Manager’s Preferences Modeling Within Multi-Criteria Flowshop Scheduling Problem: A Metaheuristic Approach

Mohamed Anis Allouche

Abstract

This paper proposes a metaheuristic to solve the permutation flow shop scheduling problem where several criteria are to be considered, i.e., the makespan, total flowtime and total tardiness of jobs. The proposed metaheuristic is based on tabu search algorithm. The Compromise Programming model and the concept of satisfaction functions are utilized to integrate explicitly the manager’s preferences. The proposed approach has been tested through a computational experiment. This approach can be useful for large scale scheduling problems and the manager can consider additional scheduling criteria.

Keywords: Permutation Flowshop, Multi-Criteria Scheduling, Compromise Programming, Satisfaction Functions, manager’s Preferences.

1. INTRODUCTION

The aim of the scheduling theory is the allocation of a set of limited resources to process a given number of jobs (MacCarthy and Liu,[20]). More specifically, a scheduling problem consists of finding the sequence of a set of jobs (tasks) to be processed on different machines, so that technological constraints are satisfied and one or several performance criteria are optimized (T’kindt and Billaut, [23]). The scheduling literature reveals that several criteria should be considered to present more realistic solutions to the production manager. However, it is generally impossible to find a sequence that optimizes simultaneously different conflicting scheduling criteria. Thus, the manager must consider the sequence of the best compromise. Hence, the manager needs to make some trade-offs between the scheduling criteria. Thus, the obtained solution will be a satisfactory solution.

Several approaches and models are proposed to solve the scheduling problem, namely discrete variable mathematical programming, simulation techniques and network analysis. Specific algorithms and heuristics have been utilized to deal with the scheduling problem. The choice of an appropriate approach depends on the complexity of the problem, the number of the jobs to be scheduled, the configuration of the machines, the production system and the nature of the job arrivals (static or dynamic). For example, Gangadhran and Rajendran [10] have applied the Simulated Annealing technique to minimize the makespan and the total flow time (ΣC). Kondakci et al. [17] have utilized the shortest processing time and the earliest due date rules to minimize the total flow time and the maximum tardiness penalties. We also notice many other algorithms have been developed to deal with multi-criteria scheduling problems.

Gupta et al. [16] have proposed some heuristics to solve a bi-criteria scheduling problem. Arroyo and Armentano [5] have proposed a partial enumeration heuristic for multi-objective flowshop scheduling problem where they provide the manager with approximate Pareto optimal solutions.
Their heuristic offers a set of feasible solutions and the manager's preferences are partially considered according to a posteriori articulation.

Armentano and Arroyo [4] have proposed a new tabu search algorithm for multi-objective combinatorial problems with the aim of obtaining a good approximation of the Pareto-optimal or efficient solutions. A nice feature of this multi-objective algorithm is that it introduces only one additional parameter, namely, the number of paths. This algorithm is applied to the permutation flowshop scheduling problem in order to minimize the criteria of makespan and maximum tardiness. For instances involving two machines, the performance of the algorithm is tested against the Branch-and-Bound enumeration algorithm, and for more than two machines it is compared with that of a tabu search algorithm and a genetic local search algorithm, both from the literature. Computational results show that the heuristic proposed by Armentano and Arroyo [4] yields to a better approximation than these algorithms.

Allahverdi [1] was interested in a machine flowshop problem with the objective of minimizing a weighted sum of makespan and maximum tardiness. Varadharajan and Rajendran [26] presented a multi-objective simulated annealing algorithm (MOSA) for permutation flowshop scheduling to minimise the makespan and total flowtime for jobs. The MOSA seeks to obtain non-dominated solutions through the implementation of a simple probability function that attempts to generate solutions on the Pareto-optimal front. Framinan and Leisten [7] tackle the problem of total flowtime and makespan minimisation in a permutation flowshop. The authors have introduced a multi-criteria iterated greedy search algorithm. Their algorithm iterates over a multi-criteria constructive heuristic approach to yield a set of Pareto-efficient solutions.

Loukil et al. [19] have adapted the MOSA to solve a multi-criteria flowshop scheduling problem. Lemesre et al. [18] proposed an exact method to solve a bi-criteria scheduling problem named the parallel partitioning method. This method allows the generation of efficient solutions. According to Lemesre et al [18], their method requires less CPU time comparatively to the two phases method of Ulungu and Teghem [25]. In their book, T’kindt and Billaut [23] present a quite complete literature review regarding multi-criteria scheduling theory.

