Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A two-agent scheduling problem in a two-machine flowshop

Mohammad-Hasan Ahmadi-Darani[a], Ghasem Moslehi[a] and Mohammad Reisi-Nafchi[a*]

[a]*Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran*

C H R O N I C L E

Article history:
Received March 18 2017
Received in Revised Format August 20 2017
Accepted August 30 2017
Available online August 30 2017

Keywords:
Scheduling
Flowshop
Two-agent
Mathematical programming
Tabu search

A B S T R A C T

In recent years, many studies on the multi-agent scheduling problems in which agents compete for using the shared resources, have been performed. However, relatively few studies have been undertaken in the field of the multi-agent scheduling in a flowshop environment. To bridge the gap, this paper aims at addressing the two-agent scheduling problem in a two-machine flowshop. Because of the importance of delay penalties and efficient resource utilization in many manufacturing environments, the objective is to find an optimal schedule which has the minimum total tardiness for the first agent's jobs, under the makespan limitation for the second agent. Since this problem is strongly *NP-hard*, several theorems and properties of the problem are proposed to apply in exact and meta-heuristic methods. Also, for some instances of the problem for which exact methods cannot achieve optimal solutions in a reasonable amount of time, a tabu search algorithm is developed to achieve near-optimal solutions. Computational results of the tabu search algorithm show that the average absolute error value is lower than 0.18 percent for instances with 20 to 60 jobs in size.

1. Introduction

The scheduling problems in which several agents compete for using the shared resources have recently attracted the attention of many researchers. In these problems, contrary to the general multi-objective problems in which the optimality criteria reflect data from the consideration of all jobs, each agent has its own set of jobs and follows its own optimality criterion only. Baker and Smith (2003) and Allesandro Agnetis et al. (2004) introduced the two-agent scheduling problems for the first time. They introduced the positive combination of each agent's objective, constrained optimization, and Pareto optimization case of the problem. In the constrained optimization case, the goal is to optimize the objective function of one agent under the condition that the objective functions of the other agents are smaller than the predetermined values. Also, in the Pareto optimization problem, the goal is to find a set of non-dominant solutions and their related sequences.

In this paper, a two-agent constrained optimization scheduling problem in the two-machine flowshop is studied. The goal of this problem is to minimize the total tardiness of the first agent's jobs subject to the condition that the makespan of the second agent's jobs is allowed to be lower than an upper bound. In

* Corresponding author Tel.: +98 3133911486; fax: +98 3133915526
E-mail: reisi.m@cc.iut.ac.ir (M. Reisi-Nafchi)

© 2018 Growing Science Ltd. All rights reserved.
doi: 10.5267/j.ijiec.2017.8.005

the following, a number of related studies dealing with the constrained optimization case of the multi-agent scheduling problems are reviewed.

Kellerer and Strusevich (2010) provided a Fully Polynomial Time Approximation Scheme (FPTAS) for the scheduling problem in a single-machine environment with the objective of minimizing the total weighted completion times of the first agent while the makespan of the second agent was bounded. Wu et al. (2011) offered a branch and bound algorithm and some heuristic procedures for the problem in a single-machine environment with learning effects to minimize the total tardiness of the first agent and a limited number of tardy jobs of the second agent. Wu (2014) investigated a two-agent single-machine scheduling problem with learning effects via an objective function which minimizes the weighted completion time of all the jobs subject to a constraint that one agent's makespan cannot exceed a prescribed upper bound. They proposed a branch and bound algorithm along with three simulated annealing procedures for the problem. Gajpal et al. (2014) investigated a single machine scheduling problem with the aim of minimizing the total weighted completion times of the first agent's jobs and bounded completion times of the second agent, and presented three heuristics for this problem. Yin et al. (2012a) addressed a two-agent scheduling problem in a single-machine environment with arbitrary release dates, where the objective was to minimize the total tardiness of one agent, while keeping the total lateness of the other agent below or at a fixed level. They developed a mixed integer programming (MIP), a branch and bound procedure, and a marriage in honey-bees optimization algorithm (MBO) for this problem. Yin et al. (2012b) considered a single machine scheduling problem with deteriorating jobs where the objective of each agent was the maximum amount of a regular function of completion times of jobs. For this problem, they provided an optimal procedure with polynomial time complexity. Wu et al. (2013) deliberated upon a two-agent single-machine scheduling problem with deteriorating jobs. The objective of this problem was to minimize the total weighted number of tardy jobs of the first agent's jobs subject to the condition that the maximum lateness of the second agent is allowed to be lower than an upper bound. They proposed a branch and bound algorithm and a tabu search algorithm for this problem. Liu et al. (2013) proposed a new scheduling model with two-agent and sum-of-processing-times-based deterioration. The objective is to minimize the total completion time of the first agent with the restriction that the makespan of the second agent cannot exceed a given upper bound. They proposed optimal properties of the problems and the optimal polynomial time algorithms to solve it. Cheng et al. (2011) considered a two-agent single-machine scheduling problem involving deteriorating jobs and learning effects, simultaneously. The objective is to minimize the total weighted completion time of the jobs of the first agent with the restriction that no tardy job is allowed for the second agent. They developed a branch-and-bound and several simulated annealing algorithms to solve the problem. For more studies on this line of research, the reader may refer to Alessandro Agnetis et al. (2007) and Leung et al. (2010).

As seen above, all the reviewed studies are concerned with the single machine environment. However, it is of note that usually more than one operation must be executed on every job in many manufacturing systems. Often, these operations have to be performed on all the jobs in the same order, implying that the jobs have to follow the same route. This environment is referred to as a flowshop (Pinedo, 2002). According to the literature, few studies have been conducted on multi-agent scheduling in a flowshop environment. Alessandro Agnetis et al. (2004) showed that the two-agent constrained optimization scheduling problem in a two machine flowshop intended to minimize the makespan of the first agent's jobs with the limited makespan of the second agent is NP-hard. For this problem, Luo et al. (2012) provided a dynamic programing algorithm with pseudo-polynomial time complexity and a FPTAS. Lee et al. (2010) considered a two-agent scheduling problem in the two-machine flowshop, where the objective was to minimize the total tardiness of the first agent's jobs with the restriction that the number of tardy jobs of the second agent is zero. They proposed a branch and bound algorithm and a simulated annealing algorithm and claimed that the branch and bound algorithm can solve problems involving up to 15 jobs in size in a reasonable amount of time. Lee et al. (2011) considered a two-agent scheduling problem in the two-machine flowshop, where the objective was to minimize the total completion time of the first agent's jobs with the restriction that the number of tardy jobs of the second agent is zero. They presented a branch and bound and a simulated annealing procedure to solve the problem, and showed that the branch and bound algorithm is capable of solving instances containing up to 20 jobs in size in a reasonable amount of time. Fan and Cheng (2016) studied two-agent scheduling in a two-machine flowshop. The cost function was the weighted sum of some common regular functions, including the makespan and the total completion time. For the first problem, they proposed an ordinary NP-hardness proof and a pseudo-polynomial-time algorithm. For the second problem, they proposed an approximation algorithm based on linear programming relaxation of the problem. Shiau et al. (2015) studied a two-agent scheduling problem in a two machine flowshop with learning effects. The objective is to minimize the total completion time of the jobs from one agent, given that the maximum tardiness of the jobs from the other agent cannot exceed a bound. They also provided a branch-and-bound algorithm for the problem. In addition, They presented several genetic algorithms to obtain near-optimal solutions. Lei (2015) studied flow shop scheduling problem with two agents and considered its feasibility model. In this problem the goal was to minimize the makespan of the first agent and the total tardiness of the second agent simultaneously under the given upper bounds. They proposed a simple variable neighborhood search (VNS) algorithm for this problem. Recently, Perez-Gonzalez and Framinan (2014) and Alessandro Agnetis et al. (2014) have reviewed the multi agent scheduling problems, which can be referred to for more insight into such problems.

As single agents (one objective) in the two machine flowshop are related to the problem considered in this paper, some relevant studies are reviewed in the following. At this point, it is noteworthy to say that the essential conclusion ever reached for minimizing makespan in a two-machine flowshop in the scheduling theory is known as Johnson's rule (Johnson, 1954). Johnson (1954) provided the optimal schedule using a simple procedure with polynomial time complexity for this problem. Although finding the optimal solution for the problem of minimizing the total tardiness in this environment is limited to search in the permutation schedule, the problem of minimizing total tardiness in the case of considering zero value of the due date values reduces to the problem of minimizing the total completion time, which is a problem with strongly NP-hard computational complexity (Graham, 1979). Thus, in most of such studies, implicit enumeration approaches have been used. Sen et al. (1989), Kim (1993), Pan and Fan (1997), Pan et al. (2002) and Schaller (2005) provided a branch and bound algorithm for minimizing the total tardiness in the two-machine flowshop. For this problem, Kharbeche and Haouari (2012) presented simple MIP models based on the position of jobs in the sequence. Their results showed that the efficiency of mathematical models is higher than that of the branch and bound algorithm. Hoogeveen and Velde (1995), Della Croce et al. (1996), Federico Della Croce et al. (2002) and Akkan and Karabatı (2004) developed a branch and bound algorithm for the two-machine flowshop scheduling problem with the objective of minimizing the total completion times. Haouari and Kharbeche (2013) presented an assignment-based lower bound for the two-machine flowshop scheduling problems with a regular additive performance criterion. Their computational results showed that the quality of this lower bound is better than the lower bounds in the literature focusing on the special cases of minimizing both the total completion time and the total tardiness in the two-machine flowshop.

