Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 281 (2011) 5–19

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Multi-objective Variable Neighborhood Search Algorithms for a Single Machine Scheduling Problem with Distinct due Windows

José Elias Claudio Arroyo[1,2] Rafael dos Santos Ottoni[3]
Alcione de Paiva Oliveira[2]

Departamento de Informática
Universidade Federal de Viçosa
Viçosa - MG - Brazil

Abstract

In this paper, we compare three multi-objective algorithms based on Variable Neighborhood Search (VNS) heuristic. The algorithms are applied to solve the single machine scheduling problem with sequence dependent setup times and distinct due windows. In this problem, we consider minimizing the total weighted earliness/tardiness and the total flowtime criteria. We introduce two intensification procedures to improve a multi-objective VNS (MOVNS) algorithm proposed in the literature. The performance of the algorithms is tested on a set of medium and larger instances of the problem. The computational results show that the proposed algorithms outperform the original MOVNS algorithm in terms of solution quality. A statistical analysis is conducted in order to analyze the performance of the proposed methods.

Keywords: Multi-objective optimization, local search heuristics, job scheduling.

1 Introduction

The single machine scheduling problem (SMSP) has been extensively investigated during the last decades [16][27][4][15]. Most of the contributions consider a single optimization criterion, although in practice the Decision Maker often faces several (usually conflicting) criteria. The main criteria to be considered are the minimization of the maximum completion time (i.e. makespan), the minimization of the total production time or flow time and the minimization of the total tardiness. The use of these objectives is well-justified in practice, as makespan minimization implies

[1] This research was funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and the Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

[2] Email:{jarroyo,alcione}@dpi.ufv.br

[3] Email: voviz.ottoni@gmail.com

doi:10.1016/j.entcs.2011.11.022

the maximization of the throughput and resource utilization, while the flow time minimization is related to average cycle time minimization and low Work-In-Process. The total tardiness criterion is related to job due dates and it is of great importance in manufacturing systems because when a job is not completed by its due date, certain costs and penalties are incurred.

Most research on job scheduling problems assumes that setup times are independent of the sequence of jobs [1]. However, in production systems, when a machine switches from processing one job to another, sequence dependent setup times are incurred. The magnitudes of setup times often depend on the similarity of the process technology requirements of two consecutive jobs. Typically, large setup times are associated with two consecutive jobs if they differ significantly in processing requirements or utilize different process technologies. The minimization of a single criterion in the SMSP with sequence dependent setup times is a NP-hard problem [6] [16], thus it is very unlikely to develop an efficient algorithm to solve it exactly.

In this work we address the SMSP with earliness and tardiness penalties. The general cases of this problem are those that allow inserted idle times together with distinct due dates of jobs. The assumption of no inserted idle times is inconsistent with the earliness/tardiness criterion because earliness is an irregular performance measure [3]. This scheduling problem is a very important and frequent industrial problem that is common to most Just in Time (JIT) production environments. JIT consists in delivering products and services at the right time for immediate use, having as main objective the continuous search for improvement of the production process, which is obtained and developed through reduced inventories [20][28]. JIT scheduling problems are very common in industry. In the JIT scheduling environment, the job should be finished as close to the due date as possible. An early job completion results in inventory carrying costs, such as storage and insurance costs. On the other hand, a tardy job completion results in penalties, such as loss of customer goodwill and damaged reputation.

In the literature, the SMSP with earliness and tardiness penalties are studied by various authors from a single-objective point of view. Most of the works consider distinct or common due dates. Lee and Choi [17] study the problem considering distinct due dates. They present an optimal algorithm, with polynomial complexity, to determine the optimal completion time for each job in a schedule determined by a Genetic Algorithm (GA). This optimal algorithm is used because may be interesting anticipate a job, even paying a penalty, if the penalty is shorter than the penalty generated by the tardiness [24]. In [21] distinct due dates are also considered and a hybrid heuristic which combines local search heuristics and GA is used. In [11], [20] and [29], the problem is studied considering common due dates. These authors propose different heuristics, such as, Tabu Search, GA, Variable Neighborhood Search (VNS) and Recovering Beam Search.

Wang and Yen [28] present a mathematical formulation of the SMSP with earliness and tardiness penalties considering distinct due windows instead of distinct due dates. They do not consider setup times. Wang and Yen [28] extend the optimal timing algorithms of Lee and Choi [17] to the case with distinct due windows and a

Tabu Search algorithm is proposed to generate approximate schedules. In [23] and [24] adaptive GAs are proposed to solve the problem with distinct due windows and sequence dependent setup times.

In this paper we deal with this last problem, the SMSP with earliness and tardiness penalties, considering distinct due windows and sequence dependent setup time. The objective of the problem is to determine feasible job schedules in order to minimize the total flow time subject to the minimum total weighted earliness/tardiness penalties of the jobs. The goal is to provide the decision maker with a set of efficient schedules (Pareto-optimal solutions) such that he/she may choose the most suitable schedule.

