Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

An Adaptive Genetic Algorithm for solving the Single Machine Scheduling Problem with Earliness and Tardiness Penalties

Fabio Fernandes Ribeiro, Sergio Ricardo de Souza
DPPG, CEFET/MG
Belo Horizonte, Brazil
fabiobh@gmail.com, sergio@dppg.cefetmg.br

Marcone Jamilson Freitas Souza
Federal University of Ouro Preto
Ouro Preto, Brazil
marcone@iceb.ufop.br

***Abstract*—This paper deals with the Single Machine Scheduling Problem with Earliness and Tardiness Penalties, considering distinct time windows and sequence-dependent setup time. Due to the complexity of this problem, an adaptive genetic algorithm is proposed for solving it. Many search operators are used to explore the solution space where the choice probability for each operator depends on the success in a previous search. The initial population is generated by applying GRASP to five dispatch rules. For each individual generated, a polynomial time algorithm is used to determine the initial optimal processing date for each job. During the evaluation process, the best individuals produced by each crossover operator, in each generation undergo refinement in order to improve quality of individuals. Computational results show the effectiveness of the proposed algorithm.**

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

I. Introduction

Today, scheduling is one of the most studied problem types [1]. There are two main aspects: the first concerns their practical importance, with various applications in several industries, such as chemical, metallurgic and textile industries. The second aspect deals with the difficulty of solving most of the problems of this class. This paper deals with the Single Machine Scheduling Problem with Earliness and Tardiness Penalties (SMSPETP) with distinct time windows and sequence-dependent setup time. To our knowledge, this problem has not yet been the object of much attention from the scientific community, as can be seen in a recent survey [1].

The lateness penalty criteria and earliness production shares the goal of the Just-in-Time philosophy, in which they produce only when it is called for. There are time windows for each job, according to [2], to deal with uncertain situation or tolerance for due date. We suppose that these time interval operations can be solved without costs. On the other hand, in the majority of industrial processes, machines can be prepared to do new jobs, including the to clean up, set up, obtain tools, adjust tools, position and inspect materials that will be used in the process. The preparation time needed for this is known as setup time. Many production scheduling studies disregard this time or include it in the operation processing time. This act simplifies the analysis but affects the quality of the solution when setup time varies relevantly in function of machine job sequence. This work considers setup time to be dependent on production scheduling. Since it was showed in [3] that a simplified version of this problem is NP-Hard, using metaheuristics to solve this problem is justified.

Simplified versions of this problem have been studied by various authors. [4] uses Tabu Search and Genetic Algorithms for solving the problem with common delivery date. This algorithm uses optimal solution properties to explore the solution space. [5] solves this problem considering common delivery dates and setup time included in a job processing time and independent of production sequence. The author uses Recovering Beam Search (RBS), an improved version of the Beam Search (BS) method, a branch-and-bound algorithm at which only the most promising w nodes of each search tree level are selected for a future ramification, while the rest of nodes are cut forever. In order to avoid eliminating the wrong decision about which nodes are followed, RBS Algorithm uses a recovered tool that searches for better partial solutions which control those selected previously. [6] used genetic algorithm to solve the problem with different delivery dates. In this last work, a specific algorithm, with polynomial complexity, was developed to determine the optimal process conclusion date for each job in a sequence produced by Genetic Algorithm. This algorithm is necessary because finishing a job early can be attractive even paying a penalty, if the penalty is less than the penalty generated by the lateness. [2] studied this problem considering separate time windows, in place of due dates. The algorithm of [6] was adapted 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 is proposed here in which the initial population was generated by a construction method based on GRASP [7] and five dispatch rules. During the evaluation process, the population passes through conventional mutation and crossover process. However, the crossover uses criteria based on solution quality generated by each crossover operator to choose which operator will be used. In addition, in a determined number of generations, the probability an operator being chosen is update according to the solution quality for each operator. A local search is then

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

applied in the best produced offspring, for each operator, to refine them. The survival population is composed of 95% offspring chosen by the elitism procedure. The other 5% are randomly chosen and pass through the mutation process, in which two jobs will be switched to warranty population diversity. Path Relinking is applied periodically by connecting the best individual found so far with the best offspring generated in the last 5 generations. Population improvement occurs until the stop criterion is reached.

This work will be carried out using the following structure: section 2 details the studied problem; section 3 presents the adaptive algorithm for solving SMSETP; section 4 shows and discuss the results. Finally, section 5 ends the work and shows future opportunities to improve the proposed algorithm.

