On the set of solutions of the open shop problem[★]

Heidemarie Bräsel, Martin Harborth, Thomas Tautenhahn and Per Willenius
Faculty of Mathematics, PSF 4120, Otto-von-Guericke-Universität Magdeburg, D-39016 Magdeburg, Germany
E-mail: {braesel;harborth;tautenhahn;willenius}@mathematik.uni-magdeburg.de

In the classical open shop problem, n jobs have to be processed on m machines, where both job orders and machine orders can be chosen arbitrarily. A feasible (i.e., acyclic) combination of all job orders and machine orders is called a (multi-) sequence. We investigate a set of sequences which are structurally optimal in the sense that there is at least one optimal sequence in this set for each instance of processing times. Such sequences are called irreducible. Investigations about irreducible sequences are believed to provide a powerful tool to improve exact and heuristic algorithms. Furthermore, structural properties of sequences are important for problems with uncertain processing times.

We prove necessary and sufficient conditions for the irreducibility of a sequence. For several values of n and m, we give the numbers of all sequences, of the sequences satisfying each of these conditions and of the irreducible sequences, respectively. It turns out that only a very small fraction of all sequences is irreducible. Thus, algorithms which work only on the set of irreducible sequences instead of the set of all sequences can potentially perform much better than conventional algorithms.

Keywords: open shop, irreducible sequence, enumerational results

AMS subject classification: Primary 90B35; Secondary: 90C27, 05A15

1. Introduction

We consider a nonpreemptive open shop problem which can be stated as follows. There are given $n \geq 2$ independent jobs $J_1,\ldots,J_n$ and $m \geq 2$ machines $M_1,\ldots,M_m$. Each job J_i has to be processed on each machine M_j for a processing time p_{ij} which is known in advance. Such a processing of a job J_i on a machine M_j is called operation (i, j). Each job can be processed on at most one machine at a time and each machine can process at most one job at a time. All job orders (i.e., the order in which a certain machine processes the corresponding jobs) and all machine orders (the order in which

[★] Supported by INTAS (Project 93-257), by DFG (Project ScheMA), and by German State Sachsen-Anhalt.

a certain job is processed on the corresponding machines) can be chosen arbitrarily. The completion time of a job J_i, i.e., the time when the last operation of this job is completed, is denoted by C_i. We have to find a schedule which minimizes the make-span $C_{\max} = \max_i\{C_i\}$.

For arbitrary processing times, this problem is polynomially solvable in the case of two machines (see Gonzalez and Sahni [13]). For three machines, the problem is binary NP-hard and it becomes unary NP-hard if the number of machines is part of the input (see Garey and Johnson [11]).

In practice, open shop problems usually occur in testing components of complex devices or in doing maintenance work, for instance on cars. In this area, it is often the case that the exact processing times are erroneous, difficult to find out in advance or simply unknown. Thus, it appears to be useful to study structural properties of schedules which do not depend on the given processing times.

In the literature, such structural investigations were done by Ashour [2], Akers and Friedman [1], who considered the job shop problem with two jobs, and Conway et al. [9], who established structural properties of "potentially optimal" schedules for a flow shop problem.

To characterize a solution of the open shop problem with a regular criterion, i.e., an objective function which is nondecreasing in each job completion time C_i, it is sufficient to give only a feasible combination of job orders and machine orders. A schedule can be determined from these orders by performing each operation as early as possible with respect to its predecessors (semiactive schedule). In the following, we will call such a feasible combination of all job orders and machine orders a (multi-) *sequence*.

Bräsel and Kleinau [7] introduced a partial order "$\succeq$" over the set of sequences with the property that $A \succeq B$ implies $C_{\max}(A) \geq C_{\max}(B)$ for all possible instances of processing times, where $C_{\max}(A)$ denotes the makespan of the semiactive schedule specified by sequence A. The local minima according to this partial ordering, i.e., the "structurally optimal" sequences, are called *irreducible sequences*. The set of irreducible sequences for the open shop problem with n jobs and m machines is denoted by $S^I_{n,m}$. In [15], special classes of irreducible sequences were established. Furthermore, it was proved that $|S^I_{n,2}| = n!(n-1)$ holds. For several small formats up to $n = 3$, $m = 4$, Bräsel and Kleinau [6] gave the total number of sequences. For $m = 2$, this number is described by an explicit formula. In this case, the ratio between the number of irreducible sequences and the number $|S_{n,2}|$ of all sequences is

$$\frac{|S^I_{n,2}|}{|S_{n,2}|} = \frac{n!\,(n-1)}{n!\left(n! + \sum_{k=1}^{n} \frac{n!}{k!}\binom{n}{k}\right)}, \quad \text{i.e.,} \quad \lim_{n\to\infty} \frac{|S^I_{n,2}|}{|S_{n,2}|} = 0.$$

An irreducibility test given by Kleinau [15] needs exponential time. In the case that the underlying operation set has a tree structure, a polynomial irreducibility test is given by Tautenhahn [16].

In this paper, we study the set of all sequences and its structural properties for complete operation sets. We consider several necessary conditions for irreducibility which can be tested in polynomial time. We give enumerational results on the sets of all sequences, of irreducible sequences and of sequences satisfying these special conditions. Our investigations are based on an efficient enumeration algorithm for sequences. Besides the obtained numbers for the various classes of sequences which are of combinatorial interest, the technique used for the enumeration gives us a deeper insight into the sequences and especially into the concept of irreducibility.

2. Basic concepts

A combination of job orders and machine orders can be described by a digraph $SG = (V, E)$. The vertex set $V = \{1,,\dots,n\} \times \{1,\dots,m\}$ corresponds to the set of operations. The arc set consists of two subsets $E = E_{MO} \cup E_{JO}$ representing the machine orders and the job orders, respectively. An arc from a vertex (i,j) to a vertex (i,l) exists in E_{MO} if and only if operation (i,j) is a (not necessarily direct) predecessor of operation (i,l) in the machine order of job J_i. Analogously, an arc from a vertex (i,j) to a vertex (k,j) exists in E_{JO} if and only if operation (i,j) is a predecessor of operation (k,j) in the job order of machine M_j. A vertex (i,j) is called a source (sink) if there is no arc ending (starting) in (i,j). A combination of job orders and machine orders is feasible if and only if the graph SG does not contain any cycle. In this case, the graph SG is called a *sequence graph*.

In the following, we will represent sequences using the block matrices model introduced by Bräsel [3]. A sequence graph can be described by its rank matrix $A = (a_{ij})$. Each element a_{ij} is equal to the number of vertices on a longest path in SG from a source to vertex (i,j). As was observed in [3], every matrix A with the properties that

- the entries in each row and column are pairwise distinct positive integers, and
- for each entry $a_{ij} > 1$ there exists an entry $a_{ij} - 1$ in row i or in column j

is the rank matrix of a uniquely defined sequence graph $SG(A)$, and vice versa, the rank matrix of each sequence graph meets the conditions above. So, there is a one-to-one correspondence between matrices satisfying these conditions and sequences. For this reason, every such matrix A will be shortly referred to as a sequence.