The most used methods for solving multi-objective combinatorial optimization problems are metaheuristics [13] [8]. Metaheuristic methods were originally conceived for single-objective optimization and the success achieved in their application to a very large number of problems has stimulated researchers to extend them to multi-objective combinatorial optimization problems. Applications of the VNS metaheuristic for multi-objective optimization are scarce. To the best of our knowledge, the first multi-objective VNS (MOVNS) algorithm was proposed by Geiger [9]. In [9], the MOVNS was applied to solve the permutation flow shop scheduling problem minimizing different combinations of criteria. The MOVNS of Geiger was used to solve other multi-objective problems in [18] and [19].

We propose two algorithms based on VNS metaheuristic to solve the bi-objective SMSP. VNS is a stochastic local search method that is based on the systematic change of the neighborhood during the search. The proposed algorithms are based on the algorithm developed by Geiger [9]. We include an intensification procedure based on constructing non-dominated solutions according to information taken on non-dominated partial solutions rather than evaluating complete solutions generated in the neighborhood of existing solutions. Simulation results and comparisons demonstrate the effectiveness, efficiency, and robustness of the proposed algorithms.

The remainder of this paper is organized as follows. The multi-objective problem definition is described in Section 2. Section 3 provides a detailed description of the MOVNS algorithms. Results of computational experiments to evaluate the performance of the proposed algorithms are reported in Section 4. Finally, Section 5 provides the concluding remarks.

2 Problem Statement

The multi-objective SMSP examined in this paper is stated as follows. There is a set of n jobs to be processed on a continuously available single machine. The machine can process only one job at a time. Each job j is available for processing at time zero, has a known processing time p_j, a due window $[de_j, dt_j]$ and earliness (α_j) and tardiness (β_j) penalties (de_j is the earliest due date and dt_j is the latest due date). Between the processing of two consecutive jobs i and j is considered a sequence dependent setup time s_{ij}. In this problem, a job j should preferably be completed within its due window $[de_j, dt_j]$. If the completion time C_j of job j is in

Fig. 1. a) schedule with machine idle time; (b) Schedule without machine idle time

the due window, i.e., if $C_j \in [de_j, dt_j]$, then the job j has no tardiness ($T_j = 0$) and no earliness ($E_j = 0$). Otherwise, it incurs on earliness ($\alpha_j E_j$) or tardiness ($\beta_j T_j$) penalties. The earliness and the tardiness are computed as $E_j = max\{0, de_j - C_j\}$ and $T_j = max\{0, C_j - dt_j\}$, respectively.

The objective of the problem is to determine feasible schedules with minimum total weighted earliness/tardiness penalties of the jobs (f_1) and minimum total flow time (f_2). For a sequence of jobs (permutation of the n jobs) $s = (i_1, ..., i_j, ..., i_n)$, the criteria f_1 and f_2 are computed as:

$$f_1(s) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j) \tag{1}$$

$$f_2(s) = \sum_{j=1}^{n} C_j \tag{2}$$

In the addressed problem the occurrence of machine idle time is allowed. This may be required to complete a job within its due window, avoiding earliness. To compute $f_1(s)$ and $f_2(s)$ of a given schedule s, first, the optimal starting time and completion time of each job in s are calculated. These times are calculated by using the optimal timing algorithm proposed by Wang and Yen [28]. Given a job sequence, the optimal timing algorithm decides the optimal completion time according to the corresponding due window for each job. The goal of this optimal algorithm is to minimize earliness/tardiness penalties for each sequence.

In this work, the first objective (f_1) is considered more important than the second (f_2). f_2 is computed after computing the optimal completion times of the jobs. Below we present an example in which two job sequences are evaluated by using the optimal timing algorithm of Wang and Yen [28].

Example 2.1 Let us consider the 5-jobs instance specified in Table 1. To simplify the example we do not consider setup times. Figures 1 (a) and (b) illustrate the schedule of the job sequences (1, 5, 3, 4, 2) and (5, 4, 1, 2, 3), respectively. The values of the objectives for the first sequence are $f_1 = 0$ and $f_2 = 360$. For the second sequence they are $f_1 = 580$ and $f_2 = 138$. Note that, in the first schedule there are machine idle times and the completions times of the jobs j are into of the corresponding due windows $[de_j, dt_j]$. In general, these objectives are conflicting meaning that a solution that improves one objective function will deteriorate the other.

Multi-objective optimization considers a vector $f(s) = (f_1(s), ..., f_r(s))$ of optimality criteria at once. The image of a solution s (in the decision space) is the

Table 1
Data for an instance with 5 jobs

Parameters of jobs	job 1	job 2	job 3	job 4	job 5
Processing time (p_j)	9	15	8	12	5
Earliest due date (de_j)	15	150	22	140	21
Latest due date (dt_j)	25	170	30	180	22
Earliness penalty (α_j)	3	2	4	1	5
Tardiness penalty (β_j)	7	6	8	4	10

point $z = f(s)$ in the objective space. As the relevant optimality criteria are often of conflicting nature, usually there is not a single solution s optimizing all components of $f(s)$ at once. Optimality in multi-objective optimization problems is therefore understood in the sense of Pareto-optimality, and the resolution of multi-objective optimization problems lies in the identification of all elements belonging to the Pareto or efficient set, containing all alternatives s which are not dominated by any other alternative s'. To properly compare two solutions in a multi-objective optimization problem, some definitions are needed. Without loss of generality we assume the minimization of the optimality criteria $f_i, i = 1, ..., r$.

