Research Article

Improved Tabu Search for Parallel Identical Machines with Availability Zones \((T_z, S_z)\)

\(^1\text{Rachid Zitouni and } ^2\text{Omar Selt}

\(^1\text{Laboratory of Fundamental and Numerical Mathematics, University of Setif 1, Algeria}

\(^2\text{Department of Mathematics, University of M'sila, Algeria}

Abstract: In this study we propose a new polynomial metaheuristic elaboration (tabu search) for solving scheduling problems. This method allows us to solve the scheduling problem of \(n\) Tasks on \(m\) identical parallel machines with availability zones \((T_z, S_z)\). Since this problem is NP-complete in the strong sense and finding an optimal solution appears unlikely, we suggested an heuristic based on choosing the most available machine. To improve the performance of this heuristic, we used diversification strategies with the aim of exploring unvisited regions of the solution space and two well-known neighborhoods (neighborhood by swapping and neighborhood by insertion). It must be noted that tasks movement can be within one machine or between different machines. Note that all data in this problem are integer and deterministic. The performance criterion to optimize in this problem which we denote as \(P_{\text{max}} \rightarrow \text{min} - C_{\text{max}}\) is the weighted sum of the end dates of tasks.

Keywords: Metaheuristic, parallel identical machines, scheduling, tabu search, unavailability periods

INTRODUCTION

A scheduling problem consists in organizing tasks realization time with consideration of time constraints (time limits, tasks series character) and constraints related to using and availability of required resources.

The scheduling constitutes a solution to the considered problem, describes the tasks execution and resources location during time and aims to satisfy one or many objectives.

A scheduling problem under machines availability constraints has been studied by many authors. For example, \(P_{\text{max}} \rightarrow \text{min} - C_{\text{max}}\) has been studied by Lee (1996, 1997, 1999) and Schmidt (2000). The tabu search is a metaheuristic originally developed by Glover and Hanafi (2002) and independently by Hansen (1986).

This method combines a local search procedure with a certain number of rules and mechanism which allows surmounting the obstacle of local optima without cycling. Toward furthermore, it proved high efficiently in resolution of the problems NP-complete and approximate more the optimal solution.

The scheduling problem of a single machine with minimization of the weighted sum of the end dates of tasks. Without unavailability constraint is optimally resolved by using the WSPT (Weighted Shortest Processing Time) rules. The case of several machines is studied by many authors like Belouadah \textit{et al.} (1992), Sadfi (2002) and Haouari and Ladhari (2003).

Kacem \textit{et al.} (2005) have studied the problem \(1 \rightarrow \text{min} - \sum C_j\) and have compared two exact methods: one is the Branch and Bound, the other is the integer programming. They have concluded that Branch and Bound method have better performance and it allowed resolving instances of more than 1000 tasks.

Corresponding Author: Omar Selt, Department of Mathematics, University of M'sila, Algeria

This work is licensed under a Creative Commons Attribution 4.0 International License (URL: http://creativecommons.org/licenses/by/4.0/).
Adamu and Adewumi (2012, 2013) have studied the problem \( P_m \parallel \sum_{j=1}^{n} w_j (U_j + V_j) \), they proposed some metaheuristics for scheduling on parallel identical machines to minimize weighted number of early and tardy jobs.

Yun-Chia et al. (2013), they carried out a comparative study of different (a genetic algorithm, particle swarm optimization and simulated annealing with their hybrids) metaheuristics for identical machines.

In this study, the results of Adamu and Adewumi (2012, 2013) research work are exploited to develop a different new metaheuristic to solve the scheduling problem under different constraints.

**Problem statement:** This problem consists in scheduling \( n \) tasks for \( m \) parallel identical machines \{\( M_1, M_2, \ldots, M_m \)\} where \( n > m \geq 2 \), with unavailability periods.

We assume that the tasks \{\( j_1, j_2, \ldots, j_n \)\} are all available at \( t = 0 \) and their operation times are independent from the choice of machines performing these tasks. In the generic case of the problem, each one of the \( m \) machines can show some unavailability periods during scheduling horizon and each task must be executed onetime.

This problem noted by \( P_m \parallel N - C / \sum_{j=1}^{n} w_j C_j \) consists in assigning \( n \) tasks to \( m \) machines over availability intervals in a manner to enforce the weighted sum of the end dates of tasks, referred to as \( \sum_{j=1}^{n} w_j C_j \) to be minimal.

