Computing 51, 95–110 (1993)

Computing

Constructive Heuristic Algorithms For The Open Shop Problem

H. Bräsel, T. Tautenhahn, and **F. Werner,** Magdeburg

Received December 2, 1992; revised March 26, 1993

Abstract — Zusammenfassung

Constructive Heuristic Algorithms for the Open Shop Problem. In this paper we consider constructive heuristic algorithms for the open shop problem with minimization of the schedule length. By means of investigations of the structure of a feasible solution two types of heuristic algorithms are developed: construction of a rank-minimal schedule by solving successively weighted maximum cardinality matching problems and construction of an approximate schedule by applying insertion techniques combined with beam search. All presented algorithms are tested on benchmark problems from the literature. Our computational results demonstrate the excellent solution quality of our insertion algorithm, especially for greater job and machine numbers. For 29 of 30 benchmark problems with at least 10 jobs and 10 machines we improve the best known values obtained by tabu search.

AMS Subject Classification: 90B35

Key words: Scheduling, open shop problems, heuristic algorithms.

Konstruktive Heuristiken für das Open Shop Problem. Mit dem Ziel der Minimierung der Gesamtbearbeitungszeit werden konstruktive Heuristiken für das open shop Problem betrachtet. Durch strukturelle Untersuchungen einer zulässigen Lösung werden zwei Arten von Heuristiken entwickelt: Konstruktion eines rangminimalen Bearbeitungsplanes durch sukzessives Lösen von gewichteten Matchingproblemen mit maximaler Kardinalität und Konstruktion einer Näherungslösung durch Anwendung von Einfügungstechniken kombiniert mit beam search. Die Verfahren werden an den aus der Literatur bekannten Benchmark Beispielen getestet. Die Resultate unserer Testrechnungen demonstrieren eindrucksvoll die Qualität unseres Einfügungsalgorithmus, insbesondere für wachsende Auftrags- und Maschinenzahl. Für 29 der 30 Benchmark Beispiele mit mindestens 10 Aufträgen und 10 Maschinen wird die mit Tabusuche ermittelte Näherungslösung verbessert.

1. Introduction

In this paper we consider the $[O//C_{max}]$ open shop problem: n jobs J_i with $i \in I = \{1,\ldots,n\}$ have to be processed on m machines M_j with $j \in J = \{1,\ldots,m\}$. The processing time t_{ij} of each operation (i,j) which represents the processing of J_i on M_j is given. We denote by O the set of operations with $t_{ij} > 0$. Both, the machine and job orders, can be chosen arbitrarily (open shop). The usual assumptions hold, i.e. each machine can process at most one job at a time and each job can be processed on at most one machine at the same time. Preemption of an operation is not allowed. Now we look for a feasible combination of the machine and job orders (schedule)

which minimizes the function $f = \max_{1 \le i \le n} \{C_i\} = C_{max}$ where C_i denotes the completion time of J_i.

In the case of arbitrary processing times, the problem $[O//C_{max}]$ is NP-hard [8]. If all $t_{ij} = 1$, the problem is polynomially solvable (see [1], [6], or [11]). There also exist polynomial algorithms for $n = 2$ or $m = 2$ and for the case that preemption of operations is allowed [6].

An exact solution of our open shop problem is possible by explicit or implicit enumeration of the schedules (see [2] and [7]). Clearly, we can apply this exact algorithm only for small n and m. In the case $n = 3$ and $m = 4$ there exist already 3.733.056 schedules. Therefore it is necessary to develop good heuristics. The quality of such approximation algorithms essentially depends on the structural analysis of the given problem and the corresponding feasible solution. Up to now only a few results have been obtained in this field.

Chen and Strusevitch [4] investigated a class of approximate solutions which they called dense schedules. A schedule is called dense if there exists an idle time on a machine only if there does not exist a job that waits for the processing on this machine. Let C^s_{max} and C^*_{max} be the schedule lengths of a dense schedule and of the optimal schedule, respectively. Then they proved for the $[O3//C_{max}]$ problem that $\frac{C^s_{max}}{C^*_{max}} \le \frac{5}{3}$ holds and that this bound is tight.

Taillard considered tabu search as an iterative heuristic and he gave computer results for benchmark problems in [10]. Bräsel and Tautenhahn presented constructive heuristics in [3].

In our paper we give some constructive heuristics which are based on a structural analysis of a schedule. In Section 2 we introduce some basic concepts. In Section 3 we derive a class of heuristics based on the generation of rank-minimal schedules.

In Section 4 we use the idea of a branch and bound algorithm for the construction of an approximate solution. By means of beam search we obtain very good results. Section 5 contains computer results of the developed heuristics where a comparison with the results of Taillard obtained by tabu search is included. Our paper ends with some concluding remarks in Section 16.

2. On the Mathematical Model

A schedule for the open shop problem can be represented by a graph $G = (V, E)$. Figure 1 shows an example for the graph G which represents a feasible schedule if G does not contain any cycle.

