TWO-MACHINE FLOW SHOP SCHEDULING PROBLEM WITH TWO COMPETING AGENTS

Abdennour AZERINE

RECITS laboratory,
Faculty of Mathematics,
USTHB University,
BP 32 El-Alia, Bab Ezzouar,
Algiers, Algeria.
Centre de Recherche
sur l'Information Scientifique et
Technique (CERIST)
05, Rue des 3 frères aissou,
Ben Aknoun Algiers Province,
Algeria

Mourad BOUDHAR

RECITS laboratory,
Faculty of Mathematics,
USTHB University,
BP 32 El-Alia, Bab Ezzouar,
Algiers, Algeria.

Djamal REBAINE

Département d'informatique et de mathématique,
Université du Québec à Chicoutimi, Qubec, Canada

ABSTRACT

In this article, we study the two-machine flow shop scheduling with two competing agents. The complexity of special cases is investigated with respect to the makespan and the total completion time. The second part concerns the minimization of the linear combination of the total tardiness and the number of tardy jobs. We presented a branch-and-bound algorithm along with a heuristic and a meta-heuristic. A computational analysis is then conducted.

1. INTRODUCTION

The usual flow shop problem can be described as follows, given n jobs to be processed on m machines. Each job must be processed first on machine 1, and then on machine 2, and and so on until machine m. Each job has its associated processing time on each machine. It is assumed that, at each instant of time, any machine can process only one job, and a job is processed by at most one machine. Let us observe that it is a common practice in the literature to focus on permutation schedules when studying the flow shop models, even though this assumption of restricting a flow shop to a permutation is only true for regular criteria and $m \leq 3$.

With respect to the makespan criterion, the two-machine flow shop problem has been proved to be polynomially solvable in $O(n \log n)$ by [Johnson(1954)]. The problem becomes strongly $\mathscr{N}\mathscr{P}$-hard with respect to the total completion time criterion (see [Garey et al.(1976)Garey, Johnson, and Sethi]), so it is also with respect to the total tardiness.

A branch-and-bound algorithm along with several lower bounds and dominance rules has been proposed by [Kim(1993)], which can solve all instances with 16-jobs and more then half instance with 24-jobs. This algorithm is shown that it outperforms the algorithm provided in [Sen et al.(1989)Sen, Dileepan, and Gupia] that can solve instances with only 12 jobs.

For the two-machine flow shop with regard to the total tardiness criterion, a branch-and-bound algorithm with new dominance conditions is developed in [Pan and Fan(1997)] that can solve instances up to 16 jobs in less than one-hour, whereas [Pan et al.(2002)Pan, Chen, and Chao] implemented four dominance rules and a lower bound in a branch-and-bound scheme that can solve instances up to 18 jobs and most instances with 20 jobs withing one-hour.

A few papers dealt with the two-machine flow shop with respect to tardy jobs criterion. Let us recall the corresponding decision problem is strongly $\mathcal{NP}$-hard ([Lenstra et al.(1977)Lenstra, Kan, and Brucker]). [Lawler and Moore(1969)] addressed this problem with a common due date and provided a pseudo-polynomial dynamic programming based algorithm with running time $O(nd^2)$, where d is the value of the common due date. [Hariri and Potts(1989)] considered the m-machine permutation flow shop problem and proposed a branch-and-bound algorithm using lower bounds based on the solution of single machine sub-problems that can solve instances with 20 jobs for the three-machine problem, and up to 15 jobs for the four-machine. When it comes to the two-agent flow shop problem, this problem has received increasing attention in recent years. Before proceeding any further, we briefly define this problem ; more details on this are given in Section 2. In a two-agent scheduling problem, the set of jobs is divided into two subsets to be processed on the same set of machines. Each subset has to be scheduled with respect to a given criterion. Let us now mention the following results. [Agnetis et al.(2004)Agnetis, Mirchandani, Pacciarelli, and Pacifici] studied the complexity of this problem with respect to the makespan criterion for both agents. They provided an $\mathcal{NP}$-hardness result for the ε-constraint model where the objective in this approach is to minimize the criteria of the first agent respecting the constraint that the cost function of the second agent is below or equal a given fixed value. This model has also been investigated by [Luo et al.(2012)Luo, Chen, and Zhang], and proved that it is weakly $\mathcal{NP}$-hard. A second approach, involving the weighted combination of the same two criterias (makespan for both agents) to a single objective function, is also considered in the the same paper. The authors showed that it is weakly $\mathcal{NP}$-hard, they proposed pseudo-polynomial time algorithm that runs in $O(nP^4)$ where P is the sum of the processing times of all the jobs on the two machines and a fully polynomial-time approximation scheme.

