FEP WORKING PAPERS
FEP WORKING PAPERS

RESEARCH WORK IN PROGRESS

N. 312, FEB. 2009

GENETIC ALGORITHMS FOR SINGLE MACHINE SCHEDULING WITH QUADRATIC EARLINESS AND TARDINESS COSTS

JORGE M. S. VALENTE*[1]
MARIA R. A. MOREIRA*[2]
ALOK SINGH[3]
RUI A. F. S. ALVES*

*FACULDADE DE ECONOMIA, UNIVERSIDADE DO PORTO
[1]LIAAD - INESC PORTO L. A., [2]EDGE
[3]DEP. OF COMPUTER AND INFORMATION SCIENCES,
UNIVERSITY OF HYDERABAD

U.PORTO
FEP FACULDADE DE ECONOMIA UNIVERSIDADE DO PORTO

Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs

Jorge M. S. Valente *
LIAAD - INESC Porto L. A.,
Faculdade de Economia, Universidade do Porto, Portugal

Maria R. A. Moreira
EDGE, Faculdade de Economia, Universidade do Porto, Portugal

Alok Singh
Department of Computer and Information Sciences,
University of Hyderabad, India

Rui A. F. S. Alves
Faculdade de Economia, Universidade do Porto, Portugal

February 10, 2009

Abstract

In this paper, we consider the single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. We propose a genetic approach based on a random key alphabet, and

*Corresponding author. Address: Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal. Telephone: + 351 22 557 11 00. Fax: + 351 22 550 50 50. E-mail: jvalente@fep.up.pt.

present several algorithms based on this approach. These versions differ on the generation of both the initial population and the individuals added in the migration step, as well as on the use of local search. The proposed procedures are compared with the best existing heuristics, as well as with optimal solutions for the smaller instance sizes.

The computational results show that the proposed algorithms clearly outperform the existing procedures, and are quite close to the optimum. The improvement over the existing heuristics increases with both the difficulty and the size of the instances. The performance of the proposed genetic approach is improved by the initialization of the initial population, the generation of greedy randomized solutions and the addition of the local search procedure. Indeed, the more sophisticated versions can obtain similar or better solutions, and are much faster. The genetic version that incorporates all the considered features is the new heuristic of choice for small and medium size instances.

Keywords: scheduling, single machine, quadratic earliness and tardiness, genetic algorithms

1 Introduction

In this paper, we consider the single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. Formally, the problem can be stated as follows. A set of n independent jobs $\{1, 2, \cdots, n\}$ has to be scheduled 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 preemptions are not allowed. Job $j, j = 1, 2, \cdots, n$, requires a processing time p_j and should ideally be completed on its due date d_j. Also, let h_j and w_j denote the earliness and tardiness penalties of job j, respectively. Given a schedule, the earliness and tardiness of job j are 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 job j. The objective is then to find a schedule that minimizes the sum of the weighted quadratic earliness and tardiness costs

$\sum_{j=1}^{n} \left(h_j E_j^2 + w_j T_j^2\right)$, subject to the constraint that no machine idle time is allowed.

Even though scheduling models with a single processor may appear to arise infrequently in practice, this scheduling environment actually occurs in several activities (for a specific example in the chemical industry, see Wagner et al. (2002)). Also, the performance of many production systems is quite often dictated by the quality of the schedules for a single bottleneck machine. Moreover, the study of single machine problems provides results and insights that prove valuable for scheduling more complex systems.

Early/tardy scheduling models have received considerable and increasing attention from the scheduling community, due to their practical importance and relevance. In fact, scheduling problems with earliness and tardiness costs are compatible with the concepts of supply chain management and just-in-time production. Indeed, these production strategies, which have been increasingly adopted by many organisations, view both early and tardy deliveries as undesirable.

In this paper, we consider quadratic earliness and tardiness penalties, instead of the more usual linear objective function, in order to penalize more heavily deliveries that are quite early or tardy. On the one hand, this is appropriate for practical settings where non-conformance with the due dates is increasingly undesirable. Moreover, on the other hand, the quadratic penalties also avoid schedules in which a single or only a few jobs contribute the majority of the cost, without regard to how the overall cost is distributed.

The assumption that no machine idle time is allowed is actually appropriate for many production settings. Indeed, when the capacity of the machine is limited when compared with the demand, the machine must be kept running in order to satisfy the customers' orders. Idle time must also be avoided for machines with high operating costs, since the cost of keeping the machine running idle is then higher than the earliness costs. The assumption of no idle time is additionally justified when starting a new production run involves high setup costs or times (e.g. furnaces or similar machines), since stopping and restarting the machine is not a viable option in these settings. Some specific examples of production environments where the no idle time

assumption is appropriate have been given by Korman (1994) and Landis (1993).

This problem has been previously considered, and both exact and heuristic approaches have been proposed. A lower bounding procedure and a branch-and-bound algorithm were developed in Valente (2007a), while Valente & Alves (2008) presented several dispatching heuristics, as well as simple improvement procedures. Classic, filtered and recovering beam search algorithms were considered in Valente (2008a), and Valente & Moreira (2008) proposed various greedy randomized dispatching heuristics.

The corresponding problem with linear costs $\sum_{j=1}^{n}(h_j E_j + w_j T_j)$ has also been considered by several authors. Lower bounds and branch-and-bound algorithms were presented by Abdul-Razaq & Potts (1988), Li (1997), Liaw (1999) and Valente & Alves (2005c). Several heuristic approaches were also proposed in Ow & Morton (1989), Valente & Alves (2005a,b) and Valente et al. (2006).

Problems with a related quadratic objective function have also been previously studied. Schaller (2004) considered the single machine problem with inserted idle time and a linear earliness and quadratic tardiness $\sum_{j=1}^{n}\left(E_j + T_j^2\right)$ objective function, while the no idle time version of this problem was analysed in Valente (2007b, 2008b).

The minimization of the quadratic lateness, where the lateness of job j is defined as $L_j = C_j - d_j$, has also been studied by Gupta & Sen (1983), Sen et al. (1995), Su & Chang (1998) and Schaller (2002). Baker & Scudder (1990) and Hoogeveen (2005) provide excellent surveys of scheduling problems with earliness and tardiness penalties, while a review of scheduling models with inserted idle time is given in Kanet & Sridharan (2000).

In this paper, we present several genetic algorithms, and analyse their performance on a wide range of instances. The proposed genetic approach uses a random key alphabet. Therefore, each chromosome is encoded as a vector of random numbers. The various versions of the genetic approach differ on the generation of both the initial population and the individuals added in the migration step, as well as on the use of local search. The genetic algorithms are then compared with existing procedures, as well as

with optimal solutions for some instance sizes.

The remainder of this paper is organized as follows. In section 2, we describe the proposed genetic algorithm approach, and present the several versions that were considered. The computational results are reported in section 3. Finally, we provide some concluding remarks in section 4.

2 The proposed genetic algorithm procedures

In this section, we first briefly describe the main features of genetic algorithms. Then, the encoding used to represent the problem solutions is presented. The evolutionary strategy, i.e. the transitional process between consecutive populations, is also described. Finally, we present the six different versions that were considered for the proposed genetic algorithm approach.

2.1 Genetic algorithms

Genetic algorithms are adaptive methods that can be used to solve optimization problems. The term genetic algorithm was first used by Holland in its book *Adaptation in Natural and Artificial Systems* (Holland, 1975). This book was fundamental to the creation of what is now a large and active field of research. Even though Holland's work placed little emphasis on optimization, the majority of the research on genetic algorithms has indeed since been focused on solving optimization problems. Due to their increasing popularity in recent years, the literature on genetic algorithms now includes a quite large number of papers. References describing in detail the genetic algorithm approach and its applications can be found in Goldberg (1989) and Reeves (1997, 2003).

Genetic algorithms are based on the evolution process that occurs in natural biology. Indeed, over the generations, and as first stated by Charles Darwin in *The Origin of the Species*, natural populations tend to evolve according to the principles of natural selection or survival of the fittest. The genetic algorithms mimic this process, by evolving populations of solutions to optimization problems.

