

An Approximation Algorithm for the UET Two-Machine Open-Shop Problem with Time Delays

Alix Munier-Kordon

Laboratoire LIP6, Université Pierre et Marie Curie, 4 place Jussieu, 75 252, Paris Cedex 05, France,
Alix.Munier@lip6.fr

Djamal Rebaine

Département d'informatique et de mathématique, Université du Québec à Chicoutimi, 555, Boul. de l'Université,
Québec, Canada G7H 2B1, drebaine@uqac.ca

This paper addresses the problem of scheduling n unit execution time (UET) jobs with time delays considerations on a two-machine open-shop environment. The criterion we are considering is the makespan. A simple approximation algorithm, with an asymptotic performance guarantee of $\frac{5}{4}$, is presented and proved.

Keywords: Makespan, Open-shop, Theoretical Scheduling, Time delays.

1 Introduction

This paper addresses the problem of scheduling n unit execution times (UET) jobs with time delays considerations on a two-machine open-shop environment. Each job comprises two operations. One operation has to be processed by one machine and the other by the second machine. As usual, we assume that the operations of a job cannot be processed at the same time, and a machine can only process at most one operation at a time.

In this study, we suppose that a time delay, τ_i , is associated with each job $i \in J$ to denote the minimum time which must elapse between the completion of one of its operation and the start of the other operation. In other words, if we denote by S_i and C_i , respectively, the start time of the second operation and the finish time of the first operation of job i , then a schedule to be valid must be such that $S_i \geq C_i + \tau_i$, for $i = 1, \dots, n$. The purpose is to find a valid schedule which minimises the overall completion time criterion, known as the makespan.

Motivation for the formulated problem comes from real applications. In the traditional shop scheduling, it is a common practice to assume that once a job has finished one of its operation, it becomes immediately available for further processing. However, in many applications, this assumption is not justified. Indeed, there is often a significant time delay between the completion of an operation and the beginning of the next operation of the same job. In addition to delineating the borderline between polynomiality and intractability, in some cases, the execution times might even be negligible compared to the time delays: assuming in this case that the processing times of the jobs are unitary, and therefore have a small influence on the makespan of the schedule. Time delays may be attributed, for example, to transportation times of the jobs on the machines or, in some other applications, to the different times needed by the drying processes of the jobs before they can be handled by another processing stage.

Open-shop problems with time delays may be used to model for example the timetable problems in which students have to meet each of their professors. A time delay refers in this case to the time taken by a student to go from one professor to another. More real world applications may be found in Gonzalez [1] and Lopez and Roubellat [3].

The open-shop scheduling problem with time delays was first introduced in Rayward-Smith and Rebaine [4], and shown that the corresponding two-machine problem is \mathcal{NP} -hard even for identical time delays. A further result is given in Yu [2] as the unary \mathcal{NP} -hardness is shown for the UET case. Therefore, looking for well solvable cases and heuristic algorithms is well justified.

The search for well solvable cases implies the search for special values of time delays for which the corresponding problem can be solved to optimality in polynomial time. Whereas, in the heuristic approach, we usually seek algorithms for which we can evaluate their worst case performance by measuring the distance between the value of the solution it generates and the optimal value. In the case of two machines and arbitrary processing times, Strusevich [6] developed an $O(n \log n)$ $3/2$ -approximation algorithm. In Rebaine and Strusevich [5], an $O(n)$ $4/3$ -approximation algorithm is presented for the case of two values of time delays: a time delay for the set of jobs going from the first to the second machine, and another time delay for the other direction. When the time delays are smaller than the processing times, then it is shown in the latter paper that the corresponding problem is solvable in linear time.

This paper is devoted to the development of an approximation algorithm for the UET two-machine open-shop with arbitrary integral time delays. It is organized as follows. In Section 2, we present preliminary results. Section 3 presents the Sorting Algorithm, and a special case for which it is optimal. The proof of the $5/4$ -asymptotic performance guarantee of this algorithm is presented in Section 4. Section 5 is our conclusion.

2 Preliminary results

In this section, we present results needed for the next sections. First, we introduce the following definition.

