Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 162 (2005) 99–111
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Tabu search algorithms for job-shop problems with a single transport robot

Johann Hurink [a,*], Sigrid Knust [b,1]

[a] University of Twente, Department of Applied Mathematics, Faculty of Electrical Engineering, Mathematics and Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands
[b] Universität Osnabrück, Fachbereich Mathematik/Informatik, 49069 Osnabrück, Germany

Received 1 July 2001; accepted 14 October 2003
Available online 24 January 2004

Abstract

We consider a generalized job-shop problem where the jobs additionally have to be transported between the machines by a single transport robot. Besides transportation times for the jobs, empty moving times for the robot are taken into account. The objective is to determine a schedule with minimal makespan.

We present local search algorithms for this problem where appropriate neighborhood structures are defined using problem-specific properties. An one-stage procedure is compared with a two-stage approach and a combination of both. Computational results are presented for test data arising from job-shop benchmark instances enlarged by transportation and empty moving times.

Keywords: Scheduling; Job-shop problem; Robot; Transportation; Tabu search

1. Introduction

In this paper we consider a generalized job-shop scheduling problem where the jobs additionally have to be transported between the machines by a single transport robot. A *job-shop problem with transportation times and a single robot* is a generalization of the classical job-shop problem and may be formulated as follows: We are given m machines and n jobs. Each job consists of a chain of operations which have to be processed in this order. With each operation a dedicated machine is associated on which the operation has to be processed without preemption for a given duration. Each machine can process at most one operation at a time. Additionally, *transportation times* are considered. They occur if a job changes from one machine to another and depend on the jobs and the machines between which the transport takes place. We assume that all these transport operations have to be done by a single transport robot which can handle at most one job at a time. Furthermore, we assume that we have unlimited buffer

* Corresponding author. Tel.: +31-53-4893447; fax: +31-53-4894858.
E-mail addresses: j.l.hurink@utwente.nl (J. Hurink), sigrid@informatik.uni-osnabrueck.de (S. Knust).
[1] Supported by the Deutsche Forschungsgemeinschaft, Project 'Komplexe Maschinen-Schedulingprobleme'.

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2003.10.034

space between the machines. The objective is to determine a feasible schedule which minimizes the makespan, i.e. the completion time of the operation processed last.

If for the robot only the given transportation times are important, we may consider the robot as an additional "machine" which has to "process" all transport operations. Therefore, in this case the problem is equivalent to a classical job-shop problem with $m + 1$ machines. Since the robot has to process many more operations than the other machines (each second operation of a job), it is also called a bottleneck machine. However, in practice in addition to the transportation times also *empty moving times* arise when the robot moves empty between two machines without carrying a job. These empty moving times may be regarded as sequence-dependent setup times on the robot and, thus, the empty moving times imply that the robot cannot be treated in the same way as the other machines. Consequently, the job-shop problem with transportation times and a single robot consists of scheduling a set of "classical" machines and a special machine on which additionally sequence-dependent setup times have to be taken into account.

In this paper we propose two different approaches to integrate a transportation stage into procedures which schedule the machines. In an one-stage procedure we try to deal with the problem on the whole and do not distinguish between the robot and the machines. Another possibility is to apply a two-level approach, where on the first level machine orders for the job-shop machines are fixed and on the second level a corresponding robot order is constructed. For these two approaches and a combination of both we present tabu search methods to calculate heuristic solutions for the considered problem.

Whereas scheduling the machines in a classical job-shop has been studied over a long period (for summaries, see e.g. [2] or [9]), scheduling problems with transportation aspects have received much attention in the literature only in recent years (for a survey cf. [5]). Although there are many differences between the various models, they all deal with the interaction between the transportation routing and the classical job scheduling decisions.

For shop problems with transportation times, unlimited buffer space and a single robot only few literature is available. Kise [11] proved that minimizing the makespan in a two-machine flow-shop with constant transportation times and a single robot is already NP-hard. Additional complexity results for such problems can be found in Hurink and Knust [7]. King et al. [10] proposed a branch-and-bound algorithm for a flow-shop environment with a single robot. Langston [14] derived some approximation algorithms for a flexible flow-shop environment with two stages and interstage transportation times. Bilge and Ulusoy [1] proposed a heuristic for simultaneously scheduling the machines and vehicles in a flexible manufacturing system with job-shop structure.

The remainder of the paper is organized as follows. In Section 2 we give a formal definition of the considered scheduling problem and state some additional assumptions. In Section 3 we extend the well-known disjunctive graph model to the situation with one transport robot and derive some problem-specific properties which are the basis for our local search procedures. An one-stage approach is proposed in Section 4, a two-stage procedure and a combination of both are presented in Section 5. Finally, computational results can be found in Section 6.

2. Problem formulation

In this section we present a formal definition of the considered problem and state some additional assumptions. We are given m machines $M_1, \ldots, M_m$ and n jobs $J_1, \ldots, J_n$. Each job J_j consists of n_j operations O_{ij} $(i = 1, \ldots, n_j)$ which have to be processed in the order $O_{1j} \to O_{2j} \to \cdots \to O_{n_j,j}$. Operation O_{ij} has to be processed without preemption on a dedicated machine $\mu_{ij} \in \{M_1, \ldots, M_m\}$ for $p_{ij} > 0$ time units. Each machine can only process one operation at a time. Additionally, *transportation times* are considered. They occur if a job changes from one machine to another, i.e. if job J_j is processed on machine M_k and afterwards on machine M_l, a transportation time t_{jkl} arises. These transportation times may be job-dependent or job-independent ($t_{jkl} = t_{kl}$). We assume that all

transportations have to be done by a single transport robot R which can handle at most one job at a time. The transportation times are supposed to satisfy the following triangle inequality for all jobs J_j and all machines M_k, M_l, M_h:

$$t_{jkh} + t_{jhl} \geqslant t_{jkl}. \tag{1}$$

If this inequality does not hold, i.e. $t_{jkh} + t_{jhl} < t_{jkl}$ for some indices j, k, h, l, we could save time for the transport from M_k to M_l by first transporting job J_j from M_k to M_h and then to M_l. In practice, this situation is unlikely to occur, hence this assumption is not a real restriction.

