Theory and Methodology

Time lag size in multiple operations flow shop scheduling heuristics

J. Riezebos *, G.J.C. Gaalman

Faculty of Management and Organization, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands

Received 1 May 1996; accepted 1 December 1996

Abstract

This paper considers a multistage flow shop where jobs require multiple operations at each stage and a finish-to-start time lag between any two consecutive operations of a job: the next operation of a job cannot start until the time lag after the former operation of that job has elapsed. The effect of the size of this time lag is considered when studying the effectiveness of solution approaches for this problem. Since the problem of minimizing the makespan is shown to be NP-hard even for the two-stage case, we present a lower bound based heuristic approach that is used to construct several heuristic procedures. These heuristics use lower bounds on the minimum makespan to solve the problem. The effectiveness of these heuristics is empirically evaluated for various time lag sizes by solving a large number of randomly generated problems. We show that the relative performance of the heuristics depends on the size of the time lag. If the ratio of mean time lag and mean processing time is 20% or more, heuristics that construct an active schedule perform less well than heuristics that construct a non-delay schedule. The opposite holds true if this ratio is smaller. The performance of the widely used Shortest Processing Time heuristic (SPT) deteriorates quickly if the size of the time lags increases. We propose instead to use the Earliest Finish Time heuristic (EFT) in case time lags are present. EFT performs much better in this case and is identical to SPT if all time lags are zero. The use of the lower bound based heuristics results in an improvement of the makespan performance of up to 50% as compared with the performance of some simple dispatching heuristics that take the presence of multiple operations and time lags into account. This effect increases with the size of the time lags. © 1998 Elsevier Science B.V.

Keywords: Time lags; Flow shop scheduling; Heuristics; Multiple operations; Lower bounds

1. Introduction

Traditionally, scheduling models assume that each machine can perform only one operation of a job. However, in several practical situations a job has more operations that are to be performed by the same machine. Often, at least a definite amount of time must elapse between the processing of these operations at the machine. This time lag may be required to perform other time-consuming activities at the job. However, these activities, such as refixturing or measuring, will not require any capacity from the machine. During this time lag
the machine may remain idle, but it may also perform operations of other jobs. If during a time lag an operation of another job is started, we say that this time lag is used productively.

If the general flow shop scheduling formulation is applied to this problem, the time between the start of the first operation of the job at this machine and the end of the last operation of this job at the machine is considered to be the processing time of this job at the machine. In this way time lags cannot be used to perform operations of other jobs.

Alternatively, a policy that always uses time lags if there are schedulable operations of other jobs will not necessarily lead to a good solution either. In some cases it can be more profitable to remain idle during the time lag and then proceed with the same job. Note that it will always be profitable to use the time lag for an operation of another job that can finish processing at this machine before the time lag has elapsed.

In this paper we address the effect of using time lags on the makespan performance in a multiple operations flow shop. We consider three reasons to introduce time lags and multiple operations in flow shop scheduling problems. We propose and test several heuristics that consider the presence of time lags when scheduling the operations. By this we determine empirically the sensitivity of their performance for the time lag size.

There are three reasons for considering the presence of time lags due to multiple operations of a job at the same machine. First, a job can be considered as a batch of possibly different parts. The job is completed when all parts are produced. The multiple operations of a job at a machine correspond in this case with processing the various parts at this machine. If the process batch and transfer batch are not equal, these multiple operations are often modelled with start-to-start and finish-to-finish time lags (see Mitten, 1958; Gupta, 1972; and for a generalization Szwarc, 1983). The resulting scheduling problem has only one (pseudo) operation per job per machine and time lags between these operations at the various machines. If all jobs visit the machines in the same order, a flow shop scheduling problem results for which solution techniques exist that generate permutation schedules. Mitten (1958) was the first to work at these problems. He generalized the results of Johnson (1954) and presented a method to generate a permutation schedule for the two-stage flow shop scheduling problem in the presence of start-to-start and finish-to-finish time lags. However, note that the class of permutation schedules need not contain the optimal schedule for this scheduling problem.

The second reason to consider time lags due to multiple operations of a job at a machine can originate from the re-entry of a job at a stage that has already been visited by this job, possibly due to required rework for quality reasons. This type of finish-to-start time lag equals the total processing time required for the operations of this job at the other machines. The re-entrant flow shop is studied by Graves et al. (1983). Lev and Adiri (1984) have shown that these re-entrant flow shops are NP-complete even for the two-stage case. Heuristic procedures for the no-wait case of the two-stage re-entrant flow shop are described by Shapiro (1980).

The third reason for distinguishing multiple operations is studied in this paper. We consider the situation of a machine being capable of performing different operations or a product having to be refigured before processing of the job can continue at the machine. If the machine can perform only one operation at a time, we can model the job as having multiple operations at this machine with some time between these operations. The presence of such a finish-to-start time lag in a flow shop is also considered by Szwarc (1983) and Dell’Amico (1993).

As an illustration of this type of scheduling problem with multiple job operations per machine, consider a Flexible Manufacturing System (FMS) consisting of a number of work stations. Each work station is capable of performing more than one type of operation. Once a job has started at a work station it is taken to the next work station only when all operations at the former are completed. In order to perform two consecutive operations of a job at the same work station, refiguring of the part has to take place in a load/unload station, accompanied by (manual) transportation activities. These activities are required to prepare the job for the next operation. They do not consume any time at the FMS work stations as they are done off-line, but they do take a finite amount of time. This time lag is independent of the job sequence and need not be the same for all operations or jobs. Aanen et al. (1993) described such a flexible manufacturing system with two work-stations and developed heuristic rules to solve the scheduling problem. It is important to incorporate information about the time lags in these heuristics. In Riezebos et al. (1995) we illustrated the losses that are incurred if solution techniques are
applied that do not make use of the time lags between the multiple operations. In that paper we also presented a branch and bound approach to solve the general m-stage flow shop problem with multiple operations and time lags and used examples to illustrate the principles behind various types of lower bounds that can be applied in this approach.

From this survey we conclude that there are various reasons for considering the presence of time lags due to multiple operations of a job at the same machine. The type of flow shop scheduling problem that results varies also, according to the way time lags are used to model the situation.

We know from Szwarz (1983) and Riezebos et al. (1995) that the type of time lag determines the appropriateness of a heuristic for solving the specific flow shop scheduling problem. The objective of the current paper is to study the effect of the size of time lags on the appropriateness of heuristics.

The remainder of the paper is organized as follows. Section 2 defines the minimum makespan scheduling problem explicitly and shows that even the two-stage case is NP-hard. Section 3 describes a lower bound based heuristic approach that is used in Section 4 to construct four heuristics for this problem. This approach applies lower bounds on the makespan to determine the priority between the operations in the schedule. Section 4 also presents five modified flow shop scheduling heuristics. Section 5 deals with the simulation experiments and their results. It uses an ANOVA analysis to describe the effects of varying the size of the time lags on the effectiveness of both the proposed lower bound based heuristics and the modified flow shop scheduling heuristics. Section 6 concludes the paper and addresses promising directions for future research.

2. Problem definition and complexity results

Consider the flow shop scheduling problem with n jobs (i = 1, ..., n) and m machines (j = 1, ..., m). The kth operation of job i at machine j (denoted by O_{ijk}) has processing time T_{ijk}; the number of operations of job i at machine j is K_{ij}, and the total number of operations of job i is K^i. As we have a flow shop, operation O_{ij+1} is preceded by O_{ijk}. In general the predecessor of O_{ijk} is P(O_{ijk}). Operation O_{ijk} cannot start processing until a finish-to-start time lag of TL_{ijk} time units has elapsed since P(O_{ijk}) finished processing. Moreover, O_{ijk} cannot start before machine j has finished processing the preceding operation J(O_{ijk}) at this machine. We want to minimize makespan. F(O_{ijk}) is the completion time of O_{ijk} in a feasible schedule, F(O_{ijk}) = T_{ijk} + \max\{F(P(O_{ijk})), TL_{ijk}, F(J(O_{ijk}))\}. The minimal makespan multiple operations flow shop scheduling problem with time lags is then to determine a feasible schedule that minimizes C_{\max} = \max\{F(O_{1,m_{1,k_{1}}}), ..., F(O_{n,m_{k_{n}}})\}.

Following the standard notations for scheduling problems introduced by Graham et al. (1979), we represent this flow shop problem as a Fm/K^i ≥ m, TL_\epsilon/C_{\max} problem, where K^i ≥ m denotes the existence of multiple operations of a job at one or more machines, and TL_\epsilon the presence of time lags which are associated with operations.

The Fm/K^i ≥ m, TL_\epsilon/C_{\max} problem is NP-hard in the strong sense for all m ≥ 2. This can easily be seen since F/2/K^i = 2, TL_\epsilon/C_{\max} (the two machine flow shop problem with time lags that only vary per job) is NP-hard in the strong sense (prove by reduction to the 3-partition problem (Dell’Amico, 1993)). Kern and Nawijn (1991) already showed that the single facility problem with multiple operations and time lags is NP-hard in the ordinary sense.

