Investigação - Trabalhos em curso - nº 143, Abril 2004

Beam search algorithms for the early/tardy scheduling problem with release dates

Jorge M. S. Valente

and

Rui A. F. S. Alves

FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

www.fep.up.pt

Beam search algorithms for the early/tardy scheduling problem with release dates

Jorge M. S. Valente and Rui A. F. S. Alves
Faculdade de Economia, Universidade do Porto

April 13, 2004

Abstract

In this paper we consider the single machine earliness/tardiness scheduling problem with different release dates and no unforced idle time. We present several heuristic algorithms based on the beam search technique. These algorithms include classical beam search procedures, with both priority and total cost evaluation functions, as well as the filtered and recovering variants. Both priority evaluation functions and problem-specific properties were considered for the filtering step used in the filtered and recovering beam search heuristics. Extensive preliminary tests were performed to determine appropriate values for the parameters used by each algorithm.

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 beam search procedure with a total cost evaluation function provides very good results, but is computationally expensive and can therefore only be applied to small or medium size instances. The recovering algorithm is quite 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 são apresentadas diversas heurísticas baseadas na técnica *beam search* para o problema de sequenciamento com um único processador, custos de posse e atraso, datas de disponibilidade distintas e inexistência de tempo morto não forçado. Estas heurísticas incluem procedimentos *beam search* clássicos (utilizando funções prioridade e funções custo total), bem como as variantes *filtered* e *recovering beam search*. Dois diferentes tipos de procedimentos de filtragem - funções prioridade e regras relativas ao problema em causa - foram considerados para as heurísticas baseadas em *filtered* e *recovering beam search*. Diversos testes computacionais foram efectuados para determinar valores apropriados para os parâmetros usados pelos diversos algoritmos.

Os testes computationais 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 qua utiliza funções prioridade revelou-se substancialmente melhor do que o baseado em regras. O procedimento de *beam search* com uma função custo total proporcionou muito bons resultados, mas exige um elevado esforço computacional, pelo que apenas pode ser aplicado a instâncias de dimensão pequena ou média. 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

In this paper we consider a single machine scheduling problem with release dates and earliness and tardiness costs that can be stated as follows. A set of n independent jobs $\{J_1, J_2, \cdots, J_n\}$ has to be scheduled without preemptions on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and unforced machine idle time is not allowed. Job $J_j, j = 1, 2, \cdots, n$, becomes available for processing at its release date r_j, 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 minimises 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. It is assumed that no unforced machine idle time is allowed, so the machine is only idle if no job is currently available for processing. This assumption reflects a production setting where the cost of machine idleness is higher than the early cost incurred by completing any job before its due date, or the capacity of the machine is limited when compared with its demand, so that the machine must indeed be kept running. Some specific examples of production settings with these characteristics are provided by Korman [5] and Landis [6]. The existence of different release dates is compatible with the assumption of no unforced idle time, as long as the forced idle time caused by the presence of distinct release dates is small or inexistent. If that is not the case, that assumption becomes unrealistic, since it is then highly unlikely that either the machine idleness cost is higher than the early cost or the machine capacity is limited when compared with the demand.

As a generalization of weighted tardiness scheduling [7], the problem is strongly NP-hard. To the best of our knowledge, the only work in this problem is due to Valente and Alves ([16], [17]). In [16] they presented a branch-and-bound algorithm based on a decomposition of the problem into weighted earliness and weighted tardiness subproblems. Two lower bound procedures were presented for each subproblem, and the lower bound for the original problem is then simply the sum of the lower bounds for the two subproblems. In [17] they analyse the performance of various heuristic procedures, including dispatch rules, a greedy procedure and a decision theory search heuristic. The early/tardy problem with equal release dates and no idle time, however, has been considered by several authors, and both exact and heuristic approaches have been proposed. Among the exact approaches, branch-and-bound algorithms were presented by Abdul-Razaq and Potts [1], Li [8] and Liaw [9]. The lower bounding procedure of Abdul-Razaq and Potts was based on the subgradient optimization approach and the dynamic programming state-space relaxation technique, while Li and Liaw used Lagrangean relaxation and the multiplier adjustment method. Among the heuristics, Ow and Morton [11] developed several dispatch rules and a filtered beam search procedure. Valente and Alves [18] presented an additional dispatch rule and a greedy procedure, and also considered the use of dominance rules to further improve the schedule obtained by the heuristics. A neighbourhood search algorithm was also presented by Li [8].