In addition to the transportation times we consider *empty moving times* t'_{kl}. While the transportation times arise when a job is transported from one machine to another, the empty moving times t'_{kl} arise when the robot moves empty from machine M_k to M_l without carrying a job. For these times we assume

$$t'_{kk} = 0, \quad t'_{kh} + t'_{hl} \geqslant t'_{kl} \quad \text{and} \quad t'_{kl} \leqslant \min_{j=1}^{n} \{t_{jkl}\}. \tag{2}$$

The first assumption means that no empty moving times have to be considered if the robot waits at a machine for the next transport. The second condition is again a triangle inequality and the third states that moving empty between two machines does not take longer than moving a job between the same two machines. In practical situations these assumptions are satisfied in most cases.

As in classical shop problems we assume that sufficient buffer space exists between the machines. This means that each machine M_k has an unlimited output buffer where jobs processed on M_k and waiting for the robot may be stored. The jobs are automatically transferred into this buffer and no further times for this transfer are considered. Additionally, each machine M_l has an unlimited input buffer where jobs which have been transported and await processing on M_l may be stored.

All data p_{ij}, t_{jkl}, t'_{kl} are assumed to be nonnegative integers. The objective is to determine a feasible schedule which minimizes the makespan $C_{\max} = \max_{j=1}^{n} \{C_j\}$, where C_j denotes the completion time of the last operation $O_{n_j,j}$ of job J_j.

3. The disjunctive graph model

In this section the well-known disjunctive graph model for the classical job-shop problem developed by Roy and Sussmann [17] is extended to the job-shop problem with transportation times and a single robot. Since the classical disjunctive graph model already deals with all conflicts regarding the job-shop machines, we only have to incorporate the scheduling of the robot into the model. This is done by introducing transport operations for all needed transports as additional vertices in the disjunctive graph and requiring that these operations have to be processed by the robot. Furthermore, the empty moving times are modeled as sequence-dependent setup times. Thus, the *disjunctive graph* $G = (V, C \cup D_{\mathrm{M}} \cup D_{\mathrm{R}})$ consist of a set of vertices V containing all operations (ordinary and transport) and two dummy nodes 0 and *, a set of conjunctions C representing the job orders, and disjunctions for the machines (D_{M}) and the robot (D_{R}).

More precisely, for each job J_j ($j = 1, \ldots, n$) we introduce $n_j - 1$ so-called *transport operations* T_{ij} ($i = 1, \ldots, n_j - 1$) with precedences $O_{ij} \to T_{ij} \to O_{i+1,j}$. The processing time of T_{ij} is equal to the transportation time of job J_j from machine μ_{ij} to $\mu_{i+1,j}$, i.e. $p(T_{ij}) = t_{jkl}$, when $\mu_{ij} = M_k$, $\mu_{i+1,j} = M_l$. The robot may be considered as an additional "machine" which has to process all these transport operations. As for the classical job-shop, the conjunctions C model the order of the operations within job. In addition to the classical set of undirected machine disjunctions D_{M} (consisting of all pairs of operations which have to be processed on the same machine and are not linked by a directed path of conjunctions) we have a set of undirected robot disjunctions D_{R} (consisting of all pairs of transport operations which are not linked by a directed path of conjunctions). Such an edge $T_{ij} - T_{uv} \in D_{\mathrm{R}}$ represents the two possible orders in which the transport operations T_{ij} and T_{uv} may be processed on the robot and is weighted with the pair (t'_{kl}, t'_{gh}) of empty moving times, when $T_{ij} \in G_{hk}$, $T_{uv} \in G_{lg}$, where G_{kl} $(k, l = 1, \ldots, m)$ is the set of all transport operations with the same transport routing, i.e. we have

$T_{ij} \in G_{kl}$ if and only if $\mu_{ij} = M_k,\ \mu_{i+1,j} = M_l$.

To solve the scheduling problem we have to turn all undirected arcs in $D_M \cup D_R$ into directed ones. Concerning an edge in D_R weighted with a pair of transportation times, at this point the directed arc gets the corresponding unique weight. If we orient an edge $T_{ij} - T_{uv}$ with $T_{ij} \in G_{hk}$, $T_{uv} \in G_{lg}$ in the direction $T_{ij} \to T_{uv}$, it gets the weight t'_{kl}, and if we orient it in the other direction, it gets the weight t'_{gh}. Disjunctions which have been directed are called *fixed*. We call a set S_M of fixed machine disjunctions a *machine selection*, a set of fixed robot disjunctions S_R a *robot selection* and their union $S = S_M \cup S_R$ a *selection*. A selection S is called *complete* if all disjunctions in $D_M \cup D_R$ have been fixed, and the resulting graph $G(S) = (V, C \cup S_M \cup S_R)$ is acyclic.

Given a complete selection S we may construct a corresponding feasible schedule by longest path calculations and starting each operation as early as possible. The completion time $C_{max}(S)$ of the schedule corresponding to a selection S is given by the length of a longest path from 0 to * in the acyclic graph $G(S)$. Such a path is also called a *critical path*.

