Three hybridization models based on local search scheme for job shop scheduling problem

Tatiana Balbi Fraga

Department of Production Engineering, Centro Acadêmico do Agreste, Universidade Federal de Pernambuco, Caruaru, PE, Brazil.

Email: tatianabf_8@hotmail.com

Abstract. This work presents three different hybridization models based on the general schema of Local Search Heuristics, named Hybrid Successive Application, Hybrid Neighborhood, and Hybrid Improved Neighborhood. Despite similar approaches might have already been presented in the literature in other contexts, in this work these models are applied to analyzes the solution of the job shop scheduling problem, with the heuristics Taboo Search and Particle Swarm Optimization. Besides, we investigate some aspects that must be considered in order to achieve better solutions than those obtained by the original heuristics. The results demonstrate that the algorithms derived from these three hybrid models are more robust than the original algorithms and able to get better results than those found by the single Taboo Search.

1. Introduction

The job shop scheduling problem (JSP) comes from manufacturing environments where jobs usually differ considerably in their processing sequence and times. This problem is classified as NP-Hard in strong sense [1] and it is widely known in the technical literature as one of the most difficult issues in the combinatorial analysis field. As stated by Zhang *et al.* [2], the JSP complexity can be illustrated by the fact that a relatively small instance with 10 jobs and 10 machines, proposed by Fisher and Thompson [3], remained unsolved for more than a quarter of a century and actually it remains difficult to solve optimally benchmarks with more than 20 jobs and 20 machines. Briefly, the JSP can be presented as follows: a set of m machines, $\{M_k\}_{k=1}^{m}$, and a set of n jobs, $\{J_i\}_{i=1}^{n}$, are considered. Each job J_i must be processed exactly once in each machine M_k in a predetermined order, which may vary from one job to another. Therefore, each job can be understood as a sequence of operations, $J_i = \left(O_{ik_s}\right)_{s=1}^{m}$, where O_{ik_s} is the operation of job J_i that must be processed by the machine M_{k_s} for a deterministic processing time, $t_{ik_s} \geq 0$. Additionally: each machine can process only one job at a time; no job can be processed by two or more machines at the same moment; and preemption is not allowed. The objective of this problem is to determine the order in which jobs should be processed on each machine in order to minimize the length of time it takes to complete all operations for all jobs (usually referred to as makespan), without disregarding the restrictions imposed by the problem. Note that the dimensionality of each JSP instance can be specified as $n \times m$ and the total number of solutions is defined by $(n!)^m$.

Due to its practical importance and its complex nature, the JSP has attracted the attention of many researchers, and a wide variety of algorithms has been proposed to solve this problem. According to

Jain and Meeran [4], these algorithms can be classified, in one hand, as optimization algorithms, when they guarantee the generation of optimal solutions (eventually at infinite time and generally involving a high computational cost), and on the other hand, as approximation algorithms, which provide good solutions (but not necessarily optimal) with a reasonable computational cost. Among the optimization algorithms are the family of constructive, or efficient, algorithms for which an optimal solution is built following a simple set of rules defined by the algorithm itself [5, 6, 7], and enumerative algorithms in which successive tentative solutions are generated, one by one, and elimination procedures are used in order to restrict the search space.

The main enumerative algorithms are the Mathematical Programming Techniques, as the Mixed Integer of Manne [8], and the various Branch and Bound algorithms [9, 10]. The approximation algorithms are constituted by a great diversity of families of algorithms ranging from Priority Dispatch Rules [11] and Bottleneck Based Heuristics [12, 13, 14] to artificial intelligence methodologies, such as Constraint Satisfaction Techniques [15], Neural Networks [16] and Ant Colony Optimization algorithms [17], but their relative success is mainly attributed to a class of approximation algorithms named Local Search Heuristics, that includes the well known Simulated Annealing [18, 19], Genetic Algorithms [20, 21], Taboo Search [22, 23, 24] and, more recently, Particle Swarm Optimization [25, 26, 27] heuristics, and until now no algorithm developed was able to find optimal solutions in an acceptable processing time for all the benchmark problems proposed in the literature. An extensive discussion of the family of algorithms applied to solve the job shop scheduling problem is presented by Blazewicz *et al.* [28] and by Jain and Meeran [4].

As each family of algorithms has its own positive and negative aspects, in order to take the best characteristics of the already existing algorithms and overcome their weaknesses, a possible approach consists in the use of hybrid algorithms, and this has been the focus of many authors. These hybrid algorithms can be classified as: *blended*, when generated by the insertion of specific procedures of a family of algorithms, as the taboo list of Taboo Search or the Metropolis criterion of Simulated Annealing, into an algorithm belonging to another family of different nature; *unblended*, when generated by the simultaneous use of algorithms of different nature, *i.e.* deterministic and non-deterministic or population-based and individual-based, so that each algorithm conserves its own characteristics; or *mixed*, when both blended and unblended procedures are applied. As an example of hybrid algorithms, we mention the work by Pezzella and Merelli [29], which based on the idea that the results generated by Taboo Search are tightly dependent on its initial solutions, proposed an unblended hybrid algorithm where the Shifting Bottleneck Procedure of Adams *et al.* [12] is applied in order to generate the initial solutions. Wang and Zheng [30] proposed an unblended hybrid algorithm where initially a Genetic Algorithm is applied in order to generate the initial population, then, for each initial element the Simulated Annealing performs a Metropolis sampling until the equilibrium condition is reached, and then the Genetic Algorithms uses the solution found by Simulated Annealing to continue the parallel evolution. Azizi and Zolfaghari [19] added a taboo list in their algorithm, named Adaptive Simulated Annealing, generating a blended hybrid algorithm that improved its performance significantly. Zhang *et al.* [2] presented an unblended hybrid algorithm where initially the Simulated Annealing is applied in order to find good solutions in a relatively wide search space and then the Taboo Search is applied in order to intensify the search process.

In this work the Local Search Heuristics are designed in a general representation and, based on this representation, three hybridization models are presented and analyzed with two local search algorithms, a deterministic version of the Taboo Search algorithm, with the neighborhood structure proposed by Nowicki and Smutnicki [23] without back-propagation, and the Similar Particle Swarm Optimization algorithm suggested by Lian *et al.* [25]. As the main result, the present work shows that the complementary nature of Taboo Search and Particle Swarm Optimization guarantee the efficacy and robustness of the three models presented providing better results than the separate application of both algorithms. The remainder of the paper is organized as follows: an introduction for the Local Search Heuristics as well as the Taboo Search and Similar Particle Swarm Optimization algorithms used to test the hybridization models are given in sections 2, 3 and 4, respectively. Section 5 gives a brief description of the three hybridization models investigated. Section 6 discusses the experimental results; and, finally, in section 7 conclusions are drawn, and some proposals for future research are suggested.

2. Local Search Heuristics

The Local Search Heuristics - also referred to as neighborhood techniques - are characterized by the fact that, at each iteration, a set of new solutions is generated in the neighborhood of a set of "parent" ones by introducing small perturbations (usually called "moves") of the latter (figure 1). Then a selection procedure eliminates a part of the available solutions in order to get the new set of "parent" solutions which will be used for the neighborhood generation of the next iteration. The process continues until a given stopping criterion is reached.

According to the choice of the neighborhood generation techniques and selection criterion, different heuristics are obtained. As an example, in Taboo Search heuristic, at each cycle, the neighborhood is generated from a single parent solution, usually swapping the positions of two of its operations, and then the new parent solution is formed by the best solution within the neighborhood that does not belong to the taboo list. The Simulated Annealing heuristic usually applies the same neighborhood generation techniques that in Taboo Search but, at each cycle, just one neighbor is generated from a single parent solution and, if its makespan is smaller than the parent solution's makespan, this solution replaces the parent one, otherwise, a Metropolis criterion is applied in order to define if the generated solution replaces or not the parent one. In the case of the Genetic Algorithms and Particle Swarm Optimization heuristics, at each cycle, the neighborhood is generated from a set of parent solutions (population), usually applying crossover and mutation operations on the pairs of these solutions. The basic difference of both methods is the way in which these pairs of solutions are selected. While the Genetic Algorithm usually applies a selection criterion based on a probability related to an efficiency index of each solution, the selection criterion applied by the Particle Swarm Optimization is always based on the best solution ever generated and the best solution visited by each individual.

