Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

A Genetic Algorithm with Injecting Artificial Chromosomes for Single Machine Scheduling Problems

Pei-Chann Chang, Shih-Shin Chen, Qiong-Hui Ko, Chin-Yuan Fan.

***Abstract*— A genetic algorithm with injecting artificial chromosomes was developed for solving the single machine scheduling problems with the objective to minimize the total deviation. Artificial chromosomes are generated according to a probability matrix which was transformed from the gene structure of an elite base. A roulette wheel selection method was applied to generate an artificial chromosome by assigning genes onto each position according to the probability matrix. The higher the probability is, the more possible that the gene will show up in that particular position. By injecting these artificial chromosomes, the Genetic Algorithm will speed up the convergence of the evolutionary processes. Intensive experimental results indicate that the proposed algorithm is very encouraging and it can improve the solution quality significantly.**

Keywords: *Genetic Algorithm; Mining Gene Structure Information, Artificial Chromosomes*

I. Introduction

More and more sophisticated evolutionary algorithms (EAs) have been proposed and developed to solve combinatorial problems in recent years. Some of them were quite successful; however, it is not always clear why and how an EA works [1-4]. In this research, we took a close look at the evolutionary process for a single machine scheduling problem and came out with the new idea of generating artificial chromosomes to further improve the solution quality of the genetic algorithm. This study examined the evolutionary process and tried to develop a probability matrix to guide the evolutionary procedure. From the point of view of searching each gene allocation distribution, a simple gene mutation matrix was developed. In addition, the proposed approach was also used to illustrate how insights gained which can be further converted into our understanding of EA's behaviors and guide us in developing new and better techniques. The proposed algorithm will be tested on a single machine scheduling problems with the objectives to minimize the total deviations. In addition, the proposed algorithm will be compared with a set of dominance properties developed by us in earlier studies to evaluate the effectiveness of the new algorithm.

Manuscript received October 30, 2006.

Pei-Chann Chang is with Department of information management, Yuan-Ze University. 135 Yuan-Dong Rd., Taoyuan 32026, Taiwan, R.O.C (+886 03 4638800#2516#2525 e-mail: iepchang@saturn.yzu.edu.tw).

Shih-Shin Chen.,is with Department of Industrial Engineering and Management, Yuan-Ze University. (e-mail: s939506@mail.yau.edu.tw).

Chin-Yuan Fan ,is with Department of Industrial Engineering and Management, Yuan-Ze University (e-mail: s948906@mail.yau.edu.tw).

Qiong-Hui Ko is with Department of Industrial Engineering and Management, Yuan-Ze University (e-mail: s945408@mail.yau.edu.tw)

II. Problem Statements

Genetic algorithms (GAs) are powerful search techniques that are used successfully to solve problems in many different disciplines. The genetic algorithm relies on genetic operators for selection, crossover, mutation, and replacement. The selection operators use the fitness values to select a portion of the population to be parents for the next generation. Parents are combined using the crossover and mutation operators to produce offsprings. This process combines the fittest chromosomes and passes superior genes to the next generation, thus providing new points in the solution space. The replacement operators ensure that the 'least fit' or weakest chromosomes of the population are displaced by more fit chromosomes. However, as observed by most researchers the GA will be trapped into local optimality in the earlier stages and cannot be converged into global optimal in most of the cases. The problems with the steady states GAs having premature convergence led to the desire to further improve the convergence of the algorithm. Especially, for most combinatorial problems such as Traveling Salesman Problems (TSP), machine scheduling problems, and vehicles routing problems are very difficult to solve and even for moderate cases the GA will be converged prematurely.