In order to apply a genetic algorithm to a specific problem, it is first necessary to choose a suitable encoding or representation. In this encoding, a solution to the problem being considered is represented by a set of parameters, known in genetic terminology as genes. These parameters or genes are joined together in a string of values that represents (or encodes) the solution to the problem. In the genetic terminology, this string is denoted as a chromosome or individual. A fitness value is also associated with each chromosome. This value measures the merit or quality of the solution represented by that chromosome or individual.

At each iteration, the genetic algorithm evolves the current population of chromosomes into a new population. This evolution is conducted using selection, crossover and mutation mechanisms. Some of the current individuals may be simply selected and copied to the new population. Also, a crossover operator is used in the reproduction phase to combine parent individuals selected from the current population, in order to produce offspring which are placed in the new population. The parent chromosomes are chosen randomly, although this selection is usually performed using a scheme which favours individuals with higher fitness values. The genes of the two parents are then combined by the crossover operator, yielding one or more offspring. Finally, a mutation operator can be applied to some individuals. This mutation operator changes the genetic material of those individuals, i.e. it changes one or more of their genes.

The reproduction phase and the crossover operator tend to increase the quality of the populations, since fitter individuals are more likely to be selected as parents. However, they also tend to force a convergence of those populations (i.e. the individuals tend to become quite similar). This convergence effect can be offset by the mutation mechanism. Indeed, by changing the genetic material, the mutation operator tries to guarantee the diversity of the population, thereby ensuring a more extensive search of the solution space.

2.2 Chromosome representation and decoding

The genetic algorithm approach proposed in this paper uses the random key alphabet $U(0,1)$ proposed by Bean (1994) to encode the chromosomes. In the random key encoding, each gene is a uniform random number between 0 and 1. Consequently, each chromosome is then encoded as a vector of random keys (random numbers). Therefore, in the proposed algorithms each chromosome is composed of n genes $g_j, j = 1, 2, \cdots, n$, so the size of each chromosome is equal to the number of jobs, i.e. chromosome $= (g_1, g_2, \ldots, g_n)$.

In order to calculate the fitness of an individual, it is first necessary to decode its chromosome into the corresponding solution to the considered problem, i.e. into a sequence of the jobs. This decoding or mapping of a chromosome into a schedule is accomplished by performing a simple sort of the jobs. The priorities used in this sorting operation are provided by the genes. More specifically, the priority of job j in the sorting operation is equal to g_j (see figure 1 for an example).

An important feature of the random key encoding is the fact that all offspring generated by crossover operators correspond to feasible solutions. This is accomplished by moving the feasibility issue from the crossover operator into the chromosome decoding procedure. Indeed, if any vector of random numbers can be converted into a feasible solution, then any chromosomes obtained via the crossover operator also correspond to feasible solutions. Then, through its internal dynamics, the genetic algorithm can learn the relationship between random key vectors and solutions with good fitness and objective function values.

This feature is a significant advantage of the random key alphabet over the more natural encoding where each chromosome is a permutation of the job indexes, and its importance increases as the problem becomes more heavily constrained. This natural encoding does not require a decoding of the chromosome into the corresponding schedule. However, with the natural encoding, the crossover operation is made much more difficult and complicated by the need to assure that the resulting offspring correspond to a feasible solution.

2.3 The evolutionary strategy

A quite large number of genetic algorithm variants can be obtained by choosing different selection, reproduction, crossover and mutation operators. We now describe the evolutionary strategy employed in the proposed approach, i.e. the specific mechanisms that are used to generate a new population from the current set of individuals. Throughout the procedure, the size of the population is kept constant. This size is set equal to a multiple *pop_mult* of the size of the problem (i.e. the number of jobs n), where pop_mult is a user-defined parameter. This strategy has proved adequate in previous applications of genetic algorithms based on the same evolutionary approach (Valente & Gonçalves, 2008; Gonçalves, 2007; Gonçalves et al., 2005).

Given a current population, the next population is obtained through elitist selection, crossover and migration mechanisms. The discussion of the generation of the initial population is deferred to the next section. The calculation of the fitness value of a chromosome will also be addressed in that section.

The elitist selection strategy, proposed by Goldberg (1989), copies some of the best individuals in the current population to the new population. The number of chromosomes that are copied in this elitist selection phase is set equal to a proportion *elit_prop* of the population size, where elit_prop is a user-defined parameter. The advantage of the elitist selection strategy over the traditional generational approach where the entire population is completely replaced with new chromosomes is that the best individual in the population improves monotonically over time. A potential downside is that it can lead to a premature convergence of the population. However, this can be overcome by using high mutation or migration rates.

In the proposed evolutionary strategy, the migration mechanism is used instead of the traditional gene-by-gene mutation operator. In the migration phase, new individuals are generated and added to the new population. The number of newly generated individuals is equal to a proportion *mig_prop* of the population size, where mig_prop is a user-defined parameter. The specific process by which these new individuals are generated will be ad-

dressed in the next section. Like in the traditional mutation operator, and as previously mentioned, the migration mechanism tries to prevent premature convergence, as well as to assure the diversity of the population and an extensive search of the solution space. Due to the specific ways in which this phase is implemented (which will be described in the next section), the migration mechanism also assures that the proposed genetic approach, if allowed to run for a sufficient amount of time, will visit all possible solutions, and therefore also an optimal one.

Finally, the remaining individuals of the new population are generated via crossover. In the reproduction and crossover phase, two parents are initially selected. The first parent is chosen at random from the elite individuals in the current population (i.e. the individuals that are copied to the new population in the elitist selection phase). The second parent, on the other hand, is randomly selected from the entire current population. Then, the parameterized uniform crossover method developed by Spears & De Jong (1991), and described below, is used to create an offspring that is added to the new population. This process is repeated until the new population has been fully generated.

In the parameterized uniform crossover method, a random uniform number between 0 and 1 is generated for each gene. Then, this random number is compared with a user-defined parameter *cross_prob*. If the random number is less than or equal to the cross_prob parameter, the gene in the offspring is set equal to the corresponding gene in the first parent. Otherwise, the value of the gene is instead copied from the second parent (see figure 2 for an example).

The proposed evolutionary strategy is repeated until a stopping criterion is met. In our approach, we have chosen the number of iterations without improvement as stopping criterion. Thus, the genetic algorithms terminate when *stop_iter* populations have been generated without improving the best solution found so far, where stop_iter is a user-defined parameter. The evolutionary strategy is depicted in figure 3, and the main steps of the proposed approach are presented in figure 4.

2.4 The genetic algorithm versions

The discussion of the generation of the initial population and the new individuals added in the migration step, as well as the calculation of the fitness value, have been deferred to this section. Indeed, different strategies were considered for these issues. Therefore, we developed six genetic algorithm versions, corresponding to various combinations of these strategies.

In the version denoted by GA, and on the one hand, both the initial population and the new individuals created in the migration step are generated at random. On the other hand, the fitness value of a chromosome is set equal to the opposite of the objective function value of the corresponding sequence (i.e. the sequence obtained by decoding the chromosome, as described in the previous section).

The GA_IN version differs from the GA procedure only in the generation of the initial population, which is not fully random. In fact, in the GA_IN version, four non-random chromosomes are first introduced in the initial population. Indeed, previous studies (e.g. Reeves (1995) and Ahuja & Orlin (1997)) have shown that this can improve the performance of a genetic algorithm. In this paper, this will be referred to as initializing the first population (hence the "_IN" part of the heuristic identifier).

These four non-random chromosomes are created so that their corresponding schedules are equal to the sequences generated by the WPT_s_j_E, EDD, WPT_s_j_T and ETP_v2 dispatching rules considered in Valente & Alves (2008). The WPT_s_j_E and WPT_s_j_T heuristics performed well for instances were most jobs were early and tardy, respectively. The EDD heuristic is quite well-known and widely used, and performed better than the WPT_s_j_E and WPT_s_j_T rules when there was a greater balance between the number of early and tardy jobs. Finally, the ETP_v2 heuristic is the best-performing of the dispatching rules analysed in Valente & Alves (2008).