In the front of well solvable cases, [Mor and Mosheiov(2014)] extended some polynomial single-machine problems to the proportionate flow shop with two competing agents. For the first agent, they considered three objective functions to be minimized : maximum cost of all the jobs, total completion time, and number of tardy jobs ; with the restriction that the value of the maximum cost function of the second agent does not exceed a given value.

[Ahmadi-Darani et al.(2018)Ahmadi-Darani, Moslehi, and Reisi-Nafchi] studied the two-machine flow shop problem to minimize the total tardiness of jobs for the first agent given that the makespan of the second agent does not exceed a given upper bound. The authors presented a mathematical programming model, a tabu search algorithm and some heuristics. A variable neighborhood search for the two-agent flow shop scheduling problem with the makespan for the first agent and the total tardiness of the second agent is developed in [Lei(2015)].

In this paper, we consider the two-machine flow shop scheduling problem with two competing agents. The first agent aims to minimize the total tardiness, whereas for the second agent. the goal is the minimization of the total number of tardy jobs. We use a linear combination of the two criteria as a single objective function. Let us note that [Lee et al.(2010)Lee, Chen, and Wu] studied this problem using the ε-constraint approach. They considered the total tardiness of the jobs as an objective function for the first agent with zero tardy jobs for the second agent. They proposed a branch-and-bound and simulated annealing algorithms. Their results showed that the branch-and-bound algorithm can solve instances with 12 jobs in small amount of time.

2. PROBLEM FORMULATION

We consider the two-machine flow shop problem with two agents (users) A and B, competing to use a set of shared machines $M = \{M_1, M_2, \ldots, M_m\}$ (In our case $m = 2$). Each agent has his own set of independent jobs J_X where $X = \{A, B\}$. Let J_i^X denote job i of agent X, its processing time on machine M_j is denoted by p_{ij}^X, its due date by d_i^X and its completion time by C_{ij}^X where $i \in J_X$, and $j \in M$.To simplify the notation, we denote in some cases by a_i^X and b_i^X the processing time of job i of agent X on machine M_1 and M_2, respectively, and C_i^X the completion time of job i on the last machine. Let $T_i^X = \max\{0, C_i^X - d_i^X\}$ be the tardiness of job i and $U_i^X = 0$ if $C_i^X - d_i^X \leq 0$, and 1 otherwise. Each agent has his own objective function γ^X to minimize.

The usual way to describe and classify scheduling problems, we use the three-field notation $\alpha|\beta|\gamma$ proposed by [Graham et al.(1979)Graham, Lawler, Lenstra, and Kan]. We suppose that the processing route of each job is the same for the jobs of a given agent and it is known in advance. However, it could differ from one agent to another. To be able to distinguish between them, we provide the processing route for the first agent followed by the processing route of second agent in the β field. As an example for agent A and B, respectively, notation $M_1 \mapsto M_2, M_2 \mapsto M_1$ mean that A processes all his jobs on M_1 and then on M_2, whereas B processes his jobs on M_2 and then on M_1. In the remainder of this paper, we use the following notation to describe the processing route for each agent :

$M_j \mapsto M_k$ **:** The specified agent processes his jobs on M_j and then on M_k.

M_j **:** The specified agent processes his jobs on M_j only.

