

# **Filtered and Recovering beam search algorithms for the early/tardy scheduling problem with no idle time**

**Jorge M. S. Valente**

**and**

**Rui A. F. S. Alves**



FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

[www.fep.up.pt](http://www.fep.up.pt)

# Filtered and Recovering beam search algorithms for the early/tardy scheduling problem with no idle time

Jorge M. S. Valente and Rui A. F. S. Alves

Faculdade de Economia, Universidade do Porto  
Rua Dr. Roberto Frias, 4200-464 Porto, Portugal  
e-mails: jvalente@fep.up.pt; ralves@fep.up.pt

April 13, 2004

## Abstract

In this paper we consider the single machine earliness/tardiness scheduling problem with no idle time. We present heuristic algorithms based on the filtered and recovering beam search techniques and compare them with existing neighbourhood search and dispatch rule heuristics. Filtering procedures using both priority evaluation functions and problem-specific properties have been considered. Extensive preliminary tests were performed to determine appropriate parameter values for the beam search algorithms and the neighbourhood search procedure.

The computational results show that the recovering beam search algorithms outperform their filtered counterparts in both solution quality and computational requirements, while the priority-based filtering procedure proves superior to the rules-based alternative. The best solutions are given by the neighbourhood search algorithm, but this procedure is computationally intensive and can therefore only be applied to small or medium size instances. The recovering beam search heuristic provides results that are close in solution quality and is significantly faster, so it can be used to solve even large instances.

**Keywords:** scheduling, early/tardy, beam search, heuristics

## Resumo

Neste artigo consideramos um problema de sequenciamento com um único processador, custos de posse e atraso e inexistência de tempo morto. Heurísticas baseadas nas técnicas de *filtered* e *recovering beam search* são apresentadas e comparadas com algoritmos de pesquisa local e *dispatch rules* existentes. Dois diferentes tipos de procedimentos de filtragem - funções prioridade e regras relativas ao problema em causa - são estudados. Diversos testes computacionais foram efectuados para determinar valores apropriados para os parâmetros usados pelos diversos algoritmos.

Os testes computacionais mostram que os procedimentos de *recovering beam search* superam os algoritmos baseados em *filtered beam search* não só na qualidade da solução obtida, como também no tempo de computação. O método de filtragem que utiliza funções prioridade revelou-se substancialmente melhor do que o baseado em regras. O algoritmo de pesquisa local proporcionou os melhores resultados, mas apenas pode ser aplicado a instâncias de dimensão pequena ou média, em virtude dos seus elevados requisitos computacionais. A heurística de *recovering beam search* gera soluções com uma qualidade bastante próxima e o seu tempo de computação é substancialmente inferior, pelo que pode ser utilizada para resolver instâncias de dimensão elevada.

**Palavras-chave:** sequenciamento, custos de posse e atraso, beam search, heurísticas

## 1 Introduction

We consider a single machine scheduling problem with earliness and tardiness costs that can be stated as follows. A set of  $n$  independent jobs  $\{J_1, J_2, \dots, J_n\}$  has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine and the jobs are assumed to be continuously available from time zero onwards and machine idle time is not allowed. Job  $J_j, j = 1, 2, \dots, n$ , requires a processing time  $p_j$  and should ideally be completed on its due date  $d_j$ . For any given schedule, the earliness and tardiness of  $J_j$  can be respectively defined as  $E_j = \max\{0, d_j - C_j\}$  and  $T_j = \max\{0, C_j - d_j\}$ , where  $C_j$  is the completion time of  $J_j$ . The objective is then to find a schedule that minimizes the sum of the earliness and tardiness costs of all jobs  $\sum_{j=1}^n (h_j E_j + w_j T_j)$ , where  $h_j$  and  $w_j$  are the earliness and tardiness penalties of job  $J_j$ .

The inclusion of both earliness and tardiness costs in the objective function is compatible with the philosophy of just-in-time production, which emphasizes producing goods only when they are needed. The early cost may represent the cost of completing a project early in PERT-CPM analyses, deterioration in the production

of perishable goods or a holding cost for finished goods. The tardy cost can represent rush shipping costs, lost sales and loss of goodwill. The assumption that no machine idle time is allowed reflects a production setting where the cost of machine idleness is higher than the early cost incurred by completing any job before its due date, or the capacity of the machine is limited when compared with its demand, so that the machine must be kept running.

As a generalization of weighted tardiness scheduling [5], the problem is strongly NP-hard. A large number of papers consider scheduling problems with both earliness and tardiness costs. We will only review those papers that examine a problem that is exactly the same as ours. For more information on earliness and tardiness scheduling, interested readers are referred to Baker and Scudder [2], who provide an excellent review.

Ow and Morton [9] developed several early/tardy dispatch rules and a filtered beam search procedure. The early/tardy dispatch rules clearly outperformed known heuristics that ignored the earliness costs, but were still far from optimal, while the filtered beam search procedure provided very good solutions for small or medium size problems, but required excessive computation times for larger problems. Valente and Alves [14] considered the best dispatch heuristic of Ow and Morton and proposed an additional dispatch rule and a greedy procedure. They presented functions for determining the value of a lookahead parameter used by both dispatch rules, and also considered the use of dominance rules to further improve the schedule obtained by the heuristics. A neighbourhood search algorithm was presented by Li [6]. Exact approaches have also been proposed, and branch-and-bound algorithms were presented by Abdul-Razaq and Potts [1], Li [6] and Liaw [7]. The lower bounding procedure of Abdul-Razaq and Potts was based on the subgradient optimization approach and the dynamic programming state-space relaxation technique, while Li and Liaw used Lagrangean relaxation and the multiplier adjustment method. Valente and Alves [15] show that using better initial sequences can improve the lower bounds developed by Li and Liaw.

