GRASP, TABU SEARCH AND PATH RELINKING FOR SOLVING TOTAL EARLINESS/TARDINESS SINGLE MACHINE SCHEDULING PROBLEM WITH DISTINCT DUE WINDOWS AND SEQUENCE-DEPENDENT SETUPS

Marcone Jamilson Freitas Souza
Programa de Pós-Graduação em Engenharia Mineral
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro, 35.400-000, Ouro Preto - MG - Brazil
marcone@iceb.ufop.br

Luiz Satoru Ochi
Instituto de Computação, Universidade Federal Fluminense
Rua Passo da Pátria, 156 - Bloco E - 3^0 Andar, 24.210-240, Niterói - RJ - Brazil
satoru@ic.uff.br

Frederico Augusto de Cezar Almeida Gonçalves
Programa de Pós-Graduação em Modelagem Matemática e Computacional
Centro Federal de Educação Tecnológica de Minas Gerais
Av. Amazonas, 7675, 30.510-000, Belo Horizonte - MG - Brazil
zepfred@gmail.com

Puca Huachi Vaz Penna
Programa de Pós-Graduação em Engenharia Mineral
Universidade Federal de Ouro Preto
Campus Universitário, Morro do Cruzeiro, 35.400-000, Ouro Preto - MG - Brazil
puca@ouropreto.com.br

Abstract. *A single machine scheduling problem with distinct due windows and sequence-dependent setup times to minimize total weighted earliness and tardiness is dealt with. Due to the computational complexity of solving this problem, a heuristic approach based on GRASP, Variable Neighborhood Descent, Tabu Search and Path Relinking, so-called GTSPR, is proposed to solve it. The proposed algorithm explores the solution space by using job order exchanges and reallocations of a job or a block of jobs. For each job sequence generated by the algorithm, an optimal timing algorithm is used to determine the completion time for each job in the sequence. Computational experiments realized with instances found in literature showed that the our proposed algorithm is able to find the optimal solution quickly in small instances (8 to 12 jobs) and, in instances with up to 75 jobs, is able to improve the results from the previously encountered algorithm, with respect to the quality of the final solution and the average gap.*

Keywords: *Single Machine Scheduling, GRASP, Variable Neighborhood Descent, Tabu Search, Path Relinking*

1. INTRODUCTION

Lately, a great number of companies have adopted the just-in-time philosophy for the production of their goods. In this model, it is desirable that the product be finalized exactly on the due date, since extra costs are incurred if this is not achieved. Delayed production signifies contractual fines, not to mention customer dissatisfaction. Likewise, premature production results in extra financial costs due to earlier-than-necessary capital investment and/or stock management.

In relation to delivery, there are three types of problems found in literature: 1) those regarding common due dates; 2) those regarding distinct due dates, and 3) those regarding distinct due windows. In the first type, there is a single date for finishing all the jobs, while in the second, each job is associated with a due date. In the third type, there is a period of time for the conclusion of each job. According to Wan and Yen (2002), this latter situation occurs when there is uncertainty or a time tolerance for the delivery dates and as such, there are no penalties or additional costs, if the jobs are concluded inside the due window.

For common due date production problems involving earliness/tardiness penalties, there are many properties that can be considered in the solution algorithms, which permit an accentuated computational processing reduction when navigating the search space. For example, according to Baker and Scudder (1990), in the optimal sequence, the jobs that finish processing on or before the due date are ordered in a non-increasing ratio between the processing time and the earliness penalty, while the jobs that begin their processing on or after the due date are sequenced in a non-decreasing ratio between the processing time and the tardiness penalty. Because of this, it is said that the sequence is V-shaped. Another interesting property is that in the optimal sequence there is no idle time between two consecutive jobs.

In the case of the distinct due dates, as well as in the case of the distinct due windows, these properties are not necessarily valid, making the search space navigation more complex. For the problems in these types, it is necessary to know if machine idle time is permitted or not. According to França Filho (2007), there exists situations in which idle time is not permitted because it generates costs that are greater than those produced by premature job conclusion. However, there will be situations in which it is advisable to maintain the machine idle, even if conditions exist for the initiation of the job to be processed. Therefore, when there are no restrictions relative to machine idleness, it is necessary to determine the best date for the starting time of each job, or in other words, idle times can be inserted between jobs, in order to produce lower cost solutions.

In relation to the influence of the job preparation time (setup time) for the production line, it has been observed that a majority of the scientific researches consider such time as negligible, or else included in the job processing time. As such, setup times could be considered as being independent of the job sequence. However, there are studies, such as Panwalkar et al. (1973), referenced by Gupta and Smith (2006), which indicate that in many production processes, this setup time is dependent on the production sequence. As per Kim and Bobrowski (1994), to better model practical cases, these times should be considered explicitly whenever they are significantly greater than the processing time. A revision of the works done with production sequencing with setup times can be encountered in Allahverdi et al. (1999, 2008).

Among the production problems, Single Machine Scheduling for Minimizing Earliness and Tardiness Penalties (SMSMETP), is one of the simplest. It has practical importance due to uncountable industrial applications (Allahverdi et al., 1999), but it is not easy to solve it to optimality within acceptable computational time. This is due to the fact that SMSMETP belongs to the NP-hard class (Du and Leung, 1990; Hall et al., 1991). This conjunction of applicability and difficult optimal resolution has motivated researchers to develop efficient algorithms to

resolve it.