In the case where the above notation is not specified, this means that all the jobs have the same processing route and each job must be processed on $M1$ and then on $M2$. Our aim is to schedule these jobs non-preemptively to minimize the weighted combination of the two criteria. In the next section, we address the complexity status of the flow shop problem with various settings and objective functions.

3. COMPLEXITY RESULTS

Theorem 1 *$F2|M_1 \mapsto M_2, M_2 \mapsto M_1, p_{ij} = p_i|C_{max}^A + C_{max}^B$ is $\mathscr{N}\mathscr{P}$-hard.*

Corollary 2 *$J2|n_i \leq 2, p_{ij} = p_i|C_{max}^A + C_{max}^B$ is $\mathscr{N}\mathscr{P}$-hard.*

Theorem 3 *$F2|M_1 \mapsto M_2, M_2 \mapsto M_1, p_{ij} = p_i|C_{max}^A + \sum\limits_{i \in J_B} C_i^B$ is $\mathscr{N}\mathscr{P}$-hard.*

Theorem 4 *$F2|M_1 \mapsto M_2, M_2, p_{ij}^A = p_i|C_{max}^A + C_{max}^B$ is $\mathscr{N}\mathscr{P}$-hard.*

Corollary 5 *The two problems $F3|M_1 \mapsto M_2, M_2 \mapsto M_3|C_{max}^A + C_{max}^B$ and $F3|M_1 \mapsto M_2, M_3 \mapsto M_2|C_{max}^A + C_{max}^B$ are $\mathscr{N}\mathscr{P}$-hard even if $p_{ij} = p_i$.*

Theorem 6 *$F2|M_1 \mapsto M_2, M_2, p_{ij}^A = p_i|\sum\limits_i C_i^A + C_{max}^B$ is $\mathscr{N}\mathscr{P}$-hard.*

Theorem 7 *$F2|M_1 \longmapsto M_2, M_1|C_{max}^A + C_{max}^B$ is $\mathscr{N}\mathscr{P}$-Hard.*

Corollary 8 *$F3|M_1 \longmapsto M_2, M_1 \longmapsto M_3|C_{max}^A + C_{max}^B$ and $F3|M_1 \longmapsto M_2, M_3 \longmapsto M_1|C_{max}^A + C_{max}^B$ are $\mathscr{N}\mathscr{P}$-hard.*

Theorem 9 *$F2|M_1 \longmapsto M_2, M_1|C_{max}^A + \sum\limits_i C_i^B$ is $\mathscr{N}\mathscr{P}$-Hard.*

4. BRANCH-AND-BOUND

In the next sections, we will address the two-agent flow shop problem with two different objectives, this problem can be formulated as follows : Two agents A and B compete to process their jobs on two shared machines, starting by M_1 followed M_2. The objective of the first agent is to minimize the total tardiness while the objective of the second agent is to minimize the number of tardy jobs, we will use the linear combination as an approach to find a solution. This problem is donated by : $F2||\alpha \times \sum_{i \in J_A} T_i^A + \beta \times \sum_{i \in J_B} U_i^B$ where $0 \leq \alpha \leq 1$ and $\alpha + \beta = 1$ or $F2||\lambda \times \sum_{i \in J_A} T_i^A + (1-\lambda) \times \sum_{i \in J_B} U_i^B$ where $0 \leq \lambda \leq 1$. In order to find an optimal solution for this problem, we propose an algorithm based on branch-and-bound.

Let us recall that a branch and bound algorithm consists of breaking up the problem under study into successively smaller sub-problems, computing bounds on the objective function associated with each sub-problem, and using them to discard certain of these sub-problems from further consideration. The algorithm ends up when each sub-problem has either produced a feasible solution, or is shown to contain no better solution than the one already in hand. The best solution found at the end of the algorithm is the global optimum.

4.1. Upper Bounds

Given a node in the branching tree with a partial sequence π, π^c, where π is a partial schedule contains k-jobs with a fixed sequence of jobs, and π^c is the unscheduled part with $n-k$. We proposed a heuristic which is used as an upper bound, to find a feasible solution that has two phases. The first phase is to generate a priority list that is constructed using the jobs in π^c, whereas the second phase is used to improve the quality of the solution through a swap operator for the jobs of π^c. The obtained value is used as an upper bound on the value of the node.