In this paper we present several heuristic algorithms based on the beam search technique. These algorithms include classical beam search procedures, with both priority and total cost evaluation functions, as well as the filtered and recovering variants. We have considered both priority evaluation functions and problem-specific properties for the filtering step used in the filtered and recovering beam search heuristics. Extensive computational tests were performed to determine the parameter values that provided the best balance between solution quality and computational effort for each algorithm. 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 [10] and the image understanding [14] problems. A number of applications to scheduling problems have appeared in the literature since then. Fox [3] and Ow and Smith [13] have incorporated a beam search procedure in systems designed for complex job shop environments. Sabuncuoglu and Bayiz [15] presented beam search algorithms for the job shop problem with both makespan and mean tardiness as performance measures. Ow and Morton ([12], [11]) 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 [2] 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 β nodes (instead of all nodes) at each level of the search tree are retained for further branching; β 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.

The priority evaluation function approach can pose a slight problem. The dispatch rules used for calculating the urgency rating of the last scheduled job are usually functions of the current partial schedule, namely functions of the current time. Different nodes on the same level correspond to different partial schedules and have different completion times. Therefore, the priorities obtained for the offspring of a node cannot be legitimately compared with the priorities obtained from expanding another node at the same level; these priorities are then context-dependent. This problem can be overcome by initially selecting the best β children of the parent or root node (i.e., the node containing only unscheduled jobs). At lower levels of the search tree we find the most promising descendant of each node and retain it for the next iteration. Thus, only the best descendant of each beam node is saved for the next iteration. The total cost evaluation function is not affected by this problem, since the total cost estimates are context-independent and can be compared. We now present the main steps of both Priority Beam Search and Detailed (or Probe) Beam Search algorithms. Priority beam search uses a priority evaluation function, while detailed beam search uses a total cost evaluation function. 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.

Priority Beam Search:

1. Initialization:

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

   Branch n_0 generating the corresponding children.

   Perform a priority evaluation for each child node (usually by calculating the priority of the last scheduled job using a dispatch rule).

   Select the min $\{\beta$, number of children$\}$ best child nodes (usually the nodes with the highest priority value) and add them to B.

2. For each node in B:

   Branch the node generating the corresponding children.

   Perform a priority evaluation for each child node (usually by calculating the priority of the last scheduled job using a dispatch rule).

   Select the best child node and add it to C.

3. Set $B = C$.

   Set $C = \varnothing$.

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.

Detailed (Probe) Beam Search:

1. Initialization:

   Set $C = \varnothing$.

   Set $B = \{n_0\}$.

2. For each node in B:

    (a) Branch the node generating the corresponding children.

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

    (c) Select the $\min\{\beta, \text{number of children}\}$ best child nodes (usually the nodes with the lowest upper bound) and add them to C.

3. Set $B = \varnothing$.

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

    Set $C = \varnothing$.

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.

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 β 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 α children of each beam node for the detailed evaluation step; α is the so-called *filter width*. Recently, a different filtering procedure was used by Della Croce and T'kindt [2]. 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.

Filtered Beam Search:

1. Initialization:

   Set $C = \varnothing$.

   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 $B = \varnothing$.

   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 $C = \varnothing$.

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 σ is dominated by another partial solution $\overline{\sigma}$

having the same level of the search tree (typically by applying interchange operators to the current partial solution); if so, $\overline{\sigma}$ 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 σ 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 $C = \varnothing$.

   Set $\sigma = n_0$.

2. Branch σ 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 σ^* be the node in C with the lowest value of V.

   Set $\sigma = \sigma^*$.

   Set $C = \varnothing$.