In a traditional flow shop problem, the search for an optimal schedule is generally restricted to permutation schedules where the sequence of jobs on all machines is the same. In the Fm/K^i ≥ m, TL_\epsilon/C_{\max} problem it is not possible to consider permutation schedules, as the total number of operations of a job at a stage, K_{ij}, may vary. Therefore, we need to consider non-permutation schedules for the problem, even for the two and three stage problems. The NP-hard nature of these problems makes the search for a constructive algorithm that generates an optimal solution pointless. This brings us to the development of lower bounds that can be applied in a heuristic constructive algorithm as well as heuristic procedures to find feasible solutions for this problem.
3. Lower bound based heuristic approach

In this section we describe an algorithm that constructs heuristics that use a lower bound to find a feasible solution for the general m-stage flow shop scheduling problem with multiple operations and time lags. In Section 4 we apply this approach to generate four complex heuristics that use information on time lags between multiple operations in a flow shop. We will analyse the effect of the time lag size on the quality of the solutions that they obtain.

To construct a heuristic the heuristic constructive algorithm requires the specification of some parameter values, for example the type of lower bound applied at the various levels in the search tree. The greedy algorithm applies a depth first search in the solution tree and uses lower bounds on the makespan to determine the direction of the search. These bounds are computed for all operations that belong to a conflict set. This conflict set contains, for example, all operations that will generate a non-delay schedule (a schedule in which machines will not wait if they can start processing a job) or an active schedule (a schedule where no global left shift is possible, see Baker, 1974). A constructed heuristic terminates as soon as a feasible solution is found. Generally, this solution is not optimal, as obtaining optimality requires tracing back in the search tree as well as the use of the active conflict set. Therefore, the quality of the heuristics obtained with this approach depends on the type of lower bound applied and the restrictiveness of the conflict set. The algorithm is presented below.

3.1. Heuristic constructive algorithm

The algorithm constructs heuristics that augment an operation to a partial schedule \( P_t \), which consists of \( t - 1 \) operations that already have been augmented in earlier levels 1…t – 1 of the search tree (\( P_1 = \emptyset \)). To determine which operation is to be augmented at level \( t \), we compose the set \( S_t \) of schedulable operations. \( S_t \) consists of no more than one operation per job.

\( S_t = \{O_{11}, \ldots, O_{n1}\} \) and \( S_t \) equals \( S_{t-1} \) except that the selected operation for partial schedule \( P_{t-1} \) is eliminated from this set and its successor is added. Next we construct a conflict set \( U_t \), which is a subset of \( S_t \). We consider three possibilities for this conflict set: ‘no restrictions’, ‘active’, and ‘non-delay’. In case time lags exist, the conflict set is determined as:

no restrictions: \( U_t := S_t \)

active: \( M^* := \text{a stage j which can finish at } \Phi^* = \min \{ F(O_{ijk}) ; O_{ijk} \in S_t \} \)
\( U_t := \{ O_{ijk} ; O_{ijk} \in S_t \land j = M^* \land F(O_{ijk}) - T_{ijk} < \Phi^* \} \)

non-delay: \( M^* := \text{a stage j which can start at } \sigma^* = \min \{ F(O_{ijk}) - T_{ijk} ; O_{ijk} \in S_t \} \)
\( U_t := \{ O_{ijk} ; O_{ijk} \in S_t \land j = M^* \land F(O_{ijk}) - T_{ijk} = \sigma^* \} \).

For each operation that belongs to the subset \( U_t \) we compute a lower bound. The computations of the lower bounds for the makespan serve as the basis for selecting an operation from the conflict set. We compute the lower bound for the partial schedule that will result if \( O_{ijk} \) is selected, and this is done for all operations in the conflict set at that level \( t \). The type of lower bound applied at this level needs to be the same for these operations. However, the type of lower bound may vary between the levels in the search tree as specified by the parameter values of the heuristic that is being constructed.

\( \forall O_{ijk} \in U_t \) compute \( LB_t(O_{ijk}) \) for level \( t \) selected lower bound of schedule \( P_t \cup O_{ijk} \)

Select \( O_{ijk} \in U_t \) such that \( LB_t(O_{ijk}) \leq LB_t(O_{ijk}) \) \( \forall O_{ijk} \in U_t \)
Operation $O_{ijk}$ has the smallest lower bound and is augmented to the partial schedule $P_t$. The heuristic proceeds to level $t+1$ in the search tree. If the set $S_{t+1}$ is empty, the resulting schedule is complete and the procedure terminates after computing the makespan.

$$C_{\text{max}} := \max_{i=1,...,n} \{ F(O_{imK_w}) \};$$

Schedule $:= P_{t+1}$

(" end of heuristic constructive algorithm ")

The complexity of the heuristic constructive algorithm depends on the complexity of the lower bound procedure applied. The number of recursions of the algorithm is $K$ (the total number of operations to be scheduled) and the number of lower bound computations in a recursion equals the cardinality of the conflict set $U_i$, which is of order $n$ (the number of jobs). Therefore, the complexity of this algorithm is $o(\log K \cdot n \cdot \log (LB_i))$. The procedure $LB_i$ consists of six steps that are described below. Note that the choice in step 5 of the procedure for lower bound 1 or 2 has to be specified in advance and may depend on $t$.

3.2. Lower bound procedure $LB_i(O_{ij}, j^*, k^*)$

Step 1. Initialize and update.

(In step 1 we proceed to the next level in the search tree, add operation $O_{ij}$ to the partial schedule, determine the new contents of the set $S_t$, and update the finish time of machine $j^*$)

$$t := t + 1 \quad (" t becomes next stage ")$$

$$P_{t} := P_{t-1} \cup O_{ij}$$

$$S_{t} := \{ S_{t-1} \setminus O_{ij} \} \cup O_{ij}, \quad \text{if } k^* < K_{ij}^*,$$

$$S_{t} := \{ S_{t-1} \setminus O_{ij} \} \cup O_{ij} + 1, \quad \text{if } j^* < m \land k^* = K_{ij}^*$$

$$\phi_j := \text{Finish time of machine } j \text{ after processing the (updated) partial schedule, remains unchanged for the machines } j \neq j^*$$

$$\phi_{j^*} := F(O_{ij^*}, j^*) := \max \{ F(RO_{ij^*}, j^*) + TL_{ij^*}, F(J(O_{ij^*}, j^*)) \} + T_{ij}^*, j^*$$

Step 2. Compute start and finish times.

(Step 2 computes the earliest start and finish times of all jobs with schedulable operations)

$$\forall i \mid O_{ijk} \in S_t \text{ BEGIN (" e.g. job } i \text{ requires further processing on the } j \text{th machine ") }$$

$$ES_{ij} := \text{Earliest start time of the first unscheduled operation of job } i \text{ on machine } j$$

$$EF_{ij} := \text{Earliest finish time of job } i \text{ on machine } j$$

$$ES_{ij} := F(O_{ijk}) - T_{ijk}$$

$$EF_{ij} := F(O_{ijk}) + \sum_{k=1}^{K} (TL_{ij} + T_{ij}q)$$

$$ES_{ir} := \max \{ \phi_r, EF_{ir-1} + TL_{ir} \} \quad \forall r = j + 1 \ldots m$$

$$EF_{ir} := ES_{ir} + T_{ir} + \sum_{q=2}^{K} (TL_{iqr} + T_{iqr}) \quad \forall r = j + 1 \ldots m$$

\text{END (" } \forall i \mid O_{ijk} \in S_t \text{ ").}
Step 3. Compute $LB_{mach}$.  