Gagné et al. [9] have proposed a generic approach to finding compromise solutions for multi-objective scheduling problems using metaheuristics. As an illustration, they present a new hybrid Tabu Search/Variable Neighbourhood Search (Tabu-VNS) application of this approach for the solution of a bi-objective scheduling problem. Through numerical experiments they demonstrate its efficiency and effectiveness. They have confirmed that compromise programming with the Tabu-VNS metaheuristic generates solutions that approach those of the known reference sets. Gagné et al. [8] presented an adjustment of an Ant Colony Optimization for an eventual use in a generic research procedure of compromise solutions for single machine scheduling problem.

Our literature review of multi-criteria scheduling problems shows that a large number of the proposed approaches to solve the flow shop scheduling problems do not take into account explicitly the manager’s preferences. However, Aouni et al. [3] and Allouche et al. [2] have developed an aggregation procedure that considers three different criteria to obtain the best sequence in a flowshop production environment. The authors utilize the compromise programming model and the concept of satisfaction functions to integrate explicitly the manager’s preferences. The satisfaction functions measure the intensity of preference regarding the deviations between the achievement and the aspiration levels of the following criteria: makespan, total flow time and total tardiness. Their procedure is easy to apply and it requires few parameters (thresholds) to be provided by the manager. The satisfaction functions thresholds have a specific economic interpretation that the manager can understand and provide the values. The proposed model can be extended to introduce additional criteria. This model provides the best scheduling sequence that satisfies the manager’s preferences. However, this approach is sensitive to the size of the scheduling problem to be solved. It requires a large computational time for the large scale problems. To deal with such difficulty, we recommend the use of metaheuristics which is the purpose of this paper.
The aim of this paper is to propose a metaheuristic based on the tabu search algorithm to solve a multi-criteria scheduling problem. The new proposed approach will explicitly incorporate the structure of manager’s preferences with the use of satisfaction function concept.

This paper is organised as follows. The description of the proposed metaheuristic is given in the second section. In fact, this metaheuristic is based on three components which are the compromise programming model, the concept of satisfaction functions and the tabu search algorithm. The third section presents the different steps of the proposed metaheuristic. These steps are useful for obtaining the sequence of the best compromise. The computational experiments and results are summarized in the fourth section.

2. METAHEURISTIC COMPONENTS

Within this section, we will describe the proposed metaheuristic. This metaheuristic is based on the three following components: a) the Compromise Programming model, b) the concept of Satisfaction Functions, and c) the tabu search algorithm.

2.1. Compromise Programming Model

The Compromise Programming model (CP) was introduced first by Zeleny [29]. This model is based on the minimization of the distance between the achievement level \( f_q(x) \) of objective \( q \) and the ideal value \( g_q^* \) of this objective. This model is based on the Zeleny’s axiom of choice where the solutions that are closer to the ideal points \( g_q^* \) are preferred to those that are farther (Zeleny, [27], [28]).

2.2. Satisfaction Function in Compromise Programming Model

In this section we will utilize the concept of the satisfaction functions to formulate a scheduling model where the manager’s preferences are explicitly incorporated. Martel and Aouni [21] have introduced the concept of satisfaction functions in the Goal Programming (GP) model. Through this concept, the manager can explicitly express his/her preferences regarding the unwanted deviations between the achievement and the aspiration levels associated to the different objectives. Figure 1 illustrates the general shape of the satisfaction functions.

![FIGURE1: General shape of satisfaction function](image)

where:

- \( F_q(\delta_q) \) : value of satisfaction function for deviation \( \delta_q \),
- \( \alpha_{qd} \) : indifference threshold,
- \( \alpha_{qo} \) : nil satisfaction threshold,
- \( \alpha_{qv} \) : veto threshold.
The satisfaction functions measure the intensity of the manager’s preferences regarding the unwanted deviations between the achievement level \( f_q(x) \) and the ideal value \( g^*_q \) (\( \forall q \in Q \)). The intensity of preference \( F_q(\delta_q) \) for each objective is defined on the interval \([0; 1]\). Thus, there is no need for computing the nadir point that usually used for the normalization procedure in the CP model. In their paper, Martel and Aouni [21] propose different shapes that the manager can adopt to elucidate explicitly his/her preferences. The manager can choose or adapt one of functions presented by figure2.

