An Adaptive Genetic Algorithm to Solve the Single Machine Scheduling Problem with Earliness and Tardiness Penalties

Fábio F. Ribeiro, Sérgio R. de Souza, Marcone J. F. Souza and Rogério M. Gomes

Abstract— **This paper deals with the Single Machine Scheduling Problem with Earliness and Tardiness Penalties, considering distinct due windows and sequence-dependent setup time. Due to its complexity, an adaptive genetic algorithm is proposed for solving it. Five search operators are used to explore the solution space and the choice probability for each operator depends on the success in a previous search. The initial population is generated by the combination between construct methods based on greedy, random and GRASP techniques. For each job sequence generated, a polynomial time algorithm are used for determining the processing initial optimal date to each job. During the evolutive process, a group with the best five individuals generated by each *crossover* operator is built. Then, periodically, a Path Relinking module is applied taking as base individual the best one so far generated by the algorithm and as guide individual each one of the five best individuals generated by each crossover operator. Three variations of this algorithm were submitted to computational experiments. The results shows the effectiveness of the proposal.**

Index Terms— **Single machine scheduling, Genetic algorithm, metaheuristics.**

I. Introduction

Nowadays, scheduling problems are one of the most studied problems [1]. It occurs mainly by two aspects: the first one concerns their practical importance, with various applications in several industries, like chemical, metallurgic and textile industries. The second aspect is about the difficulty for solving the majority problems of this class. This paper deals with the Single Machine Scheduling Problem with Earliness and Tardiness Penalties (SMSPETP) with distinct due windows and sequence-dependent setup time. To our knowledge, this problem has not been still object of great attention of the scientific community, as it could seen in the recent survey [1].

The criteria to penalize the tardiness and earliness production goes to the Just-in-Time philosophy goal, in which the production occurs only when it is necessary. According to [2], the due window existence for each job is due to an uncertainly situation or tolerance related to the due date. We accept that this time interval operations can be finalized without costs. On the other hand, in industrial processes majority, the machines can be prepared to do new jobs, including the time to obtain tools, positioning materials that will be used in the process, cleaning process, prepare, tools adjustment and materials inspection. The time necessary to this preparation is known as setup time. Many production scheduling researches disregard this time or include it in the operation processing time. This act simplifies the analysis but affect the solution quality when the setup time has a relevant variability in function of the job sequence in machine. This work considers that the setup times are dependent of the production sequence. Since it was showed in [3] that a simplified version of this problem is NP-Hard, the application of metaheuristics for solving this problem is justified.

Simplified versions of this problem are studied by various authors. In [4] it is used Tabu search and Genetic Algorithms for solving the problem with common delivery date. These algorithms use optimal solution properties to explore the solution space. In [5] this problem is solved considering common delivery dates and setup time included in a job processing time and independent of production sequence. The author shows Recovering Beam Search (RBS), an improved version of Beam Search (BS) method, a branch and bound algorithm at which only w nodes more promises of each search tree levels are selected to a future ramification, while the rest of nodes are cut forever. With the objective to avoid that wrong decisions about nodes cut are followed, RBS Algorithm uses a recovered tool that searches for better partial solutions which controls the selected previous ones. [6] used genetic algorithm to solve the problem with distinct delivery dates. In this last work, a specific algorithm, with polynomial complexity, was developed to determine the optimal processing conclusion date for each job in a sequence produced by Genetic Algorithm. This algorithm is necessary because can be interesting anticipate a job, even paying a penalty, if the penalty is shortest than the penalty generated by the tardiness. [2] studied this problem considering distinct due windows, in replacement to due dates. An adaptation in the algorithm of [6] was done to determine optimal dates for each job to include this new characteristic.

To solve this scheduling problem with the characteristics presented, an Adaptive Genetic Algorithm, so-called AGA, is proposed here. In order to generate different individuals having good quality, the initial population was generated by a construction method based on GRASP [7], which uses five dispatch rules to form the individuals. During the

Fábio Fernandes Ribeiro is with the Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG, Avenida Amazonas, 7675, CEP 30.510-000, Belo Horizonte - Minas Gerais - Brazil (phone: +55 31 9901-6248; email: fabiobh@gmail.com).

Marcone Jamilson Freitas Souza is with the ICEB, Universidade Federal de Ouro Preto, Campus Universitário, CEP 35.400-000 Ouro Preto, Brazil (email: marcone@iceb.ufop.br).

Sérgio Ricardo de Souza is with the Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG, Avenida Amazonas, 7675, CEP 30.510-000, Belo Horizonte - Minas Gerais - Brazil (phone: +55 31 3319-6805; email: sergio@dppg.cefetmg.br).