Moreover, the machine orders can be written as a matrix $MO = (mo_{ij})$, which is the rank matrix of the graph $G_{MO} = (V, E_{MO})$, i.e., $mo_{ij} = k$ means that operation (i,j) is the kth operation of job J_i. Analogously, we define a matrix $JO = (jo_{ij})$ to be the rank matrix of the graph $G_{JO} = (V, E_{JO})$. Whenever possible without causing confusion, we will use concepts describing a sequence graph $SG(A)$ (like source, sink and so on) also for its rank matrix A and vice versa. Especially, we will call a set of operations in a sequence A a path if the corresponding vertices are on a common path in $SG(A)$.

The *reversed sequence* $\overline{A}$ of A is constructed from A by reversing the orientation of all arcs in $SG(A)$. Clearly, $C_{\max}(A) = C_{\max}(\overline{A})$ holds.

To motivate the concept of reducibility, consider the following two sequences:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 1 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}.$$

Whenever certain operations are on a common path in $SG(B)$, these operations are also on a common path in $SG(A)$. For instance, the operations $(1, 2) \rightarrow (1, 1) \rightarrow (2, 1)$ form a path in $SG(B)$, i.e., job J_1 is first processed on machine M_2, then on M_1 and after this operation is finished, M_1 processes job J_2. In $SG(A)$, we find these operations on the path $(1, 1) \rightarrow (1, 2) \rightarrow (2, 2) \rightarrow (2, 1)$. In a schedule according to sequence B, operations $(1, 1)$ and $(2, 2)$ may be processed in parallel, in a schedule according to sequence A this is not possible. In this sense, A is more restrictive than B. Independently of the actual processing times $C_{\max}(A) \geq C_{\max}(B)$ holds.

A sequence A is called *reducible* to a sequence B on the same set of operations, written $A \succeq B$, if and only if for each path W_1 in $SG(B)$ covering a vertex set V_{W_1} there exists a path W_2 in $SG(A)$ covering a vertex set V_{W_2} such that $V_{W_1} \subseteq V_{W_2}$ holds. It is easy to see that the relation "$\succeq$" is transitive. Note that $A \succeq B$ implies $\overline{A} \succeq B$, $A \succeq \overline{B}$, and $\overline{A} \succeq \overline{B}$. Clearly, if $A \succeq B$ holds, then $C_{\max}(A) \geq C_{\max}(B)$ is fulfilled for each given instance of processing times.

We call two sequences A and B with $A \succeq B$ and $B \succeq A$ *similar* and we denote this by $A \simeq B$. The relation "$\simeq$" is an equivalence relation. Note that $A \simeq B$ does not imply $A = B$ or $A = \overline{B}$. For example, all sequences with the property that there exists a single path covering the whole operation set are pairwise similar. Clearly, $A \succeq B$ implies $C_{\max}(A) = C_{\max}(B)$.

A sequence A is called *strongly reducible* to a sequence B, if and only if $A \succeq B$ and not $A \simeq B$ holds. We write $A \succ B$ and in this case there exists at least one path W in $SG(A)$ covering an operation set V_W such that there does not exist a path in $SG(B)$ covering all operations of V_W. Clearly, if for a given matrix of processing times the path W is the unique critical path in A with $p_{ij} > 0$ for all $(i, j) \quad V_W$, then the inequality $C_{\max}(A) > C_{\max}(B)$ holds.

We call a sequence A *irreducible* if there does not exist a sequence B with $A \succ B$. This means the irreducible sequences are the local minima with respect to the partial order "$\succ$". Summarizing, we can state

Lemma 1. For the open shop problem with n jobs, m machines and the minimum makespan criterion, at least one optimal sequence of any instance is contained in the set $S^I_{n,m}$ of irreducible sequences.

Let SG^* denote the transitive closure of the sequence graph SG and let $\widehat{SG}^*$ denote the underlying graph obtained by ignoring the orientations of all arcs of SG^*. The following theorem by Bräsel et al. [4] basically states that the path structure of a sequence is completely determined by SG^*.

Theorem 2. A sequence A is reducible to a sequence B if and only if $\widehat{SG(B)}^* \subseteq \widehat{SG(A)}^*$ holds.

This theorem allows us to decide in polynomial time whether a given sequence is reducible to another one. Namely, instead of considering the operation sets of all maximal paths in A, we only have to verify whether for each pair of operations which are on a common path in A there exists a path in B covering both operations.

Later on, we will employ the concept of lexicographic order over a set of matrices. Let $A = (a_{ij})$ and $B = (b_{ij})$ be two $n \times m$ matrices. Matrix A is *lexicographically less than* B, written $A < B$, if there exists a pair (i, j), $1 \leq i \leq n$, $1 \leq j \leq m$, such that the following holds:

- $a_{ij} < b_{ij}$, and
- $a_{kl} = b_{kl}$ for all pairs (k, l) with $1 \leq k < i$ and $1 \leq l \leq m$ or $k = i$ and $1 \leq l < j$.

3. Isomorphisms of sequences

In this section, we define three different types of isomorphism for sequences in order to get a more detailed understanding of equivalent sequence properties.

Two sequences A and B are *permutation isomorphic*, denoted by $A \sim_P B$, if there is an ordered pair (ϱ, π) of row permutation and column permutation such that B is obtained if (ϱ, π) is applied to A.

Furthermore, two sequences A and B are *graph isomorphic* if their associated sequence graphs $SG(A)$ and $SG(B)$ are isomorphic in terms of graph theory. We write $A \sim_G B$ for two graph isomorphic sequences A and B. Because of the special structure of sequence graphs, two sequences are graph isomorphic if and only if one can be transformed into the other by permuting rows, permuting columns, and reflecting in the main left-to-right diagonal (see Bräsel et al. [5]). The *transposition* $t \quad \{{}^T, 1\}$ denotes if a reflection in the main left-to-right diagonal is applied or not. Therefore, a graph isomorphism can be characterized by an ordered triple (ϱ, π, t), where ϱ is a row permutation, π is a column permutation, and t is a transposition.

Recall that the reversed sequence $\overline{A}$ of a sequence A is the sequence according to the graph obtained by reversing all arc orientations in $SG(A)$. Then the *reversion* $r \quad \{{}^-, 1\}$ is defined to be the operation which shows if a sequence is reversed or not. Finally, two sequences A and B are *structure isomorphic*, written $A \sim_S B$, if there is an ordered quadruple (ϱ, π, t, r) of row permutation, column permutation, transposition and reversion, respectively, which results in B if it is applied to A.

Each of the relations "$\sim_P$", "$\sim_G$", and "$\sim_S$" is an equivalence relation which partitions the set of sequences into *isomorphism classes*. Obviously, $A \sim_P B$ implies $A \sim_G B$, and $A \sim_G B$ implies $A \sim_S B$.