In this paper, a deterministic single machine scheduling problem without release dates is investigated and the objective is to minimize the total sum of earliness and tardiness penalties. A detailed formulation of the problem is described as follows: A set of n independent jobs $\{J_1, J_2, \cdots, J_n\}$ has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and unforced machine idle time is not allowed. Job $J_j, j = 1, 2, \cdots, n$ becomes available for processing at the beginning, requires a processing time p_j and should be completed on its due date d_j. For any given schedule, the earliness and tardiness of J_j can be respectively defined as $E_j = \max\{0, d_j - C_j\}$ and $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of J_j. The objective is then to find a schedule that minimizes the sum of the earliness and tardiness penalties of all jobs $\sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j)$, where α_j and β_j are the earliness and tardiness penalties of job J_j. The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed. The early cost may represent the cost of completing a product early, the deterioration cost for a perishable goods or a holding (stock) cost for finished goods. The tardy cost can represent rush shipping costs, lost sales and loss of goodwill. It is assumed that no unforced machine idle time is allowed, so the machine is only idle if no job is currently available for processing. This assumption reflects a production setting where the cost of machine idleness is higher than the early cost incurred by completing any job before its due date, or the capacity of the machine is limited when compared with its demand, so that the machine must indeed be kept running. Some specific examples of production settings with these characteristics are provided by Ow and Morton[15], Azizoglu, et al. [4], Wu and Storer and Chang [23], Su and Chang [19] and Su and Chang[20]. The set of jobs is assumed to be ready for processing at the beginning which is a characteristic of the deterministic problem. As a generalization of weighted tardiness scheduling, the problem is strongly NP-hard in Lenstra, Rinnooy Kan and Brucker [12]. To the best of our knowledge, the earlier work in this problem is due to Chang and Lee [7], Chang and Lee[6], Wu, S.D., Storer, R.H. and Chang [23], Chang [8]. Belouadah et al. [5] delt with the similar problem however with a different objective in minimizing the total weighted completion time and the problem is the same as discussed in Hariri and Potts[10]. Kim, Y.-D.and C. A. Yano. [11] discussed some properties of the optimal solution, and these properties are used to develop both optimal and heuristic algorithms. Later on, Akturk and Ozdemir [3][2] developed various dominance rules to solve the problem. Valente and Alves [21][22] presented a branch-and-bound algorithm based on a decomposition of the problem into weighted earliness and weighted tardiness subproblems. Two lower bound procedures were presented for each subproblem, and the lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. In Valente and Alves[22], they analyze the performance of various heuristic procedures, including dispatch rules, a greedy procedure and a decision theory search heuristic.

The early/tardy problem with equal release dates and no idle time, however, has been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, branch-and-bound algorithms were presented by Abdul-Razaq and Potts[1], Li[13] and Liaw [14]. The lower bounding procedure of Abdul-Razaq and Potts was based on the subgradient optimization approach and the dynamic programming state-space relaxation technique, while Li and Liaw used Lagrangean relaxation and the multiplier adjustment method. Among the heuristics, Ow and Morton[15] developed several dispatch rules and a filtered beam search procedure. Valente and Alves[22] presented an additional dispatch rule and a greedy procedure, and also considered the use of dominance rules to further improve the schedule obtained by the heuristics. A neighborhood search algorithm was also presented by Li [13]

III. METHODOLOGY

As surveyed in the literature, most approaches in solving the single machine scheduling problems are traditional optimization methods such as Branch and Bound; Dynamic programming; Lagrangean relaxation and Heuristics. Instead, the Genetic Algorithm is proposed in this research to solve the SME problems. However, to prevent the premature convergence, artificial chromosomes will be generated to speed up the convergence and jump out the local optimality to reach a near global optimal.

The first observation in this research is that during the evolving process of the GA, all the chromosomes will converge slowly into certain distribution after the final runs. If we take a close look at the distribution of each gene in each assigned position, we will find out that most the genes will be converged into certain locations which means gene can be allocated to the position if there is a probabilistic matrix to guide the assignment of each gene to each position.

Artificial Chromosomes are developed according to this observation and a dominance matrix will record this gene distribution information. The dominance matrix is transformed into a probability matrix to decide the next assignment of a gene to a position. Consequently, AC is integrated into the procedure of Genetic Algorithm and it attends to improve the performance of Genetic Algorithm. The primary procedure is to collect gene information first and to use the gene information to generate artificial chromosomes. Before collecting the gene information, AC collects the chromosomes whose fitness is better by comparing the fitness value of each chromosome with average fitness value of current population. Thus, the average fitness is calculated. The following is the detail procedure of the AC approach.

MainProcedure

Population: The population used in the Genetic Algorithm
Generations: The number of generations
startingGen: It determines when does the AC works
interval: The frequency to generate artificial chromosomes

```
1.  Initiate Population
2.  ConstructInitialPopulation(Population)
3.  RemovedIdenticalSolution()
4.  counter ← 0
5.  while counter < generations do
6.      Evaluate Objectives and Fitness()
7.      FindEliteSolutions(i)
8.      if counter < startingGen or counter %
      interval != 0 do
9.          Selection with Elitism Strategy()
10.         Crossover()
```

```
11.         Mutation()
12.         TotalReplacement()
13.     else
14.         CalculateAverageFitness()
15.         CollectGeneInformation()
16.         GenerateArtificialChromsomomes()
17.         $\mu + \lambda$ Replacement()
18.     End if
19.     counter ← counter + 1
20. end while
```

There are two parameters of this algorithm, which are *startingGen* and *interval*. We have done a parameter configuration experiment by DOE, and shows that there is no significant difference. So the *startingGen* and *interval* are set to 500 and 50, respectively.