In this work, the use of SMSMETP with distinct due windows and sequence-dependent setup time is dealt with. Since several special cases (such as the single machine total weighted tardiness scheduling problem (Du and Leung, 1990) and the single machine total weighted earliness and tardiness scheduling problem (Hall et al., 1991)) of this problem are NP-hard, obviously it is NP-hard, too. Surprisingly, this variation of the SMSMETP has not been given the attention it deserves. The only literature found with these characteristics was written by Gomes Jr. et al. (2007).

To solve this problem, this work proposes a 3-phase algorithm, combining the procedures from GRASP (Feo and Resende, 1995), Variable Neighborhood Descent - VND (Mladenovic and Hansen, 1997), Tabu Search - TS (Glover and Laguna, 1997) and Path Relinking - PR (Glover, 1996). In the first phase, a procedure based on GRASP is utilized to generate an initial solution, with VND being the refining procedure. In the second phase, the solution from GRASP is refined by the Tabu Search procedure. During this phase, a list of elite solutions is formed; these being of high quality and differentiated among themselves. For each best solution generated during Tabu Search, a procedure based on VND is activated to refine the solution. As a post-optimization procedure, a third phase is applied using Path Relinking to all the pairs of solutions from the elite group.

The rest of this work is structured in the following manner: a) in Section 2., related works are presented; b) in Section 3., the characteristics of the studied problem are detailed; c) Section 4. describes the algorithm developed to solve the SMSMETP; d) Section 5. presents and discusses the computational results; and e) Section 6. concludes the research and points out future prospects for the improvement of the proposed algorithm.

2. RELATED WORKS

Li (1997) dealt with the SMSMETP with common due dates with no machine idle time permitted. The author divided the problem into two sub-problems having a simpler structure, so that the inferior limit of the problem is the sum of the inferior limits of these two sub-problems. The inferior limit of each sub-problem was determined by lagrangian relaxation. The branch-and-bound algorithm was presented and used to resolve the instances with up to 50 jobs, doubling the dimension of the instances that could be resolved with exact algorithms up to that date. The author also proposed heuristic procedures based on local search to resolve the instances for more elevated dimensions. Liaw (1999) approached the same problem using a new branch-and-bound algorithm that made use of procedures to determine strong inferior and superior limits. Dominance rules were used to try to eliminate unpromising nodes in the search tree. The performance of the algorithm to solve instances for up to 50 jobs was analyzed.

Feldmann and Biskup (2003) resolved the SMSMETP with common due dates by means of three methods: an evolutive algorithm, Simulated Annealing and a modified version of the Threshold Accepting heuristic; this latter being a variant of Simulated Annealing. The same problem was treated by Hino et al. (2005), who presented methods based on the Tabu Search and Genetic Algorithms metaheuristics, as well as hybridizing them. The authors made use of the properties that the SMSMETP with common due dates has, based on the works of Baker and Scudder (1990) and others, so as to reduce the computational search effort.

Rabadi et al. (2004) focused on the SMSMETP with common due dates and sequence-dependent setup time. The authors presented a branch-and-bound procedure that is able to find the optimal solution, in acceptable time, for instances of up to 25 jobs, which represented an advance because up to that date, exact algorithms for this class of problems were only able to resolve to optimality, instances of up to 8 jobs.

Ying (2008) studied the SMSMETP with common due dates and the setup time for each job included in the processing time, independent of the production sequence. The problem was solved using the Recovering Beam Search (*RBS*) algorithm that is an enhanced version of the Beam Search (*BS*) algorithm. This, in turn, consisted of a branch-and-bound algorithm in which only the w nodes that are most promising for each level of the search tree were retained for future ramification, while the rest of the nodes were permanently pruned. To avoid wrong decisions with respect to the pruning of the nodes that lead to the optimal solution, the *RBS* algorithm utilized a recovering phase that searcheed for better partial solutions that dominated those previously selected.

Chang (1999) approached SMSMETP with distinct due dates using the branch-and-bound algorithm. The performance of this algorithm was analyzed for solving problems of up to 45 jobs. This same problem was treated by Lee and Choi (1995) using the Genetic Algorithm (*GA*). To determine the optimal date for the completion time of each job of the sequence produced by *GA*, a specific algorithm of polynomial complexity that explores the characteristics of the problem was developed.

Gupta and Smith (2006) proposed two methods: one based on GRASP and the other a space-based search heuristic, to solve the sequencing problem of a single machine, considering distinct due dates and sequence-dependent setup times. The objective was to minimize the total tardiness time. The GRASP method was divided in three stages: Construction, Refinement and Path Relinking. The authors' contribution was a new cost function for the construction stage, a new variation of the VND method for the refinement stage, and a Path Relinking stage using different neighborhoods.

Hallah (2007) dealt with the SMSMETP with distinct due dates without idle time between jobs. The author proposed a hybrid method which combined local search heuristics (dispatch rules, descent method, and Simulated Annealing) and an evolutionary algorithm.

Wan and Yen (2002) presented a method based on Tabu Search to solve SMSMETP with distinct due windows. For each job sequence generated by Tabu Search, a procedure of polynomial complexity was also activated to determine the optimal conclusion date for each job in the sequence. This latter procedure was adapted from the proposal made by Lee and Choi (1995), in which due windows were considered instead of due dates. The authors also established some properties of the treated problem and used them to reduce the computational search effort.