The version identified as GA_GR also first includes in the initial population the same four non-random chromosomes that are used in the GA_IN procedure. However, the GA_GR algorithm then differs from the GA_IN

version both in the generation of the remainder of the initial population, as well as on the migration phase. Indeed, in the GA_GR version, some chromosomes are created so that they correspond to a schedule generated by the greedy randomized (hence the "_GR" part of the heuristic identifier) dispatching rule RCL_VB proposed in Valente & Moreira (2008). The greedy randomized dispatching heuristics developed in Valente & Moreira (2008) perform a greedy randomization of the ETP_v2 rule, so a different schedule can be obtained each time one of these rules is executed. The RCL_VB strategy provided the best results, both in solution quality and in computation time, among all the approaches analysed in Valente & Moreira (2008).

The GA_GR version can be seen as performing a hybridization of the genetic algorithm and GRASP metaheuristics. Indeed, in the GA_GR heuristic, the greedy randomized strategy that is used in the construction phase of a GRASP procedure is employed to generate some chromosomes for the genetic algorithm. This hybridization has been used in previous studies, e.g. Ahuja et al. (2000) and Armony et al. (2000).

As previously mentioned, the GA_GR procedure first includes in the initial population the four non-random chromosomes also used in the GA_IN procedure. Then, a proportion $init\_gr$ of the remainder of the initial population is generated using the RCL_VB greedy randomized heuristic, where init_gr is a user-defined parameter. The remaining chromosomes of the initial population are then generated at random. In the migration phase, a proportion mig_gr of the chromosomes to be newly generated are created using the RCL_VB procedure, where mig_gr is again a user-defined parameter. The remainder of the migration phase chromosomes are then randomly generated.

The MA, MA_IN and MA_GR versions differ from their GA, GA_IN and GA_GR counterparts only in the calculation of the fitness value. In fact, these last three versions additionally use a local search procedure to improve the decoded sequence. More specifically, in order to calculate the fitness of a chromosome we first decode its corresponding sequence. Then, a local search procedure is used to improve this sequence. The fitness value is set equal to the opposite of the objective function value of this improved sequence.

Finally, the chromosome is changed (by rearranging its genes), so that it now corresponds to the improved sequence obtained after the application of the local search procedure. These last three versions of the proposed genetic approach combine a genetic evolutionary strategy with a local search procedure. Therefore, they can also be viewed as memetic algorithms (Moscato & Cotta, 2003), hence the "MA" part of their identifiers.

We considered three local search procedures: adjacent pairwise interchanges (API), 3-swaps (3SW) and first-improve interchanges (INTER). The API procedure considers pairs of adjacent jobs, while the 3SW method instead analyses three consecutive jobs. The INTER procedure, on the other hand, considers interchanges of two jobs, regardless of whether or not they are adjacent. More specifically, both the API and the 3SW procedures start at the beginning of the schedule, and terminate when the end of the sequence is reached. The API procedure interchanges a pair of adjacent jobs. If such an adjacent swap improves the objective function, the swap is retained and we move one position backward (when possible) in the sequence. Otherwise, the swap is reversed so the jobs are again scheduled in the original order, and we move one position forward in the schedule.

The 3SW procedure is similar, but it considers three consecutive jobs. All the possible permutations of the three jobs are then analysed, and the best configuration is determined. If the best configuration is different from the original order of the jobs, the jobs are scheduled according to that best configuration, and we move two positions backward (when possible) in the sequence. Otherwise, the original order of the jobs is retained, and we move one position forward in the schedule.

The INTER procedure starts at the first position in the sequence, and then considers all the successive positions, until the end of the sequence is reached. For each position in the sequence, the INTER method considers interchanging the job that is currently scheduled in that position with the jobs that are scheduled in the following positions. Whenever such an interchange improves the objective function value, that interchange is performed. If any interchange is made before the end of the sequence is reached, we start again at the beginning of the schedule. Otherwise, no change has been made to

the sequence, and the procedure terminates.

3 Computational results

In this section, we first present the set of problems used in the computational tests. Then, the preliminary computational experiments are described. These experiments were performed to determine adequate values for the parameters required by the several genetic algorithms. Finally, the computational results are presented. We first compare the genetic algorithms with existing procedures, and the heuristic results are then evaluated against optimum objective function values for some instance sizes. Throughout this section, and in order to avoid excessively large tables, we will sometimes present results only for some representative cases.

3.1 Experimental design

The computational tests were performed on a set of problems with 10, 15, 20, 25, 30, 40, 50, 75 and 100 jobs. These problems were randomly generated as follows. For each job j, an integer processing time p_j, an integer earliness penalty h_j and an integer tardiness penalty w_j were generated from one of the two uniform distributions $[45, 55]$ and $[1, 100]$, to create low (L) and high (H) variability, respectively. For each job j, an integer due date d_j is generated from the uniform distribution $[P\,(1 - T - R/2)\,, P\,(1 - T + R/2)]$, where P is the sum of the processing times of all jobs, T is the tardiness factor, set at 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, and R is the range of due dates, set at 0.2, 0.4, 0.6 and 0.8.

For each combination of problem size n, processing time and penalty variability (var), T and R, 50 instances were randomly generated. Therefore, a total of 1200 instances were generated for each combination of problem size and variability. All the algorithms were coded in Visual C++ 6.0, and executed on a Pentium IV - 2.8 GHz personal computer. Due to the large computational times that would be required, the GA heuristic was not applied to the instances with 100 jobs.

3.2 Preliminary tests

In this section, we describe the preliminary experiments that were performed to determine adequate values for the parameters required by the genetic algorithms. A separate problem set was used to conduct these preliminary experiments. This test set included instances with 25 and 50 jobs, and contained 5 instances for each combination of instance size, processing time and penalty variability, T and R. The instances in this smaller test set were generated randomly just as previously described for the full problem set. We considered the following values for the several parameters required by the proposed genetic algorithms:

pop_mult = {1, 2, 3};
elit_prop = {0.05, 0.10, 0.15, 0.20};
mig_prop = {0.10, 0.15, 0.20, 0.25};
cross_prob = {0.6, 0.7, 0.8};
stop_iter = {10, 30, 50};
init_gr = {0.1, 0.2, ... , 0.9};
mig_gr = {0.1, 0.2, ... , 0.9}.

The intervals for the elit_prop, mig_prop and cross_prob values were based on previous applications of genetic algorithms based on the same evolutionary approach (Valente & Gonçalves, 2008; Gonçalves, 2007; Gonçalves et al., 2005). Indeed, good results have consistently been obtained using values inside the considered ranges. The intervals for the pop_mult and stop_iter parameters were determined based both not only on a previous application of this evolutionary strategy to a scheduling problem (Valente & Gonçalves, 2008), but also on some initial tests. For the MA, MA_IN and MA_GR versions, we additionally considered the API, 3SW and INTER local search procedures, as previously mentioned.

The genetic algorithms were then applied to the test instances for all parameter (and local search procedure, for the MA, MA_IN and MA_GR versions) combinations. A thorough analysis of the objective function values and runtimes was then conducted, in order to select the values that provided the best trade-off between solution quality and computation time. The pa-

rameter values and local search procedure selected for the several genetic versions are given in table 1.

The same elit_prop, mig_prop and cross_prob values proved appropriate for all the versions. For the versions that use a local search procedure, the results were actually virtually identical for all the combinations of these parameters. For the other versions, there were some small differences in performance, and the chosen values provided good results for all instance types.

Table 1 shows that smaller values are required for the parameters pop_mult and stop_iter as the sophistication of the genetic versions increases. In fact, smaller populations and/or a lower number of iterations without improvement can be used, without compromising the solution quality, with the introduction in the genetic approach of features such as local search, population initialization and generation of greedy randomized solutions.

