COMPETITIVE AGENTS IMPLEMENTING PARALLEL TABU SEARCHES FOR JOB SHOP SCHEDULING PROBLEM WITH TIME LAGS

Madiha Harrabi1, Olfa Belkahla Driss1,2, Khaled Ghedira1,3
1Complex Outstanding Systems Modeling Optimization and Supervision (COSMOS Laboratory)
Ecole Nationale des Sciences de l’Informatique, Université de Manouba, Tunis
2Ecole Supérieure de Commerce de Tunis, Université de Manouba, Tunis
3Institut Supérieur de Gestion, Université de Tunis, Tunis
madiha.harrabi@gmail.com,
{olfa.belkahla, khaled.ghedira}@isg.rnu.tn

ABSTRACT
This paper deals with the Job Shop scheduling problem with constraints of minimum and maximum Time Lags (JSTL). This problem is an extension of the job shop scheduling problem, with additional constraints of minimum and maximum time lags existing between successive operations of the same job. In this work, we investigate Parallel Tabu Searches implemented by competitive agents for the Job Shop problem with Time Lags. The resolution model consists of an Interface Agent and a set of n competitive Job Agents, each one containing the tabu search core, generates an initial solution, executes the tabu search process and uses the NEH heuristic process for the diversification step. Good performances of the proposed model are shown through different comparisons on benchmarks based on instances of the literature.

KEY WORDS
Scheduling, Job Shop, Time Lags, Tabu Search, Multi-Agent System, Optimization

1. Introduction
Scheduling problems has been the subject of intensive research since the early 1950s. The Job Shop scheduling problem is defined by a set of n jobs which have to be processed on m machines. The dimension of the problem is known as n*m. The Job Shop problem with Time Lags is a special case of the classical job shop problem. This problem can be defined as a job shop problem with minimum and maximum delays existing between starting times of successive operations of the same job. Time Lags constraints could be used in several fields of industrial applications of job shop. In this paper, we propose a new model called Competitive Agents implementing Parallel Tabu Searches for solving the Job Shop problem with Time Lags (CAPTS-JSTL). The objective is the minimization of the makespan. This paper is organized as follows. Section 2 presents the Job Shop scheduling problem with Time Lags. Section 3 describes our proposed model CAPTS-JSTL. In section 4, we discuss our experimental results and in section 5, we give our conclusions and remarks with some future directions.

2. Job Shop problem with Time Lags

2.1 Literature review
Time lags constraints existing between the start and completion times of successive operations in the same job were introduced in numerous scheduling problems. [23] was investigated a polynomial algorithm in order to minimize the makespan for the flow shop scheduling problem with two machines. [27] studied the complexity of single-machine problems with time lags delays between jobs. Many works were concerned by solving the Resource-Constrained Project Scheduling Problem (RCPSP) with time lags. [15] proposed a branch and bound procedure for the resource-constrained project scheduling problem with time lags constraints between the different activities. [16] introduced an approach based on local search metaheuristic for solving the single-machine problem with additional time lags constraints. [8] proposed different heuristics methods for job shop and flow shop scheduling problems with time lags. [10] studied the permutation flow shop scheduling problems with additional constraints of minimum and maximum time lags between couples of successive operations of jobs.

[6] introduced a memetic algorithm. [17] investigated a Climbing Discrepancy Search method. [2] investigated a new heuristic using the branch and bound method including the resource constraint propagation procedure. [12] proposed a scatter search procedure combining the path relinking and the tabu search metaheuristic of [26]. [1] proposed a disjunctive graph model for the Job Shop problem with Time Lags and transport. [19], [20] and [14] investigated the job shop problem with generic time lags, i.e. existing between any pairs of operations of different jobs. Recently the multi agent systems are widely used for the resolution of job shop problems. [25] proposed a hybrid metaheuristic approach based on clustered holonic multiagent model for scheduling of machines and transport robots in job shop environment. [22] proposed a chemical reaction optimization in a multi agent model to solve the flexible Job Shop problem. The job shop problem with additional time lags constraints is NP-hard [19] so that the centralized resolution cannot be efficient and we focus for a distributed resolution of this problem. [13] proposed a tabu search metaheuristic in a multi agent model, good performances of distributed approach are shown in terms of makespan and CPU time, but there is a synchronization of message that makes agents work depending on each other. For this reason, we search for using competitive agents independently working to reach an objective in a minimum time.

