FEP WORKING PAPERS
FEP WORKING PAPERS

RESEARCH WORK IN PROGRESS

N. 250, OCT. 2007

BEAM SEARCH HEURISTICS FOR THE SINGLE MACHINE SCHEDULING PROBLEM WITH LINEAR EARLINESS AND QUADRATIC TARDINESS COSTS

JORGE M. S. VALENTE

LIAAD AND FACULDADE DE ECONOMIA,
UNIVERSIDADE DO PORTO

U.PORTO
FEP FACULDADE DE ECONOMIA UNIVERSIDADE DO PORTO

Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs

Jorge M. S. Valente*

LIAAD, Faculdade de Economia, Universidade do Porto, Portugal

October 3, 2007

Abstract

In this paper, we consider the single machine scheduling problem with linear earliness and quadratic tardiness costs, and no machine idle time. We present heuristic algorithms based on the beam search technique. These algorithms include classic beam search procedures, as well as the filtered and recovering variants. Several dispatching rules are considered as evaluation functions, in order to analyse the effect of different rules on the effectiveness of the beam search algorithms.

The computational results show that using better rules indeed improves the performance of the beam search heuristics. The detailed, filtered and recovering beam search procedures outperform the best

*Address: Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal. E-mail: jvalente@fep.up.pt.

existing heuristic. The best results are given by the recovering and detailed variants, which provide objective function values that are quite close to the optimum. For small to medium size instances, either of these procedures can be used. For larger instances, however, the detailed beam search algorithm requires excessive computation times, and the recovering beam search procedure then becomes the heuristic of choice.

Keywords: scheduling, single machine, linear earliness, quadratic tardiness, beam search, heuristics

1 Introduction

In this paper, we consider a single machine scheduling problem with linear earliness and quadratic tardiness costs, and no machine idle time. Formally, the problem can be stated as follows. A set of n independent jobs $\{J_1, J_2, \cdots, J_n\}$ 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 $J_j, j = 1, 2, \cdots, n$, requires a processing time p_j and should ideally be completed on its due date d_j. For a given schedule, the earliness and tardiness of J_j are defined as $E_j = \max\{0, d_j - C_j\}$ and $T_j = \max\{0, C_j - d_j\}$, respectively, where C_j is the completion time of J_j. The objective is to find a schedule that minimizes the sum of linear earliness and quadratic tardiness costs $\sum_{j=1}^{n} \left(E_j + T_j^2\right)$, subject to the constraint that no machine idle time is allowed.

Single machine scheduling environments actually occur in many practical operations (for a recent example in the chemical industry, see Wagner et al. (2002)). Moreover, the performance of many production systems is frequently determined by the quality of the schedules for a single bottleneck machine. Single processor models are then most useful in practice for scheduling such a machine. Also, the analysis of single machine problems provides results and insights that can often be applied to more complex scheduling environments. Indeed, multiple processor environments can often be relaxed to a single machine problem, or a sequence of such problems. Furthermore, the solution procedures for some complex systems (e.g., job shops) often require solving subproblems with a single processor.

Scheduling models with both earliness and tardiness costs are compatible with the just-in-time (JIT) production philosophy. The JIT approach focuses on producing goods only when they are needed, and therefore considers that both earliness and tardiness should be discouraged. Earliness/tardiness models are also compatible with the recent adoption of supply chain management by many organisations. This approach seeks to improve the efficiency of the supply chain, and to provide a better service to the end user, by integrating the flow of materials from suppliers to customers. The adoption of supply chain management has caused organisations to view early deliveries, in addition to tardy deliveries, as undesirable.

Linear earliness and quadratic tardiness costs are considered in this paper. On the one hand, early completions of jobs result in unnecessary inventory. The costs of maintaining and managing this inventory tend to be proportional to the quantity held in stock, and therefore a linear penalty is used for

early jobs. On the other hand, late deliveries can result in lost sales and loss of goodwill, as well as disruptions in stages further down the supply chain. In this paper, a quadratic penalty is considered for the tardy jobs. A quadratic tardiness penalty is appropriate in practice. Indeed, the tardiness is an important attribute of service quality. Also, a customer's dissatisfaction tends to increase quadratically with the tardiness, as proposed in the loss function of Taguchi (1986). Moreover, a quadratic tardiness penalty can in some situations be preferable to the more usual linear tardiness or maximum tardiness functions, as discussed in Sun et al. (1999).

We assume that machine idle time is not allowed. This assumption is 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 meet the customers' orders. Also, the assumption of no idle time is justified when the machine has high operating costs, and when starting a new production run involves large setup costs or times. Some specific examples of production settings where the no idle time assumption is appropriate have been given by Korman (1994) and Landis (1993).

This problem has been previously considered by Valente (to appear, 2006). Valente (to appear) proposed a lower bounding procedure based on a relaxation of the job completion times, as well as a branch-and-bound procedure. In Valente (2006), several dispatching heuristics are presented, and their performance is analysed on a wide range of instance types. The corresponding problem with inserted idle time has been considered by Schaller (2004). He presented a timetabling procedure to optimally insert idle time in a given sequence, as well as a branch-and-bound procedure and simple and

efficient heuristics.

The single machine problem with linear earliness and tardiness penalties $\sum_{j=1}^{n}(E_j + T_j)$ has also been previously considered by Garey et al. (1988), Kim and Yano (1994) and Schaller (2007). Garey et al. (1988) showed that the problem is NP-hard, and proposed a timetabling procedure. Several properties of optimal solutions were presented by Kim and Yano (1994), and used to develop optimal and heuristic algorithms. Schaller (2007) develops a new lower bound and a new dominance condition, and also shows how to strengthen the lower bounds proposed by Kim and Yano (1994).

The minimization of the quadratic lateness $\sum_{j=1}^{n} L_j^2$, where the lateness of J_j is defined as $L_j = C_j - d_j$, has also been previously considered. Gupta and Sen (1983) proposed both a branch-and-bound algorithm and a heuristic rule for the problem with no idle time. Su and Chang (1998) and Schaller (2002) considered inserted idle time, and proposed timetabling procedures and heuristic algorithms. Sen et al. (1995) presented a branch-and-bound procedure for the weighted problem $\sum_{j=1}^{n} w_j L_j^2$ where idle time is allowed only prior to the start of the first job. Baker and Scudder (1990) and Hoogeveen (2005) provide excellent surveys of scheduling problems with earliness and tardiness penalties. Kanet and Sridharan (2000) give a review of scheduling models with inserted idle time that complements our focus on a problem with no machine idle time.

In this paper, we present several heuristic algorithms based on the beam search technique. These algorithms include classic beam search procedures, with both priority and total cost evaluation functions, as well as the more recent filtered and recovering variants. Beam search procedures require eval-

uation functions that are usually provided by a dispatching rule. We consider four dispatching heuristics, in order to analyse the effect of different rules on the performance of the beam search algorithms. Extensive preliminary computational experiments were performed to determine appropriate values for the parameters required by the beam search procedures. The performance of the four heuristic rules is also analysed in these initial tests. The best-performing versions of the beam search algorithms are then compared with the best existing heuristic, as well as with optimal solutions.