\{Step 3 computes for all machines a lower bound, based on the required operations on this machine and the remaining processing times and time lags on the next machines\}

FOR $r := 1$ TO $m$ DO BEGIN 

\[ J_r := \{ i \mid O_{ijk} \in S_i \land r \geq j \} \text{ with cardinality } |J_r| \]

\[ h_{tr} := k \text{ if } O_{ijk} \in S_i \land r = j \]

\[ = 1 \text{ if } O_{ijk} \in S_i \land r > j \]

Initialize lower bound for machine $r$

\{the final finish time of the job that starts latest at machine $r$ is the initial lower bound\}

Select $i$: $ESt_{tr} \geq ESt_{tr} \forall l \in J_r$ \(* \text{ if there are more jobs } l \in J_r \text{ with maximal } ESt_{tr}, \text{ choose } i \text{ that has maximal } \sum_{p=r+1}^{n} \sum_{q=1}^{K_{ipq}} (TL_{ipq} + T_{ipq}) \*\)

\[ LB_{mach} := EFT_{1,m} \]

Determine lower bound for $r$

\{the total processing time of all future operations at $r$ is used and the minimal remaining time on the next machines is added to this bound. The job set is stepwise reduced, allowing for an improvement of this bound. The time lags at machine $r$ are not worked into this bound, as the machine may process operations of other jobs during these time lags\}

WHILE $|J_r| > 1$ DO BEGIN \(* \text{ if there are more jobs } l \in J_r \text{ with minimal } ESt_{tr}, \) Select $i$: $ESt_{tr} \leq ESt_{tr} \forall l \in J_r$ \text{ choose } i \text{ that has minimal } \sum_{p=r+1}^{n} \sum_{q=1}^{K_{ipq}} (TL_{ipq} + T_{ipq}) \*)

\[ LB_{mach} := \max(LB_{mach}, ESt_{tr} + \sum_{l \in J_r} K_{ipq} h_{lr}, T_{ipq} + \min_{l \in J_r} \{ \sum_{p=r+1}^{n} \sum_{q=1}^{K_{ipq}} (TL_{ipq} + T_{ipq}) \}) \]

\[ J_r := J_r \setminus i \]

END(\(* \text{ WHILE } |J_r| > 1 \*)\)

END(\(* \text{ FOR } r := 1 \text{ TO } m \*)\).

Step 4. Improve $LB_{mach}$; compute earliest start and latest finish times of operations.

\{Step 4–6 improve the lower bound $LB_{mach}$ for a subset $R$ of the machines $r$ that have maximal $LB_{mach}$. A heuristic has to specify the maximum size of $R$ in advance. Step 4 computes the earliest start and latest finish times of all future operations at machine $r$\}

$R \subseteq \{ r \mid LB_{mach} \geq LB_{mach}, j = 1, \ldots, m \}$, \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ |R| \geq 1

\forall r \in R \text{ DO BEGIN}

\[ J_{r} := \{ i \mid O_{ijk} \in S_i \land r \geq j \} \]

Compute 

\[ LFT_{ir} : \text{ Latest finish time of job } i \in J_r \text{ at machine } r \]

\[ LFT_{tr} : \text{ Latest finish time of operation } O_{irq} \]

\[ ESt_{tr} : \text{ Earliest start time of operation } O_{irq} \]
\[\begin{align*}
L_{F_{ir}} & := L_{B_{mach_r}} - \sum_{p=1}^{n} \sum_{q=1}^{K_{ir}} (T_{l_{ipq}} + T_{l_{pq}}) \quad \forall l \in J_r \\
L_{F_{ir}}, & := L_{F_{ir}} \\
L_{F_{irq}} & := L_{F_{ir}} q - T_{l_{irq+1}} - T_{l_{irq+1}} \quad \forall l \in J_r, \ q = h_r \ldots (K_{ir} - 1) \\
E_{S_{ir}}, & := E_{S_{ir}} \\
E_{S_{irq+1}} & := E_{S_{irq}} + T_{l_{irq}} + T_{l_{irq+1}} \quad \forall l \in J_r, \ q = h_r \ldots (K_{ir} - 1).
\end{align*}\]

Step 5.1. Improve \(L_{B_{mach_r}}\): IF \(L_{B_i} = I\) THEN compute lower bound I.

{\textit{Lower bound I sorts operations according to earliest start time with latest finish time as a tie breaker and constructs } } \(|H| \) \textit{sets } \(G_w \subseteq \{H[1] \ldots H[w]\}\text{i.e. set of sorted operations}. \]

\[H := \{H[1], H[2], \ldots, H[w] \text{ of operations } O_{irq}, \ l \in J_r, \ q = h_r \ldots, K_{ir} \text{ such that:} \]

\[E_{S_{H[v]}(w)} \leq E_{S_{H[v+1]}(w)} \land L_{F_{H[v]}(w)} \leq L_{F_{H[v+1]}(w)} \leq \ldots \leq L_{F_{H[w]}(w)} \text{ if } \exists E_{S_{H[w]}(w)} = E_{S_{H[v]}(w)} \]

\[G_w := \{H[v] \mid v \leq w \land E_{S_{H[v]}(w)} = E_{S_{H[w]}(w)} \quad w = 1, \ldots, |H| \}.
\]

Step 5.2. Improve \(L_{B_{mach_r}}\): IF \(L_{B_w} = II\) THEN compute lower bound II.

{\textit{Lower bound II sorts the operations according to latest finish time. } } \(G_w = H[1] \ldots H[w]\)

\[H := H[1], H[2], \ldots, \text{ of operations } O_{irq}, \ l \in J_r, \ q = h_r \ldots, K_{ir} \text{ such that:} \]

\[L_{F_{H[v]}(w)} \leq L_{F_{H[v+1]}(w)} \]

\[G_w := H[1] \ldots H[w] \quad w = 1, \ldots, |H| \] .

Step 6. Improve \(L_{B_{mach_r}}\): compute \(L_{B_{improved_{mach_r}}}\).

\[T_{max} := 0 \text{ ('Initialize Tardiness')} \]

\[\text{\textbf{FOR }} w := 1 \text{ to } |H| \text{ \textbf{DO BEGIN}} \]

\[E_{Start_w} := \min_{O_{irq} \in G} E_{S_{irq}} \]

\[L_{Finish_w} := \max_{O_{irq} \in G} L_{F_{irq}} \]

\[T_{max} := \max[T_{max}, \ E_{Start_w} + \sum_{O_{irq} \in G} T_{irq} - L_{Finish_w}] \]

\[w := w + 1 \]

\[\text{\textbf{END}} \text{ (' \textbf{FOR }} w := 1 \text{ to } |H| \text{'}) \]

\[L_{B_{improved_{mach_r}}} := L_{B_{mach_r}} + T_{max} \]

\[\text{\textbf{END}} \text{ (' } \forall r \in R \text{'}) \]

\text{(' End of Lower bound procedure ')}

In step 1–3 machine-based bounds are computed for all machines that require further processing. The largest of these bounds is improved in step 4–6, where we apply a check on the possibility of scheduling the required operations at this machine within the available time, i.e. between their earliest starting time and latest finish time. We check this by solving a one-machine scheduling problem with release dates and due dates. The set of operations that require scheduling at this machine is sorted, resulting in a list \(H\). The distinction between the two lower bounds I and II is found in the criteria for sorting and in the definition of \(G_w\), a subset of list \(H\). Lower bound I sorts the operations according to earliest release date, using their latest finish times as a tie breaker. Lower bound II sorts all operations according to these latest finish times. In step 6 we compute a lower bound for the tardiness that results if we schedule the operations that belong to the set \(G_w\) within the time frame that is determined by the earliest release date of an operation from this set and the latest finish time of possibly
another operation from this set. The contents of the set \( G_w \) depends on \( w \). In lower bound II the set \( G_w \) consists of the \( w \) operations that are first due. Lower bound I considers smaller subsets of \( H \): all operations that can finish before the earliest starting time of operation \( H[w] \) do not belong to \( G_w \). This can result in higher earliest release dates of these subsets.

The complexity of this procedure is for both lower bounds \( o(K_r \log K_r) \) (\( K_r \) is the number of operations to be processed at machine \( r \), \( K_r \geq n \)), due to sequentially scheduling and sorting \( K_r \) operations. The two lower bounds do not necessarily achieve an optimal solution to this one-machine scheduling problem. The procedure of Carlier (1982) does generate an optimal solution for the one-machine scheduling problem in \( o(K_r \log K_r) \) time. To achieve this it has to consider generally more subsets then \( |H| \), resulting in a higher time constant of the algorithm. We have chosen to apply less time-consuming approximate procedures to obtain good lower bounds on the makespan, but Carliers procedure can be applied in step 5 and 6 of the procedure as well. In Riezebos et al. (1995) we have presented various job-based, machine-based and due date-based lower bounds that can also be used in this algorithm.

4. Multiple operations flow shop scheduling heuristics

In this section we present nine heuristics that will be used in Section 5 to evaluate the effect of time lag size on the makespan performance. The first type of heuristics we test are complex heuristics that are constructed using the lower bound based approach of Section 3. These four heuristics have specially been developed for handling time lags and multiple operations in a flow shop. We are interested in their sensitivity to changes in the size of time lags, e.g., if the relative performance of these heuristics changes with the time lag size. A second type of heuristics is introduced to see if the sensitivity of heuristics to time lag size is specific for these complex lower bound based heuristics or that in another class of heuristics this sensitive can also be found, possibly accompanied with a change in relative performance of these heuristics. As this second class of heuristics we have chosen to take simple heuristics that have only been modified to make them suitable for solving this type of scheduling problem. Note that if the purpose of this paper was to show that one class of heuristics outperforms the other, other more advanced heuristics such as tabu search or genetic algorithms should be considered. We expect that the two classes of heuristics that we study will give us sufficient insight into the effect of time lag size on makespan performance.