Gomes Jr. et al. (2007) developed a mixed integer linear programming model for SMSMETP with due windows and sequence-dependent setup time for solving instances of up to 12 jobs. This model served to compare the results obtained by the heuristic method based on GRASP, Iterated Local Search (ILS) and Variable Neighborhood Descent (VND), also proposed by the authors. For each heuristically generated sequence, a procedure was activated to determine the optimal completion time for each job. The procedure used by the authors, so-called *ADDOIP*, was adapted from Wan and Yen (2002), and it included the setup time in the processing time. This was because the production sequence was already known. The developed algorithm for solving the aforementioned sequencing problem was able to find all the known optimum solutions and presented a maximum gap of 10.89%.

3. PROBLEM DESCRIPTION

The sequencing problem in this research had the following characteristics:

i) A single machine should process a set of n jobs;

ii) For each job i, there was an associated processing time P_i and a due window $[E_i, T_i]$, indicating an initial date (the earliest due date) E_i and a desired ending date (the tardiest

due date) T_i for processing completion. If job i was finalized before E_i, there was a cost α_i per unit of earliness time. In the case where the job was finalized after T_i, there was a cost β_i per unit of tardiness time. The jobs that were concluded inside the due window suffered no additional cost;

iii) The machine could perform only one job at a time and once the process was initiated, it could not be interrupted;

iv) All the jobs were available for processing on date 0;

v) Between two consecutive jobs i and j, a setup time S_{ij} was necessary. Assuming that the machine did not need initial setup time, this time was equal to 0;

vi) Idle time was permitted between two consecutive jobs.

The objective was to minimize the summation of the earliness and tardiness production costs.

4. METHODOLOGY

4.1 Sequence representation

A solution (job sequence) for the problem was represented by a vector v of n elements, where each position $i = 1, 2, \cdots, n$ indicates the order of production of the v_i job. For example, in the sequence $v = \{5, 3, 2, 1, 4, 6\}$, job 5 is the first to be performed and job 6, the last.

4.2 Neighborhood

In order to explore the solution space, the following types of movements were utilized: order exchange in the processing of two jobs in the production sequence, reallocation of a job to another position in the production sequence, and reallocation of a set of k consecutive jobs ($2 \leq k \leq n-1$). These movements defined the N^E, N^R and N^{Or} neighborhoods structures, respectively. The third structure was originally proposed by Or (1976) to explore the solution space for the Traveling Salesman Problem. The reallocations were done forwards and backwards.

Given a set of n jobs, there are $n(n-1)/2$ neighbors in N^E neighborhood. For example, the solution $v' = \{5, 4, 2, 1, 3, 6\}$ is a neighbor of the solution v in the N^E neighborhood, since it is obtained by exchanging job 3 with job 4. In the second neighborhood structure, N^R, there are $(n-1)n$ neighbors. The solution $v' = \{5, 2, 1, 4, 3, 6\}$ is an example of a v neighbor in N^R, since it is obtained from v by the reallocation of job 3 to after job 4 in the v sequence. In the third neighborhood, there are $n^2 - (2k-1)n + k(k-1)$ neighbors of k blocks. The sequence $v' = \{2, 1, 5, 3, 4, 6\}$ is an example of a neighbor of v in the N^{Or} neighborhood with a block of $k = 2$ jobs, since it differs from v by the reallocation of the $(5, 3)$ block.

4.3 Objection function

Solution v was evaluated by the following objective function f, which should be minimized:

$$f(v) = \sum_{i=1}^{n} (\alpha_i \times e_i + \beta_i \times t_i) \tag{1}$$

in which e_i and t_i ($e_i, t_i \geq 0$) indicate, respectively, the earliness and tardiness time of the job i, and α_i and β_i are the respective penalties. To determine the optimal initial processing dates for the jobs in the given sequence and consequently to determine the values for e_i and t_i above, the *ADDOIP* procedure from Gomes Jr. et al. (2007) was utilized.

4.4 Initial solution

A procedure that combined GRASP (Feo and Resende, 1995) with Variable Neighborhood Descent - VND (Mladenovic and Hansen, 1997) was used for generating the initial solution. Figure 1 schematizes this procedure.

Procedure GRASP−VND (*v, γ, GRASPMax, MRDMax*)

```
1  f* ← ∞
2  for i=1 to GRASPMax do
3  |  Choose γ ∈ Γ
4  |  v ← ConstructSolution(γ)
5  |  v' ← VND1(v, MRDMax)
6  |  if (f(v') < f*) then
7  |  |  f* ← f(v')
8  |  |  v* ← v'
9  |  end
10 end
11 v ← VND2(v*, MRDMax)
12 return v
```

Figure 1: GRASP-VND procedure used to generate the initial solution

In the construction phase (line 4 of the Figure 1), a solution was gradually formed, being that at each step, one job was added to the sequence. To chose the job to be added, a candidate list (*CL*) was elaborated using all the job still not sequenced. These were ordered by the initial date of the due window for each job, with the earliest date ($E_{\min}$) being the first on the list and the tardiest date ($E_{\max}$) being the last, like in the Earliest Due Date (*EDD*) heuristic (Baker and Scudder, 1990). From the *CL*, a restricted candidate list (*RCL*) was formed, with the jobs being better classified according to the criteria of the initial due window for the jobs. The size of this restricted list was defined by the parameter $\gamma \in [0, 1]$, chosen in a set Γ, as in Gomes Jr. et al. (2007). Belonging to the *RCL* were all the jobs i whose dates E_i were less than or equal to $E_{min} + \gamma \times (E_{max} - E_{min})$. Afterwards, a job was randomly chosen from this list, the same being added to the partial solution. The construction phase was concluded when all the jobs had been allocated. In Figure 2 is shown the pseudo-code of the GRASP-VND construction phase.

