ANALYSIS OF THE HYBRIDIZATION OF TABOO SEARCH AND PARTICLE SWARM OPTIMIZATION HEURISTICS FOR THE JOB SHOP SCHEDULING PROBLEM

Tatiana Balbi Fraga, tbfraga@iprj.uerj.br
Antônio José da Silva Neto, ajsneto@iprj.uerj.br

José Eduardo Souza de Cursi, eduardo.souza@insa-rouen.fr

Francisco José da Cunha Pires Soeiro, soeiro@uerj.br

Abstract. The scheduling problem is defined as a set of jobs that must be simultaneously processed by a set of machines. Here, each job must be processed exactly once at each machine. The processing order of the jobs through the machines is predefined and can be different for each one. As each machine can process just one job at a time, the objective of this problem consists of defining what moment each job must be processed by each machine in order to minimize the makespan, i.e., the completion time of the last job finished. This is a combinatorial optimization problem defined as strongly NP-Hard, in fact it is known by the researchers as one of the most difficult combinatorial optimization problems and despite there is a lot of methods and heuristics that could solve it, none can find optimal solutions for all the benchmarks proposed, even when considering a small problem, i.e., considering a small number of jobs and machines. From among the heuristics that could be applied to this problem, Taboo Search and Particle Swarm Optimization show a good performance for the majority of benchmarks. Usually, the Taboo Search heuristic presents a good and fast convergence to the optimal or sub-optimal points but this convergence is often interrupted by a cyclic process, on the other hand, the Particle Swarm Optimization heuristic tends towards a convergence by means of a lot of computational time. As each proposed heuristics are its positive and negative characteristics, nowadays some researchers start applying the hybrids heuristics for profit the best characteristics of each one. This work presents an analysis of the different forms of hybridization of these two heuristics, Taboo Search and Particle Swarm Optimization, showing what aspects must be considered to achieve a best solution of the one obtained by the original heuristics in a feasible computational time.

Keywords: Job Shop Scheduling Problem; Taboo Search; Particle Swarm Optimization; Hybridization.

1. INTRODUCTION

As described by Blazewics et al. (1996), a job shop consists of a set of different machines that perform the necessary operations to process different jobs. Each job has a specified processing order through the machines and, as each job visits each machine exactly once, we can define a job as a set of operations where each operation represents the part of job that must be processed by a given machine during a fixed processing time. Once the processing of one operation starts, it cannot be interrupted (non-preemption) and each job can be performed by just one machine at a time, what introduces a precedence constraint among operations of the same job (job constraint), i.e. the starting time of an operation is always bigger than the finish time of its job-preceding one. In addition, each machine can process only one job at a time (machine constraint) and, while the machine sequence of each job is fixed, when considering a set of jobs that must be simultaneously processed, the problem consists of finding the job sequences on machines which minimize the makespan, i.e. the completion time of the last operation finished. This is a combinatorial optimization problem which is classified as a strongly NP-Hard and considered as a difficult challenging problem in the literature.

Due to its stubborn nature, many researchers have focused on solve it and a wide variety of procedures have been proposed in the literature. These works generally adopt either global optimization or approximation techniques, also referred to as global and local approaches, respectively. The global approaches look for a schedule corresponding to the global minimum of the makespan among all the feasible schedules, what generally involves computationally expensive global optimization procedures and difficulties concerning the convergence. In these approaches, we may find algorithms which are mainly concerned by the computational efficiency (usually referred to as efficient methods), such as the Johnson's method (1954), and algorithms which are mainly interested in the exploration of the space of the possible solutions (usually referred to as enumerative methods). In first case an optimal solution is built by following a simple set of rules which exactly determines the processing order and in second case all feasible solutions are generated...
one by one and elimination procedures are used in order to restrict the domain of search and prevent from a complete space of solution\(^1\). The principal enumerative methods are the mathematical programing techniques, as mixed integer programing of Manne (1960), and the several Branch and Bound algorithms (Carlier and Pinson 1989, Brucker et al. 1994, etc.). As established by Jain and Meeran (1999a), the global optimization approaches have not yet attained the required level in order to solve general job shop scheduling problems, specially when considering a large number of jobs and machines. According to the authors, there are not efficient methods capable of solving job shop scheduling problems were the number of machines and the jobs is greater than 3 and in the case of enumerative methods, as the time requirement usually increases exponentially or as a high degree polynomial for a linear increase in problem size, even the few successes obtained by the Branch and Bound algorithms is mainly attributed to the technology available rather than the techniques used.