Rogério Martins Gomes is with the Centro Federal de Educação Tecnológica de Minas Gerais - CEFET/MG, Avenida Amazonas, 7675, CEP 30.510-000, Belo Horizonte - Minas Gerais - Brazil (phone: +55 31 3319-6805; email: rogerio@decom.cefetmg.br).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

evolution process, the population passes through mutation and crossover conventional process. However, the crossover uses criteria based on solution quality generated by each crossover operator to choose which operator will be used. By the way, according to how well an operator performs the probabilities it is chosen are increased or decreased during the evolutionary. A local search is applied in the best offspring produced for each operator, in order to refine it. The survival population is composed by individuals chosen by elitism technique. Then, mutation is applied in a survival population slice in order to diversify it. Periodically, a Path Relinking module is applied taking as base individual the best one so far generated by the algorithm and as guide individual each one of the five best individuals generated by each crossover operator. The population improvement occurs until the stop criteria is reached.

The remaining of this work has the following structure: section II details the studied problem; section III formulates the scheduling problem as a Mixed Integer Problem; section IV presents the adaptive algorithm for solving SMSETP; section V shows and discusses the results. Finally, section VI ends this work.

II. Problem description

The sequencing problem studied in this work is the single machine scheduling with earliness and tardiness penalties with distinct due windows and sequence-dependent setup time. It has the following characteristics:

- one machine must process a set of n jobs;
- each job has a processing time P_i, a initial date Ei and the final date Ti desired to the end of processing;
- the machine executes one job per time and if the job processing was started, the job must be finished and processing interruptions are not allowed;
- every job are available to processing in the time 0;
- When the job j is sequenced immediately after a job i, being this part of different products family, a setup time S_{ij} is necessary to set the machine. Setup times equal 0 ($S_{ij} = 0$) means products to the same family. The initial setup time are considered, that means, the setup time to the first job in the sequence is 0;
- The idle time between the execution with two consecutive jobs are allowed;
- The jobs must be finalized inside the time interval $[E_i, T_i]$, called due window. In case of job finalization before the E_i, so there is a cost to earliness. Case the job are finalized after T_i, so a cost will be generated for tardiness. The jobs finalized inside the due window there is no cost;
- The costs to earliness and tardiness of production depends on jobs, each job i have a earliness cost α_i and a tardiness cost $beta_i$;
- The objective is to minimize the summation of the earliness and tardiness penalties.

III. The mixed integer programming model

The mixed integer programming model (MIP) below, developed by [8], formulates the scheduling problem described in the previous section. This formulation uses the following notation:

- s_i: the starting time of job i;
- C_i: the completion time of job i;
- y_{ij}: binary variable that assumes value 1 if job j is processed immediately after job i and 0, otherwise;
- e_i: the earliness of job i, that is, $e_i = \max\{0, E_i - C_i\}$;
- t_i: the tardiness of job i, that is, $t_i = \max\{0, C_i - T_i\}$;
- M: a sufficiently large number;
- 0: a fictitious job, which precedes and succeeds all other jobs;

It also assumes that $P_0 = 0$, $S_{0i} = S_{i0} = 0 \quad \forall i \in \{1, 2, \cdots, n\}$

$$\min \quad Z = \sum_{i=1}^{n} (\alpha_i e_i + \beta_i t_i) \tag{1}$$

$$\text{s.t:} \quad s_j - s_i - y_{ij}(M + S_{ij}) \geq P_i - M \quad \forall\, i = 0, \ldots, n;\ \forall\, j = 0, \ldots, n;\ i \neq j \tag{2}$$

$$\sum_{j=0,\ j\neq i}^{n} y_{ij} = 1 \quad \forall\, i = 0, \ldots, n \tag{3}$$

$$\sum_{i=0,\ i\neq j}^{n} y_{ij} = 1 \quad \forall\, j = 0, \ldots, n \tag{4}$$

$$s_i + P_i + e_i \geq E_i \quad \forall\, i = 1, \ldots, n \tag{5}$$

$$s_i + P_i - t_i \leq T_i \quad \forall\, i = 1, \ldots, n \tag{6}$$

$$s_i \geq 0 \quad \forall\, i = 0, \ldots, n \tag{7}$$

$$e_i \geq 0 \quad \forall\, i = 1, \ldots, n \tag{8}$$

$$t_i \geq 0 \quad \forall\, i = 1, \ldots, n \tag{9}$$

$$y_{ij} \in \{0, 1\} \quad \forall\, i, j = 0, \ldots, n \tag{10}$$

The objective function (1) expresses the total earliness and tardiness cost. The constraints (2) establish that job j can be processed when job i is finished and the machine is prepared to processes it. The constraints (3), (4) and (10) guarantee that the variable y_{ij} assumes value 1 if and only if job j is processed immediately after job i. The constraints (5) and (6) define, respectively, the tardiness and earliness values according of the due window. The constraints (7) to (10) define the type of the variables.