Example. We consider an instance of the job-shop problem with a single robot, $m = 4$ machines, $n = 3$ jobs, 8 "ordinary" operations and 5 transport operations. The corresponding disjunctive graph can be found in Fig. 1. All transport operations which do not belong to the same job are linked by a robot disjunction weighted with the corresponding empty moving times. For example, the disjunction between the transport operations $T_{21} \in G_{23}$ and $T_{23} \in G_{13}$ is weighted with the pair (t'_{31}, t'_{32}), the disjunction between $T_{11} \in G_{12}$ and $T_{13} \in G_{41}$ is weighted with $(t'_{24}, t'_{11} = 0)$. In Fig. 2 a complete selection is shown in which the machine disjunctions are oriented into $O_{11} \to O_{23}$ on M_1, into $O_{21} \to O_{12}$ on M_2, into $O_{22} \to O_{31} \to O_{33}$ on M_3, and the robot sequence is given by $T_{11} \to T_{13} \to T_{12} \to T_{21} \to T_{23}$. A corresponding schedule can be found in Fig. 3. □

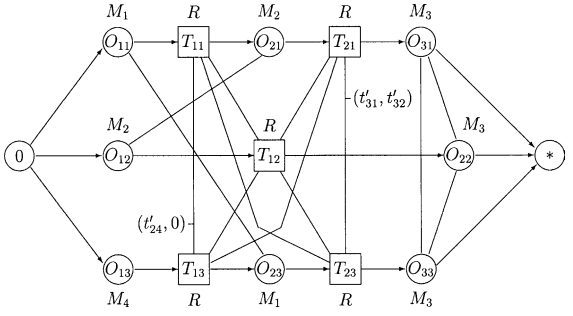

Fig. 1. Disjunctive graph for a job-shop with a single robot and $n = 3$ jobs.

In the following, we derive some problem-specific properties which are useful to define appropriate neighborhoods for local search algorithms. We represent feasible solutions by complete selections $S = S_M \cup S_R$ in the disjunctive graph where all machine and robot disjunctions are fixed such that the resulting graph $G(S) = (V, S)$ is acyclic. We consider the situation in which a complete selection S with makespan $C_{max}(S)$ is given and we want to improve this solution. In order to define appropriate neighborhood structures we use a so-called block approach. Such an approach was first proposed for the single-machine problem $1|r_j|L_{max}$ [6]. Later it was success-

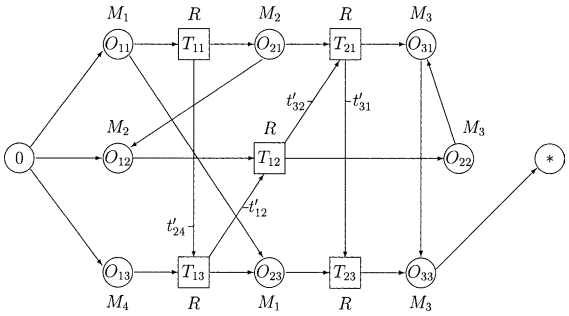

Fig. 2. A complete selection.

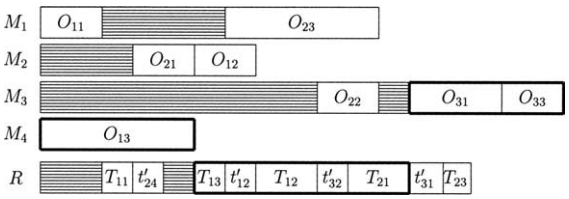

Fig. 3. A corresponding schedule.

fully adapted to some other scheduling problems (like the job-shop, flow-shop or general-shop problem, cf. [3,4,16]). With the definition of blocks it can be stated that only certain changes of a selection S may have a chance to improve the current makespan $C_{\max}(S)$ and in the search process only such solutions will be considered as candidates for neighbored solutions.

Let P^S be a critical path in $G(S)$. A sequence $u_1, \ldots, u_k$ of at least two successive nodes (i.e. $k > 1$) on P^S is called

- a *machine-block* if the operations of the sequence are processed consecutively on the same machine, and enlarging the subsequence by one operation leads to a subsequence which does not fulfill this property,
- a *robot-block* if the sequence consists of transport operations which are processed consecutively on the robot without idle times (empty moving times are no idle times), no conjunction exists between consecutive operations and enlarging the subsequence by one operation would violate one of the previous properties.

In the example above $P_1^S = (O_{13} \to T_{13} \overset{t'_{12}}{\to} T_{12} \overset{t'_{32}}{\to} T_{21} \to O_{31} \to O_{33})$ is a critical path with the machine-block (O_{31}, O_{33}) on M_3 and the robot-block (T_{13}, T_{12}, T_{21}). Another critical path is

$$P_2^S = (O_{11} \to T_{11} \to O_{21} \to O_{12} \to T_{12} \overset{t'_{32}}{\to} T_{21} \to O_{31} \to O_{33})$$

with the machine-blocks (O_{21}, O_{12}) on M_2, (O_{31}, O_{33}) on M_3 and the robot-block (T_{12}, T_{21}).

The following theorem is the basis for defining suitable neighborhoods on the set of selections. It can easily be proved by combining the corresponding proofs for the classical job-shop problem (cf. [3]) and the situation on the robot (cf. [8]). Note that for the condition on the robot the triangle inequality (2) for the empty moving times is necessary.

Theorem 1. *Let S be a complete selection with makespan $C_{\max}(S)$ and let P^S be a critical path in $G(S)$. If another complete selection S' with $C_{\max}(S') < C_{\max}(S)$ exists, then in S'*

- *at least one operation of some machine-block B on P^S has to be processed before the first or after the last operation of B, or*
- *at least two transport operations of a robot-block on P^S are processed in the opposite order.*

4. An one-stage approach

In this section we present an one-stage tabu search approach for the job-shop problem with a single transport robot. The search space of the tabu search algorithm is the set of all complete selections and in each iteration a neighbored solution is generated either by moving an operation on a machine or by moving a transport operation on the robot to another position. In the following we define suitable neighborhoods and describe some further elements of the tabu search approach. The presented neighborhoods form extensions of neighborhoods used for the job-shop problem, where the special role of the transport operations has been integrated.

Theorem 1 gives necessary conditions for improving a given solution. Both conditions stated in Theorem 1 indicate that orders of operations from blocks on a critical path have to be changed. Thus, a first neighborhood $\mathcal{N}_1$ may be defined as for the job-shop problem by interchanging neighbored operations on a critical path (cf. [13]). This neighborhood considers also the interchange of neighbored operations in the inner part of machine-blocks although due to Theorem 1 such moves cannot result in improving solutions. However, these extra moves are needed to prove that $\mathcal{N}_1$ has the nice (theoretical) property that it is *weakly connected*, i.e. from an arbitrary selection it is possible to reach a globally optimal selection via a sequence of steps in $\mathcal{N}_1$. For this proof, we first derive a property of neighbored operations within blocks on a critical path.

