Int. J. Production Economics 132 (2011) 107–121

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Heuristics for operational fixed job scheduling problems with working and spread time constraints

Deniz Türsel Eliiyi [a,*], Meral Azizoğlu [b]

[a] Izmir University of Economics, Department of Industrial Systems Engineering, Sakarya Cad., No: 156, Izmir 35330, Turkey
[b] Middle East Technical University, Department of Industrial Engineering, ODTU 06531, Ankara, Turkey

ARTICLE INFO

Article history:
Received 12 October 2009
Accepted 15 March 2011
Available online 30 March 2011

Keywords:
Fixed job scheduling
Working time constraints
Spread time constraints
Heuristics

ABSTRACT

Operational fixed job scheduling problems select a set of jobs having fixed ready and processing times and schedule the selected jobs on parallel machines so as to maximize the total weight. In this study, we consider working time and spread time constrained versions of the operational fixed job scheduling problems. The working time constraints limit the total processing load on each machine. The spread time constraints limit the time between the start of the first job and the finish of the last job on each machine. For the working time constrained problem, we present a filtered beam search algorithm that evaluates the promising nodes of the branch and bound tree. For the spread time constrained problem we propose a two phase algorithm that defines the promising sets for the first jobs and finds a solution for each promising set. The results of our computational tests reveal that our heuristic algorithms perform very well in terms of both solution quality and time.

1. Introduction

Fixed job scheduling (FJS) is an applied area of scheduling where tasks (jobs) with specified weights, ready times and deadlines are to be processed on parallel resources (machines). A task can be processed only between its ready time and deadline, and its processing time is exactly equal to the difference between these.

The FJS problems are of two types: operational and tactical. The tactical fixed job scheduling (TFJS) problems aim to minimize the number or cost of the resources while processing all tasks. The operational fixed job scheduling (OFJS) problems aim is to select a subset of tasks for processing so that the total weight is maximized.

The FJS problems have many application areas in manufacturing and service operations as mentioned in many reported work. Kovalyov et al. (2007) and Kolen et al. (2007) survey the potential applications of the FJS problems together with their theory. Bekki and Azizoglu (2008) study the OFJS problem on uniform parallel machines. Kroon (1990) and Kroon et al. (1995) mention the applications in assigning aircrafts to gates, and the capacity planning of maintenance personnel. Fischetti et al. (1987, 1992) and Martello and Toth (1986) consider an application of the TFJS problem in scheduling the bus drivers whereas Wolfe and Sorensen (2000) consider an application of the OFJS problem in scheduling the earth-observing satellites. Other cited application areas include class scheduling (Kolen and Kroon, 1991), satellite data transmission (Faigle et al., 1999) and printed circuit board manufacturing (Spieksma, 1999).

The machine operating time constraints in FJS are introduced by Fischetti et al. (1987, 1989). These constraints are of two types: working time and spread time. Working time constraints impose a limit on the processing load of each machine, letting no machine operate for more than T time units. Spread time of a machine is defined as the time between the start of its first operation and the finish of its last operation, including the idle times in between. The spread time constraints limit this time span with an upper bound, S.

The operating time constraints find their applications in both manufacturing and service operations. In manufacturing environments, it may not be economical to operate some high-precision/high-cost equipment for more than a prespecified time (working time) and/or the machines may be available for a particular specified time period (spread time). In service environments like car rental systems, hotel room reservations and personnel scheduling, the resources may have operation time constraints imposed by the technical or union regulations.

Fischetti et al. show that the TFJS problems with working time and spread time constraints are strongly NP-hard, in their 1987 and 1989 studies, respectively. They develop lower bounds and branch and bound (B&B) algorithms and report on their

* Corresponding author. Tel.: +90 232 4888357; fax: +90 232 4888475.
E-mail addresses: deniz.eliiyi@ieu.edu.tr (D.T. Eliiyi), meral@ie.metu.edu.tr (M. Azizoğlu).

0925-5273/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2011.03.018

satisfactory average-case performances based on the data generated close to the real-life situations. In their study, Fischetti et al. (1992) provide several polynomial time heuristic algorithms for both problems and report on their satisfactory worst-case performances.

The research on the OFJS problem with operating time constraints is very limited despite its practical importance. Bouzina and Emmons (1996) study the OFJS problem under working time constraints. The authors prove that the preemptive problem is polynomially solvable when all job weights are equal; however it is NP-hard in the ordinary sense when the job weights are arbitrary. The OFJS problems with spread time and working time constraints are studied by Eliiyi and Azizoglu in their 2006 and 2010 studies, respectively. They show that both problems are NP-hard in the strong sense and report their several polynomially solvable special cases. They propose B&B algorithms with several problem size reduction mechanisms and dominance conditions. For both problems, the algorithms return optimal solutions for problem instances with up to 100 jobs and 4 machines. Solyali and Ozpeynirci (2009) study the OFJS problem with spread time constraints, and develop a branch and price algorithm that outweighs the B&B algorithm proposed by Eliiyi and Azizoglu (2006). Very recently Rossi et al. (2010) develop two heuristic algorithms, namely greedy heuristic and genetic algorithm and report on the satisfactory behavior of their algorithms.

In this study, we consider the OFJS problem with working time and spread time constraints. For each problem, we develop several heuristic procedures that use efficient upper bounds and dominance conditions. Our aim is to find powerful lower bounds in reasonable times. There are two heuristic procedures that consider the spread time constraints (Rossi et al., 2010). We compare the performance of our heuristic algorithm with those procedures. Our computational study reveals that our algorithm outweighs greedy heuristic over all problem instances; however the genetic algorithm slightly outweighs our algorithm. To the best of our knowledge there is no reported heuristic study for the OFJS problem with working time constraints. As both problems have many practical applications and are to be solved quite frequently, our study fills an important gap in the literature and can be used for large problem size instances conveniently. The rest of the paper is organized as follows. We present the mathematical formulations of the problems in Section 2. We present our heuristic algorithms in Sections 3 and 4. Section 5 presents the results of our computational study designed to evaluate the performances of the algorithms. The conclusions are discussed in Section 6.

2. Problem formulations

There are n jobs to be scheduled for processing on m identical parallel machines. Job j arrives at its ready time r_j, and completes at its deadline d_j, if selected for processing. The processing time of job j is $p_j=d_j-r_j$ and its weight is w_j. Without loss of generality, we assume that all data are integer and the jobs are indexed in non-decreasing order of their ready times.

In the OFJS problem with working time constraints (OFJSW), each machine processes a maximum of T time units. We assume that the total processing time of all jobs, i.e., $\sum_{j=1}^{n} p_j$, is greater than T, so that the working time constraint is not redundant. For the OFJS problem with spread time constraints (OFJSS), recall that the spread time for machine k is defined as $(d_{k(l)}-r_{k(1)})$ where $k(1)$ is the first job and $k(l)$ is the last job processed by machine k. The spread time for machine $k(d_{k(l)}-r_{k(1)})$ cannot exceed its limit S in any feasible solution of the OFJSS problem.

The time is assumed to be divided into intervals not necessarily equal in length. We let $\{t_1, t_2, \ldots, t_z\}$ be the sequence of the r_j and d_j values in chronological order with duplicates removed, and P_a be the set of available jobs in interval a, i.e., the set of jobs that resides interval $[t_a,t_{a+1})$ between its ready time and deadline, $a=1, 2, \ldots, z-1$.

The binary assignment variable is defined as follows:

$$x_{jk}=\begin{cases}1, & \text{if job } j \text{ is processed on machine } k\\ 0, & \text{otherwise}\end{cases} \quad \forall j,k.$$

The following integer programming model formulated by Bouzina and Emmons (1996) represents the OFJSW problem:

$$\text{Maximize} \sum_{k=1}^{m}\sum_{j=1}^{n} w_j x_{jk} \tag{1}$$

subject to

$$\sum_{k=1}^{m} x_{jk} \le 1 \quad j=1,\ldots,n \tag{2}$$

$$\sum\nolimits_{j\in P_a} x_{jk} \le 1 \quad k=1,\ldots,m \quad a=1,2,\ldots,z-1 \tag{3}$$

$$\sum_{j=1}^{n} p_j x_{jk} \le T \quad k=1,\ldots,m \tag{4}$$

$$x_{jk}\in\{0,1\} \quad k=1,\ldots,m \quad j=1,\ldots,n \tag{5}$$

The objective function (1) maximizes the total weight of the processed jobs. Constraint set (2) ensures that each job is processed by at most one machine. The condition that no machine can process more than one job at a time is handled by constraint set (3). Working time constraints (T-constraints) are stated in constraint set (4). Constraint set (5) forces the integrality in job–machine assignments.

To impose the spread time limit, we define set I_j as the incompatibility set for job j, as $I_j=\{i \mid i>j;\ r_i<d_j \text{ or } d_i-r_j>S\}$. Then, the assignment of job j and any job $i\in I_j$ to the same machine violates the spread time constraint. The following integer programming model formulated by Eliiyi and Azizoglu (2006) represents the OFJSS problem:

$$\text{Maximize} \sum_{k=1}^{m}\sum_{j=1}^{n} w_j x_{jk} \tag{6}$$

subject to

$$\sum_{k=1}^{m} x_{jk} \le 1 \quad j=1,\ldots,n \tag{7}$$

$$\sum_{j\in P_a} x_{jk} \le 1 \quad k=1,\ldots,m \quad a=1,2,\ldots,z-1 \tag{8}$$

$$x_{ik}+x_{jk} \le 1 \quad k=1,\ldots,m, \quad j=1,\ldots,n-1, \ i\in I_j \tag{9}$$

$$x_{jk}\in\{0,1\} \quad k=1,\ldots,m \quad j=1,\ldots,n \tag{10}$$

The objective function (6) maximizes the total weight of the processed jobs. Constraint sets (7) and (8) ensure that each job is processed on at most one machine and no machine can process more than one job at a time, respectively. Constraint set (9) imposes the spread time constraints. Constraint set (10) forces the integrality in job–machine assignments.

3. A filtered beam search for the OFJSW problem

Beam search is a heuristic B&B algorithm, which evaluates nodes at each level, keeping the most promising β nodes while discarding the rest. β is called the beam width. As some nodes are

permanently discarded, the solution returned by the beam search heuristic may not be optimal, but smaller solution times and memory requirements are obtained when compared to B&B. The promising β nodes are determined by the beam evaluation function.

The β value and the beam evaluation function influence the performance of the beam search algorithm. A careful evaluation with many nodes at each level is time consuming while a crude prediction is quick but may discard good solutions. In an attempt to resolve this trade-off, hence increase the efficiency of a beam search algorithm, the filtered beam search is developed by Ow and Morton (1988).

In filtered beam search, at any level a quick evaluation is first made for all nodes and the most promising α nodes are selected for further branching. α is called the filter width. A thorough evaluation is then made on selected α nodes, for reducing the number of promising nodes to the beam width, β. The filtered beam search can use a powerful evaluation function in evaluating the filtered α nodes and this may lead to better solutions while keeping the solution times at a reasonable level due to the filtering process. The settings of α and β values are important and based on initial experimentation.

Many successful applications of the beam search technique are reported, some noteworthy of which are due to Morton and Pentico (1993), Della Croce et al. (2004), Ghirardi and Potts (2005) and Valente and Alves (2006). Morton and Pentico (1993) discuss the details and several variations of the beam search technique.

We propose a filtered beam search algorithm to find an approximate solution for the OFJSW problem. We now state the components of our filtered beam search algorithm.