The heuristic can be used to produce a feasible solution when $k = 0$, and generate a global upper bound.

Algorithm 1 *1: Heuristic*

```
2:  Create and initial a priority list using the jobs in π^c ;
3:  Evaluate the current list, and let Z be the value of the objective function;
4:  for i ← k+1 to n do
5:      for j ← k+1 to n do
6:          Swap the job in position i with the job in position j;
7:          Evaluate the new solution and let Z_new be the value of the objective function;
8:          if Z < Z_new then
9:              Z ← Z_new;
10:         else
11:             Swap the job in position i with the job in position j;
```

This procedure needs a priority list to be executed. So the first step to be done is the construction of an initial list. In our case, we used five different priority lists, namely $L_1, \ldots, L_5$.

- L_1 **:** Order the jobs according to the non-decreasing order of the due dates.
- L_2 **:** Calculate for each job i the value $d_i - \max\{p_{i1}, p_{i2}\}$. Order the jobs according to the non-decreasing of the obtained values.
- L_3 **:** Calculate for each job i the value $\max\{p_{i1}, p_{i2}\}$. Order the jobs according to the non-decreasing of the obtained values.
- L_4 **:** We use Johnson's rule. Construct two subset J_1 and J_2 where $J_1 = \{i | p_{i1} <= p_{i2}\}$ and $J_2 = \{i | p_{i1} > p_{i2}\}$. Order the jobs of J_1 in increasing order of p_{i1} (SPT), and the jobs in the set J_2 in decreasing order of p_{i2} (LPT). The new list is $J1$ follows by J_2.

L_5 **:** This list is obtained using the same method as L_4 except that we use $d_i - p_{i1}$ and $d_i - p_{i2}$ as an input for the SPT and LPT order.

4.2. Lower Bounds

In this subsection, we provide lower bounds. Assume that π, π^c is a sequence, where π is a partial schedule contains k-jobs already scheduled, and π^c is the unscheduled part with $n-k$ jobs. Let n'_A and n'_B denote the number of unscheduled jobs of agent A and B respectively. Furthermore, let t_1 and t_2 be the completion time of the schedule π on M_1 and M_2. Suppose that the processing times of the unscheduled jobs are $a_1, a_2, ..., a_{n-k}$ on M_1 and $b_1, b_2, ..., b_{n-k}$ on M_2, We denote by

$$d^X_{(1)}, d^X_{(2)}, ..., d^X_{(n^X)}$$

the due dates ordered according to the EDD rule, and by $a^X_{(1)}, a^X_{(2)}, ..., a^X_{(n^X)}$ the processing times on M_1 ordered according to the SPT rule. Similarly, we denote $b^X_{(1)}, b^X_{(2)}, ..., b^X_{(n^X)}$ the processing times on M_2 ordered according to the SPT rule for $X = \{A, B\}$. Then we have :

$$C_{[k+1]}(S) \geq \max\{t_1 + a^A_{(1)}, t_2\} + b^A_{(1)} \geq t_1 + a^A_{(1)} + b^A_{(1)},$$

where $C^A_{[k+1]}(S)$ denotes the completion time for the $[k+1]$ job of agent A. The same idea gives a lower bound for the completion time of the job $k+j$ as $C^A_{[k+j]}(S) \geq t_1 + b^A_{(1)} + \sum_{i=1}^{j} a^A_{(i)}$ where $1 \leq j \leq n'_A$.

We calculate a lower bound for the sequence using the obtained completion time for agent A, we donate it as LB^A_1 where $LB^A_1 = f + \sum_{i=1}^{n'_A} \max\{C_{[k+j]}(S) - d_{(j))}, 0\}$ and f is the value of the objective function for the first partial scheduled jobs.

