i=1}^{n} p_i > 2T_1$.

3.1.1 Dynamic programming

The problem can be optimally solved by applying the following dynamic programming algorithm *A*, which is a weak version of the one proposed by Lee and Liman [19]. This algorithm generates iteratively some sets of states. At every iteration *k*, a set $\mathcal{V}_k$ composed of states is generated ($1 \leq k \leq n$). Each state $[t, f]$ in $\mathcal{V}_k$ can be associated to a feasible schedule for the first *k* jobs. Variable *t* denotes the completion time of the last job scheduled on the first machine before T_1 and *f* is the total flow-time of the corresponding schedule. This algorithm can be described as follows:

Algorithm *A*

i. Set $\mathcal{V}_1 = \{[0, p_1], [p_1, p_1]\}$.

ii. For $k \in \{2, 3, ..., n\}$,
   For every state $[t, f]$ in $\mathcal{V}_{k-1}$:
   1) Put $\left[t, f + \sum_{i=1}^{k} p_i - t\right]$ in $\mathcal{V}_k$
   2) Put $[t + p_k, f + t + p_k]$ in $\mathcal{V}_k$ if $t + p_k \leq T_1$
   Remove $\mathcal{V}_{k-1}$

iii. $\varphi^*(\mathcal{P}) = \min_{[t,f] \in \mathcal{V}_n} \{f\}$.

Let *UB* be an upper bound on the optimal flow-time for problem $(\mathcal{P})$. If we add the restriction that for every state $[t, f]$ the relation $f \leq UB$ must hold, then the running time of *A* can be bounded by nT_1UB. Indeed, *t* and *f* are integers and at each iteration *k*, we have to create at most T_1UB states to construct $\mathcal{V}_k$. Moreover, the complexity of *A* is proportional to $\sum_{k=1}^{n} |\mathcal{V}_k|$.

However, this complexity can be reduced to $O(nT_1)$ as it was done by Lee and Liman [19], by choosing at each iteration *k* and for every *t* the state $[t, f]$ with the smallest value of *f*. In the remainder of the paper, algorithm *A* denotes the weak version of the dynamic programming algorithm by taking $UB = \varphi_H(\mathcal{P})$, where *H* is the heuristic proposed by Lee and Liman [19].

3.1.2 FPTAS (Kacem [9])

The FPTAS is based on two steps. First, we use the heuristic *H* by Lee and Liman [19]. Then, we apply a modified dynamic programming algorithm. Note that heuristic *H* has a worst-case performance ratio of 3/2 and it can be implemented in $O(n \log(n))$ time [19].

In the second step of our FPTAS, we modify the execution of algorithm *A* in order to reduce the running time. Therefore, the modified algorithm becomes faster and yields an approximate solution instead of the optimal schedule.

Given an arbitrary $\varepsilon > 0$, we define $LB = \frac{2\varphi_H(\mathcal{P})}{3}$, $q_1 = \left\lceil \frac{3n}{\varepsilon} \right\rceil$, $q_2 = \left\lceil \frac{n}{\varepsilon} \right\rceil$, $\delta_1 = \frac{\varphi_H(\mathcal{P})}{q_1}$ and $\delta_2 = \frac{T_1}{q_2}$. We split the interval $[0, \varphi_H(\mathcal{P})]$ into q_1 equal subintervals $I_r^1 = [(r-1)\delta_1, r\delta_1]_{1 \leq r \leq q_1}$ of length δ_1. We also split the interval $[0, T_1]$ into q_2 equal subintervals $I_s^2 = [(s-1)\delta_2, s\delta_2]_{1 \leq s \leq q_2}$ of length δ_2.