Step 1:To Calculate Average Fitness and to Convert Gene Information into Dominance Matrix

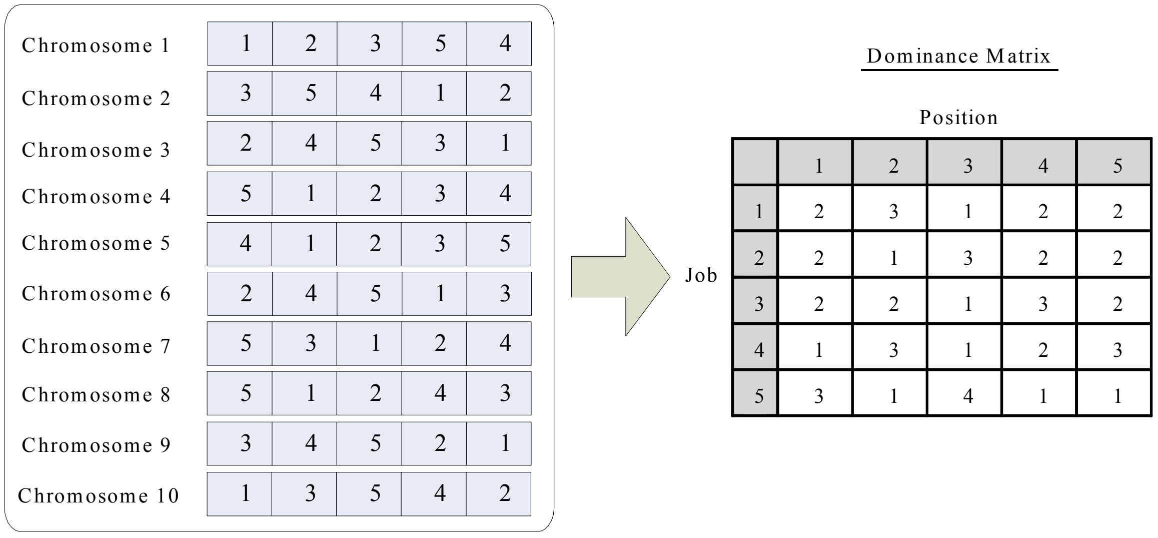
Instead of collecting all gene information from a population, the method selects better chromosomes which are compared with the average fitness of the population in the current generation. For a better chromosome, if job *i* exists at position *j*, the number of occurrence of X_{ij} is incremented by 1. Take a five jobs problem for example (see Figure 1), suppose that there are ten sequences (chromosomes) whose fitness is better than the average fitness. Then, we accumulate the gene information from these ten chromosomes into dominance matrix. For the position 1, there are two job 1, two job 2, 2 two 3, one job 4, and three job 5. Therefore, the dominance matrix contains the gene information from better chromosomes is illustrated in Figure 1.

Step 2: Generate Artificial Chromosomes

As soon as we collect gene information into dominance matrix, we are going to assign jobs onto the positions of each artificial chromosome. The assignment sequence for every position is assigned randomly, which is able to diversify the artificial chromosomes. After we determine the assignment sequence, we select one job assigned to each position by roulette wheel selection method based on the probability of each job on this position. After we assign one job to a position, the job and position in the dominance matrix are removed. Then, the method continues to select the next job until all jobs are assigned. Suppose we will assign the first job at position 3 in the beginning, which shown in Figure 2. The frequency of each job at position 3 is [1, 3, 1, 1, 4] from job 1 to job 5. Because the number of total frequency is 10, the corresponding probability for job 1 is 1/10, job 2 is 3/10, and so on. Then, we accumulate the probability from job 1 to 5 and roulette wheel select is able to apply this accumulated probability. This information is shown at the last column of the Figure 2. If there is a random probability 0.6, the job 4 is assigned to position 3.

Replacement Strategy

After injecting artificial chromosomes into the population, we use $\mu+\lambda$ strategy, which combines previous parent population and artificial chromosomes. Then, we select better μ chromosomes from the combined population. Consequently, better solutions are preserved to the next generation.