In the refinement phase of the proposed GRASP-VND procedure (Line 5 of Figure 1) a VND_1 procedure was applied to each constructed solution. This refinement heuristic explored the solution space using the N^R and N^E neighborhoods, in this order. Initially, a random descent was performed using reallocation movements. For this, a job and a position were randomly chosen, generating a neighbor sequence. If this new sequence produced a better solution, it was accepted and became the new current solution; if not, another random reallocation was tried. The reallocation phase stopped when there were *MRDMax* consecutive iterations without improvement in the best solution found so far. In this last situation, exchange movements began to be made, also in a random form. If a better solution was produced, this phase was interrupted and returned to the reallocation phase; if not, the exchanges continued. The procedure terminated when there were no improvements in the best solution while using reallocation and exchange movements. Notice that the resulting solution of this search was not necessarily the local optimum in relation to these neighborhoods, keeping in mind that the complete neighborhood was not analyzed for each iteration. Due to this fact, having executed *GRASPMax* iterations, the best solution obtained was submitted to new refinement, line 11 of Figure 1), this

Procedure `ConstructSolution(`γ`)`

$v \leftarrow \emptyset$
Initialize the set *CL* of candidate jobs
while $(CL \neq \emptyset)$ **do**
    $E_{\min} = \min_{i \in CL} E_i$
    $E_{\max} = \max_{i \in CL} E_i$
    $RCL = \{i \in CL \mid E_i \leq E_{\min} + \gamma \times (E_{\max} - E_{\min})\}$
    Choose a job $i \in RCL$ at random
    $v \leftarrow v \cup \{i\}$
    Update *CL*
end
return v

Figure 2: Construction phase of the GRASP-VND procedure

being done by the VND_2 procedure. This procedure consists of exploring the solution space using, besides traditional movements for reallocation and exchange, the movement *Or* described in Subsection 4.2. In the VND_2 procedure, the search was performed in accordance with the following strategy: 1) random descent using reallocation movements; 2) random descent with exchange movements; 3) random descent using *Or* reallocation movements of a block of k jobs $(2 \leq k \leq n-1)$; 4) complete descent using reallocation movements; 5) complete descent with exchange movements; 6) complete descent using reallocation *Or* movements of a block of k jobs $(2 \leq k \leq n-1)$. In the descents with reallocation of a block of k jobs, k initially assumed its lowest value. Achieving some improvement, it went on to the first strategy. If the contrary occurred, k was progressively increased until it obtained its maximum value. In all the random descents, the action was stopped when there were *MRDMax* iterations without improvement. It is relevant to note than the VND_2 procedure returned a local optimum with respect to the N^R, N^E and N^{Or} neighborhoods, since all the neighbors of the current solution with respect to these neighborhoods were analyzed.

4.5 Tabu Search phase

The Tabu Search (TS) procedure (Glover and Laguna, 1997) was used for refining the initial solution. This procedure began its execution using the initial solution generated by the GRASP-VND procedure (Subsection 4.4). For each iteration, the whole neighborhood of this solution was evaluated and the best neighbor was chosen. The best neighbor could not be tabu or if it was tabu it needed to satisfy the condition of being the best solution up to that point, the so-called aspiration criterion. The exploration of the neighborhood was done by using the neighborhood structures N^E and N^R, alternating them at each iteration; that is, in the first iteration, the best neighbor with exchange movements and in the second iteration, the best one with reallocation movements were explored, and so on.

When a neighbor $v' \in N^E(v) \cup N^R(v)$ was chosen, it became a current solution, independent of the fact that it was better or worse than the previous current solution. Then an attribute that characterized this solution was inserted in the Tabu list T, prohibiting its return within a limit of iterations. The objective of verifying if the movement was tabu or not was to avoid cycles of solutions visited recently. Given the exchange between job i and job j, the attribute that was inserted in the Tabu List was the sequence: (job j, job immediately after job j in the modified solution). In the case of the reallocation of job i to the before or after job j position,

the attribute that was inserted was the sub-sequence: (job i, job immediately after job i in the modified solution), if job j was not the last and if it was the last, then (job immediately before i in the modified solution, job i). The Tabu List was limited to $|T|$ elements. When the Tabu List was completed, the oldest element was removed from the list and replaced by the one most recently obtained, i.e. a queue (First In, First Out - FIFO).

To demonstrate how the Tabu list works, consider a sequence where $v = \{5, 3, 2, 1, 4\}$ and an exchange movement of job 5 with job 1. The result is the neighbor sequence $v' = \{1, 3, 2, 5, 4\}$, and the sub-sequence formed by (job 1, job immediately after job 1 in the sequence v') was included in the tabu list T, that is, $T \leftarrow T \cup \{(1, 3)\}$.