3.1. Filtered beam search tree

We use the branching scheme proposed in Eliiyi and Azizoglu (2010). A node at level l of the search tree corresponds to a partial solution in which the decisions about the first l jobs are made. For each node emanating from level l, there are at most $(m+1)$ decisions: one for rejecting job $(l+1)$ and one for processing job $(l+1)$ on machine k, $k=1, \ldots, m$. Assigning to machine k is feasible if the machine does not process any overlapping job and the T-constraint is not violated. When there is more than one machine with no job assignments, only one of them is considered in order to eliminate the duplication of the partial solutions. Fig. 1 illustrates our branching tree for three jobs and three machines.

3.2. Initial feasible solution

To find an initial feasible solution we form six feasible solutions, i.e., lower bounds (LB), using simple greedy job sequencing and machine assignment rules. We order the jobs by maximum weight per processing time (w_j/p_j), maximum weight (w_j) and the shortest processing time (p_j) rules. We assign the first job of the order using two rules: the least-loaded machine and the most-loaded machine, while satisfying T-constraints. Three sequencing rules together with two assignment rules give a total of six lower bounds, some of which may be identical.

Each of these lower bounds is improved by two polynomial-time improvement algorithms-exchange and insert and replacement-in a sequential manner. The exchange and insert type heuristic tries to exchange two scheduled jobs between machines. The algorithm performs an exchange only if any unscheduled job can be processed after the exchange. The algorithm performs the move that yields the maximum total weight, and proceeds until no moves are possible or after n iterations. The replacement heuristic exchanges a scheduled job with one or more unscheduled jobs. The algorithm considers all possible exchanges and performs the most-improving one. It stops when no further improvement is possible or after n iterations.

The maximum lower bound value after the improvements is used as the best known lower bound, i.e., the incumbent solution. The incumbent solution is updated whenever a feasible solution with a higher total weight value is reached.

3.3. Size reduction properties

Eliiyi and Azizoglu (2010) present some properties that characterize the structure of the optimal solution and use them to reduce the tree size. We also use the results of these properties to fathom the nodes of our search tree. For the sake of completeness, we state these properties and discuss their implementation in our search process.

Property 1. If $w_j > w_i$, $r_i \le r_j$ and $d_i \ge d_j$, then any optimal solution that includes job i should also include job j.

In our search tree, the node representing the rejection of job i is fathomed if there exists a scheduled job j such that $w_i > w_j$, $r_j \le r_i$ and $d_j \ge d_i$.

Property 2. If $w_j = w_i$, $r_i \le r_j$ and $d_i \ge d_j$ with strict inequality holding at least once, then there exists an optimal schedule that includes job j if job i is processed.

In our search tree, the node that rejects job i is fathomed if there exists a scheduled job j such that $w_i = w_j$, $r_j \le r_i$ and $d_j \ge d_i$, with strict inequality holding at least once.

Property 3. Let S_i be the set of jobs that dominate job i. If $|S_i| \ge m$, then no optimal solution processes job i.

In our search tree, we remove job i from the partial solution if $|S_i| \ge m$.

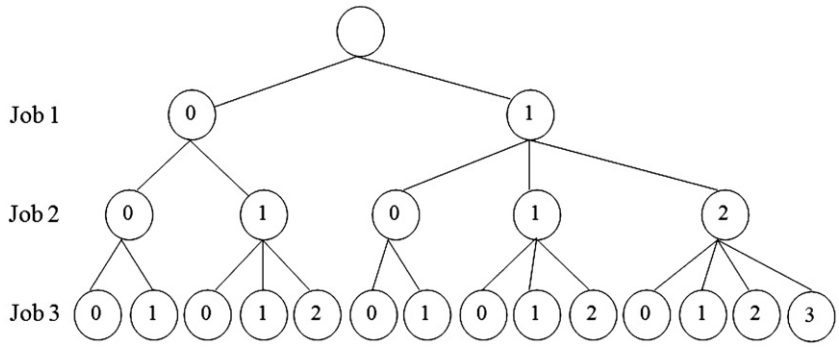

Fig. 1. The filtered beam search tree.

Property 4. If $r_i < r_j < d_i < d_j$, $w_i > w_j$, $p_i \le p_j$, and no job is available for processing during $[r_i, r_j]$, then any solution that includes job j, but not job i, cannot be optimal.

All nodes that represent the assignment of job j on any machine are fathomed if there exists an unscheduled job i (such that $p_i \le p_j$, $r_i < r_j \le d_i < d_j$, and $w_i > w_j$), and no job k that satisfies $r_i \le r_k \le d_k \le r_j$ is scheduled.

We evaluate each node that is not fathomed by the above properties, using the filter evaluation function.

3.4. Filter evaluation function

We use lower bounds as filter evaluation functions. In doing so, we form the w_j/p_j list and assign its first job to the least and most-loaded machine. The maximum of the two lower bound values is taken for evaluation. α Nodes having the highest lower bounds are selected and the rest of the nodes in the level are permanently discarded.

3.5. Beam evaluation function

We use upper bounds as beam evaluation functions. As in Eliiyi and Azizoglu (2010), we use the surrogate relaxation of the problem to obtain an upper bound. In the surrogate relaxation, constraint set (4) representing the T-constraints is replaced by the following inequality:

$$\sum_{k=1}^{m}\sum_{j=1}^{n} p_j x_{jk} \le mT \qquad (4')$$

In the absence of the overlapping constraints, this representation reduces to a single knapsack problem with capacity mT. The single knapsack problem is NP-hard in the ordinary sense (Garey and Johnson, 1979). Eliiyi and Azizoglu (2010) show that the surrogate relaxation of the problem can be solved in polynomial time when partial processing and preemption are allowed. We call this upper bound UBW_1.

When the working time constraints are ignored, the problem reduces to the classical OFJS problem, which can be solved in $O(n^3)$ time using Minimum Cost Network Flow model (see Bouzina and Emmons, 1996). Although this solution can be used as an upper bound, it is shown to be computationally inefficient by Eliiyi and Azizoglu (2010) in their B&B implementation. However, they show that the longest-path upper bound, UBW_2, is efficient when there is only a single available machine in the partial solution. This solution can be obtained in $O(n)$ time (Solyali and Ozpeynirci, 2009). If only one machine remains available, we use Min{UBW_1, UBW_2}, otherwise we use UBW_1.

The node is fathomed if its upper bound is not greater than the incumbent solution. If any upper bound leads to a feasible solution, we fathom the node and update the incumbent solution if the upper bound is greater. Among the remaining nodes, we select β nodes having largest upper bounds for further branching. We stop whenever we reach the lowest level of the search tree. The incumbent solution returned by the filtered beam search is improved using the exchange and insert and replacement improvement algorithms in a sequential manner. The pseudo-code of the procedure is provided in Fig. 2.

Our algorithm considers at most $\alpha\beta n$ nodes as $\alpha\beta$ nodes at each level are evaluated and there are n levels. The algorithm runs

```
INITIAL SOLUTION:
Compute six lower bounds, as explained in Section 3.1;
Improve each lower bound solution, store solutions in LBW1, …, LBW6;
Incumbent = Max{LBW1, …, LBW6}.
FILTERED BEAM SEARCH:
For each level (1,..., n-2) of the tree
        Clear α-list ;
        Do for each node in β-list
                Branch from node (create at most m+1 branches, using dominance conditions to decide whether to fathom);
                For each non-fathomed child node
                        Compute LBW as best of six lower bounds, store solution if better than Incumbent;
                        Store node in the α-list if LBW > minimum LBW in α-list (keep at most α nodes);
                End For
        End Do
        Clear β-list;
        Do for each node in the α-list
                Compute UBW as UBW1, or lower of UBW1 and UBW2 if only one available machine;
                Update Incumbent if solution if feasible and better than Incumbent;
                Store node in the β-list if UBW > minimum UBW in β-list (keep at most β nodes);
        End Do
End For
For all nodes in the β-list (nodes of level n-1)
        Branch from the node;
        Compute the objective function value;
        Update Incumbent if solution is better than Incumbent;
End For
IMPROVEMENT:
Update Incumbent by applying the two improvement algorithms
Return Incumbent.
```

Fig. 2. The pseudo-code of the filtered beam search algorithm for the OFJSW problem.

in polynomial time, as at each node polynomial time operations are performed and polynomial number of nodes is evaluated. The improvement procedures run also in polynomial time as the number of iterations is limited. Hence the whole algorithm runs in polynomial time.

4. Two-phase heuristic algorithm for the OFJSS problem

To find a lower bound to the OFJSS problem, we develop a two-phase heuristic algorithm. In the first phase, we enumerate first jobs for each machine and select the most promising combinations for further evaluation. As the first jobs determine the spread time limits, they have a critical importance in shaping the solution. After selecting the promising first job combinations, the second phase defines a time-phased algorithm for each selected combination. We provide the details in the following subsections.

4.1. Phase 1: A procedure to enumerate first jobs

We find the promising first jobs using the branching algorithm by Eliiyi and Azizoglu (2006). According to this algorithm, a node at level l of the search tree corresponds to a partial solution where the first jobs on the first l machines are set. We always branch to a node having a higher index to avoid the duplication of the solutions. At the lowest level of the tree, i.e., at level m, there are at most $(n!/(m!(n-m)!))$ nodes. Some of these nodes are eliminated using the following properties by Eliiyi and Azizoglu (2006).

Property 5. Let $r_i \le r_j < d_i < d_j$, and $B=\{l | d_i \le r_l < d_j\}$. Then, any optimal solution that includes job i as the first job on any machine should also include job j, if $\sum_{l \in B}(w_l+w_i) < w_j$.

Property 6. Let $r_j < r_i$, and $C=\{l | r_j+S < d_l \le r_i+S\}$. Then, the following statements are true:

i. If $r_i < d_j$, any optimal solution that includes job i as the first job on any machine should also include job j, if $\sum_{l \in C}(w_l+w_i) < w_j$.
ii. If $r_i \ge d_j$, any optimal solution that includes job i as the first job on any machine should also include job j, if $\sum_{l \in C} w_l < w_j$.

The following property follows Properties 5 and 6. Let,

$$R_i = \{j : r_i \le r_j < d_i < d_j, \text{ and } \sum_{l \in B} w_l + w_i \le w_j\},$$
where $B = \{l | d_i \le r_l < d_j\}$,

$$Q_i = \{j : r_j < r_i, \text{ and } \sum_{l \in C} w_l + w_i Y \le w_j\},$$
where $C = \{l | r_j + S < d_l \le r_i + S\}$,

where

$$Y = \begin{cases} 1 & \text{if jobs } i \text{ and } j \text{ overlap} \\ 0 & \text{otherwise} \end{cases}.$$

Property 7. If $|R_i \cup Q_i| \ge m$, then job i cannot be a first job in any optimal solution.

At level m, for each node that cannot be eliminated by the above properties, we compute two upper bounds: UBS_1 and UBS_2. We use $UBS = Min\{UBS_1, UBS_2\}$ to evaluate the node. UBS_1 is found for a known set of first jobs, by allowing preemption and partial processing (Eliiyi and Azizoglu, 2006). The algorithm proceeds in chronological sequence $\{t_1, t_2, \ldots, t_z\}$of ready times and deadlines, optimizes the assignments in each interval until all intervals are considered or no machine remains available. A solution for a specified interval $[t_a, t_{a+1})$, $a=1, 2, \ldots, z-1$, is found by assigning $O(m)$ jobs having highest weight per unit time, each to one of the available machines. UBS_2 is found by solving m single machine problems separately, hence allowing the assignment of one job to more than one machine. Each single machine problem is solved by the longest path algorithm.