Lemma 1. *Let i and j be two neighbored operations (ordinary or transport) within a block of a critical path belonging to a complete selection S. Then, in $G(S)$, besides the arc (i, j) no further path exists from i to j.*

Proof. Assume that in $G(S)$ a path $P = (i, u_1, \ldots, u_k, j)$; $k \geqslant 1$ with length ℓ exists.

Case 1. i and j belong to a machine-block.

Since all vertices are weighted with processing times greater than zero, we get

$$\ell \geqslant p_i + p_{u_1} > p_i,$$

which contradicts the fact that (i, j) is an arc of a critical path in $G(S)$.

Case 2. i and j belong to a robot-block.

In this case i and j are two transport operations which are processed consecutively by the robot. Thus, since S is a feasible selection, the path P contains no further transport operations. As a consequence, all vertices $u_1, \ldots, u_k$ have to belong to operations which are processed on the same machine (say M_1) and, thus, transport operation i has to be a transport to machine M_1 and transport operation j has to be a transport away from machine M_1. This implies that the length ℓ' of the path consisting only of the arc (i, j) is equal to $\ell' = p_i + t'_{ll} = p_i$ due to $t'_{ll} = 0$. Thus

$$\ell \geqslant p_i + p_{u_1} > \ell',$$

which contradicts the fact that (i, j) is an arc of a critical path in $G(S)$. □

Using this lemma we can prove

Theorem 2. *Neighborhood $\mathcal{N}_1$ is weakly connected.*

Proof. Let S be an arbitrary non-optimal complete selection and S^* an optimal selection. Due to Theorem 1 we know that in S for at least two operations of a block (machine-block or robot-block) on a critical path in $G(S)$ the corresponding disjunction is fixed in the opposite direction compared with S^*. Consequently, also two neighbored operations of a block with this property exist. Since by Lemma 1 their exchange leads to a feasible selection, by one step in $\mathcal{N}_1$ we can achieve a solution which has one more disjunction fixed in the same way as in S^*. Iteratively applying this step leads to a globally optimal solution by a sequence of steps in neighborhood $\mathcal{N}_1$. □

Theorem 2 gives a nice theoretical result for neighborhood $\mathcal{N}_1$, but first computational tests have shown that on the one hand, this neighborhood does not give us enough possibilities to change a solution and to reach different regions of the search space fast enough. On the other hand, as already mentioned, $\mathcal{N}_1$ contains superfluous interchanges of neighbored operations in the inner part of machine-blocks which cannot lead to an improvement of the current solution due to Theorem 1.

Since for the classical job-shop problem shifts have been successful to achieve diversification in the search process, we also will develop a neighborhood based on shift operators. For a given selection $S = S_M \cup S_R$ and a critical path P^S neighborhood $\mathcal{N}_2$ contains all feasible selections $S' = S'_M \cup S'_R$ which can be constructed as follows:

- in S'_M one operation (different from the first one) of a machine-block B on P^S is moved before all other operations in B, or one operation (different from the last one) of a machine-block B on P^S is moved after all other operations in B, or
- in S'_R an operation on a position $k \leqslant \lceil \frac{|B|}{2} \rceil$ of a robot-block B on P^S is moved before the operation on position $j \in \{1, \ldots, k-1, |B|-k+2, \ldots, |B|+1\}$ of B, or in S'_R an operation on a position $k > \lceil \frac{|B|}{2} \rceil$ of a robot-block B on P^S is moved before the operation on position $j \in \{1, \ldots, |B|-k+1, k+2, \ldots, |B|+1\}$ of B.

(Here $|B|$ denotes the number of operations in B and moving an operation before position $|B|+1$ of B means moving it after B.) For example, for the block $B = (1, 2, 3, 4, 5)$ we allow the moves $(2, 3, 4, 5, \bar{1})$, $(\bar{2}, 1, 3, 4, 5)$, $(1, 3, 4, \bar{2}, 5)$, $(1, 3, 4, 5, \bar{2})$, $(\bar{3}, 1, 2, 4, 5)$, $(1, \bar{3}, 2, 4, 5)$, $(1, 2, 4, \bar{3}, 5)$, $(1, 2, 4, 5, \bar{3})$, $(\bar{4}, 1, 2, 3, 5)$, $(1, \bar{4}, 2, 3, 5)$, $(1, 2, 3, 5, \bar{4})$, and $(\bar{5}, 1, 2, 3, 4)$.

The difference in defining the shifts for the normal machines and the robot results from the different conditions in Theorem 1. In Brucker and Thiele [4] similar moves are used within the branching step of a branch and bound approach (they use the concept of so-called extra-blocks) and there results imply that neighborhood $\mathcal{N}_2$ covers the conditions of Theorem 1 in an appropriate way.

Preliminary computational tests have shown that the navigation behavior of the new neighborhood is better than that of $\mathcal{N}_1$. Furthermore, enlarging neighborhood $\mathcal{N}_2$ by some additional operators which make the neighborhood weakly connected did not improve the quality of the results but only increased the computational times.

To use neighborhood $\mathcal{N}_2$ in a tabu search approach, we still have to define which information is stored in a tabu list. The goal of using a tabu list is to avoid coming back to a solution which has already been visited in previous iterations. If in some iteration we leave a complete selection S by moving an ordinary (transport) operation u to some other position, we store u together with its machine (robot) predecessor and successor and the makespan $C_{\max}(S)$ of the solution S. A solution S' is defined to be tabu if $C_{\max}(S')$ equals the makespan of an element in the tabu list and if the triple of operations stored in this element of the tabu list is scheduled in S' in the same way as indicated in the entry of the tabu list. We use a static tabu list, i.e. the length of the tabu list is hold constant during the whole search process. Since the objective values are already taken into account in the attributes of the solutions, no additional aspiration criteria are used.