The paper is organized as follows: the problem definition and used symbols are given in the next section. The theorems and properties of the optimal solutions are presented in section 3. In section 4, the mathematical model is introduced. Furthermore, tabu search method in section 5, computational results in section 0, and in the last section, conclusions and suggestions for future research are presented.

2. Problem definition and notations

In this paper, the two-agent flowshop scheduling problem is dealt with. As delay penalties and efficient resource utilization are more practical criteria in the scheduling problems, in the considered problem the objective is to find an optimal schedule to minimize the total tardiness of the first agent, under the situation that the makespan of second agent is bounded. All jobs are available at the zero time and have to follow the same route i.e. they must be processed first on machine 1 and then on machine 2. Let J_A

and J_B ($J_A \cap J_B = \emptyset$) denote the job sets belonging to agent *A* and agent *B*, respectively, where J_A consists of n_A jobs and J_B has n_B jobs and therefore *n* ($n = n_A + n_B$) jobs must be scheduled. For each job *j* belonging to agent x, a processing time a_j^x on the first machine, a processing time b_j^x on the second machine, and a due date d_j^x are defined. Considering a schedule *S*, the completion time of job *j* of the agent *x* on the second machine is denoted by $C_j^x(S)$ and the tardiness of this job is defined as $T_j^x(S) = Max\{0, C_j^x(S) - d_j^x\}$. Also, the makespan of agent *x* is defined as $C_{max}^x(S) = Max_{j \epsilon J_x}\{C_j^x(S)\}$. Following the notations offered by Graham et al. (1979) as well as the classification of Allesandro Agnetis et al. (2004), this problem is shown as $F2||\sum_{j=1}^{n_A} T_j^A : C_{max}^B \leq Q$. When the number of jobs belonging to agent *B* is zero or *Q* value is large enough, the problem $F2||\sum_{j=1}^{n_A} T_j^A : C_{max}^B \leq Q$ is reduced to minimizing the total tardiness in a two-machine flowshop scheduling problem, which is strongly *NP-hard* (Lenstra et al., 1977). Therefore, $F2||\sum_{j=1}^{n_A} T_j^A : C_{max}^B \leq Q$ is also strongly *NP-hard.*

Because of the limited makespan of the second agent's jobs and influence of the *Q* value on determining the solution space, different aspects of the problem with respect to different values of the *Q* are investigated as follows.

As the makespan of the second agent is limited to *Q*, if *Q* is less than the minimum time required to complete the second agent's jobs, the problem will be infeasible. Also, if it is large enough, the problem reduces to the problem of minimizing the total tardiness in the two-machine flowshop scheduling. This is why the *Q* value has to be especially addressed to study the problem. Suppose that C_x^* and $C_x^\#$ indicate the minimum and maximum time required to complete all the jobs belonging to agent x, respectively. $C_x^\#$ is obtained by applying Johnson's rule (Johnson, 1954) and $C_x^\#$ is calculated by the inverse of the sequence obtained by Johnson's rule (Johnson, 1954) By considering these values, there exist the following three cases:

First case: If $C_B^* > Q$, there is no feasible schedule,

Second case: If $C_B^* = Q$, the second agent's jobs are arranged at the beginning of the sequence based on Johnson's rule (Johnson, 1954) and the first agent's jobs are arranged after them at the end of the sequence. In this case, the problem is to find the optimal sequence of the first agent's jobs arranged after the second agent's jobs according to the total tardiness as the objective function,

Third case: Consider the situation in which the *Q* value is so large that if the first agent's jobs are arranged at the beginning of the sequence with the minimum total tardiness (the optimal sequence), the second agent's jobs can be placed after them at the end of the sequence while maintaining the feasibility condition. First, the maximum possible required time to arrange the first agent's jobs at the beginning of the sequence is calculated. Then, with regard to this calculated value as a virtual job (indicator of the first agent's jobs) the second agent's jobs are assigned after the first agent's jobs. In other words, in this case we need to suppose that *L* is the lower bound of *Q* ($L \leq Q$). To determine the value of *L*, at first, $C_A^\#$ is calculated which is equal to the upper bound of the completion time of the first agent's jobs. Then, the virtual job j_0 that has zero processing time on the first machine and processing time $C_A^\# - \sum_{j=1}^{n_1} a_j^A$ on the second machine is considered. Establish $J_B \cup j_0$ i.e. the set of the second agent's jobs and virtual job j_0 and calculate $C_{J_B \cup j_0}^*$ value as the minimum time required to complete the second agent's jobs arranged after the first agent's jobs. Therefore, with the definition of $L = \sum_{j=1}^{n_A} a_j^A + C_{J_B \cup j_0}^*$, If $L \leq Q$, the second agent's jobs are arranged at the end of the sequence according to the Johnson's rule (Johnson, 1954). The first agent's jobs are placed at the beginning of the sequence, and the two-agent scheduling problem reduces to finding the optimal sequence of the first agent's jobs at the beginning of the sequence according to the total tardiness as the objective function.
According to the above cases, the problem is investigated further for the values $C_\mathrm{B}^* < Q <$

$\sum_{j=1}^{n_A} a_j^A + C^*_{J_B \cup j_0}$, where the Q value shows its effect on the optimal sequence apart from the three mentioned cases.

3. Theorems and optimal properties of the problem

In this section, some efficient theorems and optimal properties of the problem used in the developed methods of the next sections are introduced and some of them are given in the appendix. The presented propositions are classified based on the jobs belonging to each agent. In these theorems and propositions, S is a schedule in which job j is processed before job i and S' is a schedule that is identical to S, except for the fact that jobs j and i are interchanged. Also, when both job i and job j belong to one agent, for brevity the indexes of agent are removed.

For the cases in which both job i and job j belong to the first agent with the objective of minimizing the total tardiness of jobs, the following theorems are discussed.

Theorem 1 (Pan & Fan, 1997): For any two jobs i and j belonging to the first agent, if conditions $a_i \leq a_j$, $b_i = b_j$ and $d_i \leq d_j$ are *m*et, then there exists an optimal schedule such that job j is processed after job i.

Theorem 2 (Pan et al., 2002): For job i, if there exists a job j satisfying conditions $b_j \geq b_i$, $a_i + b_i \geq a_j + b_j$ and $b_j - d_j \geq b_i - d_i$, then there exists an optimal schedule such that job i is not the first job of the sequence.

When both job i and job j belong to the second agent with the bounded makespan, the following propositions are developed in this paper.

Proposition 1: For any two adjacent jobs i and j belonging to the second agent, the Johnson's rule (Johnson, 1954) is established between them.

Proof: Based on the Johnson's rule (Johnson, 1954), if job i is assigned before job j, it must be shown that by interchanging jobs i and j in S, $C_j(S') \leq C_i(S)$. Let I_i and I_j denote idle times in S occurring on the second machine immediately prior to the processing of jobs i and j, respectively. Also, let I_i' and I_j' denote idle times in S' occurring on the second machine immediately prior to the processing of jobs i and j, respectively. Further, let F and C denote completion times of jobs that are completed before these jobs on the first machine and the second one, respectively. According to Fig. **1**, the completion times of jobs i and j in S and S' are calculated by relations $C_i(S) = C + I_j + b_j + I_i + b_i$ and $C_j(S') = C + I_i' + b_i + I_j' + b_j$, respectively. So, it is enough to show that $I_i' + I_j' \leq I_j + I_i$. The value of $I_j + I_i$ and $I_i' + I_j'$ are calculated by relations $max(0, F + a_j - C, F + a_j + a_i - b_j - C)$ and $max(0, F + a_i - C, F + a_i + a_j - b_i - C)$, respectively. According to the Johnson's rule (Johnson 1954), when the relation $min(a_i, b_j) \leq min(b_i, a_j)$ is true, job i is processed before job j. Thus, according to the value of $min(a_i, b_j)$, two cases are considered:

First case: $a_i \leq b_j$ and thus $a_i \leq a_j$ and $a_i \leq b_i$; In this case, because of $a_i \leq b_j$, $I_j + I_i = max(0, F + a_j - C)$ and due to $a_i \leq b_i$, $F + a_i + a_j - b_i - C \leq F + a_j - C$. Also, because of $a_i \leq a_j$, we have $F + a_i - C \leq F + a_j - C$ and so $I_i' + I_j' \leq I_j + I_i$.