We update the incumbent solution if UBS at a node gives a feasible solution with higher total weight value. We fathom the node if its UBS is no better than the incumbent solution. We store best γ nodes having the highest UBS values for further evaluations. The parameter γ is set by initial experimentation.

For finding an initial incumbent solution we use two procedures. Our first procedure utilizes the MCNF solution, which is optimal in the absence of the spread time constraints (Bouzina and Emmons, 1996). If the MCNF solution violates the spread time constraints on some machine, we convert this infeasible solution to a feasible one for that machine. In doing so, in the MCNF solution for each job j on violated machine k, we subtract the weights of the jobs that violate the spread time constraint assuming job j is the first job on that machine. We select the maximum of such solutions as the solution on machine k. We let the overall lower bound, LBS_1, be the sum of the weights on all machines.

Our second procedure uses a decomposition idea and solves nm longest path problems in a sequential manner. For each machine, once its first job is found, the set of jobs that can be scheduled without violating the spread time constraint is determined automatically. The longest path problem is then solved on the machine using this set in $O(n)$ time (Solyali and Ozpeynirci, 2009). Each unscheduled job is tried for the first job, and the best longest path solution is fixed as the schedule of the machine. The machine and the jobs on its longest path are eliminated from further consideration, and the procedure is repeated for the remaining machines. We let LBS_2 be the sum of the weights on all machines.

The improvement procedures (defined in Section 3.2) are also employed on the LBS_1 and LBS_2 solutions and the better of the two improved solutions is used as an initial incumbent solution.

4.2. Phase 2: A time-phased MCNF algorithm

In this section we present the details of our time-phased MCNF-based algorithm. The time-phased algorithm is run for each of the selected γ nodes (first job combinations) of Phase 1.

We assume the machines are indexed in chronological order of the ready times of their first jobs, i.e., $r_{1(1)} \le r_{2(1)} \le \ldots \le r_{k(1)} \le \ldots \le r_{m(1)}$, where $k(1)$ represents the index of the first job on machine k, $k=1, \ldots, m$. For each machine k, $k=1, \ldots, m$, start and end times are set to $s_k = r_{k(1)}$ and $e_k = r_{k(1)} + S$, respectively.

We create m intervals, one for each end time, as $[s_1, e_1)$, $[e_1, e_2)$, …, $[e_{m-1}, e_m)$. Our time-phased algorithm solves m MCNF problems, each in one interval, with the active machines, i.e., the machines that are available for processing jobs in that interval. The sum of the partial solutions' total weight values over all intervals provides a lower bound. The step-wise description of the algorithm is given below:

4.2.1. Algorithm phased MCNF

S0. Given a first job combination, index the machines in non-decreasing order of the ready times of their first jobs.
Determine start and end times and create m intervals as described above.
Let $B=\{j | j=1, \ldots, n\}$ be the set of unscheduled jobs.

S1. For each interval I_k, $k=1, \ldots, m$ corresponding to $[s_1,e_1), [e_1,e_2), \ldots, [e_{m-1},e_m)$:

Determine the set of machines that are active in the current interval, I_k, as

$$M_k = \{v \mid e_{k-1} \le s_v \le e_k, e_v \ge e_{k-1}, v = 1,\ldots,m\}.$$

Check if the first jobs exist on all active machines. If yes, F_k: set of first jobs, else $F_k=\emptyset$.

Determine the set of jobs that are available in the current interval, as

$$A_k = \{i \in B \wedge i \notin F_k \mid e_{k-1} \le r_i \le e_k\}.$$

If $F_k \neq \emptyset$, solve MCNF(M_k,F_k,A_k), else solveMCNF(M_k,A_k).

In the resulting solution, if there is an unfinished job on machine k (current machine), i.e., if a job j exists such that $d_j > e_k$:

Remove job from machine k.

Fix the schedule on machine k, eliminate machine k and fixed jobs from further consideration (delete jobs from set B).

If there is at least one unfinished job in any of the other machines:

For machines with unfinished jobs:

Update the partial schedule on those machines including the unfinished jobs.

Set the unfinished job as the first job on that machine.

Delete all jobs before the first job from set B.

For machines with no unfinished jobs:

Update the partial schedule on those machines.

Set the last finished job as the first job on that machine.

Delete all jobs before the first job from set B.

Let *early* be the earliest-starting first job among machines.

Update $s_{k+1}=r_{early}$.

Else (If no unfinished jobs on other machines):

Update the partial schedules on each machine.

Delete scheduled jobs from set B.

Update $F_k=\emptyset$, and $s_{k+1}=e_k$

End For.

S2. The solution is the sum of the weights of jobs scheduled over all machines.

S3. Improve the solution by the improvement algorithms.

Algorithm phased MCNF utilizes the following two procedures to find an optimal solution of an interval. Procedure MCNF(M_k,F_k,A_k) solves a MCNF problem to find the optimal solution for an interval when the first jobs are known (Eliiyi and Azizoglu, 2006). The solution is optimal, since the constructed network gives the optimal solution in the absence of the spread time constraints. We create such a problem for each interval by determining the active machine and job sets. The description of procedure MCNF(M_k,F_k,A_k) is given below.

4.2.2. Procedure MCNF(M_K,F_k,A_K)

Construct the following network: create a source node s, nodes $1, 2, \ldots, |A_k \cup F_k|$ for each job, and a dummy node $t=|A_k \cup F_k|+1$. Connect node s to each node in F_k with arc cost zero and capacity 1. Connect node j to $j+1$ with arc cost zero and capacity m, $j=1, \ldots, |A_k \cup F_k|$, for all $j \notin F_k$. Create arc(j,k), where k is the first job not overlapping with job j and $k \notin F_k$, for $j=1, \ldots, |A_k \cup F_k|$. If no such job exists, create arc(j,t). Each of these arcs has cos $t-w_j$ and capacity 1.

Require a flow of m units from s to t. Solve the resulting MCNF problem.

Procedure MCNF(M_k,A_k) below solves a MCNF problem to find the optimal solution for an interval when the first jobs are not known, as shown by Bouzina and Emmons (1996).

4.2.3. Procedure MCNF(M_K,A_K)

Construct the following network: create nodes $s=1, \ldots, n$ for each job, and a dummy node, $t=n+1$. Connect node j to node $j+1$ with arc cost zero and capacity m, $j=1, \ldots, n$. Create arc(j,k), where k is the first job not overlapping with job j, $j=1, \ldots, n$. If no such job exists, create arc(j,t). Each of these arcs has cos $t-w_j$ and capacity 1.

Require a flow of m from s to t. Solve the resulting MCNF problem.

The incumbent solution returned by the phased MCNF algorithm is improved by an insertion type improvement procedure that tries to insert the most-improving unscheduled job into the schedule. With this algorithm, the boundary jobs, i.e., the unfinished jobs which may have been eliminated are re-tried to be scheduled. The two improvement algorithms presented in Section 3.2 are then executed on the solution. The pseudo-code of the whole process is given in Fig. 3.

In the first phase, the algorithm evaluates $(n!/(m!(n-m)!))$ solutions. Note that $(n!/(m!(n-m)!))$ has an order of n^m, which is polynomial for fixed m and exponential for arbitrary m. Each solution is evaluated in polynomial time. The second phase also runs in polynomial time as MCNF problem is solved γ times each in $O(n^3)$ time. The improvement procedures also run in polynomial time, as stated before. Hence the whole algorithm runs in polynomial time when m is fixed and in exponential time when m is arbitrary.

5. Computational experiments

All algorithms have been implemented in MS Visual C++ 6.0, and experimentally evaluated on a 2.8 GHz Core2Duo computer with 4 GB memory. We consider two classes of random test problems adapted from Fischetti et al. (1992):

1. $r=1$: the ready times are uniform in [0,200].
2. $r=2$: 30% of the ready times are uniform in [30,40], 30% are uniform in [130,140] and 40% are uniform in [0,29], [41,129], [141,200].

As in Eliiyi and Azizoglu (2006, 2010) we use two uniform distributions for processing times: [5,10] ($p=1$) and [5,40] ($p=2$) and three uniform distributions for weights: $w_j=p_j$ ($w=1$), [5,10] (the low variability case, $w=2$) and [5,40] (the high variability case, $w=3$). We set $T=50$ and $S=100$.

For each of the twelve combinations of ready times, processing times and weights, 10 test problems are generated for $n=20, 30, \ldots, 100$ and 250 and $m=2, 3, 4$. For the OFJSS problem, we could not obtain solutions to our heuristic in reasonable time for 250 jobs and 4 machines instances. For the OFJSW problem, we include also large-sized instances with 10 machines and 250 and 500 jobs.

For the OFJSS problem, we use additional test problems generated by Rossi et al. (2010) using our generation scheme. Rossi et al. (2010) try 20, 40, 60, 80 and 100 jobs instances and generate 20 problem instances for each combination of r, p and w. We run our heuristic with their data and compare the solution values with those of their heuristics, namely greedy heuristics and genetic algorithm.

The performances of the algorithms are evaluated by their deviations from the optimal solutions and the solution times. We find the optimal solutions using the commercial software

INITIAL SOLUTION:
Compute two lower bounds LBS1, and LBS2;
Improve each lower bound solution;
Incumbent is the maximum of the improved solutions;
FIRST JOB SEARCH:
At each level (1,..., *m-2*) of the tree:
    Branch from node: Create at most *n* branches, one for each job, using dominance conditions to decide whether to fathom;
At each level *m-1* of the tree:
    Branch from the node;
    Compute UBS = $Min\{UBS_1, UBS_2\}$;
    Fathom node if necessary;
    Update Incumbent if solution is better than Incumbent;
TIME-PHASED MCNF SOLUTION:
For each of the created nodes by FIRST JOB SEARCH
    Run Algorithm Phased MCNF;
    Improve solution by insertion;
    Apply the two improvement algorithms to solution;
    Update Incumbent if solution is better than Incumbent;
End For
Return Incumbent.

Fig. 3. Pseudo-code of the two phase algorithm for the OFJSS problem.

Table 1
Performance of the beam search algorithm for $r=1$.