This list of satisfaction functions neither exhaustive nor restrictive. The manager will adopt the one that reflects better and accurately his/her preferences. For the purpose of illustration, we will utilize the satisfaction function of type (d) (Fig. 2). This function will be applied to the three criteria (makespan, total flowtime and total tardiness) that we are considering in our computational experiment.

![Different shapes of satisfaction functions](image.png)

**FIGURE2**: Different shapes of satisfaction functions

### 2.3. The tabu search algorithm

The tabu search algorithm has been developed by Glover [11] and it was the first general framework for modern heuristic search. This algorithm is considered as a general iterative approach of combinatorial optimization. The details of the tabu search algorithm are available in the following references: Glover ([12],[13]), Glover et al. [15] and Glover and Laguna [14]. In this paper, through a new metaheuristic based on tabu search algorithm, we look for a best scheduling sequence that takes into account the manager’s preferences. The details of each step of this metaheuristic will be presented in the next section.

### 3. METAHEURISTIC STEPS

In this section, we will present the different steps of the proposed metaheuristic for multi-criteria scheduling problem. The first step determines the lower bounds of each criterion and the second step consists of determining the best compromise sequence.
3.1. Step 1: Finding the Best Values
We consider the best values as the ideal points obtained by optimizing each criterion separately. For this, a metaheuristic based on a tabu search algorithm was developed and has the following characteristics:

- **Initial solution**: randomly generated;
- **Neighbourhood structure**: we consider all permutation as a solution. The permutation’s neighbourhood is created by a set of moves. Given a sequence $s$, let $i$ and $j$ be two positions in this sequence $s$. A neighbour of $s$ is obtained by interchanging the jobs in positions $i$ and $j$ which are randomly selected.
- **Selection of the best neighbour**: the objective function is to minimize the makespan, the total flowtime and the total tardiness. In fact, we define best by reference to the objective function’s value. The best neighbour has to be non tabu or tabu and satisfying the aspiration criteria.
- **Tabu list**: the size of tabu list is fixed at 7. This means that the tabu list contains 7 prohibited moves. Note that this list is managed in a circular manner.
- **Aspiration criteria**: we used the simplest form of aspiration criterion which is stated as follows: a tabu move is accepted if it produces a solution better than the best obtained so far.
- **Stopping criteria**: the algorithm is stopped when it reaches a number of iterations without improvement of the evaluation function. This parameter is fixed in advance.

The developed metaheuristic was tested through a set of problems used by Taillard [24] and are in “OR library” [22]. These problems have different sizes such as 20 jobs (5-10-20 machines), 50 jobs (5-10 machines) and 100 jobs (5 machines). For each size of problem, we have tested 10 different instances.

In order to evaluate the performance of the developed metaheuristic, we are used Taillard’s benchmarks for the makespan criterion. The obtained results are quiet similar to those presented by Taillard’s benchmarks. The platform of our computational experiments is personal computer “Pentium Centrino: Dell Latitude D810” with a 1.73 GHz processor. These set of problems have been also used to determine the best values of the total flowtime and total tardiness. So, tables 1 and 2 summarize the best values of makespan and total flowtime criteria for the problem size 20 jobs – 20 machines and the CPU time. In the other hand, in order to obtain the best value of total tardiness criterion, we are, first, referred to the Daniels and Chambers [6] technique to generate due date of jobs. The due date ($d_j$) of jobs is randomly generated within the following interval:

$$d_j \in \left[ \text{ABP}\left(1-T-\frac{R}{2}\right), \text{ABP}\left(1-T+\frac{R}{2}\right) \right],$$

$$\text{ABP}=(n+m-1)\bar{P} \text{ and } \bar{P}=\frac{\sum_{j=1}^{n} \sum_{i=1}^{m} P_{ij}}{n \times m},$$

where:

- **ABP**: the Average Busy Period that serves as an approximation of the achievement time of the job in the sequence.
- **T**: delay factor or average percentage of overdue jobs, $T \in \{0.4; 0.6; 0.8\}$,
- **R**: factor controlling the extent of due dates, $R \in \{0.2; 0.6; 1\}$,
- **$\bar{P}$**: Mean processing time,
- **n**: number of jobs,
- **m**: number of machines.
TABLE 1: Best founded values of 20PF/20/C\(\text{max}\)

<table>
<thead>
<tr>
<th>Problems</th>
<th>Best founded value</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20_20_1</td>
<td>2297</td>
<td>16.794</td>
</tr>
<tr>
<td>20_20_2</td>
<td>2099</td>
<td>19.027</td>
</tr>
<tr>
<td>20_20_3</td>
<td>2326</td>
<td>19.805</td>
</tr>
<tr>
<td>20_20_4</td>
<td>2223</td>
<td>33.380</td>
</tr>
<tr>
<td>20_20_5</td>
<td>2291</td>
<td>21.391</td>
</tr>
<tr>
<td>20_20_6</td>
<td>2226</td>
<td>31.716</td>
</tr>
<tr>
<td>20_20_7</td>
<td>2273</td>
<td>39.386</td>
</tr>
<tr>
<td>20_20_8</td>
<td>2200</td>
<td>17.355</td>
</tr>
<tr>
<td>20_20_9</td>
<td>2237</td>
<td>22.001</td>
</tr>
<tr>
<td>20_20_10</td>
<td>2178</td>
<td>31.436</td>
</tr>
</tbody>
</table>

Average value of CPU time: 25.229

TABLE 2: Best founded values of 20PF/20/\(\Sigma C_i\)

<table>
<thead>
<tr>
<th>Problems</th>
<th>Best founded value</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20_20_1</td>
<td>33816</td>
<td>41.157</td>
</tr>
<tr>
<td>20_20_2</td>
<td>31674</td>
<td>34.032</td>
</tr>
<tr>
<td>20_20_3</td>
<td>33920</td>
<td>31.281</td>
</tr>
<tr>
<td>20_20_4</td>
<td>31722</td>
<td>20.766</td>
</tr>
<tr>
<td>20_20_5</td>
<td>34557</td>
<td>48.234</td>
</tr>
<tr>
<td>20_20_6</td>
<td>32753</td>
<td>34.250</td>
</tr>
<tr>
<td>20_20_7</td>
<td>32922</td>
<td>35.516</td>
</tr>
<tr>
<td>20_20_8</td>
<td>32444</td>
<td>21.641</td>
</tr>
<tr>
<td>20_20_9</td>
<td>33623</td>
<td>42.046</td>
</tr>
<tr>
<td>20_20_10</td>
<td>32262</td>
<td>40.204</td>
</tr>
</tbody>
</table>

The average value of the CPU time: 34.912

Note that the due date of each job was computed with\(T\) equal to 0.4 and\(R\) equal to 0.6. Table 3 represents the best values of the tardiness criterion. In this context, the obtained values will help the manager to obtain a sequence which better reflects his/her preferences. In the second step, we propose a new approach that takes into account the manager’s preferences by utilizing the concept of satisfaction function. We believe that this will give more flexibility to the manager to express explicitly his preferences.

TABLE 3: Best founded value of 20PF/20/\(\Sigma T_i\)

<table>
<thead>
<tr>
<th>Problems</th>
<th>Best founded values</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20_20_1</td>
<td>11019</td>
<td>37.782</td>
</tr>
<tr>
<td>20_20_2</td>
<td>10435</td>
<td>44.391</td>
</tr>
<tr>
<td>20_20_3</td>
<td>12095</td>
<td>15.819</td>
</tr>
<tr>
<td>20_20_4</td>
<td>9509</td>
<td>48.343</td>
</tr>
<tr>
<td>20_20_5</td>
<td>13526</td>
<td>55.297</td>
</tr>
<tr>
<td>20_20_6</td>
<td>10402</td>
<td>43.253</td>
</tr>
<tr>
<td>20_20_7</td>
<td>12509</td>
<td>39.703</td>
</tr>
<tr>
<td>20_20_8</td>
<td>9334</td>
<td>38.062</td>
</tr>
<tr>
<td>20_20_9</td>
<td>10138</td>
<td>39.012</td>
</tr>
<tr>
<td>20_20_10</td>
<td>10478</td>
<td>40.204</td>
</tr>
</tbody>
</table>