Similarly, a lower bound LB^A_2 can be obtained using the processing times on M_2. Employing the fact that $C_{[k+1]}(S) \geq \max\{t_1 + a^A_{(1)}, t_2\} + b_{(1)} \geq t_2 + b^A_{(1)}$, we can derive that the completion time for the $k+j$ job is : $C_{[k+j]}(S) \geq t_2 + \sum_{i=1}^{j} b^A_{(j)}$ for $1 \leq j \leq n'_A$.

For the second agent, we can derive two lower bounds LB^B_1 and LB^B_2. This can be done by constructing two instances of a single machine problem with an objective to minimize the number of tardy jobs which is known to be solved by More's algorithm as follows : For the first instance, we set the number of jobs equal to n_B and the processing time $p_i = p^B_{i1}$ and due date $d_i = d^B_i$ for $i = 1, \ldots, n'_B$, we consider that the first agent start processing his jobs at time $t_1 + b^B_{(1)}$. The same way, we can construct a second instance by changing the processing time to $p_i = p^B_{i2}$ and the start time to $t2$. By solving these two instances, we get two lower bounds. In our branch-and-bound algorithm, we construct a tree with n level. For a given level $l-1$ with $l-1 < n$, we have a sub-sequence π represents a node with $l-1$ fixed jobs. To perform the next iteration, we create l different children, by adding unscheduled jobs at the end of the current sub-sequence π to get l schedules. The obtained solutions will be evaluated and compared with the upper bounds. Using the fact that $l-1 < n$, we have at least one child. If $l = n$, we get a complete sequence S. The best found solution is updated if the objective function of one of the produced solutions is less than the current. Otherwise, we have a partial schedule with length $l < n$. The child will be discarded when it is dominated by using the dominance properties or the value of its lower bound is greater than the value of the best feasible solution encountered so far.

4.3. Dominance rules

Property 1 *If a job $i \in J_B$ is tardy, then there exists an optimal schedule where i is sequenced at the end of the sequence.*

Property 2 *For two consecutive jobs i and j (even if they belong two different agents), if $C_j(S) \leq d_j$, $C_i(S') \leq d_i$ and $C_j(S) > C_i(S')$ then schedule j immediately after i.*

Property 3 *For two consecutive jobs i and j, if $i \in J_A$, $j \in J_B$, $C_j(S) \leq d_j$ and $C_i(S') \geq C_j(S)$ then schedule j immediately after i.*

4.4. Branching rule

To explore the tree, we use breadth-first strategy until we reach a pre-defined level r. Then for each node in level r, we start depth-first traversing search strategy. In case that $r = 0$, we obtain the classic depth-first strategy.

5. META-HEURISTICS

Since the problem is strongly $\mathcal{NP}$-hard, the heuristic approach is well justified. In this section, we propose two meta-heuristics.

5.1. Tabu search

Tabu search (TS) is a meta-heuristic based on a neighborhood structure. TS starts with an initial solution. In our case, we choose the best solution found by the heuristics. Each iteration, we choose the best solution from the neighborhood. A neighbor solution can be generated with various methods : swap, insertion, etc. For more details, see e.g. [Nowicki and Smutnicki(1996)].

The general procedure can be resumed as follows :

Algorithm 2 (h)
```
S ← GenerateInitialSolution();
S_best ← S;
TabuList ← ϕ;
while Termination conditions not met do
    Generate a set of neighbor solutions using moves that are not in tabu list;
    Evaluate the obtained solutions;
    S ← the best obtained solution;
    if S is better the S_best then
        S_best ← S;
    TabuList ← a move that generate the new solution from old solution S;
    if Inspiration criteria is met then
        delete some moves from the TabuList;
return the best found solution S_best;
```

To avoid re-visiting a solution previously generated, we use a tabu list that contains previous solutions. This list is also used to exit local minima. The algorithm stops when pre-defined criteria are satisfied.

A move : To generate a new solution, we used two types of moves :

— *Swap :* we pick a position i randomly. Next, we swap the job in position i with a job in position $j, \forall j = 1,..,n, i \neq j$ respecting the condition that the two positions $i \& j$ are not in tabu list.