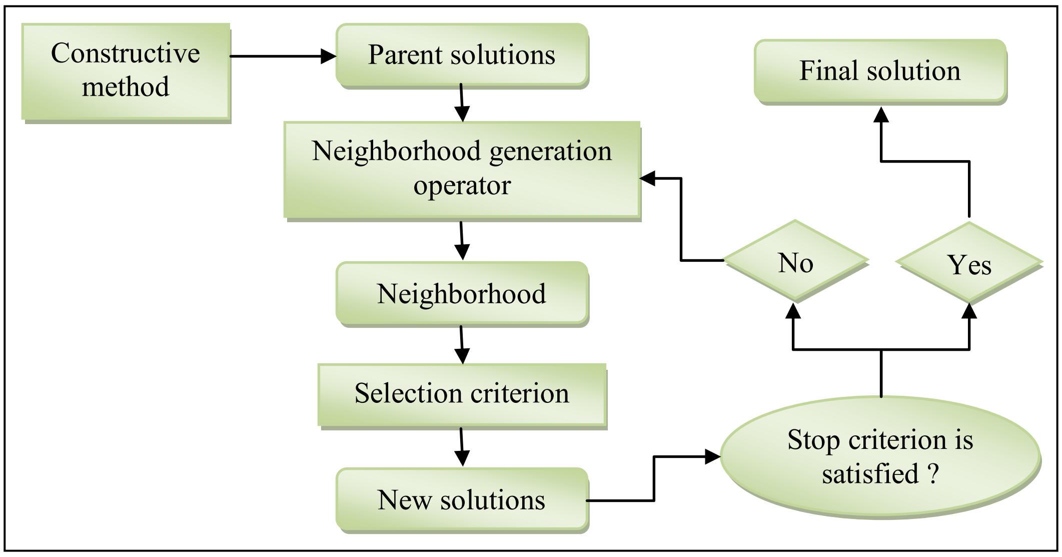

Figure 1. Scheme of Local Search Heuristics.

Additionally, the choice of the specific method that generates the initial parent solutions, the specific operator that must be applied to generate the neighborhood and the specific selection and stop criteria, allow generating a wide variety of algorithms for each family. As stated previously, in this work two algorithms were chosen to be applied in the hybrid models, a deterministic version of the

Taboo Search algorithm, with the neighborhood structure proposed by Nowicki and Smutnicki [23] without back-propagation, and the Similar Particle Swarm Optimization algorithm suggested by Lian *et al.*[25]. In the following sections, both of them will be presented.

3. Taboo Search algorithm

The Taboo Search (TS) heuristic was originally developed by Glover [31] and its first application to JSP is attributed to Taillard [22]. In its classical form the TS algorithm can be described as follows: at first an initial solution is 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 "moves" (*i.e.*, small perturbations) to the current solution and so this solution is replaced by a non taboo neighbor, *i.e.* a solution in the neighborhood generated (usually the best one), under the restriction that its generating movement doesn't belong to the taboo list. This last one is a vector containing the inverse movements that generated the last t current solutions, and is introduced as a way to prevent the search process of getting trapped in a locally optimal solution – where t is the size of taboo list and is always referred as "tenure". 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 current solution if the latter presents a smaller makespan than the first one. The process continues until a given stopping criterion is satisfied.

```
generate an initial solution
store the initial solution as the current and the best solutions
while stopping criterion is not satisfied
    generate a neighborhood
    replace the current solution by the not taboo best neighbor
    update the taboo list adding the inverse move that generated the new current solution
    and removing the oldest one (when taboo list size is bigger than $t$)
    if the makespan of the current solution is smaller than the makespan of the best solution
        replace the best solution by the current solution
    end if
end while
return the best solution
```

Figure 2. Taboo Search algorithm.

Figure 2 presents a general TS algorithm. 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 the literature (*i.e.*, see [22, 23, 24]). Among them, the one usually referred as to 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 was chosen to be applied in this work. The N5 method, the best solution selecting process, the taboo list, and the stopping criterion of the TS algorithm applied in this work will be described in more detail in the following sections.

3.1. N5 neighborhood generation method

Given a solution, a critical path can be defined as a sequence of operations wherein:

- the last element in this sequence is the last operation completed. So its final processing time is the makespan of the respective solution;

- for each operation in this sequence, except for the first one, its immediately preceding operation is either: the operation that precedes it in the machine order or in the task order. Between the two, it is the one which presents the higher finishing time that is chosen;
- the start time of the first operation of this sequence is zero.

It is important to note that a solution can contain more than one critical path, as happens when there are two or more operations that are finalized last, or when operations preceding an operation on the machine and task orders are completed at the same moment. In their article, Nowicki and Smutnicki [23] argue that the choice of the critical path does not have a strong influence on the final result. The authors suggest that it should be chosen randomly. In this study, the critical path is selected in accordance with the following priorities:

- if two operations are finalized last, the operation selected will be the one belonging to the lowest machine index;
- if the operations, which precede an operation in machine and task orders, finish at the same moment, the operation selected will be the one above in the task order.

The critical path can also be subdivided into blocks of operations where each block consists of a subsequence of successive operations that must be processed by the same machine. By the N5 method, a neighborhood is formed by swapping the first two (and the last two) operations of each block on a critical path (where each swap will generate a new neighbor). For the first block, only the last two operations are swapped and for the last block, only the first two operations are swapped. If any block is formed by only one operation, no exchange will be made and if it is formed by two operations only one exchange must be performed. (for more details see Nowicki and Smutnicki [23]).

3.2. Selecting the best non-taboo solution
In the TS operator applied in this work, a taboo list consists of a vector containing the moves inverse to those used to generate the latest t current solutions (where t is a parameter, the value of which can be adjusted). Thus, the best neighbor (the solution chosen as the new current solution) is the solution with the smallest makespan, whose motion generator is not on the taboo list. Additionally, an aspiration criterion is considered which allows the current solution to be replaced by a solution whose motion generator belongs to the taboo list if the makespan of this solution is less than that of the best solution found so far. Should there be a tie for the best solution, the current solution remains the same.

3.3. Stopping criterion
In this algorithm the iterative process stops when after $Nite$ iterations the best solution found is not updated, *i.e.* no new current solution has a lower makespan than that of the best solution found. $Nite$ is a parameter, the value of which can be adjusted.