Contrary to the optimization methods, the approximation procedures usually deliver a good (but not necessary optimal) solution in acceptable time. Therefore, despite a large variety of approximation heuristics, ranging from priority dispatch rules (Panwalkar and Iskander, 1977) and bottleneck based heuristics (Adams et al. 1988) to artificial intelligence methods, as constraint satisfaction techniques, neural networks and ant colony optimization algorithms, their relative success is mainly attributed to a class of the approximation procedures named local search heuristics and, until now, none of the heuristics developed has been able to find optimal solutions for all the benchmarks proposed, so investigations are still under progress in order to improve their performance.

1.1. Local search heuristics

The local search heuristics - also referred to as neighborhood techniques - are characterized by the fact that each iteration generates a set of new solutions in the neighborhood of a set of "parent" ones by introducing small perturbations (usually called "move") of each available solution. At each step, the available solutions are formed by the initial "parents" and the perturbations generated: a selection procedure eliminates a part of the available solutions in order to get the new set of "parents" solution which will be used for the neighborhood generation of the next iteration. The process continue until a given stopping criterion is satisfied.

\[
\begin{array}{l}
generate a set of \ N \ initial \ solutions \ (N \geq 1) \ (the \ initial \ parent \ solutions) \\
while \ stopping \ criterion \ is \ not \ satisfied \\
generate \ a \ neighborhood \ (perturbation) \\
generate \ the \ new \ parent \ solutions \ (selection) \\
end \ while \\
returns \ the \ best \ solution \ found
\end{array}
\]

Figure 1. Scheme of local search heuristics.

According to the choice of the initial solution, neighborhood generation techniques, selection process and stop criteria, different methods and heuristics are obtained. These methods and heuristics can be either deterministic or non-deterministic according to the use of a random procedure: when no random procedure is applied to generate the initial solution, or generate or select the neighborhood, the method or heuristic is deterministic, when random procedure is utilized they become non-deterministic. Actually, Taboo Search is considered as the most effective heuristic in the framework of the job shop scheduling problem - Taboo Search is a procedure that applies a list of “taboo” solutions in order to try to prevent the search process of getting stuck in a locally optimal solution. Other important heuristic is Simulated Annealing, a random search technique that was primarily introduced as an analogy from statistical physics for the computer simulation of the annealing process of a hot metal until its minimum energy state is reached. In contrast to Taboo Search, this non-deterministic heuristic tries to escape from locally optimum solutions by applying the Metropolis dynamics (Metropolis et al., 1953). As stated by Jain and Meeran (1999a) the single Simulated Annealing approach, such as the algorithms develop by Matsuo et al. (1988) and Van Laarhoven et al. (1992), remains quite poor and does not lead to good results, but its basic ideas – namely the methods of neighborhood generation and the Metropolis dynamics – may be easily introduced in other local search approaches. While Simulated Annealing is suggested by physical science, Genetic Algorithms are search techniques based on abstract models of natural evolution where the quality of “individuals” builds to the highest level compatible with the environment (constraints of the

\(^1\) In the literature a scheduling, defined by the specified sequences of operation in each machine, is also referred as solution. The reader must keep in mind that this expression does not concern the solution of the optimization problem, which is referred in the literature as optimal solution. In this paper all the expressions: solution, possible solution and sequence of operations, will be used in order to make reference to a scheduling.
problem). As presented by Jain and Meeran (1999b) despite the fact that many elaborated algorithms have been proposed, the construction of a convenient representation of the space of the potential solutions (admissible set) for the job shop problems, coherent with crossover and mutation operations, is still considered as a hard problem. In addition, many Genetic Algorithm methods are unable to converge to an optimal solution. Similar to Genetic Algorithms, Particle Swarm Optimization (Kennedy and Eberhart 1995) is a population based heuristic inspired by nature. This time, the social behavior of bird flocking or fish schooling was the source of inspiration of its concepts. Lian et al. (2006) demonstrated that when applying the same crossover and mutation operators of Genetic Algorithms in Particle Swarm and comparing the results, the second method leads to better results. Taboo Search, Genetic algorithms and Particle Swarm Optimization are usually of the family of non-deterministic heuristics but, from the abstract point of view, they structures can or not involve random steps according their implementation.