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

6. Stopping condition:

   If σ is a leaf node (it holds a complete sequence), stop; σ'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 several algorithms that were considered. We tested 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 early/tardy problem, it is necessary to specify their main components, namely the branching scheme, priority evaluation function, upper and lower bounding procedures, filtering procedure and recovering step. The branching scheme, common to all algorithms, 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. The priority evaluation function required by the priority beam search algorithm is provided by the LINET dispatch rule. This heuristic (originally developed by Ow and Morton [11] for the problem with identical release dates) provided the best results of all the dispatch rules analysed by Valente and Alves [17]. The priority index of the LINET heuristic is used to calculate the priority of the last scheduled job in each node. The detailed beam search procedure also uses the LINET dispatch rule in its total cost evaluation function. The LINET heuristic is used to complete the existing partial schedule and therefore calculate a total cost estimate.

The filtering procedure is used by both filtered and recovering beam search approaches. We considered the two types of filtering procedures that have been previously proposed. The first requires a priority evaluation function and selects the α best children of each beam node for the detailed evaluation step. This priority evaluation function is identical to the one used in the priority beam search algorithm. 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 feasibly 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 LINET dispatch rule is used to complete the existing partial schedule and calculate a total cost estimate, just as in the detailed beam search. 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 LINET heuristic. The lower bound is computed using a procedure presented by Valente and Alves [16]. This procedure relaxes the assumption that a job cannot be scheduled before its release date in order to calculate a lower bound for a problem with identical release dates. 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 $\delta\left(n-1\right), 0 \leq \delta \leq 1$ insertions have been performed; this insertion procedure is also stopped if the next insertion would lead to an infeasible schedule (i.e., the last job in the current partial sequence would be scheduled to start before its release date). The parameter δ 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 priority and detailed (or probe) beam search algorithms will be denoted as PBS and DBS, respectively. The filtered 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. We must also remark that the existence of different release dates motivated a slight change in the PBS, FBS_P and RBS_P procedures. In the previous section we indicated that the PBS procedure selects only the best child of each beam node, while the other two algorithms will also select just a single child node for detailed evaluation when the filter width is one. When release dates are allowed to be different, it is quite possible that the root node has only one (or very few) offspring. If no correction was made to the algorithms, only one node would be retained in the beam throughout the whole procedure, independently of the beam width. Therefore, in such situations the number of chosen offspring is increased temporarily in order to allow the number of beam nodes to increase up to β. The proposed algorithms were compared with two other heuristics, namely the LINET dispatch rule and the Decision Theory Search (DTS) algorithm analysed in Valente and Alves [17]. The DTS algorithm is based on the decision theory approach of Kanet and Zhou [4], but is identical to the detailed beam search algorithm with a beam width of one.

Ow and Morton [11] and Liaw [9] developed dominance rules for the problem with identical release dates. 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. These rules can still be used when the release dates are allowed to differ, provided care is taken to avoid making unfeasible job swaps. Valente and Alves [17] showed that these dominance rules could be used to improve the solution quality of several heuristic procedures with little additional computational effort. 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, 75, 100, 200, 250, 300, 400, 500 and 1000 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 release date r_j was generated from the uniform distribution $\left[0, R\sum_{j=1}^{n} p_j\right]$, where R was set at 0.25, 0.50 and 0.75. The maximum value of the range of release dates R was chosen so that the forced idle time would be small or inexistent. Preliminary tests showed that $R = 1.00$ would lead to excessive amounts of forced idle time, which would be incompatible with the assumption that no unforced idle time may be inserted in a schedule. Instead of determining due dates directly, we generated slack times between a job's due date and its earliest possible completion time. For each job J_j, an integer due date slack s_j^d was generated from the uniform distribution $\left[0, D\sum_{j=1}^{n} p_j\right]$, where the due date slack range D was set at 0.10, 0.25 and 0.50. The due date d_j of J_j was then set equal to $d_j = (r_j + p_j) + s_j^d$. For each combination of instance size, processing time and penalty variability, R and D, 50 instances were randomly generated. All the algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV - 1500 Mhz personal computer. Due to the large computational times that would be required, the DTS heuristic was not applied to the 1000 job instances, while the DBS algorithm was only used to solve instances with up to 400 jobs. 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 between solution quality and computational time, since increasing the value of the parameters usually improves the objective function value, but at the cost of increased computation times (the only exception being the γ 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:

$\alpha = \{1, 2, \ldots, 10\}$,
$\beta = \{1, 2, \ldots, 8\}$,
$\gamma = \{0.1, 0.2, \ldots, 0.9\}$,
$\delta = \{0.05, 0.10, \ldots, 0.50\}$.