The init_gr parameter is smaller for the MA_GR version. In the GA_GR procedure, a higher percentage of greedy randomized solutions is then required in order to generate an initial population that contains high quality solutions. In the MA_GR version, however, the local search procedure already improves the quality of the solutions in the initial population, so only a lower percentage of greedy randomized initial solutions is required.

For both the GA_GR and MA_GR algorithms, the most appropriate value of the mig_gr parameter was equal to 0.5. Therefore, half of the new chromosomes introduced in the migration step are produced by the greedy randomized heuristic, while the remaining half are randomly generated. This shows that the best results are obtained when there is a balance between the introduction of relatively good solutions (generated by the greedy randomized heuristic) and random solutions. Indeed, the greedy randomized solutions increase the solution quality of the population, but the random solutions are also important, since they assure the diversity of the population and avoid its premature convergence.

Finally, the API local search procedure was selected. This procedure provided results that were virtually identical to those given by the other methods, and was significantly faster. We recall that the parameter values were

selected with the objective of obtaining the best trade-off between solution quality and computation time. Therefore, lower objective function values can still be obtained for some of the test instances, at the cost of increased computation times, by increasing the pop_mult or stop_iter values.

3.3 Comparison with existing heuristics

In this section, the proposed genetic algorithms are compared with existing heuristic procedures. On the one hand, the proposed algorithms are compared with the recovering beam search heuristic, denoted as RBS, developed in Valente (2008a). Additionally, the RCL_VB_3SW greedy randomized heuristic proposed in Valente & Moreira (2008) is also included in the heuristic comparison (in the following, this procedure will be simply denoted as R_V_3). Among the existing heuristics, the RBS algorithm provides the best average performance for small and medium size instances, while the R_V_3 procedure is the heuristic of choice for medium to large problems.

The RBS algorithm includes a final improvement step that uses the 3SW improvement procedure. Also, in the R_V_3 heuristic, each of the greedy randomized constructed schedules is improved by the 3SW procedure. Therefore, the 3SW method was also applied, as a final improvement step, to the best solution generated by each of the genetic algorithms. For each instance, 10 independent runs, with different random number seeds, were performed for all versions of the genetic algorithm (as well as for the R_V_3 heuristic).

Table 2 provides the mean relative improvement in objective function value over the RBS procedure (%imp). In table 3, we give the percentage number of times the R_V_3 heuristic and the genetic algorithms performs better (<), equal (=) or worse (>) than the RBS procedure. The relative improvement over the RBS heuristic is calculated as (rbs_ofv - heur_ofv) / rbs_ofv $\times$ 100, where rbs_ofv and heur_ofv are the objective function values of the RBS procedure and the R_V_3 procedure or the appropriate genetic version, respectively. The avg (best) column provides the relative improvement calculated with the average (best) of the objective function values obtained for all the 10 runs. The results in the avg column provide

an indication of the relative improvement we can achieve if the procedure is executed only once, while the best column shows the improvement that can be obtained if the algorithm is allowed to perform 10 runs.

The processing time and penalty variability has a major impact on the difficulty of the problem, and therefore on the differences between the results obtained by the several heuristic procedures. When the variability is low, the problem is much easier, and even simple procedures can obtain optimum or near optimum results, so there is little or no room for more sophisticated procedures to provide a large improvement. This has been previously established and discussed in the previous literature on this problem, and will also be shown quite clearly by the comparison with the optimum results performed in the next section.

For the low variability instances, the performance of the several considered procedures is virtually identical. Although there are a few differences in the objective function values, as can be seen by the results in table 3, these values are nevertheless extremely close, as shown by the virtually equal to 0 relative improvement values presented in table 2. When the variability is high, however, the problem becomes considerably more difficult, and the difference in performance between the several algorithms is much more noticeable.

When the average results over the 10 runs are considered, the three genetic versions with local search are superior to their GA, GA_IN and GA_GR counterparts, as shown by the results in tables 2 and 3. Indeed, not only do the MA, MA_IN and MA_GR versions provide a larger relative improvement over the RBS heuristic, but they also give better results for a larger percentage of the test instances, and are seldom inferior to the RBS procedure. Therefore, the addition of a local search procedure improves the average performance, in terms of solution quality, of the genetic algorithms.

The average performance of the three genetic versions that use a local search procedure is quite similar. When the local search is not used, the GA_IN version is slightly inferior to the GA and GA_GR algorithms. When the best result over the 10 runs is considered, the performance of the several genetic algorithms, in terms of solution quality, is quite close. The genetic

versions that incorporate a local search procedure are extremely robust. Indeed, the best and average relative improvement values are usually quite close for the MA, MA_IN and MA_GR algorithms.

The results given in tables 2 and 3 show that the several genetic versions are clearly superior to the RBS and R_V_3 heuristics. Indeed, the genetic algorithms provide a relative improvement of about 1-3% over the RBS procedure. Also, the genetic algorithms give better results for a larger percentage of the test instances. The versions with local search, in particular, give better results for a quite large percentage of the instances, and are seldom inferior to the RBS procedure. The genetic versions also clearly outperform the R_V_3 heuristic. Furthermore, the improvement provided by the genetic algorithms over the existing procedures is increasing with the instance size.

In table 4, we present the effect of the T and R parameters on the relative improvement (calculated with the average objective function value) over the RBS procedure, for instances with 50 jobs. The relative improvement is quite minor for the extreme values of T ($T = 0.0$ and $T = 1.0$). When the tardiness factor assumes more intermediate values, the relative difference in objective function values becomes much larger. Indeed, for some parameter combinations the genetic algorithms provide a relative improvement of over 10%.

Again, this is in accordance with the results obtained in the previous literature on this problem. Indeed, the problem is much easier when most jobs are early ($T = 0.0$) or tardy ($T = 1.0$). Once more, this will also be shown quite clearly in the next section. For the more intermediate values of T, the number of early and tardy jobs becomes more balanced, and the problem becomes harder. Hence, there is more room for improvement in the harder instances with intermediate values of the tardiness factor. Therefore, the genetic versions provide a large relative improvement for the more difficult instances. In fact, as previously mentioned, for the high variability instances with an intermediate value of the tardiness factor T, the genetic algorithms can provide an improvement of over 10%.

The heuristic runtimes (in seconds) are presented in table 5. For the

genetic algorithms and the R_V_3 procedure, we provide the average runtime, i.e. the average of the runtimes for each of the 10 runs. The R_V_3 algorithm is quite clearly the fastest and most efficient of the considered heuristics. The RBS procedure is more computationally demanding, but is faster than the genetic algorithms. Nevertheless, the genetic procedures, with the exception of the GA version, are still somewhat efficient, since they are capable of solving instances with 100 jobs in about 2 seconds.

The GA procedure is considerably more computationally demanding than the other genetic versions. Indeed, the other more sophisticated versions are much faster, even though they perform an initialization of the initial population, and/or generate greedy randomized solutions and/or use a local search procedure. This is due to the lower values required for the parameters pop_mult and stop_iter, as previously mentioned. In terms of solution quality, all the genetic versions were virtually identical when the best result was considered, while the versions with local search were somewhat superior in average performance. However, in terms of computation effort, the more sophisticated versions, particularly those with local search, are clearly more efficient, and can then obtain similar or better results in less computation time.

The MA_GR version is then the recommended heuristic for small and medium instance sizes. This procedure provides the best results (along with the other versions with local search), and is the most efficient of the genetic algorithms, with the exception of the GA_IN version. For large problems, however, a genetic algorithm approach will require excessive time. The RBS procedure can be applied to slightly larger instances than the genetic algorithm, but for the quite large problems only the R_V_3 algorithm (or eventually a dispatching heuristic) will be able to provide results in reasonable computation times.

3.4 Comparison with optimum results

In this section, we compare the heuristic procedures with the optimum objective function values, for instances with up to 20 jobs. Table 6 gives the

average of the relative deviations from the optimum (%dev), calculated as $(H - O)/O \times 100$, where H and O are the heuristic and the optimum objective function values, respectively. The percentage number of times each heuristic generates an optimum schedule (%opt) is also provided.