4. Similar Particle Swarm Optimization
Particle Swarm Optimization (PSO) is a population-based heuristic developed by Kennedy and Eberhart [32]. In this method, each solution is interpreted as the position of a particle (individual of the swarm), which is represented by a D-dimensional vector, where D is the number of variables that must be determined. The trajectories of the particles look for the position corresponding to an optimal solution. According to the representation given in figure 3, 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, search for an optimum by updating their own positions. At each iteration the position of each particle is adjusted according to its velocity which is randomly generated towards the best position visited by the particle (p_{best}) and the best position visited by the swarm (g_{best}). At iteration k, for each particle i ($1 \leq i \leq NP$, where NP represents the total number of particles), the velocity v_i and the position x_i can be updated by the following equations:

$$v_i(k+1) = w \times v_i(k) + c_1 r_1 \left(p_{\text{best}_i}(k) - x_i(k)\right) + c_2 r_2 \left(g_{\text{best}}(k) - x_i(k)\right) \quad (1)$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (2)$$

The inertia weight w, first proposed by Shi and Eberhart [33], 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 exploitation of the same search space area. The constants c_1 and c_2 are learning factors used to decide whether particles prefer moving toward a p_{best} or g_{best} 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 stopping criterion 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 $p_{\text{best}}$ and $g_{\text{best}}$ position if necessary
end while
return the best global solution ($g_{\text{best}}$)
```

Figure 3. Particle Swarm Optimization algorithm for problems in continuous space.

The original PSO algorithm was developed to solve continuous optimization problems. When working with combinatorial optimization problems, we have to modify the representation of the positions and the way how velocity and movement are adjusted (some examples of adaptations of the original PSO applied for JSP can be found in [25, 26, 27]). Lian *et al.* [25] proposed a Similar Particle Swarm Optimization (SPSO) algorithm for the JSP where the position of each particle is mapped by the work procedure code (see below), which considers only feasible solutions, and its velocity and position are adjusted according to the following equations:

$$v_i(k+1) = p_{\text{best}_i}(k)\ \Theta\ g_{\text{best}}(k) \quad (3)$$

$$v_i(k+1) = \begin{Bmatrix} v_i(k+1) & if\ \ mut_i = 0 \\ M\left(v_i(k+1)\right) & if\ mut_i = 1 \end{Bmatrix} \quad (4)$$

$$x_i(k+1) = x_i(k)\ \Theta\ v_i(k) \quad (5)$$

$$x_i(k+1) = \begin{Bmatrix} x_i(k+1) & if\ \ mut_i = 0 \\ M\left(x_i(k+1)\right) & if\ mut_i = 1 \end{Bmatrix} \quad (6)$$

where Θ and $M(x)$ 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_{mut} particles are randomly chosen to suffer a mutation operation:

$$\sum_{i=1}^{NP} mut_i = N_{\text{mut}} \tag{7}$$

The mutation probability (p_{mut}), *i.e.* the percentage of particles that suffer mutation at each iteration, is defined by:

$$p_{\text{mut}} = \frac{N_{\text{mut}}}{NP} \times 100 \tag{8}$$

The authors tested four crossover (C1 – C4) and ten mutation (M1 – M10) operators on three benchmark problems proposed by Fisher and Thompson [32] named FT6, FT10 and FT20, and the SPSO algorithm has shown to be more efficient than the Genetic Algorithms in the solution of job shop scheduling problems. Despite the SPSO presenting better results than the Genetic Algorithms, these results have not yet reached the desired purpose and, additionally, like the TS heuristics, the SPSO is quite sensitive to the value of its control parameters, as, for example, the percentage of particles that suffer a mutation operation by iteration in which small values lead the algorithm to falling in a cyclic process, and the large values impaired its convergence.

From all four crossover and ten mutation operators we chose to show in this work the results of the application of 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 in the following sections.

4.1. Work procedure code

In the work procedure code (WPC), a schedule is represented by a sequence of the indexes of jobs in which each job index is repeated by the number of its corresponding operations. A corresponding operation can be known by the job index and the position that it appears, *i.e.* the first time that a job index appears in the sequence represents the first operation of this job, the second time that the same job index 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 possible solution can be represented as follows:

WPC (1 3 2 1 2 3 3 2 1)