The first four heuristics are the complex lower bound based heuristics:

1. **Active single lower bound heuristic (AS).**
   Select a schedulable operation that generates an active schedule, use lower bound I to improve the bound on the first machine \( r \) for which \(LB_{mach_r} \geq LB_{mach_p}, \forall p = 1, \ldots, m\).

2. **Active combi lower bound heuristic (AC).**
   Select a schedulable operation that generates an active schedule, use lower bound I if less than half of the total number of operations is scheduled and otherwise lower bound II, to improve the bound for all machines \( r \) for which \(LB_{mach_r} \geq LB_{mach_p}, \forall p = 1, \ldots, m\).

3. **Non-delay single lower bound heuristic (NDS).**
   Select a schedulable operation that generates a non-delay schedule, use lower bound I to improve the bound on the first machine \( r \) for which \(LB_{mach_r} \geq LB_{mach_p}, \forall p = 1, \ldots, m\).

4. **Non-delay combi lower bound heuristic (NDC).**
   Select a schedulable operation that generates a non-delay schedule, use lower bound I if less than half of the total number of operations is scheduled and otherwise lower bound II, to improve the bound for all machines \( r \) for which \(LB_{mach_r} \geq LB_{mach_p}, \forall p = 1, \ldots, m\).
The next five modified flow shop scheduling heuristics make more or less use of the available information on time lags and multiple operations in the scheduling decisions and are based on simple dispatching rules that are also used in general flow shop scheduling.

The shortest processing time heuristic is a static heuristic and is often used in testing scheduling heuristics. It uses no information about time lags when it selects an operation.

(5) Shortest processing time heuristic (SPT).

Select $O_{ijk}: T_{ijk} \leq T_{ijk}, \quad \forall O_{ijk} \in S_i$.

The dynamic maximal remaining time heuristic considers for each job a lower bound for the remaining time in the shop (a job-based lower bound) and augments the operation of the job with the largest remaining time to the schedule. It includes information about the time lags between the operations and also uses information on operations that are not yet schedulable.

(6) Maximal remaining time heuristic (MRT)

Select $O_{ijk}: \sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} (TL_{ipq} + T_{ipq}) \geq \sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} (TL_{ipq} + T_{ipq}), \quad \forall O_{ijk} \in S_i$.

The earliest finish time heuristic selects the operation that can be completed earliest and it therefore constructs an active schedule. This heuristic restricts itself to information on those operations that are schedulable at level $t$ and to their time lags. Hence, it uses no information about time lags and processing times of operations that are not yet schedulable.

(7) Earliest finish time heuristic (EFT).

Select $O_{ijk}: F(O_{ijk}) \leq F(O_{ijk}), \quad \forall O_{ijk} \in U_i, U_i$ active.

The smallest ratio heuristic and the largest ratio heuristic consider the ratio of remaining processing time and total remaining time for each job with schedulable operations. The smallest ratio heuristic gives priority to jobs with relatively large remaining time lags. The largest ratio heuristic gives priority to jobs with relatively small remaining time lags.

(8) Smallest ratio heuristic (SR)

Select $O_{ijk}: \frac{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} T_{ipq}}{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} (TL_{ipq} + T_{ipq})} \leq \frac{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} T_{ipq}}{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} (TL_{ipq} + T_{ipq})}, \quad \forall O_{ijk} \in S_i$.

(9) Largest ratio heuristic (LR)

Select $O_{ijk}: \frac{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} T_{ipq}}{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} (TL_{ipq} + T_{ipq})} \geq \frac{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} T_{ipq}}{\sum_{p=j}^{m} \sum_{q=h_{ip}}^{K_{ip}} (TL_{ipq} + T_{ipq})}, \quad \forall O_{ijk} \in S_i$.

Note that the heuristics SPT and EFT produce the same result if the remaining time lags of the schedulable operations are zero. If all time lags are zero, these heuristics are identical.
5. Effect of time lag size on performance of heuristics

Using simulation, we have analysed the nine heuristics for their effectiveness in producing tight schedules in situations with various time lag sizes. In this section the characteristics of these situations are described. Section 5.2 describes the effect of a reduction in the size of the time lags on the performance of the four complex lower bound based heuristics. Section 5.3 does the same for the five modified flow shop scheduling heuristics. We present the results of the two types of heuristics in subsequent sections to stress that comparing both types of heuristics is not the objective of this paper, as it is more important to show for both classes the existence of a significant time lag size effect and the consequences this has for the relative performance of the heuristics in the same class.

5.1. Experimental design

In flow shop scheduling research generally the number of jobs and machines is varied when testing the performance of heuristics. These factors determine the total number of operations that have to be scheduled, unless situations with multiple operations of a job at a machine are considered. In that case the total number of operations can better be varied directly. We consider the total number of operations that have to be scheduled and the distribution of these operations over machines and jobs as two main factors in the experimental design. The next design parameters were used to generate problems:

\[ K = 20,40,60,\ldots,200 \]  \hspace{1cm} \text{(total number of operations)}; \n
\[ K_{ij} \geq 2 \]  \hspace{1cm} \text{(number of operations of a job at a machine)}; \n
\[ n = 5,10,15 \]  \hspace{1cm} \text{(number of jobs) \hspace{0.2cm} (K > 100: n = 15)}; \n
\[ m = 2,4,6,8,10 \]  \hspace{1cm} \text{(number of machines)}; \n
\[ O = 0,1 \]  \hspace{1cm} \text{(0: no operation variation);} \n
\[ B = 0,1 \]  \hspace{1cm} \text{(0: no time variability; \hspace{0.2cm} T_{ijk} \approx 10 \cdot U[3,12]; \hspace{0.2cm} TL_{ijk} \approx (10 - t) \cdot U[3,12])}; \n
\[ (1: \text{varying number of operations per job and machine}); \n
\[ (1: \text{time variability:} \hspace{0.2cm} T_{ijk} \approx 10 \cdot U[6,12]; \hspace{0.2cm} TL_{ijk} \approx (10 - t) \cdot U[2,6])]. \n
We tested the heuristics on problems with a varying number \( K \) of operations, for which we selected appropriate combinations of the number of jobs and machines. We generated problems \( O = 0 \) where each job has the same number of operations, equally divided between the machines, as is done in general flow shop scheduling, and problems \( O = 1 \) that have a varying number of operations per job and per machine. The maximum number of operations of a job at a machine varied between 5, 10 and 15. We expected the performance of the heuristics to depend on the presence of this operation variation, as the choice of which operation to augment first to the schedule becomes more critical in case \( O = 1 \).

Each problem configuration is solved with two classes of time lag and processing time distributions. In the first class (\( B = 0 \)), initial time lags and processing times are both generated from a discrete uniform distribution in the range \([3,12]\) and afterwards multiplied by 10. The second class (\( B = 1 \)) considers problems with a lower variety in time lags and processing times, while at the same time the processing times are never smaller than the time lags. We modelled this by generating initial time lags from \( U[2,6] \) and processing times from \( U[6,12] \), which were also multiplied by 10. The factor 10 was due to programming reasons.

We are interested in the sensitivity of the performance of the heuristics in these scenarios with respect to the size of the time lags, i.e. we want to test whether the applicability of these heuristics changes if time lags decrease. How robust is the relative performance of these heuristics if they are applied to the same problems, with only the time lags being smaller? To study this research question, we solved each problem 10 times \((t = 0,\ldots,9)\) with all heuristics, each time reducing the time lags by 10% of the original value (i.e., increasing \( t \) with 1).
For each of the 96 scenarios that resulted \((s = 1, \ldots, 96)\) we randomly generated 100 problems \((i = 1, \ldots, 100)\). All nine heuristics \((j = 1, \ldots, 9)\) were applied to each problem. The experiments resulted in \(96(s) \cdot 100(i) \cdot 9(j) \cdot 10(t) = 864,000\) schedules with makespans \(C_{\text{max}}\). The mean makespan for the performance of heuristic \(j\) on scenario \(s\) with time lag reduction \(t \cdot 10\%\) is denoted by \(C_{m,ij}\). The proposed heuristics were programmed in Borland Pascal 7.0 and the computations were performed on a Pentium 100 MHz machine. Appendix A presents and discusses the required CPU times (in seconds) for the proposed heuristics. In our experiments these were between 0.0005 and 2.4 seconds.

5.2. Effect of time lag reduction on lower bound based heuristics

Table 1 shows the results of the four lower bound based heuristics AS, AC, NDS and NDC. We present their results for all combinations of the parameters \(K\) (total number of operations in the schedule) and \(B\) (time variability) that appeared in the scenarios. We have computed for each combination of parameters \((K,B)\) the mean makespan \(C_m\), that is the mean of all \(C_{m,ij}\) with \(s \in (K,B)\) and we present the results for the situations both with no time lag reduction \((t = 0)\) and a maximal time lag reduction \((t = 9)\).