II. Problem description

The sequencing problem studied in this work is single machine scheduling with earliness and tardiness penalties with distinct time 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 E_i and the final date T_i desired to the end of processing;
- the machine executes one job per time and if the job processing is started, the job must be finished and processing interruptions are not allowed;
- All the jobs are available to processing in the time 0;
- When the job j is sequenced immediately after a job i, if they are part of different products family, a setup time S_{ij} is necessary to set up the machine. Setup time equal 0 ($S_{ij} = 0$) means products of the same family. This means initial setup time considers setup time to the first job in the sequence to be 0;
- Idle time between execution of two consecutive jobs is allowed;
- Jobs must be finalized inside the time interval $[E_i, T_i]$, called time window. In case of job finalization before the E_i, so there is a cost for early arrival. In Case the job are finalized after T_i, a cost will be generated for tardiness. Jobs finalized within the time window have no cost;
- The costs for earliness and tardiness of production depends on the job, each job i has a earliness cost α_i and a tardiness cost β_i;
- The objective is to minimize the summation of the earliness and tardiness penalties.

A. The mixed integer programming model

The mixed integer programming model (MIP) below, based on [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 follows all other jobs;

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

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

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

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

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

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

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

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

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

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

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

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. Constraints (3), (4) and (11) guarantee that the variable y_{ij} assumes value 1 if and only if job j is processed immediately after job i. Constraints (5) and (6) define, respectively, tardiness and earliness values according of time window. Constraints (7) to (11) define the type of variables.

B. Heuristic framework

In this section, the Adaptive Genetic Algorithm (AGA) framework will be describe.

C. 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 sequence $v = \{7, 1, 5, 6, 4, 3, 2\}$, job 7 is the first to be processed and job 2, the last.

D. Evaluation of individuals

All individuals are evaluated by the same objective function, given by MIP model expression (1), in which the individual who obtained the shortest value to the objective function are considerer the best adapted.

E. Initial population construction

The initial population of the adaptive genetic algorithm proposed is generated by GRASP construction phase [7], having as guide function five dispatch rules: EDD (Earliest Due

Date), TDD (Tardiness due date), SPT (Shortest processing time), WSPT (Weight shortest processing time) and LPT (Longest processing time). For each construction (GRASP + Dispatch rule) 200 individuals are generated. Then individuals are ordered, from the best to the worst, according to the evaluated function. The Initial population is made up of the best 100 individuals generated.

1) Dispatch rules: In the EDD dispatch rule, jobs are ordered by the initial date E_i. Jobs with earlier due date will be processed before those one with later due date. By the TDD dispatch rule, jobs are ordered based on the final date T_i. Jobs with later due date will be processed before those with earlier due dates. The SPT dispatch rule builds job sequence based on processing duration. A job with a shorter processing time will be processing before one with a longer processing time. In the rule WSPT the same SPT logic is used but a weight is assigned to each job by service priority which is considered to order the jobs. Finally the LPT rule puts the jobs in order also based on processing time, but in this rule the jobs with longer processing times will be processed before those with shorter processing times.

2) GRASP construction procedure: In this construction procedure, an offspring is formed by genes that have been inserted one by one. The offspring was constructed according to partially greedy selection criteria. To estimate each gene's insertion benefit, dispatch rules EDD, TDD, SPT, WSPT and LPT were used. Each variant gives a different construction.

In the sequence, the selection criteria based on the EDD rule, in which jobs are listed by earliest due date. At each iteration of the procedure, the next candidate's genes to be included in the individual were put in a candidate list, according to the EDD rule. The best candidates are put in a Restricted Candidate List (RCL), with size controlled by parameter $\gamma \in [0, 1]$.

The procedure includes in RCL all of genes (jobs) i which dates E_i is less than or equals to $E_{min} + \gamma \times (E_{max} - E_{min})$. Then one gene of this list is randomly chosen and added to the offspring under construction. The procedure ends when all of genes have been allocated, when the offspring is completely constructed. Empirically, 10,000 individuals were generated for each one of the following values: 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20. The results shown on Table I are generated by an instance involving 20 jobs and five dispatch rules, chosen randomly at each iteration.

The "Best Result" column on table I shows the best value found for each γ parameter. The column "Average" shows the average value for the 1,000 individuals generated and the column "GAP" shows the rate variations between the average and the best result. Based on this data γ parameter set at 0.20.