Second case: $b_j < a_i$ and thus $b_j < a_j$ and $b_j < b_i$; In this case, because of $b_j < a_i$, the relation $F + a_j - C \leq F + a_j + a_i - b_j - C$ is established and the value of $I_j + I_i$ for both jobs is calculated through $I_j + I_i = max(0,\ F + a_j + a_i - b_j - C)$. In this equation, because of $b_j < b_i$ and $b_j < a_j$, the value of $(F + a_j + a_i - b_j - C)$ is larger than the values $(F + a_i + a_j - b_i - C)$ and $(F + a_i - C)$ used to calculate $I_i' + I_j'$, and therefore $I_i' + I_j' \leq I_j + I_i$. With regard to these cases, by interchanging job *i* and job *j* in *S*, the makespan of these two jobs does not increase. ■

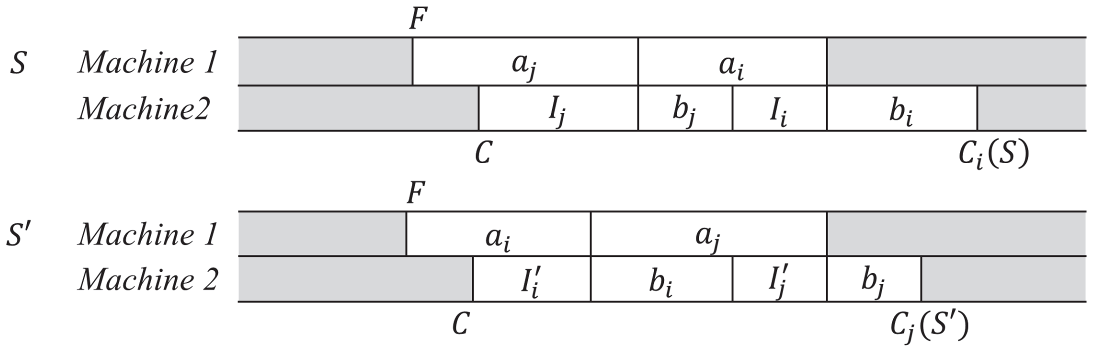

Fig. 1. Interchanging two adjacent jobs

Corollary 1: Due to the transitivity property in the Johnson's rule (Johnson 1954), if the adjacent jobs belonging to the second agent are more than two jobs, then there exists an optimal schedule such that the Johnson's rule (Johnson, 1954) is established among all of them.

Proposition 2: For two jobs *i* and *j* belonging to the second agent, if conditions $a_i \leq a_j$ and $b_i = b_j$ are met, then there exists an optimal schedule such that job *j* is processed after job *i*.

Proof: According to Fig. 2, let M_1 and M_1' be the total processing times of jobs between job *i* and job *j* on the first machine in *S* and *S'*, respectively. Also, let M_2 and M_2' be the total processing times plus idle times on the second machine in *S* and *S'*, respectively. The completion times of job *i* and job *j* in schedule *S* and *S'* are calculated by relations $C_i(S) = C + I_j + b_j + M_2 + I_i + b_i$ and $M_2' + I_j' + b_j$, and therefore we have $C_j(S') - C_i(S) = I_i' + M_2' + I_j' - (I_j + M_2 + I_i)$. Because $a_i \leq a_j$ and $b_i = b_j$, by interchanging these jobs in *S*, the completion times of jobs between job *i* and job *j* will not increase. So, if $I_i > 0$, we have $I_i' + b_i + M_2' + I_j' = I_i + b_i + M_2 + I_j$ and if $I_i = 0$, we have $I_i' + b_i + M_2' + I_j' \leq I_j + b_j + M_2 + I_i$. Therefore, $C_j(S') - C_i(S) \leq b_j - b_i$, and the completion times of subsequent jobs will not increase. □

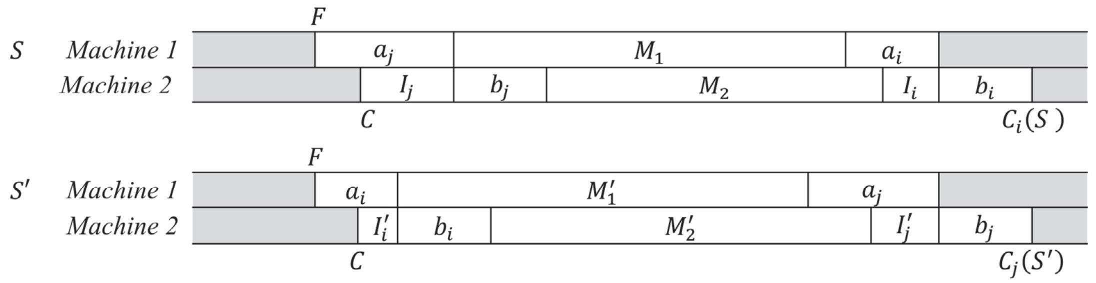

Fig. 2. Interchanging two non-adjacent jobs

Proposition 3: For two jobs *i* and *j* belonging to the second agent, if conditions $a_i \leq a_j$, $b_i \geq b_j$ and $a_i + b_i \leq a_j + b_j$ are met, then there exists an optimal schedule such that job *i* is not the first job of the sequence.

Proof: According to Fig. 3, if the first and third conditions are met, by interchanging job *i* and job *j* in *S*, the completion time of jobs between them will not increase. Thus, because of no increase in the completion time of jobs between job *i* and job *j*, similar to the proof of Proposition 2, the relation $C_j(S') - C_i(S) \leq b_j - b_i$ is established. Therefore, according to the second condition, the completion times of subsequent jobs will not increase. ■

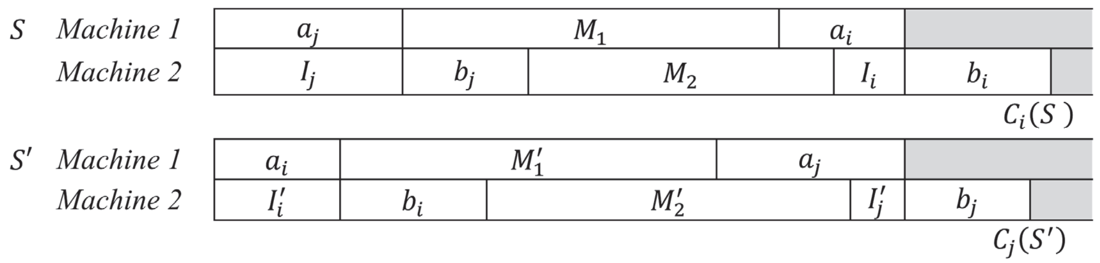

Fig. 3. Interchanging any job of the sequence with the first one

4. Mathematical programming model

Due to the complexity of the studied problem, a branch and bound algorithm is proposed to achieve the optimum solution. In this algorithm, the theorems, optimal properties (proposed in section 3 and appendix), and some lower bounds are used. However, computational results show that the branch and bound algorithm is only able to solve all the problem instances up to 16 jobs in size. Due to the low efficiency of the branch and bound algorithm compared with the mathematical model of the problem, the details of this algorithm are not presented in this paper, and instead the mathematical model is provided as the efficient exact method to optimally solve the problem. Kharbeche and Haouari (2012) presented three mathematical models for the two-machine flowshop problem to minimize the total tardiness. These models are based on the jobs' positions in the sequence and formulated by defining waiting time decision variables, completion time decision variables, and machine's idle time decision variables. Also, Chandra et al. (2009) proposed precedence-based formulations for the permutation flowshop with a common due date. However, according to the computational results of Kharbeche and Haouari (2012) and M'Hallah (2014) as well as preliminary tests which were conducted in this study, the mathematical programming models based on the position of jobs in the sequence showed higher efficiency to achieve optimal solutions. Also, among these models, the mathematical model based on the completion times of jobs on the machines showed the best performance. According to the mentioned issues, the mathematical programming model of $F2||\sum T_j^A : C_{max}^B \le Q$ is presented below, which is denoted as *ACT (Agent's jobs Completion Times)* model. This model is based on the jobs' positions in the sequence and obtained by defining the completion times on machines as decision variables. This model is formulated as relations (1) to (12). In this model, decision variable $T_{[k]}$ is equal to the tardiness of job that is scheduled in the position k, and x_{kj} is a binary variable which takes value 1 if job j is assigned to position k in the sequence and zero otherwise. Decision variable $F_{[k]}$ and decision variable $C_{[k]}$ are completion times of the scheduled job at position k on the first machine and on the second machine, respectively.

$$min \ \sum_{k=1}^{n} T_{[k]} \tag{1}$$

subject to

$$\sum_{k=1}^{n} x_{ki} = 1 \qquad i = 1,2,\dots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ki} = 1 \qquad k = 1,2,\dots,n \tag{3}$$

$$F_{[1]} = \sum_{i=1}^{n} a_i x_{1i} \tag{4}$$

$$F_{[k]} - F_{[k-1]} = \sum_{i=1}^{n} a_i x_{ki} \qquad k = 2,3,\dots,n \tag{5}$$

$$C_{[1]} = \sum_{i=1}^{n} (a_i + b_i) x_{1i} \tag{6}$$

$$C_{[k]} \ge C_{[k-1]} + \sum_{i=1}^{n} b_i x_{ki} \qquad k = 2,3,\dots,n \tag{7}$$

$$C_{[k]} \ge F_{[k-1]} + \sum_{i=1}^{n} (a_i + b_i) x_{ki} \qquad k = 2,3,\dots,n \tag{8}$$

$$T_{[k]} \ge C_{[k]} - \sum_{i \epsilon J_A} (d_i - M) x_{ki} - M \qquad k = 1,2 \dots,n \tag{9}$$

$$C_{[k]} \le Q + M\,(1 - \sum_{i \epsilon J_B} x_{ki}) \qquad k = 1,2 \dots,n \tag{10}$$

$$F_{[k]}, T_{[k]}, C_{[k]} \ge 0 \qquad k = 1,2,\dots,n \tag{11}$$

$$x_{ki} \in \{0,1\} \qquad i,k = 1,2,\dots,n \tag{12}$$

In this model, the objective function (1) minimizes the total tardiness for jobs which are assigned in different positions of the sequence. Since tardiness is calculated in this model only for the first agent's jobs, this equation is equal to minimizing the total tardiness of the first agent's jobs. Eq. (2) and Eq. (3) are assignment constraints and respectively state that at each position there is sequenced only one job and each job must be assigned to exactly one position in the sequence. Constraints (4) to (8) determine the completion times of jobs at different positions of the sequence on the first machine and the second one. Constraint (9) states that tardiness of a position is calculated if its sequenced job belongs to the first agent. Similarly, Constraint (10) states that the completion time of a position is limited to the Q value if the first agent's job is placed in this position. The formulation of *ACT* includes $O(n^2)$ binary variables, $O(n)$ continuous variables, and $O(n)$ constraints.