In WPC, the position of each operation determines the order of precedence in which it should be processed. So the sequence, in which tasks should be processed on each machine, is determined by taking operations for their respective machines in the same order as that in which they appear in the solution given by WPC. One inconvenience of this form of representation is that two apparently different solutions, when plotted by WPC, can portray the same solution (when represented by the sequence of tasks on each machine). Observe that the N5 neighborhood generating method is based on the critical path. Therefore, for the application of the TS operator, the solution should be represented by the Task Sequence Code (TSC) (*i.e.*, the sequence of tasks on each machine). During the iterative process, the representation of the solutions is converted to WPC or TSC, depending on the algorithm applied.

4.2. C1 with one 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.

4.3. 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.

5. Hybridization models

The hybridization idea presented in this work is based on the schema of Local Search Heuristics (figure 1) and consists basically on generating a hybrid neighborhood by applying two or more neighborhood generation operators. Table 1 presents some of the possible neighborhood generation operators, classified according to their nature. These operators can be the same neighborhood generation operators applied in the Local Search Heuristics, as previously mentioned, or any other algorithm capable of transforming a group of parent solutions in a new group of solutions, through the perturbation of the first, as in the case of the own Local Search Heuristics when using as their respective initial solutions, the set of parent solutions. The basic difference between both kinds of operators is that, in the first case, the hybrid algorithm generated will be blended and, in the last case, unblended.

Table 1. Neighborhood generation operators.

				Local Search Heuristics	
	Critical path swap operators	Crossover operators	Mutation operators	New solutions are generated by introducing small perturbations in just one solution	New solutions are generated by crossing two solutions
Deterministic	N1, N2, N3, N4, N5, N6 – Zhang *et al.* [24]			Taboo Search	
Nondeterministic		C1, C2, C3, C4 – Lian *et al.* [25]	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10 – Lian *et al.* [25]	Taboo Search, Simulated Annealing	Genetic Algorithms, Similar Particle Swarm Optimization

Usually the hybridization of neighborhood generation operators enables the generation of algorithms more robust, *i.e.* less sensitive to the adjustment of parameters and able to produce better results than each single algorithm separately, provided that the operators used in the hybridization are of different nature. In this study the TS algorithm (as presented in section 3) and the SPSO algorithm (as presented in section 4) were chosen as operators for generating the hybrid neighborhood because they present different aspects in different directions. This choice was based on the following considerations:

- The TS heuristic improves every single parent solution in order to bring it to a minimum point that can be local or global. The SPSO heuristic generates an iteration between the parent solutions, converging the solutions to an optimal point (local or global) common to all solutions;
- The TS heuristic can be easily presented as deterministic while the random aspect is a fundamental and essential part of the SPSO heuristic;
- The TS algorithm with the neighborhood generation method N5 proposed by Nowicki and Smutnicki [23] is still one of the most effective and efficient algorithms for solving the JSP;
- Despite the SPSO and Genetic Algorithms heuristics presenting similar characteristics, the SPSO algorithm proposed by Lian *et al.* [25] has achieved better results than the Genetic Algorithms proposed in the literature;

- Despite another PSO algorithms were developed to solve JSP, in the SPSO algorithm proposed by Lian *et al.* [25] it is not necessary to set the parameters of the original PSO (*i.e.*, the inertia weight and the learning factors).

This work analyzes some of these possibilities of combining the operators in order to select which are the main aspects to be considered to achieve an improvement in the results of the original algorithms. The following sections present the hybridization schemes analyzed in this paper, respectively defined as Hybrid Successive Application (HSA), Hybrid Neighborhood (HN) and Hybrid Improved Neighborhood (HIN).

5.1. Hybrid Successive Application

In the Hybrid Successive Application scheme (figure 4) a set of initial parent solutions is randomly generated. Then, at each cycle, the first neighborhood generation operator (NGO) is applied to the parent solutions and the other NGO are successively applied on the set of solutions furnished by the preceding one. The final neighborhood provided in each cycle corresponds to the neighborhood generated by the last NGO.

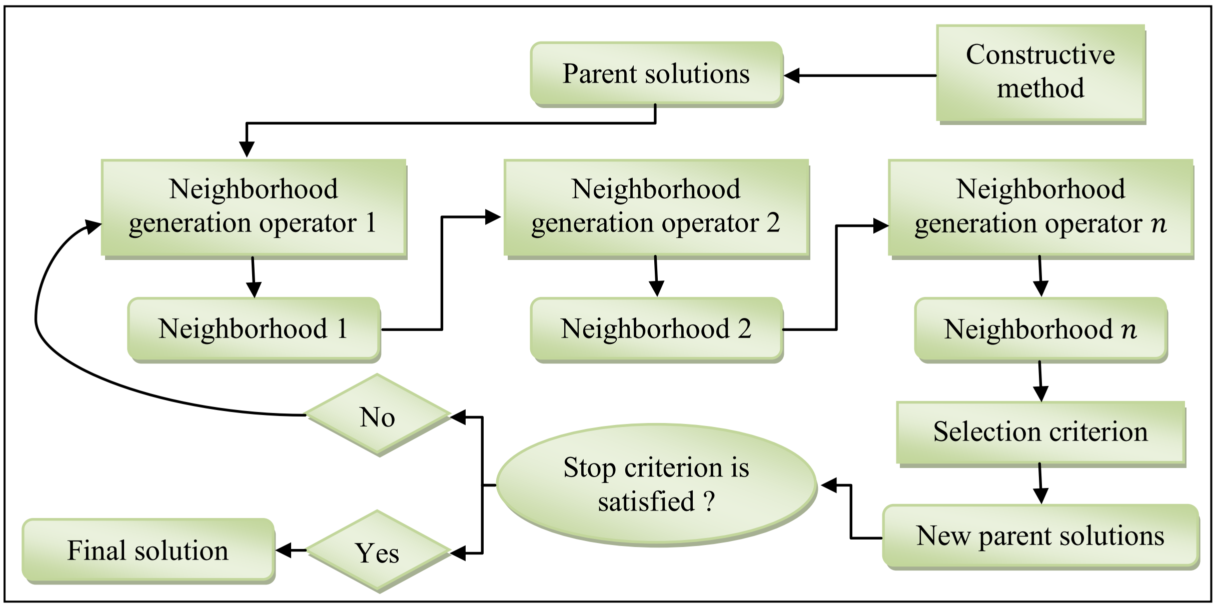

Figure 4. Scheme of the Hybrid Successive Application.

5.2. Hybrid Neighborhood