The sequences of a permutation isomorphism class are equivalent in the sense that they are the same if we apply an arbitrary renumbering of the jobs and of the machines. The sequences of a graph isomorphism class are equivalent in the sense that besides this renumbering of the jobs and of the machines, the jobs are assumed to be interchangeable with the machines. Finally, the sequences of a structure isomorphism class are equivalent in the sense that in addition to the already mentioned isomorphisms, we also obtain equivalent sequences by reversing each job order and each machine order.

As defined in section 2, a sequence A is irreducible if there is no sequence $B \not\doteq A$ with the property that for each path W_1 with vertex set V_{W_1} in $SG(B)$ there exists a path W_2 with vertex set V_{W_2} in $SG(A)$ satisfying $V_{W_1} \subseteq V_{W_2}$. For each member of an isomorphism class, the structure of the underlying sequence graph is the same with respect to the length of its paths. Therefore, irreducibility is invariant within each isomorphism class, i.e., if $A \sim_{\cdot} B$ holds for two sequences A and B, then A is irreducible if and only if B is irreducible, where "$\sim_{\cdot}$" stands for one of the isomorphism types "$\sim_P$", "$\sim_G$" and "$\sim_S$". This fact is important for the enumeration of irreducible sequences since we can reduce the computation time by considering only one sequence for each isomorphism class.

For each isomorphism type, the set of the corresponding isomorphisms forms a group, namely the groups $S_n \times S_m$, $S_n \times S_m \times \mathcal{Z}_2$, and $S_n \times S_m \times \mathcal{Z}_2 \times \mathcal{Z}_2$ for permutation isomorphism, graph isomorphism, and structure isomorphism, respectively, where S_n is the symmetric group on n letters and $\mathcal{Z}_2$ is the cyclic group of order two.

To compute the total number of sequences from a set of representatives, let us recall some concepts from basic algebra (see Burnside [8]). Let G be a group of elements that act on a finite set X. For each element $x \quad X$, the set

$$G(x) = \{g(x) \mid g \quad G\} \tag{1}$$

is called the *orbit* of x. Furthermore, for each element $x \quad X$, the *stabilizer* of x is the set

$$G_x = \{g \quad G \mid g(x) = x\}. \tag{2}$$

Clearly, the stabilizer of x is a subgroup of G, and it is well known that

$$|G(x)|\,|G_x| = |G| \tag{3}$$

holds.

Now let G be the group of isomorphisms of one type defined above and let $X = S_{n,m}$ be the set of all $n \times m$ sequences. For each $A \quad S_{n,m}$, its orbit $G(A)$ is the isomorphism class containing A, and the elements of the stabilizer G_A are the automorphisms of A.

If the number of distinct automorphisms is known for a sequence A, we are able to give the cardinality of its isomorphism class $G(A)$ by (3). Thus, given a system of distinct representatives for the isomorphism classes, the total number of sequences $|S_{n,m}|$ can easily be computed by the sum over all isomorphism classes:

$$|S_{n,m}| = \sum_{A \in R_{S_{n,m}}} |G(A)| = \sum_{A \in R_{S_{n,m}}} \frac{|G|}{|G_A|}, \tag{4}$$

where $R_{S_{n,m}}$ is a system of distinct representatives for the isomorphism classes in $S_{n,m}$. For $n \neq m$, the number $|G|$ is equal to $n!m!$, $n!m!$, and $2n!m!$ according to permutation isomorphism, to graph isomorphism, and to structure isomorphism, respectively (for $n = m$, we have $|G| = n!m!$, $2n!m!$, and $4n!m!$, respectively).

Example. Consider the following sequences under the action of the group of permutation isomorphisms:

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}.$$

The orbit $G(A)$ of the sequence A is the set $\{A, B\}$. The stabilizer G_A consists of the automorphisms ((2, 1), (2, 1)) and the identity ((1, 2), (1, 2)). The stabilizer G_C contains only the identity.

4. Enumeration of machine orders

The basic idea of our enumeration is to consider machine orders and job orders separately, and to apply isomorphism tests already on the set of machine orders. In this section, we give both an enumeration algorithm for sets of machine orders and theoretical enumerational results.

As mentioned above, a set of machine orders is described by an $n \times m$ matrix MO which contains in each row a permutation of the numbers $1,\dots,m$. For machine order matrices MO, we can apply the concepts of permutation isomorphism and structure isomorphism with the modification that the transposition is omitted since a transposed MO is not a machine order matrix. Assume that (ϱ, π, r) is an isomorphism which maps a machine order matrix MO^1 to a machine order matrix MO^2. Then (ϱ, π, r) defines a bijection between the sets of sequences corresponding to MO^1 and MO^2, respectively, mapping each sequence A^1 to an isomorphic sequence A^2. Thus, we have to enumerate a set of sequences for only one of these machine order matrices.

A machine order matrix MO is called *reduced* if the entries in the first row are in the natural order $1, 2,\dots, m$, and each row vector MO_i with $i > 1$ is lexicographically not less than row MO_{i-1}. Obviously, each class of pairwise permutation isomorphic matrices MO contains at most k different reduced matrices MO, where k is the number of different rows, since each of these rows can be used as the first row and the permutation of rows and columns is determined uniquely by this choice.

We call an automorphism (ϱ, π) of a matrix MO *nontrivial* if the column permutation π is not the identity, which means that a row $i > 1$ different from row 1 becomes the first row.

Lemma 3. For given n and m, there exists an MO of format $n \times m$ with a nontrivial automorphism if and only if n is divisible by some q, with $1 < q \leq m$.

Proof. Assume a nontrivial automorphism (ϱ, π) with row permutation ϱ and column permutation π exists. Then π contains at least one cycle π_0 of length $q > 1$. Consequently, q is the order of π_0 in the symmetric group S_m. Let MO_i denote the vector of entries in row i. We have $MO_{\varrho(i)} = \pi(MO_i)$ for all i since (ϱ, π) is an automorphism. Since for each i all entries of MO_i are different, we have $\pi(MO_i) \neq MO_i$, which implies that $\varrho(i) \neq i$ holds. This means that, unlike π, ϱ cannot have any fixed points. Let r be the length of the cycle in ϱ which contains i. Then $\varrho^r(i) = i$ and, consequently, $\pi^r(MO_i) = MO_i$, i.e., π^r is the identity. Thus, r must be a multiple of q. Applying the same argument for all rows i, each cycle in ϱ has a length which is a multiple of q. Since n is the sum of all cycle lengths, n is also a multiple of q.

Vice versa, for $n = pq$ and $q \leq m$, a matrix $MO = (mo_{ij})$ with a nontrivial automorphism is given by

$$mo_{ij} = \begin{cases} ((\lceil i/p \rceil + j - 2) \bmod q) + 1 & \text{for } j \leq q, \\ j & \text{for } j > q. \end{cases}$$

□

The number of different machine order matrices MO is $(m!)^n$. However, for the enumeration we are mainly interested in the number $f(n, m)$ of permutation isomorphism classes of machine order matrices of format $n \times m$.