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
20	1	1	0.00	0.00	100	0.00	0.15	0.60	4.00	70	0.00	0.77	0.00	0.00	100	0.00	0.65
		2	0.52	2.54	60	0.00	0.54	0.00	0.00	100	0.00	0.29	0.00	0.00	100	0.00	0.62
		3	0.00	0.00	100	0.00	0.70	0.11	0.84	80	0.00	0.81	0.00	0.00	100	0.00	0.41
	2	1	0.00	0.00	100	0.00	0.44	0.33	1.33	70	0.00	0.39	0.45	2.03	60	0.00	0.79
		2	0.28	1.49	80	0.00	1.14	0.52	3.95	80	0.00	0.89	1.12	3.49	50	0.00	1.54
		3	1.13	6.30	60	0.00	0.69	2.63	6.55	10	0.00	0.85	1.40	3.30	30	0.00	0.96
30	1	1	0.00	0.00	100	0.00	0.13	0.00	0.00	100	0.00	0.34	0.05	0.50	90	0.00	1.35
		2	0.15	0.75	80	0.00	0.52	0.46	1.16	40	0.00	0.82	0.14	0.91	80	0.00	0.73
		3	0.41	1.29	40	0.00	0.62	0.32	1.30	40	0.00	1.13	0.13	0.47	60	0.00	0.96
	2	1	0.00	0.00	100	0.00	0.17	0.13	0.67	80	0.00	0.40	0.25	1.00	70	0.00	0.97
		2	0.12	1.23	90	0.00	0.65	0.61	2.27	60	0.00	1.42	1.51	3.57	40	0.02	2.26
		3	1.17	3.65	50	0.00	0.31	1.63	5.91	20	0.00	0.97	2.23	4.21	10	0.01	1.21
40	1	1	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.13	0.00	0.00	100	0.00	0.22
		2	0.44	1.47	60	0.00	0.78	0.70	2.15	40	0.00	1.05	0.56	1.44	30	0.00	1.36
		3	0.63	1.82	50	0.00	0.68	0.83	2.30	10	0.00	1.18	0.61	1.10	0	0.00	3.21
	2	1	0.00	0.00	100	0.00	0.35	0.07	0.67	90	0.00	0.61	0.10	0.50	80	0.00	0.61
		2	0.83	3.00	50	0.00	0.51	0.96	3.03	30	0.00	1.36	1.53	3.17	10	0.00	1.40
		3	0.33	1.86	60	0.00	0.59	1.74	3.94	20	0.00	1.27	2.65	4.48	0	0.00	4.82
50	1	1	0.00	0.00	100	0.00	0.13	0.00	0.00	100	0.00	0.13	0.00	0.00	100	0.00	0.27
		2	0.28	0.76	60	0.00	0.42	0.52	1.08	30	0.00	340.76	0.49	0.84	20	0.00	1.70
		3	0.56	2.36	60	0.00	0.66	0.83	1.49	20	0.00	1.27	0.78	1.84	0	0.00	3.94
	2	1	0.00	0.00	100	0.00	0.15	0.00	0.00	100	0.00	0.38	0.05	0.50	90	0.00	0.47
		2	0.26	2.63	90	0.00	0.50	0.55	2.91	60	0.00	1.07	1.48	3.01	0	0.00	2.65
		3	0.98	4.08	50	0.00	1.03	1.58	3.95	20	0.00	1.73	1.49	4.30	0	0.00	2.88
60	1	1	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.21	0.00	0.00	100	0.00	0.19
		2	0.53	1.36	50	0.00	0.68	0.25	1.02	70	0.00	0.63	0.53	2.25	30	0.00	1.65
		3	0.54	1.39	40	0.00	0.60	0.98	2.44	0	0.00	1.48	0.33	0.73	20	0.01	4.41
	2	1	0.00	0.00	100	0.00	0.25	0.00	0.00	100	0.00	0.39	0.00	0.00	100	0.00	0.24
		2	0.41	3.06	80	0.00	0.62	0.98	3.76	40	0.00	1.42	0.96	4.03	40	0.00	4.99
		3	0.82	4.04	60	0.00	0.50	2.01	4.64	10	0.00	2.02	2.61	5.45	0	0.00	5.73
70	1	1	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.49	0.00	0.00	100	0.00	0.19
		2	0.20	0.69	70	0.00	0.62	0.38	0.96	30	0.00	0.81	0.49	1.13	30	0.00	1.44
		3	0.65	1.93	30	0.00	1.30	0.72	1.53	20	0.00	1.93	0.79	1.82	20	0.00	1.49
	2	1	0.00	0.00	100	0.00	0.22	0.00	0.00	100	0.00	0.19	0.05	0.50	90	0.00	0.23
		2	0.60	3.09	70	0.00	0.71	0.90	2.08	40	0.00	1.29	1.20	2.80	20	0.00	1.84
		3	1.42	4.10	40	0.00	0.45	1.01	2.13	10	0.00	1.77	2.20	3.16	0	0.00	2.95

Table 1 (*continued*)

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
80	1	1	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.24	0.00	0.00	100	0.00	0.28
		2	0.39	1.33	60	0.00	0.77	0.40	0.92	40	0.00	0.60	0.37	0.74	20	0.00	1.66
		3	0.73	2.63	30	0.00	0.58	0.84	2.91	30	0.00	1.18	0.76	1.82	10	0.00	1.27
	2	1	0.00	0.00	100	0.00	0.18	0.00	0.00	100	0.00	0.32	0.00	0.00	100	0.00	0.29
		2	0.52	2.17	70	0.00	0.70	1.02	3.23	30	0.00	1.48	1.35	2.25	0	0.00	2.19
		3	1.45	4.58	30	0.00	0.45	1.83	4.58	10	0.00	1.85	1.77	2.65	0	0.00	5.01
90	1	1	0.00	0.00	100	0.00	0.16	0.00	0.00	100	0.00	0.21	0.00	0.00	100	0.00	0.20
		2	0.24	1.24	70	0.01	0.88	0.22	0.88	70	0.00	1.18	0.38	1.37	40	0.00	1.31
		3	0.22	0.66	50	0.00	0.75	0.59	1.40	0	0.00	2.18	0.75	1.50	20	0.00	1.48
	2	1	0.00	0.00	100	0.00	0.15	0.00	0.00	100	0.00	0.18	0.00	0.00	100	0.00	0.27
		2	0.42	2.27	80	0.00	0.91	1.15	2.74	30	0.00	1.24	1.44	1.95	0	0.02	3.91
		3	1.19	3.40	20	0.00	1.09	2.45	5.33	0	0.00	1.43	2.02	3.33	0	0.00	3.78
100	1	1	0.00	0.00	100	0.00	0.15	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.25
		2	0.37	1.26	50	0.00	0.85	0.35	1.28	50	0.00	36.52	0.28	0.75	40	0.00	1.34
		3	0.27	0.65	20	0.00	1.27	0.31	1.11	40	0.00	1.37	0.49	1.77	20	0.00	44.86
	2	1	0.00	0.00	100	0.00	0.09	0.00	0.00	100	0.00	0.21	0.00	0.00	100	0.00	0.25
		2	0.16	1.64	90	0.00	0.39	1.00	1.95	20	0.00	1.16	1.22	3.64	10	0.00	1.57
		3	1.34	3.99	10	0.00	0.50	1.57	2.78	0	0.00	2.28	1.88	3.01	0	0.00	4.99
250	1	1	0.00	0.00	100	0.00	0.16	0.00	0.00	100	0.00	0.52	0.00	0.00	100	0.01	0.59
		2	0.16	1.10	80	0.00	0.87	0.19	1.12	70	0.01	1.18	0.24	0.86	60	0.01	3.45
		3	0.37	0.79	30	0.00	0.85	0.17	0.51	40	0.01	1.89	0.22	0.58	10	0.01	2.46
	2	1	0.00	0.00	100	0.01	0.25	0.00	0.00	100	0.01	0.44	0.00	0.00	100	0.01	0.75
		2	0.51	1.42	40	0.00	0.81	0.58	1.42	30	0.00	1.71	0.43	1.02	40	0.00	2.04
		3	0.77	2.67	40	0.00	1.01	1.13	1.98	0	0.00	1.95	0.99	1.90	0	0.01	2.41

Table 2
Performance of the beam search algorithm for $r=2$.

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
20	1	1	0.54	3.41	80	0.00	0.18	0.33	3.28	90	0.00	0.12	0.00	0.00	100	0.00	0.14
		2	0.75	2.88	60	0.00	0.38	0.55	4.13	80	0.00	0.16	0.00	0.00	100	0.00	0.25
		3	0.00	0.00	100	0.00	0.53	0.00	0.00	100	0.00	0.59	0.09	0.92	90	0.00	0.75
	2	1	0.50	1.00	50	0.00	0.50	0.53	2.00	50	0.00	0.93	0.50	1.50	50	0.00	1.03
		2	0.48	3.13	80	0.00	0.36	1.45	5.63	40	0.00	0.70	1.41	3.70	40	0.00	1.41
		3	0.38	1.86	60	0.00	0.65	0.69	2.12	50	0.00	0.76	1.98	6.46	40	0.00	0.67
30	1	1	0.00	0.00	100	0.00	0.16	0.76	2.86	50	0.00	0.78	0.29	2.38	80	0.00	0.22
		2	0.34	0.89	60	0.00	0.26	0.64	2.45	60	0.00	0.72	0.05	0.55	90	0.00	0.16
		3	0.44	2.82	60	0.00	0.37	0.49	1.77	60	0.00	0.69	0.11	0.76	80	0.00	0.21
	2	1	0.00	0.00	100	0.00	0.37	0.20	0.67	70	0.00	0.77	0.40	1.50	40	0.00	0.59
		2	0.77	4.92	80	0.00	0.83	1.33	6.00	70	0.00	0.93	2.48	6.19	10	0.00	1.79
		3	0.35	2.00	80	0.00	0.42	0.84	3.35	50	0.00	0.62	1.24	4.26	50	0.00	1.08
40	1	1	0.00	0.00	100	0.00	0.27	0.83	4.96	80	0.00	0.18	0.51	2.00	70	0.00	0.85
		2	0.49	2.34	70	0.00	0.66	0.62	1.73	40	0.00	0.74	0.81	3.20	50	0.00	1.25
		3	0.94	4.12	40	0.00	0.63	1.35	4.30	20	0.00	0.49	0.84	2.60	20	0.00	0.76
	2	1	0.00	0.00	100	0.00	0.40	0.07	0.67	90	0.00	0.45	0.20	1.00	70	0.00	0.63
		2	0.46	3.33	80	0.00	0.41	1.17	5.22	50	0.00	2.78	1.39	4.12	40	0.00	1.41
		3	1.03	4.73	60	0.00	0.74	2.07	4.78	10	0.00	1.27	2.87	4.81	0	0.00	2.26
50	1	1	0.00	0.00	100	0.00	0.12	0.00	0.00	100	0.00	0.45	0.36	2.62	80	0.00	0.43
		2	0.92	2.40	40	0.00	0.50	1.06	1.69	10	0.00	0.65	1.08	2.20	10	0.00	3.39
		3	0.56	2.29	50	0.00	1.01	1.33	3.85	20	0.00	1.12	0.95	2.39	0	0.00	1.59
	2	1	0.00	0.00	100	0.00	0.25	0.00	0.00	100	0.00	0.50	0.15	1.00	80	0.00	0.66
		2	0.52	2.90	70	0.00	0.54	2.20	4.90	20	0.00	1.07	1.58	3.23	20	0.00	2.27
		3	0.94	6.19	70	0.00	0.59	3.06	6.99	10	0.00	1.14	2.10	4.24	0	0.00	4.78
60	1	1	0.00	0.00	100	0.00	0.10	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.18
		2	0.33	1.94	80	0.00	0.47	0.73	1.54	20	0.00	1.26	1.12	1.90	0	0.00	1.77
		3	1.04	3.67	40	0.00	0.48	1.58	2.68	10	0.00	1.16	1.22	2.25	0	0.00	1.73
	2	1	0.00	0.00	100	0.00	0.39	0.00	0.00	100	0.00	0.21	0.05	0.50	90	0.00	0.42
		2	0.36	1.22	70	0.00	0.67	0.95	2.48	40	0.00	1.20	2.23	3.55	0	0.00	1.88
		3	1.02	3.83	50	0.00	0.60	1.86	4.47	10	0.00	1.23	2.18	4.48	0	0.00	3.33
70	1	1	0.00	0.00	100	0.00	0.14	0.00	0.00	100	0.00	0.35	0.00	0.00	100	0.00	0.20
		2	0.83	1.52	20	0.00	0.48	1.03	2.15	10	0.00	0.65	1.01	2.03	10	0.00	79.29
		3	0.91	1.79	40	0.00	0.72	1.72	3.69	0	0.00	1.95	1.10	1.61	0	0.00	3.69

Table 2 (*continued*)