The results given in table 6 confirm that, as mentioned in the previous section, the problem is much more difficult when the processing time and penalty variability is high. For the low variability instances, all the heuristic procedures provide optimum results for nearly all instances. In fact, the MA_GR algorithm actually generates an optimal solution for all these instances.

When the variability is high, however, the problem becomes harder, and there is more room to improve upon the RBS and R_V_3 results. Indeed, the genetic algorithms clearly outperform these procedures for the high variability instances. This is particularly evident for the better performing versions with local search. In fact, these versions provide an optimal solution for over 96% of the test instances. The GA, GA_IN and GA_GR algorithms also perform quite well. Indeed, not only is their relative deviation from the optimum extremely low, but they also provide optimal solutions for about 80-90% of the instances.

In table 7, we present the effect of the T and R parameters on the relative deviation from the optimum, for instances with 20 jobs. Again, the results given in this table confirm that, as previously mentioned, the problem is harder when there is a greater balance between the number of early and tardy jobs. In fact, when $T \leq 0.2$ or $T \geq 0.8$, all the heuristics are optimal or nearly optimal. The problem, however, becomes harder when $T = 0.4$ or $T = 0.6$, particularly when the due date range is low. For these more difficult instances, the improvement the genetic algorithms provide over the RBS and R_V_3 heuristics is much higher. Indeed, for these more difficult instances, the genetic procedures are much closer to the optimum (particularly the versions with local search, which are nearly always optimal) than the existing procedures.

4 Conclusion

In this paper, a genetic approach was proposed for the single machine scheduling problem with quadratic earliness and tardiness costs, and no machine idle time. Several genetic algorithms based on this approach were presented. These versions differ on the generation of both the initial population and the individuals added in the migration step, as well as on the use of local search.

We first performed initial experiments, in order to determine appropriate values for the parameters required by the genetic algorithms. The proposed procedures were then compared with the best existing heuristics, as well as with optimal solutions for the smaller instance sizes.

The genetic algorithms, particularly the versions with local search, clearly outperform the existing procedures. Also, the genetic heuristics are quite close to the optimum. Indeed, the versions with local search provided an optimal solution for over 96% (and in some cases actually all) of the test instances. The improvement in performance provided by the genetic algorithms is much larger for the more difficult instances, i.e. instances with a high processing time and penalty variability and a greater balance between the number of early and tardy jobs. Also, the improvement given by the genetic versions increases with the instance size.

The performance of the proposed genetic approach was improved by the initialization of the initial population, the generation of greedy randomized solutions and the addition of a local search procedure. In terms of solution quality, all the genetic versions were virtually identical when the best result was considered, while the versions with local search were somewhat superior in average performance. However, in terms of computational effort, the more sophisticated versions, particularly those which use a local search procedure, are clearly more efficient, and can then obtain similar or better results in less computation time. Therefore, the time required by the additional elements included in the more sophisticated versions is more than offset by the fact that they require smaller populations and/or a lower number of iterations without improvement.

The MA_GR algorithm is the new heuristic of choice for small and

medium instance sizes. Indeed, and on the one hand, this procedure provided the best results in terms of solution quality, along with the other genetic versions with local search. Also, and on the other hand, this is the most efficient of the genetic algorithms, with the exception of the GA_IN version.

References

Abdul-Razaq, T. & C. N. Potts (1988). Dynamic programming state-space relaxation for single machine scheduling. Journal of the Operational Research Society, 39, 141–152.

Ahuja, R. K. & J. B. Orlin (1997). Developing fitter GAs. INFORMS Journal on Computing, 9, 251–253.

Ahuja, R. K., J. B. Orlin & A. Tiwari (2000). A greedy genetic algorithm for the quadratic assignment problem. Computers & Operations Research, 27, 917–934.

Armony, M., J. C. Klincewicz, H. Luss & M. B. Rosenwein (2000). Design of stacked self-healing rings using a genetic algorithm. Journal of Heuristics, 6, 85–105.

Baker, K. R. & G. D. Scudder (1990). Sequencing with earliness and tardiness penalties: A review. Operations Research, 38, 22–36.

Bean, J. C. (1994). Genetics and random keys for sequencing and optimization. ORSA Journal on Computing, 6, 154–160.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Massachusetts: Addison-Wesley.

Gonçalves, J. F. (2007). A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. European Jorunal of Operational Research, 183, 1212–1229.

Gonçalves, J. F., J. J. M. Mendes & M. G. C. Resende (2005). A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research, 167, 77–95.

Gupta, S. K. & T. Sen (1983). Minimizing a quadratic function of job lateness on a single machine. Engineering Costs and Production Economics, 7, 187–194.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, Michigan: University of Michigan Press (re-issued in 1992 by MIT Press).

Hoogeveen, H. (2005). Multicriteria scheduling. European Journal of Operational Research, 167, 592–623.

Kanet, J. J. & V. Sridharan (2000). Scheduling with inserted idle time: Problem taxonomy and literature review. Operations Research, 48, 99–110.

Korman, K. (1994). A pressing matter. Video, 46–50.

Landis, K. (1993). Group technology and cellular manufacturing in the Westvaco Los Angeles VH department. Project report in IOM 581, School of Business, University of Southern California.

Li, G. (1997). Single machine earliness and tardiness scheduling. European Journal of Operational Research, 96, 546–558.

Liaw, C.-F. (1999). A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. Computers & Operations Research, 26, 679–693.

Moscato, P. & C. Cotta (2003). A gentle introduction to memetic algotihms. In: F. Glover & G. A. Kochenberger (editors), Handbook of Metaheuristics. Dordrecht: Kluwer Academic Publishers, (pp. 105–144).

Ow, P. S. & T. E. Morton (1989). The single machine early/tardy problem. Management Science, 35, 177–191.

Reeves, C. (2003). Genetic algorithms. In: F. Glover & G. A. Kochenberger (editors), Handbook of Metaheuristics. Dordrecht: Kluwer Academic Publishers, (pp. 55–82).

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. Computers & Operations Research, 22, 5–13.

Reeves, C. R. (1997). Genetic algorithms for the operations researcher. INFORMS Journal on Computing, 9, 231–250.

Schaller, J. (2002). Minimizing the sum of squares lateness on a single machine. European Journal of Operational Research, 143, 64–79.

Schaller, J. (2004). Single machine scheduling with early and quadratic tardy penalties. Computers & Industrial Engineering, 46, 511–532.

Sen, T., P. Dileepan & M. R. Lind (1995). Minimizing a weighted quadratic function of job lateness in the single machine system. International Journal of Production Economics, 42, 237–243.

Spears, W. M. & K. A. De Jong (1991). On the virtues of parameterized uniform crossover. In: R. Belew & L. Booker (editors), Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufman, (pp. 230–236).

Su, L.-H. & P.-C. Chang (1998). A heuristic to minimize a quadratic function of job lateness on a single machine. International Journal of Production Economics, 55, 169–175.

Valente, J. M. S. (2007a). An exact approach for single machine scheduling with quadratic earliness and tardiness penalties. Working Paper 238, Faculdade de Economia, Universidade do Porto, Portugal.

Valente, J. M. S. (2007b). Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. European Journal of Industrial Engineering, 1, 431–448.

Valente, J. M. S. (2008a). Beam search heuristics for quadratic earliness and tardiness scheduling. Working Paper 279, Faculdade de Economia, Universidade do Porto, Portugal.

Valente, J. M. S. (2008b). An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties. Asia-Pacific Journal of Operational Research, 25, 169–186.

Valente, J. M. S. & R. A. F. S. Alves (2005a). Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. Computers & Industrial Engineering, 48, 363–375.

Valente, J. M. S. & R. A. F. S. Alves (2005b). Improved heuristics for the early/tardy scheduling problem with no idle time. Computers & Operations Research, 32, 557–569.