Lemma 4. It holds that

$$\frac{1}{n}\binom{m! + n - 2}{n - 1} \leq f(n, m) \leq \binom{m! + n - 2}{n - 1}.$$

Proof. The number of reduced MO's is equal to the number of possible combinations of $n - 1$ elements out of a set of $m!$ with allowed repetition which is $\binom{m!+n-2}{n-1}$. As mentioned above, in each isomorphism class there are at most n reduced MO's. □

Theorem 5. If n is not a multiple of any p, with $1 < p \leq m$, then

$$f(n, m) = \sum_{k=1}^{n} \binom{m! - 1}{k - 1}\binom{n - 1}{k - 1}\frac{1}{k}.$$

Proof. Consider the number of reduced MO's having exactly k pairwise distinct row vectors. The first row is always the identity, thus there are $\binom{m!-1}{k-1}$ possibilities to choose this set of k row vectors (recall that each row contains a permutation of the numbers $1, \ldots, m$). Furthermore, we have to assign to each of these chosen row vectors

π_r, $r = 1,\dots,k$, a frequency value $h_r \geq 1$ which describes how many rows of the matrix are equal to π_r. Obviously, $n = h_1 + \cdots + h_k$ must be satisfied. The number of ordered partitions of n into k positive summands is known to be equal to $\binom{n-1}{k-1}$. Thus, there are exactly $\binom{m!-1}{k-1}\binom{n-1}{k-1}$ reduced matrices MO having exactly k different rows.

As already stated, we can get k reduced MO's in each isomorphism class by using each of the k different row vectors as the first row (with appropriate row and column permutations). According to lemma 3, these k reduced MO's are all different. Thus, the number of isomorphism classes of MO's with k different rows is $1/k$ times the number of reduced MO's with k different rows. The statement follows by summation over k. □

Theorem 6. It holds that

$$f(n,m) = \frac{1}{m!}\sum_{k\,|\,n} n_{km}\binom{\frac{m!}{k} + \frac{n}{k} - 1}{\frac{n}{k}},$$

where n_{km} is the number of permutations in the symmetric group S_m having order k.

Proof. The theorem is a direct application of Burnside's lemma (see [8]) which relates the number of orbits with respect to a group acting on a finite set to the number of fixed points. More precisely, let G be a group of elements that act on a finite set X and let n^* be the number of orbits. Then

$$n^* = \frac{1}{|G|}\sum_{g \in G} |\{x \in X : g(x) = x\}|$$

holds.

Let X be the set of matrices MO where each row vector MO_i, $i > 1$, is lexicographically not smaller than the previous row vector MO_{i-1} and let G be the symmetric group S_m where each element $\pi \in S_m$ is interpreted as a permutation π of the columns of a matrix MO. For each such π, an according row permutation ϱ is automatically given by π and MO itself.

If the permutation π has order k in the symmetric group S_m, then the vectors MO_i, $\pi(MO_i),\dots,\pi^{k-1}(MO_i)$ are all different. Thus, if MO is not changed by applying π, it contains either all of these vectors or none of them. The number of different MO's which are not changed by π is the number of combinations of n/k out of $m!/k$ such possible sets of row vectors with repetition allowed. □

The same approach can be used to give the number of isomorphism classes with respect to structure isomorphism. In this case, we have to choose $G = S_m \times Z_2$. However, if a given automorphism includes reversing the machine orders, the orbits of single row vectors may have different lengths and so the formula becomes much more complicated.

For the enumeration algorithm, we need one representative from each set of structure isomorphic matrices MO in order to compute a corresponding set of sequences. To accomplish this, we combine an incrementing procedure for the matrix MO with a minimality test which identifies the representative.

As a representative of an isomorphism class, we choose the matrix MO_{min} which is lexicographically minimal in this class. Clearly, this MO_{min} is reduced. Thus, to test whether a given machine order MO is the minimum of its isomorphism class, we have to compare MO with all reduced machine order matrices isomorphic to MO. These are the matrices MO^i constructed from MO by putting the ith row on top of the matrix and applying an appropriate permutation of the columns and the other rows and the matrices $\overline{MO'}^{\,i}$ which are constructed from MO^i by reversing the machine orders of all jobs. Clearly, this can be done in $O(n^2 m \log n)$ time, but the average time needed is much less, because the decision whether MO is lexicographically less than MO^i can usually be made after reading only the beginning of the second rows of both matrices (note that by using pointers, we do not have to compute MO^i explicitly).

Let us now describe the incrementing procedure. Let $a_1, \ldots, a_m$ be a row vector of MO. If $a_1 > a_2 > \cdots > a_m$ holds, this permutation is lexicographically maximal and it cannot be incremented. Otherwise, there exists an index i with $a_{i-1} < a_i$ and $a_i > a_{i+1} > \cdots > a_m$. Choose $k \geq i$ as the greatest index with $a_k > a_{i-1}$, i.e., a_k is the smallest number among $a_1, \ldots, a_m$ which is greater than a_{i-1}. Then we get the lexicographically succeeding permutation by reversing the order of $a_1, \ldots, a_m$ and swapping a_{i-1} and a_k, i.e., the permutation is

$$a_1, \quad , a_{i-2}, a_k, a_m, \quad , a_{k+1}, a_{i-1}, a_{k-1}, \quad , a_i .$$

It is easy to see that this procedure can be carried out in $O(m)$ time. The aggregated time complexity is even a constant, because we have to update any k-last element of the permutation only once in $(k-1)!$ calls of the procedure and $\sum_{k=1}^{\infty} k/(k-1)! = 2e = O(1)$.

Given a reduced MO, the lexicographically succeeding reduced MO is generated in the following way. First we determine the greatest i such that row MO_i can be incremented. This row is incremented as explained above. Then rows $MO_{i+1}, \ldots, MO_n$ are replaced by copies of the incremented vector MO_i. It is easy to see that the time complexity of this procedure is $O(nm)$ in the worst case and $O(n)$ in the average case. Thus, counting all qualitatively different matrices MO by enumeration can be done on a PC Pentium up to $n = 6$, $m = 5$ in a few seconds.

The number of classes of MO's with respect to permutation isomorphism and structure isomorphism are given in tables 1 and 2, respectively.

5. Enumeration of sequences

To enumerate the set of sequences corresponding to a given set of machine orders, we employ a modified version of the insertion method given by Bräsel and Kleinau [6].

Table 1

Number of pairwise nonisomorphic machine order matrices according to permutation isomorphism.