4.1. Optimal properties in MIP

In section 3, Theorem 1 (Pan & Fan, 1997) and Proposition 2 was proposed for the problem $F2||\sum_{j=1}^{n_A} T_j^A : C_{max}^B \leq Q$. Set Γ including the pairs of jobs i and j (in which job i is processed before job j) is formed by using the mentioned properties. Since $\sum_{k=1}^{n} kx_{ki}$ states the position index of job i, the following inequality is established for members of the set Γ:

$$\sum_{k=1}^{n} kx_{ki} + 1 \leq \sum_{k=1}^{n} kx_{kj} \quad (i,j) \in \Gamma \tag{13}$$

Also, by using Theorem 2 (Pan et al., 2002) and Proposition 3, the values of variables associated with the jobs which cannot be placed at the beginning and at the end of the sequence, are set to zero. Also, since in the members of Γ, job i is processed before job j, job j and job i cannot be placed at the beginning and at the end of the sequence, respectively.

Finally, by calculating the lower bound of completion times of jobs in different positions of the sequence, the procedure which was provided by Haouari and Kharbeche (2013), some variables corresponding to the second agent's jobs can be set to zero because of their calculated completions times' lower bound which is greater than Q. To handle this case, the following procedure is presented:

Step 1: Sort all jobs (including the first and second agent's jobs) according to the Johnson's rule (Johnson, 1954).
Step 2: Calculate the lower bound of completion times of jobs in different positions according to the procedure proposed by Haouari and Kharbeche (2013).
Step 3: Because the lower bound of the completion times of jobs in different positions has been identified, for each one of the second agent's jobs, specify the first position in which the completion time's lower bound of the job is larger than Q.
Step 4: For each one of the second agent's jobs, set to zero the value of corresponding variables in the first position determined in Step 3 up to the last position of the sequence.
Step 5: Finish.

Among the offered optimal properties, inequality (13) was introduced by Kharbeche and Haouari (2012) to improve the efficiency of mathematical programming model of minimizing the total tardiness in a two-machine flowshop. They applied Theorem 1 (Pan & Fan, 1997) to form set Γ. As mentioned before, in addition to utilizing this inequality in this study to form set Γ, Proposition 2 has been used. Also, Kharbeche and Haouari (2012) used Theorem 2 (Pan et al., 2002) to determine those jobs that are not placed at the beginning of the sequence. The other used properties in the mathematical programming model were developed in this study.

5. Tabu search algorithm

Tabu search is one of the well-known meta-heuristic algorithms which guide local search procedure by exploring the solution space of a problem beyond local optimality. This algorithm and its principles were introduced by Glover (1989, 1990) for the first time. In this section, a Multi Start Tabu Search (*MSTS*) algorithm is proposed to solve $F2||\sum T_j^A : C_{max}^B \leq Q$. The main components of this algorithm include different methods to create the neighborhoods, the neighborhood searching type including the number of neighbors and used properties to determine the neighbors, memory structures including tabu list length and aspiration criterion, and finally the condition of terminate search. MSTS creates several initial solutions from solution space at first. For each of these solutions named 'starts', a separate tabu list is created. Tabu search continues to search until reaching the stopping criterion. In each iteration, the algorithm chooses a 'start' probabilistically based on its cost to carry out searching. The best sequence corresponding to this start is replaced with this and the corresponding tabu list is updated. When tabu search is stopped, the best solution among the latest found solutions stored for each 'start' is chosen for a local search.

5.1. Initial solutions

The success of MSTS strictly depends on the initial solutions, and the aim of heuristic algorithms is to disperse these solutions widely on the solution space. To generate initial solutions, at first the second agent's jobs are sorted according to the Johnson's rule (1954). For the first agent's jobs, three orders are considered including the earliest due date (*EDD*) rule, the shortest processing time (*SPT*) rule of sum of job's processing time on both machines, and the random order of jobs (Józefowska et al., 1994). According to these orders, algorithms H1 and H2 are proposed as follows:

Algorithm H1: To generate the initial solution in this method, the sequence is divided into two parts. In the first part, the largest possible number of the first agent's jobs is placed with respect to the feasibility condition of the second agent's jobs. Then, the second agent's jobs are placed after them. The remainder of the first agent's jobs is placed at the end of sequence in the second part. This procedure is performed according to the following steps:

Step 1: Sort the second agent's jobs based on the Johnson's rule (Johnson, 1954) and sort the first agent's jobs according to each of the EDD, SPT and Rnd rules.
Step 2: Select the maximum number of the not sequenced first agent's jobs, such that the not sequenced second agent's jobs can be placed at their last authorized position before Q. Place these selected jobs in the sequence and remove them from the not sequenced first agent's jobs.
Step 3: If the entire first agent's jobs are sequenced, place the entire second agent's jobs after the last sequenced job and go to step 5; otherwise, place the first not sequenced second agent's job after the sequenced jobs in the sequence and remove them from the not sequenced second agent's jobs.
Step 4: If all the second agent's jobs are sequenced, place the not sequenced first agent's job after the last sequenced jobs in the sequence and go to step 5; otherwise go to step 2.
Step 5: Finish.

Algorithm H2: In this algorithm, the sequence is divided into two parts. First, the second agent's jobs are placed in the first part of the sequence and then the first agent's jobs are placed after them.

In the MSTS algorithm, by using H1 and H2 algorithms and considering the three *EDD*, *SPT* and *Rnd* rules as the orders of the first agent's jobs, 6 initial solutions are produced. Also, by applying a local search on any of these 6 solutions, 6 other initial solutions are generated. In the applied local search, the generated initial sequences are improved through interchanging each pair of jobs and using swapping and insertion procedures to be described in the next section. First, all the possible interchanges are checked and then a move with the greatest amount of improvement is performed. This process is continued until no further interchange leads to improvement. Thus, according to the 6 obtained initial solutions, 12 initial solutions are produced for *MSTS* algorithms. As noted, the *MSTS* algorithm selects a 'start' with the corresponding tabu list based on its cost to search in each iteration. Before starting the searching in the algorithm, the initial solutions (starts) are sorted in non-ascending order of their costs. In this order of 'starts', the probability of selecting the *i*th 'start' from among all L 'starts' for searching in current iteration is $(2*i)/L(L+1)$. After selecting a 'start' and completion of the search process, this "start" and corresponding tabu list are updated, and with regard to its cost it is placed in the proper position between all other "starts". Searching algorithm is similarly continued until the stop condition is seen.

5.2. Neighbor generation

The search space is a space of feasible solutions which can be visited during the searching procedure. Also, the neighborhood structure definition depends on the search space. At each iteration of the algorithm, a local moving is performed on the current solution, and neighbor solutions in the search space are generated. In the *MSTS*, the three methods including swapping, insertion and reversion are used to create neighbors. In these methods, two positions of the sequence are randomly chosen. Suppose that job i and job j are corresponding jobs to these positions in the sequence of $(\sigma\ i\ \pi\ j\ \omega)$. In this sequence, σ is a partial sequence of jobs before job i, π is a partial sequence of jobs between job i and job j and ω is a partial sequence of jobs after job j. In the swapping, the sequence is changed to $(\sigma\ j\ \pi\ i\ \omega)$. There are two cases in the insertion; in the first case, job i is transmitted to after job j and the sequence is changed to $(\sigma\ \pi\ j\ i\ \omega)$. In the other case, job j is transmitted to before job i and the sequence is changed to $(\sigma\ j\ i\ \pi\ \omega)$. In the reversion, jobs i and j and their between jobs are reversed and the sequence is changed to $(\sigma\ j\ \pi'\ i\ \omega)$ where π' is a reverse of π.

5.3. Neighbor search

In each iteration of the *MSTS*, a number of neighbors are generated for searching. Some optimal properties of the problem are used to generate these neighbors. As stated before, to create new sequences, two positions are randomly chosen. Then, one of the three methods of swapping, insertion and reversion are selected randomly. After generating a new neighbor, Corollary 1 (local property of Johnson's rule (Johnson, 1954)) is applied to the second agent's jobs. Also according to the proposed properties, neighbors are not produced in the three cases below:

First: If the selected jobs for moving and jobs which are placed between them belong to the second agent, according to Corollary 1, none of the three methods are applied and the neighbor is not created.
Second: If the selected jobs for moving belong to the second agent, and the conditions of Proposition 2 are established for them, swapping method is not applied.
Third: If any of the interchange methods violate the feasibility condition of the second agent, the moving will not be applied and the neighbor not created.