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
	2	1	0.00	0.00	100	0.00	0.15	0.07	0.67	90	0.00	0.31	0.05	0.50	90	0.00	0.36
		2	0.98	3.77	50	0.00	0.46	1.95	4.55	10	0.00	1.20	1.40	2.90	0	0.00	3.02
		3	1.14	3.65	40	0.00	0.48	2.20	4.84	10	0.00	1.29	2.49	4.81	0	0.00	3.91
80	1	1	0.00	0.00	100	0.00	0.16	0.00	0.00	100	0.00	0.23	0.00	0.00	100	0.00	0.18
		2	0.53	1.30	40	0.00	0.43	1.01	2.22	10	0.00	1.11	1.07	1.59	0	0.00	1.25
		3	0.70	1.39	30	0.00	0.60	1.44	2.64	10	0.00	1.57	1.20	2.19	10	0.00	1.85
	2	1	0.00	0.00	100	0.00	0.19	0.00	0.00	100	0.00	0.38	0.05	0.50	90	0.00	0.35
		2	1.10	3.19	20	0.00	0.64	1.58	3.57	0	0.00	1.60	1.56	4.24	20	0.00	4.31
		3	0.88	3.32	50	0.00	0.43	1.63	4.77	10	0.00	1.94	2.33	6.18	10	0.00	4.95
90	1	1	0.00	0.00	100	0.00	0.15	0.00	0.00	100	0.00	0.22	0.00	0.00	100	0.01	0.17
		2	0.83	3.03	50	0.00	0.51	0.83	1.43	10	0.00	1.11	1.12	1.88	0	0.01	1.73
		3	0.84	2.90	30	0.00	1.36	1.80	3.67	0	0.00	1.09	2.68	4.18	0	0.00	1.68
	2	1	0.00	0.00	100	0.00	0.21	0.00	0.00	100	0.00	0.15	0.00	0.00	100	0.00	0.26
		2	0.51	2.15	70	0.00	0.98	1.36	2.63	20	0.00	1.31	1.71	2.92	10	0.00	3.99
		3	1.36	3.75	40	0.00	1.18	2.11	5.07	0	0.00	1.48	2.59	5.52	10	0.00	5.12
100	1	1	0.00	0.00	100	0.00	0.11	0.00	0.00	100	0.00	0.26	0.00	0.00	100	0.00	0.24
		2	0.70	1.96	30	0.00	0.48	0.58	1.79	40	0.00	1.30	0.81	1.45	10	0.03	1.53
		3	0.53	2.19	30	0.00	0.43	1.07	1.65	10	0.00	1.74	1.45	2.28	0	0.00	2.23
	2	1	0.00	0.00	100	0.00	0.23	0.00	0.00	100	0.00	0.23	0.00	0.00	100	0.00	0.25
		2	0.81	3.09	60	0.00	0.32	1.21	2.80	10	0.00	1.51	2.04	3.87	0	0.02	1.63
		3	0.92	3.54	40	0.00	0.65	1.95	3.51	0	0.00	2.19	2.84	5.56	0	0.00	3.97
250	1	1	0.00	0.00	100	0.00	0.18	0.00	0.00	100	0.01	0.81	0.00	0.00	100	0.01	0.32
		2	0.69	2.89	40	0.00	0.72	0.67	1.58	30	0.01	1.51	0.49	0.63	20	0.01	2.52
		3	1.13	1.91	10	0.00	0.79	0.67	3.00	0	0.01	1.87	0.61	1.16	10	0.01	2.68
	2	1	0.00	0.00	100	0.01	0.36	0.00	0.00	100	0.01	0.49	0.00	0.00	100	0.01	0.70
		2	0.46	1.57	70	0.00	0.75	1.25	2.52	0	0.00	1.65	1.35	2.46	10	0.00	2.21
		3	1.69	3.63	20	0.00	0.59	1.61	3.31	10	0.00	2.31	2.10	4.81	10	0.01	3.00

Table 3
Performance of the beam search algorithm for large instances ($m=10$).

n	*p*	*w*	*r*=1					*r*=2				
			% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
250	1	1	0.00	0.00	100	0.01	0.39	0.00	0.00	100	0.01	0.36
		2	0.24	0.29	0	0.02	6.69	0.75	1.05	0	0.02	21.81
		3	0.32	0.66	0	0.02	9.41	1.18	2.48	0	0.03	40.55
	2	1	0.00	0.00	100	0.01	0.59	0.00	0.00	100	0.01	0.44
		2	1.11	1.94	0	0.01	26.48	1.71	2.43	0	0.01	103.00
		3	1.28	2.01	0	0.01	333.92	2.37	3.72	0	0.01	487.77[a]
500	1	1	0.00	0.00	100	0.03	2.92	0.00	0.00	100	0.03	1.24
		2	0.21	0.37	10	0.05	7.32	0.63	1.01	0	0.06	13.46
		3	0.12	0.27	10	0.06	7.53	0.79	1.36	0	0.07	18.54
	2	1	0.00	0.00	100	0.03	1.13	0.00	0.00	100	0.02	1.54
		2	0.64	1.06	0	0.02	11.60	1.55	2.37	0	0.03	44.97
		3	0.85	1.59	0	0.03	29.03	1.82	2.49	0	0.03	589.67*

[a] One out of ten instances could not be solved in 3600 s by CPLEX.

Table 4
Performance of the time-phased MCNF algorithm for $r=1$.

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
20	1	1	0.00	0.00	100	1.76	0.23	0.00	0.00	100	10.61	0.20	0.00	0.00	100	18.48	0.33
		2	0.18	1.77	90	1.67	0.16	0.00	0.00	100	9.47	0.20	0.33	3.25	90	20.17	0.32
		3	0.00	0.00	100	1.82	0.17	0.00	0.00	100	8.93	0.20	0.00	0.00	100	18.16	0.33
	2	1	0.00	0.00	100	0.37	0.16	0.04	0.43	90	5.09	0.32	0.48	1.56	60	21.49	0.33
		2	0.58	5.80	90	0.75	0.17	0.57	5.75	90	8.98	0.23	1.00	5.00	60	24.08	0.29
		3	0.00	0.00	100	0.96	0.18	0.17	1.74	90	8.76	0.25	0.94	4.88	70	17.52	0.27

Table 4 (*continued*)

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
30	1	1	0.00	0.00	100	4.06	0.12	0.00	0.00	100	16.85	0.16	0.06	0.56	90	35.06	0.32
		2	0.00	0.00	100	4.58	0.12	0.12	1.16	90	19.79	0.16	0.00	0.00	100	31.10	0.31
		3	0.19	1.92	90	5.03	0.12	0.22	1.41	80	23.40	0.15	0.12	0.87	80	47.64	0.31
	2	1	0.00	0.00	100	1.05	0.18	0.34	2.29	80	12.79	0.28	0.08	0.83	90	33.42	0.53
		2	0.00	0.00	100	1.96	0.14	0.19	0.99	80	17.84	0.15	0.93	3.08	50	38.36	0.45
		3	0.00	0.00	100	1.54	0.13	0.20	1.42	80	18.53	0.20	0.63	2.70	70	38.17	0.34
40	1	1	0.20	1.96	90	6.64	0.13	0.00	0.00	100	12.31	0.16	0.13	1.33	90	24.95	0.24
		2	0.00	0.00	100	5.86	0.15	0.11	1.09	90	12.40	0.16	0.39	1.91	70	25.36	0.26
		3	0.00	0.00	100	5.91	0.14	0.41	3.45	80	13.81	0.16	0.20	1.53	80	23.32	0.27
	2	1	0.00	0.00	100	1.54	0.17	0.18	1.45	80	8.74	0.35	0.00	0.00	100	26.00	0.66
		2	0.42	4.17	90	3.61	0.14	0.00	0.00	100	15.89	0.22	0.14	0.73	80	26.97	0.38
		3	0.23	2.28	90	2.61	0.18	0.00	0.00	100	19.25	0.22	0.69	1.68	20	22.83	0.57
50	1	1	0.00	0.00	100	7.89	0.19	0.10	0.49	80	23.65	0.22	0.45	2.63	70	47.92	0.32
		2	0.00	0.00	100	8.56	0.14	0.28	1.90	80	21.48	0.30	0.11	1.12	90	48.20	0.33
		3	0.00	0.00	100	8.70	0.16	0.36	2.93	80	23.00	0.37	0.12	0.93	80	50.52	0.33
	2	1	0.00	0.00	100	1.71	0.28	0.07	0.72	90	12.91	0.60	0.08	0.81	90	48.53	1.06
		2	0.00	0.00	100	3.58	0.19	0.75	2.08	50	21.19	0.34	0.50	1.85	70	46.78	0.65
		3	0.19	1.91	90	5.22	0.22	0.54	3.02	70	20.26	0.36	0.98	3.54	40	45.38	0.66
60	1	1	0.19	0.70	70	9.90	0.17	0.12	0.43	70	24.48	0.36	0.26	1.06	70	65.39	0.67
		2	0.05	0.51	90	9.86	0.15	0.88	2.54	40	23.95	0.32	0.73	3.01	60	61.20	0.60
		3	0.00	0.00	100	9.61	0.21	0.12	0.66	80	24.08	0.24	0.30	0.98	50	57.89	0.61
	2	1	0.00	0.00	100	2.21	0.30	0.10	1.04	90	20.57	0.71	0.13	0.78	80	60.89	1.52
		2	0.11	1.09	90	7.25	0.16	0.83	3.31	50	22.85	0.45	1.46	4.79	40	55.46	0.68
		3	0.25	2.47	90	6.27	0.21	1.01	3.28	50	22.84	0.40	0.63	3.50	60	60.15	0.70
70	1	1	0.00	0.00	100	7.38	0.22	0.54	1.23	30	22.82	0.23	0.39	1.34	50	66.37	3.84
		2	0.11	1.12	90	6.94	0.22	0.80	4.71	70	28.52	0.28	0.19	0.95	70	74.60	0.57
		3	0.09	0.58	80	6.73	0.24	0.19	1.14	70	22.69	0.28	0.20	1.01	70	61.95	0.63
	2	1	0.00	0.00	100	2.04	0.56	0.00	0.00	100	19.51	0.64	0.05	0.26	80	70.95	1.77
		2	0.55	2.86	70	4.67	0.29	0.86	4.44	50	17.38	0.36	0.89	2.58	40	52.92	0.82
		3	0.08	0.81	90	5.06	0.28	0.74	2.36	50	15.63	0.42	0.93	4.27	20	55.99	1.31
80	1	1	0.00	0.00	100	7.78	0.24	0.36	1.20	40	29.37	0.52	0.56	1.54	40	101.34	1.00
		2	0.05	0.49	90	7.38	0.21	0.40	2.11	40	27.76	0.37	0.48	1.55	30	89.76	0.77
		3	0.00	0.00	100	7.49	0.32	0.12	0.75	70	28.01	0.44	0.42	1.66	50	80.90	0.52
	2	1	0.00	0.00	100	2.19	0.54	0.00	0.00	100	23.89	0.87	0.03	0.26	90	107.38	1.96
		2	0.36	2.50	80	5.97	0.36	0.13	1.33	90	17.48	0.58	1.08	2.68	20	78.67	0.99
		3	0.00	0.00	100	5.93	0.46	0.93	3.48	40	26.05	0.55	0.71	2.18	40	72.32	0.88
90	1	1	0.00	0.00	100	9.01	0.27	0.50	1.52	40	33.65	0.45	0.96	1.57	10	177.18	1.72
		2	0.20	1.52	80	7.63	0.26	0.25	1.10	60	30.77	0.37	0.67	2.25	30	152.00	1.17
		3	0.07	0.46	80	7.84	0.28	0.47	1.37	50	33.18	0.53	0.37	1.06	40	132.40	1.36
	2	1	0.00	0.00	100	2.90	0.46	0.03	0.35	90	24.54	1.05	0.10	0.76	80	194.13	1.97
		2	0.00	0.00	100	5.32	0.30	0.56	2.47	60	24.18	0.59	0.82	1.41	10	103.65	1.32
		3	0.14	1.40	90	5.98	0.28	0.67	2.20	50	27.69	0.57	1.32	3.11	20	112.68	0.99
100	1	1	0.27	1.64	70	11.79	0.47	0.34	0.76	40	41.03	0.83	0.93	2.32	10	275.44	2.04
		2	0.14	1.44	90	9.69	0.24	0.53	1.35	40	40.65	0.68	1.66	4.23	0	247.43	0.93
		3	0.46	1.28	50	8.60	0.31	0.55	1.74	40	39.91	0.53	0.85	2.07	30	206.19	1.04
	2	1	0.00	0.00	100	3.73	0.48	0.03	0.34	90	35.36	1.26	0.05	0.51	90	300.78	2.55
		2	0.15	1.50	90	8.46	0.44	0.85	1.99	20	29.19	0.61	1.69	3.89	10	162.04	1.49
		3	0.00	0.00	100	8.09	0.48	0.95	2.85	50	21.47	0.87	1.16	2.69	30	168.70	1.49
250	1	1	0.00	0.00	100	14.95	2.55	0.24	0.67	50	682.65	9.47	–	–	–	–	–
		2	0.04	0.41	90	8.81	1.74	0.82	1.98	20	373.84	6.62	–	–	–	–	–
		3	0.38	1.46	60	9.51	1.89	1.32	2.66	10	347.58	4.70	–	–	–	–	–
	2	1	0.00	0.00	100	13.67	4.30	0.00	0.00	100	683.15	8.96	–	–	–	–	–
		2	0.12	0.64	80	11.18	1.40	1.02	4.18	20	57.79	5.90	–	–	–	–	–
		3	0.00	0.00	100	10.84	1.39	1.18	3.45	40	71.86	4.30	–	–	–	–	–

