Minimizing earliness and tardiness penalties on a single machine scheduling problem with distinct due windows and sequence-dependent setup times

Marcone J. F. Souza[a], Luiz S. Ochi[b] and Nelson Maculan[c]

[a]*Departamento de Computação, Programa de Pós-Graduação em Engenharia Mineral, Universidade Federal de Ouro Preto, 35.400-000 Ouro Preto, Brasil,* `marcone@iceb.ufop.br`
[b]*Departamento de Computação, Universidade Federal Fluminense, Rua Passo da Pátria 156, Bloco E, 3^0 Andar, 24.210-240 Niterói, Brasil,* `satoru@ic.uff.br`
[c]*Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, P. O. Box 68511, Bloco H, 21.941-972 Rio de Janeiro, Brasil,* `maculan@cos.ufrj.br`

Keywords: Single Machine Scheduling, GRASP, VND, Tabu Search, Path Relinking

Abstract. This paper deals with the problem of scheduling a single machine with distinct due windows and sequence-dependent setup times to minimize earliness and tardiness penalties. Due to its computational complexity, 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 procedure is used to determine the completion time for each job in the sequence. Computational results showed that the proposed algorithm obtained better solutions than those encountered in literature, both with respect to the quality and the average gap.

1 Introduction

This work dealt with the problem of scheduling a single machine to minimize earliness and tardiness where jobs had distinct due windows and there were sequence-dependent setup times. Since several special cases of this problem are NP-hard (such as the single machine total weighted tardiness scheduling problem (Du and Leung, 1990)), it is NP-hard, too.

Distinct due windows mean that there is a period of time for the conclusion of each job. According to Wan and Yen (2002), this 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 scheduling problems of this type, when there are no restrictions relative to machine idleness, it is necessary to determine the best date for the starting time of each job. In other words, idle times can be inserted between jobs, in order to produce lower cost solutions, even if conditions exist for the initiation of the job to be processed.

In relation to the influence of the job preparation time (setup time), 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, many studies report that setup times are dependent on processing sequence (Gupta and Smith, 2006). A comprehensive revision of the researches involving production sequencing with setup times can be found in Allahverdi et al. (2008).

Surprisingly, this generalization of the single machine scheduling problem has not been given the attention it deserves. The only literature found with these characteristics was written by Gomes Jr. et al. (2007). The last authors developed a mixed integer linear programming model for solving the aforementioned sequencing problem, which is able to solve 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 when the procedure was activated. The developed algorithm was applied for solving instances of up to 75 jobs and it was able to find all the known optimum solutions and presented a maximum gap of 10.89%.

Due to the computational complexity of solving combinatorial problems to optimality, such as this scheduling problem, heuristics approaches have been addressed by many researchers. In this sense, this work proposes a 3-phase algorithm, combining GRASP (Feo and Resende, 1995), Variable Neighborhood Descent (Mladenovic and Hansen, 1997), Tabu Search and Path Relinking (Glover and Laguna, 1997) to solve it.

The rest of this work is structured as follows: a) in Section 2, the characteristics of the studied problem are detailed; b) Section 3 describes the algorithm developed to solve it; c) Section 4 presents and discusses the computational results; and d) Section 5 concludes the research.

2 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.

3 Methodology

3.1 Sequence representation, Neighborhood and Objective function

A solution (job sequence) for the problem was represented by a list v of n jobs, where each position i indicated the production order of the job v_i. In order to explore the solution space, the following movements were utilized: order exchange in the processing of two jobs, reallocation of a job to another position in the 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, respectively. The reallocations were done forwards and backwards.

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.

3.2 GTSPR Algorithm

The proposed algorithm, so-called GTSPR, has three phases and combines the characteristics of the GRASP, Variable Neighborhood Descent (VND), Tabu Search (TS), and Path Relinking (PR) procedures. Figure 1 schematizes it. Its phases are explained in the next subsections.

Algorithm 1: GTSPR(v, γ, *GRASPMax*, *MRDMax*)

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

Figure 1: GTSPR Algorithm

3.3 GRASP-VND phase

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.