$n\backslash m$	2	3	4	5	6
2	2	5	17	73	398
3	2	10	111	2467	86787
4	3	24	762	76044	15688744
5	3	42	4095	1876255	2270743529
6	4	83	19941	39096565	274382326290
7	4	132	84825	703593825	28457281936435
8	5	222	329214	11169676185	2586055570098800
9	5	335	1168740	158855852180	209183155674562575

Table 2

Number of pairwise nonisomorphic machine order matrices according to structure isomorphism.

$n\backslash m$	2	3	4	5	6
2	2	4	13	45	230
3	2	7	67	1269	43767
4	3	16	434	38356	7854456
5	3	26	2175	939395	1135495745
6	4	50	10385	19556801	137193369114
7	4	76	43353	351827297	14228666657843
8	5	126	167102	5585002649	1293028139377488
9	5	185	589648	79428476802	104591581641412531

A sequence is built by successively inserting operations into a partial sequence at all possible positions.

We store the sequence graph in the form of a successor list for the operations of each job and machine, which allows a very fast access. The machine orders of all jobs are already given, the job orders are initially empty, i.e., there are no arcs in E_{JO}. Additionally, we compute the rank matrix A of the sequence graph to use it for both the cycle test and to have a convenient output. At the beginning, A is the rank matrix of the graph $G_{MO} = (V, E_{MO})$ containing only the arcs representing the given machine orders.

There are two possibilities to insert an operation (i, j) into the job order of machine M_j:

- Insert (i, j) as a source into the job order of machine M_j. The rank value a_{ij} remains unchanged.

- If the job order of machine M_j already contains the jobs $J_{i_1},\dots, J_{i_k}$, then choose a value x, $x = 1,\dots, k$. Insert (i, j) as the direct successor of (i_x, j) into the job order of machine M_j. The rank value a_{ij} becomes the maximum of its old value and $a_{i_x, j} + 1$.

We can do this by topological sorting, which takes $O(|E|)$ time, where E is the arc set of the sequence graph. Because we consider only the arcs between an operation and its direct successor in the job order or in the machine order, we have $|E| = O(nm)$. Given the topological order of operations, A can also be updated in linear time $O(nm)$.

To avoid unnecessary insertion attempts, we look at the cycles created. Because there was no cycle in the graph before inserting the new operation (i, j), a new cycle either leaves (i, j) via an arc in E_{MO} and enters (i, j) via the new arc in E_{JO} or vice versa. In the first case, there is a path from the inserted operation (i, j) to its predecessor (i_x, j). Then it follows that inserting (i, j) at any later position in the job order of machine M_j will not result in a feasible sequence. In the other case, if there is a cycle leaving column j only at an operation (k, j), where J_k comes after J_i in the job order on machine M_j, then (i, j) has to be inserted after (k, j) in order to get a feasible sequence.

Additionally, because we want to generate only one sequence for each isomorphism class, we fix certain arcs in the job order of machine M_1. Namely, if $MO_i = MO_k$ holds for $i < k$, we demand that $(i, 1)$ comes before $(k, 1)$.

The whole generation of a sequence has an average case time complexity of $O(nm)$. Because we need to store partial sequences, the required space is $O(n^2m^2)$.

By enumerating sequences only for matrices MO which are lexicographically minimal in their structure isomorphism class and by fixing certain precedence relations between operations of jobs having the same machine order, we generate only a few sequences in each structure isomorphism class. However, an isomorphism test is still necessary to discard all but one sequence of each isomorphism class.

A *system of distinct representatives* (SDR) for the isomorphism classes of sequences is defined by a *choice function* which chooses for each isomorphism class one sequence. To get a choice function which is compatible with the decisions made above, we impose a total ordering "$\ll$" in the set of sequences based on the lexicographic order on the set matrices defined in section 2. We define that $A \ll B$ holds if and only if $MO(A)$ is lexicographically less than $MO(B)$ or $MO(A) = MO(B)$ and A is lexicographically less than B. The choice function simply chooses the minimal sequence of each class with respect to the ordering "$\ll$". Note that this is not necessarily the lexicographically least sequence of the isomorphism class since $MO(A) < MO(B)$ does not imply $A < B$ for two sequences A and B with respect to the lexicographic order "<". Nevertheless, this choice function is applied because it allows us to make extensive use of the information already obtained by the isomorphism tests of the machine order matrices MO stated in section 4.

In the algorithm for each generated sequence A, we have to test if A is the representative of its class. This can be done by testing for all sequences A' obtained from A by applying one of the isomorphisms, respectively, whether $A \ll A'$ holds.

For sequences A' obtained from A by an isomorphism of the form (ϱ, π) or $(\varrho, \pi, ^{-})$, this test can be simplified because $MO(A')$ is obtained from $MO(A)$ by applying the same isomorphism. In our enumeration algorithm, we generate sequences only for those matrices MO which are lexicographically minimal in their structural isomorphism class. Furthermore, $A' \ll A$ implies that $MO(A')$ is not lexicographically greater than $MO(A)$. Thus, $A' \ll A$ can only occur if $MO(A') = MO(A)$ holds, i.e. when (ϱ, π) or $(\varrho, \pi, ^{-})$, respectively, is an automorphism of $MO(A)$. The automorphisms of the machine order matrices are already known from the minimality test of MO (see section 4), so that we have to test only a small number of isomorphisms of the form (ϱ, π) or $(\varrho, \pi, ^{-})$.

Additionally, when generating the sequences, we fixed the orientation of special arcs between operations belonging to jobs with identical machine orders (see section 5). Namely, if J_i and J_k with $i < k$ have the same machine order, then operation $(i, 1)$ is processed before operation $(k, 1)$, which implies $a_{i1} < a_{k1}$. This way, we get $A \ll A'$ for all sequences A' obtained from a generated sequence A by an isomorphism (ϱ, π) which is a trivial automorphism of $MO(A)$, i.e., where π is the identity and only rows with the same machine order are swapped.

For example, in the case $n = 5$, $m = 3$, there is no MO having a nontrivial automorphism according to lemma 3. Thus, we do not have to test any automorphisms of the form (ϱ, π), and for machine order matrices without any automorphism of type $(\varrho, \pi, ^{-})$, we do not have to test minimality at all.

Only in the case $n = m$ does the minimality test become more complicated since we have isomorphisms that include the transposition of matrix A. For these isomorphisms, the job orders of the initial matrix A become machine orders of A' such that we have to carry out the lexicographic comparison of matrices $MO(A')$ and $MO(A)$ explicitly.

For a given $n \times m$ sequence A, the time complexity for the procedure of all its comparisons is $O(mn^2 \log n)$, since we have to sort all rows i of the sequence having the same machine order MO_i. The average time needed is much less because we can decide that a given sequence A is not the minimal representative of its isomorphism class whenever an isomorphic sequence A' with $A' \ll A$ is encountered. Furthermore, a single lexicographic comparison of two $n \times m$ sequences certainly does not need $O(nm)$ time on average.