Definition 2.2 A solution s dominates s' if the $z = f(s)$ dominates $z' = f(s')$, this is, $f_i(s) \leq f_i(s')$ for all i and $f_i(s) < f_i(s')$ for at least one i.

Definition 2.3 A solution s is Pareto-optimal (or efficient) if there is no s' such that $f(s')$ dominates $f(s)$.

3 Multi-objective VNS Algorithms

VNS, proposed originally in [22], is a metaheuristic based on the principle of systematic change of neighborhood during the search. It has been shown to be a simple and effective method for solving single-objective optimization problems, including traveling salesman problem and scheduling problems [10].

The first application of the VNS metaheuristic for multi-objective optimization was developed by Geiger [9]. Their VNS algorithm differs from the traditional single objective VNS algorithms on the random selection of neighborhoods and arbitrary selection of the base solution from the unvisited non-dominated solutions. That is, before conducting the neighborhood search, the base solution is randomly picked from the set of non-dominated solutions for which no neighborhood search has been performed yet, and one of the defined neighborhoods is then arbitrarily selected and applied to the chosen solution. After the neighborhood search, the current set of non-dominated solutions (approximation of the Pareto front) is updated accordingly.

In this work we applied the MOVNS algorithm of Geiger [9] to solve the bi-objective single machine scheduling problem defined in Section 2. This algorithm is named MOVNS1. A pseudocode description of the MOVNS1 algorithm is presented in Algorithm 1. In MOVNS1 we use greedy heuristics to generate three initial solutions (job sequences) s^1, s^2 and s^3. The set D of non-dominated solutions is

initialized with these solutions. We used two neighborhood structures (N_1 and N_2) to generate new solutions (neighbor solutions). In each iteration of the algorithm, a base non-dominated solution is randomly selected from D. This solution is marked as *visited*, and cannot be selected in the next iterations. From the base solution, neighbor solutions are generated using a neighborhood N_i which is chosen randomly. The set D is update with the solutions $s'' \in N_i$. A solution s'' is added to the set D if $s'' \notin D$ and it is not dominated by any solution of D. The solutions of D dominated by s'' are removed from D. In this study, the algorithm stops when a maximum CPU time is reached (*StoppingCriterion*).

Algorithm 1 MOVNS1()

```
{s1, s2, s3} ← solutions constructed by using dispatching rules;
D ← set of non-dominated solutions obtained of {s1, s2, s3};
while StoppingCriterion = false do
    Select randomly an unvisited solution (base solution) s from D;
    Mark s as a visited solution;
    Select at random a neighborhood structure Ni;
    Determine randomly a solution s' from Ni(s); //shaking
    for each neighbor s'' ∈ Ni(s') do
        Evaluate the solution s'' ;
        D ← non-dominated solutions obtained of D ∪ {s''};
    end for
    if all the solutions of D are marked as visited then
        All marks must be removed;
    end if
end while
Return D
```

In this paper we propose an improvement phase, called of intensification, for the MOVNS1 algorithm. From a non-dominated neighbor solution s, new non-dominated solutions are constructed by the intensification procedure. This procedure is based on two typical approaches used in multi-objective optimization: scalarizing functions and Pareto dominance. The first approach is based on the optimization of different weighted utility functions. To select the best solution, we use the weighted linear utility function $f_w = w_1 f_1 + w_2 f_2$, where the weights w_1 and w_2 are randomly generated such that $w_1 + w_2 = 1$. Due to the randomness of weights, searching direction can be enriched, and non-dominated solutions with good diversity can be obtained. In the Pareto dominance approach, only non-dominated solutions are analyzed.

The pseudocode description of the proposed MOVNS algorithm, called MOVNS2, is presented in Algorithm 2. The algorithm has an input parameter d used in the intensification phase (d is the number of jobs to be removed from a sequence). In MOVNS2, the intensification procedure is based on scalarizing functions ($Intensification1$). The algorithm that uses the intensification procedure based on Pareto dominance approach ($Intensification2$) is called MOVNS3. The pseudocodes of the algorithms $Intensification1$ and $Intensification2$ are presented in Algorithms 3 and 4, respectively.

In the next subsections, we describe in detail each phase of the MOVNS algorithms.

Algorithm 2 MOVNS2(d)

$\{s_1, s_2, s_3\} \leftarrow$ solutions constructed by using dispatching rules;
$D \leftarrow$ set of non-dominated solutions obtained of $\{s_1, s_2, s_3\}$;
while $StoppingCriterion = false$ **do**
  Select randomly an *unvisited* solution (base solution) s from D;
  Mark s as a *visited* solution;
  Select at random a neighborhood structure N_i;
  Determine randomly a solution s' from $N_i(s)$; //shaking
  $D_a \leftarrow \Phi$ (empty set);
  for each neighbor $s'' \in N_i(s')$ **do**
    Evaluate the solution s'';
    $D_a \leftarrow$ non-dominated solutions obtained of $D_a \cup \{s''\}$;
  end for
  Select randomly a solutions s from D_a
  $D_b \leftarrow Intensification1(s, d)$;
  $D \leftarrow$ non-dominated solutions obtained of $D \cup D_a \cup D_b$;
  if all the solutions of D are marked as *visited* **then**
    All marks must be removed;
  end if
end while
Return D