The average of the CPU time: 40.1866
3.2. Step 2: Determining the best compromise solution
A new metaheuristic based on tabu search algorithm has been developed to generate the sequence of the best compromise. The manager's preferences are expressed using the satisfaction functions.

a) Basis Concepts
The basic concepts of the proposed algorithm are presented as follows:

- **Initial sequence**: it can be selected from the set of three sequences obtained by optimizing each criterion as it has been presented in the previous step. The selected sequence will be stored in memory.
- **Neighbourhood structure**: the retained neighbourhood consists on a permutation of two jobs selected randomly.
- **Selection of the best neighbour**: the best neighbour is the sequence which offers to the manager the highest satisfaction level. To do so, we evaluate the neighbourhood and we choose the best one non-tabu or tabu and satisfy the aspiration criteria.
- **Tabu list**: the tabu list is managed in a circular manner. In this list, 7 prohibited moves can appear.
- **Aspiration criteria**: we used the simplest form of aspiration criterion which is stated as follows: a tabu move is accepted if it produces a sequence better than the best obtained so far.
- **Stopping criteria**: the algorithm is stopped when it reaches a number of iterations without improvement of the evaluation function. This number is fixed in advance.

In addition to these concepts, the set of Pareto optimal solutions is used. It contains all non-dominated sequences. In fact, the principle of dominance concerns only the value of the optimized criteria and not the value of the objective function.

b) The algorithm structure
The proposed algorithm is as follows:
**Initialisation**
- Initial sequence \( s_0 \)
- \( s_n = s_0, f = f(s_0) \)
- \( LT = \emptyset, PE = \{s_0\} \)

**Iterative Processes**
1. Generate the neighborhood of the current sequence \( s_n \)
   - Select the best neighbour of \( s_n \), \( s^* \in SV(s_n) \)
2. A) if \( s^* \) is a non-dominant sequence
   - if the move \( s_n \) to \( s^* \) is non tabu
     - Update the tabu list \( LT \)
     - Update the set of Pareto optimal sequence \( PE \)
     - \( s_{n+1} \leftarrow s^* \)
     - if \( f < f(s^*) \):
       - Initialize the iteration counter
       - \( f \leftarrow f(s) \)
       - Go to 3
   - Otherwise, go to C)
B) if \( s^* \) is a dominated solution
   - if the move \( s_n \) to \( s^* \) is non tabou
     - Update the tabu list \( LT \)
     - \( s_{n+1} \leftarrow s^* \)
     - Go to C)
   - Otherwise:
     - Choose \( s' \) such that \( f(s') = \max \{ f(s_i) \mid s_i \in SV(s_n) \) and \( s_i \) non-tabu\}
     - \( s_{n+1} \leftarrow s' \)
3. If the iterations counter is less than the number of iterations without improvement, go to 1.
   - Otherwise, End.

Where:
- \( s_0 \): initial sequence;
- \( s_n \): current sequence;
- \( s_{n+1} \): new current sequence;
- \( LT \): tabu list;
- \( PE \): set of the Pareto optimal sequences;
- \( SV(s_n) \): neighbourhood of the current sequence.

This algorithm is based on two stages which are: a) generating an initial sequence, and b) generating a neighbourhood sequence. The initial sequence is chosen from among the three calculated sequences in step 1. In this stage, the tabu list is empty and the set of the Pareto optimal sequence contains only the initial sequence which later becomes the current sequence of the second phase. The neighbourhood sequence is generated in order to choose the best that has the highest satisfaction level. Dominance tests will be established followed by an update of the set of Pareto optimal sequences and hence the current sequence. For each iteration, we proceed in the same way until we reach the algorithm’s stopping condition.

c) The algorithm description
The initial sequence \( (s_0) \) is generated from step 1. At this stage, we obtain three sequences which correspond to the three criteria optimized alternately, so that the choice of the initial sequence is made in an arbitrary manner. This sequence is stored in memory as the best sequence found \( (s_n) \) and will be assigned to the set of Pareto optimal sequences. Thus, it is classified as a current.
sequence. Starting from this sequence, the iterative process of the algorithm starts by obtaining a new sequence at each iteration. Indeed, a search in the neighbourhood is made to find the best neighbour \((s^*)\). The quality of the sequence is evaluated through the objective function value, but without taking into consideration the value of the objective function of \((s_0)\). The best neighbour of the current sequence thus obtained will be tested for dominance over all the set of Pareto optimal sequences. This test is done by comparing the values obtained for each criterion. In such context, two cases appear:

- **The best neighbour of the current sequence is not dominated:**
  This means that there are no sequences to be considered, belonging to the set of Pareto optimal sequences, which dominates. Therefore, this sequence will be chosen as the new current one if it is not subject to a tabu status, or through the application of the aspiration criterion which revokes that tabu status. An update of the list of the Pareto optimal sequences will be done by eliminating any sequence dominated by \((s^*)\). At this stage, we conduct a comparison between the best objective function value of the old sequence \(f^*\) and the current sequence \(f(s^*)\). If the latter case is better than \(f^*\), its value will be stored in memory and the counter of iterations will be initialized again to make another iteration. Therefore, the iterative process stops after a certain number of iterations without improving the objective function value.

- **The best neighbour of the current sequence is dominated:**
  In such case, we will check the status of the sequence. If the movement from which the sequence was obtained is not tabu, the best neighbour is used to explore the neighbourhood in search of other sequences. It will serve as a starting point for the next iteration. Nevertheless, if this movement is tabu, the new current sequence of the next iteration is the neighbour who is not tabu and maximizes the objective function value. This process is repeated for each iteration until stopping the algorithm.

This metaheuristic was tested through a computational experience that we will present the results in the next section.

**4. COMPUTATIONAL RESULTS**

Several tests were conducted to check the performance of the proposed metaheuristic. We have used several problems, presented by Taillard [24] with different sizes such as: 20 jobs-5 machines; 20 jobs-10 machines; 20 jobs-20 machines; 50 jobs-5 machines; 50 jobs-10 machines and 100 jobs-5 machines. For each problem, we took 10 different instances. Table 4 shows the date file of one scheduling problem characterized by 20 jobs-5 machines

**4.1. Parameters file**

In this file, we inscribe the information about the best values obtained for each criterion, type of satisfaction function used, different thresholds, weights of criteria, tabu list length, neighbourhood list length and the number of iterations without improvement.
4.2. Initial sequence file

This file indicates which initial sequence will be used to start the iterative process of the algorithm. This sequence is chosen among a set of three sequences obtained through the optimization of only one criterion. We find also the values of the three optimized criteria and the value of the satisfaction level. Ten (10) test problems were generated. Table 5 and 6 summarize the obtained results. Based on the results we can conclude that the proposed approach is able to solve multi-criteria permutation flowshop problems in different sizes and integrate explicitly the manager's preferences. In fact, the obtained sequences respect the manager's preferences which may explain the high level of the achieved satisfaction. In this context, the average of satisfaction degree for the following problems (20y5), (20y10), (20y20), (50y5), (50y10) and (100y5) is equal to 1; 0.952; 0.9126; 1; 0.9508 and 1 respectively.

Based on the obtained results in Tables 5 and 6, some benefits of the proposed approach can be identified; it is fast and flexible. The speed of the approach is measured by the computation time required to solve different sizes of problems. The mean computation time is equal to 260.87 seconds for the 20 jobs -20 machines problems. This time may be acceptable. On its flexibility, the proposed approach can solve different problems with different satisfaction levels. Similarly, in its current version, this approach includes all necessary components to add other performance criteria.
Mohamed Anis Allouche

