Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 66 (2018) 47–54

www.elsevier.com/locate/endm

Algorithms based on VNS for solving the Single Machine Scheduling Problem with Earliness and Tardiness Penalties

B. F. Rosa [a,1] M. J. F. Souza [b,2] S. R. de Souza [a,3]

[a] *Federal Center of Technological Education of Minas Gerais (CEFET-MG), Belo Horizonte, MG, 30510-000, Brazil*

[b] *Federal University of Ouro Preto (UFOP), Ouro Preto, MG, 35400-000, Brazil*

Abstract

This work implements and compares four algorithms based on Variable Neighborhood Search (VNS), named RVNS, GVNS$_f$, GVNS$_r$ and GVNS$_{rf}$, for solving the Single Machine Scheduling Problem with Earliness and Tardiness Penalties (SMSPETP). Computational experiments showed that the algorithm GVNS$_f$ obtained better-quality solutions compared with the other algorithms, including an algorithm found in the literature. The algorithms GVNS$_r$ and GVNS$_{rf}$ obtained solutions close to the GVNS$_f$, and outperformed the algorithm of the literature, both with respect to the quality of the solutions and the computational times.

Keywords: Single Machine Scheduling, Sequence-Dependent Setup Times, VNS.

[1] Email: brunorosa@div.cefetmg.br
[2] Email: marcone@iceb.ufop.br
[3] Email: sergio@dppg.cefetmg.br

https://doi.org/10.1016/j.endm.2018.03.007

1 Introduction

This work addresses the Single Machine Scheduling Problem with Earliness and Tardiness Penalties (SMSPETP). In the addressed problem there is a time window for each job. Machine idle is allowed, even if there is a job to be performed. Besides this, it is necessary to setup the machine between two consecutive jobs and the setup times are sequence-dependent. According to the notation employed by [5], the SMSPET is represented by $1/s_{jk}/\sum_{j=1}^{n} w'_j E_j + \sum_{j=1}^{n} w''_j T_j$.

The problem to be dealt consists of sequencing and determining the time in which jobs must be performed in order to minimize the weighted sum of earliness and tardiness penalties in the execution of the jobs. There are a variety of applications of this problem in JIT manufacturing, semi-conductor manufacturing, chemical processing, PERT/CPM scheduling, and video on demand services, among others [3]. Since it is NP-hard problem [1], it is usually solved by heuristic methods, among them [8,6,7].

In this work, algorithms based on Variable Neighborhood Search – VNS [2] for solving the SMSPETP are proposed and tested. The determination of the optimal starting date for the execution of each job belonging to the sequence generated by these algorithms is made by the Idle Time Insertion Algorithm (ITIA) of [7]. Computational experiments are realized in order to compare the proposed algorithms with the best algorithm of [7].

The remaining of this work is organized as follows. In Section 2 the SMSPETP is described in details. The proposed algorithms are presented in Section 3, while in Section 4 the results are showed and analyzed. In Section 5 the work is concluded.

2 Characteristics of the addressed problem

The SMSPETP has the following characteristics: (i) a single machine must process a set I of n jobs; (ii) for each job $x \in I$, there is a processing time P_x and a time window $[E_x, T_x]$ in which the job x should preferably be completed (E_x indicates the earliest due date and T_x is the tardiest due date); (iii) if job x is completed before E_x, then there is a cost of α_x per unit of earliness time; (iv) if job x is completed after T_x, there is a cost of β_x per unit of tardiness time; (v) the jobs completed within their time windows do not incur costs; (vi) the machine can perform only one job at a time and once the process is initiated, it cannot be interrupted; (vii) all jobs are available for processing starting from time 0; $(viii)$ between two consecutive jobs x and $y \in I$, a setup

time of S_{xy} is required (it is assumed that the time for setting up the machine in order to process the first job in the sequence is equal to 0); (ix) idle time between the execution of two consecutive jobs is allowed.

The objective is to determine a job sequence X of I and the associated starting dates for executing the jobs that minimize the weighted sum of the earliness and tardiness for each job, that is, to minimize the value of:

$$F(X) = \sum_{x \in I} \big(\alpha_x \cdot \max(0,\ E_x - C_x) + \beta_x \cdot \max(0, C_x - T_x)\big), \qquad (1)$$

where C_x represents the time of completion of job $x \in I$ in an optimal scheduling of the sequence X.