Algorithm 3 Intensification1(s, d)

$D_b \leftarrow \Phi$;
Define randomly the weights w_1 and w_2, such that $w_1 + w_2 = 1$;
$s_p \leftarrow$ remove at random d jobs from s and insert these jobs in s_R; //s_p is a partial sequence with $n - d$ jobs
for $i \leftarrow 1$ to d **do**
  Insert job $s_R(i)$ in all positions of s_p generating a set S of $(n - d + i)$ sequences;
  Evaluate each sequence $s' \in S$;
  if $d = n$ **then**
    $D_b \leftarrow$ non-dominated solutions obtained of $D_b \cup \S$;
  else
    $s_p \leftarrow$ best solutio from S (with respect to $f_w = w_1 f_1 + w_2 f_2$);
  end if
end for
Return D_b

Algorithm 4 Intensification2(s, d)

$s_p \leftarrow$ remove at random d jobs from s and insert these jobs in s_R;
$D_b \leftarrow \{s_p\}$;
for $i \leftarrow 1$ to d **do**
  $D_a \leftarrow \Phi$;
  for each partial sequence $s_p \in D_b$ **do**
    Insert job $s_R(i)$ in all positions of s_p generating a set S of $(n - d + i)$ sequences;
    Evaluate each sequence $s' \in S$;
    $D_a \leftarrow$ non-dominated solutions obtained of $D_a \cup \{s'\}$;
  end for
  $D_b \leftarrow D_a$;
end for
Return D_b

3.1 Initial Solutions

In this work, we use simple despatching rules to generate initial non-dominated solutions (instead of generating random solutions). In the algorithms MOVNS1, MOVNS2 and MOVNS3, the set D of non-dominated solutions is initialized with three solutions (sequences): s^1, s^2 and s^3. The solutions s^1 and s^2 are generated using the Earliest Due Date (EDD) rule, in which the jobs are arranged in increasing order of the dates de_j and dt_j, respectively. The solution s^3 is constructed using the Shortest Processing Time (SPT) rule, in which the jobs are arranged in increasing order of the total processing time. The non-dominated solutions of $\{s^1, s^2, s^3\}$ are stored in the set D. This set will contain at least one solution.

3.2 Neighborhood Structures and Local Search

Local search methods usually are based on neighborhood search. These methods begin with a solution s, and generate a neighborhood of this solution. Such neighborhood contains similar solutions, obtained by applying simple changes (single moves) on the current solution s.

The MOVNS algorithms developed in this paper use two neighborhood structures: insertion and exchange. For a given sequence $s = (i_1, ..., i_n)$, the neighborhood structures are described below.

Insertion neighborhood (N_1): the neighbors of s are generated by inserting job i_q, $1 \leq q \leq n$, in another position k of the sequence, $k \leq q$. If $k < q$, then $k \neq q-1$. $N_1(s)$ neighborhood has size $(n-1)^2$.

Exchange neighborhood (N_2): the neighbors of s are generated by interchanging jobs i_q and i_p in the sequence, $1 \leq q \leq n$, $1 \leq p \leq n$, $q \neq p$. $N_2(s)$ neighborhood has size $n(n-1)/2$.

The MOVNS algorithms star with a base solution s select randomly from the current set of non-dominated solutions (D). The selected solutions are marked as visited and it will be excluded from the selection of the base one. If all members in the set D are marked as visited before reaching the stopping criterion of the algorithm, then all marks will be reset and the selection procedure can start over again.

In each iteration of the algorithms, a neighborhood structure N_i is selected randomly. The base solution s is perturbed by choosing randomly a solution s' from $N_i(s)$ neighborhood (shaking). Then, all the neighboring solutions of s' are analyzed, that is, the neighborhood $Ni(s')$ is explored.

In the MOVNS1, the set D of non-dominated solutions are update with the solutions $s'' \in Ni(s')$. In the MOVNS2 and MOVNS3 algorithms, the non-dominated neighbor solutions are stored in a set D_a. From a solution selected randomly from D_a, the intensification procedures are executed.

3.3 Intensification Procedures

In this paper we propose two intensification procedures to improve a non-dominated solution selected randomly from set D_a (set of non-dominated solutions obtained in local search). These procedures are based on a partial enumeration heuristic proposed by [25]. The first intensification procedure, $Intensification1$, is based on the optimization of different weighted utility functions ($f_w = w_1 f_1 + w_2 f_2$). The second intensification procedure, $Intensification2$, is based on the Pareto dominance approach.

Each time the $Intensification1$ algorithm is executed, new weights w_1 and w_2 are randomly generated, such that $w_1 + w_2 = 1$. Thus, the MOVNS2 algorithm can explore different search directions on the Pareto-optimal frontier [2].