In the Hybrid Neighborhood scheme (figure 5) in each cycle all the NGO are applied on the same set of parent solutions (set of initial solutions of the cycle) and then the set of parent solutions of the next cycle is randomly selected from the group of solutions furnished by all the neighborhood generation operators.

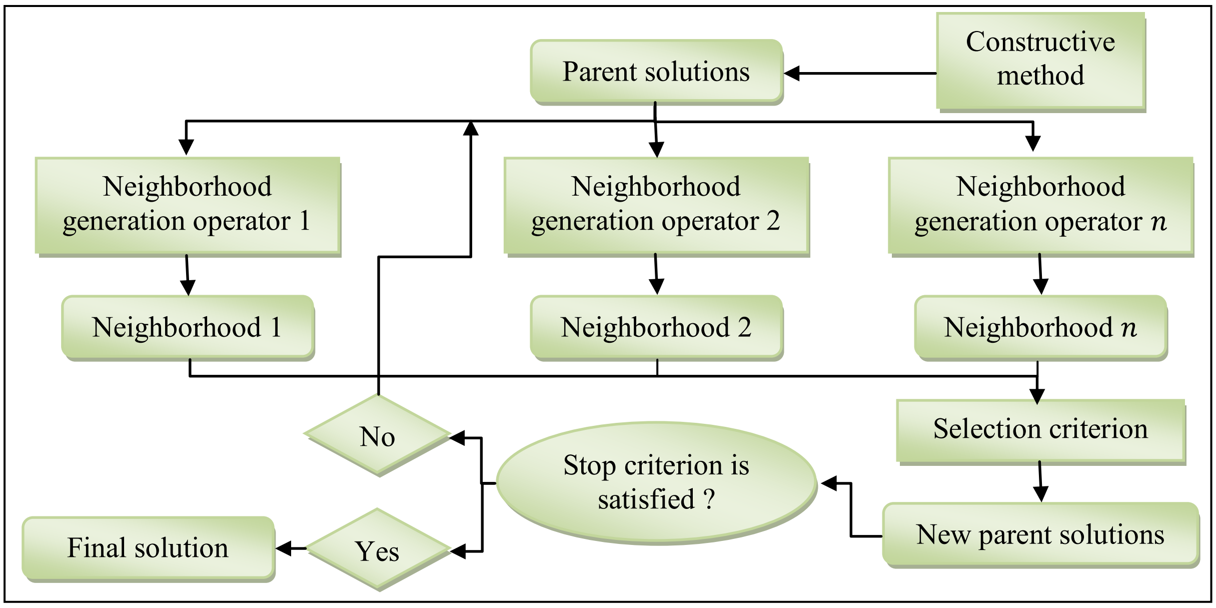

Figure 5. Scheme of the Hybrid Neighborhood.

5.3. *Hybrid Improved Neighborhood*

In the Hybrid Improved Neighborhood scheme (figure 6), the same procedure of the Hybrid Successive Application is used, but the set of new parent solutions of a cycle derives from a random selection on the group of all solutions obtained by the different neighborhood generation operators.

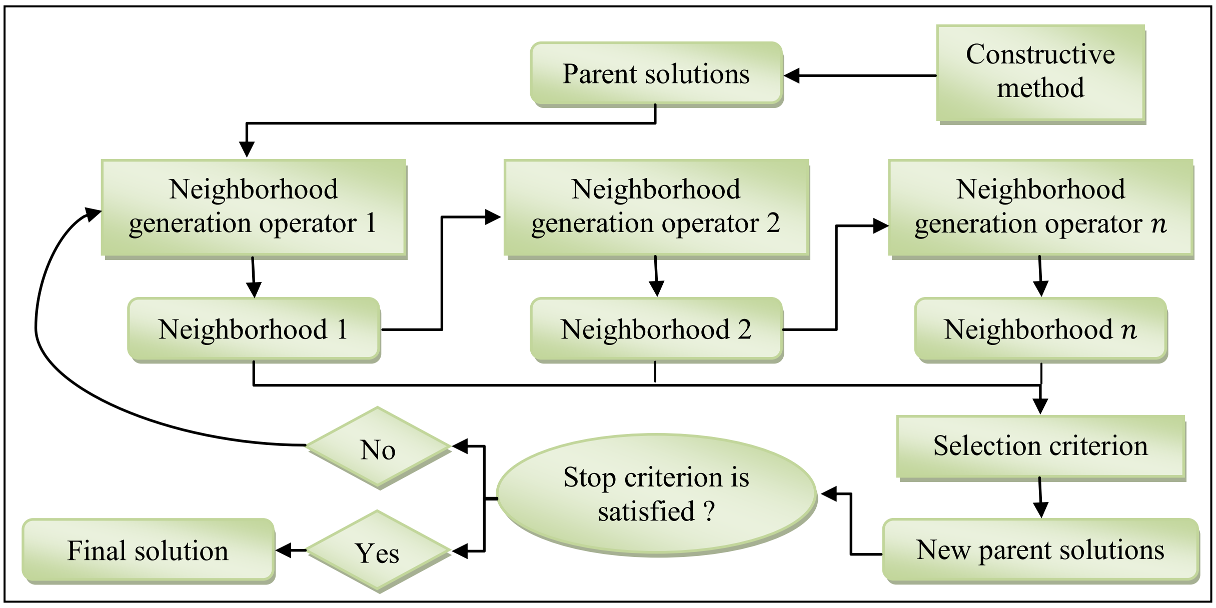

Figure 6. Scheme of the Hybrid Improved Neighborhood.

6. Computational results

To analyze the robustness and efficiency of the three hybridization models proposed in this work, the schemes presented in Figures 4, 5 and 6 were tested on the benchmark problems FT6 (with 6 jobs and 6 machines), FT10 (with 10 jobs and 10 machines) and FT20 (with 20 jobs and 5 machines) proposed by Fisher and Thompson [34] and problems ABZ5 and ABZ6 (with 10 jobs and 10 machines), and ABZ7, ABZ8 and ABZ9 (with 20 jobs and 15 machines) proposed by Adams *et al.* [12] with the algorithms TS, as presented in section 3, and SPSO, as presented in section 4, respectively, and inverting their orders. For each benchmark problem twelve initial solutions (initial configurations) were randomly generated and they were used as initial solutions of all algorithms for all the tests applied. As SPSO is a population based algorithm and TS is an individual based one, the first was applied on the set of solutions and the last was applied on every single solution. As the SPSO is non-deterministic, in order to minimize the distortion generated by the random appearance, each test was repeated five times.

Initially, a sensitivity analysis of the parameters "maximum number of iterations without upgrade" of the TS algorithm and "number of iterations" of the SPSO algorithm was performed. The results obtained demonstrated that when the SPSO algorithm starts with random positions (solutions), the method searches into the space delimited by the initial positions a local optimum and the percentage of particles that suffer mutation (p_{mut}) holds a fundamental role in the convergence process. Small values lead the algorithm to a cyclic process and large values impaired its convergence. When applying successively the TS and the SPSO algorithms, as TS walks through local minima, the SPSO tends to converge to the best point found by the previous algorithm and, unless p_{mut} has a large value, the SPSO is not able to improve the results of TS. When setting large values for p_{mut}, the SPSO allows TS to escape from local minima and the fast convergence of TS leads to a global optimal point. However, this method is quite sensitive to the p_{mut} parameter. Otherwise, when SPSO is applied with no more than one iteration, the hybrid algorithms tend to combine the global convergence property of SPSO with the local convergence property of TS, and all results tends to converge to a global optimum. When applying the HN and the HIN schemes we can observe the same behavior except that the SPSO tends to concentrate on the best available solution.

Based on this observation a sensitivity analysis of the parameters "maximum number of iterations without upgrade" and "tenure" of TS algorithm, and "percentage of particles that suffer mutation" of the SPSO algorithm, as well as a convergence analysis based on the parameter "cycles number" of the hybrid algorithms presented here, were applied fixing the parameter "number of iterations" of SPSO as one. In the next sections we present the results of these analyses - all results generated and an extensive discussion of these analyses are presented in Fraga [35].

6.1. Sensitivity analysis with respect to the maximum number of iterations without upgrade of the makespan

First, a sensitivity analysis was applied on the single TS on each one of the set of initial solutions defined, fixing the value of the parameter "tenure" to eight and setting the parameter "maximum number of iterations without upgrade" to the values 10, 10^2, 10^3, 10^4, 10^5 and 10^6. For all benchmarks analyzed, excepted FT6 for which the method found optimal solutions with no more than 10 iterations, the results demonstrated that the TS method is quite sensible to the initial solution, as it generated different results for each one. For all initial solutions of the defined set, it was identified a saturation point (ST) for the value of the parameter analyzed in which the method falls into cyclic processes and the result cannot be upgraded (figure 7). It was observed that, for most of the initial solutions, ST is about 10.000 for the problems with 10 jobs and 10 machines (FT10, ABZ5 and ABZ6) and about 100.000 for the problems with 20 jobs and 15 machines (ABZ7, ABZ8 and ABZ9), but some solutions presented smaller or larger ST.