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 computation time were selected. 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.

Heur	α	β	γ	δ
PBS	—	4	—	—
DBS	—	3	—	—
FBS_P	3	3	—	—
FBS_R	—	3	—	—
RBS_P	3	—	0.8	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 were never 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 usually given by the DBS heuristic. The DTS and RBS_P are then the best performing algorithms, followed by the FBS_P procedure. The RBS_P and DTS algorithms, in particular, are quite close to the best heuristic procedure, providing results that are usually less than 0.5% (1%) above those of the DBS algorithm for instances with low (high) processing time and penalty variability. The CBS procedure provides better results than the LINET dispatch rule, particularly for instances with high variability. The performance of the FBS_R and RBS_R algorithms is comparatively poor, since they are outperformed by even the simple LINET dispatch procedure. The algorithms with a priority evaluation function filtering procedure clearly outperform their rules-based counterparts. The simple rules that were used cannot avoid elim-

		low var					high var				
		mean ofv			# best		mean ofv			# best	
n	Heur	bfr	aft	%ch	bfr	aft	bfr	aft	%ch	bfr	aft
50	PBS	6116	5993	2.5	0	20	463630	455351	2.4	0	15
	DBS	5862	5813	1.1	132	199	442640	439277	1.0	140	216
	DTS	5924	5854	1.5	27	99	446444	441617	1.4	32	123
	FBS_P	5875	5844	0.7	107	164	444064	442086	0.6	84	161
	FBS_R	6210	6109	1.9	13	26	471941	467434	1.2	8	26
	LINET	6174	6048	2.5	1	15	469706	461679	2.3	0	11
	RBS_P	5842	5823	0.3	228	169	441436	440634	0.2	230	174
	RBS_R	6179	6109	1.2	35	27	468312	466825	0.3	34	28
100	PBS	21914	21546	2.1	1	3	1688047	1668808	1.6	0	2
	DBS	21115	20898	1.4	153	192	1615371	1602600	1.1	162	205
	DTS	21264	20991	1.7	29	85	1627424	1612292	1.3	25	82
	FBS_P	21216	21086	0.8	81	72	1629726	1624045	0.5	74	68
	FBS_R	22197	21828	2.1	0	4	1722101	1707834	1.1	0	2
	LINET	22052	21665	2.1	0	2	1705855	1686123	1.6	0	1
	RBS_P	21124	20996	0.7	190	136	1618767	1614084	0.4	196	140
	RBS_R	22089	21791	1.6	6	4	1711754	1706704	0.3	5	2
250	PBS	129856	127219	2.2	3	9	9763218	9696879	0.9	0	1
	DBS	126444	124716	1.8	169	169	9413967	9362293	0.7	217	234
	DTS	127172	125077	2.3	44	91	9446220	9389107	0.8	45	88
	FBS_P	127418	125904	1.4	88	57	9531987	9508352	0.3	73	45
	FBS_R	131051	128155	2.4	0	1	9895687	9828904	0.9	0	0
	LINET	130382	127601	2.3	1	2	9802122	9731758	0.9	0	1
	RBS_P	126958	125301	1.5	144	123	9500053	9471279	0.4	122	104
	RBS_R	130583	127992	2.1	1	0	9853781	9820274	0.3	2	1
500	PBS	502348	490681	2.5	7	8	37199701	36978650	0.7	0	0
	DBS	—	—	—	—	—	—	—	—	—	—
	DTS	494202	484877	2.6	183	246	36072578	35916221	0.6	294	329
	FBS_P	496529	488347	1.8	84	56	36598921	36489490	0.3	65	37
	FBS_R	504867	492401	2.7	0	0	37528605	37286214	0.8	0	0
	LINET	503212	491349	2.5	0	9	37303483	37072673	0.7	0	0
	RBS_P	494709	486566	1.8	173	128	36536430	36383156	0.5	91	84
	RBS_R	503587	491907	2.4	3	3	37419585	37253876	0.4	0	0
1000	PBS	1954877	1899942	2.9	13	33	145069697	143974245	0.8	0	2
	DBS	—	—	—	—	—	—	—	—	—	—
	DTS	—	—	—	—	—	—	—	—	—	—
	FBS_P	1941260	1895334	2.5	121	123	143641142	142860758	0.5	200	185
	FBS_R	1958266	1903105	2.9	0	14	145899873	144652625	0.9	0	1
	LINET	1956643	1901018	2.9	10	19	145251244	144108069	0.8	1	4
	RBS_P	1935456	1890361	2.4	300	246	143460049	142472835	0.7	249	258
	RBS_R	1955269	1901935	2.8	6	15	145664958	144614087	0.7	0	1