Chromosome 1	1	2	3	5	4
Chromosome 2	3	5	4	1	2
Chromosome 3	2	4	5	3	1
Chromosome 4	5	1	2	3	4
Chromosome 5	4	1	2	3	5
Chromosome 6	2	4	5	1	3
Chromosome 7	5	3	1	2	4
Chromosome 8	5	1	2	4	3
Chromosome 9	3	4	5	2	1
Chromosome 10	1	3	5	4	2

Dominance Matrix

Position

Job	1	2	3	4	5
1	2	3	1	2	2
2	2	1	3	2	2
3	2	2	1	3	2
4	1	3	1	2	3
5	3	1	4	1	1

Fig. 1. To collect gene information and Converted into a Dominance Matrix

Job	Pos. 3	Prob.	Accum
1	1	1/10	1/10
2	3	3/10	4/10
3	1	1/10	5/10
4	1	1/10	6/10
5	4	4/10	10/10

Fig. 2. The probability and accumulated probability of each job for position 3

Updated Dominance Matrix

Position

Job

	1	2	~~3~~	4	5
1	2	3	~~1~~	2	2
2	2	1	~~3~~	2	2
3	2	2	~~1~~	3	2
~~4~~	~~1~~	~~3~~	~~1~~	~~2~~	~~3~~
5	3	1	~~4~~	1	1

Fig. 3. The updated dominance matrix after assigning job 4 at position 3

After assigning the job 4 at position 3, suppose the position 2 is the next one to be assigned. It is shown in figure 4. Thus, in the same procedures, the probability of each job is calculated as well as the accumulated probability. Then, roulette wheel selection method will select a job based on these probability of each job. Consequently, the algorithm iteratively selects others jobs to a position until all jobs are assigned.

Job	Pos. 2	Prob.	Accum
1	3	3/7	3/7
2	1	1/7	4/7
3	2	2/7	6/7
5	1	1/7	7/7

Fig. 4. The probability and accumulated probability of each job for position 2

IV. EXPERIMENTAL RESULTS

To test the effectiveness of ACGA, we compared this algorithm with Dominance Properties, and GA with Dominance Properties. In addition, in order to make sure that the proposed algorithm works well, single machine scheduling problems with the objective to minimize the early-tardy cost are presented. The testing instances are taken from Sourd [18] for benchmark tests.

Sourd [18] provided numerous data sets, including 20, 30, 40, 50, 60, and 90. Each job set of 20 jobs to 50 jobs contains 49 combinations while there are 9 instances in the job set of 60 jobs and 90 jobs. We carried out our experiment on these 214 instances and each instance is replicated 30 times. The stopping criterion is the number of examined solutions, which is 100,000 solutions. The parameters of GA includes the crossover rate, mutation rate, and population size are set as 0.8, 0.5, and 100, respectively. The proposed algorithm is compared with a Simple Genetic algorithm (SGA), a genetic algorithm with dominance properties (GADP) that was proposed by our previous work, a GA with injecting AC (ACGA), and a hybrid algorithm ACGADP. The GADP applies a heuristic to generate a good initial population in the beginning and it is able to enhance the exploration ability of Simple Genetic Algorithm. Finally, an average relative error is applied as a performance metric that shows each average objective with respect to its optimal solution. The equation is calculated by $(avgObj - Opt)/Opt * 100\%$ where the avgObj is the average objective value obtained by each algorithm. These results are depicted at Table 1 and the completed test results are available on our website[1].

TABLE 1.

The average relative error ratio of the four algorithms (%)

Instance	SGA	GADP	ACGA	ACGADP
sks222a	5402	5291	5289	5288.7
sks225a	4174	3959	3958	3958
sks228a	2156	2085	2085	2085
sks252a	4195	3947	3979	3947
sks255a	2489	2372	2380	2372.5
sks258a	1250	1242	1200	1192.7
sks282a	4435	4353	4351	4353.8
sks285a	4643	4452	4452	4452
sks288a	3518	3421	3421	3421
sks322a	12066	11572	11577	11570
sks325a	8152	7703	7587	7587
sks328a	3556	3164	3164	3164
sks352a	8203	7395	7394	7394.2
sks355a	6849	6068	6065	6057.5
sks358a	3283	3074	3073	3072.5
sks382a	11319	11152	11149	11142
sks385a	9212	9148	9148	9148
sks388a	11499	11317	11317	11317
sks422a	26211	25658	25659	25657
sks425a	13592	12604	12606	12601