As stated before, despite the partial success achieved by the researches in developing powerful techniques, until now there is no heuristic which can find optimal solutions for all the benchmarks proposed (Jain and Meeran 1999a). In this framework, a possible approach consists in the use of a hybrid heuristic, i.e., in the simultaneous use of deterministic and non-deterministic approximation procedures. In this work we analyze some of the different possibilities of combining a deterministic and a non-deterministic local search methods, which are, respectively, the of Taboo Search method with a deterministic version of the neighborhood structure presented by Nowicki and Smutnicki (1996) without back-propagation and the Similar Particle Swarm optimization proposed by Lian et al. (2006), showing what aspects must be considered in order to obtain an improved method, with increased robustness and able to get better results than each single method separately, with a reasonable computational cost. The rest of this paper is organized as follows: The sections 2 and 3 give a brief overview of Taboo Search and Particle Swarm heuristics, respectively, and introduce the methods which hybridization will be analyzed here. The model of hybridization applied is presented in section 4. Section 5 discusses the results obtained by the different possibilities of hybridization and presents a new hybrid method. Finally, Section 6 summarize the contributions of this paper and gives the conclusions.

2 TABOO SEARCH

The Taboo Search heuristic starts with an initial solution, generated randomly or by a constructive method, which is stored as the current and the best solutions. Then, at each iteration, a neighborhood is generated by applying small perturbations (usually called "move") on the current solution and so this solution is replaced by the best neighbor, i.e. the best solution into the neighborhood generated that doesn't belong to the taboo list. The taboo list is a vector containing the last \( t \) current solutions, and is introduced as a way to prevent the search process of getting stuck in a locally optimal solution. An aspiration criterion can be defined in order to allow the replacement of the current solution by a solution that belongs to the taboo list when it is useful for the search, i.e. when this solution is better than the best solution found. The best solution is replaced by the best neighbor if the latter presents a smaller makespan than the first one. The process continues until a given stopping criterion is satisfied.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Generate an initial solution</td>
</tr>
<tr>
<td>2</td>
<td>Store the initial solution as the current and the best solutions</td>
</tr>
<tr>
<td>3</td>
<td>While stopping criterion is not satisfied</td>
</tr>
<tr>
<td></td>
<td>- Generate a neighborhood</td>
</tr>
<tr>
<td></td>
<td>- Replace the current solution by the not taboo best neighbor</td>
</tr>
<tr>
<td></td>
<td>- If the makespan of the best neighbor is smaller than the makespan of the best solution replace the best solution by the best neighbor</td>
</tr>
<tr>
<td></td>
<td>- Update the taboo list adding the new current solution and removing the oldest one (when taboo list size is bigger than ( t ))</td>
</tr>
<tr>
<td>4</td>
<td>End while</td>
</tr>
<tr>
<td>5</td>
<td>Return the best solution</td>
</tr>
</tbody>
</table>

Figure 2. Scheme of Taboo Search heuristic.

The Fig. 2 presents a general Taboo Search method. In this scheme the neighborhood generation process has a direct impact on the efficiency of the method and several neighborhood structures have been presented in literature. Among them, the one usually referred as N5 introduces the real breakthrough in both efficiency and effectiveness for the job shop problem. This method generates a neighborhood substantially smaller than the others and will be applied in this work (for details concerning to the implementation see Nowicki and Smutnicki (1996)).
The problem of Taboo Search is that their performance is quite sensitive to the initial solution and the tuning of its parameters, when they are not well adjusted to each specific problem, the method falls into cyclic processes and the global optimum is not found. The Tab. 1 presents a sensitivity analysis to the FT10 problem. These values demonstrate that the method is not robust and that it is very difficult to find optimal solutions if the exact value of the parameters are not known. An optimal solution (930) was found by the related program in just 15 seconds of CPU time, setting tenure = 30.