Afterwards, the same sensitivity analysis was applied to the three hybrid models proposed here with ten cycles, fixing the value of parameters "tenure" of TS and "percentage of particles that suffer mutation" of SPSO, respectively at eight and 20%, and setting the parameter "maximum number of iterations without upgrade" of TS to the values 10^2, 10^3, 10^4 and 10^5. Despite the hybrid algorithms not being able to find optimal solutions for all benchmark problems, when applied with ten cycles and the defined small set of initial solutions, they demonstrated to be able to significantly improve the results found by Taboo Search (figure 8), even when the order of their neighborhood generation operators was inverted (figure 9). For the 20×15 problems it was also observed that these improvements are smaller when the value of the parameter analyzed is increased. For the majority of the analyses, the

Hybrid Successive Application was the model that generated the best results, especially when considering small values for the parameter analyzed.

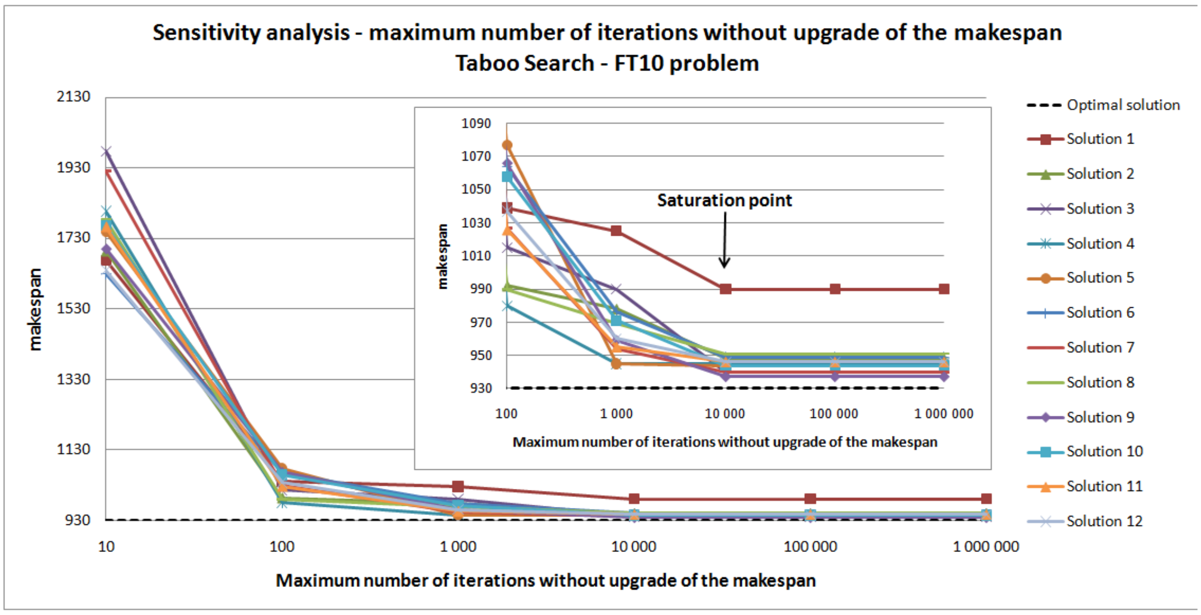

Figure 7. Sensitivity analysis with respect to the "maximum number of iterations without upgrade" of the Taboo Search algorithm for the FT10 problem with the following fixed parameter: tenure = 8.

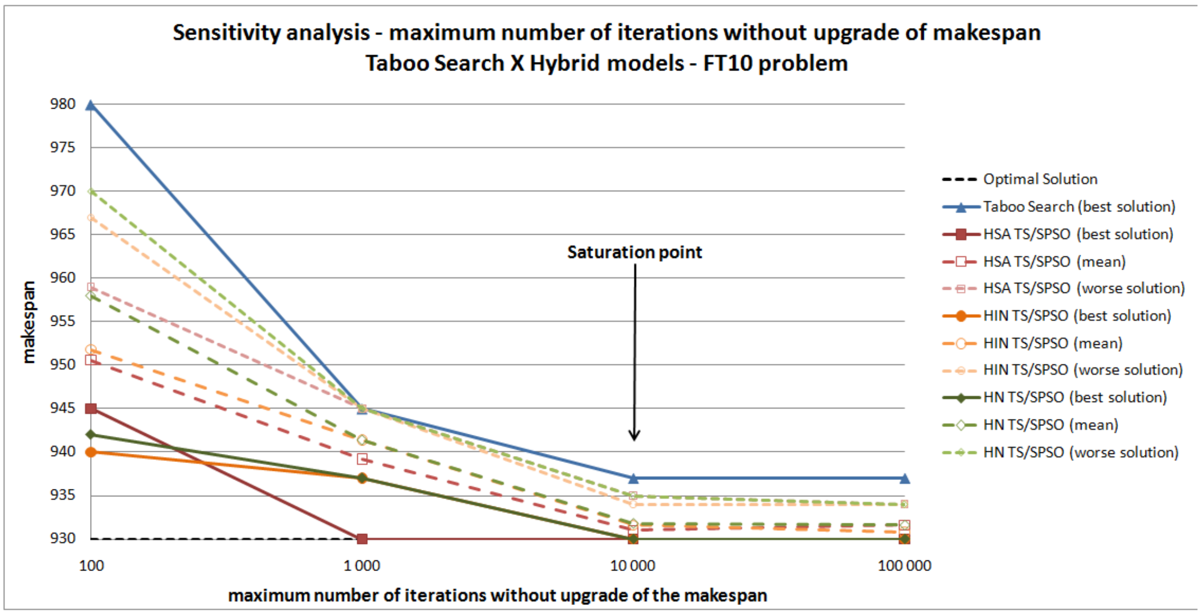

Figure 8. Sensitivity analysis with respect to the "maximum number of iterations without upgrade" of the Hybrid Models (HSA, HIN and HN) for the FT10 problem with the following fixed parameters: tenure = 8; percentage of particles that suffer mutation = 20%; cycles number = 10. Comparison between the best, the mean and the worse results generated by the three hybrid models and the best result generated by the Taboo Search algorithm.

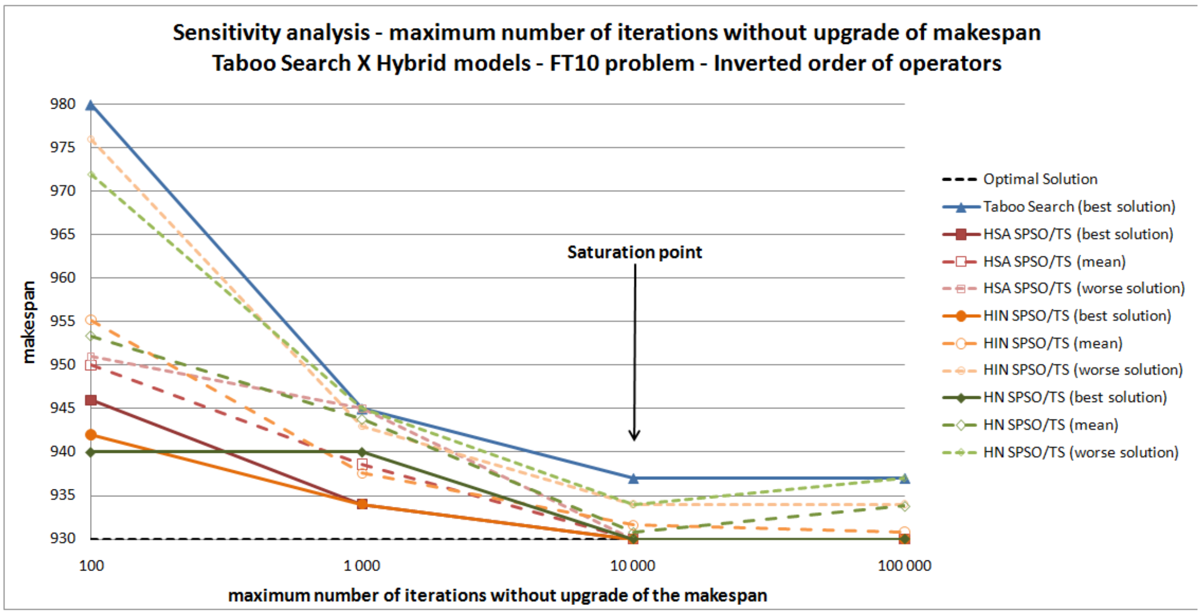

Figure 9. Sensitivity analysis with respect to the "maximum number of iterations without upgrade" of the Hybrid Models (HSA, HIN and HN), with inverted order of the neighborhood generation operators, for the FT10 problem with the following fixed parameters: tenure = 8; percentage of particles that suffer mutation = 20%; cycles number = 10. Comparison between the best, the mean and the worse results generated by the three hybrid models and the best result generated by the Taboo Search algorithm.

6.2. *Sensitivity analysis with respect to the Taboo Search tenure*

The sensitivity analysis with respect to the value of the "tenure" parameter was also first applied to the single TS algorithm on each solution of the set of initial solutions generated for each different benchmark, fixing the value of the parameter "maximum number of iterations without upgrade" of TS at 10^3 for the problem FT20, at 10^4 for the problems FT10, ABZ5 and ABZ6, and at 10^5 for the problems ABZ7, ABZ8 and ABZ9. Also the value of the parameter "tenure" was set from 6 to 15. The results confirmed the high sensitivity of the algorithm to the initial solution and also demonstrated that it is quite sensitive to the tuning of the tenure parameter. Different initial solutions presented different sensitivity for the tuning of this parameter, as can be seen in solution 6 and in solution 11 generated for the FT10 problem (figure 10). After the same analysis was applied to the hybrid models fixing the value of the parameter "maximum number of iterations without upgrade" of TS at $1O^3$ for the problem FT20, at 10^4 for the problems FT10, ABZ5 and ABZ6 and at 10^5 for the problems ABZ7, ABZ8 and ABZ9, the value of parameter "percentage of particles that suffer mutation" of SPSO at 20%, and setting the value of the parameter "tenure" from 6 to 15. The results demonstrated that, despite the hybrid algorithms also presenting sensitivity to the parameter analyzed, this sensitivity is smaller and, when the neighborhood generation operators TS and SPSO were applied, in this order, even the worst result from the five runs for which each test was applied was better or, at least, equal to the best result provided by the single TS, when applied to each component of the same set of initial solutions and with the same parameters. Additionally, the results of best solutions, in most cases, significantly improved the best result provided for the single TS. As the value of the parameter "maximum number of iteration without upgrade" of TS was fixed as 10^4 (for 10×10 problems) or 10^5 (for 20×15 problems), both models the HSA and the HN presented the best results.

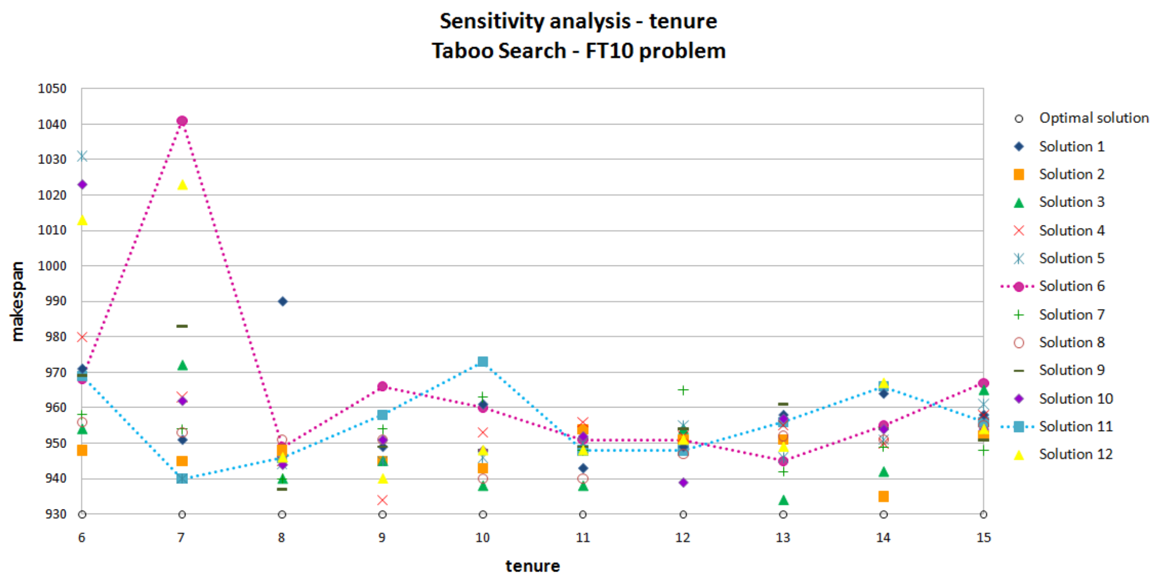

Figure 10. Sensitivity analysis with respect to tenure of the Taboo Search algorithm for the FT10 problem with the following fixed parameter: maximum number of iterations without upgrade = 10^4.

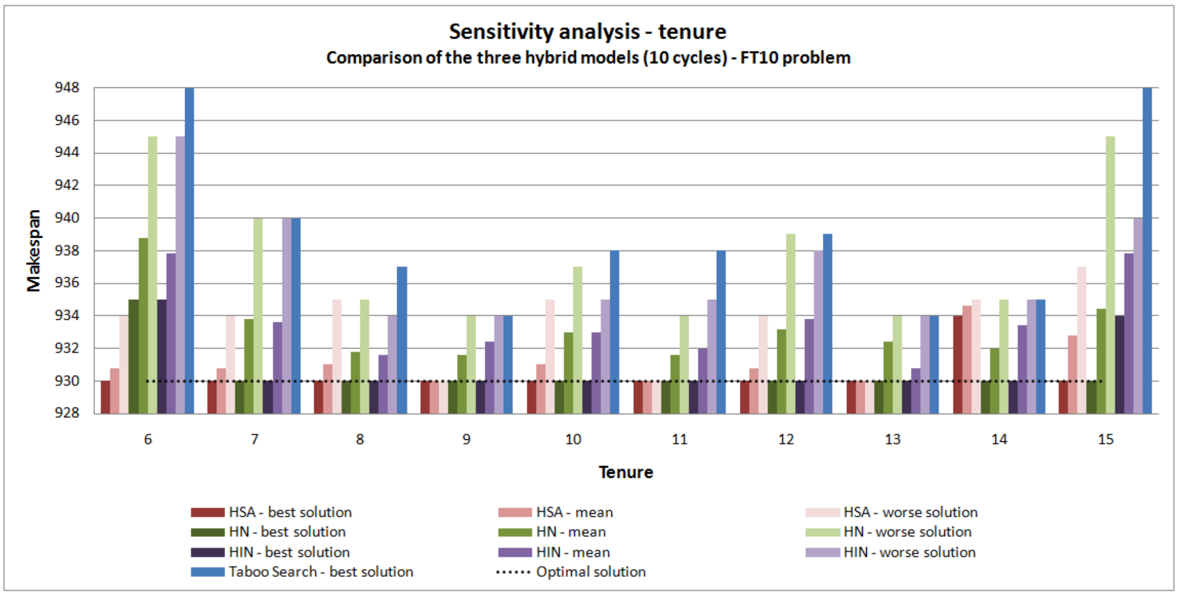

Figure 11. Sensitivity analysis with respect to the tenure of the Hybrid Models (HSA, HIN and HN) for the FT10 problem, with the following fixed parameters: maximum number of iterations without upgrade = 10^4; percentage of particles that suffer mutation = 20%; cycles number = 10. Order of neighborhood generation operators = TS – SPSO. Comparison between the best, the mean and the worse results generated by the three hybrid models and the best result generated by the Taboo Search algorithm.