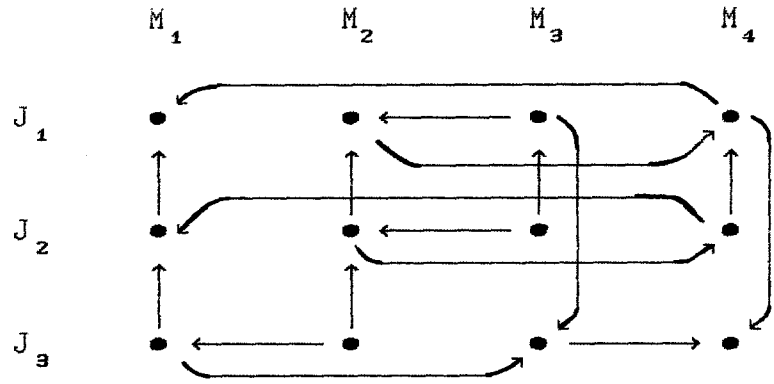


Figure 1. Graph $G = (V, E)$

Here V represents the set of operations $(i,j) \in I \times J$. The set E contains horizontal and vertical arcs which describe the machine and job orders. For instance, the machine order for job J_2 is $M_3 \to M_2 \to M_4 \to M_1$, and on machine M_3 we have the job order $J_2 \to J_1 \to J_3$.

We can describe a feasible schedule for the open shop problem also by the rank matrix $R(I \times J)$ of the graph G. For the graph given in Fig. 1, we have

$$R(I \times J) = [a_{ij}] = \begin{bmatrix} 5 & 3 & 2 & 4 \\ 4 & 2 & 1 & 3 \\ 2 & 1 & 3 & 5 \end{bmatrix}$$

Here a_{ij} gives the rank of operation (i,j) in the corresponding graph G. Notice that the machine and job orders can be directly obtained from $R(I \times J)$ by ordering the numbers rowwise (and columnwise, respectively) from the smallest integer to the largest one. Especially, we have the following property of $R(I \times J)$:

$$\text{For each } a_{ij} > 1, \text{ the integer } a_{ij} - 1 \text{ exists in row } i \text{ or in column } j. \qquad (1)$$

Note that $R(I \times J)$ is a special latin rectangle (see also [1]). For the sake of simplicity we identify in the following both the graph and the corresponding rank matrix by $R(I \times J)$.

Clearly, the above description can be also used for the case that the set O of operations is a subset of $I \times J$. In this case $R(O)$ is a partial rank matrix (partial schedule) which has property (1) and also one-to-one corresponds to a graph.

If we assign in the graph $R(I \times J)$ a vertex cost t_{ij} to each vertex (i,j), we can consider different objective functions on the set of all graphs. If we use the makespan criterion, the C_{max} value of a schedule is obtained by the length of a critical path in $R(I \times J)$.

A trivial lower bound for the makespan is given by

$$LB = \max\left\{\max_{1\le j\le m}\left(\sum_{i=1}^{n} t_{ij}\right), \max_{1\le i\le m}\left(\sum_{j=1}^{m} t_{ij}\right)\right\}. \tag{2}$$

3. A Heuristic Based on the Generation of Rank-Minimal Schedules

In this section we give different variants of a heuristic that generates rank-minimal schedules, i.e. the greatest value of the rank of all operations is minimal and equal to $\max\{n,m\}$. Let $h = \min\{n,m\}$. In our algorithm we consecutively determine h operations having the rank 1 in the heuristic solution, then h operations having rank 2 and so on. This is done by solving weighted bipartite maximum cardinality matching problems. We start with the complete bipartite graph G^* with the vertex set $I \cup J$ and then we use the following basic algorithm:

Algorithm 1: Generation of a rank-minimal schedule
begin $k := 1$;
  while $k \le \max\{n,m\}$ do
    begin determine a weighted maximum cardinality matching M (i.e. $|M| = h$) in G^* with a suitable objective function, the operations (i,j) contained in M obtain the rank k; delete all edges $(i,j) \in M$ in G^*; $k := k + 1$
    end
end.

If all $t_{ij} = 1$ then Algorithm 1 finds a schedule in which the largest rank is minimal. This follows from the fact that in this case Algorithm 1 is identical with an algorithm for the preemptive open shop problem $[O/pmtn/C_{max}]$ given by Gonzales and Sahni [6] because in this case no preemption is necessary. Because we use Algorithm 1 as a heuristic for the open shop problem with arbitrary processing times, we still have to discuss how to determine a weighted maximum cardinality matching M in each cycle. If we use a sum objective function, we can consider the problem

$$\sum\{t_{ij}/(i,j) \in M, |M| = h\} = \min! \qquad \text{or}$$

$$\sum\{t_{ij}/(i,j) \in M, |M| = h\} = \max!$$

Note that we have the classical assignment problem for $n = m$.