First computational tests with neighborhood $\mathcal{N}_2$ indicated that it is very time-consuming to determine the best non-tabu neighbor in each iteration. This is caused by the fact that the size of the neighborhood of a solution is rather large and that the evaluation of each neighbor needs a recalculation of the makespan (longest path calculation). Thus, in order to reduce this large computational effort for one iteration of the tabu search procedure, we use lower bound calculations. More precisely, for each neighbor of the current solution we first determine an easily calculable lower bound for the makespan and only if this lower bound is below the value of the best neighbor found so far, we evaluate the makespan of the neighbor exactly. Incorporating these bounds leads to a significant reduction of the computational times. The whole tabu search process is stopped after a certain number of non-improving iterations.

To start the tabu search procedure, a first solution is constructed using a priority rule based heuristic given in Brucker and Thiele [4] where the robot is considered as an additional machine and the empty moving times are regarded as setup times. Additional details of the tabu search implementation are given in connection with the presentation of computational results in Section 6.

5. A two-stage approach

In this section we present another local search algorithm which is based on Theorem 1, but which tries to deal with the different situations on the machines and the robot more individually. This algorithm is organized in two stages. While in in the outer stage an operation on a machine is moved to another position, in the inner stage the robot is optimized according to the new machine selection. In order to obtain a good robot solution we use a tabu search procedure of Hurink and Knust [8] as a subroutine.

In the one-stage approach of the previous section in each iteration a neighbored solution S' is constructed by either changing the machine selection S_{M} or the robot selection S_{R} (but not both). Since the robot corresponds to a bottleneck machine, the old robot selection S_{R} may be a bad solution for the new machine selection S'_{M} or S_{R} is even infeasible w.r.t. S'_{M} (i.e. $G(S')$ contains a cycle). To overcome this disadvantage, in an alternative approach we proceed in two stages, i.e. we define a neighborhood where after changing the machine selection the robot selection is adapted before the makespan of the new solution is determined. This means that in the second stage the situation for the robot has to be considered where all machine orders are fixed. We are interested in a robot order respecting all precedences induced by the fixed machine orders and having a minimal makespan among all robot orders which are compatible with the given machine selection. As stated in [8], the resulting robot problem may be denoted as follows in the well-known $\alpha|\beta|\gamma$-notation: $1|\mathrm{prec}(l_{ij}), r_j, s_{ij}|\max\{C_j + q_j\}$, where $\mathrm{prec}(l_{ij})$ indicates arbitrary non-negative finish-start time-lags $l_{ij} \geqslant 0$, s_{ij} stands for sequence-dependent setup

times, r_j for release dates (heads), and $C_j + q_j$ denotes for each job the sum of its completion time and its tail.

If we take the example from Section 3 with the machine selection from Fig. 2, we get a situation for the robot as presented in Fig. 4. Besides the chain precedences of the jobs we get a conjunction $T_{11} \to T_{12}$ weighted with the time-lag $p_{21} + p_{12}$ induced by the machine orientation $O_{21} \to O_{12}$. Between all transport operations which are not linked by a conjunction, a disjunction weighted with the corresponding pair of empty moving times exists (in Fig. 4 for clarity only the disjunction $T_{11} - T_{13}$ is shown).

In Hurink and Knust [8] an effective tabu search procedure for the robot problem is presented. According to Theorem 1 the neighborhood of a robot selection consists of selections in which the order of critical operations in a robot-block is changed. More specifically, the neighborhood is defined by three types of operators: either two adjacent transport operations of a block are interchanged, the first block operation is shifted to the right or an internal block operation is shifted to the end of the block. Although a neighbored selection differs only slightly from the current selection, the starting times for all transport operations which appear after the shifted operation have to be recalculated to determine the makespan of a neighbored solution. Due to the time-lags such a calculation is very time-consuming and it is not efficient to evaluate all neighbors exactly. Thus, not the correct objective values are calculated for all neighbors, but approximate values are used (for details see [8]).

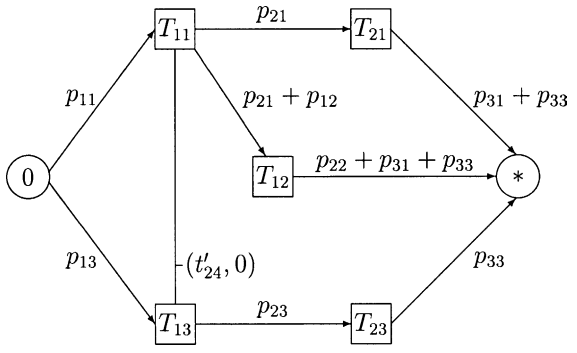

Fig. 4. The robot problem for a fixed machine selection.

Based on such a hierarchical decomposition of the considered problem a neighborhood may also be defined in a hierarchical way where first the machine selection is changed and then the robot selection is adapted. For a given selection $S = S_M \cup S_R$ neighborhood $\mathcal{N}_3$ contains all feasible selections $S' = S'_M \cup S'_R$ which can be constructed as follows:

- in S'_M one operation (different from the first one) of a machine-block B on P^S is moved before all other operations in B, or one operation (different from the last one) of a machine-block B on P^S is moved after all other operations in B, and
- S'_R is obtained from S_R by calculating a heuristic solution for the robot problem associated with the new machine selection S'_M.

For instances with large transportation times it may happen that no machine-blocks exist for a considered solution S and, thus, neighborhood $\mathcal{N}_3(S)$ is empty. To improve such a solution S, according to Theorem 1 some robot-block has to be destroyed. Since in the outer stage of our hierarchical approach we only want to change the machine selection, but not the robot selection, we allow some additional moves on the level of the machine orders. For this purpose we do not only move critical operations belonging to machine-blocks, but also single operations on the chosen critical path. According to Theorem 1 interchanging a single critical operation (not belonging to a machine-block) with its machine predecessor or successor cannot reduce the objective value, but in combination with the change of the robot selection an improving solution may be obtained (since time-lags in the robot problem are canceled by moving critical operations on the machines).