Table 1. Taboo Search sensitivity analysis for 10 initial random solutions for the FT10 problem – makespan X tenure (with the following fixed parameters: neighborhood generating method = N5; critical path = 0; and maximum number of iterations without upgrade = 10.000).

<table>
<thead>
<tr>
<th>Initial Solution</th>
<th>Tenure 5</th>
<th>Tenure 6</th>
<th>Tenure 7</th>
<th>Tenure 8</th>
<th>Tenure 9</th>
<th>Tenure 10</th>
<th>Tenure 11</th>
<th>Tenure 12</th>
<th>Tenure 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result 1</td>
<td>1668</td>
<td>1035</td>
<td>971</td>
<td>951</td>
<td>990</td>
<td>949</td>
<td>961</td>
<td>943</td>
<td>949</td>
</tr>
<tr>
<td>Result 2</td>
<td>1695</td>
<td>951</td>
<td>948</td>
<td>945</td>
<td>948</td>
<td>945</td>
<td>943</td>
<td>954</td>
<td>951</td>
</tr>
<tr>
<td>Result 3</td>
<td>1977</td>
<td>994</td>
<td>954</td>
<td>972</td>
<td>940</td>
<td>945</td>
<td>943</td>
<td>938</td>
<td>954</td>
</tr>
<tr>
<td>Result 4</td>
<td>1807</td>
<td>1136</td>
<td>980</td>
<td>963</td>
<td>945</td>
<td>934</td>
<td>953</td>
<td>956</td>
<td>952</td>
</tr>
<tr>
<td>Result 5</td>
<td>1749</td>
<td>988</td>
<td>1031</td>
<td>940</td>
<td>944</td>
<td>945</td>
<td>946</td>
<td>950</td>
<td>955</td>
</tr>
<tr>
<td>Result 6</td>
<td>1629</td>
<td>1017</td>
<td>968</td>
<td>1041</td>
<td>949</td>
<td>966</td>
<td>960</td>
<td>951</td>
<td>951</td>
</tr>
<tr>
<td>Result 7</td>
<td>1922</td>
<td>949</td>
<td>958</td>
<td>954</td>
<td>940</td>
<td>954</td>
<td>963</td>
<td>948</td>
<td>965</td>
</tr>
<tr>
<td>Result 8</td>
<td>1783</td>
<td>968</td>
<td>956</td>
<td>953</td>
<td>951</td>
<td>951</td>
<td>940</td>
<td>940</td>
<td>947</td>
</tr>
<tr>
<td>Result 9</td>
<td>1701</td>
<td>1104</td>
<td>969</td>
<td>983</td>
<td>937</td>
<td>949</td>
<td>948</td>
<td>949</td>
<td>954</td>
</tr>
<tr>
<td>Result 10</td>
<td>1766</td>
<td>1005</td>
<td>1023</td>
<td>962</td>
<td>944</td>
<td>951</td>
<td>948</td>
<td>952</td>
<td>939</td>
</tr>
</tbody>
</table>

The cyclic process of Taboo Search can be visualized in Tab. 2 which shows that, after a specific maximum number of iterations without upgrade, even a great increase in the value of this parameter is not capable of improving the solution found. While the times is proportionally increased, the solution falls into a local optima and another techniques are necessary to improve the result.

Table 2. Taboo Search sensitivity analysis for 10 initial random solutions for the FT10 problem – makespan X maximum number of iterations without upgrade (with the following fixed parameters: neighborhood generating method = N5; critical path = 0; and tenure = 5).