CPLEX 12.1. The optimal solutions are reached in reasonable times.

To set the parameters of the algorithm we carry a pilot run with fewer instances. For the OFJSW problem, we take 120 random problem instances with 40 and 80 jobs and 4 machines. To find the best values for (α,β) pair, we try $(2m,m)$, $(8m,4m)$, $(16m,8m)$ and $(20m,10m)$. As (α,β) values increase, solution quality also tends to increase at an expense of higher solution times. We find that $(16m,8m)$ is the best combination in terms of solution quality and time.

For the OFJSS problem, to find the best value for parameter γ, the pilot runs are carried with 40, 70 and 80 jobs and 4 machines, or a total of 120 instances. We try $\gamma=25m$, $\gamma=50m$, $\gamma=100m$ and find that $\gamma=100m$ provides higher good-quality solutions in reasonable times.

5.1. The OFJSW problem

In this section we discuss the performance of our filtered beam search algorithm developed for the OFJSW problem. Tables 1 and 2

give the average and maximum deviations of our lower bounds from the optimal solutions for $r=1$ and $r=2$, respectively. The "%opt" column gives the percentage of the instances our filtered beam search algorithm finds the optimal solution. The tables also include the average CPU times for the filtered beam search and CPLEX.

Table 5
Performance of the time-phased MCNF algorithm for $r=2$.

n	*p*	*w*	*m*=2					*m*=3					*m*=4				
			% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX	% dev. opt.		% opt.	Avg.	CPLEX
			Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU	Avg.	Max.		CPU	CPU
20	1	1	0.00	0.00	100	0.87	0.16	0.10	1.02	90	8.76	0.24	0.25	2.48	90	20.97	0.23
		2	0.00	0.00	100	1.06	0.16	0.00	0.00	100	6.98	0.23	0.00	0.00	100	18.80	0.22
		3	0.00	0.00	100	0.71	0.16	0.03	0.35	90	9.35	0.24	0.00	0.00	100	21.73	0.22
	2	1	0.07	0.68	90	0.37	0.17	0.52	5.18	90	3.74	0.25	0.15	1.51	90	14.68	0.23
		2	0.00	0.00	100	0.38	0.18	0.75	7.46	90	5.21	0.27	0.00	0.00	100	19.62	0.23
		3	0.00	0.00	100	0.46	0.16	0.00	0.00	100	6.38	0.25	0.20	1.95	90	18.60	0.22
30	1	1	0.00	0.00	100	2.64	0.12	0.00	0.00	100	14.74	0.18	0.28	1.59	80	32.75	0.32
		2	0.10	1.03	90	2.49	0.12	0.00	0.00	100	21.79	0.19	0.00	0.00	100	31.76	0.31
		3	0.00	0.00	100	2.00	0.13	0.00	0.00	100	21.73	0.17	0.27	2.74	90	31.40	0.31
	2	1	0.00	0.00	100	0.80	0.17	0.23	2.32	90	9.79	0.26	0.34	1.97	70	17.94	0.41
		2	0.00	0.00	100	0.82	0.14	0.00	0.00	100	14.95	0.27	0.24	2.44	90	18.07	0.29
		3	0.15	1.53	90	0.91	0.13	0.09	0.92	90	12.45	0.24	0.21	2.08	90	21.01	0.31
40	1	1	0.00	0.00	100	4.35	0.13	0.06	0.63	90	11.12	0.19	0.00	0.00	100	20.99	0.25
		2	0.00	0.00	100	4.58	0.14	0.00	0.00	100	10.72	0.19	0.22	1.63	80	18.94	0.31
		3	0.00	0.00	100	3.67	0.13	0.09	0.93	90	13.89	0.20	0.00	0.00	100	19.00	0.25
	2	1	0.27	2.67	90	1.63	0.20	0.00	0.00	100	11.21	0.30	0.44	3.46	70	18.03	0.47
		2	0.28	2.78	90	2.36	0.13	0.25	1.63	80	12.91	0.23	0.17	0.90	80	17.62	0.29
		3	0.00	0.00	100	1.95	0.13	0.46	2.50	70	10.52	0.20	1.26	3.75	30	19.81	0.34
50	1	1	0.00	0.00	100	7.17	0.14	0.00	0.00	100	20.05	0.22	0.15	1.50	90	40.11	0.29
		2	0.00	0.00	100	6.24	0.16	0.00	0.00	100	20.93	0.31	0.05	0.47	90	38.73	0.27
		3	0.00	0.00	100	7.39	0.15	0.06	0.64	90	19.55	0.25	0.38	2.26	70	45.48	0.31
	2	1	0.00	0.00	100	2.18	0.26	0.12	1.21	90	16.25	0.42	0.22	1.13	70	37.01	0.82
		2	0.00	0.00	100	2.50	0.15	0.00	0.00	100	16.50	0.31	0.00	0.00	100	41.73	0.51
		3	0.00	0.00	100	2.50	0.15	0.03	0.30	90	17.17	0.29	0.62	2.47	40	35.83	0.42
60	1	1	0.08	0.79	90	7.80	0.19	0.13	1.25	90	21.45	0.29	0.45	2.37	80	48.11	0.41
		2	0.31	3.09	90	8.13	0.22	0.00	0.00	100	20.06	0.40	0.09	0.93	90	50.70	0.45
		3	0.55	5.54	90	7.96	0.15	0.00	0.00	100	20.83	0.33	0.21	1.08	80	48.60	0.44
	2	1	0.00	0.00	100	3.08	0.31	0.18	1.80	90	17.28	0.72	0.59	2.01	60	47.35	0.97
		2	0.00	0.00	100	3.45	0.22	0.18	0.95	80	16.30	0.32	0.43	2.42	60	39.56	0.60
		3	0.00	0.00	100	5.20	0.18	0.15	0.70	70	18.69	0.51	1.14	5.96	50	41.79	0.55
70	1	1	0.00	0.00	100	5.55	0.26	0.19	1.89	90	20.79	0.30	0.29	2.90	90	51.65	0.47
		2	0.00	0.00	100	5.05	0.23	0.21	1.05	70	14.26	0.33	0.05	0.46	90	40.25	0.37
		3	0.07	0.36	80	5.56	0.20	0.00	0.00	100	13.31	0.32	0.11	0.51	60	40.10	0.31
	2	1	0.00	0.00	100	2.57	0.42	0.00	0.00	100	22.90	0.67	0.48	3.14	60	50.25	0.85
		2	0.00	0.00	100	3.04	0.23	0.10	1.01	90	11.11	0.65	0.41	0.74	40	37.30	0.65
		3	0.00	0.00	100	3.39	0.26	0.18	0.91	80	12.25	0.61	0.72	3.22	50	40.17	0.86
80	1	1	0.06	0.65	90	6.14	0.38	0.20	0.93	70	22.70	0.29	0.24	1.26	60	64.68	0.60
		2	0.28	2.15	80	6.05	0.36	0.34	2.06	60	23.85	0.35	0.39	2.80	70	43.74	0.41
		3	0.04	0.39	90	5.84	0.18	0.22	0.70	60	20.17	0.40	0.42	1.42	50	48.95	0.51
	2	1	0.00	0.00	100	4.03	0.53	0.04	0.36	90	25.32	0.76	0.00	0.00	100	92.64	1.26
		2	0.00	0.00	100	4.10	0.39	0.99	3.55	50	12.73	0.58	0.81	3.25	50	51.23	0.71
		3	0.29	2.93	90	3.25	0.42	0.02	0.21	90	12.55	0.50	1.35	4.32	20	57.22	0.94
90	1	1	0.13	0.65	80	6.25	0.19	0.10	0.53	80	27.78	0.40	0.37	1.37	70	99.63	0.70
		2	0.00	0.00	100	7.59	0.26	0.48	2.12	60	27.44	0.46	0.04	0.35	90	76.54	0.59
		3	0.16	1.56	90	6.22	0.29	0.38	1.84	70	17.97	0.46	0.10	0.47	70	76.00	0.62
	2	1	0.00	0.00	100	3.04	0.35	0.00	0.00	100	26.07	1.01	0.00	0.00	100	144.08	1.45
		2	0.29	1.80	80	4.34	0.29	0.51	2.29	60	15.52	0.56	0.47	1.84	50	55.02	1.19
		3	0.27	2.17	80	4.51	0.39	0.50	2.63	70	14.53	0.78	1.45	5.17	30	58.12	0.94
100	1	1	0.00	0.00	100	8.13	0.24	0.55	1.75	40	22.33	0.53	0.22	0.79	70	190.40	0.72
		2	0.00	0.00	100	8.62	0.27	0.26	1.35	70	18.37	0.54	0.26	1.20	70	119.35	0.98
		3	0.00	0.00	100	8.78	0.26	0.61	2.29	70	23.97	0.60	0.36	2.47	60	96.57	0.88
	2	1	0.00	0.00	100	4.61	0.50	0.07	0.70	90	28.26	1.26	0.05	0.27	80	248.86	1.82
		2	0.00	0.00	100	6.49	0.69	0.65	3.01	60	25.69	0.84	1.00	3.16	40	80.31	0.93
		3	0.00	0.00	100	6.99	0.28	0.24	0.97	70	24.66	0.75	1.35	4.15	10	108.98	0.81
250	1	1	0.00	0.00	100	15.07	1.48	0.28	1.03	40	406.03	4.13	–	–	–	–	–
		2	0.05	0.47	90	9.60	1.27	0.91	2.12	30	119.81	3.46	–	–	–	–	–
		3	0.00	0.00	100	9.04	1.14	0.65	1.66	20	88.31	3.28	–	–	–	–	–
	2	1	0.00	0.00	100	13.63	2.05	0.00	0.00	100	634.38	5.81	–	–	–	–	–
		2	0.91	3.68	70	7.05	1.30	1.51	3.35	20	34.71	5.89	–	–	–	–	–
		3	0.00	0.00	100	8.05	1.48	0.55	1.96	40	42.67	2.57	–	–	–	–	–