The two intensification procedures are composed of two stages: *destruction* and *construction*. In the destruction stage, d jobs (selected randomly) are removed from s (solution selected randomly from D_a) and a partial solution s_p (of size $n-d$) is

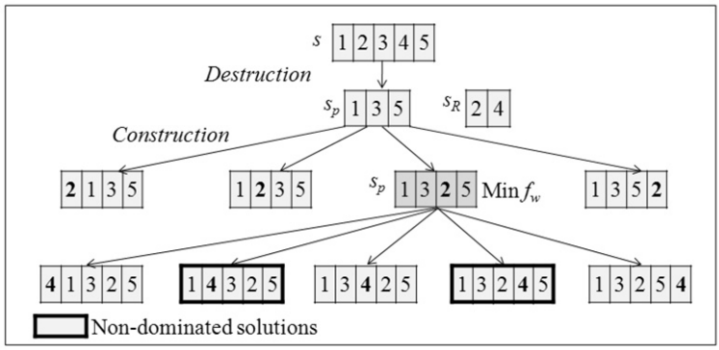


Fig. 2. Example of $Intensification1$ for $n = 5$ and $d = 2$

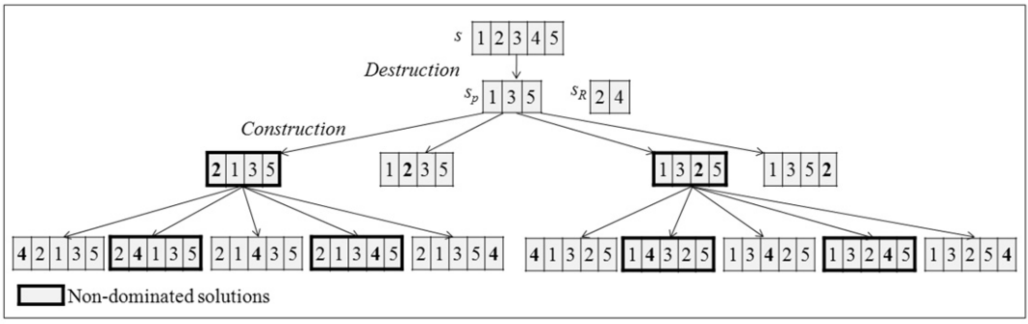


Fig. 3. Example of $Intensification2$ for $n = 5$ and $d = 2$

obtained. The removed jobs are stored in s_R ($s_R(i), i = 1, ..., d$, are the removed jobs). The construction stage has d steps. In step $i = 1$, $(n-d+1)$ partial solutions are constructed by inserting job $s_R(1)$ in all possible position of s_p.

In the $Intensification1$ algorithm, from the $(n - d + 1)$ partial solutions, the best is chosen (one that has the lowest value of f_w) and it replaces s_p. The other steps, $i = 2, ..., d$, are similar. Note that, in step $i = d$, n complete solutions are constructed. From these n complete solutions, the set D_b of non-dominated solutions is determined. The $Intensification1$ procedure returns this set. Figure 2 shows an example (for $n = 5$ and $d = 2$) of the *destruction* and *construction* stages of the $Intensification1$ procedure.

Step $i = 1$ of $Intensification2$ is the same, i.e. $(n - d + 1)$ partial solutions are constructed. From these partial solutions, the non-dominated solutions are selected. In the next step, new solutions (of size $n - d + 2$) are obtained by inserting job $s_R(2)$ in each position of the partial non-dominated solutions. From the solutions constructed in each step, in the $Intensification2$ algorithm, the non-dominated solutions are always selected. In step d, the set D_b of complete non-dominated solutions is determined. Figure 3 illustrates the idea of the $Intensification2$ procedure for $n = 5$ and $d = 2$.

4 Computational Experiments

In this work, we analyze the efficiency of the proposed intensification procedures used in the MOVNS1 algorithm by Geiger [9]. The MOVNS1 algorithm with $Intensification1$ and $Intensification2$ are called MOVNS2 and MOVNS3, respectively.

The three algorithms were coded in C++ and executed on an Intel Core Quad with a 2.4GHz and 2.0 of RAM. The algorithms were run with the same stopping criterion ($StoppingCriterion$) based on an amount of CPU time. This time is giving by $3n$ seconds (n = number of jobs) and it depends on the size of the considered instance. In this way, we assign more time to larger instances that are obviously more time consuming to solve. Stopping criteria based on CPU times are widely used for performance comparison of heuristic algorithms [25] [26].

The parameter d used in the intensification procedures was tuned experimentally. The MOVNS3 algorithm was run using different values for this parameter. The set of values tested was {2, 4, 6, 8}. The computational tests showed that the MOVNS3 algorithm generates the best results with $d = 6$. Due to lack of space, these results are not presented. The same value of the parameter d was used in the algorithm MOVNS2.

4.1 Problems Instances

The performance of the algorithms MOVNS1, MOVNS2 and MOVNS3 are compared on 72 problems instances. The sizes of these instances are n =20, 30, 40, 50, 75 and 100. The problems were generated randomly, similar to that of Wang and Yen [28] and Ribeiro et al.[23][24]. For each job j, the processing time p_j and the tardiness penalty β_j were uniformly generated in [1, 100] and [20, 100], respectively. As the majority of real cases, production lateness is less desirable than earliness, the costs for production earliness (α_j) generated k times of the cost of the same job, k being a random real number in the interval [0, 1]. The center of the due window $[de_j, dt_j]$ was uniformly generated in $[(1-T-RDD/2)TP, (1-T+RDD/2)TP]$, where TP is the total processing time of all the jobs, T is the tardiness factor, and RDD is the relative range of the due windows. T and RDD took values from {0.1, 0.2, 0.3, 0.4} and {0.8, 1.0, 1.2}, respectively. The sizes of the due windows were uniformly distributed in $[1, TP/n]$. For all pair of jobs (i, j), $i \neq j$, the setup times s_{ij} are generated uniformly from the interval [0, 50].