Another possibility is to choose the objective function

$$\max\{t_{ij}/(i,j) \in M, |M| = h\} = \min!$$

The idea is that, if any operation of rank k would start only when all operations of rank $k-1$ have been finished, it is useful to assign the same rank to such operations having a similar processing time. This can also be achieved by choosing the objective function

$$\min\{t_{ij}/(i,j) \in M, |M| = h\} = \max!$$

Because, when determining the operations with rank k, we already know all operations with a smaller rank, we can use the information about the already

determined partial schedule up to rank $k-1$. Hence, we can modify the objective functions by considering also the head r_{ij} of every operation. The head of the operation (i,j) is the time which is necessary to process all operations which are predecessor of (i,j) in the corresponding graph, i.e. the earliest time when (i,j) can start. In this way we get the objective functions

$$\min\{t_{ij} + r_{ij}/(i,j) \in M, |M| = h\} = \max! \quad \text{and}$$
$$\max\{t_{ij} + r_{ij}/(i,j) \in M, |M| = h\} = \min!$$

Example 1: Let $n = 4$, $m = 4$ and the following matrix T be given:

$$T = \begin{bmatrix} 85 & 23 & 39 & 55 \\ 85 & 74 & 56 & 78 \\ 3 & 96 & 92 & 11 \\ 67 & 45 & 70 & 75 \end{bmatrix}.$$ The trivial lower bound is 293.

Algorithm 1 yields the rank-minimal schedules R_1 (minimization) and R_2 (maximization) in the case of a sum objective function. We also give the schedule lengths and the percentage of relative deviation from the trivial lower bound for all constructed schedules.

$$R_1 = \begin{bmatrix} 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 1 & 4 & 3 & 2 \\ 3 & 2 & 4 & 1 \end{bmatrix} \quad \text{and} \quad R_2 = \begin{bmatrix} 1 & 4 & 3 & 2 \\ 3 & 2 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \end{bmatrix}$$
$f = 311$ (6.1%) $\qquad$ $f = 311$ (6.1%)

If we use the bottleneck objective functions, we obtain in the case of minimization R_3 and in the case of maximization R_4:

$$R_3 = \begin{bmatrix} 3 & 1 & 2 & 4 \\ 4 & 3 & 1 & 2 \\ 2 & 4 & 3 & 1 \\ 1 & 2 & 4 & 3 \end{bmatrix} \quad \text{and} \quad R_4 = \begin{bmatrix} 1 & 4 & 3 & 2 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 3 & 2 & 1 & 4 \end{bmatrix}.$$
$f = 304$ (3.8%) $\qquad$ $f = 312$ (6.5%)

Finally, the schedules R_5 and R_6 are obtained by applying the modified bottleneck objective functions in the case of minimization and of maximization, respectively:

$$R_5 = \begin{bmatrix} 4 & 2 & 3 & 1 \\ 2 & 4 & 1 & 3 \\ 1 & 3 & 4 & 2 \\ 3 & 1 & 2 & 4 \end{bmatrix} \quad \text{and} \quad R_6 = \begin{bmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}.$$
$f = 294$ (0.3%) $\qquad$ $f = 304$ (3.8%)

Remark: This algorithm ensures that all paths in $R(I \times J)$ according to a constructed approximate schedule have the same number $a = \max\{n, m\}$ of vertices. Let $t_1, \ldots, t_a$ be the processing times of all operations of the critical path. Then we have for the objective value C_{max} of our heuristic solutions

$$C_{max} \leq (t_1 + t_2) + (t_3 + t_4) + \cdots + (t_{a-1} + t_a) \text{ if } a \text{ is even or}$$

$$C_{max} \leq (t_1 + t_2) + (t_3 + t_4) + \cdots + (t_{a-2} + t_{a-1}) + (t_{a-1} + t_a) \text{ if } a \text{ is odd.}$$

Each of the $\left\lceil \frac{a}{2} \right\rceil$ pairs of processing times belongs to one row or one column. Therefore the optimal objective value C^* satisfies

$$\begin{aligned} \left\lceil \frac{a}{2} \right\rceil C^* &\geq \left\lceil \frac{a}{2} \right\rceil \max\{(t_1 + t_2), (t_3 + t_4), \ldots, (t_{a-1} + t_a)\} \\ &\geq (t_1 + t_2) + \cdots + (t_{a-1} + t_a) \\ &\geq C_{max} \end{aligned}$$

and hence we obtain $\dfrac{C_{max}}{C^*} \leq \left\lceil \dfrac{\max\{n, m\}}{2} \right\rceil$.

The implemented algorithms have a worst case complexity of $O(\max\{n, m\}^4)$ if sum objective functions have been used and $O(\max\{n, m\}^3 \log(\max\{n, m\})$ in the case of bottleneck objective functions, respectively.

4. An Insertion Algorithm

The heuristic considered in this section is a special restricted branch and bound algorithm which is based on an enumerative algorithm for schedules (see [2]). Let an operation set O and a partial schedule $R(O)$ be given. Now we have to insert an operation $(k, l) \in I \times J \setminus O$ into this partial schedule with the properties

- all precedence constraints between the operations of O are the same in both $R(O) = [a_{ij}]$ and $R(O \cup \{(k, l)\}) = [a^*_{ij}]$ and
- we obtain a new rank matrix, i.e. property (1) holds again.