Table 6
Performance of the time-phased MCNF algorithm vs. GH and GGA by Rossi et al. (2010) for $r=1$.

n	p	w	$m=2$									$m=3$									$m=4$								
			MCNF			GH			GGA			MCNF			GH			GGA			MCNF			GH			GGA		
			% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.
			Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.	
20	1	1	0.17	3.23	95	1.21	7.14	75	0.00	0.00	100	0.00	0.00	100	0.68	5.84	80	0.00	0.00	100	0.00	0.00	100	0.68	6.40	80	0.00	0.00	100
		2	0.00	0.00	100	0.90	6.72	80	0.00	0.00	100	0.16	3.13	90	0.95	4.55	75	0.00	0.00	100	0.00	0.00	100	1.14	5.94	70	0.00	0.00	100
		3	0.23	4.35	95	0.65	4.32	70	0.00	0.00	100	0.39	7.38	95	0.51	3.38	75	0.00	0.00	100	0.46	5.24	85	0.48	5.30	85	0.00	0.00	100
	2	1	0.38	7.14	95	0.17	2.82	90	0.00	0.00	100	0.13	2.08	90	0.64	4.87	75	0.00	0.00	100	0.41	3.06	80	0.51	3.55	80	0.00	0.00	100
		2	0.00	0.00	100	0.65	4.76	80	0.10	2.38	95	0.16	2.04	90	2.03	9.71	50	0.00	0.00	100	0.20	1.63	85	1.20	9.68	60	0.00	0.00	100
		3	0.17	3.19	95	1.26	6.10	65	0.00	0.00	100	0.00	0.00	100	1.19	9.27	70	0.00	0.00	100	0.53	8.40	85	0.65	6.36	80	0.00	0.00	100
40	1	1	0.17	1.99	90	1.65	7.50	50	0.00	0.00	100	0.10	1.18	90	0.62	2.94	55	0.00	0.00	100	0.06	0.82	90	1.72	5.88	25	0.00	0.00	100
		2	0.07	1.38	95	1.34	5.03	40	0.00	0.00	100	0.02	0.39	95	1.10	10.00	55	0.00	0.00	100	0.34	2.28	80	1.07	7.69	55	0.00	0.00	100
		3	0.00	0.00	100	2.24	8.88	45	0.00	0.00	100	0.05	0.96	95	0.95	5.26	60	0.00	0.00	100	0.09	1.43	90	1.07	4.44	45	0.00	0.00	100
	2	1	0.00	0.00	100	0.63	3.35	65	0.00	0.00	100	0.14	1.66	85	0.90	4.17	40	0.03	0.78	95	0.35	3.20	65	1.17	3.70	40	0.00	0.00	100
		2	0.13	1.03	85	1.39	4.92	45	0.06	1.64	95	0.30	2.61	85	2.43	15.71	35	0.00	0.00	100	0.37	3.47	75	1.97	6.82	25	0.00	0.00	100
		3	0.95	18.03	95	1.50	9.44	45	0.05	1.01	95	0.70	4.58	55	1.35	7.30	40	0.00	0.00	100	0.36	1.77	60	2.03	8.14	15	0.00	0.00	100
60	1	1	0.14	2.16	90	1.49	9.52	35	0.00	0.00	100	0.03	0.63	95	0.44	2.08	65	0.00	0.00	100	0.32	1.67	60	1.22	3.77	40	0.00	0.00	100
		2	0.13	1.67	85	0.53	2.40	60	0.00	0.00	100	0.08	1.54	95	1.15	4.43	40	0.00	0.00	100	0.34	1.69	70	1.69	7.55	30	0.00	0.00	100
		3	0.06	0.89	85	1.76	8.89	30	0.00	0.00	100	0.04	0.48	90	0.75	6.69	55	0.00	0.00	100	0.40	1.42	65	1.66	3.68	15	0.00	0.00	100
	2	1	0.93	17.65	95	0.63	2.80	60	0.00	0.00	100	0.30	1.68	75	1.12	4.93	40	0.02	0.19	95	0.19	1.80	75	1.48	6.59	20	0.00	0.00	100
		2	0.47	3.90	80	0.59	2.58	60	0.00	0.00	100	0.25	3.06	80	3.03	10.67	35	0.00	0.00	100	0.18	1.44	75	1.91	8.70	45	0.00	0.00	100
		3	0.09	1.17	85	2.27	7.74	40	0.00	0.00	100	0.41	1.88	70	1.15	6.05	55	0.00	0.00	100	0.53	2.31	50	2.56	8.78	45	0.00	0.00	100
80	1	1	0.08	0.72	85	1.76	19.23	50	0.00	0.00	100	0.24	0.90	65	1.36	4.74	20	0.00	0.00	100	0.25	1.03	65	0.97	2.94	20	0.03	0.49	95
		2	0.36	2.33	75	0.99	4.26	45	0.03	2.56	95	0.00	0.00	95	1.33	6.07	30	0.00	0.00	100	0.27	2.86	70	1.24	3.87	35	0.00	0.00	100
		3	0.12	1.26	90	0.51	3.49	60	0.04	1.20	95	0.17	2.01	80	1.19	3.92	50	0.00	0.00	100	0.36	2.48	55	1.44	3.52	10	0.00	0.00	100
	2	1	0.00	0.00	100	0.73	3.77	50	0.06	0.71	95	0.01	0.25	90	1.56	6.98	30	0.00	0.00	100	0.49	2.17	30	1.35	8.20	10	0.00	0.00	100
		2	0.06	0.61	90	2.23	25.93	55	0.09	2.50	95	0.41	2.11	65	2.69	6.56	25	0.04	0.00	100	0.48	3.81	70	2.15	9.52	30	0.06	0.85	95
		3	0.42	3.17	80	2.30	11.43	35	0.07	2.79	95	0.67	3.95	65	3.01	15.02	30	0.00	0.00	100	0.68	2.81	45	3.01	6.31	15	0.00	0.00	100
100	1	1	0.20	1.69	75	1.89	7.46	25	0.00	0.00	100	0.36	1.15	55	1.29	4.17	20	0.06	0.55	90	0.32	1.28	45	1.52	4.42	20	0.03	0.22	90
		2	0.07	1.36	95	2.86	33.33	50	0.00	0.00	100	0.27	1.15	60	1.38	7.30	30	0.05	0.76	90	0.51	2.38	40	1.21	4.19	30	0.00	0.00	100
		3	0.25	2.11	75	1.59	9.20	35	0.00	0.00	100	0.20	1.20	55	1.25	6.19	15	0.00	0.00	100	0.34	1.89	45	2.08	5.70	0	0.00	0.00	100
	2	1	0.03	0.62	95	1.07	4.41	25	0.00	0.00	100	0.06	0.54	85	1.47	7.02	35	0.03	0.39	95	0.14	0.87	65	2.15	9.43	25	0.01	0.15	95
		2	0.06	1.10	95	1.71	8.00	45	0.00	0.00	100	0.58	2.34	40	1.70	5.33	10	0.08	1.15	95	1.11	3.60	35	2.51	5.32	15	0.00	0.00	100
		3	0.26	2.81	90	1.29	9.32	55	0.03	0.59	95	0.46	2.87	65	2.99	8.06	20	0.00	0.00	100	0.65	3.88	50	1.92	5.10	15	0.00	0.00	100

Table 7
Performance of the time-phased MCNF algorithm vs. GH and GGA by Rossi et al. (2010) for $r=2$.

n	p	w	$m=2$									$m=3$									$m=4$								
			MCNF			GH			GGA			MCNF			GH			GGA			MCNF			GH			GGA		
			% dev. opt.		% opt.	% dev. opt.		% opt	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.	% dev. opt.		% opt.
			Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.		Avg.	Max.	
20	1	1	0.00	0.00	100	0.61	6.82	85	0.00	0.00	100	0.00	0.00	100	0.49	8.75	90	0.00	0.00	100	0.18	3.51	95	0.28	2.63	85	0.00	0.00	100
		2	0.00	0.00	100	1.31	16.67	80	0.00	0.00	100	0.00	0.00	100	1.20	6.31	70	0.00	0.00	100	0.11	1.08	90	0.89	9.80	80	0.00	0.00	100
		3	0.00	0.00	100	1.18	7.64	75	0.00	0.00	100	0.00	0.00	100	0.47	9.45	95	0.00	0.00	100	0.00	0.00	100	0.20	3.70	90	0.00	0.00	100
	2	1	0.20	2.43	90	0.22	3.73	90	0.00	0.00	100	0.02	0.45	95	0.13	2.24	90	0.00	0.00	100	0.46	2.51	70	0.53	3.88	75	0.00	0.00	100
		2	0.00	0.00	100	0.92	8.77	80	0.00	0.00	100	0.10	1.98	95	1.41	6.98	70	0.00	0.00	100	0.32	5.15	85	1.71	6.82	40	0.00	0.00	100
		3	0.00	0.00	100	0.44	4.09	80	0.00	0.00	100	0.34	2.99	85	1.41	6.98	70	0.00	0.00	100	0.03	0.59	95	1.71	6.82	40	0.00	0.00	100
40	1	1	0.51	4.73	80	1.85	13.04	45	0.00	0.00	100	0.18	3.51	95	0.36	2.56	80	0.00	0.00	100	0.03	0.64	95	0.38	2.99	75	0.00	0.00	100
		2	0.00	0.00	100	1.00	8.51	80	0.00	0.00	100	0.11	1.08	90	0.69	3.68	70	0.00	0.00	100	0.00	0.00	100	1.65	5.30	30	0.00	0.00	100
		3	0.10	1.98	95	1.21	7.09	65	0.00	0.00	100	0.00	0.00	100	1.62	5.66	45	0.00	0.00	100	0.07	1.33	90	0.68	4.06	45	0.00	0.00	100
	2	1	0.18	3.48	95	0.44	4.41	80	0.00	0.00	100	0.31	2.31	80	1.18	5.56	35	0.00	0.00	100	0.50	3.22	70	1.14	6.45	55	0.00	0.00	100
		2	0.00	0.00	100	1.81	15.00	70	0.00	0.00	100	0.38	4.11	75	1.49	7.58	50	0.00	0.00	100	0.09	1.75	95	1.93	17.39	55	0.00	0.00	100
		3	0.20	1.94	85	2.35	12.59	60	0.13	3.51	95	0.28	1.99	80	2.02	9.35	30	0.03	0.87	95	0.17	1.00	70	1.93	8.23	35	0.00	0.00	100
60	1	1	0.00	0.00	100	0.58	4.81	75	0.00	0.00	100	0.33	1.78	70	0.94	5.79	45	0.00	0.00	100	0.19	1.18	75	0.46	3.51	60	0.00	0.00	100
		2	0.00	0.00	100	0.15	1.56	90	0.00	0.00	100	0.08	1.02	90	0.63	4.55	65	0.00	0.00	100	0.27	2.10	80	1.35	5.41	40	0.00	0.00	100
		3	0.00	0.00	100	0.28	1.25	65	0.00	0.00	100	0.18	2.32	90	0.76	2.83	50	0.00	0.00	100	0.19	1.76	75	0.97	4.56	35	0.00	0.00	100
	2	1	0.00	0.00	100	0.62	3.81	75	0.19	2.35	85	0.27	1.86	70	1.26	4.35	30	0.04	1.71	95	0.20	2.04	75	0.82	5.63	50	0.00	0.00	100
		2	0.15	2.88	95	1.09	6.06	65	0.00	0.00	100	0.25	2.38	85	1.60	10.62	50	0.00	0.00	100	0.54	3.77	65	3.13	15.71	30	0.00	0.00	100
		3	0.07	1.34	95	3.01	13.54	35	0.00	0.00	100	0.70	3.89	55	1.29	7.52	50	0.00	0.00	100	0.44	5.24	80	3.51	25.56	30	0.00	0.00	100
80	1	1	0.06	1.05	95	0.30	2.30	65	0.00	0.00	100	0.20	2.56	90	1.04	3.68	45	0.00	0.00	100	0.11	0.81	85	0.52	1.92	50	0.00	0.00	100
		2	0.02	0.45	95	0.79	3.64	45	0.03	2.56	85	0.17	1.81	90	1.41	4.12	35	0.11	2.14	95	0.26	1.92	80	0.90	5.11	50	0.00	0.00	100
		3	0.15	2.86	95	1.19	6.73	45	0.06	2.86	90	0.32	2.49	75	1.17	3.77	45	0.01	0.32	95	0.22	2.03	75	1.00	7.37	25	0.00	0.00	100
	2	1	0.19	1.52	80	0.89	4.29	50	0.00	0.00	100	0.20	1.26	80	0.78	3.11	35	0.00	0.00	100	0.49	1.95	60	1.21	5.70	35	0.00	0.00	100
		2	0.00	0.00	100	0.68	3.70	70	0.06	1.04	95	0.27	1.91	75	2.03	5.81	20	0.00	0.00	100	0.54	3.76	65	2.55	8.16	25	0.03	0.45	95
		3	0.48	2.50	75	1.72	11.05	60	0.06	0.87	95	0.45	3.61	70	1.18	8.21	45	0.07	2.44	95	0.65	5.08	70	2.35	11.66	20	0.00	0.00	100
100	1	1	0.07	0.78	85	0.47	3.50	60	0.00	0.00	100	0.31	1.88	65	0.86	3.59	40	0.03	0.36	95	0.21	1.37	60	0.95	3.97	40	0.02	0.35	95
		2	0.09	1.16	90	0.30	2.33	80	0.03	1.16	95	0.11	1.25	90	0.68	3.59	50	0.06	1.32	90	0.49	3.18	40	2.02	19.64	30	0.00	0.00	100
		3	0.04	0.43	85	0.86	4.47	45	0.00	0.00	100	0.23	2.83	80	1.21	4.72	25	0.00	0.00	100	0.20	1.61	65	0.73	2.15	30	0.00	0.00	100
	2	1	0.04	0.39	90	0.91	3.85	40	0.06	0.59	95	0.26	1.70	75	1.16	4.59	45	0.05	0.80	95	0.52	2.16	55	1.41	8.06	15	0.03	0.61	95
		2	0.21	2.78	80	2.74	25.93	40	0.00	0.00	100	0.12	1.24	85	1.91	10.64	35	0.00	0.00	100	0.85	2.81	40	2.80	7.92	10	0.00	0.00	100
		3	0.09	1.66	95	2.04	6.29	40	0.18	2.63	95	0.12	0.93	75	1.97	14.69	30	0.02	0.71	95	0.76	2.26	35	2.70	15.22	30	0.00	0.00	100