In the construction phase, a solution was gradually formed, being that at each step, one job was added to the sequence. To choose 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. 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 the refinement phase, a VND_1 procedure was applied to each constructed solution. This refinement heuristic explored the solution space performing random descents in the N^R and N^E neighborhoods, in this order. The search was stopped when there were *MRDMax* iterations without improvement. 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 of construction and refinement, the best solution obtained was submitted to new refinement, this being done by the VND_2 procedure. This last procedure consists of exploring the solution space using, besides traditional movements for reallocation and exchange, the *Or* movement. 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* 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 its maximum value is attained.

3.4 Tabu Search phase

The Tabu Search (TS) procedure (Glover and Laguna, 1997) began its execution using the initial solution generated by the GRASP-VND procedure (Subsection 3.3). For each iteration, the whole neighborhood of the current 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.

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 $|TL|$ elements and it worked as a queue (First In, First Out - FIFO).

Every time there was an improvement in the best solution found so far, the VND_2 procedure which was described in Subsection 3.3 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.

3.5 Path Relinking phase

Path Relinking (Glover and Laguna, 1997) was used as post-refinement strategy. 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. The objective of this strategy was to avoid almost-similar solutions from becoming part of the Elite Group.

The PR procedure was implemented as follows. For each pair of elite solutions, the path in the neighborhood space that led toward the guide solution was bidirectional. That is, it could be either from the the worst solution to the best, or from the best solution to the worst. After each attribute insertion, the base solution was submitted a local search (in this case, a random descent search using order exchange and reallocation moves). 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.

4 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. For each group with the same number of jobs, there were 12 instances with different settings. These authors utilized standard approaches to generate single machine scheduling problem instances.

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. The algorithm parameters were calibrated empirically and the values used were:

GRASPMax = 6, *MRDMax* = $7 \times n$, *TSMax* = $2 \times n$, $|TL| = 2 \times n$, $|EG| = 5$, *divElite* = 0.4, 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 75-job instances, was *MRDMax* set at $5 \times n$ and *TSMax* at $1 \times n$.

Two performance measures were applied to each phase of the algorithm, given by the formulas $imp_i^{best} = \frac{f_i^{GJr^*} - f_i^{GTSPR^*}}{f_i^{GJr^*}}$ and $gap_i^{avg} = \frac{\bar{f}_i^{GTSPR^*} - f_i^*}{f_i^*}$. In these formulas, for each instance i of the group, $f_i^{GTSPR^*}$ represents the best value found by the GTSPR algorithm, $f_i^{GJr^*}$ is the best value found by the Gomes Jr. et al. (2007) algorithm, f_i^* is the best value known and $\bar{f}_i^{GTSPR^*}$ is the average value found by GTSPR in thirty executions. The first formula calculates the improvement of the GTSPR algorithm over the Gomes Jr. et al. (2007) algorithm, in relation to the best solution found by each one, while the second determines the gap of the average results in relation to the best known values.

Table 1 shows the improvement results and the average gap for each group of instances and for each phase of the GTSPR algorithm. The 'time' columns refer to the accumulated average time of CPU processing, in seconds. It also presents the computational time required by the Gomes Jr. et al. (2007) algorithm, as well as 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. For larger instances notice that the CPLEX was not able to reach the global optimality within 3600 seconds or there was memory allocation problem.

Table 1: 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

From Table 1 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 solver; i.e. in 12-job instances 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. However, the TS phase reduced this difference and improved the results for all these instances. For 25-jobs instances, GTSPR required the PR phase to achieve better results than those achieved by the aforementioned authors. In 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%.

Considering all the instances, the 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%.

5 Conclusions

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, VND, Tabu Search and Path Relinking, so-called GTSPR, was proposed to solve it.

The GTSPR algorithm explored 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 procedure was used to determine the completion time for each job in the sequence. It 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. 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.

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., Ng, C., Cheng, T. C. E. and Kovalyov, M. Y.: 2008, A survey of scheduling problems with setup times or costs, *EJOR* **187**, 985–1032.

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

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

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

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

Gupta, S. R. and Smith, J. S.: 2006, Algorithms for single machine total tardiness scheduling with sequence dependent setups, *EJOR* **175**, 722–739.

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

Wan, G. and Yen, B. P. C.: 2002, Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties, *EJOR* **142**, 271–281.