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
Universidade do Estado do Rio de Janeiro, Instituto Politecnico. Rua Alberto Rangel, s/n. Vila Nova. 28630-050. Nova Friburgo, RJ – Brasil.

José Eduardo Souza de Cursi, eduardo.souza@insa-rouen.fr
Institut National de Sciences Appliquees de Rouen, Laboratoire de Mécanique de Rouen. Avenue de l'Université – BP 8 – 76801 Saint-Étienne-du-Rouvray – France.

Francisco José da Cunha Pires Soeiro, soeiro@uerj.br
Universidade do Estado do Rio de Janeiro, Centro de Tecnologia e Ciências, Departamento de Engenharia Mecânica. Rua São Francisco Xavier, 524 s/5024-A – Maracanã – 20550013 - Rio de Janeiro, RJ – Brasil.

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 most 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 must 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 biggest 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 heurisitics

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.

```
generate a set of N initial solutions (N ≥ 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
```

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 Laarhooven *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 Smutinicki (1996) without back-propagation and the Similar Particle Swarm optimization proposed by Lian *et al*. (2006), showing what aspects must be considered int 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.

```
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
    if the makespan of the best neighbor is smaller than the makespan of the best solution replace the best
    solution by the best neighbor
    update the taboo list adding the new current solution and removing the oldest one (when taboo list size
    is bigger than t)
end while
return the best solution
```

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 Smutinicki (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).

	Initial Solution	**Tenure**								
		5	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Result 1	1668	1035	971	951	990	949	961	**943**	949	958
Result 2	1695	951	948	945	948	945	**943**	954	951	951
Result 3	1977	994	954	972	940	945	**938**	**938**	954	934
Result 4	1807	1136	980	963	945	<u>**934**</u>	953	956	952	955
Result 5	1749	988	1031	**940**	944	945	946	950	955	946
Result 6	1629	1017	968	1041	949	966	960	951	951	**945**
Result 7	1922	949	958	954	**940**	954	963	948	965	942
Result 8	1783	968	956	953	951	951	**940**	**940**	947	952
Result 9	1701	1104	969	983	**937**	949	948	949	954	961
Result 10	1766	1005	1023	962	944	951	948	952	**939**	957

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).

	Initial Solution	**Maximum number of iterations**				
		10	**100**	**1.000**	**10.000**	**100.000**
Result 1	1668	1128	1056	**1035**	**1035**	**1035**
Result 2	1695	1091	979	**951**	**951**	**951**
Result 3	1977	1350	1088	1021	**994**	**994**
Result 4	1807	1162	**1136**	**1136**	**1136**	**1136**
Result 5	1749	1117	1074	**988**	**988**	**988**
Result 6	1629	1225	1053	**1017**	**1017**	**1017**
Result 7	1922	1044	1017	<u>**949**</u>	**949**	**949**
Result 8	1783	1091	1017	**968**	**968**	**968**
Result 9	1701	1133	**1104**	**1104**	**1104**	**1104**
Result 10	1766	1103	1015	**1005**	**1005**	**1005**

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 which is randomly generated toward the best position visited by the particle (*pbest*) and the best position visited by the swarm (*gbest*). At iteration k, for each particle i ($1 < i < NP$), the velocity v_i and the position x_i can be updated by the following equations:

$$v_i(k+1) = w \times v_i(k) + c_1 r_1 (pbest_i(k) - x_i(k)) + c_2 r_2 (gbest(k) - x_i(k)) \tag{1}$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{2}$$

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:

$$v_i(k+1) \;=\; pbest_i(k) \;\; \Theta \;\; gbest(k) \tag{3}$$

$$v_i(k+1) \;=\; \begin{Bmatrix} v_i(k+1) & if\ mut_i = 0 \\ M\,(v_i(k+1)) & if\ mut_i = 1 \end{Bmatrix} \tag{4}$$

$$x_i(k+1) \;=\; x_i(k) \;\; \Theta \;\; v_i(k+1) \tag{5}$$

$$x_i(k+1) \;=\; \begin{Bmatrix} x_i(k+1) & if\ mut_i = 0 \\ M\,(x_i(k+1)) & if\ mut_i = 1 \end{Bmatrix} \tag{6}$$

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* particles are randomly chosen to suffer a mutation operation:

$$\sum_{i=1}^{NP} mut_i = N \tag{7}$$

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:

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
⋮
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
⋮
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.

```
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
            ⋮
    apply method M on the L solutions generated by the method M – 1
    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 + … + M)
    and store them as the N current solutions
end while
return the best global solution
```

Figure 6. Scheme of hybrid improved neighborhood.

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%).