Because T and RDD take 4 and 3 different values, respectively, there are total 12 settings for both parameters. One problem instance was generated for each pair of (T, RDD) in problem size n, giving $12 \times 6 = 72$ problem instances in total.

4.2 Performance Measures

For each problem instance, we compare the non-dominated solutions obtained by the three algorithms. We denoted by D_1, D_2 and D_3 the sets of non-dominated solutions (approximated Pareto fronts) obtained by the algorithms MOVNS1, MOVNS2

and MOVNS3, respectively. Since for the addressed problem the optimal Pareto front for each instance is not known, a reference set, constituted by gathering all non-dominated solutions obtained by the three tested algorithms, is used. The reference set (the best known Pareto front) is denoted by Ref. The performance of an algorithm is then measured in terms of the quality of the solutions obtained by this algorithm with respect to the solutions in Ref. In this paper, three measures are used: cardinal measure, distance metric and hypervolume indicator.

Cardinal measure: for each algorithm we compute the number of obtained non-dominated solutions that belong to the reference set, i.e. $|Ref \cap D_1|, |Ref \cap D_2|$ and $|Ref \cap D_3|$.

Distance metric: we also measure the quality of the sets $D_i, (i = 1, 2, 3)$ relative to the reference set Ref by using the following metrics [5][14]:

$$d_{av}(D_i) = \frac{1}{|Ref|} \sum_{y \in Ref} min\{d(x, y) | x \in D_i\} \tag{3}$$

$$d_{max}(D_i) = max_{y \in Ref}\{min\{d(x, y) | x \in D_i\}\} \tag{4}$$

being $d(x, y)$:

$$d(x, y) = \sqrt{(f_1^*(y) - f_1^*(x))^2 + (f_2^*(y) - f_2^*(x))^2} \tag{5}$$

where $f(.)$ is the objective function i normalized according to the set of solutions in the reference set Ref. The normalization is accomplished in the following manner:

$$f_i^*(x) = 100 \times \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{6}$$

where f_i^{max} and f_i^{min} are, respectively, the maximum and minimum values of the objective function i in the reference set Ref. The d_{av} and d_{max} indicators are a widely employed measure for multi-objective problems [12] [7].

Hypervolume indicator: this metric was introduced by Zitzler and Thiele [30] and it can be defined as follows:

$$H(D_i) = H^*(Ref) - H^*(D_i) \tag{7}$$

where $H^*(X)$ is the hypervolume (area in the case of two objectives) of the solution space dominated by the solutions of the set X, i.e. the portion of the objective space that is dominated by X. Note that we considerered the hypervolume difference to the reference set Ref, and we denoted this indicator as H. Smaller values of $H(D_i)$ correspond to higher quality of the solutions in D_i and it indicates both a better convergence to as well as a good coverage of the reference set.

4.3 Comparison of Results

The three algorithms were run five independent times (replicates) for all the 72 instances of the problem. The sets D_1, D_2 and D_3 contain the non-dominated solu-

Table 2
Performance of the algorithms: cardinal measure

		MOVNS1		MOVNS2		MOVNS3	
n	$\|Ref\|$	$\|D_1\|$	$\|Ref \cap D_1\|$	$\|D_2\|$	$\|Ref \cap D_2\|$	$\|D_3\|$	$\|Ref \cap D_3\|$
20	1071	996	446	987	761	1050	628
30	1316	836	75	917	272	1321	981
40	1249	676	28	718	161	1280	1060
50	1235	558	12	635	148	1282	1075
75	1364	460	9	406	63	1435	1292
100	843	152	6	192	67	889	770
Total	7078	3678	576	3855	1472	7257	5806

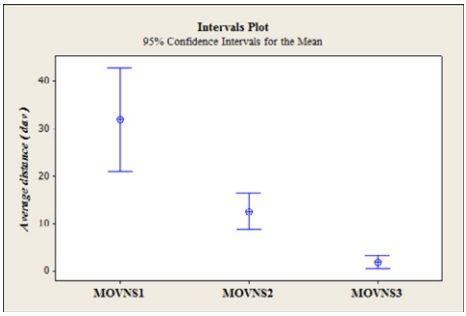
Table 3
Performance of the algorithms: distance and hypervolume measures

	MOVNS1			MOVNS2			MOVNS3		
n	d_{av}	d_{max}	$H \times 10^{-5}$	d_{av}	d_{max}	$H \times 10^{-5}$	d_{av}	d_{max}	$H \times 10^{-5}$
20	8.2	23.0	1836.8	0.6	9.5	137.4	8.0	19.6	1799.2
30	4.7	15.3	5751.9	3.3	18.1	3960.8	0.8	8.8	672.7
40	11.2	24.9	30058.4	8.2	29.1	22582.4	0.4	4.4	806.6
50	16.4	38.0	89502.2	9.7	37.3	48277.5	0.6	5.1	2088.0
75	37.7	65.0	642678.5	16.2	45.0	273728.3	0.2	5.4	8302.4
100	113.2	154.8	1430734.9	37.1	82.5	546607.6	1.0	6.0	36927.1
Average	31.9	53.5	366760.5	12.5	36.9	149215.7	1.8	8.2	8432.7

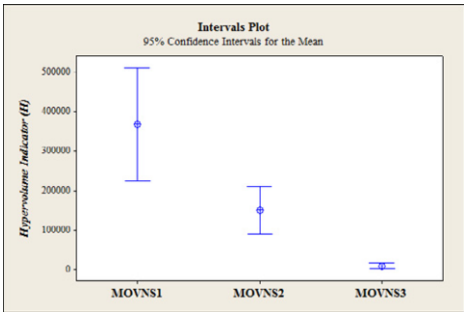
tions found among all the runs. The cardinal, distance and hypervolume measures are calculated for these sets.