The remainder of this paper is organized as follows. In section 2, we describe the beam search approach and its several variations, and present the proposed heuristic procedures and their implementation details. The computational results are reported in section 3. Finally, some concluding remarks are given in section 4.

2 The beam search heuristics

2.1 History and review

Beam search is a heuristic method for solving combinatorial optimization problems. It consists of a truncated branch-and-bound procedure with a breadth-first strategy in which only the most promising nodes at each level of the search tree are kept for further branching. The remaining nodes are pruned off, and no backtracking is performed, so the running time is polynomial in the problem size.

The classic or traditional beam search algorithm was first used in artificial

intelligence problems by Lowerre (1976) and Rubin (1978). Two variations of the classic beam search approach have since been proposed. Ow and Morton (1988, 1989) developed a variation of the beam search technique called filtered beam search. Recently, Della Croce and T'kindt (2002) and Della Croce et al. (2004) proposed an approach denoted by recovering beam search.

Beam search algorithms have been applied to several combinatorial optimization problems, particularly in the scheduling field. Some recent applications of beam search heuristics to scheduling problems include Sabuncuoglu and Bayiz (1999), Della Croce and T'kindt (2002), Della Croce et al. (2004), Valente and Alves (2005), Ghirardi and Potts (2005) and Esteve et al. (2006).

In the following subsections, we first present the classic beam search technique. Then, the more recent filtered and recovering approaches are described. Finally, we present the proposed beam search algorithms, as well as some implementation details.

2.2 Classic beam search

The classic beam search method consists of a truncated branch-and-bound algorithm in which only the most promising β nodes at each level of the search tree are selected as nodes to branch from; β is the so-called *beam width*. The remaining nodes are ignored, and backtracking is now allowed. Therefore, classic beam search algorithms cannot recover from wrong decisions, and are not guaranteed to find an optimal solution. A larger beam width allows for greater safety, but at the cost of increased computation time.

The node evaluation process is crucial for the effectiveness of a beam

search procedure. Two different types of evaluation functions have been used in classic beam procedures: *priority evaluation functions* and *total cost evaluation functions*. The priority evaluation functions simply calculate an urgency rating for the last job added to the current partial sequence. This is usually done by using the priority index of a dispatching heuristic. Total cost evaluation functions, on the other hand, calculate an estimate of the minimum total cost of the best solution that can be reached from the current node. A dispatching rule is typically used to schedule the remaining jobs, in order to complete the existing partial schedule. Priority evaluation functions have a local view of the problem, because they only consider the next decision to be made (i.e., the next job to schedule). Total cost evaluation functions, however, have a global view, since they project from the current partial solution to a complete schedule in order to estimate the total cost.

The priority evaluation functions are usually context-dependent, which can pose a slight problem. Indeed, the priority index that is used to calculate the urgency rating of the last scheduled job usually depends on the current partial schedule, particularly on the current time. Different nodes on the same level of the search tree may have different completion times, since they correspond to different partial schedules. Therefore, the priority values are context-dependent, meaning that the priorities calculated for the offspring of one node cannot be legitimately compared with those obtained from the branching of another node. This problem, however, can be solved by initially selecting the best β children of the root node. Then, at lower levels of the search tree, only the best descendant of each beam node is kept for further branching, so the number of beam nodes is kept at β. The total

cost evaluation functions are not affected by this problem. Indeed, the total cost estimates are context-independent, and can be compared for all offspring nodes.

The main steps of priority beam search (PBS) and detailed beam search (DBS) algorithms are now presented. The priority (detailed) beam search procedure uses a priority (total cost) evaluation function. In the following, B is the set of beam nodes , C is a set of offspring nodes and n_0 is the root node.

Priority Beam Search:

Step 1. Initialization:

Set $B = \varnothing$, $C = \varnothing$.

Branch n_0, generating the corresponding children.

Calculate the priority of the last scheduled job for each child node.

Select the best β child nodes and add them to B.

Step 2. Node selection:

For each node in B:

(a) Branch the node, generating the corresponding children.

(b) Calculate the priority of the last scheduled job for each child node.

(c) Select the best child node and add it to C.

Set $B = C$ and $C = \varnothing$.

Step 3. Stopping condition:

If the nodes in B are leaf (i.e., they hold a complete sequence), select the node with the lowest total cost as the best sequence found and stop.

Otherwise, go to step 2.

Detailed Beam Search:

Step 1. Initialization:

Set $B = \{n_0\}$ and $C = \varnothing$.

Step 2. Branching:

For each node in B:

(a) Branch the node, generating the corresponding children.

(b) Calculate an upper bound on the optimal solution value for each child node.

(c) Select the best β child nodes and add them to C.

Set $B = \varnothing$.

Step 3. Node selection:

Select the best β nodes in C and add them to B.

Set $C = \varnothing$.

Step 4. Stopping condition:

If the nodes in B are leaf, select the node with the lowest total cost as the best sequence found and stop.

Otherwise, go to step 2.

2.3 Filtered and recovering beam search

The priority and total cost evaluation functions have opposite advantages and weaknesses. On the one hand, the priority evaluation is quick, but it is rather crude and potentially inaccurate, and may result in discarding good solutions. The total cost evaluation, on the other hand, is more accurate, but much more time consuming. The filtered and recovering beam search algorithms try to

combine crude and accurate evaluations, in order to provide a high quality evaluation, within reasonable computation time. This is achieved by using a two-stage approach. The filtered and recovering algorithms first apply a computationally inexpensive *filtering step*. In this step, a crude evaluation is performed, in order to select only a reduced number of the offspring of each beam node. The selected nodes are then accurately evaluated, and the best β nodes are kept for further branching.

Two different types of filtering step have been proposed. In the approach developed by Ow and Morton (1988, 1989), a priority evaluation function is used to calculate an urgency rating for each offspring. Then, the best α children of each beam node are selected for accurate evaluation, where α is the so-called *filter width*. Recently, in conjunction with the development of the recovering beam search approach, a new type of filtering phase has been introduced by Della Croce and T'kindt (2002) and Della Croce et al. (2004). In this approach, probem-dependent dominance conditions (denoted as valid dominance conditions), when available, are applied together with so-called pseudo-dominance conditions (which hold in a heuristic context only). Whenever a valid dominance condition or a pseudo-dominance condition applies for a given node, that node is pruned. The priority function approach was then originally applied in filtered beam search algorithms, while the dominance conditions filtering procedure was developed for the recovering beam search heuristic. Nevertheless, either of these two filtering procedures can be used in both filtered and recovering algorithms.