Definition 1. *For each job, $j \in J$, a valid schedule is going to process one operation earlier than the other - such an operation is called a first operation. The remaining operation is a second operation.*

Lemma 1. *There exists an optimal schedule in which the completion time of any first operation on each machine is not greater than that of any second operation*

Proof. Assume that a machine processes the second operation of job k immediately before the first operation of job j . These can be interchanged still obtaining a valid schedule. Repeating this argument establishes the lemma. \square

Lemma 2. *There exists an optimal schedule in which the first operations and the second operations on each machine are processed continuously.*

Proof. Let us consider an optimal schedule satisfying Lemma 1. Assume that there exists an idle time between two consecutive first operations i and j , in that order. It is clear that j can be shifted to the left so as there is no idle time between i and j without increasing the makespan. Similarly, assume that there exists an idle time between two consecutive second operations i and j , in that order. It is clear that i can be shifted to the right so as there is no idle time between i and j without increasing the makespan. Repeating this argument establishes the lemma. \square

Lemma 3. *The makespan of an optimal schedule, ω_{opt} , satisfies the following lower bound*

$$\omega_{opt} \geq \left\lceil \frac{\sum_{i=1}^n \tau_i}{n} \right\rceil + \left\lceil \frac{n}{2} \right\rceil.$$

Proof. Without loss of generality, we may assume that an optimal schedule satisfies Lemmas 1 and 2. Let T_1 , with cardinality n_1 (respectively T_2 , with cardinality n_2), be the set of jobs processed first on machine 1 (respectively 2) and then on machine 2 (respectively 1) in an optimal schedule. Let us first observe that $n_1 + n_2 = n$. Let also σ_1 (respectively σ_2) and π_1 (respectively π_2) be the job processing sequences of T_1 (respectively T_2) on machine 1 (respectively 2) and machine 2 (respectively 1), respectively, as illustrated by Figure 1.

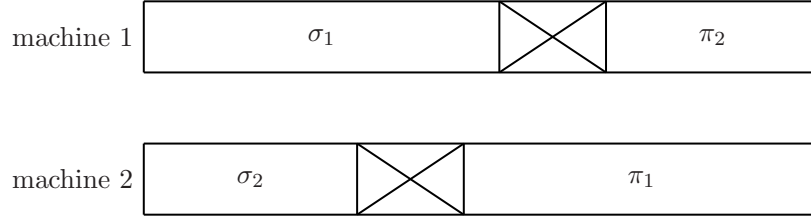


Figure 1: The permutations $\sigma_1, \sigma_2, \pi_1$ and π_2

To be valid, this schedule must be such that

$$\omega_{opt} - n_1 + \pi_1(j) - \sigma_1(j) \geq \tau_j + 1; \text{ for all } j \in T_1. \quad (1)$$

If we consider T_2 then by symmetry we get the following inequality:

$$\omega_{opt} - n_2 + \pi_2(j) - \sigma_2(j) \geq \tau_j + 1; \text{ for all } j \in T_2. \quad (2)$$

Now, if we add up the n_1 inequalities of (1), we get the new following inequality:

$$n_1\omega_{opt} - n_1^2 + \sum_{j \in T_1} \pi_1(j) - \sum_{j \in T_1} \sigma_1(j) \geq \sum_{j \in T_1} \tau_j + n_1. \quad (3)$$

Similarly, if we add up the n_2 inequalities of (2), we end up with the new following inequality:

$$n_2\omega_{opt} - n_2^2 + \sum_{j \in T_2} \pi_2(j) - \sum_{j \in T_2} \sigma_2(j) \geq \sum_{j \in T_2} \tau_j + n_2. \quad (4)$$

As $\pi_1(j)$ and $\sigma_1(j)$ for T_1 and $\pi_2(j)$ and $\sigma_2(j)$ for T_2 are permutations whose values are in $\{1, \dots, n_1\}$ and $\{1, \dots, n_2\}$, respectively, it follows that

$$\begin{aligned} \sum_{j \in T_1} \pi_1(j) - \sum_{j \in T_1} \sigma_1(j) &= 0, \\ \sum_{j \in T_2} \pi_2(j) - \sum_{j \in T_2} \sigma_2(j) &= 0. \end{aligned}$$

Now, if we add up inequalities of (3) with inequalities of (4), then we obtain the following:

$$n\omega_{opt} \geq \sum_{i=1}^n \tau_i + n + n_1^2 + n_2^2.$$

As $n_1 = n - n_2$, it is easy to see that $n_1^2 + n_2^2$ has its minimum at $n_1 = \frac{n}{2}$. Therefore,

$$\omega_{opt} \geq \frac{\sum_{i=1}^n \tau_i}{n} + 1 + \frac{n}{2}.$$

It then follows that

$$\begin{aligned} \omega_{opt} &\geq \left\lceil \frac{\sum_{i=1}^n \tau_i}{n} + 1 + \frac{n}{2} \right\rceil \\ &\geq \left\lceil \frac{\sum_{i=1}^n \tau_i}{n} \right\rceil + \left\lceil \frac{n}{2} \right\rceil. \end{aligned}$$