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

inating 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 difference in distribution between the heuristic results before and after the dominance rules is statistically significant. The improvement provided by the dominance rules is much higher for instances with low processing time and penalty variability. For these problems, even the best heuristics can benefit from a 1% to 2% decrease in objective function value.

The effect of the R and D parameters on the relative objective function value improvement is given in table 3 for the RBS_P heuristic. The relative improvement is usually non-decreasing with the due date slack range D, and the highest relative improvement values occur when D is equal to 0.50. The improvement provided by the dominance rules is usually lower when D is equal to 0.10 and the range of release dates R is set at 0.25 or 0.50.

		low var			high var		
n	R	D=0.10	D=0.25	D=0.50	D=0.10	D=0.25	D=0.50
100	0.25	0.3	0.7	0.8	0.1	0.1	0.2
	0.50	0.1	0.5	1.1	0.1	0.3	0.6
	0.75	0.5	0.5	1.4	0.4	0.3	1.0
300	0.25	0.6	1.7	3.2	0.1	0.3	0.5
	0.50	0.7	1.4	2.3	0.1	0.3	0.5
	0.75	1.1	1.2	1.8	0.4	0.4	0.8
500	0.25	0.8	2.1	3.9	0.2	0.4	0.8
	0.50	0.9	2.0	2.9	0.1	0.4	0.6
	0.75	1.4	1.2	1.3	0.5	0.5	1.0

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. The dominance rules require little additional computational effort, and their use is therefore recommended, since they allow for improvements in solution quality. The DBS and DTS heuristics are computationally demanding, and can therefore be used only for small or medium size instances. The filtered beam search algorithms, and particularly the PBS and recovering beam search procedures, are much faster and be applied even to large 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 require lower computation times when the variability is high. The algorithms with a rule-based filtering procedure are faster than their priority evaluation function counterparts. The recovering algorithms are also much faster than their filtered alternatives, and can solve even medium and large instances within reasonable computation times.

	low var				high var			
Heur	n=200	n=400	n=500	n=1000	n=200	n=400	n=500	n=1000
PBS	0.192	1.484	3.038	28.341	0.184	1.464	2.950	27.674
DBS	8.459	146.384	—	—	8.451	148.175	—	—
DTS	2.484	35.235	83.383	—	2.488	35.983	83.938	—
FBS_P	0.565	4.871	9.710	83.023	0.612	4.889	9.703	83.176
FBS_R	0.377	3.221	6.551	66.426	0.313	2.427	4.742	41.565
LINET	0.001	0.001	0.003	0.009	0.000	0.002	0.002	0.010
RBS_P	0.326	2.266	4.251	31.948	0.334	2.315	4.351	33.103
RBS_R	0.215	1.538	2.957	26.681	0.180	1.180	2.151	17.111
PBS + DR	0.195	1.503	3.072	28.596	0.185	1.470	2.962	27.758
DBS + DR	8.461	146.402	—	—	8.453	148.182	—	—
DTS + DR	2.487	35.248	83.405	—	2.489	35.987	83.945	—
FBS_P + DR	0.568	4.888	9.740	83.245	0.613	4.895	9.713	83.256
FBS_R + DR	0.381	3.240	6.584	66.662	0.314	2.433	4.754	41.650
LINET + DR	0.003	0.014	0.025	0.138	0.001	0.005	0.009	0.043
RBS_P + DR	0.329	2.285	4.285	32.189	0.335	2.322	4.364	33.199
RBS_R + DR	0.218	1.558	2.991	26.939	0.181	1.186	2.162	17.201