There are significant differences in the performance of the heuristics on these problems. In case \(t = 0\) (no time lag reduction) the heuristics AS and AC always perform less well than the heuristics NDS and NDC, which apply non-delay scheduling. This difference in makespan performance between the active and non-delay based heuristics increases with the size of the problem, i.e. the number of operations. The differences due to the use of either a single lower bound heuristic (AS or NDS) or a combination of two lower bounds (AC or NDC) are small. The combi lower bound heuristics slightly outperform the single lower bound heuristics. In scenarios with low variation in processing times and time lags \((B = 1)\), the NDS heuristic generally outperforms the NDC heuristic, while the opposite holds true in case \(B = 0\).

Table 1
Mean makespan of lower bound based heuristics, time lag reduction \(t = 0\) vs. \(t = 9\)

<table>
<thead>
<tr>
<th>(K)</th>
<th>(B)</th>
<th>1 AS</th>
<th>2 AC</th>
<th>3 NDS</th>
<th>4 NDC</th>
<th>1 AS</th>
<th>2 AC</th>
<th>3 NDS</th>
<th>4 NDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>1198.6</td>
<td>1199.3</td>
<td>1197.8</td>
<td>1193.5</td>
<td>969.4</td>
<td>978.2</td>
<td>1000.4</td>
<td>1007.8</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1289.9</td>
<td>1293.3</td>
<td>1270.0</td>
<td>1272.4</td>
<td>1124.3</td>
<td>1144.7</td>
<td>1181.8</td>
<td>1185.1</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>2155.6</td>
<td>2152.3</td>
<td>2126.7</td>
<td>2122.6</td>
<td>1695.8</td>
<td>1706.9</td>
<td>1767.4</td>
<td>1774.4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2306.9</td>
<td>2305.8</td>
<td>2270.2</td>
<td>2277.5</td>
<td>1981.5</td>
<td>1998.8</td>
<td>2096.0</td>
<td>2099.5</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>3109.1</td>
<td>3087.1</td>
<td>3069.8</td>
<td>3050.3</td>
<td>2464.9</td>
<td>2470.8</td>
<td>2544.2</td>
<td>2556.6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3318.7</td>
<td>3317.2</td>
<td>3248.3</td>
<td>3247.6</td>
<td>2852.4</td>
<td>2871.6</td>
<td>2990.8</td>
<td>3000.4</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>3917.9</td>
<td>3886.1</td>
<td>3839.1</td>
<td>3819.6</td>
<td>3020.1</td>
<td>3020.2</td>
<td>3102.9</td>
<td>3111.0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4136.1</td>
<td>4125.3</td>
<td>4008.7</td>
<td>4011.5</td>
<td>3468.5</td>
<td>3482.6</td>
<td>3649.7</td>
<td>3658.9</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>5349.7</td>
<td>5279.5</td>
<td>5191.5</td>
<td>5159.6</td>
<td>4046.5</td>
<td>4017.8</td>
<td>4160.0</td>
<td>4174.0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5467.8</td>
<td>5407.5</td>
<td>5253.8</td>
<td>5251.1</td>
<td>4477.6</td>
<td>4463.1</td>
<td>4647.5</td>
<td>4662.2</td>
</tr>
<tr>
<td>120</td>
<td>0</td>
<td>5136.8</td>
<td>5082.1</td>
<td>5049.7</td>
<td>4989.7</td>
<td>4497.4</td>
<td>4540.9</td>
<td>4548.7</td>
<td>4572.7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5867.1</td>
<td>5915.6</td>
<td>5623.4</td>
<td>5628.0</td>
<td>5250.7</td>
<td>5302.3</td>
<td>5385.3</td>
<td>5401.2</td>
</tr>
<tr>
<td>140</td>
<td>0</td>
<td>4487.1</td>
<td>4522.5</td>
<td>4394.6</td>
<td>4397.7</td>
<td>3699.1</td>
<td>3757.8</td>
<td>3796.2</td>
<td>3816.1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4964.3</td>
<td>5064.2</td>
<td>4810.4</td>
<td>4837.6</td>
<td>4332.3</td>
<td>4425.0</td>
<td>4520.7</td>
<td>4570.6</td>
</tr>
<tr>
<td>160</td>
<td>0</td>
<td>6190.3</td>
<td>6093.8</td>
<td>6036.3</td>
<td>5975.6</td>
<td>5259.0</td>
<td>5319.3</td>
<td>5309.7</td>
<td>5351.4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6954.5</td>
<td>7033.3</td>
<td>6695.4</td>
<td>6711.7</td>
<td>6189.1</td>
<td>6255.0</td>
<td>6351.9</td>
<td>6420.8</td>
</tr>
<tr>
<td>180</td>
<td>0</td>
<td>6371.8</td>
<td>6324.7</td>
<td>6192.8</td>
<td>6153.8</td>
<td>5325.7</td>
<td>5370.3</td>
<td>5408.0</td>
<td>5449.3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7034.4</td>
<td>7124.9</td>
<td>6750.8</td>
<td>6773.8</td>
<td>6171.2</td>
<td>6226.0</td>
<td>6381.4</td>
<td>6413.8</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>6754.8</td>
<td>6718.1</td>
<td>6550.6</td>
<td>6512.3</td>
<td>5598.7</td>
<td>5662.3</td>
<td>5685.1</td>
<td>5727.5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7484.6</td>
<td>7570.8</td>
<td>7173.9</td>
<td>7195.6</td>
<td>6527.1</td>
<td>6602.4</td>
<td>6750.2</td>
<td>6800.0</td>
</tr>
</tbody>
</table>

\(K = \) total number of operations in the schedule.
\(B = 0:\) \(T_{ij} = 10 \cdot U[3,12], T_{ij} = (10-i) \cdot U[3,12];\)
\(B = 1:\) \(T_{ij} = 10 \cdot U[6,12], T_{ij} = (10-i) \cdot U[2,6].\)
Table 2
ANOVA analysis on lower bound based heuristics and time lag reduction factor

<table>
<thead>
<tr>
<th>Main effect of J and T (βj and βi)</th>
<th>J = 1 AS</th>
<th>J = 2 AC</th>
<th>J = 3 NDS</th>
<th>J = 4 NDC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>βj = 30.7</td>
<td>βj = 30.6</td>
<td>βj = −35.0</td>
<td>βj = −26.2</td>
</tr>
<tr>
<td>T</td>
<td>βi</td>
<td>Interaction effect of J and T (γji)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>332.6</td>
<td>56.2</td>
<td>35.2</td>
<td>−35.0</td>
</tr>
<tr>
<td>1</td>
<td>272.6</td>
<td>54.3</td>
<td>36.4</td>
<td>−38.3</td>
</tr>
<tr>
<td>2</td>
<td>200.6</td>
<td>43.2</td>
<td>34.0</td>
<td>−34.3</td>
</tr>
<tr>
<td>3</td>
<td>130.0</td>
<td>33.2</td>
<td>28.9</td>
<td>−29.2</td>
</tr>
<tr>
<td>4</td>
<td>54.9</td>
<td>22.3</td>
<td>21.5</td>
<td>−22.2</td>
</tr>
<tr>
<td>5</td>
<td>−25.1</td>
<td>5.9 *</td>
<td>12.2 *</td>
<td>−11.2 *</td>
</tr>
<tr>
<td>6</td>
<td>−103.4</td>
<td>−11.1 *</td>
<td>−3.0 *</td>
<td>2.7 *</td>
</tr>
<tr>
<td>7</td>
<td>−191.2</td>
<td>−35.0</td>
<td>−25.5</td>
<td>25.2</td>
</tr>
<tr>
<td>8</td>
<td>−284.4</td>
<td>−64.8</td>
<td>−51.9</td>
<td>52.4</td>
</tr>
<tr>
<td>9</td>
<td>−386.7</td>
<td>−104.2</td>
<td>−87.8</td>
<td>89.9</td>
</tr>
</tbody>
</table>

Coefficients marked with * are not statistically significant at a 5% level.

There is a remarkable difference in the relative performance of the heuristics with respect to the size of the time lag reduction. When the time lags are reduced to 90% of their initial values (t = 9), the active based heuristics AS and AC outperform the non-delay based heuristics NDS and NDC completely. Furthermore, the single lower bound heuristics dominate the combi lower bound heuristic in almost all cases.

At what time lag reduction percentage does this shift in relative performance of these two types of heuristics take place? Information on this can be used to specify the preference ranges for applying either a non-delay or an active based heuristic to problems with time lags.