The recovering beam search (RBS) approach differs from the filtered beam search (FBS) algorithm in two major ways. First, the accurate evaluation

in the filtered beam search procedure relies on an upper bound on the total cost of the best solution that can be reached from the current node. In the RBS approach, on the other hand, each node is evaluated by both lower and upper bounds. More specifically, each node is evaluated by the function $V = (1 - \gamma)\, LB + \gamma UB$, where $0 \leq \gamma \leq 1$ is a user-defined parameter and LB and UB are the lower and upper bound values, respectively. Therefore, the evaluation function V is a weighted sum of the lower and upper bounds, with the weight of the upper bound being given by the parameter γ.

Second, the RBS algorithm includes a so-called *recovering phase*. This phase is performed after the accurate evaluation of the nodes that passed the filtering step, and it selects the β nodes that will be kept for further branching. In this phase, the candidate nodes are considered in non-decreasing order of their evaluation function. For each node, the recovering step checks whether the current partial solution σ is dominated by another partial solution $\overline{\sigma}$ having the same level of the search tree; this is typically done by applying neighbourhood operators. If a better partial solution $\overline{\sigma}$ does exist, then σ is replaced by $\overline{\sigma}$. If the possibly modified node is not already in the set of beam nodes, then the node is added to B. This process is repeated until either β nodes have been selected, or no additional candidate nodes remain.

The recovering step often allows the RBS procedure to recover from previous incorrect decisions, a feature which is not present in the classic or filtered beam search algorithms. Indeed, in PBS, DBS and FBS procedures, if a node leading to the optimal solution is pruned, there is no way to reach that solution afterwards. The recovering phase tries to overcome this problem by

using neighbourhood operators to search for improved solutions. Note also that, in the recovering step, a partial solution can only be replaced by another partial solution with the same tree level. Therefore, the total number of explored nodes is still polynomial in the problem size. We now present the main steps of both filtered and recovering beam search algorithms. In the RBS algorithm, let n_{best} and UB_{best} denote the current best node and the current best upper bound, respectively.

Filtered Beam Search:

Step 1. Initialization:

Set $B = \{n_0\}$ and $C = \varnothing$.

Step 2. Filtering step:

For each node in B:

(a) Branch the node, generating the corresponding children.

(b) Add to C all the child nodes that are not eliminated by the filtering procedure.

Set $B = \varnothing$.

Step 3. Node selection:

Calculate an upper bound on the optimal solution value for all nodes in C.

Select the best β nodes in C and add them to B.

Set $C = \varnothing$.

Step 4. Stopping condition:

If the nodes in B are leaf, select the node with the lowest total cost as the best sequence found and stop.

Otherwise, go to step 2.

Recovering Beam Search:

Step 1. Initialization:

Set $B = \{n_0\}$, $C = \varnothing$, $n_{best} = \varnothing$ and $UB_{best} = \infty$.

Step 2. Filtering step:

For each node in B:

(a) Branch the node, generating the corresponding children.

(b) Add to C all the child nodes that are not eliminated by the filtering procedure.

Set $B = \varnothing$.

Step 3. Accurate evaluation:

For all nodes $n_k, k = 1, \ldots, |C|$ in C:

(a) Calculate a lower bound LB_k and an upper bound UB_k on the optimal solution value of node n_k.

(b) Compute the evaluation function $V = (1 - \gamma)\, LB_k + \gamma UB_k$.

(c) If $UB_k < UB_{best}$, set $n_{best} = n_k$ and $UB_{best} = UB_k$.

Step 4. Recovering step:

Sort all nodes in C in non-decreasing order of the evaluation function value.

Set $k = 1$.

While $|B| < \beta$ and $k \leq |C|$:

(a) Let σ represent the partial solution associated with the current node n_k.

(b) Search for a partial solution $\overline{\sigma}$ that dominates σ by means of neighbourhood operators.

(c) If $\overline{\sigma}$ is found, set $\sigma = \overline{\sigma}$.

(d) If $n_k \notin B$

i. Set $B = B \cup \{n_k\}$.

ii. If $UB_k < UB_{best}$, set $n_{best} = n_k$ and $UB_{best} = UB_k$.

(e) Set $k = k + 1$.

Step 5. Stopping condition:

If the nodes in B are leaf, stop with n_{best} and UB_{best} as the best node and lowest total cost found, respectively.

Otherwise, go to step 2.

2.4 Implementation details

The implementations details of the beam search procedures will now be presented. We considered both priority and detailed classic beam search algorithms, as well as filtered and recovering beam search procedures. In order to apply these algorithms to the single machine linear earliness and quadratic tardiness problem, it is necessary to specify their main components, such as branching scheme, evaluation functions, filtering procedure and recovering step. In the following, we first describe the branching scheme, which is common to all the algorithms. Then, we describe the several dispatching rules that were used as evaluation functions. Finally, some additional implementation details are provided for each type of algorithm.

Branching scheme

The branching procedure is identical for all algorithms. A forward branching scheme is used: the sequence is constructed by adding one job at a time

starting from the first position. Therefore, a branch at level l of the search tree indicates the job scheduled in position l.

Dispatching rules

A dispatching rule is required by the several beam search variants, in order to provide a priority evaluation function and/or to calculate an upper bound. Four dispatching heuristics were considered, with the purpose of analysing the effect of different rules on the performance of the beam search procedures. More specifically, we used the EDD, SPT_s_j, CS_AS and EQTP_EXP dispatching rules presented in Valente (2006). The EDD (SPT_s_j) heuristic performed well for instances where most jobs are early (tardy). The CS_AS procedure combines the EDD and SPT_s_j rules, and the EQTP_EXP dispatching rule was the best-peforming of the heuristics considered in Valente (2006). Therefore, four versions (corresponding to the four dispatching rules) were then considered for each type of beam search procedure. In the following, the CS_AS and EQTP_EXP rules will be denoted simply as CS and EQTP.

Priority beam search

Priority beam search algorithms require a priority evaluation function. For each of the four versions of the PBS procedure, the priority function is provided by the priority index of the appropriate dispatching rule (EDD, SPT_s_j, CS or EQTP). Therefore, the evaluation value of a node is obtained by calculating the appropriate priority index of the last scheduled job.

Detailed beam search

Detailed beam search algorithms require a total cost evaluation function, i.e., an upper bounding procedure. For each DBS version, this upper bounding procedure is provided by the appropriate dispatching heuristic. Therefore, and for a given node, the appropriate rule is used to sequence the remaining unscheduled jobs, thereby completing the existing partial schedule. The evaluation value of the node is then equal to the cost of the complete schedule.

Filtered beam search

Filtered beam search algorithms require a filtering procedure and an upper bounding procedure. The upper bounding procedure is provided by the relevant dispatching rule, just as previously described for the DBS algorithms. The filtering step uses a priority evaluation function filter. Therefore, a priority evaluation function is used to calculate an urgency rating for each offspring of a given node, and the best α children are then chosen for the detailed evaluation step. The priority evaluation function is given by the priority index of the appropriate dispatching heuristic, just as previously described for the PBS algorithms.