<table>
<thead>
<tr>
<th>Initial Solution</th>
<th>Maximum number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Result 1</td>
<td>1668</td>
</tr>
<tr>
<td>Result 2</td>
<td>1695</td>
</tr>
<tr>
<td>Result 3</td>
<td>1977</td>
</tr>
<tr>
<td>Result 4</td>
<td>1807</td>
</tr>
<tr>
<td>Result 5</td>
<td>1749</td>
</tr>
<tr>
<td>Result 6</td>
<td>1629</td>
</tr>
<tr>
<td>Result 7</td>
<td>1922</td>
</tr>
<tr>
<td>Result 8</td>
<td>1783</td>
</tr>
<tr>
<td>Result 9</td>
<td>1701</td>
</tr>
<tr>
<td>Result 10</td>
<td>1766</td>
</tr>
</tbody>
</table>
3. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a population based heuristic developed by Kennedy and Eberhart (1995). In this method, each solution is interpreted as the position of a particle (individual of the swarm), which is represented by a vector \( d \) dimensional, where \( d \) is the number of variables that must be adjusted. The trajectories of the particles look for the position corresponding to an optimal solution. The implementation of PSO can be described as follows: the initial position of the swarm is randomly generated and then the individuals or potential solutions, named particles, searches for an optima by updating its own positions. At each iteration the position of each particle is adjusted according to its velocity and position are adjusted with the following equations:

where the position of each particle is mapped by the work procedure code (see below), which consider only feasible movement are adjusted. Lian (2006) proposed a Similar Particle Swarm algorithm to the job shop scheduling for an optima by updating its own positions. At each iteration the position of each particle is adjusted according to its position if necessary. The inertia weight \( w \), first proposed by Shi and Eberhart (1998), is used to control exploration and exploitation. A larger \( w \) can prevent particles to becoming trapped in local optima, and a smaller \( w \) encourages particle exploiting the same search space area. The constants \( c_1, c_2 \) are learning factors used to decide whether particles prefers moving toward a \( pbest \) or \( gbest \) position. Usually \( c_1 = c_2 = 2 \). The \( r_1 \) and \( r_2 \) are random variables between 0 and 1.

\[
\begin{align*}
v_i(k+1) &= w \times v_i(k) + c_1 r_1 (pbest_i(k) - x_i(k)) + c_2 r_2 (gbest(k) - x_i(k)) \\
x_i(k+1) &= x_i(k) + v_i(k+1)
\end{align*}
\]

The inertia weight \( w \), first proposed by Shi and Eberhart (1998), is used to control exploration and exploitation. A larger \( w \) can prevent particles to becoming trapped in local optima, and a smaller \( w \) encourages particle exploiting the same search space area. The constants \( c_1, c_2 \) are learning factors used to decide whether particles prefers moving toward a \( pbest \) or \( gbest \) position. Usually \( c_1 = c_2 = 2 \). The \( r_1 \) and \( r_2 \) are random variables between 0 and 1.

---

initialize a population of particles with random initial positions and velocities on \( d \) dimensional space
while stop criteria is not satisfied
    update the velocity of each particle, according to Eq. (1)
    update the position of each particle, according to Eq. (2)
    evaluate the fitness value of each position according to the desired optimization fitness function and, at the same time, update \( pbest \) and \( gbest \) position if necessary
end while
return the best global solution (gbest)

---

Figure 3. Scheme of Particle Swarm heuristic for problems in continuous space.

The original Particle Swarm has been developed to solve continuous optimization problems. When working with combinatorial optimization ones, we have to modify the representation of the positions and the way where velocity and movement are adjusted. Lian et al. (2006) proposed a Similar Particle Swarm algorithm to the job shop scheduling where the position of each particle is mapped by the work procedure code (see below), which consider only feasible solutions, and its velocity and position are adjusted with the following equations:

\[
\begin{align*}
v_i(k+1) &= pbest_i(k) \ominus gbest(k) \\
v_i(k+1) &= \begin{cases} 
   v_i(k+1) & \text{if } mut_i = 0 \\
   M(v_i(k+1)) & \text{if } mut_i = 1
\end{cases} \\
x_i(k+1) &= x_i(k) \ominus v_i(k+1) \\
x_i(k+1) &= \begin{cases} 
   x_i(k+1) & \text{if } mut_i = 0 \\
   M(x_i(k+1)) & \text{if } mut_i = 1
\end{cases}
\end{align*}
\]