In this paper we consider heuristic algorithms based on both the filtered and the recovering beam search approaches. These algorithms are compared with the best existing dispatch rule and the neighbourhood search procedure proposed by Li [6]. We have considered filtering procedures using both priority evaluation functions and problem-specific properties. Extensive computational tests were performed to determine the parameter values that provided the best balance between solution

quality and computational effort for the beam search algorithms and the neighbourhood search heuristic. We also consider using some dominance rules to improve the solutions obtained by the heuristics.

The remainder of the paper is organized as follows. In section 2 we describe the beam search approach and its several variations. The proposed algorithms and the choices made for their main components are presented in section 3. The computational results are reported in section 4 and some concluding remarks are given in section 5.

## 2 The beam search approach

Beam search is a heuristic method for solving combinatorial optimization problems. It consists in an adaptation of branch-and-bound in which only some nodes are evaluated. In the beam search procedure, only the most promising nodes at each level of the search tree are selected for further branching, while the remaining nodes are pruned off permanently. Since a large part of the search tree is pruned aggressively, and only some nodes are retained at each level, the running time is polynomial in the problem size.

Beam search was first used in the artificial intelligence community for the speech recognition [8] and the image understanding [12] problems. A number of applications to scheduling problems have appeared in the literature since then. Fox [4] and Ow and Smith [11] have incorporated a beam search procedure in systems designed for complex job shop environments. Sabuncuoglu and Bayiz [13] presented beam search algorithms for the job shop problem with both makespan and mean tardiness as performance measures. Ow and Morton ([10], [9]) proposed a variation of this technique called filtered beam search and applied it to the single machine early/tardy problem. Recently, Della Croce and T'kindt [3] presented another variation of the beam search approach. This new algorithm, called recovery beam search, was tested on the single machine completion time problem with release dates.

The classic beam search approach consists in a truncated branch and bound where only the most promising  $\beta$  nodes (instead of all nodes) at each level of the search tree are retained for further branching;  $\beta$  is the so-called *beam width*. The other nodes are simply discarded and there is no backtracking, since the intent of this technique is to search quickly. Therefore, beam search methods are not guaranteed to find an optimal solution and cannot recover from wrong decisions: if

a node leading to the optimal solution is discarded during the search process, there is no way to reach that optimal solution afterwards. The beam search approach recognizes this danger by selecting a number (the beam width) of promising paths to search concurrently. A wider beam width allows greater safety, but at the cost of increased computational effort.

The node evaluation process at each level is a key issue in the beam search technique. Two different types of evaluation functions have been used: *priority evaluation functions* and *total cost evaluation functions*. A priority evaluation function simply calculates a priority or urgency rating, typically by computing the priority of the last job added to the sequence using a dispatch rule. A total cost evaluation function calculates an estimate of the minimum total cost of the best solution that can be obtained from the partial schedule represented by the node. This is usually done by using a dispatch rule to complete the existing partial schedule. The priority evaluation function has a local view of the problem, since it considers only the next decision to be made (the next job to schedule), whereas the total cost evaluation function has a more global view, since it projects from the current partial solution to a complete schedule in order to estimate the total cost.

Priority evaluation functions are computationally cheap, but are potentially inaccurate and may result in discarding good nodes. Total cost evaluation functions, on the other hand, are more accurate but require a much higher computational effort. The filtered beam search method uses both crude and accurate evaluations in a two-stage approach, thus trying to provide a computationally efficient evaluation that does not degrade the quality of the search. A computationally inexpensive filtering procedure is first applied in order to select some of the children of each beam node for a more accurate evaluation. The selected nodes are then accurately evaluated using a total cost evaluation function and the best  $\beta$  nodes are retained for further branching. Typically, the filtering procedure uses a priority evaluation function to calculate an urgency value for each offspring and then selects the best  $\alpha$  children of each beam node for the detailed evaluation step;  $\alpha$  is the so-called *filter width*. Recently, a different filtering procedure was used by Della Croce and T'kindt [3]. This new procedure uses problem-specific properties to determine the nodes that advance to the detailed evaluation step. We now present the main steps of the filtered beam search algorithm. Let  $B$  be the set of nodes retained in the beam for further branching and  $C$  be a set of offspring nodes. Also let  $n_0$  be the parent or root node, i.e., the node that contains only unscheduled jobs.

*Filtered Beam Search:*

1. Initialization:

Set  $C = \emptyset$ .

Set  $B = \{n_0\}$ .

2. For each node in  $B$ :

(a) Branch the node generating the corresponding children.

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

3. Set  $B = \emptyset$ .

For all nodes in  $C$ :

(a) Perform a detailed evaluation for that node (usually by calculating an upper bound on the optimal solution value of that node)

(b) Select the  $\min\{\beta, |C|\}$  best nodes in  $C$  (usually the nodes with the lowest upper bound) and add them to  $B$ .

(c) Set  $C = \emptyset$ .

4. Stopping condition:

If the nodes in  $B$  are leaf (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.

The recovering beam search algorithm, like the filtered beam search method, also uses both crude and accurate evaluations in a two-stage approach. However, it differs from filtered beam search in three major ways. First, only one node is retained at each level of the search tree in order to minimise the computation time required by the procedure; this means the beam width has a pre-defined value of one ( $\beta = 1$ ). Second, the accurate evaluation is performed by calculating a weighted sum of both lower and upper bounds on the total cost of the best solution that can be obtained from the partial schedule represented by the node. Finally, once the best node and the corresponding best partial solution are retained at each level

of the search tree, a *recovering step* is then applied. This recovering step checks whether the current partial solution  $\sigma$  is dominated by another partial solution  $\bar{\sigma}$  having the same level of the search tree (typically by applying interchange operators to the current partial solution); if so,  $\bar{\sigma}$  becomes the new current partial solution. Since the recovering step can only replace a partial solution with another partial solution with the same depth of the search tree, the total number of explored nodes is polynomial. Recovering beam search and classic or filtered beam search methods deal in different ways with the danger of discarding a node leading to the optimal solution during the search process. While classic or filtered beam search allow a number of paths to be searched concurrently, recovering beam search retains only one node at each level and relies on the recovering step to recover from previous wrong decisions. We now present the main steps of the recovering beam search algorithm; let  $\sigma$  denote the node that is retained in the beam and  $0 \leq \gamma \leq 1$  be the upper bound weight in the weighted sum of lower and upper bounds.