In the Figure 1, the GRASP construction phase is shown. In this figure, g_{min} represents the best value for the dispatch rule adopted and g_{max}, the worst one.

F. The Adaptive Genetic Algorithm applied to SMSETP

Figure 2 shows the pseudo-code of the proposed Adaptive Genetic Algorithm (AGA). The algorithm phases are described bellow.

TABLE I
RESULTS FOR γ PARAMETER

γ	Best result	Average	*gap* (%)
0.02	154,150	163,582	6.12
0.04	154,150	163,645	6.16
0.06	131,063	173,691	32.52
0.08	120,521	173,101	43.63
0.10	120,149	181,906	51.40
0.12	120,521	187,383	55.48
0.14	115,283	183,307	59.01
0.16	118,838	183,310	54.17
0.18	118,991	196,007	64.72
0.20	114,256	191,689	67.77

procedure $Construction(g(.), \gamma, v)$;
1 $v \leftarrow \emptyset$;
2 Initialize a set C of candidate genes;
3 while $(C \neq \emptyset)$ do
4 $\quad g_{min} = \min\{g(t) \mid t \in C\}$;
5 $\quad g_{max} = \max\{g(t) \mid t \in C\}$;
6 $\quad RCL = \{t \in C \mid g(t) \leq g_{min} + \gamma \cdot (g_{max} - g_{min})\}$;
7 $\quad$ Select, randomly, a gene $t \in RCL$;
8 $\quad v \leftarrow v \cup \{t\}$;
9 $\quad$ Update C;
10 end-while;
11 Return v;
end $Construction$;

Fig. 1. Procedure to build an individual

1) Individual selection method: After evaluating the population, individuals are selected by the binary tournament method in which the main goal is to allow the best adapted individuals to be selected.

2) Crossover: After evaluating the population, individuals are selected for reproduction by the selection method described above. The crossover process uses the following 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 justified because these operators are the operators that most commonly solve problems like this by genetic algorithm [6]. The One Point Crossover operators (OX) select a cut point randomly. Jobs on the right side of relative 1 and 2's cut points are copied to offspring 1 and 2, respectively. Offspring 1 and 2 are produced following the job sequence of relatives 2 and 1, respectively. In the Similar Job Order Crossover (SJOX), the two parents are examined job by job. Should the same job appear in the same position in both relatives, it is copied to the both offspring. In the sequence, a cut point is randomly chosen and the missing jobs in the offspring 1 and 2 are copied to the relatives 1 and 2 respectively and then the jobs by the left side of cut point are completed following the job sequence of relatives 2 and 1 to the offspring 1 and 2 respectively. Proposed by [9] specifically to the job Schedule problem, the Relative Job Order Crossover (RRX) preserves the relative job order and even preserves some jobs in the absolute positions in the sequence. It divides the

Algorithm *AGA*(*maxger, nind, probcross, probmut*);
1 $t \leftarrow 0$;
2 Generate Initial Population $P(t)$;
3 Evaluate $P(t)$;
4 while ($t \leq maxger$) do
5 $t \leftarrow t + 1$;
6 Generate $P(t)$ by $P(t-1)$;
7 while ($i \leq$ *numind*) do
8 $i \leftarrow 1$;
9 *cross* $\leftarrow$ Randomly number from 1 to 100;
10 if (*cross* $\leq$ *probcrossover*) then
11 Select individual;
12 Crossover;
13 end-if;
14 Evaluate $P(t)$;
15 end-while;
16 Define survivors;
17 if ($t \mod 5 = 0$) then
18 Update choose probability of operators ($p_{(O_i)}$);
19 Execute Local Search;
20 Apply Path Relinking;
21 end-if;
22 end-while;
end *AGA*;