Procedure TS (*v*, $|T|$, *divElite, EG, TSMax, MRDMax*)

1 $v^* \leftarrow v$; /* Best solution */
2 *Iter* $\leftarrow 0$; /* Iteration counter */
3 *BestIter* $\leftarrow 0$; /* Iteration in which the best solution was found */
4 $T \leftarrow \emptyset$; /* Tabu List */
5 *EG* $\leftarrow \emptyset$; /* Elite Group */
6 **while** (*Iter* $-$ *BestIter* $\leq$ *TSMax*) **do**
7 *Iter* $\leftarrow$ *Iter* $+ 1$
8 $v' \leftarrow v \oplus m$; /* Let v' the best neighbor of v in $N(v)$ ($N = N^E$ if *Iter* is odd, and $N = N^R$ if *Iter* is even), so that the movement m is not tabu ($m \notin T$) or, if tabu, v' satisfies the aspiration criterion $f(v') < f(v^*)$ */
9 Update the Tabu List T
10 Update the Elite Group *EG*
11 $v \leftarrow v'$
12 **if** $f(v) < f(v^*)$ **then**
13 $v_1 \leftarrow \text{VND}_2(v, \textit{MRDMax})$
14 **if** $(f(v_1) < f(v))$ **then**
15 $v \leftarrow v_1$
16 $T \leftarrow \emptyset$
17 Update the Elite Group *EG*
18 **end**
19 $v^* \leftarrow v$
20 *BestIter* $\leftarrow$ *Iter*
21 **end**
22 **end**
23 $v \leftarrow v^*$
24 **return** v

Figure 3: Refining procedure for a solution using Tabu Search

Every time there was an improvement in the best solution found so far, the VND_2 procedure which was described in Subsection 4.4 was applied. If it was successful, the Tabu list was restarted, since the region of the solution space had changed. The Tabu Search procedure was finalized when *TSmax* iterations were run without improvement in the best solution. The Tabu Search pseudo-code is described in Figure 3. The update of the Elite Group (*EG*), lines 10 and 17 of Figure 3, was elaborated according to Subsection 4.6.

4.6 Path Relinking phase

Path Relinking (PR) has been suggested as an approach to integrate intensification and diversification strategies (Glover, 1996). This approach generates new solutions by exploring trajectories that connect high-quality solutions. Given a pair of solutions, this approach starts with one of them, i.e. the base solution, and arrives at the other, i.e. guide solution, by gradually adding the attributes of the guide solution, one-by-one, to the base solution.

In the proposed algorithm, Path Relinking was used as a tool for post-refinement and was applied after the Tabu Search (TS). To do so, during the exploration of the search space by the TS procedure, PR utilized a set of solutions known as the Elite Group (*EG*). To become part of this group, each solution candidate should satisfy one of the following criteria: 1) be better than the best of the Elite Group solutions; 2) be better than the worst of the Elite Group solutions and be different from them by a determined percentage of attributes, given by *divElite%*. The considered attribute was the pair of consecutive jobs i and j. As such, for example, the sequences $v_1 = \{1, 4, 3, 2, 5, 6\}$ and $v_2 = \{5, 3, 2, 1, 4, 6\}$, have 40% of their attributes equal, which are $(1, 4)$ and $(3, 2)$. The objective of this strategy was to avoid almost-similar solutions from becoming part of the Elite Group. Having the Elite Group already formed, when a new solution enters, the worst evaluation leaves.

The PR procedure was implemented as follows. For each pair of elite solutions, the path in the neighborhood space that leads toward the guide solution was bidirectional. That is, it could be either from the the worst solution to the best (Forward Path Relinking), or from the best solution to the worst (Backward Path Relinking).

Having determined the base and guide solutions, the procedure was executed by gradually inserting an attribute of the guide solution into the base solution. For example, given the guide solution = $\{5, 3, 2, 1, 4, 6\}$, for each step, inserted was a pair of consecutive jobs for this solution, that is, $(5, 3), (3, 2), (2, 1), (1, 4), (4, 6)$, into the base solution. Each pair was inserted in the order in which it appeared in the guide solution. After each insertion, the jobs from the base solution that were substituted left the position that they had occupied and assumed the positions of those that had entered. After which, the base solution was submitted a local search (in this case, a random descent search using order exchange and reallocation moves, as described in Subsection 4.4). During the local search the attributes from the guide solution that had been incorporated into the base solution were fixed. For each step, the base solution received all the non-incorporated attributes of the guide solution, one at a time, generating a new sequence for each specific attribute of the guide solution. At the end of each step, the inserted attribute that produced the best solution after the local search was definitely incorporated into the base solution, becoming fixed. The procedure finalized when the guide solution was achieved; that is, when the base solution comes to have all of the attributes of the guide solution.

Every time the PR procedure generated a better solution than the best one of the Elite Group, the insertion of attributes was interrupted; the Elite Group updated, and the procedure restarted with the application of Path Relinking involving the best solution and the remaining ones of the Elite Group. If the procedure generated a solution better than the worst of the Elite Group and the established diversity criteria were satisfied, then this solution replaced the worst solution of the Elite group. In this last case, the Path Relinking procedure was applied again, involving the new solution of the Elite Group and the remaining ones. The procedure finished after *PRMax* iterations without improvement in the best solution. In this case, it returned the best solution found during the search and the Elite Group.

4.7 GTSPR Algorithm

The proposed algorithm, denominated GTSPR, had three phases and combined the GRASP-VND, Tabu Search (TS), and Path Relinking (PR) procedures previously presented. Its pseudo-code is presented in Figure 4.