<table>
<thead>
<tr>
<th>Problems</th>
<th>Satisfaction level</th>
<th>Problems</th>
<th>Satisfaction level</th>
<th>Problems</th>
<th>Satisfaction level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job_20_5_1</td>
<td>1</td>
<td>Job_20_10_1</td>
<td>0.9985</td>
<td>Job_20_20_1</td>
<td>1</td>
</tr>
<tr>
<td>Job_20_5_2</td>
<td>1</td>
<td>Job_20_10_2</td>
<td>0.6168</td>
<td>Job_20_20_2</td>
<td>1</td>
</tr>
<tr>
<td>Job_20_5_3</td>
<td>1</td>
<td>Job_20_10_3</td>
<td>1</td>
<td>Job_20_20_3</td>
<td>0.96</td>
</tr>
<tr>
<td>Job_20_5_4</td>
<td>1</td>
<td>Job_20_10_4</td>
<td>1</td>
<td>Job_20_20_4</td>
<td>1</td>
</tr>
<tr>
<td>Job_20_5_5</td>
<td>1</td>
<td>Job_20_10_5</td>
<td>0.93</td>
<td>Job_20_20_5</td>
<td>0.69</td>
</tr>
<tr>
<td>Job_20_5_6</td>
<td>1</td>
<td>Job_20_10_6</td>
<td>1</td>
<td>Job_20_20_6</td>
<td>0.834</td>
</tr>
<tr>
<td>Job_20_5_7</td>
<td>1</td>
<td>Job_20_10_7</td>
<td>1</td>
<td>Job_20_20_7</td>
<td>0.82</td>
</tr>
<tr>
<td>Job_20_5_8</td>
<td>1</td>
<td>Job_20_10_8</td>
<td>0.9814</td>
<td>Job_20_20_8</td>
<td>0.96</td>
</tr>
<tr>
<td>Job_20_5_9</td>
<td>1</td>
<td>Job_20_10_9</td>
<td>1</td>
<td>Job_20_20_9</td>
<td>0.894</td>
</tr>
<tr>
<td>Job_20_5_10</td>
<td>1</td>
<td>Job_20_10_10</td>
<td>1</td>
<td>Job_20_20_10</td>
<td>0.968</td>
</tr>
</tbody>
</table>

Average 1 Average 0.952 Average 0.9126

**TABLE 5:** Obtained satisfaction level for problems (20-5), (20-10) and (20-20)

<table>
<thead>
<tr>
<th>Problems</th>
<th>Satisfaction level</th>
<th>Problems</th>
<th>Satisfaction level</th>
<th>Problems</th>
<th>Satisfaction level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job_50_5_1</td>
<td>1</td>
<td>Job_50_10_1</td>
<td>0.9955</td>
<td>Job_100_5_1</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_2</td>
<td>1</td>
<td>Job_50_10_2</td>
<td>1</td>
<td>Job_100_5_2</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_3</td>
<td>1</td>
<td>Job_50_10_3</td>
<td>1</td>
<td>Job_100_5_3</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_4</td>
<td>1</td>
<td>Job_50_10_4</td>
<td>0.8919</td>
<td>Job_100_5_4</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_5</td>
<td>1</td>
<td>Job_50_10_5</td>
<td>1</td>
<td>Job_100_5_5</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_6</td>
<td>1</td>
<td>Job_50_10_6</td>
<td>1</td>
<td>Job_100_5_6</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_7</td>
<td>1</td>
<td>Job_50_10_7</td>
<td>1</td>
<td>Job_100_5_7</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_8</td>
<td>1</td>
<td>Job_50_10_8</td>
<td>0.9888</td>
<td>Job_100_5_8</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_9</td>
<td>1</td>
<td>Job_50_10_9</td>
<td>0.848</td>
<td>Job_100_5_9</td>
<td>1</td>
</tr>
<tr>
<td>Job_50_5_10</td>
<td>1</td>
<td>Job_50_10_10</td>
<td>0.7837</td>
<td>Job_100_5_10</td>
<td>1</td>
</tr>
</tbody>
</table>

Average 1 Average 0.9508 Average 0.9126

**TABLE 6:** Obtained satisfaction level for problems (50-5), (50-10) and (100-5)

5. CONCLUSION

In this paper, we have presented a new metaheuristics to solve multi-criteria permutation flowshop problems taking into account the manager’s preferences. This metaheuristic is an improvement of the model proposed by Allouche et al. [2] which can be now useful for large scale multi-criteria scheduling problems. This metaheuristic is based on the tabu search algorithm. The concept of satisfaction functions was utilized to integrate explicitly the manager’s preferences. The obtained sequence can be qualified as the best compromise. We have considered three scheduling criteria. However, this metaheuristic can be extended for additional criteria that the manager may want to consider. We believe that our approach is easy to use and requires a small number of parameters to be provided by the manager. This approach can be qualified as a good tool for multi-criteria scheduling problems.

6. REFERENCES


22. OR-Library: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/flowshopinfo.html