6. Necessary conditions for irreducibility

The huge number of sequences even for comparatively small formats makes it impossible to apply an irreducibility test with high time complexity to all sequences. Thus, we need fast tests to exclude reducible sequences before applying the final test. In this section, we prove some necessary conditions for irreducibility of sequences which can be tested without computing the transitive closure of the sequence graph. Moreover, some of these conditions can be tested even for incomplete sequences so that we can avoid generating certain sequences at all.

Lemma 7. Let a sequence A contain two jobs J_i and J_k and a machine M_j such that (i, j) is the last operation of job J_i, (k, j) is the first operation of job J_k and (i, j) is the direct predecessor of (k, j) in the job order of machine M_j. Then reversing the arc $((i, j), (k, j))$ creates a sequence B with $A \succeq B$. If there is an l such that $l \neq j$ with $a_{kl} \leq a_{ij} + 3$ or $a_{kj} \leq a_{il} + 3$, then $A \succ B$ holds.

Proof. It is easy to see that the reversion of the arc $((i, j), (k, j))$ does not create new sets of operations lying on a common path, thus $A \succeq B$ holds. To prove $A \succ B$, we will show that under the condition $a_{kl} \leq a_{ij} + 3$ there does not exist a path in $SG(B)$ covering both operations (i, j) and (k, l). Assume that there is such a path. Because $SG(A)$ is acyclic, this path has to go from (i, j) to (k, l). The direct successor of (i, j) on this path has to be some operation (s, j) which satisfies $a_{sj} \geq a_{ij} + 2$ because the direct successor of (i, j) in $SG(A)$ is (k, j). Furthermore, there must be another operation on this path between (s, j) and (k, l), which is a contradiction to the condition $a_{kl} \leq a_{ij} + 3$. An analogous argument holds if we have $a_{kj} \leq a_{il} + 3$. □

Theorem 8. Let A be a sequence and let (i, j) be an operation in A such that (i, j) has at least one successor but no successor of (i, j) in row i or column j has a direct predecessor outside row i and column j, respectively. Then A is strongly reducible to some sequence B.

Proof. We construct a sequence B from A by deleting operation (i, j) and reinserting it as a sink in both the job order of machine M_j and the machine order of job J_i. Obviously, if a new path was created this way, it would have to contain operation (i, j) since the rest of the sequence remains unchanged. Assume there is an operation (k, l) which is on a common path with (i, j) in $SG(B)$ but not in $SG(A)$. Because (i, j) is a sink in $SG(B)$, this path has to start at (k, l) and eventually it enters row i or column j. Since there was no path from (k, l) to (i, j) in $SG(A)$, the operation where the new path in $SG(B)$ enters row i or column j must have been a successor of (i, j) in $SG(A)$. This is a contradiction to the assumption that no successor of (i, j) in row i or column j has a direct predecessor outside row i and column j. Thus, by theorem 2, $A \succeq B$ holds.

Without loss of generality, let

$$a_{\max} = \max\{a_{pj} : p = 1, \ldots, n\} \geq \max\{a_{iq} : q = 1, \ldots, m\}.$$

Choose k such that $a_{kj} = a_{\max}$. Thus, (k, j) is a successor of (i, j) and, by assumption, (k, j) has no predecessor outside column j. Furthermore, any operation (k, l) for $l \neq j$ is a successor of (k, j) and, consequently, it is on a common path with (i, j) in $SG(A)$. From $a_{kl} > a_{\max}$, it follows that there is no path in $SG(B)$ starting at operation (k, l) and leading into row i or column j. On the other hand, there are no paths in $SG(B)$ leaving (i, j). Thus, we have $A \succ B$. □

Theorem 9. Let A be an $n \times m$ sequence with the property that each job J_i, $i = 1,\ldots, n$, is first processed on the same machine M_j. Then A is strongly reducible to some sequence B.

For the proof, the reader is referred to Bräsel et al. [4].

In the enumeration algorithm, this theorem is applied in two stages. First, the condition can already be tested on the set of machine orders. Thus, for certain sets of job orders there cannot exist irreducible sequences and, therefore, we do not have to enumerate the corresponding sets of sequences. At the second stage, it is applied to the transposed sequences after the enumeration is done.

Theorem 10. An $n \times m$ sequence having an operation with rank $nm - 2$ is not irreducible for $n, m \geq 3$.

Proof. If an operation has rank $nm - 2$, then there exists a path W from a source to this operation covering all but two operations of the sequence. We have to consider several cases according to the relation of these two operations not covered by W to each other.

(1) These two operations are neither in the same row nor in the same column. Without loss of generality, let these operations be (1, 1) and (2, 2). If the source of W is neither in row 1 nor in column 2, then we exclude operation (1, 2) from the sequence and reinsert it as source in both the machine order of job J_1 and the job order of machine M_2. Since this operation was on a common path with each of the other operations, this does not create new sets of operations lying on a common path. Furthermore, operation (1, 2) is no longer on a common path with the source of W. If the source of W is in row 1 or column 2, we use the same argumentation to reinsert operation (2, 1) as a source.

(2) The two operations not covered by W are in the same row or column. Without loss of generality, let these operations be (1, 1) and (1, 2), and let (1,1) be a predecessor of (1, 2).

  (a) The source of W is in row 1. If there is an operation of W outside row 1 which is on a common path with both (1, 1) and (1, 2) and which is not in the same column as the source of W, then we can strongly reduce the sequence by shifting this operation to the source position in its job order and in its machine order. Thus, this operation becomes a source of the whole sequence and we are done.
  Otherwise, operation (1, 1) is the source in the job order of machine M_1 and each operation (1, k) which is a predecessor of (1, 1) is the source in the job order of its machine M_k. Furthermore, for each job J_i, $i > 1$, operation (i, 1) is a successor of operation (i, 2). This situation is illustrated in figure 1, where S denotes the source of the path W. We strongly reduce the sequence by

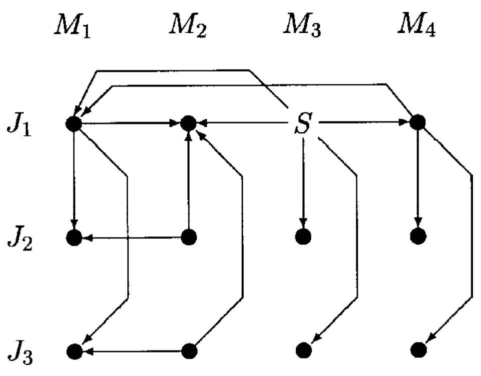

Figure 1. Part of the transitive closure of the sequence graph in case 2(a).

shifting operation (2, 2) to the source position. The only operation which was not on a common path with (2, 2) is operation (1, 1). Since every arc entering row 1 has a terminal vertex which is a successor of operation (1, 1), there is no path between operations (2, 2) and (1, 1) in the updated sequence. Furthermore, operation (2, 2) is not on a common path with the former source of W.