To detect where a shift in the relative performance of the heuristics takes place, we have applied an ANOVA analysis. Analysis of variance gives insight whether the means of subsets of makespans into which the total set of makespans are broken are significantly different from each other. The total variance in the Cmax,ij, is first explained by the covariates (K, O, B) that together describe the scenario effect s. These factors do have an important influence on the length of the makespan, and therefore it is necessary to determine their effect on the makespan. The remaining variance is decomposed over the main factors j (type of heuristic applied, parameters βj) and t (size of time lag reduction, parameters βj) and their interaction effect γji. ANOVA determines the mean of these subsets and tests if the deviation from the overall mean is significant. It is therefore very useful in determining the region of time lag sizes where a shift in performance of the two types of heuristics takes place.

We have applied ANOVA on the following model:

\[ C_{\text{max,ij}} = [\delta_1 K + \delta_2 O + \delta_3 B] + \beta_j J + \beta_i T + \gamma_{ji} J \cdot T + \epsilon_{ij}, \]

The results of the analysis are presented in Table 2.

The main effect T shows the reduction in the mean makespan caused by the reduction of the time lags. The effect on the mean makespan of a reduction of the time lags by 10% can be estimated to be between 1.2%–2.4%. The main effect J shows that the differences between the single bound heuristics and their accompanying combi bound heuristics are small (30.7/30.6; −35.0/−26.2) compared with the differences between the active based and non-delay based heuristics. The non-delay lower bound heuristics have the best overall performance.

The interaction effect shows that a significant shift in the relative performance of the heuristics takes place. The sign of the coefficient indicates the direction of the effect on the mean makespan: a negative sign corresponds with a lower mean makespan. For reduction percentages in the range of 0 to 40%, the γ coefficients of the non-delay based heuristics have a negative sign, while the active based heuristics have a positive sign. However, when time lags are reduced to 70% or more, the opposite holds true. This shows that the
performance of the non-delay based heuristics deteriorates as the time lags become smaller. To see at which time lag reduction percentages one type of heuristics really outperforms the other, we have to look at the combined effect of heuristics and time lag reduction. The following results:

\[
\beta_j + \gamma_k \geq \beta_k + \gamma_k, \quad \forall j = 1, 2 \text{ (active)}; \forall k = 3, 4 \text{ (non-delay) and } t \leq 6,
\]

\[
\beta_j + \gamma_k \leq \beta_k + \gamma_k, \quad \forall j = 1, 2 \text{ (active)}; \forall k = 3, 4 \text{ (non-delay) and } t \geq 8.
\]

The shift in performance takes place between a time lag reduction of 60%–80%. At what time lag reduction percentages the shift in performance of these types of heuristics takes place depends on \( B \), the time variability in the scenario. Fig. 1 shows this shift in performance for scenarios with \( B = 0 \) (high variation, equally distributed processing times and time lags) and \( B = 1 \) (smaller time lags, larger processing times, less variation). \( B = 0 \) scenarios show the shift at a 80% reduction, \( B = 1 \) scenarios at a 60% reduction. Note that the ratio of mean time lags to mean processing times is at the reduction percentages given in both situations about 20%.

Fig. 1 illustrates the deterioration of the dominance of the non-delay based heuristics over the active based heuristics in case reduction percentages become larger. The horizontal axis shows the reduction in time lags \((t \cdot 10\%)\). The vertical axis gives the percentage of scenarios that showed a continuing significantly better performance for one type of heuristic. The two curves that describe the dominance of the non-delay based heuristic in case \( B = 0 \) and \( B = 1 \) present the percentage of scenarios that showed this dominance for all time lag reduction percentages from 0 to \( t \). The other two curves describe the dominance of the active based heuristic for all time lag reduction percentages from 90% down to \( t \).

From Fig. 1 we conclude that the deterioration of the non-delay based heuristics starts earlier in case \( B = 1 \), accompanied by a more prevalent dominance of the active based heuristics. The pattern is also obvious if \( B = 0 \). Application of active based heuristics is hence preferred only in situations where the ratio of time lags and processing times is 20% or lower.

What explanation can we give for the differences in the performance of the active and non-delay based heuristics? From the theory on scheduling we know that the schedule with minimal makespan belongs to the set of active schedules. This set contains many schedules with enforced delays, but the probability that the optimal solution is a schedule with such enforced delays is not high. The set of non-delay schedules is a subset of the set
of active schedules and contains no schedules with enforced delays between operations. The set of non-delay schedules does not necessarily include the optimal schedule (Baker, 1974).

The active based heuristics apply the same lower bounds to a larger conflict set of schedulable operations as compared with the non-delay based heuristics. Hence, the active based heuristics may give priority to a schedulable operation that starts later than the earliest possible starting time of one of the other schedulable operations. An operation that is active but not non-delay can only be given priority by the active based heuristic if the computed lower bound for this operation is not larger than the lower bounds obtained for the other operations, among which the non-delay schedulable operations. The significant difference in performance between the two types of heuristics found in our experiments indicates that it is not a matter of coincidence that a delay operation is selected by the active based heuristics.

Each time an operation has to be added to the partial schedule, all the operations from the conflict set are examined by computing their lower bounds. The lower bound computation solves a relaxation of the remaining scheduling problem, so an optimal solution to this remaining scheduling problem is not guaranteed. Selecting the operation with the lowest lower bound therefore need not result in an optimal schedule.

Active based heuristics only give priority to an operation that enforces a delay if it has the lowest lower bound. This may be the best choice, as the optimal solution can possess such a delay. However, in case time lags are not reduced our results show that this often only seems to result in a better solution while in the end turning out to be worse. For, the non-delay based heuristics finally outperform the active based heuristics in that case, even while their set of operations is just smaller and only contains operations that can also be selected by the active based heuristics. The remarkable result that the non-delay based heuristics finally outperform the active based heuristics is possibly due to the fact that active based heuristics can give priority to the next operation of the same job. Non-delay based heuristics give priority to the job that can start as soon as possible, and can therefore only give priority to the next operation of a job if no operation of another job can be started before at this machine. Selecting the next operation of a job might influence the cardinality of the conflict set in the succeeding nodes of the search tree such that the number of alternative operations decreases. This may be a reason why a non-delay policy often yields a more balanced schedule that allows the productive use of time lags in the remaining part of the schedule, and this might result in a shorter makespan.

In case time lags are very small, it may be more efficient to wait until the next operation of the current job can start than to give priority to another job that can start immediately. This may be a reason why in that situation active based heuristics perform better than non-delay based heuristics. Appendix B illustrates this for an example problem. The heuristics AC and NDC were used to solve this problem for no reduction and maximal reduction of time lags. The Gantt charts show that AC does indeed give priority to the next operation of the same job. More research on the theoretical aspects of this phenomenon is required. We expect that more variety in the number of remaining operations of a job at a machine will make this effect more obvious. Then, non-delay heuristics give priority to jobs with less remaining operations instead of allowing some idle time and proceeding with the longer job.

5.3. Effect of time lag reduction on modified flow shop scheduling heuristics

We present the results of the five modified flow shop scheduling heuristics in a similar way to Section 5.2. Table 3 shows the mean make spans obtained by these heuristics for the various scenarios. Note that these heuristics were applied to exactly the same problems as the four lower bound based heuristics had solved. Therefore, studying the differences between the results in Tables 1 and 3 gives an impression of the difference in performance of the complex and simple heuristics. The makespan performance of the lower bound based heuristics is much better than the performance of the modified flow shop scheduling heuristics, as was expected. The use of a lower bound based heuristic leads in some scenarios to an improvement of more than 50% in the mean makespan of the schedules obtained by the heuristics 5–9. This effect is more prevalent in case the total number of operations that have to be scheduled is high, and in case the size of the time lag is high.

Of the modified flow shop scheduling heuristics the EFT heuristic performs best. Note that this heuristic
Table 3
Mean makespan of modified flow shop scheduling heuristics, reduction $t = 0$ vs. $t = 9$