Recovering beam search

Recovering beam search algorithms require a filtering procedure, upper and lower bounding procedures for the accurate evaluation step, and an improvement procedure for the recovering step. The filtering and upper bounding procedures are identical to those used in the FBS algorithms. The lower

bounding procedure is provided by the method proposed in Valente (to appear). For a given node, this procedure is used to calculate a lower bound for the remaining unscheduled jobs. The lower bound of the node is then equal to the sum of the cost of the existing partial schedule and the lower bound calculated for the unscheduled jobs.

We considered three simple improvement procedures for the recovering step: adjacent pairwise interchange (API), 3-swaps (3SW) and largest cost insertion (LCI). The API procedure, at each iteration, considers in succession all adjacent job positions. A pair of adjacent jobs is then swapped if such an interchange improves the objective function value. This process is repeated until no improvement is found in a complete iteration. The 3SW procedure is similar, but it considers three consecutive job positions instead of an adjacent pair of jobs. All possible permutations of these three jobs are then analysed, and the best configuration is selected. Once more, the procedure is applied repeatedly until no improvement is possible. The LCI method selects at each iteration the job with the largest objective function value. The selected job is then removed from its position i in the schedule, and inserted at position j, for all $j \neq i$. The best insertion is then performed if it improves the objective function value. This process is repeated until no improving move is found.

3 Computational results

In this section, we first present the set of test problems used in the computational tests. The preliminary computational experiments are then described. These experiments were performed, on the one hand, in order to

select appropriate values for the parameters required by the several beam search heuristics. On the other hand, these preliminary tests were also used to analyse the performance of the beam search procedures under the alternative rules that were considered (EDD, SPT_s_j, CS and EQTP), in order to select the best-performing rule. Finally, we present the computational results. The beam search procedures are first compared with the best existing dispatching heuristic, and the heuristic results are then evaluated against the optimum objective function values for the smaller 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, 100, 250, 500 and 750 jobs. These problems were randomly generated as follows. For each job J_j, an integer processing time p_j was generated from one of the two uniform distributions $[45, 55]$ and $[1, 100]$, in order to obtain low (L) and high (H) variability, respectively, for the processing time values. For each job J_j, an integer due date d_j was 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 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 processing time 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 detailed beam search algorithm was only used on instances with up to 100 jobs, while the filtered and recovering procedures were not applied to the 750 job instances.

3.2 Preliminary tests

In this section, we describe the preliminary computational experiments. These initial experiments were used to determine adequate values for the various parameters required by the beam search algorithms. Moreover, these preliminary tests were also used to analyse the performance of the four heuristic rules that were considered for each beam search procedure, in order to select the best-performing rule. A separate problem set was used to conduct these preliminary experiments. This test set included instances with 25, 50, 75 and 100 jobs, and contained 5 instances for each combination of instance size, processing time variability, T and R. The instances in this smaller test set were generated randomly just as previously described for the full problem set.

Extensive tests were first conducted to determine appropriate values for the beam width, filter width and upper bound weight parameters. We recall there is a trade-off between solution quality and computation time, since increasing the value of the beam or filter width parameters usually improves the objective function value, at the cost of increased computational effort.

The following values were considered for these parameters:

$\alpha = \{1, 2, \ldots, 10\}$,

$\beta = \{1, 2, \ldots, 8\}$,

$\gamma = \{0.1, 0.2, \ldots, 0.9\}$.

The preliminary tests were also used to select an adequate improvement procedure for the recovering step in the RBS algorithm. As mentioned before, the API, 3SW and LCI procedures were considered.

The several versions of the beam search algorithms were applied to the test instances for all combinations of the relevant parameter values. For the RBS procedures, the three improvement procedures were also tested. We then calculated and plotted the mean objective function values and runtimes. A thorough analysis of these data showed the usual behaviour: the computation time increased linearly with the beam and filter width parameters, while the solution quality improved, but with diminishing returns. We then selected the parameter values and the improvement procedure that seemed to provide the best trade-off between solution quality and computation time. For all beam search versions, a value of 3 was chosen for both the beam and the filter width parameters. In the four RBS versions, the upper bound weight was set at 0.8, and the API procedure was chosen for the recovering step.

The performance of the alternative rules that were considered for each beam search procedure (EDD, SPT_s_j, CS and EQTP) was also analysed in the preliminary computational tests, in order to determine the best-performing rule. Table 1 presents, for each beam search algorithm, the average of the relative improvements in objective function value over the EDD rule (%imp),

as well as the percentage number of times a rule achieves the best objective function value found when compared with the other rules (%best). More precisely, the relative improvement over the EDD rule is calculated as (edd_ofv - rule_ofv) / edd_ofv $\times$ 100, where edd_ofv and rule_ofv are the objective function values obtained by the EDD rule and the appropriate rule (SPT_s_j, CS or EQTP), respectively. These values are omitted for the EDD rule, since they would all be necessarily equal to 0.

The SPT_s_j rule provides the best objective function value for a larger percentage of instances than the EDD rule. However, the relative improvement values are quite negative, and therefore the SPT_s_j rule gives, on average, an objective function value that is much larger than the one achieved by the EDD heuristic. The quite negative relative improvement values are essentially due to the inferior performance of the SPT_s_j heuristic for instances with a low tardiness factor (i.e., instances where most jobs will be completed early). Indeed, the SPT_s_j heuristic actually provides better results than the EDD rule for instances with a high tardiness factor, but this is more than offset by a quite poor performance for the low tardiness factor instances. This is to be expected, since as we mentioned earlier, the EDD rule performs better for instances with a larger number of early jobs, while the SPT_s_j heuristic is instead suited to instances where most jobs will be tardy.

For instances with low processing time variability, the objective function values provided by the EDD, CS and EQTP rules are quite close. Nevertheless, the CS and EQTP rules provide the best results for a larger number of instances. This is particularly clear for the EQTP rule, which provides

the best results for over 90% of the test instances. The CS and (especially) the EQTP rules clearly outperform the EDD rule for the high variability instances. In fact, these rules not only provide a quite significant relative improvement, but also give the best results for a much larger number of instances.

For the high variability instances, the improvement provided by the CS and EQTP rules over the EDD heuristic is higher for the PBS procedure, which relies only on a priority evaluation. Therefore, it certainly seems that a high quality priority function should be used in beam search algorithms. Even though the relative improvement is smaller for the FBS procedure (which uses both priority and detailed evaluations), and also for the DBS algorithm (which uses only a detailed evaluation), the more sophisticated CS and EQTP rules nevertheless still provide a significant improvement. Hence, a good rule should also be used to obtain an upper bound estimate in beam search procedures. The objective function values provided by the several rules are closer for the RBS procedure. This is to be expected, since the RBS algorithm uses a recovering step that corrects previous wrong decisions. Therefore, incorrect choices made previously by an inferior rule can later be corrected in the recovering step, and the several rules then provide results that are much closer.

The EQTP rule is then selected, since it provides the best performance. In fact, this rule not only achieves the best results for a quite large percentage of the test instances, but it also provides a large relative improvement over the EDD heuristic for the instances with a high processing time variability. In the following sections, we will therefore present results only for the beam

search versions that use the EQTP rule.