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 heuristics are usually somewhat closer to the optimum for problems with a low processing time and penalty variability. The average performance of the DBS heuristic is quite good, since it provides results that are 0.5% to 1.5% above the optimum and it generates an optimum solution for over 70% (40%) of the 20 (30) job test instances. The RBS_P procedure also performs quite well. This heuristic provides solutions that are about 1% to 2% above the optimum and it calculates an optimal schedule for roughly 60% (30%) of the 20 (30) job test instances.

Heur	low var				high var			
	n=20		n=30		n=20		n=30	
	%dev	#opt	%dev	#opt	%dev	#opt	%dev	#opt
PBS	6.3	37	7.5	5	6.5	35	9.0	6
DBS	1.0	292	2.3	121	1.1	285	2.4	118
DTS	2.3	182	3.6	54	2.3	168	3.7	52
FBS_P	1.2	266	2.3	111	1.2	271	2.6	114
FBS_R	5.6	146	9.4	48	5.7	155	9.2	36
LINET	7.5	31	9.5	4	8.2	28	10.6	5
RBS_P	1.1	254	2.0	136	1.0	269	2.3	129
RBS_R	5.1	170	8.4	68	4.9	190	8.4	59
PBS + DR	3.2	151	4.5	59	3.4	153	5.8	47
DBS + DR	0.5	343	1.5	192	0.8	327	1.6	188
DTS + DR	1.2	263	2.3	138	1.4	251	2.4	123
FBS_P + DR	0.8	311	1.8	169	0.9	313	2.1	172
FBS_R + DR	4.3	185	7.5	73	4.8	195	8.1	62
LINET + DR	4.2	136	6.4	49	5.0	136	7.3	41
RBS_P + DR	0.9	265	1.8	145	1.0	276	2.0	143
RBS_R + DR	4.5	180	7.2	72	4.8	192	8.1	63

Table 5: Comparison with optimum objective function values

In table 6 we present the effect of the R and D parameters on the relative deviation from the optimum for the RBS_P + DR heuristic. The relative deviation appears to increase with the due date slack range D, particularly when R is lower than 0.75. The heuristic performance is worst when D is equal to 0.50 and R is equal to 0.25 or 0.50. The heuristics are usually closer to the optimum when D is equal to 0.10 and the range of release dates R is set at 0.25 or 0.50. These results are similar to those reported for the relative improvement provided by the dominance rules, and seem to indicate that the problem is harder when the due date slack range is high and the release dates are not widely spread.

The DBS procedure provides very good results, but its computational requirements are only acceptable for small or medium size instances. The RBS_P heuristic is close to the DBS algorithm in solution quality and is significantly faster, solving even large instances within reasonable computation times. Therefore, this procedure is the heuristic of choice for medium and large size problems.

		low var			high var		
n	R	D=0.10	D=0.25	D=0.50	D=0.10	D=0.25	D=0.50
20	0.25	0.4	0.6	2.2	0.1	1.1	2.5
	0.50	0.5	0.9	1.9	0.9	1.0	1.7
	0.75	0.4	1.1	0.4	0.3	0.4	1.1
30	0.25	0.3	2.1	3.4	0.3	1.5	5.0
	0.50	0.7	1.6	4.4	0.5	2.2	4.2
	0.75	0.8	1.6	1.5	1.6	1.0	1.7

Table 6: Relative deviation from the optimum for the RBS_P + DR heuristic

5 Conclusion

In this paper we considered the single machine scheduling problem with earliness and tardiness penalties, different release dates and no unforced idle time. We considered heuristics based on the beam search technique and presented classical beam search algorithms, as well as the filtered and recovering variants. Both priority evaluation functions and problem-specific properties were considered for the filtering step used in the filtered and recovering beam search heuristics. The algorithms 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 use of the dominance rules is recommended, since they can improve the solution quality, particularly for instances with a low processing time and penalty variability, and require little additional computational effort. 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 DBS 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 quite 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 achieve a good balance between solution quality and computational efficiency. These results confirm the potential of this recently introduced 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] 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.