The resolution of this problem involves two actions, which must be performed in sequence: (i) determine the job processing sequence, i.e., the order in which the jobs must be executed; and (ii) determine the starting date for each job, such that the weighted sum of earliness and tardiness for all jobs is minimized.

3 Algorithms based on VNS applied to SMSPETP

Four algorithms based on Variable Neighborhood Search (VNS) metaheuristic [2] were tested for solving SMSPETP: (i) Reduced Variable Neighborhood Search – RVNS; (ii) General Variable Neighborhood Search – GVNS$_f$; (iii) GVNS$_r$; and (iv) GVNS$_{rf}$. All of them use the classic VNS algorithm, which is presented by the Algorithm 1.

Algorithm 1 VNS$(I, n, F(.), N_1(.), N_2(.), N_3(.), \text{VNSmax})$

```
1: X ← InitialSolution(I, n);
2: Iter ← 0;  f* ← F(X);
3: while Iter < VNSmax do
4:     k ← 1;  Iter ← Iter + 1;
5:     while k ≤ 3 do
6:         Randomly generates a neighbor X' ∈ N_k(X);
7:         X'' ← LocalSearch(X', F(.));
8:         if F(X'') < F(X) then
9:             X ← X'';  f* ← F(X);  k ← 1;  Iter ← 0;
10:        else
11:            k ← k + 1;
12: return f*;
```

In the Algorithm 1, VNSmax indicates the maximum number of iterations without improvement in the best solution found. The details of the procedures InitialSolution, F and LocalSearch, as well as those for the neighborhoods N_1, N_2 and N_3 are presented in the next subsections. The first tested algorithm, named RVNS, consists in excluding the procedure LocalSearch, corresponding to line 7 of Algorithm 1. The second tested algorithm, called GVNS$_f$, uses VND$_f$ as the local search procedure; the third tested algorithm, named GVNS$_r$, uses, as the local search, the procedure VND$_r$. The last algorithm, named GVNS$_{rf}$, consists in applying the local search VND$_f$ after GVNS$_r$. The local search procedures VND$_r$ and VND$_f$ are described in Subsection 3.4.

3.1 Initial solution construction

A solution X for the SMSPETP with n jobs is represented by a vector of n positions. Each index $i = 1, 2, \ldots, n$ indicates the job to be executed at position i of X. For example, in the sequence $X = (5, 1, 2, 6, 4, 3)$, job 5 is the first to be executed, and job 3 is the last.

An initial solution is constructed by an adaptation of the Earliest Due Date (EDD) heuristic [5]. The procedure initializes with a null sequence. For each iteration, the job not yet sequenced that has the completion window with the smallest starting date is inserted at the end of the current subsequence. Ties are broken by choosing the job that has the completion window with the smallest ending date. If the tie persists, a job is chosen randomly. The construction procedure is stopped when all the jobs are sequenced.

3.2 Neighborhoods of a solution

Such as in [7], three types of movements are used to explore the solution space of the problem: (i) pairwise interchange; (ii) one job reallocation; and (iii) k-jobs subsequence reallocation, with $1 \leq k < n$. These movements define the neighborhoods N_1, N_2 and N_3, respectively.

3.3 Evaluation of a solution

A solution X is evaluated by the function F described in Eq. (1). The optimal date of each job of X is obtained by applying the algorithm ITIA of [7].

3.4 Local search methods

Two local search procedures, based on the Variable Neighborhood Descent – VND method [4], are used: (i) VND$_f$; and (ii) VND$_r$. Both use movements

based on neighborhoods N_1, N_2, and N_3 in order to explore the solution space.

The local search VND_f uses the following deterministic sequence of local searches: (LS1) First Improvement using the neighborhood N_1; and (LS2) First Improvement using the neighborhood N_3. When there are not neighbor solutions X' that improve the current solution X during the local search LS1, then the neighborhood is changed to the next one (N_3), that is, the local search LS2 is applied. LS2 initializes with 1-job reallocation movements. If it is not possible to improve the current solution with these movements, then the size of the subsequence reallocations is increased in one unit, that is, 2-jobs subsequence reallocations are used. The maximum size of the subsequence reallocation is $n-1$. Whenever a solution X' that improves the current one X is found, then X' becomes the new current solution and the search returns to the local search LS1. The method VND_f finishes when the local search LS2 does not improve the solution from the local search LS1. In this case, the final solution is a local optimum in relation to the neighborhoods N_1 and N_3. It is important to note that the neighborhood N_2 uses 1-job reallocation movements. Thus, as N_2 is contained in N_3, then the final solution of the method VND_f is also a local optimum in relation to the neighborhood N_2.