3.3 Heuristic results

In this section, we present the computational results for the heuristic procedures. In addition to the beam search algorithms, we also include, for comparison purposes, the best-performing of the existing procedures, namely the EQTP dispatching rule. Table 2 gives the average of the relative improvements in objective function value over the EQTP procedure (%imp), as well as the percentage number of times a heuristic achieves the best result when compared with the other heuristics (%best). The relative improvement over the EQTP heuristic is calculated as (eqtp_ofv - heur_ofv) / eqtp_ofv $\times$ 100, where heur_ofv and eqtp_ofv are the objective function values of the appropriate heuristic and the EQTP dispatching rule, respectively. The relative improvement values are omitted for the EQTP heuristic, since they are necessarily equal to 0.

The PBS algorithm fails to improve on the EQTP dispatching rule. Indeed, both the objective function values and the percentage of best results are quite close for these two heuristics. The negative average relative improvement values for some instance sizes may seem surprising, since it appears that the PBS algorithm should generate the EQTP sequence, and therefore could not provide results inferior to those of the EQTP dispatching rule. However, the PBS algorithm is only guaranteed to generate the EQTP sequence if there are no ties in the selection of jobs during the various iterations, or if those ties are resolved in the same way. Due to the nature of both the

instance data and the EQTP priority index, ties may indeed occur when a job is to be selected for the next position. Also, for computational efficiency concerns, these ties are not guaranteed to be solved identically in the EQTP and PBS heuristics. Therefore, it is possible for the PBS heuristic not to generate the EQTP sequence, and consequently to provide an inferior result.

The DBS, FBS and RBS procedures provide an improvement over the EQTP dispatching heuristic, particularly for the instances with high processing time variability. The best results are given by the RBS and DBS algorithms. The RBS procedure usually provides a higher relative improvement, while the DBS heuristic generally achieves the best results for a larger number of instances.

The FBS algorithm is also superior to the EQTP dispatching rule, but is outperformed by the DBS and RBS procedures. The DBS algorithm performs a thorough evaluation for all nodes, which can explain its superior performance, since the FBS procedure only calculates an upper bound estimate for the nodes that are not eliminated by the filtering step. The RBS heuristic, on the other hand, not only uses a weighted average of both upper and lower bounds in the detailed evaluation step, but also benefits from the recovering step, which uses local search to correct previous wrong decisions.

The relative improvement given by the RBS, DBS and FBS algorithms is much larger for the high variability instances. Indeed, the improvement provided by the RBS and DBS procedures is about 3-4% for instances with high variability. For the low variability instances, however, the relative improvement is usually below 1%.

In table 3, we present the effect of the T and R parameters on the relative

improvement over the EQTP dispatching rule, for instances with 50 jobs. The improvement given by the beam search algorithms is minor when a larger number of jobs are completed after their dates ($T \geq 0.6$). In fact, when most jobs are tardy ($T = 1.0$), the objective function values are quite close for all the heuristic procedures. The improvement provided by the RBS, DBS and FBS algorithms, however, is quite significant for instances with a low tardiness factor ($T \leq 0.4$), where a larger proportion of jobs are completed before their due dates. Indeed, the relative improvement given by the RBS and DBS algorithms is about 10-20% for some of the parameter combinations with $T = 0.2$ or $T = 0.4$.

The heuristic runtimes (in seconds) are presented in table 4. The DBS procedure is computationally demanding, and therefore can only be used for small to medium size instances. The FBS and RBS algorithms are faster, and can be applied to larger instances. The PBS procedure is the fastest of the beam search procedures. However, the EQTP dispatching rule is more computationally efficient, and provides results of similar quality.

The RBS or DBS procedures are then recommended for small to medium size instances. For somewhat larger instances, the RBS heuristic is the procedure of choice, since it provides much better results than the FBS algorithm, and is only slightly more computationally intensive. For quite large instance sizes, the EQTP dispatching rule is the only procedure that can provide results in a reasonable computation time.

3.4 Comparison with optimum results

In this section, the heuristic results are compared with the optimum objective function values, for instances with up to 20 jobs. In table 5, we present 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 given.

From table 5, it can be seen that the heuristics are quite close to the optimum when the processing time variability is low. The RBS procedure, in particular, performs extremely well. In fact, this heuristic not only provides objective function values that are less than about 1% above the optimum, but also generates an optimum solution for a quite large number of instances. The DBS and FBS procedures also perform quite well. These heuristics also provide an optimum solution for a large number of the test instances, and their average deviation from the optimum is usually less than 1%. Even the simpler PBS and EQTP procedures perform well, providing results that are about 1-2% above the optimum.

The performance of the heuristics, however, deteriorates when the variability of the processing times increases, particularly for the simpler EQTP and PBS procedures. The RBS procedure still performs quite well for instances with high variability, since its average deviation from the optimum is less than 1%, and it provides an optimum solution for over half of the test instances. The performance of the DBS and FBS procedures is also quite good. Indeed, these procedures give results that are about 3% above the

optimum. The EQTP and PBS heuristics, however, perform poorly for the high variability instances, since they are 10-20% above the optimum.

These results are in line with those presented in the previous section for the relative improvement provided by the beam search heuristics. As mentioned in the previous section, the relative improvement over the EQTP dispatching heuristic was lower (higher) for the instances with a low (high) processing time variability. We can now see that there was indeed little room for improvement in the low variability instances. In fact, the EQTP heuristic is already close to the optimum for instances with a low variability. When the variability is high, however, the EQTP heuristic performs poorly, and therefore it is possible to obtain a larger relative improvement.

The effect of the T and R parameters on the relative deviation from the optimum is presented in table 6, for instances with 20 jobs. The heuristics are much closer to the optimum when a larger number of jobs is tardy ($T \geq 0.6$). Actually, when most of the jobs complete after their due dates ($T = 1.0$), the heuristic procedures are usually optimal or nearly optimal. The relative deviation from the optimum is higher for instances with a larger proportion of early jobs (particularly instances with $T = 0.2$ or $T = 0.4$).

4 Conclusion

In this paper, we considered the single machine scheduling problem with linear earliness and quadratic tardiness costs, and no machine idle time. Several heuristics based on the beam search approach were presented. These algorithms included classic beam search procedures, as well as the filtered and

recovering variants. Beam search algorithms require evaluation functions, which are typically provided by dispatching rules. Four dispatching heuristics were considered, so as to analyse the effect of different rules on the performance of the beam search algorithms.

We performed extensive preliminary computational experiments, in order to determine adequate values for the parameters required by the several beam search procedures. The performance of the alternative dispatching rules was also analysed in these initial experiments. The results show that using better rules for priority and/or detailed evaluations improves the performance of the beam search heuristics. Indeed, the more sophisticated CS and EQTP rules provided an improvement over the simpler EDD rule, particularly for the high variability instances.