Valente, J. M. S. & R. A. F. S. Alves (2005c). Improved lower bounds for the early/tardy scheduling problem with no idle time. Journal of the Operational Research Society, 56, 604–612.

Valente, J. M. S. & R. A. F. S. Alves (2008). Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. Computers & Operations Research, 35, 3696–3713.

Valente, J. M. S. & J. F. Gonçalves (2008). A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. Working Paper 264, Faculdade de Economia, Universidade do Porto, Portugal.

Valente, J. M. S., J. F. Gonçalves & R. A. F. S. Alves (2006). A hybrid genetic algorithm for the early/tardy scheduling problem. Asia-Pacific Journal of Operational Research, 23, 393–405.

Valente, J. M. S. & M. R. A. Moreira (2008). Greedy randomized dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. Working Paper 286, Faculdade de Economia, Universidade do Porto, Portugal.

Wagner, B. J., D. J. Davis & H. Kher (2002). The production of several items in a single facility with linearly changing demand rates. Decision Sciences, 33, 317–346.

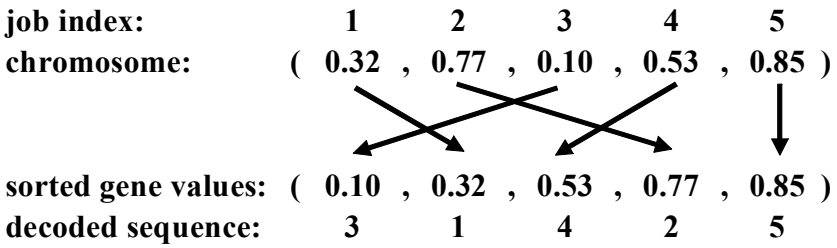

Figure 1: Chromosome decoding example

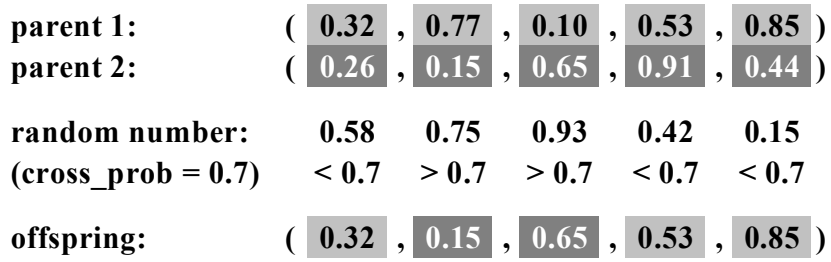

Figure 2: Parameterized uniform crossover example

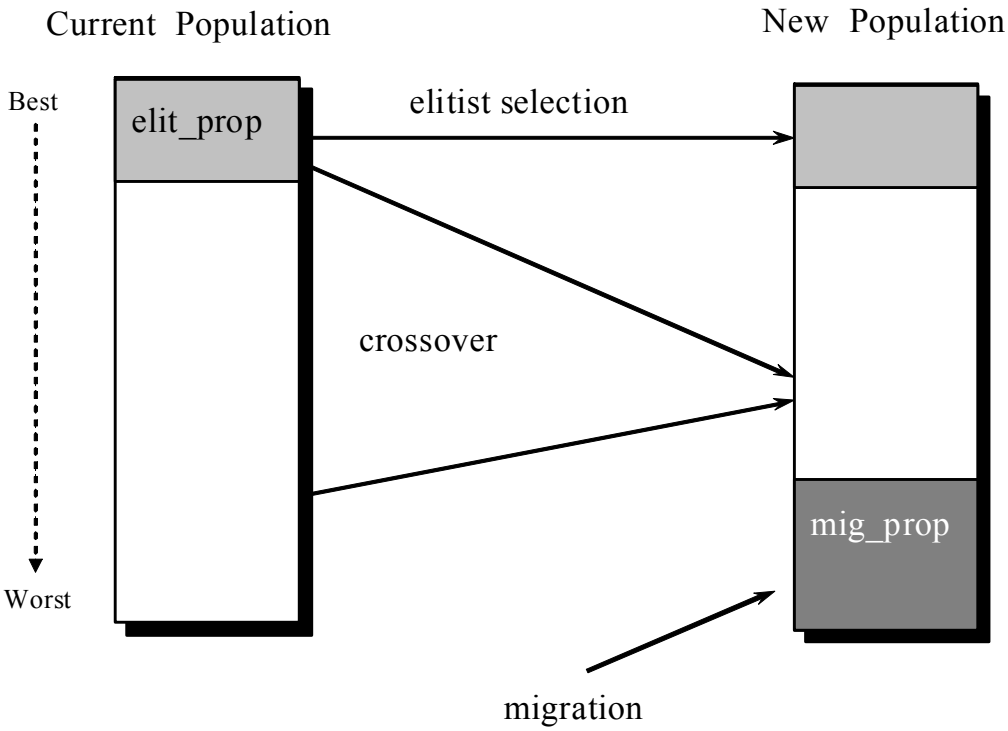

Figure 3: Evolutionary Strategy

	GA	GA_IN	GA_GR	MA	MA_IN	MA_GR
pop_mult	3	2	2	1	1	1
elit_prop	0.05	0.05	0.05	0.05	0.05	0.05
mig_prop	0.25	0.25	0.25	0.25	0.25	0.25
cross_prob	0.8	0.8	0.8	0.8	0.8	0.8
stop_iter	30	30	30	10	10	10
init_gr	—	—	0.4	—	—	0.1
mig_gr	—	—	0.5	—	—	0.5
local search	—	—	—	API	API	API

Table 1: Parameter values

```
Genetic approach
{
        Generate Initial Population P_t = P_0;
        Evaluate Population P_0;
        Update Best Solution;
        Set iter_no_improv = 0;

        While (iter_no_improv < stop_iter)
        {
                Generate New Population P_t+1
                {
                        Perform Elitist Selection;
                        Perform Migration;
                        Perform Crossover;
                }

                Evaluate P_t+1;

                If (new best solution is found)
                {
                        Update Best Solution;
                        Set iter_no_improv = 0;
                }
                Else
                        Set iter_no_improv = iter_no_improv + 1;

                Set P_t = P_t+1;
        }
}
```

Figure 4: Genetic Approach

		$n = 25$		$n = 50$		$n = 75$		$n = 100$	
var	heur	best	avg	best	avg	best	avg	best	avg
L	R_V_3	0.0000	-0.0002	-0.0002	-0.0003	-0.0001	-0.0001	-0.0003	-0.0003
	GA	0.0000	-0.0004	0.0002	0.0000	0.0001	0.0000	—	—
	GA_IN	0.0000	-0.0006	0.0001	-0.0003	0.0000	-0.0002	0.0000	-0.0001
	GA_GR	0.0000	-0.0006	0.0001	-0.0002	0.0001	-0.0001	0.0000	-0.0001
	MA	0.0000	0.0000	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001
	MA_IN	0.0000	0.0000	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001
	MA_GR	0.0000	0.0000	0.0002	0.0002	0.0001	0.0001	0.0001	0.0001
H	R_V_3	0.1463	-0.8608	0.3346	-0.2810	0.4328	-0.2006	0.1056	-0.2668
	GA	1.3646	0.8392	2.2310	2.0569	2.4296	2.3216	—	—
	GA_IN	1.2580	0.5794	2.1925	1.8719	2.3971	2.1718	2.7643	2.6584
	GA_GR	1.2856	0.9202	2.1972	1.9714	2.4123	2.2653	2.7603	2.6033
	MA	1.3673	1.2935	2.2345	2.1961	2.4337	2.4055	2.7828	2.7678
	MA_IN	1.3675	1.2987	2.2337	2.2012	2.4336	2.4115	2.7822	2.7554
	MA_GR	1.3653	1.2837	2.2340	2.1800	2.4327	2.3964	2.7828	2.7681