5.4. Memory structure and search terminate condition

From among the tabu search elements which distinguish tabu search from local search, its memory and tabu structures are more important. Tabu structures are used to prevent the cycle and escape of local optimums. As noted, there can be set some conditions for the repeal of tabu structures which are known as aspiration criteria. In *MSTS*, the aspiration criteria allow to accept a tabu movement whose obtained solution is the best solution encountered by the algorithm up to the current iteration. Finally, after a stop condition is met, the best found solution is improved by a local search. Three stop conditions are used in *MSTS,* which can stop the algorithm by establishing each of them. These conditions include the number of iterations without improving the best found solution, reaching a certain maximum iteration, and reaching a zero objective value as the optimal solution during the search process. The pseudo-code of the *MSTS* algorithm is shown in Fig. 3.

```
Input: An instance of size n, parameters L, t, and k.
Output: The best neighbor of S which has the minimum cost among all Ss neighbors encountered by the
algorithm.
Code
1. Begin
2. Set nbr ← Null Sequence; nbr-cost← ∞;
3. Sort the agent A jobs and agent B jobs in EDD and JR, respectively;
4. Generate L initial sequences for the L starts;
5. Sort the L sequences generated in Step 4 in non-decreasing order of their costs;
6. for iter from 1 to k do
7.     Choose a start (S1) probabilistically to carry out a tabu search iteration, for this start;
8.     Obtain the best tabu and non-tabu neighbors of $S_1$ by using neighborhood search procedure;
9.     Select the best neighbor $S_{nbr}$ keeping a check on the aspiration criterion, and update the tabu list using
       tabu tenure t;
10.    Replace $S_1$ by $S_{nbr}$ as the latest sequence for that start. If stop criteria encountered, go to 12;
11. End for;
12. Perform a neighborhood local search on the best sequence among all starts;
13. Obtain the best sequence $S^*$ encountered during the search;
14. Set nbr ← $S^*$; nbr-cost ← cost of $S^*$;
15. Output nbr and nbr-cost;
16. End
```

Fig. 3. the *MSTS* pseudo-code

6. Computational results

In this section, the results of solving some problem instances are provided to evaluate the efficiency of the mathematical programming model and tabu search procedures. The proposed procedures were coded in Visual studio 2013 programming environment and in C# programming language. Also, CPLEX 12.6 was used to solve the mathematical programming model. The time limit to optimally solve the instances was set to 1800 seconds. The proposed procedures have been implemented on a computer with Intel® Core ™ i7-2600M CPU @ 3.4 GHz and 4 GB RAM in Windows 7 with 32-bit operating system.

6.1. Instances

To evaluate the effectiveness of the provided procedures for solving the problem, the number of instances was considered based on the study of Lee et al. (2010), Lee et al. (2011) and Yin et al. (2012ab). For all instances, the processing times of the first agent's jobs and second agent's jobs were randomly generated from a uniform distribution on the interval [1, 10]. The due dates of the first agent's jobs follow a discrete uniform distribution on the interval [δ (1-τ-R/2), δ (1-τ+R/2)], where R and τ are measures of dispersion and tardiness factor, respectively. Also, the value δ is the lower bound of completion time of all jobs and it is equal to the summation of the processing times on the second machine and the minimum processing time of all jobs on the first machine. For parameters R and τ, the values {0.25, 0.50} and {0.25, 0.50, 0.75} are selected, respectively. Also, three values of {0.25, 0.50, 0.75} for the proportions of the first agent jobs to the total jobs (ρ) are considered. According to the study of Yin et al. (2012ab), if L is the lower bound of Q and U is its upper bound, the value of $L + q$ (U-L) is considered in the generated instances as the Q value, where q is one of the three values {0.25, 0.50, 0. 75}. Given the negligible impact of various R values on the performance of the presented methods, the results are not provided for various R values. Therefore, according to the values of τ, ρ and q, the instances were generated in 18(2×3×3) different groups. These groups are named as *G01* to *G18*. For each one of the three intended values of R, 10 instances and therefore 30 instances were generated in each group.

6.2. Evaluation of mathematical models

In this section, the results of solving instances by the mathematical programming model are presented. As noted, some optimum properties were used in this model in order to increase its efficiency. Computational results are presented in Table 1. In this table, the column '*ACT'* shows the results of the basic mathematical model without using optimum properties, and the column '*MACT'* (Modified *ACT*) indicates the results of the mathematical model with the consideration of optimal properties. In this table, the column '*No of Opt'* shows the number of instances which were solved optimally. In this column, if for one of the three values for *R* at least one instance could not reach the optimal solution in 1800 seconds as solution time, solving other instances with this value of *R* and results of the 10 instances corresponding to this value of *R* are not reported. Thus, according to the three values of *R*, values of this field can be 30, 20 and 10. The column '*Avg. CPU time'* represents the mean solution times of instances. Also, the column '*Improve'* shows the percentage value of improvement in solution time by using optimum properties. These values were calculated by the following equation:

$$\text{CPU time improvement} = \frac{(\text{CPU time elapsed by ACT})-(\text{CPU time elapsed by MACT})}{\text{CPU time elapsed by ACT}}*100\% \qquad (14)$$

Table 1
The MIP models results

		ACT		MACT				ACT		MACT				ACT		MACT		
n	**Group**	**No of Opt.**	**Avg. CPU time**	**No of Opt.**	**Avg. CPU time**	**Improve (%)**	**Group**	**No Of Opt.**	**Avg. CPU time**	**No of Opt.**	**Avg. CPU time**	**Improve (%)**	**Group**	**No Of Opt.**	**Avg. CPU time**	**No of Opt.**	**Avg. CPU time**	**Improve (%)**
20		30	0.69	30	0.28	59.42		30	1.21	30	0.69	42.98		30	1.24	30	4.42	-256.45
24		30	1.01	30	0.55	45.54		30	1.91	30	0.84	56.02		30	3.24	30	0.74	77.16
28	***G01***	30	1.87	30	1.08	42.25	***G07***	30	4.35	30	1.61	62.99	***G13***	30	11.72	30	1.18	89.93
32	***ρ=0.25***	30	2.68	30	1.96	26.87	***ρ=0.50***	30	10.86	30	3.43	68.42	***ρ=0.75***	30	11.42	30	2.57	77.50
36	***τ=0.25 q=0.25***	30	4.31	30	2.87	33.41	***τ=0.25 q=0.25***	20	75.56	30	7.43	-	***τ=0.25 q=0.25***	20	17.68	30	3.13	-
40		30	7.94	30	4.32	45.59		10	9.4	20	8.73	-		20	27.18	30	3.40	-
60		-	-	10	66.74	-		-	-	-	-	-		10	33.46	10	26.56	20.62
20		30	0.38	30	0.25	34.21		30	0.85	30	0.50	41.18		30	0.99	30	0.65	34.34
24		30	0.80	30	0.72	10.00		30	1.39	30	0.85	38.85		30	1.82	30	1.15	36.81
28	***G02***	30	1.32	30	0.75	43.18	***G08***	30	2.20	30	1.05	52.27	***G14***	30	2.90	30	2.51	13.45
32	***ρ=0.25***	30	1.67	30	1.52	8.98	***ρ=0.50***	30	3.78	30	2.23	41.01	***ρ=0.75***	30	5.43	30	5.01	7.73
36	***τ=0.25 q=0.50***	30	2.35	30	1.92	18.30	***τ=0.25 q=0.50***	30	5.55	30	3.37	39.28	***τ=0.25 q=0.50***	30	8.29	30	4.51	45.60
40		30	3.52	30	2.92	17.05		30	35.48	30	4.85	86.33		30	43.46	30	21.52	50.48
60		20	30.42	30	21.00	-		-	-	10	79.29	-		-	-	10	26.12	-
20		30	0.27	30	0.18	33.33		30	0.65	30	0.31	52.31		30	0.93	30	0.51	45.16
24		30	0.35	30	0.31	11.43		30	0.81	30	0.49	39.51		30	1.63	30	1.20	26.38
28	***G03***	30	0.70	30	0.61	12.86	***G09***	30	1.40	30	0.91	35.00	***G15***	30	2.08	30	1.41	32.21
32	***ρ=0.25***	30	0.98	30	0.80	18.37	***ρ=0.50***	30	1.81	30	1.51	16.57	***ρ=0.75***	30	2.63	30	2.12	19.39
36	***τ=0.25 q=0.75***	30	1.10	30	1.05	4.55	***τ=0.25 q=0.75***	30	2.26	30	2.26	0.00	***τ=0.25 q=0.75***	30	3.10	30	2.40	22.58
40		30	1.67	30	2.22	-32.93		30	3.78	30	3.04	19.58		30	5.17	30	3.82	26.11
60		30	6.33	30	13.00	-105.37		30	74.41	30	17.89	75.96		20	101.47	20	32.40	68.07
20		30	0.63	30	0.29	53.97		30	1.41	30	0.97	31.21		30	2.01	30	1.59	20.90
24		30	0.99	30	0.58	41.41		30	2.54	30	1.48	41.73		30	7.77	30	3.59	53.80
28	***G04***	30	1.69	30	1.17	30.77	***G10***	30	4.70	30	1.96	58.30	***G16***	30	25.87	30	9.89	61.77
32	***ρ=0.25***	30	2.82	30	2.32	17.73	***ρ=0.50***	30	15.18	30	3.81	74.90	***ρ=0.75***	20	22.40	20	9.88	55.89
36	***τ=0.50 q=0.25***	30	3.89	30	3.34	14.14	***τ=0.50 q=0.25***	30	15.46	30	5.11	66.95	***τ=0.50 q=0.25***	-	-	20	32.70	-
40		30	6.05	30	4.17	31.07		-	-	30	7.48	-		-	-	10	14.26	-
60		-	-	30	38.00	-		-	-	10	475.85	-		-	-	-	-	-
20		30	0.39	30	0.26	33.33		30	1.34	30	0.60	55.22		30	2.66	30	1.57	40.98
24		30	0.69	30	0.46	33.33		30	2.24	30	0.86	61.61		30	9.85	30	9.21	6.50
28	***G05***	30	1.30	30	0.95	26.92	***G11***	30	2.91	30	1.74	40.21	***G17***	30	34.83	30	14.11	59.49
32	***ρ=0.25***	30	1.81	30	1.50	17.13	***ρ=0.50***	30	5.42	30	5.03	7.20	***ρ=0.75***	20	15.37	30	60.92	-
36	***τ=0.50 q=0.50***	30	2.69	30	2.07	23.05	***τ=0.50 q=0.50***	30	10.14	30	4.19	58.68	***τ=0.50 q=0.50***	20	15.00	30	7.66	-
40		30	4.25	30	3.38	20.47		30	24.62	30	5.85	76.24		20	43.11	20	15.93	63.05
60		30	38.81	30	27.00	30.43		-	-	10	50.83	-		-	-	10	152.15	-
20		30	0.33	30	0.22	33.33		30	1.14	30	0.60	47.37		30	7.38	30	4.12	44.17
24		30	0.50	30	0.32	36.00		30	2.05	30	1.90	7.32		30	38.75	30	51.76	-33.57
28	***G06***	30	0.73	30	0.68	6.85	***G12***	30	2.26	30	1.20	46.90	***G18***	10	65.14	30	130.41	-
32	***ρ=0.25***	30	1.28	30	0.98	23.44	***ρ=0.50***	30	4.76	30	5.48	-15.13	***ρ=0.75***	20	153.28	20	55.54	63.77
36	***τ=0.50 q=0.75***	30	1.63	30	1.38	15.34	***τ=0.50 q=0.75***	30	6.05	30	3.51	41.98	***τ=0.50 q=0.75***	-	-	-	-	-
40		30	2.64	30	2.52	4.55		30	6.83	30	5.08	25.62		-	-	-	-	-
60		30	10.06	30	20.00	-98.81		30	126.43	30	33.15	73.78		-	-	-	-	-