As can be observed from Tables 1 and 2, the performance of the algorithm is superb in terms of both solution time and solution quality. All deviations from the optimal solutions are very small and the deviations increase slightly with increases in problem size. We find optimal solutions for all instances with $w=1$. We observe that the instances with $r=1$ and $p=1$ are easier to solve than those in $r=2$ and $p=2$, respectively. The hardest to solve instances are from $n=250$, $r=2$, $p=3$ and $w=3$ combination. Even for this combination the average deviations are 1.69%, 1.61% and 2.1%, for $m=2$, 3 and 4, respectively. Hence we can conclude that our algorithm performs consistently well over all parameter combinations. It can also be observed from the tables that the number of jobs and machines do not have a significant effect on the performance of our algorithm in terms of solution time. We find that almost all CPU times are below 0.01 s even for the largest problem size with 250 jobs and 4 machines. CPLEX solves those problems in about 2–3 s.

In Table 3, we report on the performance of our filtered beam search algorithm for large sized instances with 10 machines and 250 and 500 jobs. Instances with 20 machines are also tried; however the CPLEX software could not find the optimal solutions in one hour for these instances.

As can be observed from Table 3, when $w=1$, all solutions returned by the filtered beam search algorithm are optimal. For $w=2$ or 3, the majority of the average deviations are below 1%. The CPU times for the algorithm are very small as well; the largest average CPU times are 0.03 and 0.07 s, for $n=250$ and $n=500$, respectively. On the other hand, CPLEX cannot solve two problem instances in 3600 s.

5.2. The OFJSS problem

In this section we discuss the performance of our time-phased MCNF algorithm (together with phase 1, i.e., branching phase) developed for the OFJSS problem. Tables 4 and 5 report the average and maximum deviations of our lower bounds from the optimal solutions by the CPLEX software for $r=1$ and $r=2$, respectively. The tables also include the average CPU times. Empty entries in the tables indicate that our algorithm could not return solutions in one-hour.

It can be observed from the tables that our algorithm for the OFJSS problem performs well. The largest average deviations are 1.69% and 1.35% for $r=1$ and $r=2$, respectively. The majority of the average deviations are below 1%. When $m=2$, the algorithm finds optimal solutions for majority of the problem instances and when $m=4$, the algorithm finds optimal solutions for about 60–70% of the instances. The maximum deviations are also very small; hence our algorithm has a consistently well behavior over all problem instances.

Main drawback of the time-phased MCNF algorithm is its solution complexity, which is dramatically affected by the number of machines. We observe that the algorithm cannot solve medium and large sized instances with more than four machines in reasonable time. This is due to the fact that the algorithm performs the branching phase in $O(n^m)$ time, which increases exponentially with the number of machines. This observation is in line with the observations of Eliiyi and Azizoglu (2006) for the B&B method. Note from the tables that when $n=250$, the average CPU times are around 10 s when $m=2$. However, when m becomes 4, the instances could not be solved in one hour.

We also include an experiment using the data used in Rossi et al. (2010). We report the results of our experiment in Tables 6 and 7 for $r=1$ and $r=2$, respectively.

As can be observed from the tables our algorithm performs very well over all problem sizes and parameter combinations. It produces optimal solutions to the majority of the instances when n and m are relatively small. For almost all problem combinations more than half of the solutions are optimal and almost all average deviations are below 1%. The largest average deviation of the genetic algorithm and our algorithm are 0.2% and 1.11%, respectively. Hence genetic algorithm performs slightly better than our algorithm. Our algorithm performs better than greedy heuristic for all problem sizes and parameter combinations. When $r=1$, our average deviations are below 0.93%, 0.7% and 1.11% for $m=2$, 3 and 4, respectively. These respective deviations are 2.86%, 3.01% and 3.01% for the greedy heuristic.

Rossi et al. (2010) compare their solution times with those of the commercial solver XPress MP. The results of their computational study reveal that the solver returns optimal solutions in less than 4 minutes whereas their genetic algorithm runs in about one second.

6. Conclusion

In this study we consider the operational fixed job scheduling problem under working time and spread time constraints. Both problems have important practical implications, and are shown to be NP-hard in the strong sense. We develop several polynomial time heuristics for both problems that employ powerful bounding procedures and reduction properties. An extensive computational study is carried out to observe the performances of the algorithms compared to the optimal solutions.

Our algorithms perform very well for problems up to 100 jobs and 4 machines and 250 jobs and 3 machines, for the OFJSS problem and up to 500 jobs and 10 machines for the OFJSW problem. The computation times do not increase significantly with the problem size, hence the instances with many more jobs and machines could be solved. As the problems have many practical applications and are likely to be solved frequently due to their operational nature, our approximation algorithms may prove quite useful for practitioners.

Our algorithms can be modified to the operational fixed job scheduling problems where working time and spread time constraints are imposed simultaneously.

References

Bekki, B., Azizoglu, M., 2008. Operational fixed ınterval scheduling problem on uniform parallel machines. International Journal of Production Economics 112, 756–768.

Bouzina, K.I., Emmons, H., 1996. Interval scheduling on identical machines. Journal of Global Optimization 9, 379–393.

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

Eliiyi, D.T., Azizoglu, M., 2006. Spread time constraints in operational fixed job scheduling. International Journal of Production Research 44, 4343–4365.

Eliiyi, D.T., Azizoglu, M., 2010. Working time constraints in operational fixed job scheduling. International Journal of Production Research 48, 6211–6233.

Faigle, U., Kern, W., Nawijn, W.M., 1999. A greedy online algorithm for the k-track assignment algorithm. Journal of Algorithms 31, 196–210.

Fischetti, M., Martello, S., Toth, P., 1987. The fixed job schedule problem with spread-time constraints. Operations Research 35, 849–858.

Fischetti, M., Martello, S., Toth, P., 1989. The fixed job schedule problem with working-time constraints. Operations Research 37, 395–403.

Fischetti, M., Martello, S., Toth, P., 1992. Approximation algorithms for fixed job schedule problems. Operations Research 40, S96–S108.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA.

Ghirardi, M., Potts, C.N., 2005. Makespan minimization for scheduling unrelated parallel machines: a recovering beam search approach. European Journal of Operational Research 165, 457–467.

Kolen, A.J.W., Kroon, L.G., 1991. On the computational complexity of (maximum) class scheduling. European Journal of Operational Research 54, 23–38.

Kolen, A.W.J., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.R., 2007. Interval scheduling: a survey. Naval Research Logistics 54, 530–543.

Kovalyov, M.Y., Ng, C.T., Cheng, T.C.E., 2007. Fixed interval scheduling: models, applications, computational complexity and algorithms. European Journal of Operational Research 178, 331–342.
Kroon, L.G., 1990. Job scheduling and capacity planning in aircraft maintenance. Ph.D. Thesis. Rotterdam School of Management. Erasmus University. The Netherlands.
Kroon, L.G., Salomon, M., Van Wassenhove, L.N., 1995. Exact and approximation algorithms for the operational fixed interval scheduling problem. European Journal of Operational Research 82, 190–205.
Martello, S., Toth, P., 1986. A heuristic approach to the bus driver scheduling problem. European Journal of Operational Research 24, 106–117.
Morton, T.E., Pentico, D.W., 1993. Heuristic Scheduling Systems. Wiley, NY.
Ow, P.S., Morton, T.E., 1988. Filtered beam search in scheduling. International Journal of Production Research 26, 35–62.
Rossi, A., Singh, A., Sevaux, M., 2010. A metaheuristic for the fixed job scheduling problem under spread time constraints. Computers and Operations Research 37, 1045–1054.
Solyali, O., Ozpeynirci, O., 2009. Operational fixed job scheduling problem under spread time constraints: a branch-and-price algorithm. International Journal of Production Research 47, 1877–1893.
Spieksma, F.C.R., 1999. On the approximability of an interval scheduling problem. Journal of Scheduling 2, 215–227.
Valente, J.M.S., Alves, R.A.F.S., 2006. Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent set-ups. Computers and Operations Research 35, 2388–2405.
Wolfe, W.J., Sorensen, S.E., 2000. Three scheduling algorithms applied to the earth observing systems domain. Management Science 46, 148–168.