Algorithm 4: GTSPR(v, γ, *GRASPMax*, *MRDMax*)

1 $v_0 \leftarrow$ GRASP-VND$(v, \gamma, \textit{GRASPMax}, \textit{MRDMax})$
2 $v_1 \leftarrow$ TS$(v_0, |T|, \textit{TSMax}, \textit{divElite}, \textit{EG}, \textit{MRDMax})$
3 $v \leftarrow$ PR$(\textit{EG}, \textit{divElite}, \textit{MRDMax})$
4 return v

Figure 4: GTSPR Algorithm

5. COMPUTATIONAL RESULTS

The instances used for tests were the same as in Gomes Jr. et al. (2007), which were designed for 8, 9, 10, 11, 12, 15, 20, 25, 30, 40, 50 and 75 jobs. These authors generated the instances with the same parameters used in Wan and Yen (2002), Liaw (1999) and Mazzini and Armentano (2001). In their data, the processing time P_i was uniformly and discretely distributed between 1 and 100. The centers of the due windows were uniformly and integrally distributed in the $[(1-T-\textit{RDD}/2)\times\textit{MS}, (1-T+\textit{RDD}/2)\times\textit{MS}]$ interval, with *MS* being the total processing time of all the jobs, T the tardiness factor, and *RDD* the relative range of the due window. The factors T and *RDD* took values in $\{0.1, 0.2, 0.3, 0.4\}$ and $\{0.8, 1.0, 1.2\}$, respectively. The sizes of the windows were uniformly distributed in $[1, \textit{MS}/n]$. The tardiness costs (β_i) were generated in discrete uniform distribution between 20 and 100, while the earliness weights (α_i) were generated as k times the tardiness costs of the same job, k being an uniform random variable in the $[0, 1]$ interval. The setting was based on the assumption that tardiness is less desirable than earliness. Because T took four different values and *RDD* took three different values, there were 12 instances for each number of jobs. The setup times were generated in discrete uniform distribution in the $[0, 50]$ interval and were symmetrical, that is, $S_{ij} = S_{ji}$.

The proposed algorithm, GTSPR, was coded in C++ programming language and executed in a PC Intel Core 2 Quad (Q6600) 2.40 GHz with 4 GB of RAM running Windows XP Professional. Even though the processor for this equipment had 4 cores, the algorithm was not optimized for multiprocessing. Thirty executions were performed for each instance.

The algorithm parameters were calibrated empirically after preliminary tests. They are presented in Table 1, where n represents the number of jobs. The Γ set of γ parameters for the GRASP construction phase was the same as used by Gomes Jr. et al. (2007), that is, $\Gamma = \{0, 0.02, 0.04, 0.12, 0.14\}$. Only in the instances involving 75 jobs, was *MRDMax* set at $5 \times n$ and *TSMax* at $1 \times n$.

Table 2 presents the results for the computational experiments. In it, the first column indicates the number of jobs and in each set of three columns presents the contribution of each phase of the GTSPR algorithm to minimize the evaluation function f. Column 'GRASP-VND phase' shows the results from the initial solution phase; column 'TS phase', the Tabu Search refinement phase; and column 'PR phase', the Path Relinking post-optimization phase results. The 'time' columns refer to the accumulated average time of CPU processing, in seconds. The two last columns, 'Gomes Jr. time' and 'CPLEX time' give the time in seconds as required by the Gomes Jr. et al. (2007) algorithm and the CPLEX 9.0 solver.

Two performance measures were applied to each phase of the algorithm, given by formulas

Table 1: GTSPR Algorithm Parameters

Parameter	Description	Value
GRASPMax	Maximum number of GRASP iterations	6
MRDMax	Iterations without improvement in Random Descent Methods	$7 \times n$
TSMax	Iterations without Tabu Search improvement	$2 \times n$
PRMax	Iterations without Path Relinking improvement	2
$\|T\|$	Maximum size of Tabu List	$2 \times n$
$\|EG\|$	Maximum size of Elite Group	5
divElite	Diversification index between members of Elite Group	0.4

(2) and (3). In these formulas, for each instance i of the group, $f_i^{GTSPR^*}$ represents the best value found by the proposed algorithm, $f_i^{GJr^*}$ is the best value found by Gomes Jr. et al. (2007), f_i^* is the best value known and $\bar{f}_i^{GTSPR^*}$ is the average value found by GTSPR in thirty executions of the instance i.

$$imp_i^{best} = \frac{f_i^{GJr^*} - f_i^{GTSPR^*}}{f_i^{GJr^*}} \tag{2}$$

$$gap_i^{avg} = \frac{\bar{f}_i^{GTSPR^*} - f_i^*}{f_i^*} \tag{3}$$

The Formula (2) was used to calculate the improvement of the GTSPR algorithm over the Gomes Jr. et al. (2007) algorithm, in relation to the best solution found by each one. In the second performance measure, Formula (3) was applied to determine the gap of the average results in relation to the best known values. In this measure, a small gap indicates an algorithm more robust.

Table 2 shows the improvement results and the average gap for each group of instances with the same number of jobs and for each phase of the GTSPR algorithm. It also presents the computational time required by the Gomes Jr. et al. (2007) algorithm, as well the time spent by the CPLEX solver, 9.0 version. To encounter the optimal solutions was used the mathematical programming model developed by the authors. Notice that the CPLEX time was limited to 3600 seconds. In 25% of the instances involving 11 jobs and 41.7% of those with 12 jobs, CPLEX did not reach the global optimality inside this time limit. With more than 12 jobs CPLEX was not able to reach the global optimality inside this time limit or there was memory allocation problem.