— *Insertion :* we choose a job in position i randomly, then we move this job after a job in position $j, \forall j = 1,..,n, i \neq j$ respecting the condition that the two positions $i \& j$ are not in tabu list.

Tabu list : We use a list to save moves. A move is the position i and a position j that gives the best new solution. This move is added to the end of the list.

Aspiration criterion : To make moves acceptable again, we set the length of the tabu list to 20. When the length of this list reached its maximum, the oldest five moves are deleted.

Stopping Criterion The number of operations, set to 1000, is used as a termination condition.

6. COMPUTATIONAL STUDY

In order to assess the performance of the proposed algorithm, we carried out numerical experiments with respect to the efficiency of the mathematical model, and the branch-and-bound algorithm in terms of the CPU time. We also studied the quality of the solutions produced by the meta-heuristic algorithm. The proposed branch-and-bound algorithm and the meta-heuristic were coded using C++. The computational experiments were carried out on a PC with Intel(R) Core(TM) i7-2670QM, CPU 2.20 GHZ and 8.00 GB RAM on Windows 7 operating system.

6.1. Computational results Exact methods

In our computational experiments, we generate the instances randomly. The integer processing times were generated from a uniform distribution $[1,25]$ and $[25,100]$, the due dates were generated from another uniform distribution over the integers between $T(1-\tau-R/2)$ and $T(1-\tau+R/2)$ where R is a parameter called the due date range, and τ is called the tardiness factor. The combination of (τ, R) took the values of $(0.25, 0.25)$, $(0.25, 0.50)$, $(0.25, 0.75)$, $(0.50, 0.25)$, $(0.50, 0.50)$, and $(0.50, 0.75)$. The value T is the summation of the processing times on machine M_2 and the minimum processing time of all jobs on machine M_1. Twenty instances were generated for each combination and interval. Problems with 15 and 16 jobs were considered. The number of jobs of agent A is set to $floor\{n/2\}$ (the greatest integer less than or equal to n/2) and $\lambda = 0.1$. CPU time limited to one hour, we count the number of times an optimal solution is found, we show also the maximum, the minimum and the average time to find solutions. The results were summarized in Table, 1 and 2. We used the following notation :

*Nb op***t :** Number of optimal solution.

B&B **:** Branch-and-bound.

For $n = 15$, we can see in Table 1 that the branch-and-bound algorithm can solve all the instances in no time with different value of τ and R. It becomes easier to solve instances when τ and R are increasing.

Table 2 presents the performance of the branch-and-bound algorithm for $n = 16$, it can be seen that solving instances with this size is getting harder when $r = 0.25$ and $R = 0.25$ or $R = 0.5$ but it can solve all the instances withing one hour. This lack of performance is due to the weakness of the lower bound. However, it can solves all the instances in no-time for the rest combination of r and R.

Interval			[1,25]	[25,100]
τ	R	f	*B&B*	*B&B*
0.25	0.25	Nb opt	20	20
		Min	1.523	0.914
		Max	22.784	17.005
		Mean	6.91905	5.02485
	0.50	Nb opt	20	20
		Min	0.95	0.977
		Max	9.329	5.556
		Mean	2.8678	2.56955
	0.75	Nb opt	20	20
		Min	0.775	0.766
		Max	4.573	1.699
		Mean	1.26375	0.9591
0.50	0.50	Nb opt	20	20
		Min	0.823	0.776
		Max	2.123	1.275
		Mean	1.1448	0.86685
	0.75	Nb opt	20	20
		Min	0.787	0.769
		Max	1.185	0.808
		Mean	0.86505	0.7837

TABLE 1 – The Performance of the *B&B* algorithm ($n = 15$).

6.2. Computational results for the heuristics

For small size instances with 15 jobs, we experimentally compared the effectiveness of solutions generated by the meta-heuristic procedure with the optimal solutions produced by the branch-and-bound procedure, the data is generating according to two different processing time intervals ($[1,25]$ and $[25,100]$) and due dates range (The combination of τ and R with $\tau \in \{0.25, 0.5\}$ and $R \in \{0.25, 0.5, 0.75\}$), the obtained results are reported in Table 3.