2.2 Problem formulation

The problem linear formulation of Job Shop with Time Lags is an extension of the Manne’s formalization [21] of classical job shop with additional minimum and maximum time lags constraints. The Job Shop Scheduling problem with Time Lags is formulated as follows.

- \( M \) set of machines
- \( O \) set of operations to schedule
- \( Z \) completion time of the last operation
- \( E_K \) set of operations processed on machine \( k \)
- \( t_i \) starting time of operation \( i \)
- \( p_i \) processing time of operation \( i \)
- \( s_i \) next operation of operation \( i \) depending on job
- \( TL_{ij}^{\text{min}} \) minimal time-lags between \( i \) and \( s_i \)
- \( TL_{ij}^{\text{max}} \) maximal time-lags between \( i \) and \( s_i \)
- \( a_{ij} = \begin{cases} 1 & \text{if operation } j \text{ is processed before } i \\ 0 & \text{otherwise} \end{cases} \)
- \( H \) a positive large number

Minimize \( Z \).

Subject to
\[
(H + p_i) a_{ij} + (t_i - t_j) \geq p_j \forall (i, j) \in E_k, \forall k \in M \quad (1)
\]
\[
(H + p_j) (1 - a_{ij}) + (t_i - t_j) \geq p_j \forall (i, j) \in E_k, \forall k \in M \quad (2)
\]

\[
t_i + p_i + TL_{ij}^{\text{min}} \leq t_i \forall i \in O \quad (3)
\]

\[
t_i \leq t_i + p_i + TL_{ij}^{\text{max}} \forall i \in O \quad (4)
\]

\[
Z \geq t_i + p_i \forall i \in O \quad (5)
\]

\[
t_i \geq 0 \forall i \in O \quad (6)
\]

\[
a_{ij} \in \{0; 1\} \forall (i, j) \in E_k, \forall k \in M \quad (7)
\]

3. Competitive agents implementing parallel Tabu Searches for Job Shop problem with Time Lags

3.1 Global View

A multi agent system is a one of the most interesting programming paradigms, used to facilitating the development of distributed and decentralized architecture. It is a computerized system includes different intelligent agents with the possibility of the cooperation, communication, and coordination with each other to attend common objectives. Our proposed model called Competitive Agents implementing Parallel Tabu Searches for the Job Shop problem with Time Lags (CAPTS-JSTL) consists of two types of agents: an Interface Agent and \( n \) competitive Job Agents where \( n \) is the number of jobs.

3.2 Agent description

- Interface Agent

This type of agent has the intermediate role between the user and Job Agents. It aims to launch the program and to create different Job Agents. It subsequently provides the necessary information for each Job Agent. Finally, it receives optimal solutions found by different Job Agents, evaluates these solutions and chooses the optimal solution with the minimum makespan. It returns the optimal solution to the user when the problem has been resolved.

- Job Agent

The Job Agents contain the tabu search core. Each Job Agent builds an initial solution with a fixed job that cannot be moved during the resolution process to ensure the diversification of solutions built by several Job Agents. Then, it calculates the inactivity intervals for each resource; chooses the moving job and starts its own tabu search process. The knowledge of the Job Agent is composed mainly of the tabu search parameters. Static knowledge of the Job Agent consists of: processing time of jobs and time lags between start and completion times of successive operations in the same job, initial solution \( S_0 \) from which begins the optimization process, fixed job, stopping criterion, size of the tabu list and diversification procedure. Dynamic knowledge of Job Agent consists of: makespan of the current solution, inactivity intervals of resources, job to move, neighbor solutions, makespans of the
neighbor solutions, best solution found and its makespan, elements of the tabu list, number of iterations, number of iterations after the last improvement. The Job Agent is satisfied when the fixed stopping number of iterations is reached; then it sends the best found solution to the Interface Agent.

3.3 Global dynamic

The Interface Agent creates different Job Agents according to the job number and provides for them different problem informations. Each Job Agent generates an initial solution and fixed a job that cannot be removed among the resolution process, calculates inactivity interval for each resource, chooses the moving operation which have the last execution operation of the current solution, creates neighbor solutions and evaluates these solutions. It selects the best non-tabu neighbor among the neighborhoods of the current solution for the next iteration. If there is any improvement in the current solution after such number of iterations, the diversification step was launched, and a new solution was generated using the NEH heuristic and the resolution process, calculates inactivity intervals for each resource, chooses the moving operation which have the last execution operation of the current solution, creates neighbor solutions and evaluates these solutions. It selects the best non-tabu neighbor among the neighborhoods of the current solution for the next iteration. If there is any improvement in the current solution after such number of iterations, the diversification step was launched, and a new solution was generated using the NEH heuristic and the resolution process was started by the Job Agent. After a maximum number of iterations, each Job Agent sends the best solution to the Interface Agent, then the latter chooses the optimal solution, returns its found makespan and detects that the problem has been solved.