(b) The source of W is not in row 1. If there is a column $j \geq 3$ such that the source of W is not in column j, then operation $(1, j)$ is on a common path with each of the other operations. It is easy to see that the sequence can be reduced by shifting $(1, j)$ to the source position. A similar reduction can be done if the sink of W is not in column j; in this case, $(1, j)$ is reinserted as a sink.

Assume now that there is no column $j \geq 3$ not containing the source of W or not containing the sink of W. It follows that $m = 3$ holds and that the source and the sink of W are both in column 3. Analogously to case 2(a), we may assume that (1, 1) is the source in the job order of machine M_1 and (1, 2) is the sink in the job order of machine M_2 (note that for the existence of an arc $((1, 1), (i, 1))$ and an arc $((i, 2), (1, 2))$, it is sufficient that the source *or* the sink of W is not in row i). Because there is only one further operation in row 1, (1, 1) has to be a source of the sequence graph or (1, 2) is a sink. If (1, 1) is a source, we reduce the sequence by reinserting operation $(k, 2)$ as a source, where $k \neq 1$ is a row which does not contain the source of W (the row containing the sink of W is suitable). Otherwise, we shift an operation $(k, 1)$ to sink position, where k is a row not containing the sink of W. □

7. An irreducibility test

In order to decide about irreducibility of a given sequence A, it is necessary to find out which operations are on a common path in the corresponding sequence graph

$SG(A)$. More precisely, we have to compute the transitive closure $SG(A)^*$. An algorithm given by Goralcikova and Koubek [14] computes the transitive closure of a graph $G = (V, E)$ in $O(|V| \cdot |E|)$ time. In the case of a sequence graph, it is sufficient to consider for each vertex only the two leaving arcs going to the direct successor in the job order and in the machine order, respectively. Thus, we can compute the transitive closure $SG(A)^*$ of the sequence graph in $O(n^2m^2)$ time.

For the sake of simplicity, in the following we write that we create a "new path" if we create a new set of operations lying on a common path. Having the complete information about the path structure in a form which allows very fast access, it is worth coming back once more to the standard reductions described in the previous section. In particular, trying to reinsert an operation as source or sink works well. Recall that theorem 8 uses very strong assumptions to make sure that no new paths are created by reinserting an operation (i, j) as a sink. Namely, no successor of (i, j) in row i or column j may have a predecessor outside this row or column, respectively, because those operations would be candidates for creating a new path together with (i, j). Using $SG(A)^*$, we can weaken this assumption by ignoring all operations which are on a common path with (i, j) anyway.

Assume we want to test whether a given sequence A can be strongly reduced to some sequence B by deleting operation (i, j) and reinserting it as a sink. We have to ensure that in $SG(B)$:

(1) no new paths are created.
Let (k, j) and (i, l) be the last operations in the job order of machine M_j and in the machine order of job J_i, respectively. If a new path is created by reinserting operation (i, j) as a sink, then it has to go from an operation (p, q) with $p \neq i$ and $q \neq j$ to operation (i, j). Clearly, this new path has to go via (k, j) or (i, l). Consequently, A reduces to the new sequence B if and only if there is an arc $((i, j), (p, q))$ or $((p, q), (i, j))$ in $SG(A)^*$ for each operation (p, q) for which $SG(A)^*$ contains the arc $((p, q), (i, l))$ or $((p, q), (k, j))$.

(2) at least one path is destroyed.
Certainly, no arc $((p, q), (i, j))$ in $SG(A)^*$ can be destroyed by reinserting (i, j) as a sink. Thus, a destroyed path either started at (i, j) or went via (i, j).

(a) Consider an operation (p, q) with the property that $SG(A)^*$ contains an arc $((i, j), (p, q))$. If such an operation is on a common path with (i, j) in the modified sequence B, then analogously to condition 1 this has to be a path from (p, q) to (i, j) going via operation (i, l) or (k, j) (recall that these operations are the two direct predecessors of (i, j) in the modified sequence). Thus, operations (i, j) and (p, q) are not on a common path any longer if and only if $SG(A)^*$ contains neither arc $((p, q), (i, l))$ nor $((p, q), (k, j))$.

(b) If in A there is a predecessor of (i, j) in row i and a successor of (i, j) in column in j, or vice versa, such that each path from this predecessor to this successor contains (i, j), then reinserting (i, j) as a sink destroys all paths between these two operations. Since all maximal paths in A going via (i, j) have to contain a direct predecessor and a direct successor of (i, j), it is sufficient to test whether all paths between two such operations will be destroyed.

First we test whether there is a path in A from the direct predecessor (i, p) of (i, j) in row i to the direct successor (q, j) of (i, j) in column j not containing operation (i, j) itself. Let (i, r) be the direct successor of (i, j) in row i and let (s, p) be the direct successor of (i, p) in column p. There is a path from (i, p) to (q, j) which does not contain (i, j), if and only if (q, j) can be reached from (i, r) or (s, p), i.e., if $((i, r), (q, j))$ or $((s, p), (q, j))$ is an arc in $SG(A)^*$.

Analogously, we have to test whether there is a path from the direct predecessor of (i, j) in column j to the direct successor of (i, j) in row i which does not contain (i, j).

Thus, to test whether A can be strongly reduced by reinserting the operation (i, j) as a sink (or source, respectively), we have to check $O(nm)$ arcs in $SG(A)^*$. If this graph is stored as an adjacency matrix, this can be done in $O(nm)$ time. We obtain

Theorem 11. To verify whether a given $n \times m$ sequence can be strongly reduced by deleting an operation and reinserting it as a source or as a sink can be done in $O(n^2m^2)$ time and in $O(n^2m^2)$ space.

If a sequence cannot be strongly reduced by any of the tests described above, we have to make a decision about its irreducibility by a further test.

We implemented a modified version of the irreducibility test given by Kleinau [15]. The basic idea of this test is to reverse the orientation of the arcs in all suitable subsets of arcs in $SG(A)$ and to verify whether this leads to a sequence B with $A \succ B$. Contrary to the original version, we use theorem 2 to test in polynomial time whether $A \succ B$ holds and we combine arcs into classes to decrease the number of arc sets which have to be considered. Clearly, the algorithm can only work in practice if the number of arc sets suitable for reversing is not too large.

Consider the underlying undirected graph $\widehat{SG(A)}^*$ of the transitive closure of the sequence graph $SG(A)$. Assume there does not exist an edge $\{(i, l), (k, j)\}$ in $\widehat{SG(A)}^*$. Due to theorem 2, this edge does not exist in $\widehat{SG(B)}^*$ for any sequence B with $A \succeq B$. We can conclude that $SG(B)$ contains either both arcs $((i, j), (k, j))$ and $((i, j), (i, l))$ or both arcs $((k, j), (i, j))$ and $((i, l), (i, j))$. Otherwise, these two arcs would form a path and, consequently, $\widehat{SG(B)}^*$ would contain the edge $\{(i, l), (k, j)\}$, which is a contradiction to theorem 2. In this sense, the orientations of these two arcs imply each other.