For the description of the insertion we denote the subset of operations for the job k and the machine l by O_k and O^*_l, respectively. Furthermore, let $A_k = \{a_{ij}/(i,j) \in O_k\}$ and $A^*_l = \{a_{ij}/(i,j) \in O^*_l\}$. There exist three possibilities to insert (k, l):

(c1) Let $a^*_{kl} = 1$, that means l is the first machine in the machine order of job k and k is the first job in the job order on machine l;

(c2) Let $a^*_{kl} = b + 1$ with $b \in \{A_k \cup A^*_l\}$, that means: the operation (k, l) becomes a direct successor of one of the operations of job k or on machine l.

(c3) If there exists a pair of operations (k, w) and (v, l) with the property that there does not exist any path between them in the graph $R(O)$ we obtain:

- if $a_{kw} \leq a_{vl}$ we choose (k, l) as direct successor of (v, l) and direct predecessor of (k, w), that means $a^*_{kl} = a_{vl} + 1$ and $a^*_{kw} = a_{vl} + 2$.

- if $a_{k2} \geq a_{vl}$ we choose (k, l) as direct successor of (k, w) and direct predecessor of (v, l), that means $a^*_{kl} = a_{kw} + 1$ and $a^*_{vl} = a_{kw} + 2$.

Clearly, if $a_{kw} = a_{vl}$, both cases yield a new schedule. In all cases we have to modify all a_{ij} if (i,j) is a successor of (k, l) when determine the new rank matrix. In [2], it is shown that there does not exist any other possibility to obtain a new schedule. We illustrate the insertion of an operation by the following example:

Example 2: Let a partial schedule $R(O)$ be given in Fig. 2:

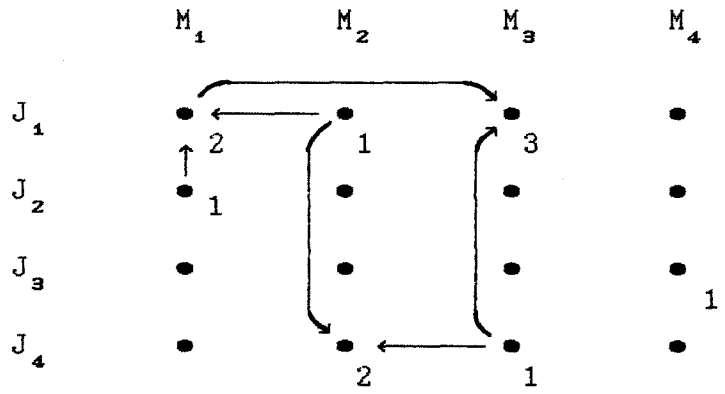


Figure 2. $R(O)$

The vertices are marked with the ranks of the operations. We assume that the next operation which has to be inserted is (2, 3). There exist 5 possibilities for this operation. We give the corresponding graphs and also the new rank of each operation.

(c1) yields $a^*_{23} = 1$, i.e. we get Fig. 3:

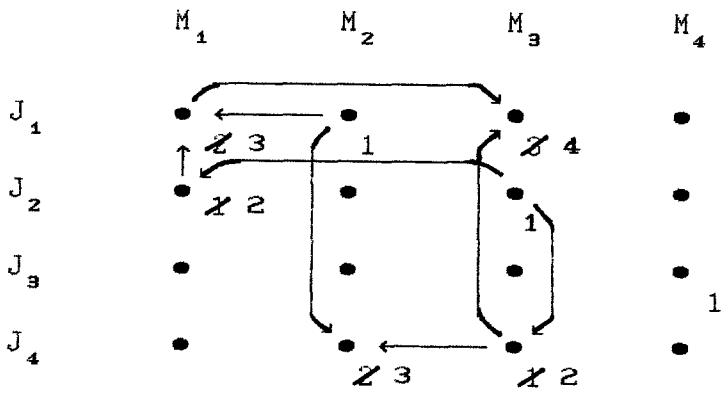


Figure 3. R_1

In the case (c2) we can choose $a^*_{23} \in \{2,4\}$. We obtain the acyclic graphs according to Figs. 4 and 5:

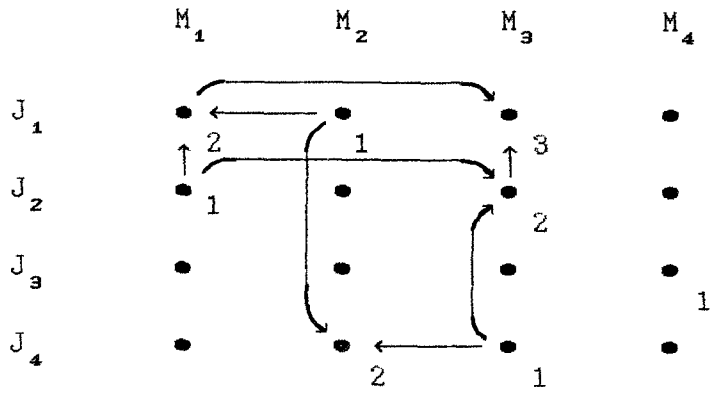


Figure 4. R_2

and

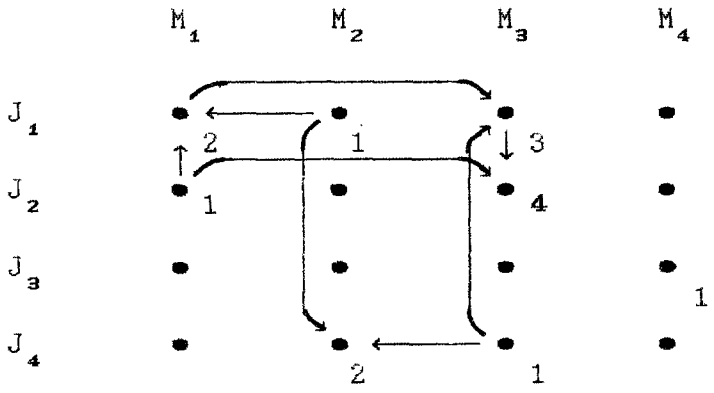


Figure 5. R_3

There does not exist any path between the operations $(2,1)$ and $(4,3)$ in $R(O)$. Therefore, the case (c3) yields Figs. 6 and 7:

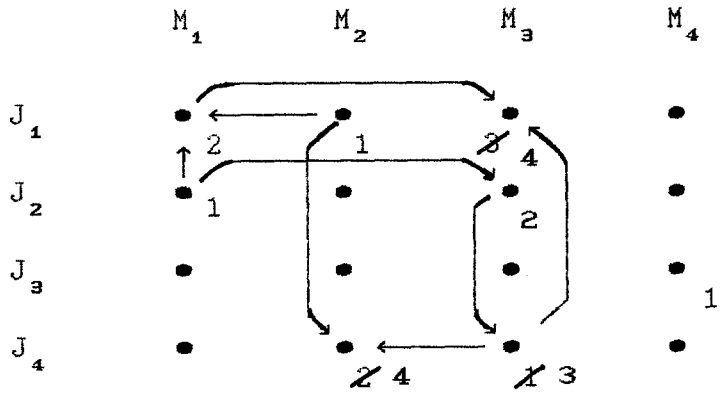


Figure 6. R_4

and

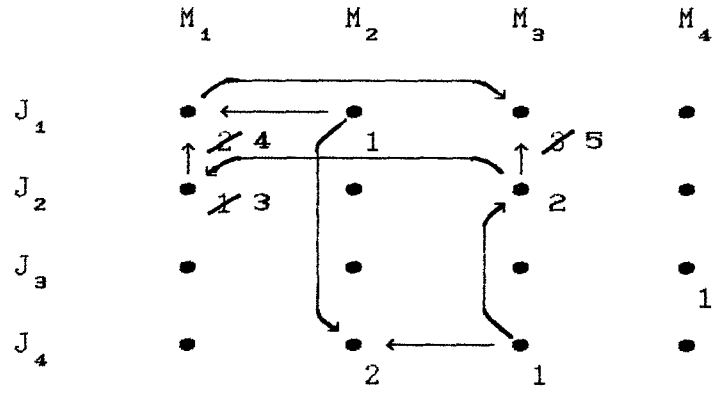


Figure 7. R_5

Let us consider the paths through the inserted operation (k, l). Obviously, the longest path on (k, l) in the case (c3) is longer than the longest paths on this operation in the cases (c1) and (c2). Therefore, we apply in our heuristic only the cases (c1) and (c2) for inserting the new operation.

The order in which the operations will be successively inserted into the schedule is called insertion order IO, $IO = \{(i_1,j_1),(i_2,j_2),\ldots,(i_{nm},j_{nm})\}$.

Firstly, we determine $h = \min\{n, m\}$ operations that obtain the rank 1 in the partial schedule. Starting with the first row in T, we determine the largest processing time

t_{1j_1} and then we exclude column j_1. Then we look for the largest processing time among all not excluded columns in row 2 and so on. So we obtain a feasible assignment consisting of h operations. The remaining operations are successively inserted in non-decreasing order, i.e.

$$t_{i_{h+1}j_{h+1}} \geq t_{i_{h+2}j_{h+2}} \geq \cdots \geq t_{i_{nm}j_{nm}}.$$

Clearly, the determination of IO requires $O(n \cdot m \cdot \log(n \cdot m))$ time.

In our algorithm we combine these insertion techniques with beam search. That means, a limited number of solution paths in the branching tree is investigated in parallel. Each partial schedule has a "father" and some "sons". The father is the immediate predecessor in the branching tree, i.e. the partial schedule in which the last operation had been inserted. The sons result from the insertion of the next operation. The number of parallel solution paths is given by the beamwidth p. We consider the following two variants of beam search:

INSERT1: In each step we select the p-best sons from the whole set, i.e. the selected sons must not have different fathers.