From Table 3, we can observe that for $\tau = R$, the rules L_3 and L_1 outperform the other rules. But L_1 has the best performance for the general cases. L_2 gives a near performance to L_1, and the two rules L_4 and L_5 are the worst rules even if they are based on Jackson's rule. For tabu search algorithm, we can see that for $\tau = 0.25$, the swap move is the best option. However, for $\tau = 0.50$, the insertion move gives better performance.

7. CONCLUSION

Our study focuses on the two-machine flow shop with two competing agents to minimize the linear combination of the total tardiness and the number of tardy jobs. We discussed the computational complexity of various special cases. To solve it optimally, we proposed a procedure based on the branch-and-bound algorithm which uses several dominance properties and bounds. Moreover, a heuristic based on different priority lists along with a tabu search algorithm that uses two different moving mechanisms are proposed to find a near-optimal solution. Finally, computational results for randomly generated problem instances are reported.

Interval		[1,25]		[25,100]	
r	R	mean	max	mean	max
0.25	0.25	750.9206	2087.8	241.056	728.299
	0.5	402.2024	1101.51	1624.8056	2945.8
	0.75	91.5664	406.83	0.2634	1.182
0.25	0.5	1.412	5.886	0.2698	1.093
	0.75	8.6002	35.427	0.053	0.144

TABLE 2 – The Performance of the *B&B* algorithm ($n = 16$).

8. REFERENCES

[Agnetis et al.(2004)Agnetis, Mirchandani, Pacciarelli, and Pacifici] Allesandro Agnetis, Pitu B. Mirchandani, Dario Pacciarelli, and Andrea Pacifici. Scheduling problems with two competing agents. *Oper. Res.*, 52(2) :229–242, March 2004. ISSN 0030-364X.

[Ahmadi-Darani et al.(2018)Ahmadi-Darani, Moslehi, and Reisi-Nafchi] M Ahmadi-Darani, G Moslehi, and M Reisi-Nafchi. A two-agent scheduling problem in a two-machine flowshop. *International Journal of Industrial Engineering Computations*, 9(3) :289–306, 2018.

[Garey et al.(1976)Garey, Johnson, and Sethi] M. R. Garey, D. S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.*, 1(2) :117–129, May 1976. ISSN 0364-765X.

[Graham et al.(1979)Graham, Lawler, Lenstra, and Kan] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier, 1979.

[Hariri and Potts(1989)] AMA Hariri and CN Potts. A branch and bound algorithm to minimize the number of late jobs in a permutation flow-shop. *European Journal of Operational Research*, 38(2) :228–237, 1989.

[Johnson(1954)] Selmer Martin Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1) :61–68, 1954.

[Kim(1993)] Yeong-Dae Kim. A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. *Computers & Operations Research*, 20(4) :391 – 401, 1993. ISSN 0305-0548.

[Lawler and Moore(1969)] Eugene L Lawler and J Michael Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16 (1) :77–84, 1969.

[Lee et al.(2010)Lee, Chen, and Wu] Wen-Chiung Lee, Shiuan-kang Chen, and Chin-Chia Wu. Branch-and-bound and simulated annealing algorithms for a two-agent scheduling problem. *Expert Syst. Appl.*, 37(9) :6594–6601, September 2010. ISSN 0957-4174.

[Lei(2015)] Deming Lei. Variable neighborhood search for two-agent flow shop scheduling problem. *Computers & Industrial Engineering*, 80 :125–131, 2015.

[Lenstra et al.(1977)Lenstra, Kan, and Brucker] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. In P.L. Hammer, E.L. Johnson, B.H. Korte, and G.L. Nemhauser, editors, *Studies in Integer P rogramming*, volume 1 of *Annals of Discrete Mathematics*, pages 343 – 362. Elsevier, 1977.