[1] http://ppc.iem.yzu.edu.tw/publication/sourceCodes/InjectionArtificialChromosomes/

sks428a	7741	7129	7129	7129
sks452a	12634	11367	11406	11367
sks455a	7566	6405	6427	6405
sks458a	5587	4303	4321	4300.4
sks482a	20122	19580	19573	19562
sks485a	16023	15309	15338	15350
sks488a	17999	16881	16863	16863
sks522a	4.483	0.044	0.010	0.003
sks525a	2.674	0.012	0.020	0.004
sks528a	11.47	0.213	0.370	0.056
sks552a	8.590	0.000	0.136	0.000
sks555a	20.08	0.550	0.285	0.216
sks558a	39.40	0.545	0.000	0.000
sks582a	4.349	0.637	0.029	0.168
sks585a	4.165	0.008	0.044	0.089
sks588a	6.352	0.008	0.004	0.004
sks622a	4.576	0.000	0.167	0.000
sks625a	5.720	0.095	0.123	0.095
sks628a	8.119	0.059	0.070	0.047
sks652a	7.211	0.000	0.227	0.000
sks655a	18.44	0.000	0.371	0.000
sks658a	32.13	0.001	0.329	0.002
sks682a	2.604	0.584	0.090	0.540
sks685a	4.359	0.032	0.050	0.016
sks688a	6.781	0.307	0.262	0.283
sks922a	5.769	0.861	0.060	0.534
sks925a	6.108	0.010	0.037	0.024
sks928a	23.81	0.494	0.438	0.574
sks952a	12.79	0.056	0.202	0.043
sks955a	32.25	0.268	0.330	0.265
sks958a	53.73	0.135	0.401	0.336
sks982a	3.784	0.434	0.031	0.475
sks985a	8.519	0.186	0.151	0.190
sks988a	11.51	0.011	0.022	0.013
Avg ER	9.971	0.251	0.173	0.109

GADP, ACGA, and ACGADP outperform the SGA in the average error ratio because the total average ratio of SGA is 9.971% while other three algorithms is less than 0.26%. To distinguished the performance of the four algorithms, we test it by ANOVA which shows that there is a significant different among these method so the Duncan grouping results is applied. Table 2 shows that there is no much difference between ACGA and ACGADP. In addition, there is no difference between ACGADP and GADP. However, ACGA, ACGADP, and SGA are not in the same group, it means there is significant difference between each of them. Consequently, ACGA is the best algorithm, ACGADP and GADP are second rank, and SGA is the worst.

TABLE 2.
The Duncan grouping result for the four algorithms

Duncan Grouping		Mean	N	Method
	A	13982.894	6420	SGA
	B	12827.096	6420	GADP
	B			
C	B	12816.471	6420	ACGADP
C				
C		12813.276	6420	ACGA

To show the convergence process for these difference algorithms, i.e., SGA, GADP, ACGA and ACGADP, instance sks988a is applied as a demonstration. Figure 4 shows that GADP has the quickest convergence then ACGA and lastly is SGA. However, after 20 generation ACGA has almost the same solution quality as GADP.

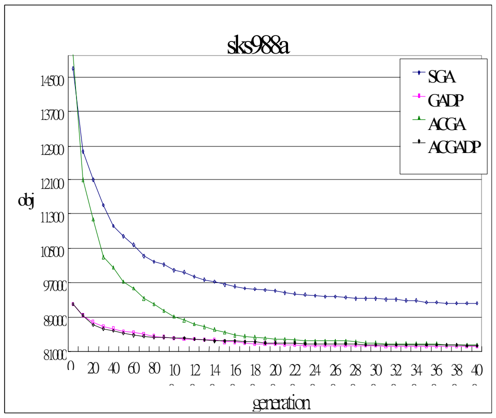


Fig. 4. The convergence diagram of the four algorithms in job 90 problem (sks988a)

V. DISCUSSION AND CONCLUSIONS

This research proposes a Genetic Algorithm with injecting artificial chromosomes in solving the single machine scheduling problems with the objective of minimizing the total deviation. From the experimental results, we find out that the proposed algorithm is able to obtain a very good solution quality when compared with SGA and GADP. Without any complex mathematic calculation and proofing, ACGA can solve the problem in as good as or a better solution quality than GADP. The reason is that the dominance matrix can truly capture the gene information and prohibits jobs that are assigned to inappropriate positions. Then, jobs are potentially to be selected when they are dedicated to a position with higher probability by roulette wheel selection method. AC will create a much diversified population but with the probability to reach a global optimal. Consequently, after the intensive experiments in the single machine scheduling problem, the result is very satisfactory and convincing and we expect to apply the ACGA to other combinatorial problems in the near future.