Our algorithm A'_ε generates reduced sets $\mathcal{V}_k^{\#}$ instead of sets $\mathcal{V}_k$. The algorithm can be described as follows:

Algorithm A'_ε

i. Set $\mathcal{V}_1^{\#} = \{[0, p_1], [p_1, p_1]\}$

ii. For $k \in \{2, 3, ..., n\}$,

For every state $[t, f]$ in $\mathcal{V}_{k-1}^{\#}$

1) Put $\left[t, f + (\sum_{i=1}^{k} p_i - t)\right]$ in $\mathcal{V}_k^{\#}$

2) Put $[t + p_k, f + (t + p_k)]$ in $\mathcal{V}_k^{\#}$ if $t + p_k \leq T_1$

Remove $\mathcal{V}_{k-1}^{\#}$

Let $[t, f]_{r,s}$ be the state in $\mathcal{V}_k^{\#}$ such that $f \in I_r^1$ and $t \in I_s^2$ with the smallest possible t (ties are broken by choosing the state of the smallest f).

Set $\mathcal{V}_k^{\#} = \left\{[t, f]_{r,s} \,|\, 1 \leq r \leq q_1, 1 \leq s \leq q_2\right\}$.

iii. $\varphi_{A'_\varepsilon}(\mathcal{P}) = \min_{[t,f] \in \mathcal{V}_n^{\#}} \{f\}$.

The worst-case analysis of our FPTAS is based on the comparison of the execution of algorithms A and A'_ε. In particular, we focus on the comparison of the states generated by each of the two algorithms. We can remark that the main action of algorithm A'_ε consists in reducing the cardinal of the state subsets by splitting $[0, \varphi_H(\mathcal{P})] \times [0, T_1]$ into $q_1 q_2$ boxes $I_r^1 \times I_s^2$ and by replacing all the vectors of $\mathcal{V}_k$ belonging to $I_r^1 \times I_s^2$ by a single "approximate" state with the smallest t.

Theorem 14 (Kacem [9]) *Given an arbitrary* $\varepsilon > 0$, *algorithm* A'_ε *can be implemented in* $O(n^3/\varepsilon^2)$ *time and it yields an output* $\varphi_{A'_\varepsilon}(\mathcal{P})$ such that: $\varphi_{A'_\varepsilon}(\mathcal{P}) / \varphi^*(\mathcal{P}) \leq 1 + \varepsilon$.

From Theorem 14, algorithm A'_ε is an FPTAS for the unweighted version of the problem.

3.2 The weighted case

In this section, we consider the weighted case of the problem, i.e., for every job i, we have an arbitrary w_i. Jobs are indexed in non-decreasing order of p_i/w_i.

In this case, we can easily remark the following property.

Proposition 3 (Kacem [9]) *If* $\sum_{i=1}^{n} p_i \leq 2T_1$, *then problem* $(\mathcal{P})$ *has an FPTAS.*

Based on the result of Proposition 3, we only consider the case where $\sum_{i=1}^{n} p_i > 2T_1$.

3.2.1 Dynamic programming

The problem can be optimally solved by applying the following dynamic programming algorithm *AW*, which is a weak extended version of the one proposed by Lee and Liman [19]. This algorithm generates iteratively some sets of states. At every iteration k, a set $\mathcal{U}_k$ composed of states is generated ($1 \leq k \leq n$). Each state $[t, p, f]$ in $\mathcal{U}_k$ can be associated to a feasible schedule for the first k jobs. Variable t denotes the completion time of the last job scheduled before T_1 on the first machine, p is the completion time of the last job scheduled on the second machine and f is the total weighted flow-time of the corresponding schedule. This algorithm can be described as follows:

Algorithm *AW*

i. Set $\mathcal{U}_1 = \{[0, p_1, w_1 p_1], [p_1, 0, w_1 p_1]\}$.

ii. For $k \in \{2, 3, ..., n\}$,

For every state $[t, p, f]$ in $\mathcal{U}_{k-1}$:

1) Put $[t, p+p_k, f+w_k(p+p_k)]$ in $\mathcal{U}_k$
2) Put $[t+p_k, p, f+w_k(t+p_k)]$ in $\mathcal{U}_k$ if $t + p_k \leq T_1$
Remove $\mathcal{U}_{k-1}$

iii. $\varphi^*(\mathcal{P}) = \min_{[t,p,f]\in\mathcal{U}_n}\{f\}$.