6.3. Sensitivity analysis with respect to the SPSO mutation probability

The mutation probability sensitivity analysis was applied on the hybrid models fixing the value of the parameter “maximum number of iteration without upgrade” of TS at $1O^3$ for the problem FT20, at

10^4 for the problems FT10, ABZ5 and ABZ6 and at 10^5 for the problems ABZ7, ABZ8 and ABZ9, the value of parameter "tenure" of TS at eight, and setting the value of the parameter "percentage of particles that suffer mutation" of SPSO from 0 to 100%. In this analysis it was identified a small variation of the results generated for each value of the parameter analyzed. As the hybrid models were applied with no more than one iteration of the algorithm SPSO by cycle, this variation must be main attributed to the random nature of the hybrid models more than to the sensitivity to the parameter analyzed.

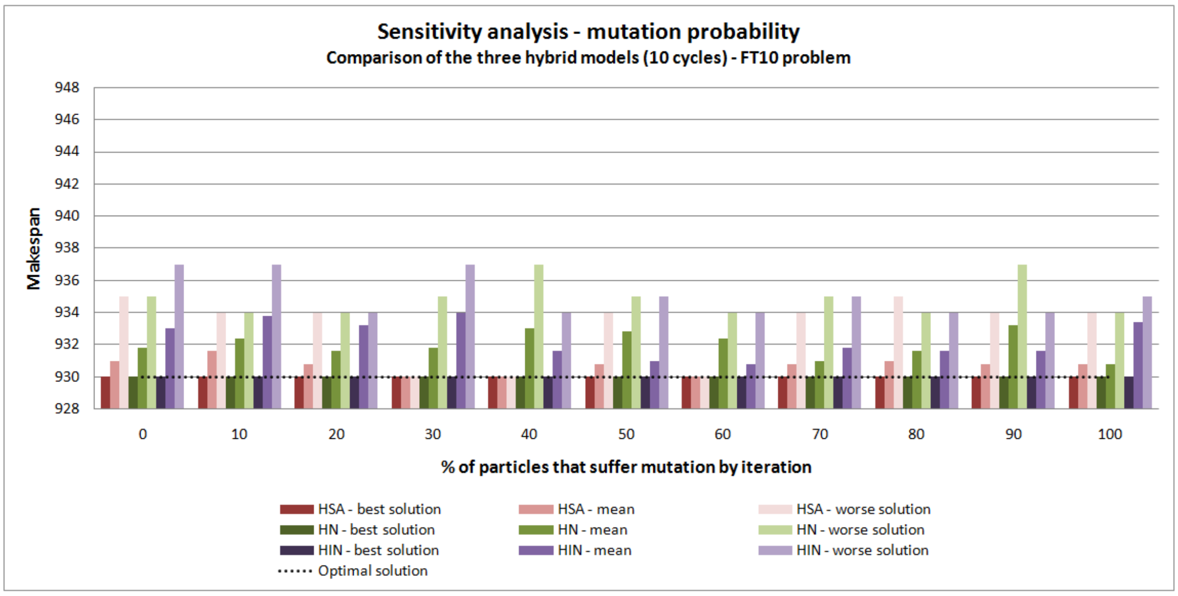

Figure 12. Sensitivity analysis with respect to the "percentage of particles that suffer mutation" of the Hybrid Models (HSA, HIN and HN) for the FT10 problem, with the following fixed parameters: maximum number of iterations without upgrade = 10^4; tenure = 8; cycles number = 10. Order of neighborhood generation operators = TS – SPSO.

6.4. *Convergence analysis*

For the convergence analysis, the three hybrid algorithms were tested fixing the value of parameters "tenure" of TS and "percentage of particles that suffer mutation" of SPSO, respectively, at eight and 20%, the value of the parameter "maximum number of iterations without upgrade" of TS at 10^2 for FT20 problem, at 10^3 for FT10, ABZ5 and ABZ6 problems and at 10^4 for ABZ7, ABZ8 and ABZ9 problems and fixing the maximum number of cycles for FT20, FT10, ABZ6 and ABZ7 problems at 10^2 and at 10^3 for ABZ7, ABZ8 and ABZ9. For the FT20 and the 10×10 problems (FT10, ABZ5 and ABZ6) all the solutions converged to the optimal solutions with a smaller number of cycles. In the case of the 20×15 problems (ABZ7, ABZ8 and ABZ9), despite the hybrid algorithms demonstrated to be able to improve the results found by the single Taboo Search, the computation time has exceeded the maximum time we allowed for our numerical experiments (one week) without finding the optimal solution and without completing the limit defined by the number of cycles. This failure may be connected to the fact that all experiments involved a small set of initial solutions (12).

7. Conclusions

In this work, we presented three possible schemes of hybridization based on Local Search Heuristics, defined as Hybrid Successive Application, Hybrid Neighborhood, and Hybrid Improved Neighborhood. These models were tested applying a Taboo Search (TS) and a Similar Particle Swarm Optimization (SPSO) algorithms as their neighborhood generation operators. Many sensitivity analyzes have been performed on the single TS and SPSO as well as on the hybrid algorithms derived from the three hybrid approaches presented. The results confirmed the following conclusions. The single TS and SPSO algorithms are very sensitive to the tuning of their respective parameters. Also, the results of the single TS algorithm strongly depends on the initial solutions. The TS and SPSO algorithms have complementary nature, however, when they are applied in the hybrid schemes, we constated that the SPSO must be used with no more than one iteration.

In addition, the hybrid algorithms have shown good properties of robustness and accuracy. The results demonstrated the ability of the hybrid algorithms to improve the original techniques significantly, generating better results at acceptable computing times. The convergence to an optimal solution has been obtained in all the experiments, except for the 20×15 problems, where the computational time has exceeded the previously fixed limit (one week). Since all the tests involved a small set of initial solutions, increasing the size of this set may significantly improve the results. This will be the subject of further research.

Another important remark is that, as previously observed, the hybridization is more efficient when the algorithms involved have complementary characteristics. We suggest that the algorithms must be classified into classes according to some general criterions and the hybridization has to be performed by using elements of different classes. This point will also be matter of future work.

Acknowledgements
The author are grateful for the financial support provided by: MCTI, Ministério da Ciência, Tecnologia e Inovação; CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico; and CAPES, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

References

[1] Garey M R, and Johnson D S 1979 *Computers and Intractability: A Guide to the Theory of NP-Completeness* (San Francisco, CA: Freeman and Company)
[2] Zhang C Y, Li PG, Rao YQ and Guan ZL 2008 A very fast TS/SA algorithm for the job shop scheduling problem *Computers & Operations Research* **35** 282–94