References

[1] Abdul-Razaq, T., and Potts, C. N. " Dynamic programming state-space relaxation for single machine scheduling." *Journal of the Operational Research Society* 39 ,1988, 141—152.

[2] Akturk, M.S., and Ozdemir, D. "A new dominance rule to minimize total weighted tardiness with unequal release dates." *European Journal of Operational Research* 135 ,2001, 394-412.

[3] Akturk, M.S., and Ozdemir, D. "An exact approach to minimizing total weighted tardiness with release date." *IIE Transactions* 32,2000, 1091-1101.

[4] Azizoglu, M., Kondakci, S. and Ömer Kirca. " Bicriteria scheduling problem involving total tardiness and total earliness penalties." *International Journal of Production Economics*, Vol. 23, No. 1-3, 1991,pp. 17-24.

[5] Belouadah, H., Posner, M.E., and Potts, C.N. "Scheduling with release dates on a single machine to minimize total weighted completion time." *Discrete Applied Mathematics* 36,1992, 213-231.

[6] Chang, P.C. and Lee, H.C.,"A Greedy Heuristic for Bi-criterion Single Machine Scheduling Problems," *Computers and Industrial Engineering*, Vol. 22, No. 2, 1992b,pp. 121-131.

[7] Chang, P.C. and Lee, H.C.,"A Two Phase Approach for Single Machine Scheduling : Minimizing the Total Absolute Deviation," *Journal of the Chinese Institute of Engineers*, Vol. 15, No. 6, 1992a, pp. 735-742.

[8] Chang, P.C., ,"A Branch and Bound Approach for Single Machine Scheduling with Earliness and Tardiness Penalties," *Computers & Mathematics with Applications*, Vol. 37, 1999,pp. 133-144.

[9] Francis Sourd, "Earliness-tardiness scheduling with setup considerations." *Computers and Operations Research* **32** ,2005. pp. 1849–1865

[10] Hariri, A.M.A., and Potts, C.N. "Scheduling with release dates on a single machine to minimize total weighted completion time."*Discrete Applied Mathematics* 36,1983, 99-109.

[11] Kim, Y.-D., C. A. Yano. "Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates." Naval Research Logistics. 1994,.41(7) 913–933.

[12] Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. " Complexity of machine scheduling problems."A*nnals of Discrete Mathematics* 1 ,1977, 343—362.

[13] Li, G." Single machine earliness and tardiness scheduling. "*European Journal of Operational Research* 96,1997, 546—558.

[14] Liaw, C.-F. "A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. "*Computers & Operations Researc*h 26,1999, 679—693.

[15] Ow, P. S., and Morton, E. T. "The single machine early/tardy problem." *Management Science* 35 ,1989, 177—191.

[16] Ow, P. S., and Morton, T. E. "Filtered beam searches in scheduling." International Journal of Production Research 26,1988, 35—62.

[17] Ow, P. S., and Smith, S. F. "Viewing scheduling an opportunistic problem solving process." *Annals of Operations Research* 12, 1988, 85-108.

[18] Sourd, F., and Sidhoum, S.K, "An efficient algorithm for the earliness tardiness," http://www-poleia.lip6.fr/ sourd/, 2005.

[19] Su, L.H. and Chang, P.C.,"A Heuristic to Minimize A Quadratic Function of Job Lateness on A Single Machine," *International Journal of Production Economics*, Vol. 55, No. 2, 1998.pp. 169-175.

[20] Su, L.H. and Chang, P.C.,"Scheduling n jobs on one machine to minimize the maximum lateness with a minimum number of tardy jobs," *Computers and Industrial Engineering*, Vol.40, No.4, ,2001,pp.349-360.

[21] Valente, J. M. S., and Alves, R. A. F. S." Heuristics for the early/tardy scheduling problem with release dates. "Working Paper 129, Faculdade de Economia do Porto, Portugal, 2003a.

[22] Valente, J. M. S., and Alves, R. A. F. S. "Improved heuristics for the early/tardy scheduling problem with no idle time." Working Paper 126, Faculdade de Economia do Porto, Portugal, 2003b.

[23] Wu, S.D., Storer, R.H. and Chang, P.C.,"One Machine Rescheduling Heuristic With Efficiency and Stability as Criteria," *Computers and Operations Research*, Vol. 20, No. 1, ,1993, pp. 1-14.