Thus, we enlarge $\mathcal{N}_3(S)$ by all feasible selections $S' = S'_M \cup S'_R$, where in S'_M a single critical operation on P^S is interchanged with its machine predecessor or successor, and S'_R is again a solution associated with the new machine selection S'_M. The resulting neighborhood $\mathcal{N}_4$ is used in a tabu search algorithm, which will be described in more detail next. As in the tabu search procedure from

Section 4, we store attributes which characterize visited solutions during the search process in a static tabu list. If in a complete selection $S' = S'_M \cup S'_R$ the machine selection S'_M results from S_M by moving an operation u, we store the objective value $C_{max}(S)$ besides the triple consisting of u, its old machine predecessor and successor in S_M. A solution S is defined to be tabu if $C_{max}(S)$ equals the objective value of an element in the tabu list and the associated triple of this element in the tabu list is reconstructed in S. Note that contrary to the one-stage approach only machine operations are stored in the tabu list, the robot sequence is implicitly taken into account by the C_{max}-value. Again, the whole search process is stopped after a certain number of non-improving iterations.

During the search process we use two versions of the tabu search algorithm for the robot problem. First we evaluate all neighbored solutions in $\mathcal{N}_4(S)$ by applying the tabu search algorithm using only a small number of iterations. the best neighbored solution which is not tabu and try to improve the corresponding robot selection using a larger number of iterations.

In order to obtain starting solutions for the two-stage tabu search procedure, we use two modified versions of the priority-based heuristic from Brucker and Thiele [4]. Besides the one-stage version described in the previous section, we use a two-stage version in which we first only calculate a machine selection S_M and determine a corresponding robot selection S_R by the robot tabu search algorithm afterwards.

Finally, we describe some combinations of the one-stage and the two-stage approach. We combine these two approaches in order to intensify and diversify the search process. In a first version we alternately apply the two algorithms allowing for each algorithm a constant number of non-improving iterations. At first the algorithm starts with the two-stage approach and stops after q non-improving iterations. After that we allow $f * q$ non-improving iterations of the one-stage approach, where the factor f is a given constant. The algorithm continues changing between the two approaches until no further improvement of the best known solution occurs in both methods. In a second version we change the order of the two methods beginning with the one-stage approach. We use the same strategy of organizing the number of iterations.

In a third version we do not have a constant factor f but keep it as a variable during the search process. After each run of $f * q$ non-improving iterations of the one-stage and q non-improving iterations of the two-stage approach, we calculate a new value for f by

$$f := C * \frac{\text{iter}_2}{\text{iter}_1},$$

where iter_2 (iter_1) is the absolute number of iterations of the two-stage (one-stage) approach from the last run and C is a given constant. The idea of such a modified value is that the algorithm tries to regulate the ratio between the number of iterations of the two procedures by itself. For example, if iter_2 is big (i.e. there were many improving solutions in the last run of the two-stage approach), and iter_1 is small (i.e. there were not many improving solutions in the last run of the one-stage approach), it could be good to intensify the one-stage part in the search. To run this approach we only have to specify the initial value f^0 of f and the value of C which may be obtained after testing the algorithm with different values.

6. Computational results

In this section we present some computational results for the described tabu search algorithms. We implemented all procedures in C and tested them on a Sun Ultra 2 work station (167 MHz) with operating system Solaris 2.5 and 320 MB general storage.

Since no test instances for the job-shop problem with transportation times were available from the literature, we modified $m \times n$ benchmark problems for the classical job-shop problem, where m denotes the number of machines and n the number of jobs. We used the well-known 6×6 and 10×10 instances $P1$ and $P2$ from Muth and Thompson [15]. In both instances the number of operations per job is equal to the number of machines (i.e. $n_j = m$ for $j = 1, \ldots, n$) and each job is processed on each machine exactly once. The processing

times of the operations in the instance $P1$ are from the interval $[1,10]$ and in $P2$ from the interval $[1,100]$. Various test instances were obtained by adding transportation and empty moving times with different characteristics.

For the transportation times t_{jkl} we distinguished the following four cases: job- and machine-dependent times t_{jkl} randomly generated from the interval $[1,10]$ (adjusted in such a way that the triangle inequality holds), job-independent transportation times t_{kl} analogously to the first case, job-independent transportation times $t_{kl} = D|k-l|$ with different values D (proportional to the distance between the corresponding machines when they are assumed to be ordered in a single line), and constant transportation times $t_{jkl} = T$. Analogously, we distinguished the following three cases for the empty moving times: randomly generated values t'_{kl}, values $t'_{kl} = d|k-l|$ depending on the machine distances and constant times $t'_{kl} = t$.

In this way we obtained several instances with $6\times5=30$ or $10\times9=90$ transport operations (arising from the 6×6 and 10×10 job-shop instances $P1$ and $P2$, respectively). Since the processing times in instance $P2$ are very large (from the interval $[1,100]$), the time horizon for the modified instances is often also very large ($C_{\max} \in [1000,3000]$). Therefore, we also generated some instances in which the processing times are scaled by a factor $0 < f < 1$, i.e. we replaced the processing times p_{ij} by $\lceil f \cdot p_{ij} \rceil$.

After some first computational tests with a large test set we tried identifying interesting instances which are not easy to solve (i.e. for which priority rules do not produce solution values with small deviations from lower bound values). In the following we will only report results for these 30 instances (10 6×6 instances with 30 transport operations and 20 10×10 instances with 90 transport operations).

In order to estimate the quality of the presented procedures we compared the results of the one-stage procedure with the two-stage procedure and their combination. Furthermore, we calculated lower bounds LB_1 for the instances using the techniques of constraint propagation and linear programming (cf. [12]). Unfortunately, for the 10×10 instances the linear programming bounds could not be calculated, i.e. for these instances we could compare our results only with relatively weak lower bounds LB_0 (obtained with simple constraint propagation techniques).