IV. Heuristic framework

In this section, the Adaptive Genetic Algorithm (AGA) framework is detailed.

A. Individual representation

An individual (solution) to this problem is represented by a vector v of n genes (jobs), with position i of each gene showing the production sequence of job v_i. For example, in the sequence $v = \{7, 1, 5, 6, 4, 3, 2\}$, the job 7 is the first to be processed and job 2, the last.

```
procedure Construction(g(.), γ, v);
1  v ← ∅;
2  Initialize a set C of candidate genes;
3  while (C ≠ ∅) do
4      g_min = min{g(t) | t ∈ C};
5      g_max = max{g(t) | t ∈ C};
6      RCL = {t ∈ C | g(t) ≤ g_min + γ × (g_max − g_min)};
7      Select, randomly, a gene t ∈ RCL;
8      v ← v ∪ {t};
9      Update C;
10 end-while;
11 Return v;
end Construction;
```

Fig. 1. Procedure to build an individual

B. Evaluation of individuals

An individual is evaluated by the objective function presented in equation (1) of the MIP model (see section III), which determines the total earliness and tardiness penalties. The individual who obtained the shortest value to objective function is considered the most adapted.

C. Initial population construction

The initial population of the proposed AGA is generated by GRASP construction phase ([7]), having as guide function five dispatch rules (EDD, TDD, SPT, WSPT and LPT), as described below. For each construction (GRASP + Dispatch rule), 200 individuals are generated. Next, the individuals are ordered, from the best to the worst, according to the evaluation function values. The initial population are composed by the best 100 individuals of this set.

1) Dispatch rules: In the dispatch rule EDD (Earliest Due Date), the jobs are ordered by the earliest due date. Jobs with earliest due date will be processed before that one with tardiness due date. By the dispatch rule TDD (Tardiness Due Date), the jobs also be ordered based on due date, but jobs with tardiness due date will be processed before that one with earliest due date. The dispatch rule SPT (Shortest Processing Time) build job sequence based on duration of processing. The job with shortest processing time will be processed before that one with longest processing time. In the rule WSPT (Weight Shortest Processing Time) the same logic of SPT is used, but a weight assigned to each job according to its priority is used to order the jobs. Finally, in the LPT (Longest Processing Time) rule, the jobs are ordered based on the processing time too, but in this case the jobs with longest processing time will be processed before that one with shortest processing time.

2) GRASP construction procedure: In this construction procedure, an offspring is formed by inserting one gene at a time. The offspring is constructed according to partially greedy selection criteria. To estimate the insertion benefit of each gene, dispatch rules EDD, TDD, SPT, WSPT and LPT were used. Each rule gives a different construction.

In Figure 1, the GRASP construction phase is showed. In this figure, $g_{\min}$ represents the best value according to the dispatch rule and $g_{\max}$, the worst one.

D. The Adaptive Genetic Algorithm applied to SMSETP

Figure 2 shows the pseudo-code of the proposed Adaptive Genetic Algorithm (AGA). In this figure, $ger_{\max}$ represents the maximum numbers of generations, n_{ind} the size of the population, p_c the crossover probability, p_m the mutation probability, $P(t)$ the population at generation t, and *freq*, the frequency of updating the probabilities of crossover operators. The algorithm phases are described following.

```
Algorithm AGA(ger_max, n_ind, p_c, p_m, freq);
1  t ← 0;
2  Generate Initial Population P(t);
3  Evaluate P(t);
4  while (t < ger_max) do
5      t ← t + 1;
6      P(t) ← P(t − 1);
7      i ← 0; { number of new individuals }
8      while (i < n_ind) do
9          Select two individuals from P(t − 1);
10         cross ← Randomly number from 1 to 100;
11         if (cross ≤ p_c ) then
12             Choose a crossover operator O_k;
13             Apply the chosen crossover operator;
14             i ← i + 2;
15             Incorporate the new individuals to P(t);
16         end-if;
17     end-while;
18     Evaluate P(t);
19     Define n_ind survivors;
20     Apply mutation with probability p_m
           in all members of population P(t)
21     if (t mod freq = 0) then
22         Update the probability of selecting
               each crossover operator (p_(O_k));
23         Execute Local Search;
24         Apply Path Relinking;
25     end-if;
26 end-while;
end AGA;
```

Fig. 2. Pseudo-code of the proposed Adaptive Genetic Algorithm

1) Individual selection method: The individuals are selected to reproduction (line 9 of Figure 2) by the tournament selection. In the implemented strategy, only one tournament involving all the population is realized. The winner of the tournament, that is, the one with the best fitness, is selected for crossover with each individual from the population. Therefore, all the individuals are selected to reproduction.