3.4 Optimization process

3.4.1 Tabu Search

The Tabu Search (TS) metaheuristic was proposed by [11]. This algorithm has been successfully used for solving different combinatorial optimization problems such as scheduling problems. It is a metaheuristic based on the local search. The idea of local search is to visit search space composed of all feasible solutions in order to reach the optimal solution.

At each iteration the most appropriate neighborhood which may not be an improving solution is selected. Then the search is repeated starting from the best found solution as current solution. The particularity parameter of TS is the tabu list which explicitly employs the history of the search, and storing the selected solutions for a certain number of iterations known as Tabu list size, the parameter of tabu list was used to prevent the resolution process from local minima and to implement a good explorative strategy.

Several studies in the literature have shown the efficiency of this method for solving many difficult problems, this mainly due to the tabu search parameters: initial solution, neighborhood function, tabu list and diversification process.

3.4.2 Initial Solution

The algorithm of tabu search starts with an initial solution to begin the search of better configurations in the search space. In the implementation of this work, we use an initial solution generated randomly by the Job Agent. It inserts the fixed job at the earliest time, then, it randomly inserts the other jobs. The generated initial solution will be used to start the neighborhood step and visit other search space.

3.4.3 Neighborhood function

In the literature, we can find different structure of neighborhood function. Among the most used ones we can cite:

- **Swapping move:** given a permutation \( \pi \) of \( n \) jobs; \( j \) and \( k \) \( \in \{1, \ldots, n\} \), two consecutive positions in \( \pi \), the permutation that results of such a move is obtained by swapping the two jobs in position \( j \) and \( k \). The size of the neighborhood is \( (n-1)! \). Experiments show that such a neighborhood yields local minima and requires high calculation time.

- **Exchange move:** given a permutation \( \pi \) of \( n \) jobs; \( j \) and \( k \) \( \in \{1, \ldots, n\} \), two positions in \( \pi \), the permutation that results of such a move is obtained by placing the jobs in positions \( j \) and \( k \). The size of the neighborhood is \( n(n-1)/2 \), these moves find good quality of schedules but the neighborhood exploration is in \( O(n^3 m) \) which leads to a high computational time.

- **Insertion move:** given a permutation \( \pi \) of \( n \) jobs; \( j \) and \( k \) \( \in \{1, \ldots, n\} \), two positions in \( \pi \), the permutation that results of such move is obtained by exchanging the job from position \( i \) to position \( j \). For all neighborhood types the positions \( j \) and \( k \) can be specified randomly or by applying some predefined criterion.

In our implementation approach, we use the insertion move neighborhood structure. The move strategy consists to move different operations of the last executed job in the current solution, and insert them in the inactivity intervals calculated by the job Agent. The Job Agents try to find the new location for each operation that respects the precedence constraints and time lags constraints. After moving all operations, the Job Agent generates the neighborhood solution and starts the evaluation step.

3.4.4 Evaluation of the current solution

Once the neighborhood step finished, the Job Agent evaluates the neighborhoods of the current solution and chooses the best neighbor to be used for the next iteration.

3.4.5 Tabu List

An important parameter of the Tabu Search metaheuristic is the use of a tabu list in which we save different past movement of jobs. The tabu list size was fixed and
3.4.6 Diversification

The diversification step allows getting out of a local optimum. In fact, if the number of iteration after the last improvement exceeds such predefined number, it will be necessary to start the diversification process. In this case the Job Agent generates a new schedule using the NEH process in order to get out of a local optimum and provide a good exploration of the search space.

The NEH algorithm for makespan minimization is composed of different steps; in the first step, jobs are ordered in non-increasing sums of processing times on the machines. In the second step, we take the first two jobs and schedule them in order to minimize the partial makespan as if there were only these two jobs. Then, we insert the other jobs at their positions which minimize the partial makespan among the possible ones.