[3] Fox, M. S. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling.* Ph.d. thesis, Carnegie-Mellon University, USA, 1983.

[4] Kanet, J. J., and Zhou, Z. A decision theory approach to priority dispatching for job shop scheduling. *Production and Operations Management 2* (1993), 2–14.

[5] Korman, K. A pressing matter. *Video* (February 1994), 46–50.

[6] Landis, K. Group technology and cellular manufacturing in the westvaco los angeles vh department. Project report in iom 581, School of Business, University of Southern California, 1993.

[7] Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1* (1977), 343–362.

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

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

[10] Lowerre, B. T. *The HARPY Speech Recognition System.* Ph.d. thesis, Carnegie-Mellon University, USA, April 1976.

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

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

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

[14] Rubin, S. *The ARGOS Image Understanding System.* Ph.d. thesis, Carnegie-Mellon University, USA, April 1978.

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

[16] Valente, J. M. S., and Alves, R. A. F. S. An exact approach to early/tardy scheduling with release dates. Working Paper 128, Faculdade de Economia do Porto, Portugal, 2003.

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

[18] 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.

Working papers mais recentes

Nº 142	Jorge M. S. Valente and Rui A. F. S. Alves, *Filtered and Recovering beam search algorithms for the early/tardy scheduling problem with no idle time*, April 2004
Nº 141	João A. Ribeiro and Robert W. Scapens, *Power, ERP systems and resistance to management accounting: a case study*, April 2004
Nº 140	Rosa Forte, *The relationship between foreign direct investment and international trade. Substitution or complementarity? A survey*, March 2004
Nº 139	Sandra Silva, *On evolutionary technological change and economic growth: Lakatos as a starting point for appraisal*, March 2004
Nº 138	Maria Manuel Pinho, *Political models of budget deficits: a literature review*, March 2004
Nº 137	Natércia Fortuna, *Local rank tests in a multivariate nonparametric relationship*, February 2004
Nº 136	Argentino Pessoa, *Ideas driven growth: the OECD evidence*, December 2003
Nº 135	Pedro Lains, *Portugal's Growth Paradox, 1870-1950*, December 2003
Nº 134	Pedro Mazeda Gil, *A Model of Firm Behaviour with Equity Constraints and Bankruptcy Costs*, November 2003
Nº 133	Douglas Woodward, Octávio Figueiredo and Paulo Guimarães, *Beyond the Silicon Valley: University R&D and High-Technology Location*, November 2003.
Nº 132	Pedro Cosme da Costa Vieira, *The Impact of Monetary Shocks on Product and Wages: A neoclassical aggregated dynamic model*, July 2003.
Nº 131	Aurora Teixeira and Natércia Fortuna, *Human Capital, Innovation Capability and Economic Growth*, July 2003.
Nº 130	Jorge M. S. Valente and Rui A. F. S. Alves, *Heuristics for the Early/Tardy Scheduling Problem with Release Dates*, May 2003.
Nº 129	Jorge M. S. Valente and Rui A. F. S. Alves, *An Exact Approach to Early/Tardy Scheduling with Release Dates*, May 2003.
Nº 128	Álvaro Almeida, *40 Years of Monetary Targets and Financial Crises in 20 OECD Countries*, April 2003.
Nº 127	Jorge M. S. Valente, *Using Instance Statistics to Determine the Lookahead Parameter Value in the ATC Dispatch Rule: Making a good heuristic better*, April 2003.
Nº 126	Jorge M. S. Valente and Rui A. F. S. Alves, *Improved Heuristics for the Early/Tardy Scheduling Problem with No Idle Time*, April 2003.
Nº 125	Jorge M. S. Valente and Rui A. F. S. Alves, *Improved Lower Bounds for the Early/Tardy Scheduling Problem with No Idle Time*, April 2003.
Nº 124	Aurora Teixeira, *Does Inertia Pay Off? Empirical assessment of an evolutionary-ecological model of human capital decisions at firm level*, March 2003.
Nº 123	Alvaro Aguiar and Manuel M. F. Martins, *Macroeconomic Volatility Trade-off and Monetary Policy Regime in the Euro Area*, March 2003.
Nº 122	Alvaro Aguiar and Manuel M. F. Martins, *Trend, cycle, and non-linear trade-off in the Euro Area 1970-2001*, March 2003.
Nº 121	Aurora Teixeira, *On the Link between Human Capital and Firm Performance. A Theoretical and Empirical Survey*, November 2002.
Nº 120	Ana Paula Serra, *The Cross-Sectional Determinants of Returns: Evidence from Emerging Markets' Stocks*, October 2002.
Nº 119	Cristina Barbot, *Does Airport Regulation Benefit Consumers?*, June