where \( \ominus \) and \( M(v) \) represents, respectively, the crossover and mutation operators applied in Genetic Algorithms and the boolean variable \( mut_i \) is a flag destined to indicate if the mutation operation is \( mut_i = 1 \) or not \( mut_i = 0 \) applied on particle \( i \). At each iteration, \( N \) particles are randomly chosen to suffer a mutation operation:
\[ \sum_{i=1}^{NP} \text{mut}_i = N \]  \hspace{1cm} (7)

The authors tested 4 crossover (C1 – C4) and 10 mutation (M1 – M10) operators on three benchmarks (FT6, FT10 and FT20 – Fisher and Thompson, 1963) and the Similar Particle Swarm has shown to be more efficient than Genetic Algorithms to solve job shop scheduling problems. In this work we apply the M7 (single job moving-inserting) mutation operator and a new crossover operator, described as C1 with 1 floating point. The working procedure code and the reported operators are described as follows:

**Work procedure code:** in work procedure code (WPC), a scheduling is represented by a sequence of the indexes of jobs in which one is repeated in the sequent by the number of its corresponding operations. A corresponding operation can be known by the job and the position that it appears, i.e. the first time that a job appears in the sequence represents the first operation of this job, the second time that the same job appears in the sequence represents the second operation of this job and so on. For example: in a job shop problem with 3 jobs and 3 machines a sequence can be represented as follows:

\[ \text{WPC (0 2 1 0 1 2 2 1 0) = sequence of operations (1 7 4 2 5 8 9 6 3)} \]

**C1 with 1 floating point:** a crossing point is randomly selected along the length of the first chromosome (sequence of operations). The sub-section of jobs from the first position to the crossing point is copied into the offspring. The remaining places of the offspring are filled up by taking in order each legitimate gene of the second chromosome.

**Single job moving-inserting (M7):** one moving and one inserting point are randomly selected along the length of the chromosome then the job at moving point are moved and placed in the inserting point.

Despite the Similar Particle Swarm presents better results than the Genetic Algorithms, this results are not yet reach the desired but and additionally, as Taboo Search heuristics, the Similar Particle Swarm is quite sensitive to the value of its parameter, as, for example, the percentage of particles that suffer a mutation operation per iteration where small values lead the algorithm to falling in a cyclic process and the big values block its convergence.

**4. THE HYBRID APPROACH**

As previously observed, the hybrid approach consists in the simultaneous use of deterministic and a non-deterministic approaches. The method of hybridization proposed in this paper consists in utilizing the schema of local search methods (Fig. 1) combining different methods (the deterministic Taboo Search, as presented in section 2, and the non-deterministic Similar Particle Swarm Optimization, as presented in section 3), i.e. applying different methods to generate and improve the neighborhood or utilizing successively the solution found by a determined method as initial solution of the other one. This work analyzes some of these possibilities in order to select what are the main aspects to be considered in order to achieve an improvement in the results of both the original methods. The Figs. 3, 4 and 5 present the schemes of hybridization analyzed in this paper, respectively defined as hybrid successive application, hybrid neighborhood and hybrid improved neighborhood. In hybrid successive application scheme (Fig. 4) a set of initial solutions is randomly generated and so at each cycle the different methods are successively applied on the set of solutions furnished by the preceding one. The same procedure is used in hybrid improved neighborhood scheme (Fig. 6) but the initial set of solutions of a cycle is issued from a random selection on the group of all solutions obtained by the different methods. In hybrid neighborhood scheme (Fig. 5) in each cycle all the methods are applied on the same initial solution (initial solution of the cycle) and then a set of solutions is randomly selected into the group of solutions furnished by all the methods.
generate \( N \) initial solutions and store them as the \( N \) current solutions
while stop criteria is not satisfied
  apply method 1 on the \( N \) current solutions
  return \( X \) selected solutions (best or not) generated by this method
  apply method 2 on the \( X \) solutions generated by the method 1
  return \( Y \) selected solutions (best or not) generated by this method
  \[
  \vdots
  \]
  apply method \( M \) on the \( L \) solutions generated by the method \( M - 1 \)
  return \( N \) selected solutions (best or not) generated by this method and store them as the \( N \) current solutions
end while
return the best global solution