*Recovering Beam Search:*

1. Initialization:

Set  $C = \emptyset$ .

Set  $\sigma = n_0$ .

2. Branch  $\sigma$  generating the corresponding children. Add to  $C$  the child nodes that are not eliminated by the filtering procedure.

3. For all nodes in  $C$ :

(a) Calculate a lower bound  $LB$  and an upper bound  $UB$  on the optimal solution value of that node.

(b) Compute the evaluation function  $V = (1 - \gamma) LB + \gamma UB$ .

4. Let  $\sigma^*$  be the node in  $C$  with the lowest value of  $V$ .

Set  $\sigma = \sigma^*$ .

Set  $C = \emptyset$ .

5. Recovering step: search for a partial solution  $\bar{\sigma}$  that dominates  $\sigma$  by means of interchange operators. If  $\bar{\sigma}$  is found, set  $\sigma = \bar{\sigma}$ .

6. Stopping condition:

If  $\sigma$  is a leaf node (it holds a complete sequence), stop;  $\sigma$ 's total cost is the best objective function value found.

Otherwise, go to step 2.

### 3 The proposed heuristic procedures

In this section we describe the filtered and recovering beam search algorithms that were considered. In order to apply these algorithms to the early/tardy problem, it is necessary to specify their main components, namely the branching scheme, upper and lower bounding procedures, filtering procedure and recovering step. The branching scheme and the filtering procedure are common to both algorithms. The branching scheme is the usual  $n$ -ary forward branching: the sequence is constructed by adding one job at a time starting from position 1; the search tree is such that a branch at level  $l$  indicates the job scheduled in position  $l$ . We considered the two types of filtering procedures that have been previously proposed. The first requires a priority evaluation function and selects the  $\alpha$  best children of each beam node for the detailed evaluation step. This priority evaluation function is provided by the EXPET dispatch rule presented by Ow and Morton [9]. The value of the lookahead parameter used by this heuristic was calculated using the functions developed by Valente and Alves [14], since these have been shown to outperform a fixed value approach. The priority index of the EXPET heuristic is used to calculate the priority of the last scheduled job in each node. The second filtering procedure uses problem-specific properties to determine the nodes that advance to the detailed evaluation step. Let  $x$  be a partial sequence and let  $i, j \notin x$  be a pair of jobs that can be scheduled in the next position in the sequence. We now present three criteria that were used to determine the nodes that are eliminated and do not advance to the detailed evaluation step.

**Criterion 1** *If  $i$  and  $j$  are both early, regardless of their order, in the next two positions in the sequence, and  $h_i/p_i \leq h_j/p_j$ , then job  $j$  is eliminated.*

**Criterion 2** *If  $i$  and  $j$  are both tardy, regardless of their order, in the next two positions in the sequence, and  $w_i/p_i \geq w_j/p_j$ , then job  $j$  is eliminated.*

**Criterion 3** *If  $j$  is always early and  $i$  is always tardy when scheduled in the next two positions in the sequence, then job  $j$  is eliminated.*

Criteria 1 and 2 are based on local optimality conditions for weighted earliness and weighted tardiness scheduling, respectively. Criterion 3 simply eliminates a job that is early in the next two positions whenever a tardy job is present. The filtered beam search procedure also requires a total cost evaluation function. The EXPET heuristic is used to complete the existing partial schedule and therefore calculate a total cost estimate. The detailed evaluation step in the recovering beam search algorithm requires both upper and lower bound procedures. The upper bound is once again calculated using the EXPET dispatch rule. The lower bound is computed using the procedure presented by Liaw [7]. The initial sequence required by this lower bound is generated using the WSPT or WLPT rules, as appropriate, followed by the application of some dominance rules, as described in Valente and Alves [15]. Finally, the recovering step uses an insertion procedure to detect whether the current partial solution is dominated by another partial solution having the same level of the search tree. The last job in the current partial schedule is inserted before the previously scheduled jobs until a maximum of  $\delta(n-1)$ ,  $0 \leq \delta \leq 1$  insertions have been performed. The parameter  $\delta$  controls the extent of the local search performed during the recovering step, since it determines the maximum number of insertions (alternative schedules) that are considered.

From now on the beam search algorithms with priority evaluation function and problem-specific rules filtering procedures will be respectively identified as FBS\_P and FBS\_R. Similarly, RBS\_P and RBS\_R will denote the recovery beam search algorithms with priority-based and rules-based filtering procedures. The FBS\_P heuristic is quite similar to the filtered beam search algorithm previously presented by Ow and Morton [9]. The only difference is in the choice of the lookahead parameter value required by the EXPET dispatch rule, since a fixed value was used in Ow and Morton's algorithm. The proposed algorithms were compared with two other heuristics, namely the EXPET dispatch rule and the neighbourhood search algorithm presented by Li [6], which will be denoted as NSearch. The NSearch heuristic generates an initial sequence using a procedure that is identical to a beam search algorithm that uses a total cost evaluation function and has a beam width of one; the EXPET dispatch rule was used to provide the total cost estimate. A local search swap procedure is then applied to improve this initial sequence. This procedure uses a set of small swap neighbourhoods and it requires the maximum swap distance  $\lambda(n-1)$ ,  $0 \leq \lambda \leq 1$ , to be specified (see Li's paper for details). The parameter  $\lambda$  controls the dimension of the set of neighbourhoods and the extent of

the local search procedure.