It must be noted that there is \( (n!)^m \) possibility to assign \( n \) tasks to \( m \) machines.

**METHODOLOGY**

**Neighborhood structure:** Neighborhood determination constitutes the most important stage in metaheuristic methods elaboration. In the following part, we use two Neighborhoods, (neighborhood by swapping) and (neighborhood by insertion).

It must be noted that tasks movement can be within one machine or between different machines.

**Neighborhood by swapping:**

**Definition:** Consider a sequence \( \sigma \) composed of \( n \) tasks. A neighborhood \( \sigma' \) is obtained by permuting two tasks \( j \) and \( j' \) of respectively \( k \) and \( k' \) positions \( \sigma \) with \( k' = k + 1, k + 2, \ldots, n \).

The set:

\[
N_1(\sigma) = \{ \sigma, \sigma \text{ is obtained by permutation of two tasks} \}
\]

is called neighborhood of \( \sigma \) this set is consequently obtained by permutation of all tasks of \( \sigma \) two by two.

**Proposition 1:** Consider a sequence \( \sigma \), the set's cardinal of \( N_1(\sigma) \) is \( \frac{n(n-1)}{2} \).

**Proof:** The permutation of all tasks. Two by two consists in permuting each task of the sequence with all remained tasks, without identical ones. The number of possible permutations in a sequence \( \sigma \) composed of \( n \) tasks is: \( (n - 1) + (n - 2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} \).

**Neighborhood by insertion:**

**Definition:** Consider a sequence \( \sigma \) composed of \( n \) tasks. A neighborhood \( \sigma' \) is obtained by inserting one task \( j \) of a position \( k \) in a new position \( k' \) in the sequence \( \sigma \).

The set:

\[
N_2(\sigma) = \{ \sigma', \sigma' \text{ is obtained by inserting a task of position } k \text{ in } k' \}
\]

is a neighborhood of \( \sigma \), this set is consequently obtained by realizing all possible insertions of all tasks of \( \sigma \).

**Proposition 2:** Consider a sequence \( \sigma \), the set's cardinal of \( N_2(\sigma) \) is \( (n-1)^2 \).

**Proof:** Inserting a task \( j \) of position \( k \) in another position \( k' \) in the sequence \( \sigma \) allows getting \( n-1 \) possible insertions. Hence, for \( n \) tasks, there is \( n(n-1) \) insertions to be done. To avoid getting identical sequences, adjacent tasks insertions are counted once. Consequently \( n-1 \) insertions will be deleted. Finally, the number of obtained insertions is: \( n(n-1) - (n-1) = (n-1)^2 \).

**Tabu list structure:** The tabu method is based on the principle that consists in maintaining in memory the last visited solutions and in forbidding the return to them for a certain number of iterations. The aim is to provide sufficient time to the algorithm so it can leave the local optimum. In other words, the tabu method conserves in each stage a list \( L \) of solutions (Tabu's) which it is forbidden to pass-by temporarily. The necessary space for saving a set of solutions tabs in the memory is indispensable.

The list, that we propose, contains the found solutions sequences. After many tests, a dynamic size list, which varies according to the search amelioration state, is conceived. The initial size of this list is considered to be \( (3\sqrt{n})/2 \) where \( n \) is the tasks number. After that, during the search, when 5 successive iterations pass without amelioration of solution, the list is reduced to a number inferior or equal to \( 3n \). On the other hand, when 5 successive iterations pass and the solution is ameliorated, the list is increased to a number superior or equal to \( 2n \). The Tabu list is consequently
dynamic and its size varies within the interval $(\sqrt{n}, 2\sqrt{n})$. The decrease or the increase of list size must always be done at the end of the list.

**Heuristic for the problem (P):** An initial solution is always necessary. For this reason, we suggest in this part the following heuristic: assign the (best) task $h$ where $\left(\frac{p_h}{w_h} \min_{j \in J} \frac{p_j}{w_j} \right)$ to the best machine (the most available) based on two principles justified by the two following propositions.

**Proposition 3:** In an optimal scheduling, it is necessary to schedule the tasks, in each availability period of the machine according to the order SWPT.