Table 2 presents the comparison among MOVNS1, MOVNS2 and MOVNS3 regarding the cardinal measure. For each group of 12 instances of size n, Table 2 shows the total number of reference solutions $|Ref|$, the total number of solutions obtained by each algorithm ($|D_1|$, $|D_2|$ and $|D_3|$) and the total number of reference solutions provided by each algorithm ($|Ref \cap D_1|$, $|Ref \cap D_2|$ and $|Ref \cap D_3|$). Note that, the algorithms MOVNS1, MOVNS2 and MOVNS3 generate their own set of non-dominated solutions (D_1, D_2 and D_2), which do not necessarily belong to Ref. We note that for all groups of instances, the algorithm MOVNS3 determines a greater number of solutions in both sets D_3 and $|Ref \cap D_3|$. For all the 72 instances tested, a total of 7078 reference solutions were obtained, from which 576 (8.1%), 1472 (20.8%) and 5806 (82.0%) reference solutions were obtained, respectively, by MOVNS1, MOVNS2 and MOVNS3. Based on the cardinal measure, for all groups of instances, except for the group of instances with $n = 20$, the algorithm MOVNS3 is superior to the algorithms MOVNS1 and MOVNS2, ($|Ref \cap D_3| > |Ref \cap D_1|$ and $|Ref \cap D_3| > |Ref \cap D_2|$). The algorithm MOVNS2 is superior to MOVNS1 for all groups of instances ($|Ref \cap D_2| > |Ref \cap D_1|$).

In Table 3, the results obtained by the three algorithms are compared by using the distance measures and hypervolume indicator. For each group of instances and for each algorithm, the average values of d_{av}, d_{max} and H are presented. We can see that for all groups of instances, except for the group of instances with $n = 20$, the MOVNS3 algorithm also performs better than the other algorithms regarding the three measures. The algorithm MOVNS2 performs better than MOVNS1 and MOVNS3 for instances with $n = 20$. For all groups of instances, the algorithm MOVNS2 is notoriously better than MOVNS1. From the computational tests, we can see that the best results are provided by the proposed MOVNS algorithms, specially the MOVNS3 version.



(a) d_{av} performance measure



(b) H (hypervolume) performance measure

Fig. 4. Intervals plot

In order to validate the results, an Analysis of Variance (ANOVA), is applied in order to check if the observed differences are statistically significant. The ANOVA has been carried out employing the d_{av} and H measures as response variables. All the tests have been executed with a confidence level of 95% ($\alpha = 0.05$). With respect to d_{av} and H, the ANOVA analysis indicates that these measures are statistically different for the three algorithms (p-values = 0.00). From the individual 95% Confidence Intervals, we observed that the intervals of the algorithms are not overlapping. This means that the considered measures are statistically different. We also carry out a multiple comparison test of Tukey in order to verify that the results are statistically significant. Figures 4a and 4b depict the means plots with Tukey confidence intervals with a 95% confidence level from the ANOVA test and report on the interaction between the response variables d_{av} and H, respectively. We can clearly see that there are statistically significant differences between the d_{av} and H values among the algorithms. We can observe that the algorithm MOVNS3 shows the best performance.

5 Conclusions

In this work was applied the MOVNS algorithm of Geiger [9] to solve the single machine scheduling problem with distinct due windows. Two criteria are simultaneously minimized, the total earliness/tardiness penalties and the total flow time. We proposed two new versions of the MOVNS algorithm, named MOVNS2 and MOVNS3. Our algorithms use an intensification procedure based on a partial enumeration heuristic. It should be noted that by using the intensification procedure has improved the solution quality significantly.

After the computational experiments and statistical analysis we can conclude that the proposed algorithms (MOVNS2 and MOVNS3) show an excellent performance overcoming the original MOVNS. For future study, this work would be extended to apply the MOVNS with intensification to solve other scheduling problems.

References

[1] Allahverdi, A., C.T. Ng, T.C.E. Cheng and M.Y. Kovalyov, *A survey of scheduling problems with setup times or costs*, Eur. Jour. of Operational Research, **187**, (2008), 985-1032.

[2] Arroyo, J.E.C., P.S. Vieira and D.S. Vianna, *A GRASP algorithm for the multi-criteria minimum spanning tree problem*, Annals of Operations Research, **159**, (2008), 125-133.

[3] Baker, K.R. and G.D. Scudder, *Sequencing with earliness and tardiness penalties: a review*, Operations Research, **38**, (1990),22-36.