[Luo et al.(2012)Luo, Chen, and Zhang] Wenchang Luo, Lin Chen, and Guochuan Zhang. Approximation schemes for two-machine flow shop scheduling with two agents. *Journal of Combinatorial Optimization*, 24(3) :229–239, 2012.

interval	τ	R	f	L_1	L_2	L_3	L_4	L_5	TSS	TSI
[1,25]	0.25	0.25	opt	5	3	4	4	4	17	13
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	0.810	0.810	0.767	1.209	0.952	0.116	0.133
			mean	0.249	0.240	0.272	0.371	0.367	0.009	0.024
		0.5	opt	9	6	4	5	2	17	17
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	1.000	2.800	2.800	4.000	6.600	0.056	0.052
			mean	0.285	0.555	0.524	0.634	0.979	0.006	0.004
		0.75	opt	10	11	6	2	1	16	15
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	1.000	1.591	2.850	2.850	2.850	0.950	0.950
			mean	0.235	0.252	0.844	1.112	1.318	0.110	0.106
	0.5	0.5	opt	0	0	2	0	0	5	7
			min	0.173	0.090	0.000	0.163	0.090	0.000	0.000
			max	0.667	0.667	0.701	0.727	0.701	0.288	0.288
			mean	0.323	0.307	0.237	0.409	0.361	0.061	0.057
		0.75	opt	0	1	0	1	1	3	6
			min	0.079	0.000	0.062	0.000	0.000	0.000	0.000
			max	0.966	0.500	1.500	1.000	1.000	0.310	0.310
			mean	0.301	0.305	0.385	0.393	0.422	0.093	0.092
[25,100]	0.25	0.25	opt	5	3	3	3	3	18	13
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	4.750	4.750	4.750	4.750	4.750	4.750	4.750
			mean	0.516	0.564	0.553	0.595	0.636	0.245	0.309
		0.5	opt	14	13	9	5	2	20	18
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	1.000	1.000	2.000	2.000	2.000	0.000	0.500
			mean	0.192	0.242	0.583	0.683	0.842	0.000	0.028
		0.75	opt	12	13	6	4	2	17	14
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	1.000	2.000	3.000	3.800	2.000	1.000	1.900
			mean	0.337	0.351	0.942	1.125	1.283	0.111	0.303
	0.5	0.5	opt	3	2	6	4	5	13	9
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	0.600	0.600	0.500	0.500	0.500	0.250	0.250
			mean	0.255	0.265	0.189	0.190	0.179	0.044	0.065
		0.75	opt	5	4	2	1	3	6	5
			min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			max	0.905	1.000	0.891	0.667	0.667	0.333	0.469
			mean	0.314	0.372	0.356	0.355	0.314	0.161	0.181

TABLE 3 – The Performance of the heuristic and meta-heuristics vs. the branch-and-bound ($n = 15$).

[Mor and Mosheiov(2014)] B. Mor and G. Mosheiov. Polynomial time solutions for scheduling problems on a proportionate flowshop with two competing agents. *Journal of the Operational Research Society*, 65(1) :151–157, Jan 2014. ISSN 1476-9360.

[Nowicki and Smutnicki(1996)] Eugeniusz Nowicki and Czesław Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Re-*

search, 91(1) :160–175, 1996.

[Pan and Fan(1997)] Jason Chao-Hsien Pan and En-Tsu Fan. Two-machine flowshop scheduling to minimize total tardiness. *International Journal of Systems Science*, 28(4) :405–414, 1997.

[Pan et al.(2002)Pan, Chen, and Chao] Jason Chao-Hsien Pan, Jen-Shiang Chen, and Chii-Ming Chao. Minimizing tardiness in a two-machine flow-shop. *Computers & Operations Research*, 29(7) :869–885, 2002.

[Sen et al.(1989)Sen, Dileepan, and Gupia] Tapan Sen, Parthasarati Dileepan, and Jatinder N.D. Gupia. The two-machine flowshop scheduling problem with total tardiness. *Computers & Operations Research*, 16(4) :333 – 340, 1989. ISSN 0305-0548.