**Proof:** It results directly by adjacent task exchange like used by Smith (1956) for the corresponding periods.

**Proposition 4:** It is not useful to let the machine (idle) if a task can be assigned to this machine.

### NOTATIONS

We denote by:

- $J = \{1, 2, ..., n\}$ : The set of tasks
- $p_h$ : Execution time of the task $h$
- $p_h^{(i)}$ : Execution time of the task $h$ assigned to the machine $i \in I$
- $I = \{1, 2, ..., m\}$ : The set of machine
- $I_{\text{NA}}$ : The set of non-assigned machines
- $\alpha$ : Number of availability zones
- $Z = \{1, 2, ..., \alpha\}$ : Availability zones
- $S_z^{(i)} (z \in Z)$ : The beginning of the unavailability time of the machine $i \in I$
- $T_z$ : Final time
- $T_z^{(i)} (z \in Z)$ : The end of the unavailability time of the machine $i \in I$
- $\sigma_z^{(i)} (z \in Z)$ : The set of partial sequences assigned to the machine $i \in I$
- $\sigma_z = \sigma_z^{(1)} \cup \sigma_z^{(2)} \cup ... \cup \sigma_z^{(m)}$
- $J_z^{(i)} (z \in Z)$ : The task assigned to the machine $i$ with $T_z^{(i)} \leq t^{(i)} \leq S_z^{(i)}$
- $C_z^{(i)} (z \in Z)$ : Execution time of the task $j \in J_z^{(i)}$.

We assume that for each task $h \in J$, there is at least one machine $i \in I$ such that $p_h \leq S_z^{(i)} - T_z^{(i)}$.

### FORMULATIONS MATHEMATICS

Consider $I = \{1, 2, ..., m\}$, $J = \{1, 2, ..., n\}$, $\sigma = (\sigma (1), \sigma (2), ..., \sigma (n))$ a sequence of $n$ tasks and $P$ the set of all possible sequences permutations.

This problem is formulated as an integer linear programming model:

$$f = \min_{\sigma \in P} \sum_{j=1}^{\infty} w(j) \sigma(j) C(j)$$

Such that

$$\sum_{j \in \sigma(i)} p(j) \leq S_z^{(i)},$$

$$S_z^{(i)} \in \mathbb{N}; z = 1, 2, ..., \alpha; \sigma(j) = \sigma^{(i)} \cup \sigma^{(j)} \cup ... \cup \sigma^{(n)}$$

$$C(i) = \sum_{j \in \sigma(i)} p(j) + T_z^{(i)}, \quad T_z^{(i)} \in \mathbb{N};$$

$$i = 1, 2, ..., m; \quad j = 1, 2, ..., n$$

### Algorithm:

**Initialization:**

- $J = \{1, 2, ..., n\}$; $I = \{1, 2, ..., m\}$; $Z = \{1, 2, ..., \alpha\}$; $I_{\text{NA}} = I$;
- $S_z^{(i)} = T_z$ (given), $\sigma = \emptyset$, $f_0 = 0$, $z = 1$, $C^{(0)} = 0$ and $T_z^{(i)} = 0$.

Sort task $h \in J$ in increasing order according to the criterion $p_h / w_h$ in a list $L_1$

Sort task $h \in J$ in increasing order according to the criterion $p_h$ in a list $L_2$

While $L_i \neq \emptyset$ and $z \leq \alpha$ do

Begin

Set $p_h = p_h / w_h$ from the top list of $L_1$.

Set $p_h = p_h$ from the top list of $L_2$.

Determine the machine $k \in I_{\text{NA}}$ and the task $h \in J$ such that $S_z^{(i)} - C_j^{(i)} = \max_{i \in I_{\text{NA}}} \left\{ S_z^{(i)} - C_j^{(i)} \right\} = \min(p_h, p_{h_2})$

If $\{k\} = \emptyset$ then

Determine the machine $k \in I$ and the task $h \in J$ such that $S_z^{(i)} - C_j^{(i)} = \max_{i \in I} \left\{ S_z^{(i)} - C_j^{(i)} \right\} = \min(p_h, p_{h_2})$