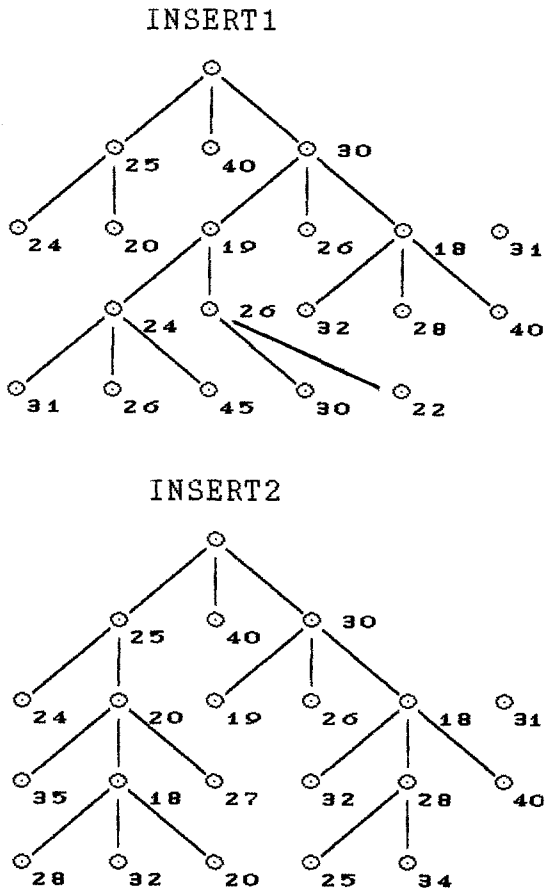
INSERT2: For each of the p fathers we select the best son, i.e. all sons have different fathers (as long as we do not have p fathers the first criterion is applied).

We still have to decide which sons are the best in each step. In order to do this, we assign the following costs to a partial schedule. If operation (k, l) has been inserted last, the corresponding costs of the resulting schedule are given by the longest path through the operation (k, l) in the graph $R(O')$ belonging to the resulting schedule $R(O')$ with $O' = O \cup \{(k, l)\}$.

Our experiments demonstrate that the above criterion is more suitable than taking the longest path in $R(O')$ as costs. For all possible insertions where the inserted operation does not belong to a longest path, there would be no sharp criterion to evaluate these insertions. However, applying our criterion, the costs are not necessarily monotonically non-decreasing with respect to the successive insertion of operations.

Example 3 demonstrates both types of beam search for $p = 2$. The numbers denote the costs of the corresponding schedules of a fictitous example.

Example 3:



In the following we denote by $SR = \{R_1^v, \ldots, R_p^v\}$ the set of fathers in step v. For each father R_μ^v we determine the set SR_μ^v of sons by inserting the next operation.

Then the algorithm is as follows:

Algorithm 2: Heuristic Insertion Algorithm
{Input: n, m, T, p;
Output: heuristic schedule $R(I \times J)$}

S0: determine the insertion order $IO = \{(i_1, j_1), \ldots, (i_{nm}, j_{nm})\}$;
$h := \min\{n, m\}$;
$O = \bigcup_{\mu=1}^{h} \{(i_\mu, j_\mu)\}$; $R(O) := [a_{ij}]$ with $a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in O \\ * & \text{otherwise} \end{cases}$;
$SR := \{R_1^1 = R(O)\}$; $v := 1$; $p^* := 1$;

S1: for $\mu := 1$ to p^* do
generate the sons of $R_\mu^\nu \in SR$ by inserting operation $(i_{h+\nu}, j_{h+\nu})$ according to (c1) and (c2) which form the sets SR_μ^ν;

S2: for all generated sons, i.e. for the set $S^* = \bigcup_{\mu=1}^{p^*} SR_\mu^\nu$, determine the costs, i.e. the lengths of the longest path in the corresponding graphs through $(i_{h+\nu}, j_{h+\nu})$ in each case;

S3: $p^* := \min\{p, |S^*|\}$;
select p^* generated sons by applying INSERT1 or INSERT2 which form the set SR;

S4: $\nu := \nu + 1$; if $\nu \leq m \cdot n - h$ then goto S1;

S5: determine the heuristic solution $R(I \times J) \in SR$ with the best C_{max} value.

Clearly, our insertion algorithm has the complexity $O(n^2m^2)$.

We only note that an alternative way to insert the operations consists in determining a topological enumeration of the operations, i.e. we consider such permutations of the operations of O that (i,j) is sequenced before (i^*,j^*) if in $R(O)$ a path exists from (i,j) to (i^*,j^*). However, in such a case a partial schedule may be represented by several different sequences of the operation.

5. Computational Results

All algorithms presented in this paper have been implemented in TURBO-PASCAL on a PS/2 computer in the 55 SX version. The test problems are either taken from the literature or randomly generated. We test only problems with $n = m$ (analogously to Taillard), i.e. we have to solve n assignments problems. Computational investigations have shown that such problems are harder solvable than the other ones.

As regards the determination of schedules with minimal rank by solving assignment problems, we applied the following variants:

random—we randomly generate a schedule with minimal rank;
summin, summax—solution of the corresponding assignment problems with sum objective function and minimization and maximization, respectively;
botmin, botmax—solution of the assignment problems with bottleneck objective function and minimization and maximization, respectively;
mbotmin, mbotmax—modified strategies of botmin and botmax (i.e. consideration of heads of operations).