From Table 2 it is possible to affirm that to find the best solutions for instances involving up to 20 jobs, the first phase of the GTSPR algorithm was sufficient. In these instances, the average gap was reduced from 4.93% in the GRASP-VND phase to 1.93% in the TS phase, and to 1.25% in the PR phase. In 75-job instances, the GRASP phase surpassed the best solutions of Gomes Jr. et al. (2007) by 0.63%. In the next phase, the final solutions were improved by 4.40%, and in the last phase, by 5.01%. In the instances of up to 12 jobs, the time demanded to find the optimal solutions was very small in comparison to that of the CPLEX 9.0 solver; i.e. CPLEX required approximately 30 minutes to solve the problems, while GTSPR demanded only 0.15 seconds with a 0.09% gap.

For instances of 25 to 50 jobs, it can be observed that the GRASP-VND phase of the GTSPR algorithm returned worse results than those of the Gomes Jr. et al. (2007) algorithm.

Table 2: Result comparison between each phase of GTSPR, Gomes Jr. et al. (2007) and CPLEX

#	GRASP-VND phase			TS phase			PR phase			Gomes Jr.	CPLEX
jobs	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	time[1]	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	time[1]	$\overline{imp}^{best}$	$\overline{gap}^{avg}$	time[1]	time[2]	time[1]
	%	%	(s)	%	%	(s)	%	%	(s)	(s)	(s)
8	0.00	0.03	0.01	0.00	0.00	0.02	0.00	0.00	0.02	0.04	1.14
9	0.00	0.23	0.02	0.00	0.00	0.03	0.00	0.00	0.04	0.07	24.13
10	0.00	0.38	0.03	0.00	0.00	0.05	0.00	0.00	0.06	0.11	55.29
11	0.00	0.81	0.04	0.00	0.07	0.07	0.00	0.01	0.10	0.20	1519.30
12	0.00	0.62	0.05	0.00	0.20	0.11	0.00	0.09	0.15	0.29	1869.55
15	0.00	3.98	0.13	0.00	1.93	0.30	0.00	1.25	0.46	0.94	-
20	0.00	4.93	0.43	0.00	1.84	1.29	0.00	1.11	2.05	4.35	-
25	-0.84	6.19	1.11	-0.18	2.41	4.17	0.00	1.70	6.62	13.29	-
30	-0.62	7.88	2.74	0.00	3.37	12.07	0.00	2.57	18.66	40.07	-
40	-1.42	10.03	9.96	0.17	4.47	53.42	0.17	3.58	84.16	155.79	-
50	-2.29	12.06	29.62	0.69	5.28	201.09	0.89	4.47	305.28	492.28	-
75	0.63	8.96	300.69	4.40	4.10	1183.58	5.01	2.41	2696.99	1368.08	-
Avg	-0,38	4.68	-	0.42	1.97	-	0.51	1.43	-	-	-

[1] CPU time in seconds in a PC Intel Core 2 Quad (Q6600) 2.40 GHz with 4 GB of RAM

[2] CPU time in seconds in a PC Athlon XP 64 Bits 3000+ (approximately 2 GHz) with 1 GB of RAM

However, the TS phase reduced this difference and improved the results of the aforementioned authors for all these instances, except for the 25-jobs ones. In this latter, GTSPR required the PR phase to achieve better results than those achieved by the Gomes Jr. et al. (2007) algorithm. For the remaining instances of this range, the proposed algorithm was able to surpass their results in up to 5.58%, on the average. In relation to the average gap for these instances, the variability of the final solutions was reduced from 12.06% in the GRASP-VND phase to 5.28% in the TS phase, and to 4.47% in the PR phase, while in Gomes Jr. et al. (2007), the average gap was 10.89%.

With respect to the time demanded to solve the instances, it was not possible to make a direct comparison between the respective algorithms because the equipment used was different. However, it was possible to verify that GTSPR required a small time to find solutions equal or better than those produced by the Gomes Jr. et al. (2007) algorithm. For example, in instances with 20 jobs, the best solutions were found by the GRASP-VND phase of GTSPR in 0.43 seconds, compared to 4.35 seconds for the Gomes Jr. et al. (2007) algorithm. In instances involving 50 jobs, 201.09 seconds was required by the TS phase of GTSPR to find better solutions than the Gomes Jr. et al. (2007) algorithm, which required 492.28 seconds.

6. CONCLUSIONS AND FUTURE WORKS

This paper had its focus on the problem of scheduling a single machine to minimize earliness and tardiness, with distinct due windows and sequence-dependent setup times.

Due to the combinatorial nature of this scheduling problem, an algorithm based on GRASP, Variable Neighborhood Descent (VND), Tabu Search (TS) and with Path Relinking (PR), so-called GTSPR, was proposed to solve it.

The GRASP metaheuristic was used to construct an initial solution. In this phase of the algorithm, the local search was done by the Variable Neighborhood Descent heuristic. In the second phase, the resulting solution of GRASP-VND was refined by the Tabu Search metaheuristic. The third phase consisted in applying Path Relinking as a mechanism of post-optimization.