Results indicate that the *MACT* is more efficient than *ACT* in most cases. The *MACT* is able to solve all the instances containing up to 40 jobs in size, in 92.56% of groups and instances having up to 60 jobs in size, in 84.92% of groups. These results for *ACT* are equivalent to 86.11% and 77.77%, respectively. Also, mean percentage of improvement in solution time is equal to 30.40%. According to the results, when the number of first agent's jobs increases, more time is required to solve instances.

6.3. Evaluation of the performance of Tabu search algorithm

In order to evaluate the performance of the tabu search algorithm, the error of its solutions according to the optimum solutions was calculated. In the following, the tuned parameters of the algorithm along with the results are presented.

6.3.1. The MSTS tuned parameters

In this section, parameters of the MSTS are tuned by using the Taguchi method. The tuned parameters include the number of examined neighbours in each iteration, the tabu list length and the number of iterations without any improvement as a stop condition. As mentioned, there are 4 parameters, and each one has 3 levels that showed in Table 2. Thus, the proper orthogonal array was selected by Minitab Software and the L9 orthogonal array (OA) was used.

Table 2
Factors levels

Factors	Levels
A: The number of examined neighbours in each iteration	n/2, n, 2n
B: The tabu list length	n/4, n/2, n
C: The number of iterations without any improvement	40n, 50n, 60n
D: The maximum number of iteration	8000, 10000, 12000

The average mean and S/N ratio at each level for relative deviation (RD) values are shown in Fig. 4 and Fig. 5, respectively.

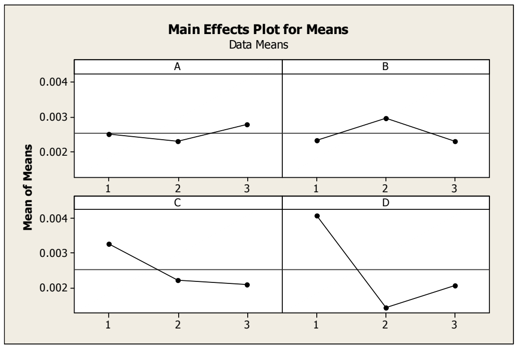

Fig. 4. Main effects plot for means

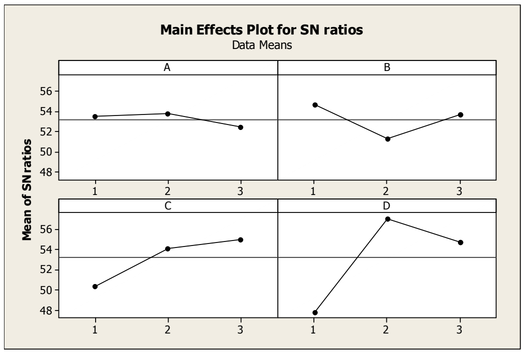

Fig. 5. Main effects plot for SN ratios

As indicated in these figures, better robustness of the algorithm is achieved when the number of examined neighbours in each iteration set as *n* and the maximum number of iteration set as 10000. For parameter "the number of iterations without any improvement", whether 40*n* or 60*n* can be selected and for increasing effectiveness of MSTS, this parameter set as 60*n*. Finally, for parameter "the tabu list length", two level *n*/2 and *n* can be selected, too. To increase efficiency and saving time, the length of the tabu list was set to *n*/2. In Table 3, the selected values for the parameters of the algorithm are presented.

Table 3
The MSTS parameters tuning results

No of Neighbor	Tabu list length	Not Improve	Max iteration
n	n/2	60n	10000

6.3.2. Absolute error

The absolute error percentage (*AEP*) of the *MSTS* algorithm is calculated according to Eq. (15).

$$AEP = \frac{(Optimal\ Obj.\ Value) - \ (MSTS\ Obj.Value)}{Optimal\ Obj\ Value} * 100\% \tag{15}$$

The values of average absolute error percentage *(AEP)* of the *MSTS* algorithm are presented in Table 4. These values are obtained by comparing the solution of *MSTS* algorithm with the optimum solutions of *MACT* model. In Table 4, '*No of Inst.*' column shows the number of instances which were solved by *MSTS* algorithm. The results show that *AEP* and the average solution time of the *MSTS* algorithm for solved problems are 0.08% and 26.16 seconds, respectively.