Let *UB'* be an upper bound on the optimal weighted flow-time for problem ($\mathcal{P}$). If we add the restriction that for every state [*t*, *p*, *f*] the relation $f \leq UB'$ must hold, then the running time of *AW* can be bounded by nPT_1UB' (where *P* denotes the sum of processing times). Indeed, *t*, *p* and *f* are integers and at each iteration *k*, we have to create at most PT_1UB' states to construct $\mathcal{U}_k$. Moreover, the complexity of *AW* is proportional to $\sum_{k=1}^{n}|\mathcal{U}_k|$.
However, this complexity can be reduced to $O(nT_1)$ by choosing at each iteration *k* and for every *t* the state [*t*, *p*, *f*] with the smallest value of *f*.
In the remainder of the paper, algorithm *AW* denotes the weak version of this dynamic programming algorithm by taking $UB' = \varphi_{HW}(\mathcal{P})$, where *HW* is the heuristic described later in the next subsection.

3.2.2 FPTAS (Kacem [9])

Our FPTAS is based on two steps. First, we use the heuristic *HW*. Then, we apply a modified dynamic programming algorithm.
The heuristic *HW* is very simple! We schedule all the jobs on the second machine in the WSPT order. It may appear that this heuristic is bad, however, the following Lemma shows that it has a worst-case performance ratio less than 2. Note also that it can be implemented in $O(n \log(n))$ time.

Lemma 1 (Kacem [9]) *Let* $\rho(HW)$ *denote the worst-case performance ratio of heuristic HW. Therefore, the following relation holds*: $\rho(HW) \leq 2$.

From Lemma 3, we can deduce that any heuristic for the problem has a worst-case performance bound less than 2 since it is better than *HW*.
In the second step of our FPTAS, we modify the execution of algorithm *AW* in order to reduce the running time. The main idea is similar to the one used for the unweighted case (i.e., modifying the execution of an exact algorithm to design FPTAS). In particular, we follow the splitting technique by Woeginger [28] to convert *AW* in an FPTAS.
Using a similar notation to [28] and given an arbitrary $\varepsilon > 0$, we define $\Delta = 1 + \frac{\varepsilon}{2n}$, $L_1 = \lceil \ln(T_1)/\ln(\Delta) \rceil$, $L_2 = \lceil \ln(P)/\ln(\Delta) \rceil$ and $L_3 = \lceil \ln(\varphi_{HW}(\mathcal{P}))/\ln(\Delta) \rceil$.
First, we remark that every state $[t, p, f] \in \mathcal{U}_k$ verifies

$$[t, p, f] \in [0, T_1] \times [0, P] \times [1, \varphi_{HW}(\mathcal{P})].$$

Then, we split the interval $[0,T_1]$ into L_1+1 subintervals I_r^1 ($I_0^1 = [0,1[$ and $I_r^1 = [\Delta^{r-1}, \Delta^r]_{1\leq r\leq L_1}$).
We also split the intervals [0, *P*] and $[1, \varphi_{HW}(\mathcal{P})]$ respectively, into L_2+1 subintervals I_s^2 ($I_0^2 = [0,1[$ and $I_s^2 = [\Delta^{s-1}, \Delta^s]_{1\leq s\leq L_2}$) and into L_3 subintervals $I_l^3 = [\Delta^{l-1}, \Delta^l]_{1\leq l\leq L_3}$.
Our algorithm AW'_ε generates reduced sets $\mathcal{U}_k^{\#}$ instead of sets $\mathcal{U}_k$. This algorithm can be described as follows:

Algorithm *AW'*

i. Set $\mathcal{U}_1^{\#} = \{[0, p_1, w_1p_1], [p_1, 0, w_1p_1]\}$
ii. For $k \in \{2, 3, \ldots, n\}$,

For every state $[t, p, f]$ in $\mathcal{U}_{k-1}^{\#}$
1) Put $[t, p + p_k, f + w_k(p + p_k)]$ in $\mathcal{U}_k^{\#}$
2) Put $[t + p_k, p, f + w_k(t + p_k)]$ in $\mathcal{U}_k^{\#}$ if $t + p_k \leq T_1$
Remove $\mathcal{U}_{k-1}^{\#}$
Let $[t, p, f]_{r,s,l}$ be the state in $\mathcal{U}_k^{\#}$ such that $t \in I_r^1$, $p \in I_s^2$ and $f \in I_l^3$ with the smallest possible t (ties are broken by choosing the state of the smallest f).
Set $\mathcal{U}_k^{\#} = \{[t, p, f]_{r,s,l} \mid 0 \leq r \leq L_1,\ 0 \leq s \leq L_2,\ 1 \leq l \leq L_3\}$.

iii. $\varphi_{AW'_\varepsilon}(\mathcal{P}) = \min_{[t,p,f] \in \mathcal{U}_n^{\#}} \{f\}$.