The proposed algorithm explored the solution space by using job order exchanges and

reallocation of a job or a block of jobs. For each job sequence generated by the algorithm, an optimal timing algorithm was used to determine the completion time for each job in the sequence.

The GTSPR algorithm was applied to instances involving up to 75 jobs and compared with the results of those obtained by an algorithm found in literature and the CPLEX 9.0 solver.

In instances of up to 12 jobs, the GTSPR algorithm very quickly obtained the optimal solutions with a very small gap as compared with the CPLEX 9.0 solver.

In relation to the Gomes Jr. et al. (2007) algorithm, GTSPR improved the results of three groups of instances (those involving 40, 50 and 75 jobs) and equaled the others.

The three phases of GTSPR were important to make the algorithm more robust. Considering all the instances, its TS phase reduced the average gap of GRASP-VND phase from 4.68% to 1.97% and this latter was further reduced in the PR phase to 1.43%, on the average.

For future research, we indicate the following improvements: (1) Apply Path Relinking periodically during the exploration of the solution space and not only as a post-optimization strategy; (2) Apply the Proximate Optimality Principle during the construction phase of the algorithm and (3) Develop proprieties that can be applied to the problem in order to reduce the search effort and, consequently, the processing time of the algorithm.

Acknowledgements

The authors would like to thank Gomes Jr. et al. (2007) for making available their instances and bound values. This work was supported by FAPERJ grant E-26/101.023/2007, Brazil.

References

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T., 1999. A review of scheduling research involving setup considerations. *OMEGA*, vol. 27, pp. 219–239.

Allahverdi, A., Ng, C., Cheng, T. C. E., & Kovalyov, M. Y., 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, vol. 187, pp. 985–1032.

Baker, K. R. & Scudder, G. D., 1990. Sequencing with earliness and tardiness penalties: A review. *Operations Research*, vol. 38, pp. 22–36.

Chang, P. C., 1999. A branch and bound approach for single machine scheduling with earliness and tardiness penalties. *Computers and Mathematics with Applications*, vol. 37, n. 10, pp. 133–144.

Du, J. & Leung, J. Y. T., 1990. Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, vol. 15, pp. 483–495.

Feldmann, M. & Biskup, D., 2003. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers and Industrial Engineering*, vol. 44, pp. 307–323.

Feo, T. A. & Resende, M. G. C., 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, vol. 6, pp. 109–133.

França Filho, M. F., 2007. *GRASP and Tabu Search applied to the scheduling in parallel machines*. Phd thesis, Programa de Engenharia Elétrica e de Computaço, UNICAMP, Campinas, Brazil.

Glover, F., 1996. Tabu search and adaptive memory programming - advances, applications and challenges. In Barr, R. S., Helgason, R. V., & Kennington, J. L., eds, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pp. 1–75. Kluwer Academic Publishers.

Glover, F. & Laguna, M., 1997. *Tabu Search*. Kluwer Academic Publishers.

Gomes Jr., A. C., Carvalho, C. R. V., Munhoz, P. L. A., & Souza, M. J. F., 2007. A hybrid heuristic method for solving the problem of scheduling single machine with earliness and tardiness penalties (in portuguese). In *Proceedings of the XXXIX Brazilian Symposium of Operational Research (XXXIX SBPO)*, pp. 1649–1660, Fortaleza, Brazil.

Gupta, S. R. & Smith, J. S., 2006. Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, vol. 175, pp. 722–739.

Hall, N. G., Kubiak, W., & Sethi, S. P., 1991. Earliness/tardiness scheduling problems - ii: Deviation of completion times about a restrictive common due date. *Operations Research*, vol. 39, pp. 847–856.

Hino, C. M., Ronconi, D. P., & Mendes, A. B., 2005. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, vol. 160, pp. 190–201.

Kim, S. C. & Bobrowski, P. M., 1994. Impact of sequence dependent setup time on job shop scheduling performance. *International Journal of Production Research*, vol. 32, n. 7, pp. 1503–1520.

Lee, C. Y. & Choi, J. Y., 1995. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Research*, vol. 22, pp. 857–869.

Li, G., 1997. Single machine earliness and tardiness scheduling. *European Journal of Operational Research*, vol. 96, pp. 546–558.

Liaw, C. F., 1999. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers and Operations Research*, vol. 26, pp. 679–693.

Mazzini, R. & Armentano, V. A., 2001. A heuristic for single machine scheduling with early and tardy costs. *European Journal of Operational Research*, vol. 128, pp. 129–146.

Mladenovic, N. & Hansen, P., 1997. Variable neighborhood search. *Computers and Operations Research*, vol. 24, pp. 1097–1100.

Or, I., 1976. *Traveling salesman-type combinational problems and their relation to the logistics of blood banking*. Phd thesis, Northwestern University, USA.

Panwalkar, S. S., Dudek, R. A., & Smith, M. L., 1973. Sequencing research and the industrial scheduling problem. In Elmaghraby, S. E., ed, *Symposium on the Theory of Scheduling and its Applications*, pp. 29–38.

Rabadi, G., Mollaghasemi, M., & Anagnostopoulos, G. C., 2004. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers and Operations Research*, vol. 31, pp. 1727–1751.

Wan, G. & Yen, B. P. C., 2002. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, vol. 142, pp. 271–281.

Ying, K. C., 2008. Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm. *Computers and Industrial Engineering*, vol. to appear (doi:10.1016/j.cie.2008.01.008).