Preliminary computational tests showed that the quality of the different procedures (varying the starting solutions, the tabu list lengths and the stopping criteria) differs from instance to instance. Thus, for the final computational tests we decided to run the procedures several times with different parameters. The one-stage and two-stage procedures were executed 6 times, the combination 12 times (additionally varying 2 different values for the number q of non-improving iterations). Additionally, for each run a time-limit of 10 minutes was imposed. Concerning the combination of the one- and the two-stage procedure the third version with a variable factor f outperformed the two other versions with constant factors. After some preliminary tests we took $C = 1000$ and an initial value of $f^0 = 30$.

For each instance and each of the three approaches we determined the best value UB^{best} and the average value UB^{av} obtained in a test series with the specified number of runs (6 and 12, respectively). For these heuristic solution values UB we determined the relative deviation $\Delta(LB) = \frac{UB-LB}{LB}$ from a lower bound value LB. In Tables 1 and 2 we report the average and maximal relative deviations $\Delta(LB)_{av}$ and $\Delta(LB)_{max}$ (in %) as well as the average and maximal computational times (in minutes:seconds) for the $P1$- and $P2$-instances, respectively.

Table 1 shows that for the small instances the one-stage approach outperforms the others. Good solutions can be obtained within small computational times. On the other hand, Table 2 gives the impression that all three approaches do not differ a lot concerning their quality and do not give very good results. However, we have to take into account that the deviations given in Table 2 are based on the weak lower bound LB_0. For the $P1$-instances the better bound LB_1 has an average deviation of $\frac{LB_1-LB_0}{LB_0} = 7.8\%$ from LB_0. Thus, we may expect that for the larger $P2$-instances the average optimal value deviates a lot from LB_0 and, therefore, we may argue that the achieved quality

Table 1
Results for the 10 *P*1-instances

		UB^{best}	UB^{av}	time
One-stage	$\Delta(LB_1)_{av}$	2.2	3.9	1:17
	$\Delta(LB_1)_{max}$	6.1	7.5	2:57
Two-stage	$\Delta(LB_1)_{av}$	4.3	5.6	0:56
	$\Delta(LB_1)_{max}$	7.8	9.6	2:55
Combination	$\Delta(LB_1)_{av}$	3.9	5.0	1:46
	$\Delta(LB_1)_{max}$	7.8	8.7	5:30

Table 2
Results for the 20 *P*2-instances

		UB^{best}	UB^{av}	time
One-stage	$\Delta(LB_0)_{av}$	18.1	21.3	2:41
	$\Delta(LB_0)_{max}$	27.4	30.1	10:00
Two-stage	$\Delta(LB_0)_{av}$	20.8	23.8	5:55
	$\Delta(LB_0)_{max}$	36.8	42.5	10:00
Combination	$\Delta(LB_0)_{av}$	19.8	23.0	5:43
	$\Delta(LB_0)_{max}$	37.2	39.8	10:00

is good and that again the one-stage approach outperforms the two other approaches.

The results given in Tables 1 and 2 are obtained using a time-limit of 10 minutes. Additional tests have shown that for the one-stage approach the results do not improve a lot if larger computational times are allowed. On the other hand, since in the two-stage method the effort of evaluating neighbored solutions is very high, in the same amount of time much less solutions can be visited than in the one-stage procedure. If we allow computational times up to one hour, the quality of the combined approach improves a lot. The deviations of the best results (UB^{best}) reduce to $\Delta(LB_0)_{av} = 16.1\%$ and $\Delta(LB_0)_{max} = 34.0\%$ and, thus, for longer computational times the combined approach outperforms the one-stage approach.

Furthermore, a closer look at the individual results for the different instances shows that the approaches behave quite differently for different instances. Table 3 contains the following information:

- $\sum t_{jkl}$: the sum of all transportation times of the transport operations;
- UB^0: the best objective value obtained by a version of the priority-based heuristic;
- UB^{one}: the best objective value obtained within 10 min with the one-stage approach;
- UB^{two}: the best objective value obtained within 10 min with the two-stage approach;
- UB^{comb}_{short}: the best objective value obtained within 10 min with the combined approach;
- UB^{comb}_{long}: the best objective value obtained within one hour with the combined approach;
- LB_1: the lower bound obtained with constraint propagation and linear programming;
- LB_0: the weak lower bound obtained only with constraint propagation.

A '*' indicates a best solution found within 10 minutes and a '+' indicates that the long run with the combined approach gave a better solution than the best solution found within 10 minutes with one of the three approaches.

Since the differences between the results of the different approaches are often quite large and since no tendency for a real best method can be given, in practice one should decide to use not only one of the presented approaches, but one should choose two different approaches to calculate solutions. Based on our test results, a combination of the one-stage approach with a short computational time and the combined method