[4] Chen, W. J., *Minimizing total flow time in the single-machine scheduling problem with periodic maintenance*, Jour. of the Operational Research Society, **57**, (2005), 410-415.

[5] Czyzak, P. and A. Jaszkiewicz, *Pareto simulated annealing - a metaheuristic technique for multiple objective optimization*, J Multi-Crit Decis Making, **7**, (1998), 34-47.

[6] Du, J., and J.Y.T. Leung, *Minimizing total tardiness on one machine is NP-hard*, Mathematics of Operations Research, **15**, (1990), 483-495.

[7] Framinan, J.M. and R. Leisten, *A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria*, OR Spectrum, 30 (2008),787-804.

[8] Gandibleux, X. and M. Ehrgott, *1984-2004, 20 years of multiobjective metaheuristics*, Lecture Notes in Computer Science, **3410**, (2005), 33-46.

[9] Geiger, M.J., *Randomized variable neighborhood search for multi objective optimization*, 4th EU/ME: Design and Evaluation of Advanced Hybrid Meta-Heuristics, (2004), 34-42.

[10] Hansen, P. and N. Mladenovic, *Variable Neighborhood Search*, "Handbook of metaheuristics", Kochenberger G.A and Glover F. Eds. Kluwer Academic, (2003), 145-184.

[11] Hino,C.M., D.P. Ronconi and A.B. Mendes, *Minimizing earliness and tardiness penalties in a single-machine problem with a common due date*, Eur. Jour. of Operational Research, **160**, (2005), 190-201.

[12] Ishibuchi, H., T. Yoshida and T. Murata, *Balance between genetic local search in memetic algorithms for multiobjective permutation flowshop scheduling*, IEEE Trans Evol Comput **7** (2003), 204-223.

[13] Jones, D.F., S.K. Mirrazavi, and M. Tamiz, *Multi-objective metaheuristics: An overview of the current state-of-art*, Eur. Jour. of Operational Research, **137**, (2002), 1-19.

[14] Knowles, J.D., and D.W.Corne, *On metrics for comparing non-dominated sets*, in: *Proc. of Congress on Evolutionary Computation*, Hawaii, (2002), 711-716.

[15] Kuo, W-H. and D-L. Yang, *Minimizing the makespan in a single machine scheduling problem with a time-based learning effect*, Information Processing Letters, **97**, (2006), 64-67.

[16] Lenstra, J.K., A.H.G.R. Kan and P. Brucker, *Complexity of machine scheduling problems*, Annals of Discrete Mathematics (1977), 343-362.

[17] Lee, C.Y. and Y.J. Choi, *A Genetic Algorithm for Job Sequencing Problems with Distinct Due Dates and General Early-Tardy Penalty Weights*, Computers & Operations Research, **22**(8), (1995), 857-869.

[18] Liang, Y-C., H-L.A. Chen, and C-Y. Tien, *Variable Neighborhood Search for Multi-Objective Parallel Machine Scheduling Problems*, in: *Proc. of the 8th International Conference on Information and Management Sciences* (IMS 2009), China, (2009), 519-522.

[19] Liang, Y-C and M-H. Lo, *Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm*, Jour. of Heuristics, **16**, (2010), 511-535.

[20] Liao, C-J. and C-Cg. Cheng, *A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date*, Comp Ind Eng., **52**, (2007), 404-413.

[21] M'Hallah, R., *Minimizing total earliness and tardiness on a single machine using a hybrid heuristic*, Computers & Operations Research, **34**, (2007), 3126-3142.

[22] Mladenovic, N. and P. Hansen, *Variable Neighborhood Search*, Computers and Operations Research, **24**, (1997), 1097-1100.

[23] Ribeiro, F.F., Souza, S.R. and Souza, M.J.F., *An adaptive genetic algorithm for solving the single machine scheduling problem with earliness and tardiness penalties*, in: *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*. San Antonio, USA, (2009), 698-703.

[24] Ribeiro, F.F., S.R. Souza, M.J.F. Souza, and R.M. Gomes, *An adaptive Genetic Algorithm to Solve the Single Machine Scheduling Problem with Earliness and Tardiness*, in: *Proc. of the IEEE Congress on Evolutionary Computation*, (2010), 1-8.

[25] Ruiz, R. and T. Stützle, *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem*, Eur. Jour. of Operational Research, **177**, (2007), 2033-2049.

[26] Ruiz, R. and T. Stützle, *An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives*, Eur. Jour. of Operational Research, **187**, (2008), 1143-1159.

[27] Tan, K.C., R. Narasimhan, P.A. Rubin, and G.L. Ragatz, *A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times*, OMEGA, **28** (2000), 313-326.

[28] Wang, G. and B.P. Yen, *Tabu Search for Single Machine Scheduling with Distinct Due windows and Weighted Earliness/Tardiness Penalties*, Eur. Jour. of Operational Research, **142**, (2002), 129-146.

[29] Ying, K.C., *Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm*, Computers and Industrial Engineering, **55**, (2008) 494-502.

[30] Zitzler, E., L. Thiele, M. Laumanns, C.M. Foneseca and F.V. Grunert, *Performance Assessment of Multiobjective Optimizers: An Analysis and Review*, IEEE Transactions on Evolutionary Computation, **7**, (2003), 117-132.