Moreover, we tested both insertion variants INSERT1 (selection of p best schedules from all schedules of this step) and INSERT2 (each partial schedule is replaced by one of its sons in the branching tree). We applied the beamwidths $p = 1$, 2 and 3. Moreover, BestAP represents the best value of all the strategies for determining a schedule with minimal rank and BestINSERT describes the best result of all tested insertion methods.

Table 1 presents the results for the benchmark problems given by Taillard [10]. For each stated value of n and m in Table 1, 10 problems have been considered. The average percentage of deviation of the objective value from the trivial lower bound (2) and, in parentheses, the average computation time in seconds are included in the tables.

Table 1. Results for the benchmark problems given by Taillard

(n, m)	(4, 4)	(5, 5)	(7, 7)	(10, 10)	(15, 15)	(20, 20)
Random	25.08%	20.99%	23.73%	28.48%	24.19%	23.80%
Summin	11.33%	13.26%	11.35%	11.47%	9.02%	7.14%
	(0.01s)	(0.03s)	(0.11s)	(0.44s)	(2.09s)	(6.46s)
Summax	9.33%	15.05%	11.27%	11.01%	8.76%	7.21%
	(0.01s)	(0.01s)	(0.11s)	(0.44s)	(2.09s)	(6.46s)
Botmin	11.35%	14.32%	13.44%	11.80%	8.58%	5.93%
	(< 0.01s)	(0.01s)	(0.03s)	(0.11s)	(0.28s)	(0.71s)
Botmax	11.33%	13.84%	13.12%	14.70%	13.07%	10.06%
	(< 0.01s)	(0.01s)	(0.03s)	(0.11s)	(0.28s)	(0.71s)
Mbotmin	13.11%	11.53%	11.45%	10.04%	9.46%	9.39%
	(< 0.01s)	(0.01s)	(0.03s)	(0.11s)	(0.38s)	(0.94s)
Mbotmax	13.75%	12.45%	14.54%	13.82%	11.88%	11.28%
	(< 0.01s)	(0.01s)	(0.03s)	(0.11s)	(0.38s)	(0.94s)
BestAP	8.00%	9.22%	9.14%	8.84%	6.58%	5.32%
Insert1						
$p = 1$	6.70%	9.04%	5.54%	1.55%	0.32%	0.10%
	(0.17s)	(0.44s)	(2.45s)	(16.32s)	(154.11s)	(791.53s)
$p = 2$	5.48%	6.53%	4.18%	1.73%	0.56%	—
	(0.32s)	(0.91s)	(4.71s)	(31.97s)	(305.03s)	
$p = 3$	4.11%	5.40%	3.52%	1.45%	0.26%	—
	(0.46s)	(1.32s)	(7.01s)	(47.86s)	(461.90s)	
Insert2						
$p = 2$	5.24%	7.32%	4.06%	1.38%	0.32%	—
	(0.32s)	(0.90s)	(4.78s)	(32.16s)	(308.44s)	
$p = 3$	5.15%	6.01%	3.97%	1.11%	0.28%	—
	(0.47s)	(1.31s)	(7.11s)	(48.11s)	(461.50s)	
Best Insert	4.11%	5.08%	2.39%	0.62%	0.14%	0.10%

Considering the determination of schedules with minimal rank, none of the proposed strategies outperforms the remaining variants. Because of the low computation times it is recommendable to apply all versions and take the best obtained schedule as heuristic solution. Depending on n and m, the average deviation from the lower bound ranges from about 5% to 9% where the deviation is smaller for the greater values of n and m.

It can be seen from Table 1 that all variants of the insertion algorithm yield excellent results. As expected, for larger beamwidth the results are better. For $n = m = 15$, we have already less than 0.60% deviation from the lower bound for all tested

variants. The results for the (20, 20) problems are already for $p = 1$ so good that runs with a greater beamwidth are not necessary. Therefore, dashes are contained on the corresponding positions of Table 1. INSERT1 and INSERT2 yield similar results.

For the problems with at most 5 jobs, we know the optimal value by means of the branch and bound algorithm presented in [7]. So we additionally mention that the average deviation of the best value of the insertion variants from the optimal value is 1.61% for the (4, 4) problems and 3.36% for the (5, 5) problems.

Table 2 compares the best known values contained in the Taillard paper with the best objective value obtained by the variants of our insertion algorithm. We only mention that Taillard's best values have been obtained with expensive tabu search where the number of iterations ranges from $3 \cdot 10^5$ to $3 \cdot 10^6$ for these problems (for more details of tabu search see [5]).