2) Crossover: In order to form a new population, the selected individuals by the selection method above are submitted to one of the following crossover operators: (*i*) One Point Crossover (OX), (*ii*) Similar Job Order Crossover (SJOX), (*iii*) Relative Job Order Crossover (RRX), (*iv*) Based Order Uniform Crossover (BOUX) and (*v*) Partially Mapped Crossover (PMX). This choice was taken by the fact of these

operators being the most common ones to solve scheduling problems by genetic algorithm [6].

In the One Point Crossover operator (OX), a cut point is selected at random. The jobs by the right side of cut point of parents 1 and 2 are copied to the offspring 1 and 2, respectively. The offspring 1 and 2 are completed following the job sequence of parents 2 and 1, respectively.

In the Similar Job Order Crossover (SJOX), the two parents are examined job by job. Case the same job appears in the same position in both parents, it is copied to the both offspring. Next, a cut point is randomly chosen and the missing jobs in the offspring 1 and 2 are copied to the parents 1 and 2 respectively and then the jobs by the left side of cut point are filled according to the job sequence of parents 2 and 1 to form the offspring 1 and 2 respectively.

Proposed by [9] specifically to the job scheduling problem, the Relative Job Order Crossover (RRX) operator not only preserves the order of the jobs of parents but also preserves some jobs on their absolute positions of the sequence. It divides the jobs into two sets and mixes them according to the order of both parents, without random decisions. According to this scheme, exactly eight offspring are generated, being two of them clones of the parents. In this work these two clones are ignored. As a fixed size of population is adopted, it becomes necessary to select some of the individuals generated to incorporate them into the new population. In this work, two of six offspring with the shortest values of objective function are used. The crossover process happens in two phases. In the first phase the parents are divided in two parts according to the cut point randomly selected. The first part of each parent is copied to the offspring. The second phase consists of adding the missing genes to the offspring. These genes are copied of the same position from the parent who did not provide genes to the offspring yet. For example, if an offspring received genes of parent 1 in the first phase, it will be filled with genes of parent 2 in the second phase.

The Based Order Uniform Crossover (BOUX) was developed in order to avoid infeasible solutions are built. According [6], the BOUX is considered one of crossover operators that has the best adherence to the scheduling problems. The crossover procedure starts with a string that has the same size of the parents, which save the values 0 or 1, randomly chosen. In the sequence, the crossover is realized gene by gene, respecting the following rules: Case the bit is equal 0, the offspring 1 receive gene of the same position of parent 1 and the offspring 2 the gene of parent 2. Case the bit is equal 1, the offspring 1 receive the gene of the same position of parent 2 and offspring 2 the gene of parent 1. Before the gene insertion into the new individual, the procedure checks whether the gene is already included in the offspring being generated and, in positive case, the gene to be inserted is the one who is in the same position of the other parent. If the problem persists, the job sequence of parent indicated by the bit is used.

The Partially Mapped Crossover (PMX) executes a mapping of parent fragment into their offspring. The missing genes are obtained from the job sequence of the other parent. In this work two cut points are randomly chosen. The cut point 1 always is shorter than the cut point 2. The choice of these points allows the extraction of a fragment from each parent. The offspring 1 inherits the fragment of parent 2 and the offspring 2, the fragment of parent 1. To maintain the feasibility of the new individuals created, the remaining genes are filled in the order of the other parent (who has not provided the fragment), always verifying if the gene to be inserted is already in the offspring.

The choice probability of crossover operators modifies according to the quality of individuals produced by the operators in the past generations. More specifically, let O_k, with $k = 1, \cdots, 5$, the five *crossover* operators. Initially, each *crossover* operator O_k has the same probability to be chosen, that means, $p(O_k) = 1/5$. Let $f(s^*)$ be the best individual so far and A_k the average value found for each operator O_k since the last update. Case the operator was not chosen in last five generations, make $A_k = 0$. Then, calculate $q_k = f(s^*)/A_k$ and $p(O_k) = q_k / \sum_{j=1}^{5} q_j$ for all $k = 1, \cdots, 5$. Observe that how much better the individual is, more high is the value of q_k and, consequently, the probability of choosing the O_k operator is increased. Therefore, during the algorithm evolution, the best operator have its chance of choice increased. This procedure is inspired in *Reactive GRASP* algorithm, proposed by [10].