Table 2: Comparison with the RBS heuristic - relative improvement

		$n = 25$			$n = 50$			$n = 75$			$n = 100$		
var	heur	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$	$<$	$=$	$>$
L	R_V_3	0.6	98.8	0.6	2.4	95.9	1.7	6.1	92.1	1.8	2.7	88.3	9.1
	GA	0.4	97.6	2.0	1.8	93.2	5.0	4.0	90.5	5.5	—	—	—
	GA_IN	0.4	97.1	2.5	1.2	93.3	5.5	1.9	92.2	5.9	2.5	88.6	8.9
	GA_GR	0.4	97.4	2.2	1.4	93.7	5.0	2.5	92.5	5.1	3.7	88.8	7.5
	MA	0.7	99.3	0.0	2.9	97.1	0.1	7.1	92.9	0.0	10.5	89.5	0.0
	MA_IN	0.7	99.3	0.0	2.9	97.1	0.0	7.1	92.9	0.0	10.5	89.5	0.0
	MA_GR	0.7	99.3	0.1	2.9	97.1	0.0	7.1	92.9	0.0	10.4	89.6	0.0
H	R_V_3	10.8	75.7	13.5	23.0	57.6	19.4	34.4	43.2	22.4	31.8	40.2	28.0
	GA	18.5	63.1	18.3	35.5	43.6	20.9	46.0	30.8	23.3	—	—	—
	GA_IN	15.1	66.9	18.1	30.2	48.9	20.9	37.6	42.0	20.4	41.5	40.7	17.8
	GA_GR	16.8	75.5	7.8	35.8	52.1	12.2	45.8	40.4	13.7	49.2	37.9	12.9
	MA	22.3	75.6	2.1	42.9	54.8	2.3	56.9	40.9	2.2	64.5	33.9	1.6
	MA_IN	22.3	75.8	1.9	43.0	54.9	2.2	56.9	41.2	1.9	64.3	34.2	1.5
	MA_GR	22.3	76.6	1.2	42.9	55.5	1.6	57.2	41.3	1.5	64.3	34.6	1.1

Table 3: Comparison with the RBS heuristic - percentage of better, equal and worse results

		low var				high var			
heur	T	R=0.2	R=0.4	R=0.6	R=0.8	R=0.2	R=0.4	R=0.6	R=0.8
R_V_3	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	-0.01	-0.05	-0.02	-0.01
	0.4	0.00	0.00	0.00	0.00	-0.79	-0.32	-1.17	-1.17
	0.6	0.00	0.00	0.00	0.00	-1.34	-1.63	-0.71	-0.05
	0.8	0.00	0.00	0.00	0.00	0.42	0.13	-0.02	-0.02
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
GA_IN	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	-0.04	-0.06	-0.04	-0.01
	0.4	0.00	0.00	0.00	0.00	4.30	1.37	0.06	-0.55
	0.6	0.00	0.00	0.00	0.00	16.45	10.12	7.02	2.33
	0.8	0.00	0.00	0.00	0.00	3.53	0.45	-0.01	0.00
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GA_GR	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.04	-0.01	0.00	0.01
	0.4	0.00	0.00	0.00	0.00	4.41	1.63	0.60	-0.09
	0.6	0.00	0.00	0.00	0.00	16.43	10.13	7.20	2.84
	0.8	0.00	0.00	0.00	0.00	3.59	0.49	0.02	0.02
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
MA_IN	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.10	0.01	0.00	0.02
	0.4	0.00	0.00	0.00	0.00	5.24	2.25	1.13	0.27
	0.6	0.00	0.00	0.00	0.00	16.96	10.91	8.28	3.42
	0.8	0.00	0.00	0.00	0.00	3.64	0.51	0.04	0.03
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
MA_GR	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.02
	0.4	0.00	0.00	0.00	0.00	5.07	2.18	1.02	0.28
	0.6	0.00	0.00	0.00	0.00	16.92	10.91	8.21	3.36
	0.8	0.00	0.00	0.00	0.00	3.64	0.51	0.04	0.03
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00

Table 4: Relative improvement over the RBS heuristic for instances with 50 jobs

var	heur	$n = 15$	$n = 25$	$n = 50$	$n = 75$	$n = 100$
L	RBS	0.002	0.007	0.037	0.104	0.226
	R_V_3	0.001	0.003	0.008	0.017	0.020
	GA	0.032	0.129	0.923	3.007	—
	GA_IN	0.008	0.019	0.068	0.149	0.262
	GA_GR	0.011	0.040	0.163	0.467	0.972
	MA	0.010	0.030	0.214	0.691	1.656
	MA_IN	0.009	0.029	0.209	0.682	1.637
	MA_GR	0.009	0.024	0.154	0.484	1.139
H	RBS	0.002	0.007	0.038	0.109	0.240
	R_V_3	0.002	0.004	0.013	0.030	0.034
	GA	0.033	0.131	0.937	3.046	—
	GA_IN	0.011	0.032	0.159	0.441	0.982
	GA_GR	0.014	0.065	0.488	2.110	3.647
	MA	0.010	0.032	0.253	0.907	2.362
	MA_IN	0.010	0.031	0.240	0.857	2.258
	MA_GR	0.009	0.026	0.183	0.631	1.523

Table 5: Runtimes (in seconds)

		$n = 10$		$n = 15$		$n = 20$	
var	heur	%dev	%opt	%dev	%opt	%dev	%opt
L	RBS	0.0000	99.92	0.0000	100.00	0.0000	99.67
	R_V_3	0.0020	99.65	0.0000	99.75	0.0000	99.64
	GA	0.0000	99.40	0.0000	98.93	0.0000	98.30
	GA_IN	0.0000	99.27	0.0002	98.60	0.0001	97.58
	GA_GR	0.0000	99.63	0.0000	99.03	0.0000	98.24
	MA	0.0000	100.00	0.0000	100.00	0.0000	99.99
	MA_IN	0.0000	100.00	0.0000	100.00	0.0000	99.98
	MA_GR	0.0000	100.00	0.0000	100.00	0.0000	100.00
H	RBS	0.2211	95.67	0.9067	87.92	1.3971	82.50
	R_V_3	0.2938	93.12	0.7071	85.74	1.0254	80.36
	GA	0.0010	90.48	0.0001	82.77	0.0024	77.57
	GA_IN	0.1597	90.10	0.1502	81.11	0.1310	76.64
	GA_GR	0.0440	94.66	0.0596	88.48	0.0475	85.44
	MA	0.0000	99.97	0.0000	98.61	0.0000	96.44
	MA_IN	0.0000	99.99	0.0000	98.90	0.0000	96.63
	MA_GR	0.0000	99.99	0.0000	98.91	0.0002	96.82

Table 6: Comparison with optimum objective function values

		low var				high var			
heur	T	R=0.2	R=0.4	R=0.6	R=0.8	R=0.2	R=0.4	R=0.6	R=0.8
RBS	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00
	0.4	0.00	0.00	0.00	0.00	3.88	6.22	0.68	0.80
	0.6	0.00	0.00	0.00	0.00	10.94	3.63	2.64	0.74
	0.8	0.00	0.00	0.00	0.00	2.61	1.05	0.03	0.17
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
R_V_3	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.09	0.03	0.01	0.01
	0.4	0.00	0.00	0.00	0.00	3.24	5.27	0.58	1.77
	0.6	0.00	0.00	0.00	0.00	7.89	2.68	2.12	0.50
	0.8	0.00	0.00	0.00	0.00	0.13	0.28	0.00	0.00
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GA_IN	0.0	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.19	0.03	0.08	0.02
	0.4	0.00	0.00	0.00	0.00	0.31	0.32	0.63	0.96
	0.6	0.00	0.00	0.00	0.00	0.22	0.08	0.08	0.04
	0.8	0.00	0.00	0.00	0.00	0.03	0.03	0.00	0.10
	1.0	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01
GA_GR	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.09	0.00	0.01	0.01
	0.4	0.00	0.00	0.00	0.00	0.12	0.39	0.14	0.01
	0.6	0.00	0.00	0.00	0.00	0.22	0.07	0.01	0.00
	0.8	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MA_IN	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MA_GR	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.4	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
	0.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 7: Relative deviation from the optimum for instances with 20 jobs