The best-performing versions of the beam search algorithms were then compared with the best existing heuristic (the EQTP dispatching rule), as well as with optimal solutions. The best results are given by the RBS and DBS algorithms, and the FBS procedure also provides an improvement over the best existing heuristic. The relative improvement given by the RBS, DBS and FBS algorithms is much larger for the high variability instances. The several heuristic procedures, particularly the RBS procedure, were quite close to the optimum for instances with low processing time variability. The RBS and DBS algorithms still performed quite well for the high variability instances, giving results that are about 1% and 3% above the optimum, respectively. The EQTP and PBS heuristics, however, perform poorly for these instances, since they are 10-20% above the optimum.

The RBS or DBS procedures are recommended for small to medium size

instances. For somewhat larger instance sizes, the DBS heuristic requires an excessive computation time, and the RBS procedure is then the heuristic of choice. For extremely large instance sizes, however, dispatching heuristics are the only procedure that can provide results within reasonable computation times.

References

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

Della Croce, F., M. Ghirardi and R. Tadei (2004), Recovering beam search: Enhancing the beam search approach for combinatorial problems, Journal of Heuristics, 10, 89–104.

Della Croce, F. and V. T'kindt (2002), A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem, Journal of the Operational Research Society, 53, 1275–1280.

Esteve, B., C. Aubijoux, A. Chartier and V. T'kindt (2006), A recovering beam search algorithm for the single machine just-in-time scheduling problem, European Journal of Operational Research, 172, 798–813.

Garey, M. R., R. E. Tarjan and G. T. Wilfong (1988), One-processor scheduling with symmetric earliness and tardiness penalties, Mathematics of Operations Research, 13, 330–348.

Ghirardi, M. and C. N. Potts (2005), Makespan minimization for scheduling

unrelated parallel machines: A recovering beam search approach, European Journal of Operational Research, 165, 457–467.

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

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

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

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

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.

Lowerre, B. T. (1976), The HARPY Speech Recognition System, Ph.d. thesis, Carnegie-Mellon University, USA.

Ow, P. S. and T. E. Morton (1988), Filtered beam search in scheduling, International Journal of Production Research, 26, 35–62.

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

Rubin, S. (1978), The ARGOS Image Understanding System, Ph.d. thesis, Carnegie-Mellon University, USA.

Sabuncuoglu, I. and M. Bayiz (1999), Job shop scheduling with beam search, European Journal of Operational Research, 118, 390–412.

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.

Schaller, J. (2007), A comparison of lower bounds for the single-machine early/tardy problem, Computers & Operations Research, 34, 2279–2292.

Sen, T., P. Dileepan and 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.

Su, L.-H. and 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.

Sun, X., J. S. Noble and C. M. Klein (1999), Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness, IIE Transactions, 31, 113–124.

Taguchi, G. (1986), Introduction to Quality Engineering, Asian Productivity Organization, Tokyo, Japan.

Valente, J. M. S. (2006), Heuristics for the single machine scheduling problem with early and quadratic tardy penalties, Working Paper 234, Faculdade de Economia, Universidade do Porto, Portugal (to appear in European Journal of Industrial Engineering).

Valente, J. M. S. (to appear), An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties, Asia-Pacific Journal of Operational Research, to appear.

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

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

Table 1: Preliminary results

			$n = 25$		$n = 50$		$n = 75$		$n = 100$	
var	heur	rule	%imp	%best	%imp	%best	%imp	%best	%imp	%best
L	PBS	EDD	—	5.00	—	5.83	—	5.00	—	1.67
		SPT_s_j	-172.54	30.00	-110.67	30.00	-451.52	31.67	-154.00	29.17
		CS	-0.53	40.83	0.14	41.67	0.26	36.67	0.38	37.50
		EQTP	-0.10	94.17	-0.03	94.17	0.25	95.83	-0.33	95.00
	DBS	EDD	—	23.33	—	7.50	—	6.67	—	5.00
		SPT_s_j	-2.08	35.00	-3.29	33.33	-8.77	35.00	-1.83	30.83
		CS	0.10	57.50	0.30	53.33	0.34	50.00	0.42	49.17
		EQTP	-0.95	90.83	-0.43	92.50	-0.62	93.33	-0.48	93.33
	FBS	EDD	—	20.83	—	5.00	—	5.00	—	4.17
		SPT_s_j	-101.75	30.00	-90.30	30.00	-314.23	31.67	-123.62	29.17
		CS	0.01	45.83	0.17	48.33	0.13	45.00	0.21	42.50
		EQTP	-0.38	94.17	0.16	94.17	-0.07	95.83	-0.11	95.00
	RBS	EDD	—	60.83	—	59.17	—	60.00	—	58.33
		SPT_s_j	-23.42	44.17	-14.05	38.33	-65.02	34.17	-17.85	34.17
		CS	0.07	66.67	0.07	66.67	0.09	59.17	0.08	60.83
		EQTP	-0.21	90.83	-0.07	87.50	-0.65	94.17	-0.75	93.33
H	PBS	EDD	—	3.33	—	1.67	—	3.33	—	2.50
		SPT_s_j	-137.62	17.50	-169.23	13.33	-123.77	12.50	-125.97	18.33
		CS	16.02	33.33	18.47	26.67	19.58	23.33	20.42	26.67
		EQTP	19.07	79.17	23.47	82.50	23.83	87.50	26.12	90.83
	DBS	EDD	—	14.17	—	5.83	—	4.17	—	3.33
		SPT_s_j	-4.21	33.33	-0.28	29.17	0.37	22.50	-0.85	25.00
		CS	5.08	53.33	5.96	36.67	7.28	34.17	7.67	36.67
		EQTP	4.19	60.00	4.01	72.50	5.53	80.00	6.10	79.17
	FBS	EDD	—	6.67	—	4.17	—	0.00	—	0.83
		SPT_s_j	-121.82	21.67	-179.08	21.67	-129.20	20.00	-133.34	21.67
		CS	5.64	30.00	10.15	35.83	12.73	29.17	14.62	30.83
		EQTP	9.99	85.00	14.51	78.33	16.47	87.50	19.86	89.17
	RBS	EDD	—	48.33	—	38.33	—	43.33	—	36.67
		SPT_s_j	-58.00	25.83	-110.13	13.33	-102.29	9.17	-109.15	10.00
		CS	0.70	40.83	1.01	25.00	1.25	16.67	1.41	15.00
		EQTP	2.35	85.83	2.57	67.50	2.56	63.33	2.89	65.00

Table 2: Heuristic results

		$n = 25$		$n = 50$		$n = 100$		$n = 500$	
var	heur	%imp	%best	%imp	%best	%imp	%best	%imp	%best
L	EQTP	—	24.42	—	14.83	—	16.33	—	45.25
	PBS	-0.39	24.58	-0.30	14.50	-0.26	16.08	-0.21	43.75
	DBS	0.56	81.50	0.73	88.25	0.57	91.50	—	—
	FBS	0.35	55.33	0.38	48.67	0.08	42.08	-0.05	47.42
	RBS	1.12	73.33	0.99	56.42	0.58	48.50	0.25	99.67
H	EQTP	—	7.17	—	0.75	—	0.33	—	13.67
	PBS	0.04	7.25	-0.07	0.83	-0.05	0.33	-0.04	13.67
	DBS	4.33	44.33	3.73	54.17	3.49	65.58	—	—
	FBS	3.95	35.58	2.24	32.17	1.22	32.83	0.33	42.50
	RBS	5.51	77.42	4.00	42.92	3.00	24.17	2.06	72.33