Once the new schedule was generated with the NEH heuristic, the resolution process of the tabu search algorithm was started by the Job Agent, the new solution of the NEH heuristic was considered as an initial solution and the number of iteration after the last improvement is reset.

3.4.7 Stopping criterion

The stopping criterion used is the maximum number of iteration. The resolution process attempts to attend the solution after a predefined number of iteration. Then, the Job Agent returns the best found scheduling as the best solution.

4. Experimental results

4.1 Implementing environment

To evaluate the efficiency of the parallel tabu search approach used for solving the problem, we have done same experiments of job shop scheduling problems with additional minimum and maximum time lags. The CAPTS-JSTL is implemented in the JADE platform on an Intel Core i3 with 3GB.

The parallel tabu search was evaluated among different instances of the job shop problem from the literature. We use the instances of Fisher and Thompson [9], Lawrence [18] and Carlier [3]. The following table presents details of instances used.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Number of Jobs</th>
<th>Number of machines</th>
<th>Number of operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ft06</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>la01</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>la02</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>la03</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>la04</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>la05</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>car5</td>
<td>10</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>car6</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>car7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>car8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

For all instances, $TL_{min}^{ij, ij+1} = 0$. For each instance, $TL_{max}^{ij, ij+1} = \{0, 0.5, 1, 2\}$. Instances are designed with Name, $TL_{min}^{ij, ij+1}$, $TL_{max}^{ij, ij+1}$, for example ft06_0_0.5 is the instance of Fisher and Thompson 6 with $TL_{min}^{ij, ij+1} = 0$ and $TL_{max}^{ij, ij+1} = 0.5$.

We executed 1000 iterations for each instance. For instances of Fisher and Thompson and Lawrence, we compare the CAPTS-JSTL with both centralized and distributed approaches. Results are compared with the approach “based on operation insertion heuristic OI” of [4] which used the genetic algorithm and the generalized disjunctive constraint propagation “based on a job insertion heuristic JI” [2] and the centralized tabu search algorithm [5]. Then we compare the results with the “distributed approach MATS-JSTL” [13] which is a multi-agent approach based on tabu search algorithm.

For instances of Carlier, we choose to compare our parallel tabu searches with the centralized one [5] and the memetic algorithm [6]. Then, we compared our CAPTS-JSTL with the distributed approach MATS-JSTL [13]. The results of comparison between several approaches are given in the following tables.

4.2 Comparison results

4.2.1 Fisher and Thompson instances

For the ft06 instances, our CAPTS-JSTL approach gives better results than the Job Insertion heuristic method [2]
and the multi agent method (MATS-JSTL) [13] in 100% of instances. Compared with the genetic algorithm of [4] our approach gives better results in 50% of instances and compared with the centralized tabu search [5] we find better makespan value in 75% of instances, see table 2.