Valente and Alves [14] showed that the dominance rules presented by Ow and Morton [9] and Liaw [7] could be used to improve the solution quality of dispatch procedures with little additional computational effort. Ow and Morton's rule imposes a condition on adjacent pairs of jobs, while the dominance rule presented by Liaw applies to non-adjacent jobs with identical processing times. We also consider using these dominance rules as an improvement step. Once an initial solution has been obtained by the heuristics, these rules are applied as follows. First, the adjacent dominance rule of Ow and Morton is used. When a pair of adjacent jobs violates that rule, those jobs are swapped. This procedure is repeated until no improvement is found by the adjacent rule in a complete iteration. Then the non-adjacent rule is applied. Once again, if a pair of jobs violates the rule those jobs are swapped, and the procedure is repeated until no improvement is made in a complete iteration. The above two steps are repeated while the number of iterations performed by the non-adjacent rule is greater than one (i.e., while that rule detects an improvement).

## 4 Computational results

In this section we present the results from the computational tests. A set of problems with 15, 20, 25, 30, 50, 100, 200, 250, 300, 400 and 500 jobs was randomly generated as follows. For each job  $J_j$  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  $[1, 10]$  and  $[1, 100]$ , to create low and high variability, respectively. For each job  $J_j$ , an integer due date  $d_j$  is generated from the uniform distribution  $[P(1 - L - R/2), P(1 - L + R/2)]$ , where  $P$  is the sum of the processing times of all jobs,  $L$  is the lateness 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, processing time and penalty variability,  $L$  and  $R$ , 50 instances were generated. All the algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV - 1500 Mhz personal computer. The NSearch heuristic was only used to solve instances with up to 250 jobs, since it would require excessive computational times for larger problems. Throughout this section, and in order to avoid excessively large tables, we will sometimes present results only for some representative cases.

Extensive computational tests were first performed to determine appropriate values for the parameters used by the several algorithms. A trade-off exists be-

tween solution quality and computational time, since increasing the value of the parameters usually improves the schedule objective function value, but at the cost of increased runtimes (the only exception being the  $\gamma$  parameter). Therefore, we tried to determine the values that provided the best balance between solution quality and computational effort. The following values were considered for the several parameters:

$$\begin{aligned}\alpha &= \{1, 2, \dots, 10\}, \\ \beta &= \{1, 2, \dots, 8\}, \\ \gamma &= \{0.1, 0.2, \dots, 0.9\}, \\ \delta &= \{0.05, 0.10, \dots, 0.50\}, \\ \lambda &= \{0.05, 0.10, \dots, 0.50\}.\end{aligned}$$

The algorithms were then applied to selected problem sizes for all combinations of the relevant parameter values. The objective function values and runtimes were then thoroughly analysed and the parameter values that seemed to provide the best trade-off between solution quality and computational time were chosen. These values are presented in table 1 and they provided an adequate compromise between schedule quality and computational effort for all the problem types considered in these preliminary tests. The NSearch algorithm was also included in these tests since no previous recommendation was available for the value of  $\lambda$ ; therefore, our study provides a guideline for future implementations of this procedure.

Heur	$\alpha$	$\beta$	$\gamma$	$\delta$	$\lambda$
FBS_P	3	3	—	—	—
FBS_R	—	3	—	—	—
NSearch	—	—	—	—	0.15
RBS_P	3	—	0.7	0.10	—
RBS_R	—	—	0.8	0.10	—

Table 1: Heuristic parameter values

In table 2 we present the average objective function value (mean ofv) for each heuristic, both before (bfr) and after (aft) the application of the dominance rules, and the average of the relative improvements in the objective function values (%ch), calculated as  $(H - H_{DR})/H * 100$ , where  $H$  and  $H_{DR}$  are the objective function values of a heuristic before and after the dominance rules, respectively. We also give the number of times each heuristic produces the best result when compared with the other heuristics (#best), both before and after the use of the dominance rules. A test was also performed to determine if the differences between the heuristic objective

function values before and after the dominance rules are statistically significant. Given that the heuristics were used on exactly the same problems, a paired-samples test is appropriate. Since the hypothesis of the paired-samples t-test were not all met, the non-parametric Wilcoxon test was selected. The significance values of this test, i.e., the level of significance values above which the equal distribution hypothesis is to be rejected, were nearly always equal to 0.000, and only in three cases concerning the application of the NSearch heuristic to small instance sizes were these values larger than 0.05.

From the objective function values, and the number of times each heuristic is the best, we can conclude the following. The best results are given by the NSearch heuristic, followed by the RBS\_P and the FBS\_P algorithms. The RBS\_P algorithm, in particular, is close to the best heuristic procedure, providing results that are usually less than 1% above those of the NSearch procedure. The performance of the FBS\_R and RBS\_R algorithms is comparatively poor, since they are outperformed by even the simple EXPET dispatch rule. The algorithms with a priority evaluation function filtering procedure clearly outperform their rules-based counterparts. The simple rules that were used cannot avoid eliminating nodes that would lead to good solutions and better rules would therefore be required in order to make the rule filter procedures competitive. The recovering beam search algorithms also provide better results than their filtered beam search alternatives, for both types of filtering procedure. From table 2 we can also see that the use of the dominance rules improves the heuristic results. The Wilcoxon test values also indicate that the differences in distribution between the heuristic results before and after the dominance rules are statistically significant. The improvement provided by the dominance rules increases with the instance size and is higher for instances with low processing time and penalty variability. The results of the NSearch heuristic are only slightly improved by the dominance rules, which is hardly surprising, since this procedure already performs an extensive local search. The improvement is higher for the worst performing heuristics, but even the RBS\_P and FBS\_P algorithms can benefit from 2% to 4% (1% to 2%) improvements for medium and large instances with low (high) variability.