3) Local Search: Same as mentioned above, periodically a local search is applied to the best individual generated by each *crossover* operator. The Random Descent is used as local search method. This method uses two types of movement to explore the search space: the exchange of two jobs of the sequence and the job relocation to another production sequence. The method works as follow: two jobs belonging to an individual are selected at random and the positions are exchanged. If the new individual is better than the current one according to the evaluation function, it is accept and pass to be the current individual; otherwise, another movement are randomly chosen. If during $RDM_{\max}$ iterations is not found a better individual than the current one, then relocate movements are used. If there is any improvement in this phase, the method returns to use exchange movements; otherwise, the local search is ended up after $RDM_{\max}$ iterations without improvement.

4) Path Relinking: The path relinking is a procedure that integrates intensification strategies and diversification during the search process [11]. It generate new individuals exploring paths that connect high quality individuals. Given a pair of individuals, the search starts with one of them, called base individual, come to another one called guide individual, adding step by step, guide individuals attributes to base individual. In this problem, during the evolutive process, a group with the best five individuals generated by each *crossover* operator are build. So, at five generations, Path Relinking are triggered taking as base solution the best individual generated by the method and as guide individual each one of the five best individuals generated by each crossover operator. This

procedure is called Truncated Backward Path Relinking, and when 75% of guide individual added to the base solution, procedure are stopped. It was considered as attribute a job position of production sequence. For each job candidate to insertion, a local search method like described previously is applied, and a movement of a fixed job is not allowed.

E. Individual survival

The survivors are certain by the elitism technique where individuals more adapted will survive.

F. Stop criteria

The maximum number of generations is used to stop the adaptive genetic algorithm.

G. Variants of proposed algorithm

In this work, three variants of AGA were developed.

In variant 1, called AGA1, the refresh of the crossover operator selection rate happens at each five generations, that is, *freq* = 5 in Fig. 2. After that, all elite group members are submitted to local search (see local search at page 4). Next, these refined solutions are submitted to the path relinking (see path relinking at page 4). In this variant, the elite group are composed by the best individual produced by each crossover operator in last five generations.

Variant 2, called AGA2, differs of the AGA1 variant by elite group composition. In this variant, elite group is composed by the best individual produced by each crossover operator throughout the search, that is, globally and not just at last five generations.

In Variant 3, called AGA3, the refresh of crossover operator selection rate, the application of local search to the members of the elite group, as well as the use of path relinking happens at each ten generations (*freq* = 5 in Fig. 2). The elite group is also composed by 5 individuals. The first three individuals are the best one produced throughout the search. The fourth individual is the best one produced in the past ten generations, if this individual differs from the others by at least 30%, according to the diversity index. Case the individual does not fit in this criteria, the second best solution is analyzed and so on until one of them meets this criteria. The fifth member is chosen by randomly selection of an individual from a set of the best ten individuals produced over past ten generations. To compute the diversity index of two individuals, we sum the number of different genes in a same position and divide by the total number of genes of the individual. For example, the diversity index of the individuals $v_1 = \{3, 1, 4, 5, 2\}$ and $v_2 = \{3, 5, 4, 1, 2\}$ is 40%, because in the second and fourth positions the genes are different and the individuals have 5 genes.

V. Computational tests

The proposed algorithm was developed in C++ language, using Borland C++ Builder 5.0 compiler. The parameters used were obtained experimentally and they are presented in Table I.

TABLE I
ADAPTIVE GENETIC ALGORITHM PARAMETERS

Parameters	Values
Parameter γ of the GRASP construction phase	0.20
Maximum number of iterations of Random Descent ($RDM_{\max}$)	$7 \times n$
Maximum number of generations of AGA ($ger_{\max}$)	100
Crossover probability (p_c)	80%
Mutation probability (p_m)	5%

Three set of instances are used to test the three variants of AGA. The first one is the same of [12]. These authors generated instances randomly with job number equal to 6, 7, 8, 9, 10, 11, 12, 15, 20, 25, 30, 35, 40, 50 and 75, using the same parameters of [2], [13] and [14]. In this set, there are 12 instances for each number of jobs and the setup times are symmetric, that is, $S_{ij} = S_{ji}$. The second one is the same of [8] and it was generated based on works from [15] and [2] with job number equal to 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 and 20. In this second set, the setup times satisfy the triangular inequality, that is, $S_{ik} \leq S_{ij} + S_{jk} + p_j$. The third set is the same of [8] and there are instances with number of jobs equal to 6, 7, 8, 9, 10, 11, 12, 15, 20, 25, 30, 35, 40, 50, 75 and 100. In the last set, no particular property is satisfied. There are 16 instances for each number of jobs of the second and the third set.