Table 3: Relative improvement over the EQTP heuristic, for instances with 50 jobs

		low var				high var			
heur	T	$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$	$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$
PBS	0.0	0.00	0.00	0.00	-0.01	-0.01	-0.02	-0.01	-0.01
	0.2	-2.98	-3.91	-0.05	-0.07	-1.99	0.09	0.00	0.01
	0.4	0.00	0.00	0.00	-0.21	0.18	0.00	0.02	0.12
	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
DBS	0.0	0.00	0.02	0.05	0.09	0.25	0.56	0.86	1.58
	0.2	4.83	5.26	1.29	1.51	13.54	17.03	12.53	13.77
	0.4	0.03	0.06	0.18	4.12	1.29	1.06	2.70	22.73
	0.6	0.00	0.00	0.00	0.00	0.65	0.20	0.14	0.21
	0.8	0.00	0.00	0.00	0.00	0.34	0.05	0.03	0.03
	1.0	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01
FBS	0.0	0.00	0.01	0.04	0.07	0.29	0.55	0.85	1.28
	0.2	2.47	3.33	0.67	0.66	6.32	10.87	6.41	6.57
	0.4	0.01	0.02	0.07	1.77	1.04	0.80	1.85	15.33
	0.6	0.00	0.00	0.00	0.00	0.56	0.24	0.17	0.27
	0.8	0.00	0.00	0.00	0.00	0.20	0.07	0.03	0.03
	1.0	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01
RBS	0.0	0.00	0.02	0.05	0.09	0.33	0.67	0.98	1.59
	0.2	10.60	8.16	1.01	1.11	24.65	18.10	10.20	11.11
	0.4	0.02	0.03	0.09	2.66	1.36	1.02	2.56	21.42
	0.6	0.00	0.00	0.00	0.00	0.77	0.18	0.11	0.27
	0.8	0.00	0.00	0.00	0.00	0.52	0.07	0.02	0.01
	1.0	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00

Table 4: Heuristic runtimes (in seconds)

var	heur	$n = 25$	$n = 50$	$n = 100$	$n = 250$	$n = 500$	$n = 750$
L	EQTP	0.000	0.000	0.001	0.004	0.015	0.034
	PBS	0.002	0.007	0.036	0.509	5.408	32.398
	DBS	0.021	0.295	4.553	—	—	—
	FBS	0.004	0.029	0.206	3.369	28.250	—
	RBS	0.006	0.033	0.227	3.473	28.567	—
H	EQTP	0.000	0.000	0.001	0.004	0.016	0.036
	PBS	0.002	0.008	0.047	0.536	5.430	32.747
	DBS	0.022	0.316	4.938	—	—	—
	FBS	0.005	0.031	0.232	3.613	29.943	—
	RBS	0.006	0.035	0.250	3.711	30.282	—

Table 5: Comparison with optimum objective function values

		$n = 10$		$n = 15$		$n = 20$	
var	heur	%dev	%opt	%dev	%opt	%dev	%opt
L	EQTP	1.78	45.58	2.14	34.50	1.83	28.17
	PBS	1.44	50.33	2.51	35.83	2.33	29.25
	DBS	0.10	89.50	0.45	76.08	0.69	68.08
	FBS	0.22	83.67	0.63	64.67	1.10	60.00
	RBS	0.02	97.00	0.03	83.17	0.13	73.25
H	EQTP	22.14	22.25	16.45	11.92	11.96	8.67
	PBS	17.99	24.08	15.39	12.67	12.20	9.08
	DBS	3.13	52.75	3.54	38.58	3.22	33.17
	FBS	2.73	51.92	2.91	38.08	3.79	32.58
	RBS	0.46	88.83	0.89	75.83	0.81	56.83

Table 6: Relative deviation from the optimum for instances with 20 jobs

		low var				high var			
heur	T	$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$	$R = 0.2$	$R = 0.4$	$R = 0.6$	$R = 0.8$
EQTP	0.0	0.19	0.09	0.08	0.11	0.66	1.64	2.65	2.64
	0.2	17.05	13.56	3.98	2.23	60.07	91.36	26.80	20.54
	0.4	0.10	0.13	0.42	5.92	6.34	4.57	13.49	44.53
	0.6	0.02	0.02	0.01	0.01	3.97	1.38	1.23	1.88
	0.8	0.01	0.01	0.00	0.00	1.68	0.82	0.30	0.25
	1.0	0.00	0.00	0.00	0.00	0.03	0.05	0.05	0.06
PBS	0.0	0.28	0.10	0.11	0.12	0.67	1.64	2.65	2.64
	0.2	20.10	21.02	4.74	2.21	60.38	87.69	39.06	19.12
	0.4	0.10	0.12	0.41	6.62	6.33	4.45	13.39	43.84
	0.6	0.02	0.02	0.01	0.01	3.97	1.38	1.19	1.79
	0.8	0.01	0.01	0.00	0.00	1.12	0.80	0.23	0.25
	1.0	0.00	0.00	0.00	0.00	0.03	0.05	0.05	0.06
DBS	0.0	0.05	0.01	0.01	0.02	0.38	0.87	1.28	1.11
	0.2	8.18	6.48	1.55	0.09	24.13	14.95	5.70	4.13
	0.4	0.03	0.01	0.03	0.18	2.32	0.95	3.54	14.43
	0.6	0.01	0.00	0.00	0.00	1.72	0.26	0.30	0.49
	0.8	0.00	0.00	0.00	0.00	0.28	0.27	0.06	0.08
	1.0	0.00	0.00	0.00	0.00	0.01	0.02	0.01	0.01
FBS	0.0	0.01	0.00	0.01	0.02	0.15	0.26	0.51	0.52
	0.2	11.19	7.91	2.07	0.66	30.45	28.12	5.79	6.56
	0.4	0.02	0.04	0.20	4.23	3.14	1.18	2.39	7.78
	0.6	0.01	0.00	0.00	0.00	2.43	0.27	0.24	0.52
	0.8	0.00	0.00	0.00	0.00	0.43	0.18	0.04	0.07
	1.0	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01
RBS	0.0	0.00	0.00	0.00	0.01	0.04	0.11	0.28	0.15
	0.2	1.86	0.21	0.23	0.27	3.97	4.52	2.31	1.42
	0.4	0.01	0.03	0.08	0.50	0.68	0.34	0.72	3.79
	0.6	0.00	0.00	0.00	0.00	0.67	0.13	0.06	0.04
	0.8	0.00	0.00	0.00	0.00	0.04	0.06	0.00	0.01
	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Recent FEP Working Papers