Thus the result. □

Lemma 4. *The makespan of an optimal schedule ω_{opt} , satisfies the following lower bound.*

$$\omega_{opt} \geq n.$$

Proof. As each of the n UET jobs is processed by each machine, the lemma follows immediately. □

3 Presentation of the Sorting Algorithm

In this section, we present a polynomial time algorithm, the Sorting Algorithm, that solves approximately the UET two-machine open-shop with time delays. We also present a simple case for which the Sorting Algorithm generates an optimal solution.

The Sorting Algorithm can be described as follows. First, the jobs are sorted in non-increasing order of the time delays. Then, the first operations are processed using a list scheduling algorithm starting with machine 1. Next, the second operations are executed using again a list scheduling algorithm. The running time of this algorithm is clearly dominated by the sorting procedure which can be implemented in $O(n \log n)$.

Sorting Algorithm

1. rename the jobs such that $\tau_1 \geq \tau_2 \geq \dots \geq \tau_n$;
2. for ($i = 1; i \leq n; i++$)
 schedule job i on the first available machine starting with machine 1;
3. for ($i = 1; i \leq n; i++$)
 schedule job i as soon as possible on the corresponding machine.

For example, let us consider an instance of the problem defined by $n = 8$ jobs with $\tau_1 = \tau_2 = 7$, $\tau_3 = \tau_4 = 6$, $\tau_5 = 4$, $\tau_6 = 3$ and $\tau_7 = \tau_8 = 1$. The schedule obtained by the Sorting Algorithm is pictured by Figure 2.

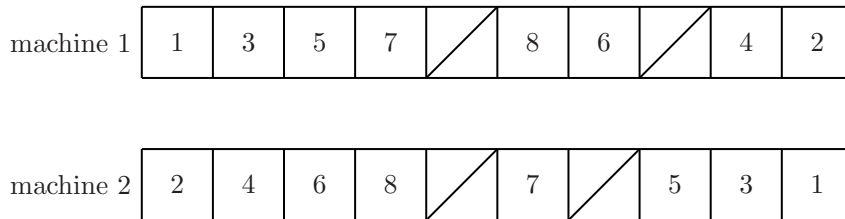


Figure 2: A schedule built by the Sorting Algorithm

Theorem 1. *If the time delays are distinct, then the Sorting Algorithm generates an optimal solution.*

Proof. Let us first observe that the shape of the schedule generated by the Sorting Algorithm satisfies the shape of the schedule of Lemma 1. Now, consider the first operation of two consecutive jobs i and j , in this order, on one of the two machines. We have that $\tau_i \geq \tau_j$. Moreover, as the time delays are all distinct, then clearly we have that $\tau_i \geq \tau_j - 2$. It follows that the start time of the second operation of job i is strictly greater than the start time of the second operation of job j . Therefore, as job 1 is processed first on machine 1, then the makespan of the schedule generated by the Sorting algorithm is clearly $C_{\max} = \tau_1 + 2$, which is a simple lower bound on the makespan. Thus the result. \square

4 Performance ratio of the Sorting Algorithm

In this section, we present the worst-case performance of the Sorting Algorithm. To do so, we first discuss two special cases, then we proceed with the general case.

4.1 First special case

Let r be a positive integer such that $r \geq \lceil \frac{n}{2} \rceil$. We suppose here that I_1 is an instance of the UET two-machine open-shop problem such that, using the Sorting Algorithm, the second operations are available at time r . More formally, for any $i \in \{1, \dots, n\}$, we have that $\tau_i = r - \lceil \frac{i}{2} \rceil$. Observe that the makespan of the schedule obtained by the Sorting Algorithm is $\omega_A(I_1) = r + \lceil \frac{n}{2} \rceil$. Let us first proceed with the following technical result.