3.2.3 Worst-case analysis and complexity

The worst-case analysis of the FPTAS is based on the comparison of the execution of algorithms *AW* and *AW'*ε. In particular, we focus on the comparison of the states generated by each of the two algorithms.

Theorem 15 (Kacem [9]) *Given an arbitrary* ε > 0, *algorithm AW'*ε *yields an output* $\varphi_{AW'_\varepsilon}(\mathcal{P})$ *such that:* $\varphi_{AW'_\varepsilon}(\mathcal{P}) - \varphi^*(\mathcal{P}) \leq \varepsilon\varphi^*(\mathcal{P})$ *and it can be implemented in* $O(|I|^3\, n^3/\varepsilon^3)$ *time, where* $|I|$ *is the input size of I.*

From Theorem 15, algorithm *AW'*ε an FPTAS for the weighted version of the problem.

4. Conclusion

In this chapter, we considered the non-resumable version of scheduling problems under availability constraint. We addressed the criterion of the weighted sum of the completion times. We presented the main works related to these problems. This presentation shows that some problems can be efficiently solved (as an example, some proposed FPTAS have a strongly polynomial running time). As future works, the idea to extend these results to other variants of problems is very interesting. The development of better approximation algorithms is also a challenging subject.

5. Acknowledgement

This work is supported in part by the Conseil Général Champagne-Ardenne, France (Project OCIDI, grant UB902 / CR20122 / 289E).

6. References

Adiri, I., Bruno, J., Frostig, E., Rinnooy Kan, A.H.G., 1989. Single-machine flow-time scheduling with a single breakdown. *Acta Informatica* 26, 679-696. [1]

Belouadah, H., Posner, M.E., Potts, C.N., 1992. Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics* 36, 213- 231. [2]

Breit, J., 2006. Improved approximation for non-preemptive single machine flow-time scheduling with an availability constraint. *European Journal of Operational Research*, doi:10.1016/j.ejor.2006.10.005 [3]

Chen, W.J., 2006. Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. *Journal of the Operational Research Society* 57, 410-415. [4]

Eastman, W. L., Even, S., Issacs, I. M., 1964. Bounds for the optimal scheduling of n jobs on m processors. *Management Science* 11, 268-279. [5]

Gens, G.V., Levner, E.V., 1981. Fast approximation algorithms for job sequencing with deadlines. *Discrete Applied Mathematics* 3, 313—318. [6]

Ibarra, O., Kim, C.E., 1975. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM* 22, 463—468. [7]

Kacem, I., 2007. Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. *Journal of Combinatorial Optimization*, doi : 10.1007/s10878-007-9102-4. [8]

Kacem, I., 2007. Fully Polynomial-Time Approximation Schemes for the Flowtime Minimization Under Unavailability Constraint. *Workshop Logistique et Transport*, 18-20 November 2007, Sousse, Tunisia. [9]

Kacem, I., 2007. Approximation algorithm for the weighted flowtime minimization on a single machine with a fixed non-availability interval. *Computers & Industrial Engineering*, doi: 10.1016/j.cie.2007.08.005. [10]

Kacem, I., Chu, C., 2006. Worst-case analysis of the WSPT and MWSPT rules for single machine scheduling with one planned setup period. *European Journal of Operational Research,* doi:10.1016/j.ejor.2006.06.062. [11]

Kacem, I., Chu, C., Souissi, A., 2008. Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times. *Computers & Operations Research*, vol 35, n°3, 827 − 844, doi:10.1016/j.cor.2006.04.010. [12]

Kacem, I., Chu, C., 2007. Efficient branch-and-bound algorithm for minimizing the weighted sum of completion times on a single machine with one availability constraint. *International Journal of Production Economics*, 10.1016/j.ijpe.2007.01.013. [13]