Nº 249	T. Andrade, G. Faria, V. Leite, F. Verona, M. Viegas, O.Afonso and P.B. Vasconcelos, *"Numerical solution of linear models in economics: The SP-DG model revisited"*, October 2007
Nº 248	Mário Alexandre P. M. Silva, *"Aghion And Howitt's Basic Schumpeterian Model Of Growth Through Creative Destruction: A Geometric Interpretation"*, October 2007
Nº 247	Octávio Figueiredo, Paulo Guimarães and Douglas Woodward, *"Localization Economies and Establishment Scale: A Dartboard Approach"*, September 2007
Nº 246	Dalila B. M. M. Fontes, Luís Camões and Fernando A. C. C. Fontes, *"Real Options using Markov Chains: an application to Production Capacity Decisions"*, July 2007
Nº 245	Fernando A. C. C. Fontes and Dalila B. M. M. Fontes, *"Optimal investment timing using Markov jump price processes"*, July 2007
Nº 244	Rui Henrique Alves and Óscar Afonso, *"Fiscal Federalism in the European Union: How Far Are We?"*, July 2007
Nº 243	Dalila B. M. M. Fontes, *"Computational results for Constrained Minimum Spanning Trees in Flow Networks"*, June 2007
Nº 242	Álvaro Aguiar and Inês Drumond, *"Business Cycle and Bank Capital: Monetary Policy Transmission under the Basel Accords"*, June 2007
Nº 241	Sandra T. Silva, Jorge M. S. Valente and Aurora A. C. Teixeira, *"An evolutionary model of industry dynamics and firms' institutional behavior with job search, bargaining and matching"*, April 2007
Nº 240	António Miguel Martins and Ana Paula Serra, *"Market Impact of International Sporting and Cultural Events"*, April 2007
Nº 239	Patrícia Teixeira Lopes and Lúcia Lima Rodrigues, *"Accounting for financial instruments: A comparison of European companies' practices with IAS 32 and IAS 39"*, March 2007
Nº 238	Jorge M. S. Valente, *"An exact approach for single machine scheduling with quadratic earliness and tardiness penalties"*, February 2007
Nº 237	Álvaro Aguiar and Ana Paula Ribeiro, *"Monetary Policy and the Political Support for a Labor Market Reform"*, February 2007
Nº 236	Jorge M. S. Valente and Rui A. F. S. Alves, *"Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties"*, February 2007
Nº 235	Manuela Magalhães and Ana Paula Africano, *"A Panel Analysis of the FDI Impact on International Trade"*, January 2007
Nº 234	Jorge M. S. Valente, *"Heuristics for the single machine scheduling problem with early and quadratic tardy penalties"*, December 2006
Nº 233	Pedro Cosme Vieira and Aurora A. C. Teixeira, *"Are Finance, Management, and Marketing Autonomous Fields of Scientific Research? An Analysis Based on Journal Citations"*, December 2006
Nº 232	Ester Gomes da Silva and Aurora A. C. Teixeira, *"Surveying structural change: seminal contributions and a bibliometric account"*, November 2006
Nº 231	Carlos Alves and Cristina Barbot, *"Do low cost carriers have different corporate governance models?"*, November 2006
Nº 230	Ana Paula Delgado and Isabel Maria Godinho, *"Long term evolution of the size distribution of Portuguese cities"*, September 2006
Nº 229	Sandra Tavares Silva and Aurora A. C. Teixeira, *"On the divergence of evolutionary research paths in the past fifty years: a comprehensive bibliometric account"*, September 2006
Nº 228	Argentino Pessoa, *"Public-Private Sector Partnerships in Developing Countries: Prospects and Drawbacks"*, September 2006
Nº 227	Sandra Tavares Silva and Aurora A. C. Teixeira, *"An evolutionary model of*

	firms' institutional behavior focusing on labor decisions", August 2006
Nº 226	Aurora A. C. Teixeira and Natércia Fortuna, *"Human capital, trade and long-run productivity. Testing the technological absorption hypothesis for the Portuguese economy, 1960-2001",* August 2006
Nº 225	Catarina Monteiro and Aurora A. C. Teixeira, *"Local sustainable mobility management. Are Portuguese municipalities aware?",* August 2006
Nº 224	Filipe J. Sousa and Luís M. de Castro, *"Of the significance of business relationships",* July 2006
Nº 223	Pedro Cosme da Costa Vieira, *"Nuclear high-radioactive residues: a new economic solution based on the emergence of a global competitive market",* July 2006
Nº 222	Paulo Santos, Aurora A. C. Teixeira and Ana Oliveira-Brochado, *"The 'de-territorialisation of closeness' - a typology of international successful R&D projects involving cultural and geographic proximity",* July 2006
Nº 221	Manuel M. F. Martins, *"Dilemas macroeconómicos e política monetária: o caso da Zona Euro",* July 2006
Nº 220	Ana Oliveira-Brochado and F. Vitorino Martins, *"Examining the segment retention problem for the "Group Satellite" case",* July 2006
Nº 219	Óscar Afonso Rui and Henrique Alves, *"To Deficit or Not to Deficit": Should European Fiscal Rules Differ Among Countries?",* July 2006
Nº 218	Rui Henrique Alves and Óscar Afonso, *"The "New" Stability and Growth Pact: More Flexible, Less Stupid?",* July 2006
Nº 217	J Maciej Cieślukowski and Rui Henrique Alves, *"Financial Autonomy of the European Union after Enlargement",* July 2006
Nº 216	Joao Correia-da-Silva and Carlos Hervés-Beloso, *"Prudent Expectations Equilibrium in Economies with Uncertain Delivery",* June 2006
Nº 215	Maria Rosário Moreira and Rui Alves, *"How far from Just-in-time are Portuguese firms? A survey of its progress and perception",* June 2006
Nº 214	Maria Fátima Rocha and Aurora A.C. Teixeira, *"A cross-country evaluation of cheating in academia: is it related to 'real world' business ethics?",* June 2006
Nº 213	Maria Rosário Moreira and Rui Alves, *"Does Order Negotiation Improve The Job-Shop Workload Control?",* June 2006
Nº 212	Pedro Cosme da Costa Vieira and Aurora A. C. Teixeira, *"Human capital and corruption: a microeconomic model of the bribes market with democratic contestability",* May 2006
Nº 211	Ana Teresa Tavares and Aurora A. C. Teixeira, *"Is human capital a significant determinant of Portugal's FDI attractiveness?"*, May 2006
Nº 210	Maria Rosário Moreira and Rui Alves, *A new input-output control order release mechanism: how workload control improves manufacturing operations in a job shop*, April 2006
Nº 209	Patrícia Teixeira Lopes and Lúcia Lima Rodrigues, *Accounting for Financial Instruments: An Analysis of the Determinants of Disclosure in the Portuguese Stock Exchange*, April 2006

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