Table 3
Individual results for all 30 instances

Instance	$\sum t_{jkl}$	UB^0	UB^{one}	UB^{two}	UB^{comb}_{short}	UB^{comb}_{long}	LB_1	LB_0
$P1\_t_{jkl}\_t'_{kl}.1$	124	137	134*	137	137	–	133	126
$P1\_t_{jkl}\_t'_{kl}.2$	118	135	129*	132	134	–	128	121
$P1\_t_{jkl}\_t'_{kl}.3$	132	146	144*	146	144*	–	142	134
$P1\_D1\_d1$	66	90	87*	88	88	–	82	70
$P1\_D1\_t1$	66	83	81*	83	83	–	77	70
$P1\_D2\_d1$	132	156	148*	155	153	–	147	134
$P1\_D3\_d1$	198	220	217	219	216*	–	213	200
$P1\_t_{kl}\_t'_{kl}.1$	121	139	137*	138	141	–	136	123
$P1\_T2\_t1$	60	77	74*	76	74*	–	71	63
$P1\_T3\_t0$	90	94	92*	94	93	–	92	92
$P2\_D1\_d1$	223	1088	1044	1035	1013*	990+	–	880
$P2\_D1\_t0$	223	1088	1042	1055	989*	989	–	880
$P2\_D1\_t1$	223	1088	1016	1021	995*	989+	–	880
$P2\_D2\_d1$	446	1123	1070	1064	1004*	993+	–	892
$P2\_D3\_d1$	669	1163	1070*	1084	1078	1072	–	906
$P2\_D5\_t2$	1115	1444	1325*	1390	1383	1371	–	1167
$P2\_T1\_t1$	90	1092	1006*	1053	1022	1018	–	874
$P2\_T2\_t1$	180	1094	1015*	1058	1053	1030	–	880
$P2\_T5\_t2$	450	1102	1102	1102	1090*	1020+	–	898
$P2\_t_{jkl}\_t'_{kl}.1$	338	1101	1082	1066*	1089	1027+	–	890
$P2\_t_{jkl}\_t'_{kl}.2$	333	1097	1035*	1063	1087	1033+	–	891
$P2\_t_{jkl}\_t'_{kl}.3$	298	1093	1039*	1065	1081	989+	–	888
$P2\_t_{jkl}\_t'_{kl}.4$	368	1096	1045*	1070	1084	997+	–	893
$P2\_t_{kl}\_t'_{kl}.1$	337	1098	1086	1090	1061*	1018+	–	888
$P2\_t_{kl}\_t'_{kl}.2$	385	1102	1028*	1073	1058	1014+	–	896
$P2f0.5\_D1\_d1$	223	600	555*	578	562	558	–	482
$P2f0.5\_D1\_t1$	223	595	544*	559	551	542	–	482
$P2f0.5\_D2\_d1$	446	689	633*	680	674	666	–	497
$P2f0.5\_D2\_t0$	446	611	578*	603	595	595	–	497
$P2f0.5\_D2\_t1$	446	635	613*	627	621	620	–	497

with a longer computational time seems to be the best.

Summarizing, we can state that the presented approaches are able to produce solutions of a high quality for the tested instances. The deviations of the best solution values obtained in any version of the tests are $\Delta(LB_1)_{av} = 2.1\%$, $\Delta(LB_1)_{max} = 6.1\%$ for the $P1$-instances and $\Delta(LB_0)_{av} = 14.7\%$, $\Delta(LB_0)_{max} = 27.4\%$ for the $P2$-instances.

7. Concluding remarks

We developed different tabu search approaches for a generalization of the job-shop problem where additionally transportation aspects are taken into account. The approaches differ in the way the transportation is treated. For the one-stage approach the complete problem is transformed into a job-shop problem with sequence-dependent setup times, where the transportation leads to an additional machine. The tabu search method treats all machines (inclusive the "transportation" machine) in the same way. The two-stage approach distinguishes between transportation on the robot and processing on the machines. In the outer stage sequences on the machines are fixed and based on this schedule in the second stage a transport schedule is calculated. In both stages tabu search is used to determine the solutions.

The computational results show that both approaches are able to produce good solutions within reasonable time. However, if only short computational times are available, the one-stage

approach outperforms the two-stage approach. For longer computational times a combination of both methods is most successful. Furthermore, the tests have shown that the success of the methods may differ from instance to instance and, thus, in practice all methods may be worthwhile to be used.

For further research it would be interesting to develop some acceleration techniques for the two-stage approach and some methods which are able to produce stronger lower bounds for large instances.

References

[1] Ü. Bilge, G. Ulusoy, A time window approach to simultaneous scheduling of machines and material handling system in an FMS, Operations Research 43 (1995) 1058–1070.

[2] J. Blażewicz, W. Domschke, E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, European Journal of Operational Research 93 (1996) 1–33.

[3] P. Brucker, B. Jurisch, B. Sievers, A branch and bound algorithm for the job-shop problem, Discrete Applied Mathematics 49 (1994) 107–127.

[4] P. Brucker, O. Thiele, A branch and bound method for the general-shop problem with sequence dependent setup-times, OR Spektrum 18 (1996) 145–161.

[5] Y. Crama, V. Kats, J. van de Klundert, E. Levner, Cyclic scheduling in robotic flowshops, Annals of Operations Research 96 (2000) 97–124.

[6] J. Grabowski, E. Nowicki, S. Zdrzalka, A block approach for single machine scheduling with release dates and due dates, European Journal of Operational Research 26 (1986) 278–285.

[7] J. Hurink, S. Knust, Makespan minimization for flow-shop problems with transportation times and a single robot, Discrete Applied Mathematics 112 (2001) 199–216.

[8] J. Hurink, S. Knust, A tabu search algorithm for scheduling a single robot in a job-shop environment, Discrete Applied Mathematics 119 (2002) 181–203.

[9] A.S. Jain, S. Meeran, Deterministic job-shop scheduling: past, present and future, European Journal of Operational Research 113 (1999) 390–434.

[10] R.E. King, T. Hodgson, F.W. Chafee, Robot task scheduling in a flexible manufacturing cell, IIE Transactions 25 (1993) 80–87.

[11] H. Kise, On an automated two-machine flowshop scheduling problem with infinite buffer, Journal of the Operational Research Society of Japan 34 (1991) 354–361.

[12] S. Knust, Shop-scheduling problems with transportation, Ph.D. Thesis, Fachbereich Mathematik/Informatik Universität Osnabrück, 1999.

[13] P.J.M. van Laarhoven, E.H.L. Aarts, J.K. Lenstra, Job-shop scheduling by simulated annealing, Operations Research 40 (1992) 113–125.

[14] M.A. Langston, Interstage transportation planning in the deterministic flow-shop environment, Operations Research 35 (1987) 556–564.

[15] J.F. Muth, G.L. Thompson, Industrial Scheduling, Prentice Hall, Englewood Cliffs, NJ, 1963.

[16] E. Nowicki, C. Smutnicki, A fast tabu search algorithm for the job shop problem, Management Science 42 (1996) 797–813.

[17] B. Roy, B. Sussmann, Les problemes d'ordonnancement avec constraintes disjonctives, Note DS no. 9 bis, SEMA, Paris, 1964.