Kellerer, H., Strusevich, V.A., Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Working Paper*, Submitted. [14]

Kovalyov, M.Y., Kubiak, W., 1999. A fully polynomial approximation scheme for weighted earliness-tardiness problem. *Operations Research* 47: 757-761. [15]

Lee, C.Y., 1996. Machine scheduling with an availability constraints. *Journal of Global Optimization* 9, 363-384. [16]

Lee, C.Y., 2004. Machine scheduling with an availability constraint. In: Leung JYT (Ed), *Handbook of scheduling: Algorithms, Models, and Performance Analysis*. USA, FL, Boca Raton, chapter 22. [17]

Lee, C.Y., Liman, S.D., 1992. Single machine flow-time scheduling with scheduled maitenance. *Acta Informatica* 29, 375-382. [18]

Lee, C.Y., Liman, S.D., 1993. Capacitated two-parallel machines sceduling to minimize sum of job completion times. *Discrete Applied Mathematics* 41, 211-222. [19]

Qi, X., Chen, T., Tu, F., 1999. Scheduling the maintenance on a single machine. *Journal of the Operational Research Society* 50, 1071-1078. [20]

Sadfi, C., Penz, B., Rapine, C., Blaÿzewicz, J., Formanowicz, P., 2005. An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. *European Journal of Operational Research* 161, 3-10. [21]

Sadfi, C., Aroua, M.-D., Penz, B. 2004. Single machine total completion time scheduling problem with availability constraints. *9th International Workshop on Project Management and Scheduling* (PMS'2004), 26-28 April 2004, Nancy, France. [22]

Sahni, S., 1976. Algorithms for scheduling independent tasks. *Journal of the ACM* 23, 116—127. [23]

Schmidt, G., 2000. Scheduling with limited machine availability. *European Journal of Operational Research* 121, 1-15. [24]

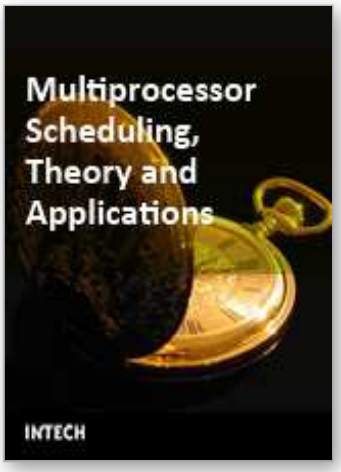
Smith, W.E., 1956. Various optimizers for single stage production. *Naval Research Logistics Quarterly* 3, 59-66. [25]

Wang, G., Sun, H., Chu, C., 2005. Preemptive scheduling with availability constraints to minimize total weighted completion times. *Annals of Operations Research* 133, 183-192. [26]

Webster, S.,Weighted flow time bounds for scheduling identical processors. *European Journal of Operational Research* 80, 103-111. [27]

Woeginger, G.J., 2000. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS) ?. *INFORMS Journal on Computing* 12, 57—75. [28]

Woeginger, G.J., 2005. A comment on scheduling two machines with capacity constraints. *Discrete Optimization* 2, 269—272. [29]



Multiprocessor Scheduling, Theory and Applications

Edited by Eugene Levner

ISBN 978-3-902613-02-8

Hard cover, 436 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

A major goal of the book is to continue a good tradition - to bring together reputable researchers from different countries in order to provide a comprehensive coverage of advanced and modern topics in scheduling not yet reflected by other books. The virtual consortium of the authors has been created by using electronic exchanges; it comprises 50 authors from 18 different countries who have submitted 23 contributions to this collective product. In this sense, the volume can be added to a bookshelf with similar collective publications in scheduling, started by Coffman (1976) and successfully continued by Chretienne et al. (1995), Gutin and Punnen (2002), and Leung (2004). This volume contains four major parts that cover the following directions: the state of the art in theory and algorithms for classical and non-standard scheduling problems; new exact optimization algorithms, approximation algorithms with performance guarantees, heuristics and metaheuristics; novel models and approaches to scheduling; and, last but least, several real-life applications and case studies.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821