Lemma 5.

$$\sum_{j=1}^n \left\lceil \frac{j}{2} \right\rceil = \left\lceil \frac{n}{2} \right\rceil \left(n - \left\lceil \frac{n}{2} \right\rceil + 1 \right).$$

Proof. First, observe that

$$\sum_{j=1}^n \left\lceil \frac{j}{2} \right\rceil = 2 \sum_{\alpha=1}^{\lceil \frac{n}{2} \rceil} \alpha + \left\lceil \frac{n}{2} \right\rceil \left(n - 2 \left\lceil \frac{n}{2} \right\rceil \right).$$

Therefore,

$$\begin{aligned} \sum_{j=1}^n \left\lceil \frac{j}{2} \right\rceil &= \left\lceil \frac{n}{2} \right\rceil \left(\left\lceil \frac{n}{2} \right\rceil + 1 \right) + \left\lceil \frac{n}{2} \right\rceil \left(n - 2 \left\lceil \frac{n}{2} \right\rceil \right) \\ &= \left\lceil \frac{n}{2} \right\rceil \left(n - \left\lceil \frac{n}{2} \right\rceil + 1 \right). \end{aligned}$$

Thus the result. \square

Lemma 6.

$$\frac{\sum_{j=1}^n \tau_j}{n} \geq r - \frac{3}{2} - \frac{n}{4}.$$

Proof. By definition, we have that

$$\sum_{i=1}^n \tau_i = nr - \sum_{i=1}^n \left\lceil \frac{i}{2} \right\rceil.$$

Now, if we use Lemma 5, then we get the following:

$$\begin{aligned} \sum_{i=1}^n \tau_i &= nr - \left\lceil \frac{n}{2} \right\rceil \left(n - \left\lceil \frac{n}{2} \right\rceil + 1 \right) \\ &\geq nr - n \left(\frac{n+1}{2} \right) + \left(\frac{n}{2} \right)^2 - \left(\frac{n+1}{2} \right). \end{aligned}$$

Therefore,

$$\begin{aligned} \sum_{i=1}^n \tau_i &\geq nr - \frac{(n+1)^2}{2} + \left(\frac{n}{2} \right)^2 \\ &= nr - \frac{n^2}{4} - n - \frac{1}{2}. \end{aligned}$$

Since $n \geq 1$, then the result follows. □

Theorem 2.

$$\frac{\omega_A(I_1)}{\omega_{opt}(I_1)} \leq \frac{5}{4} + \frac{3}{2n}.$$

Proof. From Lemmas 3 and 6, we deduce that

$$\begin{aligned} \omega_{opt}(I_1) &\geq r - \frac{3}{2} - \frac{n}{4} + \left\lceil \frac{n}{2} \right\rceil \\ &= \omega_A(I_1) - \frac{3}{2} - \frac{n}{4}. \end{aligned}$$

From Lemma 4, it follows that

$$\omega_A(I_1) \leq \frac{5}{4} \omega_{opt}(I_1) + \frac{3}{2}.$$

Thus the result. □

We now show that the above bound is tight. Let k be a strictly positive integer. Consider an instance of the problem defined by $n = 4k + 2$ jobs with, for any job $i \in \{1, \dots, n\}$, $\tau_i = 3k - \lceil \frac{i}{2} \rceil$. The length of the schedule obtained by the Sorting Algorithm is clearly $\omega_A = 5k + 1$. Now, an optimal schedule with no idle time slots may be built by processing the first operation of every odd (respectively even) job by machine 1 (respectively 2). Jobs with odd delays are processed first on both machines in decreasing order followed by those with even delays in decreasing order. An optimal schedule for $k = 3$ is pictured by Figure 3. This gives a schedule of length $\omega_{opt} = 4k + 2$. Thus the tightness of the above bound.

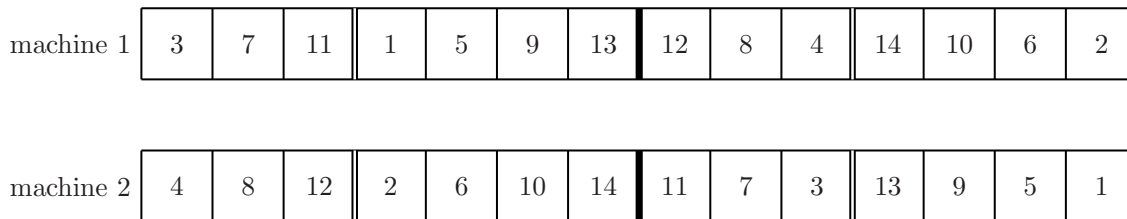


Figure 3: An optimal schedule for the first special case with $k = 3$

4.2 Second special case

We now consider a second instance I_2 of the above problem such that, using the Sorting Algorithm, the second operation of jobs processed first by machine 1 (respectively 2) are available at time r (respectively $r - 1$). More formally, for any $i \in \{1, \dots, n\}$, we set $\tau_i = r - \lceil \frac{i+1}{2} \rceil$. We observe that $\omega_A(I_2) = r + \lceil \frac{n}{2} \rceil$. Proofs of Lemma 7 and Theorem 3 are similar to the previous case and are thus omitted.