Table 4
The AEP for the MSTS algorithm

		MSTS					**MSTS**					**MSTS**			
n	**Group**	**No of Inst.**	**Avg. CPU time**	**AEP**		**Group**	**No of Inst.**	**Avg. CPU time**	**AEP**		**Group**	**No of Inst.**	**Avg. CPU time**	**AEP**	
				Avg.	**max**				**mean**	**max**				**mean**	**max**
20		30	0.08	0	0		30	0.08	0	0		30	4.74	0	0
24		30	0.15	0	0		30	0.14	0	0		30	5.13	0.29	8.57
28	***G01 ρ=0.25 τ=0.25 q=0.25***	30	0.25	0	0	***G07 ρ=0.50 τ=0.25 q=0.25***	30	0.26	0.31	6.58	***G13 ρ=0.75 τ=0.25 q=0.25***	30	5.53	0.04	1.23
32		30	0.40	0	0		30	0.42	0.01	0.21		30	5.94	0.09	2.02
36		30	0.64	0	0		30	0.70	0.31	4.46		30	6.35	0.06	1.14
40		30	1.92	0	0		20	1.06	0.12	2.42		30	6.84	0	0
60		10	4.62	0	0		-	-	-	-		10	20.08	0.11	0.90
20		30	0.08	0	0		30	0.08	0	0		30	7.07	0	0
24		30	0.14	0	0		30	0.14	0.14	4.26		30	7.45	0.23	5.71
28	***G02 ρ=0.25 τ=0.25 q=0.50***	30	0.25	0	0	***G08 ρ=0.50 τ=0.25 q=0.50***	30	0.25	0	0	***G14 ρ=0.75 τ=0.25 q=0.50***	30	7.84	0.16	2.53
32		30	0.35	0	0		30	0.37	0.16	4.88		30	8.24	0.20	2.75
36		30	0.60	0	0		30	0.57	0.10	1.55		30	8.63	0	0
40		30	0.86	0	0		30	0.80	0	0		30	9.13	0.20	2.46
60		30	4.48	0.02	0.72		10	5.99	0.05	0.46		10	27.08	0.15	0.90
20		30	0.06	0	0		30	0.07	0	0		30	9.40	0.09	2.70
24		30	0.13	0	0		30	0.13	0	0		30	9.78	0.05	1.43
28	***G03 ρ=0.25 τ=0.25 q=0.75***	30	0.24	0	0	***G09 ρ=0.50 τ=0.25 q=0.75***	30	0.23	0	0	***G15 ρ=0.75 τ=0.25 q=0.75***	30	10.18	0.03	1.01
32		30	0.34	0	0		30	0.33	0	0		30	10.55	0	0
36		30	0.45	0	0		30	0.46	0	0		30	10.93	0	0
40		30	0.74	0	0		30	0.68	0	0		30	11.36	0.03	0.82
60		30	3.23	0	0		30	3.44	0.16	3.73		20	18.93	0.05	0.95
20		30	0.08	0	0		30	0.08	0.17	2.75		30	11.75	0.05	1.58
24		30	0.15	0	0		30	0.16	0.05	1.36		30	12.16	0.33	2.01
28	***G04 ρ=0.25 τ=0.50 q=0.25***	30	0.26	0	0	***G10 ρ=0.50 τ=0.50 q=0.25***	30	0.26	0.08	1.96	***G16 ρ=0.75 τ=0.50 q=0.25***	30	12.60	0.25	4.26
32		30	0.41	0	0		30	0.44	0.03	0.78		20	19.32	0.06	0.92
36		30	0.65	0.05	1.16		30	0.74	0.27	3.84		20	20.01	0.14	1.78
40		30	0.96	0	0		30	1.12	0	0		10	40.08	0.04	0.30
60		30	5.03	0	0		10	6.41	0.15	0.75		-	-	-	-
20		30	0.08	0	0		30	0.08	0.45	12.94		30	13.75	0.15	3.85
24		30	0.15	0	0		30	0.46	0.06	1.78		30	14.15	0.20	2.15
28	***G05 ρ=0.25 τ=0.50 q=0.50***	30	0.25	0	0	***G11 ρ=0.50 τ=0.50 q=0.50***	30	0.87	0.16	4.71	***G17 ρ=0.75 τ=0.50 q=0.50***	30	14.59	0.16	1.90
32		30	0.39	0	0		30	1.33	0.39	6.40		30	15.07	0.38	6.76
36		30	0.65	0	0		30	1.80	0.01	0.34		30	15.54	0.20	1.42
40		30	0.96	0	0		30	2.38	0.12	1.44		20	23.65	0.35	2.76
60		30	4.78	0.28	4.21		10	6.08	0.32	1.53		10	47.08	0.33	0.96
20		30	0.06	0	0		30	2.41	0	0		30	16.08	0.68	6.48
24		30	0.13	0	0		30	2.79	0	0		30	16.50	0.44	5.63
28	***G06 ρ=0.25 τ=0.50 q=0.75***	30	0.24	0	0	***G12 ρ=0.50 τ=0.50 q=0.75***	30	3.19	0	0	***G18 ρ=0.75 τ=0.50 q=0.75***	30	16.93	0.79	13.92
32		30	0.37	0	0		30	3.61	0	0		20	25.78	0.08	1.29
36		30	0.54	0	0		30	4.11	0	0		-	-	-	-
40		30	0.90	0	0		30	4.64	0.54	16.23		-	-	-	-
60		30	4.50	0	0		30	7.39	0.20	3.70		-	-	-	-

Fig. 6 shows the *AEP* of the *MSTS* algorithm for the proportion of the first agent's jobs (*ρ)*. The results indicate that this value decreases by increasing the number of the second agent's jobs. This is because of applying Corollary 1 (local property of Johnson's rule (Johnson, 1954)) in searching procedure. When the number of second agent's jobs increases, according to Corollary 1, the optimal order of the second agent's adjacent jobs is Johnson's order that reduces the search space. However, for the first agent's adjacent jobs a predefined sequence does not exist. So, when the first agent's job is increased, the solution space and the number of its points, which the search algorithm fails to investigate, are increased and this leads to the increase in the value of *AEP*.

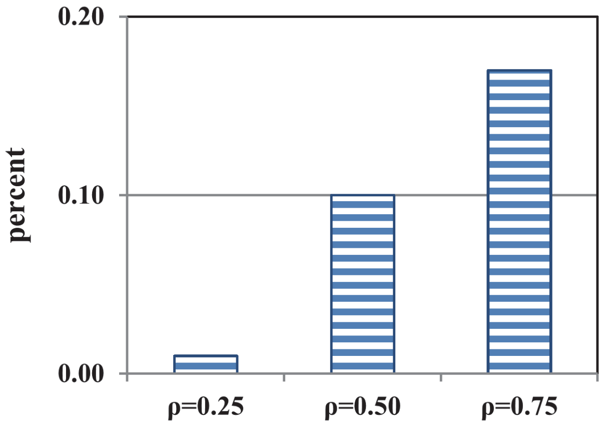

Fig. 6. The AEP of the MSTS for *ρ* values

7. Conclusion

In this paper, the two-agent scheduling problem in a two-machine flowshop with the aim of minimizing the total tardiness of first agent's jobs under the bounded makespan of the second agent's jobs was studied. A mathematical programming model with the theorems and useful properties were used to optimally solve the problem. Computational results showed that the presented mathematical programming model is able to solve all the instances in 84.56% of groups with up to 60 jobs in size. Also, the computational results indicated that the proposed tabu search algorithm had high efficiency according to the average absolute error value which is less than 0.18%. Future research line related to this problem could involve feasibility optimization, Pareto optimization and optimization of the weighted sum of the agent's objective. Also, because few studies have been done on the multi-agent scheduling in the flowshop environment, the important problems not solved in this field of scheduling are recommended as further areas of inquiry.

References

Agnetis, A., Billaut, J.-C., Gawiejnowicz, S., Pacciarelli, D., & Souhal, A. (2014). *Multi-agent scheduling*. Berlin: Springer Berlin Heidelberg.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling Problems with Two Competing Agents. *Operations Research, 52*(2), 229-242. doi:10.2307/30036575

Agnetis, A., Pacciarelli, D., & Pacifici, A. (2007). Multi-agent single machine scheduling. *Annals of Operations Research, 150*(1), 3-15. doi:10.1007/s10479-006-0164-y

Akkan, C., & Karabatı, S. (2004). The two-machine flowshop total completion time problem: Improved lower bounds and a branch-and-bound algorithm. *European Journal of Operational Research, 159*(2), 420-429.

Baker, K. R., & Smith, J. C. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling, 6*(1), 7-16.

Chandra, P., Mehta, P., & Tirupati, D. (2009). Permutation flow shop scheduling with earliness and tardiness penalties. *International Journal of Production Research, 47*(20), 5591-5610.

Cheng, T. E., Wu, W.-H., Cheng, S.-R., & Wu, C.-C. (2011). Two-agent scheduling with position-based deteriorating jobs and learning effects. *Applied Mathematics and Computation, 217*(21), 8804-8824.

Della Croce, F., Ghirardi, M., & Tadei, R. (2002). An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. *European Journal of Operational Research, 139*(2), 293-301.

Della Croce, F., Narayan, V., & Tadei, R. (1996). The two-machine total completion time flow shop problem. *European Journal of Operational Research, 90*(2), 227-237.

Fan, B. Q., & Cheng, T. C. E. (2016). Two-agent scheduling in a flowshop. *European Journal of Operational Research, 252*(2), 376-384. doi:10.1016/j.ejor.2016.01.009

Gajpal, Y., Dua, A., & Sahu, S. N. (2014). Heuristics for single machine scheduling under competition to minimize total weighted completion time and makespan objectives. *Lecture Notes in Management Science, 6*, 99-105.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing, 1*(3), 190-206.

Glover, F. (1990). Tabu search—part II. *ORSA Journal on computing, 2*(1), 4-32.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics, 5*, 287-326.

Haouari, M., & Kharbeche, M. (2013). An assignment-based lower bound for a class of two-machine flow shop problems. *Computers & Operations Research, 40*(7), 1693-1699.

Hoogeveen, J., & Velde, S. v. d. (1995). Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. *Operations Research Letters*, 205-208.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research Logistics Quarterly, 1*(1), 61-68.

Józefowska, J., Jurisch, B., & Kubiak, W. (1994). Scheduling shops to minimize the weighted number of late jobs. *Operations Research Letters, 16*(5), 277-283.

Kellerer, H., & Strusevich, V. A. (2010). Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica, 57*(4), 769-795.

Kharbeche, M., & Haouari, M. (2012). MIP models for minimizing total tardiness in a two-machine flow shop. *Journal of the Operational Research Society, 64*(5), 690-707.

Kim, Y.-D. (1993). A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. *Computers & Operations Research, 20*(4), 391-401.

Lee, W.-C., Chen, S.-K., Chen, C.-W., & Wu, C.-C. (2011). A two-machine flowshop problem with two agents. *Computers & Operations Research, 38*(1), 98-104. doi:10.1016/j.cor.2010.04.002

Lee, W.-C., Chen, S.-k., & Wu, C.-C. (2010). Branch-and-bound and simulated annealing algorithms for a two-agent scheduling problem. *Expert Systems with Applications, 37*(9), 6594-6601.

Lei, D. (2015). Variable neighborhood search for two-agent flow shop scheduling problem. *Computers & Industrial Engineering, 80*, 125-131. doi:http://dx.doi.org/10.1016/j.cie.2014.11.024

Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics, 1*, 343-362.

Leung, J. Y. T., Pinedo, M., & Wan, G. (2010). Competitive Two-Agent Scheduling and Its Applications. *Operations Research, 58*(2), 458-469. doi:10.2307/40605929

Liu, P., Yi, N., Zhou, X., & Gong, H. (2013). Scheduling two agents with sum-of-processing-times-based deterioration on a single machine. *Applied Mathematics and Computation, 219*(17), 8848-8855.