These instances are available for download at http://www.iceb.ufop.br/decom/prof/marcone/projects/scheduling/instances.htm

All of experiments were executed in a Pentium Core 2 Duo 2.1 GHz computer with 4 GB of RAM and Windows Vista operational system. The variants of AGA (AGA1, AGA2 and AGA3) were tested 30 times for each instance. The description, details and results of each one of the experiments are described in the following subsections.

A. Results using the first set of instances

The Table II shows the results encountered using the first set of instances. The first column shows the number of jobs. The second, third and fourth columns show how much the average of solutions of each variant differ from the best solution known. In the fifth, sixth and seventh columns are showed how much the best solutions generated differ from the best solution known. In the eighth, ninth and tenth columns the average processing time of the algorithm is showed.

Table III compares AGA1 with the algorithm GTSPR, proposed by [16]. In this table, "% Improv." indicates how much AGA1 improves the solutions produced by GTSPR with relation to the average deviation (or to the best deviation).

B. Results using the second set of instances

Table IV shows the results encountered by the variants of AGA in the second set of instances. The notation used is the same as the previous case.

Table V compares the performance of AGA1 with the CPLEX solver, version 10, applied to the Time-Indexed

TABLE II

RESULTS OF AGA VARIANTS IN THE FIRST SET OF INSTANCES

#	*Deviation* of avg			*Deviation* of best solution			Time (s)		
Jobs	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.94	0.93	0.70
9	0.15%	0.16%	0.24%	0.00%	0.00%	0.00%	1.26	1.25	0.78
10	0.24%	0.25%	0.41%	0.00%	0.00%	0.00%	1.6	1.59	0.99
11	0.03%	0.05%	0.10%	0.00%	0.00%	0.00%	2.21	2.20	1.64
12	0.07%	0.08%	0.21%	0.00%	0.00%	0.00%	2.81	2.80	2.44
15	0.76%	0.80%	1.16%	0.00%	0.00%	0.00%	6.02	5.98	6.09
20	0.73%	0.75%	0.85%	0.00%	0.00%	0.00%	20.6	20.47	17.87
25	1.02%	1.08%	1.42%	0.00%	0.00%	0.00%	45.72	45.44	39.65
30	1.60%	1.82%	2.64%	0.00%	0.00%	0.00%	112.06	111.38	41.65
40	2.33%	2.34%	3.56%	0.08%	-0.08%	-0.09%	335.88	333.81	41.61
50	4.06%	4.37%	6.32%	0.02%	-0.31%	-1.11%	896.1	890.60	222.04
75	6.52%	9.48%	11.86%	0.04%	-4.40%	-1.76%	2000.05	1992.73	1242.06
Avg	**1.46%**	**1.77%**	**2.40%**	**0.011%**	**-0.399%**	**-0.246%**	**285.85**	**284.10**	**134.79**

TABLE III

RESULTS OBTAINED BY AGA1 AND GTPRS IN THE FIRST SET

#	*Deviation* of average			*Deviation* of best solution			Time (s)	
Jobs	AGA 1	GTPRS	% Improv.	AGA 1	GTPRS	% Improv.	AGA 1	GTPRS
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.94	0.06
9	0.15%	0.00%	-15.00%	0.00%	0.00%	0.00%	1.26	0.09
10	0.24%	0.00%	-24.00%	0.00%	0.00%	0.00%	1.60	0.15
11	0.03%	0.00%	-3.00%	0.00%	0.00%	0.00%	2.21	0.25
12	0.07%	0.00%	-7.00%	0.00%	0.00%	0.00%	2.81	0.36
15	0.76%	1.25%	64.08%	0.00%	0.00%	0.00%	6.02	0.46
20	0.73%	1.11%	51.87%	0.00%	0.00%	0.00%	20.60	2.05
25	1.02%	1.60%	56.53%	0.00%	0.00%	0.00%	45.72	6.62
30	1.60%	2.57%	60.61%	0.00%	0.00%	0.00%	112.06	18.66
40	2.33%	3.77%	61.84%	0.08%	0.00%	8.00%	335.88	84.16
50	4.06%	5.58%	37.64%	0.02%	0.00%	2.00%	896.10	305.28
75	6.52%	7.41%	13.69%	0.04%	0.00%	4.00%	2005.05	3.472.26
Avg	**1.46%**	**2.01%**	**2.27%**	**0.011%**	**-0.40%**	**-0.25%**	**285.85**	**324.20**

formulation of Mixed-Integer Programming (MIP-TI) proposed by [8]. A signal "-" indicates that CPLEX was not able to solve the corresponding instances because there were memory overflow.