The effect of the  $L$  and  $R$  parameters on the relative objective function value improvement for the RBS\_P heuristic is given in table 3. The lateness factor  $L$  has a clear effect on the improvement provided by the dominance rules: the relative improvement is higher for  $L$  values of 0.4 and 0.6, and decreases as the lateness factor

n	Heur	low var					high var				
		mean ofv			#best		mean ofv			#best	
		bfr	aft	%ch	bfr	aft	bfr	aft	%ch	bfr	aft
25	EXPET	3340	3293	2.5	142	326	260429	257669	2.0	126	300
	FBS_P	3265	3251	0.8	406	657	253815	253079	0.6	380	619
	FBS_R	3356	3312	2.3	257	382	262064	260193	1.3	202	297
	NSearch	3238	3235	0.1	933	895	251143	251142	0.0	923	892
	RBS_P	3251	3245	0.4	670	697	252205	252088	0.1	622	621
	RBS_R	3340	3307	1.7	375	400	260509	259878	0.5	309	326
50	EXPET	12854	12648	3.0	41	128	989785	980605	1.9	24	104
	FBS_P	12681	12584	1.5	130	322	974932	971959	0.7	106	275
	FBS_R	12971	12741	3.2	97	156	999953	993667	1.2	54	96
	NSearch	12486	12469	0.3	951	865	959949	959777	0.0	923	875
	RBS_P	12604	12543	0.9	342	375	967496	966348	0.2	372	382
	RBS_R	12923	12729	2.7	158	173	995566	992580	0.5	117	118
100	EXPET	50435	49504	3.6	6	41	3888904	3857544	1.8	3	19
	FBS_P	50091	49433	2.6	56	186	3860076	3846720	0.7	38	125
	FBS_R	50772	49738	3.8	30	78	3924426	3899674	1.2	8	17
	NSearch	49169	49012	0.7	987	887	3779215	3778321	0.1	1015	964
	RBS_P	49831	49321	1.9	155	170	3835655	3824142	0.5	159	169
	RBS_R	50653	49708	3.5	78	79	3913752	3896253	0.8	40	37
250	EXPET	311547	304605	4.5	0	12	23964254	23741373	2.0	0	4
	FBS_P	310537	304515	3.9	20	114	23892049	23730626	1.3	33	80
	FBS_R	312393	305158	4.6	11	45	24105594	23889467	1.7	0	1
	NSearch	304462	302721	1.2	1109	935	23408844	23371735	0.3	1139	1082
	RBS_P	309496	304154	3.4	19	73	23810268	23664902	1.1	35	37
	RBS_R	312038	305061	4.4	45	39	24069759	23879233	1.4	1	1
500	EXPET	1236802	1207258	4.8	128	165	95017831	93922951	2.3	39	95
	FBS_P	1234168	1207035	4.4	240	401	94842757	93890141	1.9	600	624
	FBS_R	1237715	1208056	4.8	50	164	95327723	94216125	2.2	1	1
	NSearch	—	—	—	—	—	—	—	—	—	—
	RBS_P	1231300	1206041	4.2	442	379	94630661	93757891	1.8	468	441
	RBS_R	1236800	1207849	4.7	366	156	95246781	94193574	2.0	107	57

Table 2: Heuristic results: objective function value and number of times each heuristic gives the best result

approaches its extreme values. When  $L$  is equal to 0.0 or 1.0, the dominance rules provide little or no improvement. This result is to be expected, since the heuristics are more likely to be closer to the optimum for the extreme  $L$  values, where the early/tardy problem is easier, therefore limiting the dominance rules' possibilities for improvement. In fact, an optimum schedule could easily be generated if all jobs were early or late (see [9] for details), and for a lateness factor of 0.0 (1.0) most jobs will indeed be early (late). For intermediate  $L$  values there is a greater balance between the number of early and tardy jobs, and the problem becomes much harder.

n	$L$	low var				high var			
		$R=0.2$	$R=0.4$	$R=0.6$	$R=0.8$	$R=0.2$	$R=0.4$	$R=0.6$	$R=0.8$
100	0.0	0.0	0.1	0.3	0.5	0.0	0.0	0.3	0.3
	0.2	1.0	0.6	2.0	2.7	0.1	0.1	0.9	1.8
	0.4	4.3	3.9	4.0	3.5	1.0	0.5	0.9	1.3
	0.6	8.1	4.5	4.2	1.8	1.8	1.1	0.4	0.6
	0.8	2.9	0.1	0.2	0.2	0.5	0.1	0.1	0.1
	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
250	0.0	0.0	0.2	0.3	0.4	0.0	0.1	0.3	0.4
	0.2	2.0	1.3	1.3	1.4	0.5	0.3	1.1	1.4
	0.4	8.8	7.3	7.2	7.6	2.8	1.7	1.5	2.2
	0.6	13.8	11.8	8.5	2.6	5.2	3.4	1.6	1.5
	0.8	6.0	0.2	0.3	0.4	2.2	0.0	0.1	0.2
	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3: Relative improvement for the RBS\_P heuristic

In table 4 we present the heuristic runtimes (in seconds); results obtained after the application of the dominance rules are indicated by appending "+ DR" to the heuristic identifiers. We can see that the dominance rules require little additional computational effort, and their use is therefore recommended, since they allow for improvements in the objective function values. The NSearch heuristic is computationally demanding, and can therefore only be used for small or medium size instances. The variability of the processing times and penalties only has a significant effect on the runtimes of the FBS\_R and RBS\_R procedures, which are faster when the variability is high. The algorithms with a rule-based filtering procedure are then faster than their priority evaluation function counterparts when the variability is high. For a low processing time and penalty variability, however, the priority function procedures are faster for instances with 250 or more jobs. The recovering algorithms are much faster than their filtered alternatives, and can solve

even medium and large instances within reasonable computation times.