Recent FEP Working Papers

Nº 311	Abel Costa Fernandes, "*Explaining Government Spending: a Cointegration Approach"*, February 2009
Nº 310	João Correia-da-Silva, "*Uncertain delivery in markets for lemons"*, January 2009
Nº 309	Ana Paula Ribeiro, "*Interactions between Labor Market Reforms and Monetary Policy under Slowly Changing Habits"*, January 2009
Nº 308	Argentino Pessoa and Mário Rui Silva, "*Environment Based Innovation: Policy Questions"*, January 2009
Nº 307	Inês Drumond and José Jorge, "*Basel II Capital Requirements, Firms' Heterogeneity, and the Business Cycle"*, January 2009
Nº 306	Adelaide Maria Figueiredo, Fernanda Otília Figueiredo and Natália Pimenta Monteiro, "*Labor adjustments in privatized firms: a Statis approach"*, December 2008
Nº 305	Manuela A. D. Aguiar and Sofia B. S. D. Castro, "*Chaotic and deterministic switching in a two-person game"*, December 2008
Nº 304	Ana Pinto Borges and João Correia-da-Silva, "*Using Cost Observation to Regulate Bureaucratic Firms"*, December 2008
Nº 303	Miguel Fonseca, "*The Investment Development Path Hypothesis: a Panel Data Approach to the Portuguese Case"*, December 2008
Nº 302	Alexandre Almeida, Cristina Santos and Mário Rui Silva, "*Bridging Science to Economy: The Role of Science and Technologic Parks in Innovation Strategies in "Follower" Regions"*, November 2008
Nº 301	Alexandre Almeida, António Figueiredo and Mário Rui Silva, "*From Concept to Policy: Building Regional Innovation Systems in Follower Regions"*, November 2008
Nº 300	Pedro Quelhas Brito, "*Conceptualizing and illustrating the digital lifestyle of youth"*, October 2008
Nº 299	Argentino Pessoa, "*Tourism and Regional Competitiveness: the Case of the Portuguese Douro Valley"*, October 2008
Nº 298	Aurora A.C. Teixeira and Todd Davey, "*Attitudes of Higher Education students to new venture creation: a preliminary approach to the Portuguese case"*, October 2008
Nº 297	Carlos Brito, "*Uma Abordagem Relacional ao Valor da Marca"*, October 2008
Nº 296	Pedro Rui M. Gil, Paulo Brito and Óscar Afonso, "*A Model of Quality Ladders with Horizontal Entry"*, October 2008
Nº 295	Maria Manuel Pinho, "*The political economy of public spending composition: evidence from a panel of OECD countries"*, October 2008
Nº 294	Pedro Cosme da Costa Vieira, "*O Subsídio de Desemprego e a Relação Negativa entre Salário e Risco de Falência: Uma Teoria em Equilíbrio Parcial"*, October 2008
Nº 293	Cristina Santos, Alexandre Almeida and Aurora A.C. Teixeira, "*Searching for clusters in tourism. A quantitative methodological proposal"*, September 2008
Nº 292	Alexandre Almeida and Aurora A.C. Teixeira, "*One size does not fit all… An economic development perspective on the asymmetric impact of Patents on R&D"*, September 2008
Nº 291	Paula Neto, António Brandão and António Cerqueira, "*The Impact of FDI, Cross Border Mergers and Acquisitions and Greenfield Investments on Economic Growth"*, September 2008
Nº 290	Cosme, P., "*Integrating fire risk into the management of forests"*, September 2008
Nº 289	Cosme, P., "*A comment on efficiency gains and myopic antitrust authority in a dynamic merger game"*, September 2008
Nº 288	Moreira, R., "*Workart – A Gestão e a Arte"* (1st Prize of the 2nd Edition of FEP/AEFEP- Applied Research in Economics and Management), August 2008
Nº 287	Vasco Leite, Sofia B.S.D. Castro and João Correia-da-Silva, "*The core periphery model with asymmetric inter-regional and intra-regional trade costs"*, August 2008
Nº 286	Jorge M. S. Valente and Maria R. A. Moreira, "*Greedy randomized dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties"*, August 2008

Nº 285	Patricia Teixeira Lopes and Rui Couto Viana, "*The transition to IFRS: disclosures by Portuguese listed companies"*, August 2008
Nº 284	Argentino Pessoa, "*Educational Reform in Developing Countries: Private Involvement and Partnerships"*, July 2008
Nº 283	Pedro Rui Mazeda Gil and Óscar Afonso, "*Technological-Knowledge Dynamics in Lab-Equipment Models of Quality Ladders"*, July 2008
Nº 282	Filipe J. Sousa and Luís M. de Castro, "*How is the relationship significance brought about? A critical realist approach"*, July 2008
Nº 281	Paula Neto; António Brandão and António Cerqueira, "*The Macroeconomic Determinants of Cross Border Mergers and Acquisitions and Greenfield Investments"*, June 2008
Nº 280	Octávio Figueiredo, Paulo Guimarães and Douglas Woodward, "*Vertical Disintegration in Marshallian Industrial Districts"*, June 2008
Nº 279	Jorge M. S. Valente, "*Beam search heuristics for quadratic earliness and tardiness scheduling"*, June 2008
Nº 278	Nuno Torres and Óscar Afonso, "*Re-evaluating the impact of natural resources on economic growth"*, June 2008
Nº 277	Inês Drumond, "*Bank Capital Requirements, Business Cycle Fluctuations and the Basel Accords: A Synthesis"*, June 2008
Nº 276	Pedro Rui Mazeda Gil, "*Stylized Facts and Other Empirical Evidence on Firm Dynamics, Business Cycle and Growth"*, May 2008
Nº 275	Teresa Dieguez and Aurora A.C. Teixeira, "*ICTs and Family Physicians Human Capital Upgrading. Delightful Chimera or Harsh Reality?"*, May 2008
Nº 274	Teresa M. Fernandes, João F. Proença and P.K. Kannan, "*The Relationships in Marketing: Contribution of a Historical Perspective"*, May 2008
Nº 273	Paulo Guimarães, Octávio Figueiredo and Douglas Woodward, "*Dartboard Tests for the Location Quotient"*, April 2008
Nº 272	Rui Leite and Óscar Afonso, "*Effects of learning-by-doing, technology-adoption costs and wage inequality"*, April 2008
Nº 271	Aurora A.C. Teixeira, "*National Systems of Innovation: a bibliometric appraisal"*, April 2008
Nº 270	Tiago Mata, "*An uncertain dollar: The Wall Street Journal, the New York Times and the monetary crisis of 1971 to 1973"*, April 2008
Nº 269	João Correia-da-Silva and Carlos Hervés-Beloso, "*General equilibrium with private state verification"*, March 2008
Nº 268	Carlos Brito, "*Relationship Marketing: From Its Origins to the Current Streams of Research"*, March 2008
Nº 267	Argentino Pessoa, "*Kuznets's Hypothesis And The Data Constraint"*, February 2008
Nº 266	Argentino Pessoa, "*Public-Private Sector Partnerships In Developing Countries: Are Infrastructures Responding To The New Oda Strategy"*, February 2008
Nº 265	Álvaro Aguiar and Ana Paula Ribeiro, "*Why Do Central Banks Push for Structural Reforms? The Case of a Reform in the Labor Market"*, February 2008
Nº 264	Jorge M. S. Valente and José Fernando Gonçalves, "*A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties"*, January 2008

Editor: Sandra Silva (sandras@fep.up.pt)
Download available at:
http://www.fep.up.pt/investigacao/workingpapers/workingpapers.htm
also in http://ideas.repec.org/PaperSeries.html

www.fep.up.pt

Faculdade de Economia da Universidade do Porto
Rua Dr. Roberto Frias, 4200-464 Porto | Tel. 225 571 100
Tel. 225571100 | www.fep.up.pt