4.2.2 Lawrence instances

Table 3

<table>
<thead>
<tr>
<th>Instances</th>
<th>JI</th>
<th>OI</th>
<th>TS</th>
<th>MATS-JSTL</th>
<th>CAPTS-JSTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>la1_0_0</td>
<td>1258</td>
<td>1504</td>
<td>1473</td>
<td>1020</td>
<td>1012</td>
</tr>
<tr>
<td>la1_0_0.5</td>
<td>1063</td>
<td>1474</td>
<td>758</td>
<td>980</td>
<td>924</td>
</tr>
<tr>
<td>la1_0_1</td>
<td>928</td>
<td>1114</td>
<td>916</td>
<td>907</td>
<td>891</td>
</tr>
<tr>
<td>la1_0_2</td>
<td>967</td>
<td>948</td>
<td>732</td>
<td>894</td>
<td>856</td>
</tr>
<tr>
<td>la2_0_0</td>
<td>1082</td>
<td>1416</td>
<td>1436</td>
<td>1085</td>
<td>1034</td>
</tr>
<tr>
<td>la2_0_0.5</td>
<td>1011</td>
<td>1207</td>
<td>1153</td>
<td>973</td>
<td>952</td>
</tr>
<tr>
<td>la2_0_1</td>
<td>935</td>
<td>1136</td>
<td>900</td>
<td>694</td>
<td>686</td>
</tr>
<tr>
<td>la2_0_2</td>
<td>928</td>
<td>895</td>
<td>681</td>
<td>874</td>
<td>673</td>
</tr>
<tr>
<td>la3_0_0</td>
<td>1081</td>
<td>1192</td>
<td>1108</td>
<td>1178</td>
<td>1018</td>
</tr>
<tr>
<td>la3_0_0.5</td>
<td>930</td>
<td>1085</td>
<td>1052</td>
<td>946</td>
<td>874</td>
</tr>
<tr>
<td>la3_0_1</td>
<td>886</td>
<td>931</td>
<td>847</td>
<td>894</td>
<td>718</td>
</tr>
<tr>
<td>la3_0_2</td>
<td>808</td>
<td>787</td>
<td>671</td>
<td>761</td>
<td>630</td>
</tr>
<tr>
<td>la4_0_0</td>
<td>1207</td>
<td>1346</td>
<td>1275</td>
<td>1393</td>
<td>1237</td>
</tr>
<tr>
<td>la4_0_0.5</td>
<td>780</td>
<td>1156</td>
<td>1106</td>
<td>1234</td>
<td>1015</td>
</tr>
<tr>
<td>la4_0_1</td>
<td>1010</td>
<td>857</td>
<td>950</td>
<td>915</td>
<td>826</td>
</tr>
<tr>
<td>la4_0_2</td>
<td>982</td>
<td>838</td>
<td>642</td>
<td>827</td>
<td>783</td>
</tr>
<tr>
<td>la5_0_0</td>
<td>1080</td>
<td>1224</td>
<td>1128</td>
<td>1168</td>
<td>1168</td>
</tr>
<tr>
<td>la5_0_0.5</td>
<td>935</td>
<td>1208</td>
<td>957</td>
<td>962</td>
<td>817</td>
</tr>
<tr>
<td>la5_0_1</td>
<td>814</td>
<td>964</td>
<td>761</td>
<td>803</td>
<td>624</td>
</tr>
<tr>
<td>la5_0_2</td>
<td>749</td>
<td>683</td>
<td>600</td>
<td>763</td>
<td>593</td>
</tr>
</tbody>
</table>

For the Lawrence instances with 10 jobs and 5 machines, our CAPTS-JSTL approach gives better results than the “Job Insertion heuristic” [2] and tabu search method of [5] in 90% of instances. Compared with genetic algorithm of [4] and the “MATS-JSTL” of [13], our approach gives better results in 100% of Instances, see table 3.

4.2.2 Carlier instances

Table 4

<table>
<thead>
<tr>
<th>Instances</th>
<th>TS</th>
<th>Memetic</th>
<th>MATS-JSTL</th>
<th>CAPTS-JSTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car5_0_0</td>
<td>111495</td>
<td>7821</td>
<td>9431</td>
<td>8145</td>
</tr>
<tr>
<td>Car5_0_0.5</td>
<td>10293</td>
<td>7821</td>
<td>8761</td>
<td>7762</td>
</tr>
<tr>
<td>Car5_0_1</td>
<td>8910</td>
<td>7821</td>
<td>8435</td>
<td>7724</td>
</tr>
<tr>
<td>Car5_0_2</td>
<td>8281</td>
<td>7720</td>
<td>8174</td>
<td>7702</td>
</tr>
<tr>
<td>Car6_0_0</td>
<td>11243</td>
<td>8313</td>
<td>9784</td>
<td>8367</td>
</tr>
<tr>
<td>Car6_0_0.5</td>
<td>9100</td>
<td>8330</td>
<td>9632</td>
<td>8324</td>
</tr>
</tbody>
</table>

For Carlier’s instances, results show that our distributed approach gives makespan value better than centralized tabu search of [5] and the distributed approach of [13] in 100 % of instances and better than the memetic algorithm of [6] in 50 % of instances, see table 4.

5. Conclusion and Future works

We propose the parallel tabu search algorithm in a multi agent approach for solving the Job Shop scheduling problem with additional constraints of minimum and maximum time lags between the start and completion times of successive operations in the same job. Our model consists of an Interface Agent and n competitive Job agents, each one containing the tabu search core, responsible for executing the search process and satisfying all problem constraints. Several experimental results show that the CAPTS-JSTL gives good solutions for most instances of the problem. Due to good results of the proposed approach, we can develop the parallelization of other metaheuristic algorithms in order to solve the same problem. We can also adopt the parallel tabu search for other extensions of the job shop scheduling problem.

References

[5] A. Caumond, M. Gourgand, P. Lacomme and N. Tchernev, Métaheuristiques pour le problème de job shop avec time lags : Jm||ls j(i)Cmax. 5ème conférence Francophone de Modélisation et SIMulation.