C. Results using the third set of instances

Table VI shows the results reached by the variants of AGA in the third set of instances. The notation used is also the same as the previous case.

D. Considerations about the experiments

Three variants of the proposed AGA were submitted to three set of experiments.

Tables II, III, IV, V and VI compare the performance of the variants of AGA with the different solution methods proposed in the literature.

In the first set of experiments, AGA1 was compared with GTSPR, an algorithm developed by [16]. By the results encountered, we can see that this AGA variant was more efficient than GTSPR, because the variability of average results is lower and AGA1 produced final solutions with better quality.

In the second set of experiments, AGA1 was compared with the results reached by an optimizer applied to a mathematic formulation proposed by [8]. Results shows that AGA1 is more efficient because it reached all of the best solutions in lower computational time without deviations.

In the third set of experiments, the three variants of AGA were compared. It can be verified that AGA1 had the best performance since it produced the best solutions and the smallest average deviations for the solutions. These results were not compared with others from the literature because we did not find any algorithm that uses this data set.

VI. CONCLUSIONS

This paper treated the single machine scheduling problem with earliness and tardiness penalties, considering distinct due windows and sequence-dependent setup time. To solve this problem an adaptive genetic algorithm was proposed, where the initial population was generated by a procedure GRASP, using as a guide function dispatch rules EDD (*Earliest Due Date*), TDD (*Tardiness Due Date*), SPT (*Shortest Processing Time*), WSPT (*Weight Shortest Processing Time*) e LPT (*Longest Processing Time*). During the evaluation process, population pass through selection, crossover and mutation process. In crossover, five operators, OX (*One Point Crossover*), SJOX (*Similar Job Order Crossover*), BOUX (*Based Order Uniform Crossover*), PMX (*Partially Mapped Crossover*) e RRX (*Relative Job Order Crossover*), are used, being the best solutions produced by each operator submitted

TABLE IV

RESULTS OBTAINED BY AGA1, AGA2 AND AGA3 IN THE SECOND SET

#	*Deviation* of average			*Deviation* of best solution			Time (s)		
Jobs	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3
6	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.65	0.71	0.65
7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.86	1.33	0.82
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.13	1.22	1.01
9	0.00%	0.01%	0.00%	0.00%	0.01%	0.00%	1.47	1.76	1.26
10	0.00%	0.00%	0.03%	0.00%	0.00%	0.03%	1.78	2.21	1.53
11	0.00%	0.07%	0.12%	0.00%	0.07%	0.12%	2.37	2.81	1.91
12	0.00%	0.03%	0.04%	0.00%	0.03%	0.04%	3.88	3.80	2.44
13	0.00%	0.10%	0.20%	0.00%	0.10%	0.20%	4.87	5.37	3.09
14	0.00%	0.06%	0.14%	0.00%	0.06%	0.14%	5.75	6.14	3.47
15	0.02%	0.19%	0.27%	0.19%	0.19%	0.27%	8.50	7.99	4.43
16	0.06%	0.20%	0.20%	0.06%	0.20%	0.20%	11.64	8.97	2.69
17	0.00%	0.31%	0.31%	0.00%	0.31%	0.31%	0.61	10.88	3.26
18	0.00%	0.42%	0.42%	0.00%	0.42%	0.42%	0.61	15.23	4.57
19	0.04%	0.23%	0.23%	0.04%	0.23%	0.25%	18.17	19.09	5.73
20	0.02%	0.43%	0.44%	0.02%	0.43%	0.44%	30.32	17.63	10.95
Avg	**0.01%**	**0.14%**	**0.16%**	**0.02%**	**0.14%**	**0.16%**	**6.17**	**7.01**	**3.19**

TABLE V

COMPARING AGA1 WITH MIP-TI

#	*Deviation* of average		*Deviation* of the best		Time (s)	
Jobs	AGA 1	MIP-TI	AGA 1	MIP-TI	AGA 1	MIP-TI
6	0.00%	0.00%	0.00%	0.00%	0.65	8.77
7	0.00%	0.00%	0.00%	0.00%	0.86	20.96
8	0.00%	0.00%	0.00%	0.00%	1.13	34.67
9	0.00%	0.00%	0.00%	0.00%	1.47	79.24
10	0.00%	0.00%	0.00%	0.00%	1.78	196.17
11	0.00%	0.00%	0.00%	0.00%	2.37	453.88
12	0.00%	0.38%	0.00%	6.25%	3.88	845.25
13	0.00%	-	0.00%	-	4.87	-
14	0.00%	-	0.00%	-	5.75	-
15	0.02%	-	0.19%	-	8.50	-
16	0.06%	-	0.06%	-	11.64	-
17	0.00%	-	0.00%	-	0.61	-
18	0.00%	-	0.00%	-	0.61	-
19	0.04%	-	0.04%	-	18.17	-
20	0.02%	-	0.02%	-	30.32	-