Heur	low var				high var			
	n=200	n=250	n=400	n=500	n=200	n=250	n=400	n=500
EXPET	0.001	0.002	0.004	0.007	0.001	0.002	0.004	0.007
FBS_P	1.203	2.355	9.584	19.048	1.239	2.383	9.745	19.266
FBS_R	1.052	2.288	11.799	26.233	0.510	1.027	4.444	9.037
NSearch	10.939	25.644	—	—	11.121	25.984	—	—
RBS_P	0.579	1.094	4.143	7.977	0.603	1.145	4.356	8.230
RBS_R	0.563	1.172	5.599	12.334	0.315	0.585	2.227	4.464
EXPET + DR	0.005	0.008	0.022	0.038	0.002	0.004	0.009	0.016
FBS_P + DR	1.206	2.362	9.605	19.085	1.241	2.386	9.751	19.279
FBS_R + DR	1.056	2.295	11.822	26.273	0.512	1.030	4.452	9.050
Nsearch + DR	10.941	25.647	—	—	11.121	25.985	—	—
RBS_P + DR	0.583	1.102	4.166	8.017	0.604	1.147	4.364	8.244
RBS_R + DR	0.567	1.178	5.621	12.372	0.316	0.587	2.233	4.476

Table 4: Runtimes (in seconds)

The heuristic results were also compared with the optimum objective function values for instances with up to 30 jobs. In table 5 we present the average of the relative deviations from the optimum (%dev), calculated as  $(H - O) / O * 100$ , where  $H$  and  $O$  are the heuristic and optimum objective function values, respectively. The number of times each heuristic generates an optimum schedule (#opt) is also given. All the heuristics are closer to the optimum when the variability is low. The average performance of the NSearch heuristic is quite good, since it provides results that are 1.5% to 2.5% above the optimum and it provides an optimum solution for about 50% of the test instances. The performance of the RBS\_P procedure is also quite adequate. This heuristic generates solutions that are 2% to 3% above the optimum and it calculates an optimum schedule for about 50% (30%) of the 20 (30) job instances.

In table 6 we present the effect of the  $L$  and  $R$  parameters on the relative deviation from the optimum for the RBS\_P + DR heuristic. The lateness factor  $L$  has a significant impact on the relative deviation from the optimum. The heuristics are optimal or nearly optimal when  $L$  is equal to 0.0 or 1.0, and the performance deteriorates as the lateness factor assumes values nearer the middle of its range. These results are once again expected, since the early/tardy problem is harder for the intermediate  $L$  values.

The NSearch heuristic provides very good results, but can only be applied to

Heur	low var				high var			
	n=20		n=30		n=20		n=30	
	%dev	#opt	%dev	#opt	%dev	#opt	%dev	#opt
EXPET	8.6	170	9.0	107	9.6	161	10.4	76
FBS_P	2.7	428	4.5	233	3.5	399	5.5	198
FBS_R	9.2	323	11.0	200	10.3	313	12.9	129
NSearch	1.4	746	1.9	498	1.6	705	2.4	506
RBS_P	2.0	566	3.2	334	2.5	577	3.5	330
RBS_R	8.2	396	10.0	257	9.1	416	11.6	209
EXPET + DR	5.2	339	5.9	200	7.1	353	7.9	178
FBS_P + DR	1.9	626	3.4	377	2.8	625	4.7	378
FBS_R + DR	6.4	421	7.8	261	9.0	407	11.4	192
Nsearch + DR	1.3	751	1.8	501	1.6	705	2.4	506
RBS_P + DR	1.7	602	2.7	359	2.4	600	3.3	348
RBS_R + DR	6.3	418	7.5	273	8.7	420	11.1	216

Table 5: Comparison with optimum objective function values

n	L	low var				high var			
		R=0.2	R=0.4	R=0.6	R=0.8	R=0.2	R=0.4	R=0.6	R=0.8
		20	0.0	0.0	0.1	0.2	0.2	0.0	0.1
	0.2	2.3	2.5	1.5	1.4	1.4	2.2	3.2	1.9
	0.4	2.0	3.6	6.7	3.7	4.4	4.4	9.1	7.0
	0.6	3.9	4.6	3.3	2.6	5.1	5.1	4.7	5.7
	0.8	1.1	0.4	0.6	0.6	1.8	0.8	0.6	0.4
	1.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.1
30	0.0	0.0	0.1	0.1	0.2	0.0	0.1	0.1	0.4
	0.2	1.4	3.3	2.2	2.0	1.6	3.2	1.9	1.8
	0.4	4.9	6.5	9.0	8.9	6.7	7.3	8.8	13.1
	0.6	4.4	6.9	5.6	6.0	6.8	8.9	8.0	7.1
	0.8	1.3	0.6	0.5	0.9	1.4	1.2	1.0	0.7
	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1

Table 6: Relative deviation from the optimum for the RBS\_P + DR heuristic

small or medium size instances due to its high computational requirements. The RBS\_P algorithm is close to the NSearch heuristic in solution quality, particularly after the application of the dominance rules, and is significantly faster. Therefore, this procedure is the heuristic of choice for medium and even large size problems.

## 5 Conclusion

In this paper we considered the single machine scheduling problem with earliness and tardiness penalties and no idle time. We presented heuristic algorithms based on the filtered and recovering beam search techniques and compared them with existing neighbourhood search and dispatch rule heuristics. A filtering procedure is required by the filtered and recovering beam search algorithms. We have considered filtering procedures using both priority evaluation functions and problem-specific properties. The beam search algorithms and the neighbourhood search heuristic use several parameters whose value must be specified. We performed extensive computational tests to determine the parameter values that provided the best balance between solution quality and computational effort. The use of some dominance rules to improve the solutions obtained by the heuristics was also considered.