Lemma 7.

$$\frac{\sum_{j=1}^n \tau_j}{n} \geq r - \frac{11}{4} - \frac{n}{4}.$$

Theorem 3.

$$\frac{\omega_A(I_2)}{\omega_{opt}(I_2)} \leq \frac{5}{4} + \frac{11}{4n}.$$

4.3 General case

Let I be an instance of the UET two-machine open-shop with $\omega_A(I) > n$. Let V_1 (respectively V_2) be jobs from I for which the first (respectively second) operations are processed by machine 1 (respectively 2) according to the Sorting Algorithm. Let us define the release date, r_i , of the second operation of job $i \in \{1, \dots, n\}$ by $r_i = \tau_i + \lceil \frac{i}{2} \rceil$.

Let r^* be the smallest integer such that there is no idle time slots in the interval $[r^*, \omega_A(I)]$. If there is an idle time slot at time $r^* - 1$ on machine 1, we set $k^* = 2$, otherwise $k^* = 1$.

Lemma 8. *If $M = \{j \in V_{k^*}, r_j \geq r^*\}$ then $\omega_A(I) = r^* + |M|$.*

Proof. The second operation of jobs from M are available after time r^* . Moreover, since there exists an idle time slot at time $r^* - 1$, every job $j \in V_{k^*}$ with $r_j < r^*$ is completed at time $r^* - 1$. Thus the result. \square

Let $t_1, \dots, t_k, k = |M|$, be the successive completion times of the first operations of jobs from M . For an illustrative purpose, let us go back to the example pictured by Figure 2. We have that $r^* = 8$, $M = \{2, 4\}$, and $k^* = 2$. The completion times are $t_1 = 1$ and $t_2 = 2$.

Lemma 9. *For any $\alpha \in \{1, \dots, k\}$, $t_\alpha = \alpha$.*

Proof. Let us consider a job $i \in V_{k^*} \setminus M$ whose first operation is processed at time $t_\alpha - 2$ for $\alpha \in \{1, \dots, k\}$, and let $j \in M$ whose first operation is processed at time $t_\alpha - 1$. From the Sorting Algorithm, we have that $\tau_i \geq \tau_j$ and $j = i + 2$. Therefore,

$$\begin{aligned} r_i &= \tau_i + \left\lceil \frac{i}{2} \right\rceil \geq r_j - 1 \\ &\geq r^* - 1. \end{aligned}$$

Since the second operation of job i is not available at time $r^* - 1$, because of the idle time slot, we get $r_i \geq r^*$ and $i \in M$, thus a contradiction. \square

Lemma 10. *If $k^* = 1$, then there exists an instance I_2 of the second special case such that*

$$\frac{\omega_A(I)}{\omega_{opt}(I)} \leq \frac{\omega_A(I_2)}{\omega_{opt}(I_2)}.$$

Proof. From Lemma 9 the Sorting Algorithm generates a schedule in which the jobs in $M = \{1, 3, \dots, 2k - 1\}$ are first processed on machine 1. Then for any job $i \in M$, we have that $\tau_i \geq r^* - \lceil \frac{i}{2} \rceil$.

An instance I_2 of the second special case is built from I by setting $n' = 2k - 1$ and $\tau'_i = r^* - \lceil \frac{i+1}{2} \rceil$ for $i \in \{1, \dots, n'\}$.

We first show that $\omega_A(I) = \omega_A(I_2)$. Indeed, from Lemma 8, we have that $\omega_A(I) = r^* + |M| = r^* + k$. Now, when considering only jobs of I_2 , we have that $r_i = r^*$. As these jobs are scheduled by the Sorting Algorithm, it follows that $\omega_A(I_2) = r^* + k$. Thus the result.