End if

If $\{k\} = \emptyset$ then

425
Begin
Assigned the task \( h \) to the machine \( k \)
Delete the task \( h \) from the two lists \( L_1 \) and \( L_2 \)
Compute \( C_z^{(i)} = \sum_{j \in j_i} p_j + T_z^{(i)} \);
Determine \( \sigma_z^{(i)} = \sigma_z^{(i)} \cup [h] \) and \( f_\sigma = f_\sigma + w_k C_z^{(i)} \);
Set \( I_{\sigma} = I_{\sigma} \setminus \{k\} \)
End
Else
Begin
Set \( z = z + 1 \); \( I_{\sigma} = I \);
End
End if
End

Computational analysis:
Data generation: The heuristic were tested on problems generated with 500 tasks similar to that used in previous studies (Adamu and Abass, 2010; Baptiste et al., 2000; Ho and Chang, 1995; M’Hallah and Bulfin, 2005); for each task \( j \) an integer processing time \( p_j \) was randomly generated in the interval \((1, 99)\) with a weight randomly \( w \) chosen in interval \((1, 10)\).

The search time to define a neighborhood and to determine minimal cost is chosen equal to 90s.

The number of machines fixed (3 machines) with (3 availability zones for each machine).

Diversification strategies: The final time to execute this problem is chosen as \( T_f = 1200 \) sec. It is divided according to diversification strategy to three times \( T_1 \), \( T_2 \) and \( T_3 \). After many experiments, these periods are chosen as follows:

\[ T_1 = 700 \text{ sec Initial starting time: uses long term memory to store the frequency of the moves executed through out the search}. \]

\[ T_2 = 300 \text{ sec First restarting time makes use of influential moves}. \]

\[ T_3 = 200 \text{ sec Second restarting time also makes use of influential moves}. \]

The Table 1 presents:

- The initial mean values of objective function corresponding to initial sequence
- The initial mean values of objective function obtained by using on one hand, the neighborhood by swapping and on the other hand, the neighborhood by insertion
- The average times corresponding to the two neighborhoods
- The percentage of cost improvement
- The best costs

RESULTS

The results listed in Table 1 shows clearly that the tabu method based on neighborhood by insertion presents best costs compared with tabu method based on neighborhood by swapping. This is due to the fact that the first neighborhood ensures a faster tasks movement besides that the search space is richer with optimals partials sequences in each availability zones \((T_z, S_z)\).

This can also be explained by the nature of used neighborhoods, besides the left shifting of other tasks in the swapping neighborhood. The results show that execution time obtained by the first neighborhood is acceptable.

On the other hand, the heuristic amelioration rate between the two neighborhoods is remarkable (Fig. 1a and b). It is also noted that the cost amelioration rate of the proposed tabu search heuristic is situated between 0.4 and 14%.

<table>
<thead>
<tr>
<th>n</th>
<th>Initial cost by Tabu search by swapping</th>
<th>Tabu search by insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(AC of 5 instances)</td>
<td>AC</td>
</tr>
<tr>
<td>50</td>
<td>46290</td>
<td>41142</td>
</tr>
<tr>
<td>500</td>
<td>27743207</td>
<td>2771998</td>
</tr>
<tr>
<td>500</td>
<td>278074</td>
<td>2693834</td>
</tr>
<tr>
<td>2576238</td>
<td>2638554</td>
<td>470</td>
</tr>
<tr>
<td>500</td>
<td>4222734</td>
<td>4273518</td>
</tr>
<tr>
<td>500</td>
<td>4422156</td>
<td>4386046</td>
</tr>
<tr>
<td>4649564</td>
<td>4497934</td>
<td>486</td>
</tr>
</tbody>
</table>

AC : Average costs, AT: Average time, PIIC: Percentage of improvement of the initial costs
CONCLUSION

In this study, a metaheuristic polynomial approach (Tabu search) as solution for tasks scheduling problem with parallel identical machines and availability zones \((T_z, S_z)\) is presented.

The developed approach uses a diversification technique based on search restarting from the point of the solution that was chosen among the earlier best unmaintained found solutions, by considering that the tabu list is dynamic and its size varies according to amelioration state of the solution.

According to the curried out tests, it can be concluded that the proposed approach ensure better results (heuristic amelioration cost up to 14\%). It must be noted that the neighborhood by insertion presents the best costs with an acceptable execution time.
REFERENCES


End note:
1: A machine is supposed to be the most available if it has an availability period the most close to $t = 0$ and it is able to realize the required task.