[3] Muth J F and Thompson G L 1963 *Industrial Scheduling* (Englewood Cliffs, NF: Prentice-Hall)
[4] Jain A S and Meeran S 1999 Deterministic job-shop scheduling: past, present and future *Journal of Operational Research* **113** 390–434
[5] Johnson S M 1954 Optimal two- and three-stage production schedules with set-up times included *Naval Research Logistics Quarterly* **1** 61–8
[6] Akers S B 1956 A graphical approach to production scheduling problems *Operations Research* **4** 244–5
[7] Jackson J R 1956 An extension of Johnson's result on job lot scheduling *Naval Research Logistics Quarterly,* **3(3)** 201–3
[8] Manne A S 1960 On the job-shop scheduling problem *Operations Research* **8** 219–23
[9] Carlier J and Pinson E 1989 An algorithm for solving the job shop problem *Management Science* **35(29)** 164–76
[10] Brucker P, Jurisch B and Siever B 1994 A branch and bound algorithm for the job-shop scheduling problem *Discrete Applied Mathematics* **49** 105–27
[11] Panwalkar S S and Iskander W 1977 A survey of scheduling rules *Operations Research* **25(1)** 45–61
[12] Adams J, Balas E and Zawack D 1988 The shifting bottleneck procedure for job shop scheduling *Management Science* **34** 391–401
[13] Dauzere-Peres S and Lasserre J -B 1993 A modified shifting bottleneck procedure for job-shop scheduling *International Journal of Production Research* **31** 923–32
[14] Wenqi H and Aihua Y 2004 An improved shifting bottleneck procedure for the job shop scheduling problem *Computers & Operations Research* **31** 2093–110
[15] Erschler J, Roubellat F and Vernhes J P 1976 Finding some essential characteristics of the feasible solutions for a scheduling problem *Operations Research* **24(4)** 774–83

[16] Sabuncuoglu I and Gurgun B 1996 A neural network model for scheduling problems *European Journal of Operational Research* **93** 288–99
[17] Blum C and Sampels M 2004 An ant colony optimization algorithm for shop scheduling problems *Journal of Mathematical Modelling and Algorithms* **3** 285–308
[18] Van Laarhoven P J M, Aarts E H L and Lenstra J K 1992 Job shop scheduling by simulated annealing *Operations Research* **40(1)** 113–25
[19] Azizi N and Zolfaghari S 2004 Adaptive temperature control for Simulated Annealing: a comparative study *Computers & Operations Reasearch* **31** 2439–51
[20] Cheng R, Gen M and Tsujimura Y 1996 A tutorial survey of job-shop scheduling problems using genetic algorithms - I. representation *Computers & Industrial Engineering* **30(4)** 983–97
[21] Cheng R, Gen M and Tsujimura Y 1999 A tutorial survey of job-shop scheduling problems using genetic algorithms: Part II. Hybrid genetic search strategies *Computers & Industrial Engineering* **36(2)** 343–364
[22] Taillard É D 1994 Parallel taboo search techniques for the job shop scheduling problem *ORSA Journal on Computing* **6(2)** 108–117
[23] Nowicki E and Smutnicki C 1996 A fast taboo search algorithm for the job shop problem *Management Science* **42(6)** 797–813
[24] Zhang C Y, Li P, Guan Z and Rao Y 2007 A taboo search algorithm with a new neighborhood structure for the job shop scheduling problem *Computers & Operations Research* **34** 3229–42
[25] Lian Z, Jiao B and Gu X 2006 A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan *Applied Mathematics and Computation* **183** 1008–17
[26] Sha D Y and Hsu C-Y 2006 A hybrid particle swarm optimization for job shop scheduling problem *Computers & Industrial Engineering* **51** 791–808
[27] Lin T-L, Horng S-J, Kao T-W, Chen Y-H, Run R-S, Chen R-J, Lai J-L and Kuo I-H 2010 An efficient job-shop scheduling algorithm based on particle swarm optimization *Expert Systems with Applications* **37** 2629–36
[28] Blazewicz J, Domschke W and Pesch E 1996 The job shop scheduling problem: conventional and new solution techniques *European Journal of Operational Research* **93(1)** 1–33
[29] Pezzella F and Merelli E 2000 A taboo search method guided by shifting bottleneck for the job shop scheduling problem *European Journal of Operational Research* **120** 297–310
[30] Wang L and Zheng D-Z 2001 An effective hybrid optimization strategy for job-shop scheduling problems *Computers & Operations Research* **28** 585–96
[31] Glover F 1989 Taboo search–Part I *ORSA Journal on Computing* **1(3)** 190–206
Glover F 1990 Taboo search–Part II *ORSA Journal on Computing* **2(1)** 4–32
[32] Kennedy J and Eberhart R 1995 Particle Swarm Optimization *Int. Conf. on Neural Networks (Perth, Australia)*
[33] Shi Y and Eberhart R C 1998 Parameter selection in particle swarm optimization *Proceedings of the 7th international conference on evolutionary programming (New York).*
[34] Fisher H and Thompson G L 1963 Probabilistic learning combinations of local job-shop scheduling rules *Industrial Scheduling*, ed J Muth and G Thompson (Englewood Cliffs, New Jersey: Prentice Hall) pp 225–51.
[35] Fraga T B 2010 *Proposition and analysis of hybrid models for the job shop scheduling problem* (*Ph.D. Thesis*) (Nova Friburgo, RJ, BR: Universidade do Estado do Rio de Janeiro - Instituto Politécnico)