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

jobs in two sets and mixes them by order of both relatives, without random decisions. In to this scheme, exactly eight offspring are generated, two being relative clones. In the RRX operator implemented in this work the two clones will be ignored. As the fixed population size is adopted, the selection of some offspring will be necessary to add them to the population. In this work, two of six offspring with the shortest objective function value will be used. The crossover process has two phases: In the first phase the relatives will be divided into two parts according to randomly selected cut point. The first part of each relative is copied to the offspring. The second phase is adding the missing genes to the offspring. These genes are copied according to the sequence of relative genes that have not given genes to an offspring yet. For example, if an offspring receives genes from relative 1 in this first phase, it will be completed with genes of relative 2. The Based Order Uniform Crossover (BOUX) was developed with in order to avoid construction of non-feasible solutions. According [6], the BOUX is considered one of crossover operator that has the best adherence to the schedule problems. The crossover procedure started with a string construction that has the same relative size, and save the values 0 or 1, randomly chose. Next the crossover is carried out gene by gene, respecting the following rules: Should the bit equal 0, offspring 1 receives a gene that is in the same position in relative 1 and offspring 2 receives relative 2's gene. If the bit equals 1, offspring 1 receives the gene from the same position in relative 2 and offspring 2 receives the gene from relative 1. Before the gene is inserted in the sequence, the procedure checks if the gene is already part of the offspring and if it is, the gene inserted is from the same position of another relative. If the problem persists, the job sequence of the relative indicated by the bit is used. The Partially Mapped Crossover (PMX) executes the relative fragment map of this offspring. The missing genes are copied following the job sequence of another relative. In this work two cut points are randomly chosen. Cut point 1 is always shorter than cut point 2. The choice of these points allowed the extraction of each relative fragment so that offspring 1 gives relative 2 the fragment and offspring 2 gives relative 1 the fragment. To keep the offspring viable, missing genes are filled following job order of the relative that doesn't contribute with a fragment, if the gene to be inserted is already in the offspring.

Crossover choice probability of an operator changes according to the quality of individuals produced by operators in past generations. More specifically, let O_i, with $i = 1, \cdots, 5$, the five *crossover* operators. Initially, each *crossover* operator O_i has the same probability of being chosen, which means, $p(O_i) = 1/5$. Let $f(s^*)$ be the best individual found and A_i the average individual value found for each operator O_i since the last update. If the operator has not been chosen in last five generations, make $A_i = 1$. Let $q_i = f(s^*) = A_i$ and $p(O_i) = q_i / \sum_{j=1}^{5} q_j$ where $i = 1, \cdots, 5$. Observe that the better the individual are, the higher the q_i value and consequently, the better the probability $p(O_i)$ has of choosing operator O_i. Therefore, during the evolution of the algorithm, the best operator has a better chance of being chosen. This procedure is inspired by *Reactive GRASP* algorithm, proposed by [10].

3) Local search: As stated previously, every five generations a local search is applied for the best individual generated by each *crossover* operator. The local search used is Random Descending, which uses two kinds of movement to explore a search space: the switch of two jobs of the sequence and job relocation to another production sequence. The method works as follows: For an individual, two jobs are randomly selected and the positions are exchanged. If the new movement is better, according to the evaluation function, it is accepted and becomes the current solution; otherwise, another movement is randomly chosen. If during *MRDmax* any solution better than the current is generated, relocation movement is used. If there is improvement, the method goes back to using exchange movements; otherwise, local search ends when *MRDmax* iterations pass without improvement.

4) Path Relinking: Path Relinking is a procedure that integrates intensification and diversification strategies during the search process [11]. It generates new individuals by exploring paths that connect high quality individuals. Given a pair of individuals, a search starts with one of them, called the base individual,moves on to another called guide individual, step by step adding guide individual attributes to the base individual. In this problem, during the evolution process, a group of the best five individuals generated by each *crossover* operator are build. So, after each five generations, Path Relinking is triggered

using the best individual generated by AGA as base solution and the best individual generated by each crossover operator as guide individual. This procedure is called Truncated Backward Path Relinking, and when 75% of guide individual has been added to the base solution, the procedure is stopped. It considers production sequence job position as an attribute. For each job candidate inserted, a local search method is applied as previously described, and moving candidate jobs is not allowed.

G. Individual survival

Individual survival is certain by the elitism technique. 95% of the best adapted individuals will survive and the others 5% is made up of individuals chosen randomly from the current population and submitted to mutation in which the production sequence of two jobs are exchanged.

H. Stop criteria

The maximum number of generations is used as a stopping criteria of the adaptive genetic algorithm.

III. Computational tests

The proposed algorithm was developed in C++ language, using Borland C++ Builder 5.0 compiler. The parameters used have been obtained experimentally and are presented on table II.