TABLE VI

RESULTS OBTAINED BY AGA1, AGA2 AND AGA3 IN THE THIRD SET OF INSTANCES

#	*Deviation* of average			*Deviation* of the best			Time (s)		
Jobs	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3	AGA 1	AGA 2	AGA 3
6	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.76	0.61	0.65
7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.76	0.94	0.81
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.03	1.21	1.00
9	0.00%	0.13%	0.20%	0.00%	0.13%	0.20%	1.35	1.53	1.26
10	0.00%	0.03%	0.06%	0.00%	0.03%	0.06%	1.61	1.88	1.65
11	0.00%	0.11%	0.17%	5.97%	0.11%	6.16%	2.20	0.73	1.92
12	0.00%	0.01%	0.11%	0.00%	0.01%	0.11%	3.32	3.09	2.37
15	0.01%	0.14%	0.37%	0.01%	0.14%	0.37%	7.49	5.60	4.38
20	0.30%	0.64%	0.81%	0.30%	0.64%	0.81%	23.88	10.18	10.61
30	1.20%	1.19%	1.35%	1.36%	1.51%	1.50%	172.51	64.19	47.13
40	0.98%	1.21%	0.98%	1.03%	1.42%	2.34%	801.67	155.05	98.05
50	1.14%	1.46%	1.14%	1.26%	2.24%	2.58%	1575.11	529.11	416.78
75	0.00%	1.26%	2.36%	0.09%	1.67%	3.60%	4978.02	3381.58	1398.82
100	0.25%	2.50%	1.10%	0.25%	3.14%	2.41%	18107.72	23209.42	15853.97
Average	**0.28%**	**0.62%**	**0.62%**	**0.73%**	**0.79%**	**1.44%**	**1834.10**	**1954.82**	**1274.24**

to local search and path relinking procedures. The path relinking procedure connects the best produced solution with each best solution produced by each operator.

By the end, three set of instances were used to test the proposed algorithm, and three variants of AGA were developed. The results achieved by the proposed algorithm were compared with those produced by other algorithms from the literature. In these experiments, the proposed algorithm

presented high quality solutions with lower GAP, always reaching the best known value. The developed algorithm produced solutions better than the best solutions found from the literature, besides presenting a minor variability of final solutions, which shows its robustness.

ACKNOWLEDGMENT

The authors would like to thank CEFET/MG, CAPES, CNPq and FAPEMIG for supporting the development of this research.

REFERENCES

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

[2] G. Wan and B. P. C. Yen, "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, 2002.

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

[4] C. M. Hino, D. P. Ronconi, and A. B. Mendes, "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, 2005.

[5] K. C. Ying, "Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm," *Computers and Industrial Engineering*, vol. 55, no. 2, pp. 494–502, 2008.

[6] C. Y. Lee and J. Y. Choi, "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, 1995.

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

[8] B. F. Rosa, M. J. F. Souza, and S. R. Souza, "Formulations of mathematical programming for the the single machine scheduling with distinct time windows and sequence-dependent setup time (in portuguese)," in *Proceedings of the XXXII Congresso Nacional de Matemática Aplicada e Computacional – CNMAC 2009*, Cuiabá, Brazil, 2009.

[9] P. A. Rubin and G. L. Ragatz, "Scheduling in a sequence dependent setup environment with genetic search," *Computers and Operations Research*, vol. 22, pp. 85–89, 1995.

[10] M. Prais and C. C. Ribeiro, "An application to a matrix decomposition problem in tdma traffic assignment," *INFORMS - Journal on Computing*, vol. 12, pp. 164–176, 2000.

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

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

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

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

[15] G. Rabadi, M. Mollaghasemi, and G. C. Anagnostopoulos, "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, 2004.

[16] P. H. V. Penna, "A hybrid heuristic algorithm for minimizing the costs with earliness and tardiness in environments with time windows and setup times dependent of the production sequence (in portuguese)," Master's Thesis, Mineral Engineering Program, Federal University of Ouro Preto, Ouro Preto, Brazil, 2009.