The computational results show that the dominance rules can improve the solution quality with little additional computational effort, and their use is therefore recommended. The algorithms with a priority evaluation function filtering procedure outperform their rules-based counterparts in solution quality. The recovering beam search procedures are clearly superior to the filtered beam search alternatives, since they not only provide better solutions, but are also faster. The best results are given by the NSearch heuristic, but this algorithm is computationally demanding and can be applied only to small or medium size instances. The RBS\_P procedure provides results that are close to the best in solution quality and is significantly faster. Therefore, this procedure is then the heuristic of choice for medium and even large size problems. The performance of the recovering beam search algorithm was quite adequate, and this heuristic approach seems to strike a good balance between solution quality and computational efficiency. These results confirm the potential of the recently introduced recovering beam search technique, and as a possible step for future research it certainly seems worthy to investigate its behaviour on other problems.

## References

- [1] ABDUL-RAZAQ, T., AND POTTS, C. N. Dynamic programming state-space relaxation for single machine scheduling. *Journal of the Operational Research Society* 39 (1988), 141–152.
- [2] BAKER, K. R., AND SCUDDER, G. D. Sequencing with earliness and tardiness penalties: A review. *Operations Research* 38 (1990), 22–36.
- [3] DELLA CROCE, F., AND T’KINDT, V. A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem. *Journal of the Operational Research Society* 53 (2002), 1275–1280.
- [4] FOX, M. S. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Ph.d. thesis, Carnegie-Mellon University, USA, 1983.
- [5] LENSTRA, J. K., RINNOOY KAN, A. H. G., AND BRUCKER, P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics* 1 (1977), 343–362.
- [6] LI, G. Single machine earliness and tardiness scheduling. *European Journal of Operational Research* 96 (1997), 546–558.
- [7] LIAW, C.-F. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers & Operations Research* 26 (1999), 679–693.
- [8] LOWERRE, B. T. *The HARPY Speech Recognition System*. Ph.d. thesis, Carnegie-Mellon University, USA, April 1976.
- [9] OW, P. S., AND MORTON, E. T. The single machine early/tardy problem. *Management Science* 35 (1989), 177–191.
- [10] OW, P. S., AND MORTON, T. E. Filtered beam search in scheduling. *International Journal of Production Research* 26 (1988), 35–62.
- [11] OW, P. S., AND SMITH, S. F. Viewing scheduling as an opportunistic problem-solving process. *Annals of Operations Research* 12 (1988), 85–108.
- [12] RUBIN, S. *The ARGOS Image Understanding System*. Ph.d. thesis, Carnegie-Mellon University, USA, April 1978.

- [13] SABUNCUOGLU, I., AND BAYIZ, M. Job shop scheduling with beam search. *European Journal of Operational Research* 118 (1999), 390–412.
- [14] VALENTE, J. M. S., AND ALVES, R. A. F. S. Improved heuristics for the early/tardy scheduling problem with no idle time. Working Paper 126, Faculdade de Economia do Porto, Portugal, 2003.
- [15] VALENTE, J. M. S., AND ALVES, R. A. F. S. Improved lower bounds for the early/tardy scheduling problem with no idle time. Working Paper 125, Faculdade de Economia do Porto, Portugal, 2003.

## **Working papers mais recentes**

Nº 141	João A. Ribeiro and Robert W. Scapens, <a href="#"><i>Power, ERP systems and resistance to management accounting: a case study</i></a> , April 2004
Nº 140	Rosa Forte, <a href="#"><i>The relationship between foreign direct investment and international trade. Substitution or complementarity? A survey</i></a> , March 2004
Nº 139	Sandra Silva, <a href="#"><i>On evolutionary technological change and economic growth: Lakatos as a starting point for appraisal</i></a> , March 2004
Nº 138	Maria Manuel Pinho, <a href="#"><i>Political models of budget deficits: a literature review</i></a> , March 2004
Nº 137	Natércia Fortuna, <a href="#"><i>Local rank tests in a multivariate nonparametric relationship</i></a> , February 2004
Nº 136	Argentino Pessoa, <a href="#"><i>Ideas driven growth: the OECD evidence</i></a> , December 2003
Nº 135	Pedro Lains, <a href="#"><i>Portugal's Growth Paradox, 1870-1950</i></a> , December 2003
Nº 134	Pedro Mazedo Gil, <a href="#"><i>A Model of Firm Behaviour with Equity Constraints and Bankruptcy Costs</i></a> , November 2003
Nº 133	Douglas Woodward, Octávio Figueiredo and Paulo Guimarães, <a href="#"><i>Beyond the Silicon Valley: University R&amp;D and High-Technology Location</i></a> , November 2003.
Nº 132	Pedro Cosme da Costa Vieira, <a href="#"><i>The Impact of Monetary Shocks on Product and Wages: A neoclassical aggregated dynamic model</i></a> , July 2003.
Nº 131	Aurora Teixeira and Natércia Fortuna, <a href="#"><i>Human Capital, Innovation Capability and Economic Growth</i></a> , July 2003.
Nº 130	Jorge M. S. Valente and Rui A. F. S. Alves, <a href="#"><i>Heuristics for the Early/Tardy Scheduling Problem with Release Dates</i></a> , May 2003.
Nº 129	Jorge M. S. Valente and Rui A. F. S. Alves, <a href="#"><i>An Exact Approach to Early/Tardy Scheduling with Release Dates</i></a> , May 2003.
Nº 128	Álvaro Almeida, <a href="#"><i>40 Years of Monetary Targets and Financial Crises in 20 OECD Countries</i></a> , April 2003.
Nº 127	Jorge M. S. Valente, <a href="#"><i>Using Instance Statistics to Determine the Lookahead Parameter Value in the ATC Dispatch Rule: Making a good heuristic better</i></a> , April 2003.
Nº 126	Jorge M. S. Valente and Rui A. F. S. Alves, <a href="#"><i>Improved Heuristics for the Early/Tardy Scheduling Problem with No Idle Time</i></a> , April 2003.
Nº 125	Jorge M. S. Valente and Rui A. F. S. Alves, <a href="#"><i>Improved Lower Bounds for the Early/Tardy Scheduling Problem with No Idle Time</i></a> , April 2003.
Nº 124	Aurora Teixeira, <a href="#"><i>Does Inertia Pay Off? Empirical assessment of an evolutionary-ecological model of human capital decisions at firm level</i></a> , March 2003.
Nº 123	Alvaro Aguiar and Manuel M. F. Martins, <a href="#"><i>Macroeconomic Volatility Trade-off and Monetary Policy Regime in the Euro Area</i></a> , March 2003.
Nº 122	Alvaro Aguiar and Manuel M. F. Martins, <a href="#"><i>Trend, cycle, and non-linear trade-off in the Euro Area 1970-2001</i></a> , March 2003.
Nº 121	Aurora Teixeira, <a href="#"><i>On the Link between Human Capital and Firm Performance. A Theoretical and Empirical Survey</i></a> , November 2002.
Nº 120	Ana Paula Serra, <a href="#"><i>The Cross-Sectional Determinants of Returns: Evidence from Emerging Markets' Stocks</i></a> , October 2002.
Nº 119	Cristina Barbot, <a href="#"><i>Does Airport Regulation Benefit Consumers?</i></a> , June 2002.
Nº 118	José Escaleira, <a href="#"><i>A Procura no Sector das Artes do Espectáculo. Tempo e Rendimento na Análise das Audiências. Um Estudo para Portugal</i></a> ,