The local search VND_r uses the following sequence of local searches: (LS3) Random Descent using the neighborhood N_1; (LS4) Random Descent using the neighborhood N_2; and (LS5) Random Descent using the neighborhood N_3. When a solution X' that improves the current one X is found, then X' becomes the new current solution and the search returns to the local search LS3. After VNDmax iterations without improvement, the next local search is triggered. The method VND_r is finished when the current solution is not improved in these three local searches.

4 Computational experiments

The algorithms RVNS, GVNS_f, GVNS_r and GVNS_{rf}, as well as the local search methods VND_f and VND_r were implemented in `C++`, using the compiler `g++`, version 4.8.5, for their execution. A set of instances of [7], involving 10, 20, 30, 40, 50 and 75 jobs, were used in order to test these algorithms. As each set contains 16 instances, then 96 instances were used.

The experiments were performed on a computer with Intel® Xeon(R) CPU E5620 @ 2.40GHz × 16, with 48 GB of RAM and CentOS Linux 7 operational system. Although the processor of this computer has more than one core, the algorithms are not optimized for multi-processing.

In order to compare the results produced by the algorithms, the relative

average deviation, calculated by the expression $\Delta_i^{\text{avg}} = \left(\overline{f}_i^{A} - f_i^{\star}\right)/f_i^{\star}$ was used. In this expression, $f_i^{\star}$ represents the best value to instance i found by the best algorithm of [7] and $\overline{f}_i^{A}$ represents the average value produced by algorithm A. The smaller the value of Δ_i^{avg} is, the better the algorithm will be.

Initially, the set of 50-jobs instances was used to calibrate the parameter VNDmax of the local search VND_r. The values $5n$, $7n$, $9n$, $11n$ and $13n$ were experimented, where n represents the number of jobs. Each instance was solved 30 times by this procedure. The results are reported in Table 1. In this table, the column $\overline{\Delta^{\text{avg}}}$ shows the average values of the gaps (Δ_i^{avg}) in each instance i, while the column $\bar{t}$ shows the average required time (in seconds) for solving this set of instances.

Table 1
Influence of the values VNDmax in the procedure VND_r for 50-jobs instance set

$3n$		$5n$		$7n$		$9n$		$11n$		$13n$	
$\overline{\Delta^{\text{avg}}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{\text{avg}}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{\text{avg}}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{\text{avg}}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{\text{avg}}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{\text{avg}}}$ (%)	$\bar{t}$ (s)
15.18	0.03	11.62	0.04	9.62	0.05	8.32	0.06	7.62	0.08	7.19	0.09

According to Table 1, as the value of VNDmax is increased, the better is the average value of the solutions and the greater is the average time demanded. The difference between the value of $\overline{\Delta^{\text{avg}}}$ with VNDmax $= 11n$ and the value of $\overline{\Delta^{\text{avg}}}$ with VNDmax $= 13n$ is lower than 0.5%. In addition, the time demanded with VNDmax $= 13n$ is longer than the time required with VNDmax $= 11n$. For these reasons and as the local search is applied many times, the value of VNDmax was set at $11n$ in the other experiments.

In order to calibrate the parameter VNSmax of the proposed algorithms, the set of 30-jobs instances was used. The following values are tried: n, $2n$, $3n$, $4n$ and $5n$, where n represents the number of jobs. Each instance was solved 30 times by the algorithm GVNS_f and the results are summarized in Table 2. In this Table, the columns "$\overline{\Delta^{\text{avg}}}$" and "$\bar{t}$" have the same meaning of Table 1.

According to Table 2, as the value of VNSmax is increased, the better is the average value of the solutions and the greater is the average time demanded. When VNSmax $> 2n$, the difference between the values of $\overline{\Delta^{\text{avg}}}$ is less than 0.05%, but the time required grows a lot. Thus, the value of VNSmax was fixed at $2n$ in the algorithms RVNS, GVNS_f, GVNS_r and GVNS_{rf}.