Figure 4. Scheme of hybrid successive application.

generate \( N \) initial solutions and store them as the \( N \) current solutions
while stop criteria is not satisfied
  apply method 1 on the \( N \) current solutions
  return \( X \) selected solutions (best or not) generated by this method
  apply method 2 on the \( N \) current solutions
  return \( Y \) selected solutions (best or not) generated by this method
  \[
  \vdots
  \]
  apply method \( M \) on the \( N \) current solutions
  return \( W \) selected solutions (best or not) generated by this method
  select \( N \) solutions (best or not) into \( K \) solutions generated by the methods 1 to \( M \) \( (K = X + Y + \ldots + M) \)
  and store them as the \( N \) current solutions
end while
return the best global solution

Figure 5. Scheme of hybrid neighborhood.
The fast convergence of Taboo Search leads to an optimal global point. However, this method is quite sensible to big values lead the algorithm to a cyclic process and big values block its convergence. When applying successively Taboo Search and Similar Particle Swarm, as Taboo Search walks through local minimums, the Similar Particle Swarm tends to converge to the best point found by the previous method and, unless \( P_{mut} \) has a big value, the Particle Swarm is not able to improve the results of Taboo Search. When setting big values for \( P_{mut} \) the Particle Swarm allows Taboo Search to escape from local minimums and the fast convergence of Taboo Search leads to an optimal global point. However, this method is quite sensible to large values of \( P_{mut} \), as the Particle Swarm is not able to improve the results of Taboo Search.

**5. COMPUTATIONAL EXPERIMENTS**

Table 3. Resume of hybrid applications for 10 initial random solutions for the FT10 problem (with the following fixed parameters: maximum number of cycles = 5 – Taboo Search: neighborhood generating method = N5; critical path = 0; and tenure = 8; – Similar Particle Swarm: crossover method = C1 with 1 floating point; mutation method = M7; and percentage of particles that suffer mutation = 20% and 80%).

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Method</th>
<th>Successive Application</th>
<th>Hybrid Neighborhood</th>
<th>Hybrid improved neighborhood</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SPSO-TSA 20%</td>
<td>SPSO-TSA 80%</td>
<td>TSA-SPSO 20%</td>
</tr>
<tr>
<td>TSA : 10 SPSO : 10</td>
<td>Makepan</td>
<td>1043</td>
<td>962</td>
<td>1027</td>
</tr>
<tr>
<td></td>
<td>NSolEval</td>
<td>8722</td>
<td>14119</td>
<td>10067</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>TSA : 100 SPSO : 100</td>
<td>Makepan</td>
<td>983</td>
<td>942</td>
<td>955</td>
</tr>
<tr>
<td></td>
<td>NSolEval</td>
<td>68056</td>
<td>97256</td>
<td>72524</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>19</td>
<td>26</td>
<td>21</td>
</tr>
<tr>
<td>TSA : 1,000 SPSO : 1,000</td>
<td>Makepan</td>
<td>945</td>
<td>938</td>
<td>945</td>
</tr>
<tr>
<td></td>
<td>NSolEval</td>
<td>627445</td>
<td>794550</td>
<td>629890</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>162</td>
<td>207</td>
<td>184</td>
</tr>
<tr>
<td>TSA : 10,000 SPSO : 10,000</td>
<td>Makepan</td>
<td>940</td>
<td>938</td>
<td>937</td>
</tr>
<tr>
<td></td>
<td>NSolEval</td>
<td>5891463</td>
<td>1381868</td>
<td>5867281</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>1513</td>
<td>355</td>
<td>1506</td>
</tr>
<tr>
<td>TSA : 10,000 SPSO : 1</td>
<td>Makepan</td>
<td>930</td>
<td>934</td>
<td>934</td>
</tr>
<tr>
<td></td>
<td>NSolEval</td>
<td>1731952</td>
<td>4542479</td>
<td>3851663</td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>452</td>
<td>1132</td>
<td>1002</td>
</tr>
</tbody>
</table>