<table>
<thead>
<tr>
<th>$K$</th>
<th>$B$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
<th>$8$</th>
<th>$9$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
<th>$8$</th>
<th>$9$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SPT</td>
<td>MRT</td>
<td>EFT</td>
<td>SR</td>
<td>LR</td>
<td>SPT</td>
<td>MRT</td>
<td>EFT</td>
<td>SR</td>
<td>LR</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1666.4</td>
<td>1460.5</td>
<td>1268.2</td>
<td>1612.6</td>
<td>1626.5</td>
<td>1159.6</td>
<td>1292.2</td>
<td>1027.2</td>
<td>1309.0</td>
<td>1045.3</td>
</tr>
<tr>
<td>1</td>
<td>1626.0</td>
<td>1633.3</td>
<td>1531.2</td>
<td>1688.3</td>
<td>1490.1</td>
<td></td>
<td>1348.7</td>
<td>1564.7</td>
<td>1209.9</td>
<td>1556.3</td>
<td>1191.2</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>3150.6</td>
<td>2711.4</td>
<td>2261.9</td>
<td>3035.0</td>
<td>3005.3</td>
<td>2102.7</td>
<td>2467.5</td>
<td>1855.6</td>
<td>2391.1</td>
<td>1893.4</td>
</tr>
<tr>
<td>1</td>
<td>3066.2</td>
<td>3110.6</td>
<td>2417.0</td>
<td>3137.9</td>
<td>2724.0</td>
<td></td>
<td>2446.7</td>
<td>2997.2</td>
<td>2180.2</td>
<td>2826.0</td>
<td>2134.6</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>4669.4</td>
<td>3832.8</td>
<td>3246.9</td>
<td>4394.7</td>
<td>4452.6</td>
<td>3035.9</td>
<td>3461.8</td>
<td>2667.2</td>
<td>3359.6</td>
<td>2758.9</td>
</tr>
<tr>
<td>1</td>
<td>4370.0</td>
<td>4351.6</td>
<td>3436.3</td>
<td>4478.2</td>
<td>3987.8</td>
<td></td>
<td>3498.2</td>
<td>4187.5</td>
<td>3120.3</td>
<td>3973.9</td>
<td>3088.1</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>5939.6</td>
<td>4921.8</td>
<td>4088.9</td>
<td>5573.2</td>
<td>5548.7</td>
<td>3755.6</td>
<td>4498.7</td>
<td>3310.6</td>
<td>4129.8</td>
<td>3363.6</td>
</tr>
<tr>
<td>1</td>
<td>5441.7</td>
<td>5655.4</td>
<td>4314.6</td>
<td>5587.4</td>
<td>4938.1</td>
<td></td>
<td>4291.2</td>
<td>5440.5</td>
<td>3837.5</td>
<td>4881.8</td>
<td>3790.6</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>7868.7</td>
<td>6486.0</td>
<td>5491.5</td>
<td>7566.4</td>
<td>7341.0</td>
<td>4957.1</td>
<td>5898.7</td>
<td>4411.1</td>
<td>5429.9</td>
<td>4475.3</td>
</tr>
<tr>
<td>1</td>
<td>7226.6</td>
<td>6937.4</td>
<td>5629.3</td>
<td>7254.0</td>
<td>6697.1</td>
<td></td>
<td>5402.5</td>
<td>6606.4</td>
<td>4903.8</td>
<td>5984.9</td>
<td>4895.4</td>
</tr>
<tr>
<td>120</td>
<td>0</td>
<td>8867.3</td>
<td>7172.7</td>
<td>5485.1</td>
<td>8015.1</td>
<td>8186.3</td>
<td>5838.7</td>
<td>6866.1</td>
<td>4969.3</td>
<td>6466.1</td>
<td>5122.3</td>
</tr>
<tr>
<td>1</td>
<td>8464.9</td>
<td>8459.8</td>
<td>6184.3</td>
<td>8452.9</td>
<td>7493.8</td>
<td></td>
<td>6830.5</td>
<td>8310.3</td>
<td>5861.1</td>
<td>7738.4</td>
<td>5866.8</td>
</tr>
<tr>
<td>140</td>
<td>0</td>
<td>7476.3</td>
<td>7219.9</td>
<td>4697.5</td>
<td>7453.7</td>
<td>6649.1</td>
<td>5076.8</td>
<td>6816.4</td>
<td>4066.3</td>
<td>6093.8</td>
<td>4295.8</td>
</tr>
<tr>
<td>1</td>
<td>7186.1</td>
<td>8446.1</td>
<td>5226.5</td>
<td>8059.3</td>
<td>6109.5</td>
<td></td>
<td>5903.3</td>
<td>8262.1</td>
<td>4809.1</td>
<td>7416.3</td>
<td>4869.1</td>
</tr>
<tr>
<td>160</td>
<td>0</td>
<td>10880.4</td>
<td>8291.9</td>
<td>6576.2</td>
<td>9881.6</td>
<td>10067.7</td>
<td>6961.3</td>
<td>7898.7</td>
<td>5821.6</td>
<td>7680.4</td>
<td>6173.7</td>
</tr>
<tr>
<td>1</td>
<td>10141.0</td>
<td>9728.8</td>
<td>7244.7</td>
<td>10217.4</td>
<td>9100.3</td>
<td></td>
<td>8067.1</td>
<td>9533.9</td>
<td>6816.9</td>
<td>9164.3</td>
<td>7039.3</td>
</tr>
<tr>
<td>180</td>
<td>0</td>
<td>11168.8</td>
<td>9362.1</td>
<td>6768.3</td>
<td>10349.5</td>
<td>10142.2</td>
<td>7170.4</td>
<td>8897.3</td>
<td>5920.8</td>
<td>8022.0</td>
<td>6266.8</td>
</tr>
<tr>
<td>1</td>
<td>10378.9</td>
<td>11090.1</td>
<td>7429.2</td>
<td>10687.4</td>
<td>9108.5</td>
<td></td>
<td>8242.6</td>
<td>10835.1</td>
<td>6925.8</td>
<td>9553.6</td>
<td>7057.0</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>11916.9</td>
<td>10052.2</td>
<td>7168.6</td>
<td>11204.8</td>
<td>10774.9</td>
<td>7597.2</td>
<td>9562.6</td>
<td>6219.7</td>
<td>8574.7</td>
<td>6612.0</td>
</tr>
<tr>
<td>1</td>
<td>11022.2</td>
<td>11810.2</td>
<td>7909.6</td>
<td>11401.3</td>
<td>9684.0</td>
<td></td>
<td>8718.0</td>
<td>11555.3</td>
<td>7370.5</td>
<td>10118.7</td>
<td>7480.6</td>
</tr>
</tbody>
</table>

$K =$ total number of operations in the schedule.
$B = 0: T_{ijk} = 10 \cdot U[3,12], \quad T_{ijkt} = (10 - t) \cdot U[3,12];$
$B = 1: T_{ijk} = 10 \cdot U[6,12], \quad T_{ijkt} = (10 - t) \cdot U[2,6].$

generates an active schedule, while the others do not necessarily do so. The MRT heuristic performs well for small problems, but its performance deteriorates quickly for larger sized problems. The opposite holds true for the LR heuristic. The heuristics SPT and SR have a comparable performance for the situation with no time lag reduction.

The effect of reducing time lags on the makespan performance is analysed with ANOVA. Table 4 presents the results of applying the model of Section 5.2 to the heuristics 5–9. It shows that the main effects of both the type of heuristic applied and the size of the time lag reduction are statistically significant as well as their

Table 4
ANOVA analysis on modified flow shop heuristics and time lag reduction factor

<table>
<thead>
<tr>
<th>Main effect of $J$ and $T$ ($\beta_j$ and $\beta_t$)</th>
<th>$J = 5$ SPT</th>
<th>$J = 6$ MRT</th>
<th>$J = 7$ EFT</th>
<th>$J = 8$ SR</th>
<th>$J = 9$ LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J = 5$ SPT</td>
<td>$\beta_j = 291.5$</td>
<td>$\beta_j = 589.3$</td>
<td>$\beta_j = -1084.4$</td>
<td>$\beta_j = 539.8$</td>
<td>$\beta_j = -336.1$</td>
</tr>
<tr>
<td>$J = 6$ MRT</td>
<td>$\beta_j = -491.5$</td>
<td>$\beta_j = -383.7$</td>
<td>$\beta_j = -166.5$</td>
<td>$\beta_j = -94.5$</td>
<td>$\beta_j = -394.5$</td>
</tr>
<tr>
<td>$J = 7$ EFT</td>
<td>$\beta_j = -460.4$</td>
<td>$\beta_j = -309.7$</td>
<td>$\beta_j = -112.6$</td>
<td>$\beta_j = -8.8$</td>
<td>$\beta_j = 37.3$</td>
</tr>
<tr>
<td>$J = 8$ SR</td>
<td>$\beta_j = 34.9$</td>
<td>$\beta_j = 24.5$</td>
<td>$\beta_j = 13.3$</td>
<td>$\beta_j = 32.6$</td>
<td></td>
</tr>
<tr>
<td>$J = 9$ LR</td>
<td>$\beta_j = 329.2$</td>
<td>$\beta_j = 329.2$</td>
<td>$\beta_j = 232.6$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Coefficients marked with ' *' were not statistically significant at a 5% level.
interaction effect. The heuristics do not only differ in the overall makespan performance, but also in the sensitivity for reducing the time lags.

The overall performance of the heuristics MRT and EFT is good, but it is relatively less good in case of small time lags. The performance of the SR heuristic hardly varies with the size of the time lag reduction. However, the heuristics LR and SPT show a relative performance that deteriorates when the time lags become larger.

From the literature on flow shop scheduling the SPT heuristic is known to be a robust heuristic with respect to makespan performance. In our scheduling problem, the quality of the SPT schedules depends heavily on the size of the time lags. If the mean size of the time lags and processing times are of the same order, the performance of SPT is worse than any other heuristic. LR and EFT strictly dominate SPT at all time lag reduction percentages.