Number of Iterations	Method	Successive Application				Hybrid Neighborhood		Hybrid improved neighborhood			
		SPSO-TSA 20%	SPSO-TSA 80%	TSA-SPSO 20%	TSA-SPSO 80%	SPSO-TSA 20%	SPSO-TSA 80%	SPSO-TSA 20%	SPSO-TSA 80%	TSA-SPSO 20%	TSA-SPSO 80%
TSA : 10 SPSO : 10	Makespan	1043	962	1027	977	1027	1011	1041	977	1010	982
	NSolEval.	8722	14519	10067	16560	10941	13218	9493	60045	9086	13336
	CPU Time (s)	3	4	3	6	3	3	3	16	2	4
TSA : 100 SPSO : 100	Makespan	983	942	955	958	965	963	977	942	964	958
	NSolEval.	68056	97256	72524	95796	67162	82134	60045	91369	64849	81716
	CPU Time (s)	19	26	21	29	18	22	16	24	17	21
TSA : 1.000 SPSO : 1.000	Makespan	945	938	945	935	945	943	945	934	945	945
	NSolEval.	627445	794550	628900	798314	665734	796400	630332	780278	630800	769781
	CPU Time (s)	162	207	184	207	172	206	163	202	162	199
TSA : 10.000 SPSO : 10.000	Makespan	940	930	937	930	937	930	940	930	930	930
	NSolEval.	5891463	1381868	5867281	2372932	5865063	3021198	5847563	1381868	4908378	2264556
	CPU Time (s)	1513	355	1506	614	1506	776	1506	357	1260	584
TSA : 10.000 SPSO : 1	Makespan	930	934	934	934	930	937	934	937	934	934
	NSolEval.	1731952	4542479	3851663	4463389	3330616	4081269	3705411	3893905	3567571	3576298
	CPU Time (s)	452	1182	1002	1162	872	1069	965	1014	931	934

The schemes presented in Figs. 4, 5 and 6 was tested on FT10 problem with the methods Taboo Search, as presented in section 2, and Similar Particle Swarm Optimization, as presented in section 3, respectively and inversing its order. The results obtained show that when Similar Particle Swarm starts with random positions (solutions), the method searches into the space delimited by the initial positions a local optima and the percentage of particles that suffer mutation (*Pmut*) holds a fundamental role on the convergence process. Small 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 *Pmut* has a big value, the Particle Swarm is not able to improve the results of Taboo Search. When setting big values for *Pmut* 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 *Pmut* parameter. Otherwise when Particle Swarm is applied with no more than one iteration the hybrid method tends to combine the global convergence propriety of Particle Swarm with the local convergence propriety of Taboo Search and all results tends to converge to a global optima. 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 *Nite* (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 *Nite* (large quantity) iterations. Store the solutions found as the current solutions.

Step 4: Apply Similar Particle Swarm with *Nite* (large quantity) iterations on current solutions.

Step 5: Apply Taboo Search with *Nite* (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 (*Nite*) = 10.000; – Similar Particle Swarm: crossover method = C1 with 1 floating point; mutation method = M7; and number of iterations (*Nite*) = 10.000).

Pmut **Tenure**	**TS***	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
5	949	949	940	930	946	930	930	937	940	937	940	945
6	948	930	935	930	930	935	930	934	930	930	940	935
7	940	934	930	930	930	930	930	930	930	930	930	930
8	937	930	930	930	930	935	930	930	930	930	930	930
9	934	934	934	930	930	930	930	930	934	930	930	930
10	938	930	930	930	934	930	930	930	935	930	930	930
11	938	930	930	930	930	930	930	930	930	930	930	930

* 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 a 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 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

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job-shop scheduling. Manegement Science, n° 34, pp. 391-401.

Blazewicz, J., Domschke, W., and Pesch, E., 1996. The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research, n° 93, pp. 1-33.

Brucker, P., Jurisch, B., Sievers, B., 1994. A branch and bound algorithm for job-shop scheduling problem. Discrete Applied Mathematics, n° 49, pp. 105-127.

Carlier, J., Pinson, E., 1989. An algorithm for solving the job shop problem. Management Science, Vol. 35(29), pp.164-176.

Fisher, H., Thompson, G.L., 1963. Probabilistic learning combinations of local job-shop scheduling rules, J.F. Muth, G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 225-251.

Jain, A.S., Meeran, S, 1999a. A state-of-the-art review of job-shop scheduling techniques. <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_1s04/textos/jain98stateart.pdf>

Jain, A.S., Meeran, S, 1999b. Deterministic job-shop scheduling: Past, present and future. European Journal of Operations Research, n° 113, pp. 390-434.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with set-up and times included. Naval Research Logistics Quarterly, n°1, pp. 61-68.

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942-1948.

Lian, Z., Gu, X., Jiao, B., 2006. A similar particle swarm optimization algorithm for permutation flow shop scheduling to minimize makespan. Applied Mathematics and Computation, n° 175, pp. 773-785.

Manne, A.S., 1960. On the job-shop scheduling problem. Operations Research, Vol. 8, pp. 219-223.

Matsuo, H., Suh, C.J., Sullivan, R.S., 1988. A controlled search simulated annealing method for the general job-shop scheduling problem. Working paper #03/04/88, Graduate School of Business, The University of Texas at Austin, Austin, Texas, USA.

Metropolis, N., Rosenbluth, M., Teller, A., and Teller, E., 1953. Equation of state calculations by fast computing machines. Journal Chemical Physics, n° 21, pp. 1087-1092.

Nowicki, E., Smutinicki, C., 1996. A fast taboo search algorithm for the job shop problem. Management Science, n° 42, pp. 797-813.

Panwalkar, S.S., Iskander, W., 1977. A survey of scheduling rules. Operations Research, n° 25, pp. 45-61.

Shi, Y., Eberhart, R.C., 1998. Parameter selection in particle swarm optimization. In V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), Proceedings of the 7th international conference on evolutionary programming, pp. 591-600. New York: Springer.

Van Laarhooven, P.J.M., Aarts, E.H.L. and Lenstra, J.K., 1992. Job shop scheduling by simulated annealing. Operations Research, Vol. 40, n°1, pp. 113-125

9. RESPONSIBILITY NOTICE