Next, we show that, for any $i \in \{1, \dots, n'\}$, we have that $\tau'_i \leq \tau_i$. Indeed,

1. The release date r_i for $i \in M$ verifies $r_i \geq r^*$. So, for any $i \in M$, $\tau'_i \leq \tau_i$.
2. Every job $j \in \{1, \dots, n'\} \setminus M$ verifies $\tau_{j+1} \leq \tau_j$ and $\tau'_j = \tau'_{j+1}$. Since $j + 1 \in M$, we get $\tau'_{j+1} \leq \tau_{j+1}$. We then conclude that $\tau'_j = \tau'_{j+1} \leq \tau_{j+1} \leq \tau_j$.

Therefore, we obtain that $\omega_A(I) = \omega_A(I_2)$ and $\omega_{opt}(I) \geq \omega_{opt}(I_2)$. Thus the result. □

Lemma 11. *If $k^* = 2$, then there exists an instance I_1 of the first special case such that*

$$\frac{\omega_A(I)}{\omega_{opt}(I)} \leq \frac{\omega_A(I_1)}{\omega_{opt}(I_1)}.$$

Proof. From Lemma 9 the Sorting Algorithm generates a schedule in which the jobs in $M = \{2, 4, \dots, 2k\}$ are first processed on machine 2. Then, for any job $i \in M$, we have that $\tau_i \geq r^* - \lceil \frac{i}{2} \rceil$.

An instance I_1 of the first special case is built from I by setting $n' = 2k$ and $\tau'_i = r^* - \lceil \frac{i}{2} \rceil$ for $i \in \{1, \dots, n'\}$.

We first show that $\omega_A(I) = \omega_A(I_1)$. Indeed, from Lemma 8, we have that $\omega_A(I) = r^* + |M| = r^* + k$. Now, when considering only jobs of I_1 , we have that $r_i = r^*$. As these jobs are scheduled by the Sorting Algorithm, it follows that $\omega_A(I_1) = r^* + k$. Thus the result.

Next, we show that, for any $i \in \{1, \dots, n'\}$, we have that $\tau'_i \leq \tau_i$. Indeed,

1. The release date r_i for $i \in M$ verifies $r_i \geq r^*$. So, for any $i \in M$, $\tau'_i \leq \tau_i$.
2. Every job $j \in \{1, \dots, n'\} \setminus M$ verifies $\tau_j \geq \tau_{j+1}$. But $\tau_{j+1} \geq \tau'_{j+1} = \tau'_j$. Therefore, $\tau_j \geq \tau'_j$.

Therefore, we obtain that $\omega_A(I) = \omega_A(I_1)$ and $\omega_{opt}(I) \geq \omega_{opt}(I_1)$. Thus the result. □

From Theorems 2 and 3, and Lemmas 10 and 11, we deduce the following main result.

Theorem 4. *The performance ratio of the Sorting Algorithm is asymptotically bounded by $\frac{5}{4}$. Moreover, this bound is tight.*

5 Conclusion

In this paper, we considered the unit execution time (UET) two-machine open-shop with integral time delays considerations so as to minimise the makespan. We presented a simple approximation algorithm with an asymptotic performance guarantee of $\frac{5}{4}$. We also showed that, when the time delays are all distinct, then the Sorting Algorithm produces an optimal schedule. A further research would be to exhibit other well solvable cases and heuristics with better worst case bounds.

Acknowledgment

This research is partially funded for Djamel Rebaine by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

- [1] T. Gonzalez (2005), Open shop scheduling, in J.Y.T. Leung (ed): *Handbook of scheduling*, Chapman & Hall/CRC, chapter 6, 1 – 14.
- [2] W. Yu, H. Hoogeveen, J.K. Lenstra (2004), Minimizing makespan in a two-machine flow with delays and unit-time operations is NP-hard, *Journal of Scheduling*, 333 – 348.
- [3] P. Lopez and F. Roubellat (2001), *Ordonnancement de la production*, Chapter 11, Hermes Science Publications.
- [4] V.J. Rayward-Smith, D. Rebaine (1992), Open shop scheduling with delays, *Theoretical Informatics and Application*, 439 – 448.
- [5] D. Rebaine, V.A. Strusevich (1999), Two-machine open shop scheduling with special transportation times, *Journal of the Operations Research Society* **50**(7), 756 – 764.
- [6] V.A. Strusevich (1999), A heuristic for the two-machine open shop scheduling problem with transportation times, *Discrete Applied Mathematics* **2-3**, 287 – 304.