	June 2002.
Nº 117	Ana Paula Serra, <a href="#">Event Study Tests: A brief survey</a> , May 2002.
Nº 116	Luís Delfim Santos and Isabel Martins, <a href="#">A Qualidade de Vida Urbana - O caso da cidade do Porto</a> , May 2002.
Nº 115	Marcelo Cabús Klötzle and Fábio Luiz Biagini, <a href="#">A Restruturação do Sector Eléctrico Brasileiro: Uma análise comparativa com a Califórnia</a> , January 2002.
Nº 114	António Brandão and Sofia B. S. D. Castro, <a href="#">Objectives of Public Firms and Entry</a> , December 2001.
Nº 113	Ana Cristina Fernandes and Carlos Machado-Santos, <a href="#">Avaliação de Estratégias de Investimento com Opções</a> , December 2001.
Nº 112	Carlos Alves and Victor Mendes, <a href="#">Corporate Governance Policy and Company Performance: The Case of Portugal</a> , December 2001.
Nº 111	Cristina Barbot, <a href="#">Industrial Determinants of Entry and Survival: The case of Ave</a> , October 2001.
Nº 110	José Rodrigues de Jesús, Luís Miranda da Rocha e Rui Couto Viana, <a href="#">Avaliação de Pequenas e Médias Empresas e Gestão de Risco</a> , October 2001.
Nº 109	Margarida de Mello and Kevin S. Nell, <a href="#">The Forecasting Ability of a Cointegrated VAR Demand System with Endogeneous vs. Exogenous Expenditure Variable: An application to the UK imports of tourism from neighbouring countries</a> , July 2001.
Nº 108	Cristina Barbot, <a href="#">Horizontal Merger and Vertical Differentiation</a> , June 2001.
Nº 107	Celsa Machado, <a href="#">Measuring Business Cycles: The Real Business Cycle Approach and Related Controversies</a> , May 2001.
Nº 106	Óscar Afonso, <a href="#">The Impact of International Trade on Economic Growth</a> , May 2001.
Nº 105	Abraão Luís Silva, <a href="#">Chamberlain on Product Differentiation, Market Structure and Competition: An essay</a> , May 2001.
Nº 104	Helena Marques, <a href="#">The "New" Economic Theories</a> , May 2001.
Nº 103	Sofia B. S. D. Castro and António Brandão, <a href="#">Public Firms in a Dynamic Third Market Model</a> , January 2001.
Nº 102	Bernard Friot, Bernadette Clasquin & Nathalie Moncel, <a href="#">Salaire, Fiscalité et Épargne dans le Financement de l'Emploi et de la Protection Sociale: l'Exemple Européen</a> , January 2001.
Nº 101	Paulo Beleza Vasconcelos, <a href="#">Resolução Numérica de Modelos Macroeconómicos com Expectativas Racionais</a> , 2000.
Nº 100	Luis David Marques, <a href="#">Modelos Dinâmicos com Dados em Painel: Revisão da Literatura</a> , 2000.
Nº 99	Rui Henrique Alves, <a href="#">Da Moeda Única à União Política?</a> , 2000.
Nº 98	Paulo Guimarães, Octávio Figueiredo & Doug Woodward, <a href="#">A Tractable Approach to the Firm Location Decision Problem</a> , 2000.
Nº 97	António Brandão & José Escaleira, <a href="#">Trade Policy and Tacit Collusion with Price and Quantity Competition</a> , 2000.
Nº 96	Sandra Silva & Mário Rui Silva, <a href="#">Crescimento Económico nas Regiões Europeias: Uma Avaliação sobre a Persistência das Disparidades Regionais no Período 1980-95</a> , 2000.
Nº 95	José Manuel Moreira, <a href="#">Ética, Estado e Desenvolvimento Económico. Heterodoxia e Ortodoxia</a> , 2000.

Editor: Prof. Aurora Teixeira ([ateixeira@fep.up.pt](mailto:ateixeira@fep.up.pt))

Download dos artigos em:

<http://www.fep.up.pt/investigacao/workingpapers/workingpapers.htm>



FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

[www.fep.up.pt](http://www.fep.up.pt)