Table 2. Best values for the benchmark problems with at least 10 jobs and 10 machines

	Tabu	Insert	Lower Bound
(10, 10)			
1	652	645	637
2	596	588	588
3	617	611	598
4	581	577	577
5	657	641	640
6	545	538	538
7	623	625	616
8	606	596	595
9	606	595	595
10	604	602	596
(15, 15)			
1	956	937	937
2	957	918	918
3	899	871	871
4	946	934	934
5	992	950	946
6	959	933	933
7	931	891	891
8	916	893	893
9	951	908	899
10	935	902	902
(20, 20)			
1	1215	1155	1155
2	1332	1244	1241
3	1294	1257	1257
4	1310	1248	1248
5	1301	1256	1256
6	1252	1209	1204
7	1352	1294	1294
8	1269	1173	1169
9	1322	1289	1289
10	1284	1241	1241

Tabu: best value with tabu search by Taillard
Insert: best value of a variant of our insertion algorithm

In Table 2 we stated the values for the (10, 10), (15, 15) and (20, 20) problems because in 29 of the 30 stated problems we have improved the best known values from [10]. Especially we mention that the average deviation from the lower bound of Taillard's best values is about 1.76% for the (10, 10) problems, 3.38% for the (15, 15) problems and 4.71% for the (20, 20) problems.

Table 3. Further results for 300 randomly generated problems

(n, m)	(10, 10)	(15, 15)	(20, 20)
Random	15.41%	19.92%	21.24%
Summin	5.66% (.044s)	5.98% (2.09s)	5.82% (6.46s)
Summax	5.66% (0.44s)	5.94% (2.09s)	5.76% (6.46s)
Botmin	5.56% (.011s)	5.29% (0.28s)	5.03% (0.71s)
Botmax	7.74% (0.11s)	9.21% (0.28s)	9.47% (0.71s)
Mbotmin	5.14% (0.11s)	5.99% (0.38s)	6.98% (0.94s)
Mbotmax	7.94% (0.11s)	8.85% (0.38s)	8.46% (0.94s)
BestAP	3.18%	3.94%	4.25%
Insert1			
$p = 1$	0.26% (16.90s)	0.10% (157.75s)	0.07% (794.99s)
$p = 3$	0.14% (49.71s)	—	—
Insert2			
$p = 3$	0.10% (50.08s)	—	—

This confirms that our constructive insertion algorithm yields better results for larger dimensions than extensive randomized iterative algorithms. Moreover, it seems not to be necessary to add an iterative algorithm to our constructive method because the obtained results are hardly improvable.

Table 3 contains further results for randomly generated problems with $n = m = 10$, 15 and 20. For each dimension 100 test problems have been generated. It can be seen from Table 3 that the average deviations from the lower bound are still smaller then for the benchmark problems from [10]. We assume that it is very difficult to generate problems with a greater difference between the trivial lower bound and the optimal objective value.

6. Concluding Remarks

All constructive heuristic algorithms of the open shop problem presented in this paper base on an analysis of the structure of a feasible combination of job and machine orders.

Our computer results show that for large n and m the described insertion algorithm yields excellent results. It seems not to be necessary to add a further algorithm (i.e. an iterative one), but up to now the worst-case analysis of the presented algorithm is an open question.

We also mention that a similar insertion algorithm for the job shop problem is given in [12].

For all polynomially solvable special cases of the open shop problem mentioned in the introduction the optimal objective value is equal to the trivial lower bound (2). Based on our computer results, we conjecture that for our open shop problem such a property asymptotically holds for large n and m. It would be an interesting topic for further research to prove such a convergence property theoretically.

References

[1] Bräsel, H.: Lateinische Rechtecke und Maschinenbelegung; Dissertation B, TU Magdeburg (1990).
[2] Bräsel, H., Kleinau, M.: On the number of feasible schedules of the open-shop problem—an applicaton of special Latin rectangles. Optimization *23*, 251–260 (1992).
[3] Bräsel, H., Tautenhahn, T.: Zur näherungsweisen Lösung eines open-shop Problems; Wiss. Z. TU Magdeburg *33*, 41–44 (1989).
[4] Chen, B., Strusevitch, V.: Worst-case analysis of dense schedules for the three machine open shop scheduling. Preprint, Rotterdam (1991).
[5] Glover, F.: Tabu Search—Part 1. ORSA J. Comput. *1*, 190–206 (1989).
[6] Gonzales, T., Sahni, S.: Open-shop scheduling to minimize finish time. J. Assoc. Comput. Mach. *23*, 665–680 (1976).
[7] Kleinau, U.: On some new methods for solving machine scheduling problems. Preprint, 16/91, TU Magdeburg (1991).
[8] Lageweg, B. J., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G.: Computer aided complexity classification of deterministic scheduling problems. Department of Operations Research, Statistics and System Theory, Report BW 138/81 (1981).
[9] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B.: Sequencing and scheduling: algorithms and complexity. Department of Operations Research, Statistics and System theory, Report BS-R8909 (1989).
[10] Taillard, E.: Benchmarks for basic scheduling problems. ORWP *89/21* (1989).
[11] Taneav, V. S., Sotskov, Y. N., Strusewitch, V. A.: Theory of scheduling—multistage systems. Nauka, Moskau (1989) (in Russian).
[12] Werner, F., Winkler, A.: Insertion techniques for the Heuristic solution of the job shop problem. to appear in Discrete Appl. Math.

Heidemarie Bräsel
Thomas Tautenhahn
Frank Werner
Technische Universität "Otto von Guericke"
Fakultät für Mathematik
PSF 4120
D-39016 Magdeburg
Federal Republic of Germany