The results obtained by the proposed algorithms, as well as the best algorithm of [7], are reported in Table 3. In this table, the first column shows the

Table 2
Influence of the values of VNSmax in algorithm GVNS_f for 30-jobs instance set

$1n$		$2n$		$3n$		$4n$		$5n$	
$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)
0.10	3.91	0.05	7.09	0.04	10.07	0.04	13.35	0.03	16.24

number of jobs for each set of instances. The columns "$\overline{\Delta^{avg}}$" and "$\bar{t}$" have the same meaning of the Table 1.

Table 3
Results obtained by the algorithms RVNS, GVNS_f, GVNS_r and GVNS_{rf}

	[7]		RVNS		GVNS_f		GVNS_r		GVNS_{rf}	
n	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)	$\overline{\Delta^{avg}}$ (%)	$\bar{t}$ (s)
10	0.00	0.03	8.64	0.00	0.00	0.02	0.00	0.02	0.00	0.02
20	0.01	0.42	14.26	0.00	0.13	0.68	0.04	0.34	0.04	0.34
30	0.17	2.31	16.74	0.00	0.05	7.20	0.14	1.92	0.14	1.93
40	0.25	8.61	18.21	0.01	0.03	40.91	0.15	6.85	0.15	6.88
50	0.52	24.69	18.42	0.02	0.04	159.07	0.27	21.42	0.27	21.55
75	0.98	172.61	22.18	0.07	-0.18	1972.85	0.56	155.71	0.54	162.65

According to Table 3, GVNS_f was the algorithm with the best performance in relation to the quality of the average solutions. Except in 20-jobs instances, it outperformed all the other algorithms, including the best one of [7]. It is also important to note that it found new best solutions in 75-jobs instances. However, this algorithm required a very high processing time. On the other hand, the algorithms GVNS_r and GVNS_{rf} have a very similar performance and both also outperformed the best algorithm of [7]. In fact, except in 20-jobs instances, both presented better solutions and required less processing time.

5 Conclusions

This work deals with the single machine scheduling problem with distinct time windows and sequence-dependent setup times (SMSPETP). Four algorithms based on VNS metaheuristic were proposed: RVNS, GVNS_f, GVNS_r and GVNS_{rf}.

Computational results showed that the algorithm GVNS_f outperformed the others algorithms in relation to the solution quality, including the best algorithm of [7]. However, this algorithm demanded a high processing time. On the other hand, the algorithms GVNS_r and GVNS_{rf} have proved to be a good alternative for solving SMSPETP, since they obtained solutions close to the GVNS_f, outperformed the best algorithm of [7], and required a much shorter processing time.

Acknowledgement

The authors thank FAPEMIG, CNPq, CEFET-MG and UFOP for supporting the development of this research.

References

[1] Allahverdi, A., J. N. Gupta and T. Aldowaisan, *A review of scheduling research involving setup considerations*, Omega **27** (1999), pp. 219–239.

[2] Hansen, P., N. Mladenović and J. A. M. Pérez, *Variable neighbourhood search: methods and applications*, 4OR **6** (2008), pp. 319–360.

[3] Janiak, A., W. A. Janiak, T. Krysiak and T. Kwiatkowski, *A survey on scheduling problems with due windows*, European Journal of Operational Research **242** (2015), pp. 347–357.

[4] Mladenović, N. and P. Hansen, *Variable neighborhood search*, Computers & Operations Research **24** (1997), pp. 1097–1100.

[5] Pinedo, M., "Scheduling: Theory, Algorithms, and Systems," Springer New York, 2012, 4th edition.

[6] Ribeiro, F. F., M. J. F. Souza and S. R. De Souza, *An adaptive genetic algorithm to the single machine scheduling problem with earliness and tardiness penalties*, in: *Advances in Artificial Intelligence - SBIA 2010, Lecture Notes in Computer Science 6404*, Springer Berlin Heidelberg, 2010, pp. 203–212.

[7] Rosa, B. F., M. J. F. Souza, S. R. de Souza, M. F. de França Filho, Z. Ales and P. Y. P. Michelon, *Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties*, Computers & Operations Research **81** (2017), pp. 203–215.

[8] Souza, M., L. Ochi and N. Maculan Filho, *Minimizing earliness and tardiness penalties on a single machine scheduling problem with distinct due windows and sequence-dependent setup times*, in: *Proc. of the ALIO/EURO 2008 Conference*, Buenos Aires, 2008.