Luo, W., Chen, L., & Zhang, G. (2012). Approximation schemes for two-machine flow shop scheduling with two agents. *Journal of Combinatorial Optimization, 24*(3), 229-239.

M'Hallah, R. (2014). Minimizing total earliness and tardiness on a permutation flow shop using VNS and MIP. *Computers & Industrial Engineering, 75*, 142-156.

Pan, J. C.-H., Chen, J.-S., & Chao, C.-M. (2002). Minimizing tardiness in a two-machine flow-shop. *Computers & Operations Research, 29*(7), 869-885.

Pan, J. C.-H., & Fan, E.-T. (1997). Two-machine flowshop scheduling to minimize total tardiness. *International Journal of Systems Science, 28*(4), 405-414.

Perez-Gonzalez, P., & Framinan, J. M. (2014). A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research, 235*(1), 1-16. Pinedo, M. (2002). *Scheduling-Theory, Algorithms, and Analysis*. New Jersey: Prentice-Hall.

Schaller, J. (2005). Note on minimizing total tardiness in a two-machine flowshop. *Computers & Operations Research, 32*(12), 3273-3281.

Sen, T., Dileepan, P., & Gupia, J. N. (1989). The two-machine flowshop scheduling problem with total tardiness. *Computers & Operations Research, 16*(4), 333-340.

Shiau, Y.-R., Tsai, M.-S., Lee, W.-C., & Cheng, T. C. E. (2015). Two-agent two-machine flowshop scheduling with learning effects to minimize the total completion time. *Computers & Industrial Engineering, 87*, 580-589.

Wu, C.-C., Huang, S.-K., & Lee, W.-C. (2011). Two-agent scheduling with learning consideration. *Computers & Industrial Engineering, 61*(4), 1324-1335.

Wu, W.-H. (2014). Solving a two-agent single-machine learning scheduling problem. *International Journal of Computer Integrated Manufacturing, 27*(1), 20-35.

Wu, W.-H., Xu, J., Wu, W.-H., Yin, Y., Cheng, I. F., & Wu, C.-C. (2013). A tabu method for a two-agent single-machine scheduling with deterioration jobs. *Computers & Operations Research, 40*(8), 2116-2127. doi:10.1016/j.cor.2013.02.025

Yin, Y., Cheng, S.-R., & Wu, C.-C. (2012a). Scheduling problems with two agents and a linear non-increasing deterioration to minimize earliness penalties. *Information Sciences, 189*, 282-292. doi:10.1016/j.ins.2011.11.035

Yin, Y., Wu, W.-H., Cheng, S.-R., & Wu, C.-C. (2012b). An investigation on a two-agent single-machine scheduling problem with unequal release dates. *Computers & Operations Research, 39*(12), 3062-3073.

Appendix

In this section, other theorems and optimal properties of the problem are proposed. These properties were used in a branch and bound algorithm and increased its efficiency. For the case in which both jobs belong to the first agent, theorems 3 to 6 for the two-agent scheduling problem are considered.

Theorem 3 (Pan & Fan, 1997)**:** For any two adjacent jobs i and j, if $a_i \leq min\{a_j, b_i\}$, $b_i \leq b_j$ and $d_i \leq d_j$, then there exists an optimal schedule such that job j is sequenced after job i.

Theorem 4 (Schaller, 2005): For any two adjacent jobs i and j, if $C_j(S') \leq d_j$ and $C_j(S') \leq C_i(S)$, then there exists an optimal schedule such that job j is processed after job i.

Theorem 5 (Schaller, 2005): For any two jobs i and j, if $C_i(S') \geq d_i$, $C_i(S') \leq C_j(S)$ and $a_i \leq min\{a_j, b_i\}$, then there exists an optimal schedule such that job j is processed after job i.

Theorem 6 (Schaller, 2005): For two adjacent jobs i and j, if $C_i(S') \geq d_i$, $C_i(S') \leq C_j(S)$, $a_i \leq a_j$ and $b_j \leq b_i$, then there exists an optimal schedule such that job j is processed after job i.

Proposition **4** and Proposition 5 were developed in this research for the second agent's jobs. With regard to Q as the upper bound for the makespan of the second agent's jobs, proposition 4 is used to determine the feasibility of sequence (σ, σ'), where σ and σ' denote the scheduled and unscheduled jobs in the sequence, respectively.

Proposition 4: If jobs belong to the second agent in σ', assigned after σ according to the Johnson's rule (Johnson, 1954) and to be completed after Q, then the sequence (σ, σ') is not a feasible sequence.
Proof: Obvious. ■

Proposition 5: For the two jobs i and j belonging to the second agent, if conditions $a_i \leq a_j$, $b_i \geq b_j$ and $C_i(S') \leq C_j(S)$ are met, then there exists an optimal schedule such that j is processed after i.
Proof: According to Fig. 2, if the first condition and the third condition are met, by interchanging jobs i and j in S, the completion times of jobs between jobs i and j do not increase. Thus, because of no increase in the completion times of jobs between jobs i and j, similar to the proof of Proposition 2, the relation $C_j(S') - C_i(S) \leq b_j - b_i$ is established. Therefore, according to the second condition, the completion times of subsequent jobs do not increase. ■

When one of the two jobs belongs to the first agent and the other belongs to the second agent, the following propositions are proved in this study.

Proposition 6: For job $i \in J_A$ and job $j \in J_B$, if conditions $a_i^A \leq a_j^B$, $b_i^A = b_j^B$ and $C_j^B(S') \leq Q$ are met, there exists an optimal schedule that j is processed after i.

Proof: By interchanging jobs i and j in S, the tardiness of job i does not increase. Because of the first and second conditions and according to the proof of Proposition 2, the completion times of jobs between of jobs *i* and *j* do not increase. Finally, the third condition guarantees the feasibility of schedule. ■

Proposition 7: For job $i \in J_A$ and job $j \in J_B$, if conditions $a_i^A \leq a_j^B$, $b_i^A \geq b_j^B$, $C_i^A(S') \leq C_j^B(S)$ and $C_j^B(S') \leq Q$ are met, there exists an optimal schedule that *j* is processed after *i*.

Proof: By interchanging jobs *i* and *j* in *S*, the tardiness of job *i* does not increase. Because of the first and third conditions the completion times of jobs between of jobs *i* and *j* do not increase, and similar to the proof of Proposition 2, the relation $C_j^B(S') - C_i^A(S) \leq b_j^B - b_i^A$ is established. Thus, according to the second condition the completion times of subsequent jobs do not increase. Finally, the fourth condition guarantees the feasibility of schedule. ■

Proposition 8: For job i $\in J_A$ and job j $\in J_B$, if conditions $a_i^A \leq a_j^B$, $b_i^A \geq b_j^B$, $a_i^A + b_i^A \leq a_j^B + b_j^B$ and $C_j^B(S') \leq Q$ are met, there exists an optimal schedule such that i is not the first job of the sequence.

Proof: With regard to conditions from the first to the third, the procedure of proof is similar to the proof of proposition 3 and the fourth condition guarantees the feasibility of schedule. ■

Proposition 9: For job $i \in J_A$and job $j \in J_B$, if conditions $C_j^B(S') \leq Q$ and $C_j^B(S') \leq C_i^A(S)$ are met, there exists an optimal schedule such that *j* is processed after *i*.

Proof: By interchanging jobs *i* and *j* in *S*, the tardiness of job *i* does not increase. Also, the first condition guarantees that the completion times of subsequent jobs do not increase and the second condition guarantees the feasibility of schedule after interchanging jobs *i* and *j* in *S*. ■

Proposition 10: For job $i \in J_A$ and job $j \in J_B$, if conditions $C_j^A(S') \leq d_j^A$ and $C_j^A(S') \leq C_i^B(S)$ are met, there exists an optimal schedule such that *j* is processed after *i*.

Proof: The first condition states that by interchanging *j* in *S*, this job remains without tardiness. Also, the second condition guarantees that the completion times of subsequent jobs do not increase after interchanging jobs *i* and *j* in *S*. ■

Proposition 11: For job $i \in J_A$ and job $j \in J_B$, if conditions $a_i^B \leq a_j^A$, $b_i^B = b_j^A$ and $C_j^A(S') \leq d_j^A$ are met, there exists an optimal schedule such that *j* is processed after *i*.

Proof: By interchanging jobs *i* and *j* in *S*, because of the first and second conditions and similar to the proof of Proposition 2 the completion times of jobs between and subsequent of jobs *i* and *j* do not increase. Also, because of condition $C_j^A(S') \leq d_j$ by interchanging jobs *i* and *j* in *S*, job *j* remains without tardiness. ■

Proposition 12: For job i $\in J_A$ and job j $\in J_B$, if conditions $a_i^B \leq a_j^A$, $b_i^B \geq b_j^A$, $C_i^B(S') \leq C_j^A(S)$and $C_j^A(S') \leq d_j$ are met, then there exists an optimal schedule such that j is processed after i.

Proof: By interchanging jobs *i* and *j* in *S*, because of the first and third conditions the completion times of jobs between jobs *i* and *j* do not increase and because of the second condition, similar to the proof of

Proposition **2**, the completion times of subsequent jobs do not increase either. Also, condition $C_j^A(S') \leq d_j$ states that tardiness of job *j* in *S* and S' is zero. ■