The combined effect of the type of heuristic, the reduction of the time lags and their interaction is graphically depicted in Fig. 2. The vertical axis shows the difference between the mean makespan obtained by heuristic $j$ at time lag reduction percentage $t \cdot 10\%$ and the overall mean makespan obtained by all five heuristics and all time lag sizes. It illustrates clearly the change in relative performance of the heuristics due to the size of the time lag.

6. Conclusions

In this paper we have discussed the flow shop scheduling problem with multiple operations and time lags. The inherent structure of this flow shop was used in the development of a lower bound based heuristic approach, which includes two lower bounds on the makespan. This heuristic approach was used to construct four lower bound based heuristics for this problem. These heuristics were compared with other heuristics that incorporate information on time lags and multiple operations. The computational results indicate that heuristics that are based on the lower bounds that we developed perform much better than the other heuristics. The improvement in the mean makespan that we found in our experiments is up to 50%.
The four lower bound based heuristics are distinguished by the type of schedule that they obtain (non-delay or active) and by the lower bound that they apply. We found that a shift in performance between the active based heuristics and the non-delay based heuristics takes place when the ratio of mean time lags and mean processing times is about 20%. A smaller ratio (i.e. smaller mean time lags) favours the application of active based heuristics, at a larger ratio the non-delay based heuristics have a better performance. This indicates the importance of comparing the size of the time lags with the processing times when selecting a heuristic.

The effect of the size of the time lag on the performance of heuristics is also shown for simple modified flow shop scheduling heuristics. The five heuristics that we tested showed significant differences in performance with respect to the size of the problem and the size of the time lag reduction. In our experiments, the EFT heuristic performed best of these five heuristics. The performance of the SPT heuristic deteriorated in case time lags become larger. We propose to use the EFT heuristic instead of the SPT heuristic in case time lags have to be considered in the multiple operations flow shop scheduling problem.

Future research must further investigate the consequences of the size and variability of time lags on the applicability of heuristics for flow shop scheduling with time lags. It would be interesting to work on the possibility of dealing with time lags in efficient and advanced problem solving approaches such as genetic algorithms and tabu search and compare their effectiveness and sensitivity to time lag size with the four lower bound based heuristics that are tested in this paper. The effect of applying improved lower bounds in the construction of heuristics should also be tested. Our results can be used to improve dispatching heuristics. Theoretical work on the effect of time lag reduction to the applicability of active and non-delay based heuristics is required. Finally, we expect progress can be made in specifying polynomially solvable cases with two or three machines in flow shop scheduling with multiple operations and time lags.

Acknowledgements

The authors are indebted to Professor J.N.D. Gupta for his helpful suggestions and comments at an earlier stage of this research project.

Appendix A

This appendix describes the required CPU times for the proposed heuristics. Table 5 presents the mean CPU times used for the scenario’s with $K = 20, 40, \ldots, 200$ operations. We have also performed an ANOVA analysis on the CPU time. This analysis revealed that $K$ is the most important determinant of the required CPU time.

<table>
<thead>
<tr>
<th>$K$</th>
<th>1 AS</th>
<th>2 AC</th>
<th>3 NDS</th>
<th>4 NDC</th>
<th>5 SPT</th>
<th>6 MRT</th>
<th>7 EFT</th>
<th>8 SR</th>
<th>9 LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0119</td>
<td>0.01087</td>
<td>0.00899</td>
<td>0.00873</td>
<td>0.00053</td>
<td>0.00063</td>
<td>0.00055</td>
<td>0.00071</td>
<td>0.00063</td>
</tr>
<tr>
<td>40</td>
<td>0.05506</td>
<td>0.05252</td>
<td>0.04672</td>
<td>0.04427</td>
<td>0.00117</td>
<td>0.00156</td>
<td>0.00119</td>
<td>0.00189</td>
<td>0.00157</td>
</tr>
<tr>
<td>60</td>
<td>0.15716</td>
<td>0.14837</td>
<td>0.13695</td>
<td>0.12909</td>
<td>0.00193</td>
<td>0.00280</td>
<td>0.00191</td>
<td>0.00344</td>
<td>0.00280</td>
</tr>
<tr>
<td>80</td>
<td>0.23927</td>
<td>0.22466</td>
<td>0.21047</td>
<td>0.19801</td>
<td>0.00272</td>
<td>0.00400</td>
<td>0.00275</td>
<td>0.00486</td>
<td>0.00429</td>
</tr>
<tr>
<td>100</td>
<td>0.36294</td>
<td>0.33901</td>
<td>0.31213</td>
<td>0.29214</td>
<td>0.00362</td>
<td>0.00610</td>
<td>0.00368</td>
<td>0.00727</td>
<td>0.00643</td>
</tr>
<tr>
<td>120</td>
<td>1.28814</td>
<td>1.20499</td>
<td>1.17324</td>
<td>1.09932</td>
<td>0.00500</td>
<td>0.00898</td>
<td>0.00499</td>
<td>0.01100</td>
<td>0.00882</td>
</tr>
<tr>
<td>140</td>
<td>0.84501</td>
<td>0.79758</td>
<td>0.76544</td>
<td>0.72216</td>
<td>0.00601</td>
<td>0.01100</td>
<td>0.00600</td>
<td>0.01300</td>
<td>0.01100</td>
</tr>
<tr>
<td>160</td>
<td>1.83976</td>
<td>1.71798</td>
<td>1.68948</td>
<td>1.58864</td>
<td>0.00700</td>
<td>0.01300</td>
<td>0.00694</td>
<td>0.01600</td>
<td>0.01306</td>
</tr>
<tr>
<td>180</td>
<td>2.03918</td>
<td>1.90750</td>
<td>1.88156</td>
<td>1.76038</td>
<td>0.00803</td>
<td>0.01549</td>
<td>0.00794</td>
<td>0.01900</td>
<td>0.01595</td>
</tr>
<tr>
<td>200</td>
<td>2.30550</td>
<td>2.15961</td>
<td>2.14809</td>
<td>2.01686</td>
<td>0.00898</td>
<td>0.01800</td>
<td>0.00893</td>
<td>0.02200</td>
<td>0.01864</td>
</tr>
</tbody>
</table>
Table 6
Input data for example problem \((K = 20, O = 0, B = 0, t = 0)\)

<table>
<thead>
<tr>
<th>Job</th>
<th>Machine 1</th>
<th>Machine 2</th>
<th>Machine 3</th>
<th>Machine 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operation 1</td>
<td>Operation 2</td>
<td>Operation 3</td>
<td>Operation 4</td>
</tr>
<tr>
<td></td>
<td>(T_{l_{11}})</td>
<td>(T_{l_{11}})</td>
<td>(T_{l_{12}})</td>
<td>(T_{l_{12}})</td>
</tr>
<tr>
<td></td>
<td>(T_{l_{21}})</td>
<td>(T_{l_{21}})</td>
<td>(T_{l_{22}})</td>
<td>(T_{l_{22}})</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>120</td>
<td>120</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td>100</td>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>30</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>70</td>
<td>70</td>
<td>50</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>90</td>
<td>30</td>
<td>70</td>
<td>110</td>
</tr>
</tbody>
</table>

Fig. 3. Heuristic AC, reduction \(t = 0\%\), makespan 1230.

Fig. 4. Heuristic NDC, reduction \(t = 0\%\), makespan 1190.

Fig. 5. Heuristic AC, reduction \(t = 90\%\), makespan 938.

Fig. 6. Heuristic NDC, reduction \(t = 90\%\), makespan 1032.
Operation variation \((O = 1)\) does increase the required CPU time significantly, contrary to time variability \((B)\) and time lag size \((t)\).

The type of heuristic applied does have an important influence on the required CPU time, as can be seen in Table 5. This table helps to compare the efficiency of the heuristics. We see that the four lower bound based heuristics require more CPU time than the modified flow shop scheduling heuristics, as could be expected. Furthermore, the CPU time they require increases much faster in case the total number of operations that are to be scheduled increases. However, this is accompanied by a much better performance of these lower bound based heuristics in this case, as could be seen in Section 5.2. Overall, the required CPU times are such that these heuristics can be applied in real time flow shop scheduling software.

Appendix B

This appendix illustrates the difference in performance between an active and a non-delay based heuristic when applied in a situation with no reduction and with maximal reduction of time lags. Both heuristics AC and NDC apply the same lower bound. Table 6 presents the input data for the example problem, which is one of the smallest problems in our experimental design as it consists of only 20 operations. The results of the two heuristics for both time lag reduction percentages are presented in the Figs. 3–6. Fig. 5 shows that AC gives priority to the next operation of the same job at the last machine in case time lags are very small, while NDC is not allowed to do this if another operation can start before. AC outperforms NDC in this case. The opposite holds true if the time lags are not reduced (Figs. 3 and 4).

References