TABLE II
ADAPTIVE GENETIC ALGORITHM PARAMETERS

Parameters	Values
Parameter γ of GRASP construction phase	0.20
Maximum iterations of local search (*MRDmax*)	$7 \times n$
Maximum generations of AGA (*maxger*)	100
Crossover probability	80%
Mutation rate	5%

The instances used was generated by random pseudo-method based on works from [2], [12] and [13], with jobs numbers equal to 8, 9, 10, 11, 12, 15, 20, 25, 30, 35, 40, 50 and 75. In these instances, processing time P_i of jobs was uniformly and discretely distributed between 1 and 100. Time window centers were uniformly and discretely distributed in interval $[(1-TF-RDD/2)/MS, (1-TF+RDD/2)/MS]$, where *MS* is the total processing time of all jobs, *TF* the delay factor and *RDD* the relative variation of the window date. Values considered for *TF* were 0.1, 0.2, 0.3 and 0.4 while for *RDD*, the values were 0.8, 1.0, and 1.2. The window date size was uniformly and discretely distributed in interval [1, *MS*/n]. Late production cost (β_i) was uniformly and discretely distributed in the interval [20, 100]. As the majority of real cases, production lateness is less desirable than earliness, the costs for production earliness (α_i) generated k times of the cost of the same job, k being a random real number in the interval [0, 1]. Setup time (S_{ij}) was generated uniformly and discretely in the interval [0, 50]. In this data setup time was symmetrical, that is, $S_{ij} = S_{ji}$.

Since *TF* has 4 different values and *RDD* 3 different values, there are 12 instances for each job number, giving 144 total instances. All experiments were carried out on a Pentium Core 2 Duo 2.1 GHz computer with 4 GB RAM and running the Windows Vista operating system.

Each instance was solved 30 times by the proposed algorithm, with the exception of the instance involving 75 jobs, which was solved 20 times. For instances involving 75 jobs, *MRDmax* $= 2 \times n$ are used. Table III shows the results obtained by the proposed method and reproduce results obtained by [8]. The first column shows the number of jobs involved. The second and third columns present how much the average individuals of both algorithms (AGA and [8], respectively) varied from the best individual known for each instance. The last two columns presents how much the best individuals generated by both algorithm differed from the best individual known for each instance. GAP is calculated by the expression $GAP = (RMed - MR)/MR$, in which *RMed* is average result obtained by applying of each algorithm and *MR* is the best known individual of each instance.

TABLE III
ADAPTIVE GENETIC ALGORITHM RESULTS

#	*GAP* of average solution		*GAP* of the best solution	
Jobs	AGA	Gomes Jr.	AGA	Gomes Jr.
8	0.00	0.03	0.00	0.00
9	0.15	0.06	0.00	0.00
10	0.24	0.02	0.00	0.00
11	0.03	0.12	0.00	0.00
12	0.07	0.21	0.00	0.00
15	0.76	1.47	0.00	0.00
20	0.73	1.65	0.00	0.00
25	1.02	2.32	0.00	0.00
30	1.60	3.34	0.00	0.20
40	2.15	4.38	-0.25	0.18
50	3.72	6.13	-0.60	0.28
75	4.59	10.89	-1.48	0.56

As can be seen, in instances involving 11 jobs or more, the average solutions of AGA was always be less than [8]. This fact shows the robustness of the proposed algorithm, since it produces final solutions with less variability. On the other hand, AGA also is able to generate the best known solutions. In fact, AGA generated all the best known solutions to instances involving up to 30 jobs, while [8] was not always capable of this, presenting a GAP of 0.20%. Finally, for 40, 50 and 75-job instances, AGA was able to produce individuals better than the best individuals known in literature.

IV. Conclusion

This paper dealt with 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 GRASP procedure, using EDD (*Earliest Due Date*), TDD (*Tardiness Due Date*), SPT (*Shortest Processing Time*), WSPT (*Weight Shortest Processing Time*) e LPT (*Longest Processing Time*) as guide function dispatch rules. During the evaluation process, population undergo 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, with the best solutions produced by each operator are submitted to local search and path relinking. The path relinking procedure connect the best solution produced to each best solutions produced by each operator.

Finally, instances are used to test the algorithm proposed, comparing with other algorithm from the literature. In instances involving up to 30 jobs, the proposed algorithm has high quality solutions with lower GAP, always reaching the best known value. In instances involving more than 40 jobs, the algorithm developed presents solutions better than the best solutions found in the literature, as well as having less variation of final solutions, showing robustness.

ACKNOWLEDGMENT

The authors would like to thank CEFET/MG, CAPES and FAPERJ for the support to development of this work.

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] 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 scheduling problem with earliness and tardiness penalties (in portuguese)," in *Proceedings of the XXXIX Brazilian Symposium of Operational Research (XXXIX SBPO)*, Fortaleza, Brazil, 2007, pp. 1649–1660.

[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, "Reactive grasp: 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] 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.

[13] 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.