**Figure 6. Scheme of hybrid improved neighborhood.**
parameter. Otherwise when Particle Swarm is applied with no more than one iteration the hybrid method tends to combine the global convergence property of Particle Swarm with the local convergence property of Taboo Search and all results tend to converge to a global optimum. When applying the hybrid neighborhood and the hybrid improved neighborhood schemes we can observe the same behavior except that the Particle Swarm tends to concentrate on the best available solution. The hybrid improved neighborhood is the scheme that presents the best solutions (Tab.3).

5.1. A functional Hybrid TS/PSO algorithm

The Hybrid TS/PSO method proposed in this work consists in the successive application (Fig. 4) of Taboo Search and Similar Particle Swarm where the Similar Particle Swarm is applied with no more than one iteration. This method is described as follows:

Step 1: Generate $N$ random initial solutions and store it as the current solutions.
Step 2: Apply Taboo Search with $N_{ite}$ (large quantity) iterations on each current solution. Store the result as the current solutions.
Step 3: For $L$ cycles apply Similar Particle Swarm with one iteration on current solutions and on each final position of the particles apply Taboo Search with $N_{ite}$ (large quantity) iterations. Store the solutions found as the current solutions.
Step 4: Apply Similar Particle Swarm with $N_{ite}$ (large quantity) iterations on current solutions.
Step 5: Apply Taboo Search with $N_{ite}$ (large quantity) iterations on the best solution found and return the solution.

Figure 7. Hybrid Taboo Search – Similar Particle Swarm Optimization method.

Table 6. Hybrid TS/SPSO method for FT10 (comparison with Taboo Search and Similar Particle Swarm** results with the following fixed parameters: maximum number of cycles = 10 – Taboo Search: neighborhood generating method = N5; critical path = 0; and maximum number of iterations without upgrade ($N_{ite}$) = 10.000; – Similar Particle Swarm: crossover method = C1 with 1 floating point; mutation method = M7; and number of iterations ($N_{ite}$) = 10.000).

<table>
<thead>
<tr>
<th>$P_{mut}$</th>
<th>Tenure</th>
<th>TS*</th>
<th>0%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>949</td>
<td>949</td>
<td>940</td>
<td>930</td>
<td>946</td>
<td>930</td>
<td>930</td>
<td>937</td>
<td>940</td>
<td>937</td>
<td>940</td>
<td>945</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>948</td>
<td>930</td>
<td>935</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>935</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>940</td>
<td>934</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>937</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>934</td>
<td>934</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>938</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>938</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td>930</td>
<td></td>
</tr>
</tbody>
</table>

* Best solution found by application in 10 random solutions.
** The best makespan found by Similar Particle Swarm of Lian et al. (2006) for FT10 is 937.

Table 6 presents the results for FT10 and the comparison with the original Taboo Search and Similar Particle Swarm** methods. These results shown that the hybrid method is able to find very good solutions independent of the value of parameters. The CPU time of application of the complete method with 10 cycles is about 30 min but the majority of the optimal results was found in the first cycles with no more than 5 min.

6. CONCLUSIONS

In this work we analyzed three possible schemes of hybridization of Taboo Search and Similar Particle Swarm methods defined as hybrid successive application, hybrid neighborhood and hybrid improved neighborhood. The results have shown that all schemes are able to find best solutions than the original methods when comparing the same parameters. Finally a Hybrid TS/SPSO method was proposed and it was tested on FT10 problem and the results
demonstrated its robustness and its ability to significantly improve the original techniques, generating better results in an acceptable computing time. Although the proposed algorithm is tested in an important representative instance, a more comprehensive computational study should be made to test the efficiency of the approach considered. This will be matter of future work.

7. ACKNOWLEDGEMENTS

The authors acknowledge the financial support provided by FAPERJ, Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro, CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico, and CAPES, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

8. REFERENCES


9. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.