	2002.
Nº 118	José Escaleira, A Procura no Sector das Artes do Espectáculo. Tempo e Rendimento na Análise das Audiências. Um Estudo para Portugal, June 2002.
Nº 117	Ana Paula Serra, Event Study Tests: A brief survey, May 2002.
Nº 116	Luís Delfim Santos and Isabel Martins, A Qualidade de Vida Urbana - O caso da cidade do Porto, May 2002.
Nº 115	Marcelo Cabús Klötzle and Fábio Luiz Biagini, A Restruturação do Sector Eléctrico Brasileiro: Uma análise comparativa com a Califórnia, January 2002.
Nº 114	António Brandão and Sofia B. S. D. Castro, Objectives of Public Firms and Entry, December 2001.
Nº 113	Ana Cristina Fernandes and Carlos Machado-Santos, Avaliação de Estratégias de Investimento com Opções, December 2001.
Nº 112	Carlos Alves and Victor Mendes, Corporate Governance Policy and Company Performance: The Case of Portugal, December 2001.
Nº 111	Cristina Barbot, Industrial Determinants of Entry and Survival: The case of Ave, October 2001.
Nº 110	José Rodrigues de Jesús, Luís Miranda da Rocha e Rui Couto Viana, Avaliação de Pequenas e Médias Empresas e Gestão de Risco, October 2001.
Nº 109	Margarida de Mello and Kevin S. Nell, 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, July 2001.
Nº 108	Cristina Barbot, Horizontal Merger and Vertical Differentiation, June 2001.
Nº 107	Celsa Machado, Measuring Business Cycles: The Real Business Cycle Approach and Related Controversies, May 2001.
Nº 106	Óscar Afonso, The Impact of International Trade on Economic Growth, May 2001.
Nº 105	Abraão Luís Silva, Chamberlain on Product Differentiation, Market Structure and Competition: An essay, May 2001.
Nº 104	Helena Marques, The "New" Economic Theories, May 2001.
Nº 103	Sofia B. S. D. Castro and António Brandão, Public Firms in a Dynamic Third Market Model, January 2001.
Nº 102	Bernard Friot, Bernadette Clasquin & Nathalie Moncel, Salaire, Fiscalité et Épargne dans le Finacement de l 'Emploi et de la Protection Sociale: l'Example Européen, January 2001.
Nº 101	Paulo Beleza Vasconcelos, Resolução Numérica de Modelos Macroeconómicos com Expectativas Racionais, 2000.
Nº 100	Luis David Marques, Modelos Dinâmicos com Dados em Painel: Revisão da Literatura, 2000.
Nº 99	Rui Henrique Alves, Da Moeda Única à União Política?, 2000.
Nº 98	Paulo Guimarães, Octávio Figueiredo & Doug Woodward, *A Tractable Approach to the Firm Location Decision Problem*, 2000.
Nº 97	António Brandão & José Escaleira, *Trade Policy and Tacit Collusion with Price and Quantity Competition*, 2000.
Nº 96	Sandra Silva & Mário Rui Silva, *Crescimento Económico nas Regiões Europeias: Uma Avaliação sobre a Persistência das Disparidades Regionais no Período 1980-95*, 2000.

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

Download dos artigos em:

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