We say that an arc $((i, j), (k, j))$ in $SG(A)$ *directly implies* the arc $((i, j), (i, l))$, written as $((i, j), (k, j)) \Gamma ((i, j), (i, l))$, if and only if $\widehat{SG(A)}^*$ does not contain the edge $\{(i, l), (k, j)\}$. The relation $((i, l), (k, l)) \Gamma ((k, j), (k, l))$ is defined analogously.

An arc e_a *implies* e_b, written as $e_a \Gamma^* e_b$, if and only if there is a chain $e_1, \ldots, e_k$ of arcs satisfying $e_a \Gamma e_1 \Gamma e_2 \ldots \Gamma e_k \Gamma e_b$. It is easy to see that "Γ^*" is an equivalence relation. The equivalence classes induced by this relation are called *implication classes*.

The motivation of the relation Γ^* is as follows. If we want to reduce a sequence A by reversing a certain set of arcs in $SG(A)$, then either we have to reverse the orientation of all arcs of an implication class or all arcs of this class have to remain unchanged. We obtain

Corollary 12. If all arcs in $SG(A)$ belong to the same implication class, then A is irreducible.

The implemented irreducibility test partitions the arcs of $SG(A)$ into implication classes. Then for each subset of the set of implication classes, it constructs a sequence B from A by reversing the orientations of all arcs belonging to these implication classes and finally it tests whether $A \succ B$ holds.

The partition of the arc set into implication classes can be organized in a very similar way like testing connectivity of a graph (cf. Golumbic [12]) and it takes $O(n^2m^2)$ time. Let k be the number of implication classes. In the worst case, namely if all operations are on a single path, each arc is in a different implication class and we have $k = O(n^2m + nm^2)$. Because we look at all 2^k subsets of the set of implication classes, the irreducibility test needs exponential time. Fortunately, the average number of implication classes of sequences which passed all the preliminary tests described above appears to be rather small.

8. Computational results

We implemented the described algorithms and rules in C++ and tested the program on a PC Pentium 133. The numbers of sequences, irreducible sequences and isomorphism classes, respectively, are given in table 3. As we conjectured, the ratio between the numbers of irreducible sequences and all possible sequences decreases with growing n and m. For small values of n and m, this ratio is given in table 4.

As far as the enumeration of the sequences themselves is concerned, the idea of fixing the machine orders first works very well. In order to find the unique representative sequence of a certain isomorphism class (see section 5), we only have to compute a small number of sequences for this isomorphism class. More precisely, it turns out that in our algorithm the average number of sequences actually generated for each structure isomorphism class varies between 1.23 for $n = 5$, $m = 3$ and 2.28 for $n = m = 4$. In the case $n = m$, it was to be expected that we generate about twice as many sequences because of the additional isomorphisms that include the transposition. The computation time for only generating the 4×4-sequences was about 2 minutes, the isomorphism tests took a further 10 minutes.

Using the necessary conditions for irreducibility described in section 6, it was possible to exclude about half of the sequences from further consideration. The exact

Table 3

Number of $n \times n$-sequences.

n	m	$IR(n, m)$	$SI(n, m)$	$GI(n, m)$	$PI(n, m)$	$T(n, m)$
2	2	1	3	3	4	14
2	3	1	12	17	17	204
3	3	7	147	280	533	19164
2	4	2	68	106	106	5016
3	4	123	13100	25924	25924	3733056
4	4	23825	3017369	6028059	12051574	6941592576
2	5	2	422	773	773	185520
3	5	2073	895388	1789432	1789432	1288391040
4	5	??	4609489912	9218730304	9218730304	26549943275520
2	6	3	3495	6671	6671	9595440
3	6	40933	82507654	164993112	164993112	712770186240
2	7	3	33193	65461	65461	659846880
3	7	??	9748141078	19496140704	19496140704	589563294888960

$IR(n, m)$ – number of structure isomorphism classes of irreducible $n \times m$-sequences.

Arbitrary sequences:

$SI(n, m)$ – number of classes according to structure isomorphism.
$GI(n, m)$ – number of classes according to graph isomorphism.
$PI(n, m)$ – number of classes according to permutation isomorphism.
$T(n ,m)$ – total number of sequences.

Table 4

Percentage of irreducible sequences among all possible sequences.

$n \backslash m$	2	3	4	5	6
2	14.30	5.88	1.44	0.26	0.037
3		2.69	0.88	0.23	0.049
4			0.39	?	?

number of sequences left after each single test is given in table 5. Even more sequences were excluded by the source–sink test using the transitive closure of the sequence graph described in section 7. Concerning the application of irreducibility in algorithms, this test turned out to be more useful than the irreducibility test itself because it excludes a huge number of sequences in polynomial time. The computation time for these tests is less than the time saved due to the fact that many sequences are no longer generated.

Table 5

Number of sequences left after each test.

Test	3 × 4	3 × 5	4 × 4	3 × 6
Classes according to structure isomorphism, no irreducibility test	13100	895388	3017369	82507654
Source–sink test according to theorems 8 and 9 applied only on *MO*	6081	512561	2012129	69000797
Sequences with maximal rank ≥ $nm - 2$ excluded (theorem 10)	5640	494625	1872016	51921639
Test according to theorem 9 applied on *JO*	5050	454266	1852021	48854697
Test according to lemma 7 (reverse one arc)	4291	334014	1610491	31464174
Source–sink test using the transitive closure (theorem 11)	256	8866	41512	462136
Final irreducibility test	123	2073	23825	40933

The complexity of the eventual irreducibility test is exponential in the number of implication classes. Fortunately, the number of implication classes of a sequence which passed all preliminary tests is usually rather small. For instance, in the case $n = 3$, $m = 4$, the average number is 2.4 and the maximum 9 instead of maximal 30 classes for arbitrary sequences. For $n = 3$, $m = 5$, the number of implication classes of arbitrary sequences ranges from 1 to 45, whereas the tested sequences have at most 20 classes with an average of 4.6. In the case $n = m = 4$, the complete program needs 184 minutes.

For some values of n and m, in table 6 we list the numbers of structure isomorphism classes of irreducible sequences having certain maximal rank and number of implication classes, respectively. Finally, let us mention some special irreducible sequences for the case $n = m = 4$:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 5 & 10 \\ 5 & 8 & 4 & 9 \\ 6 & 7 & 8 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 3 & 8 & 9 \\ 5 & 6 & 4 & 10 \\ 6 & 1 & 7 & 2 \\ 7 & 2 & 3 & 4 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 6 & 7 & 8 \\ 4 & 5 & 6 & 7 \\ 1 & 2 & 9 & 3 \\ 3 & 4 & 8 & 1 \end{pmatrix}.$$

The sequences which are structure isomorphic to A and B are the only irreducible 4 × 4 sequences having maximal rank 10. The sequence C consists of 9 implication classes which is the maximal number of classes among all irreducible sequences of this format. The following sequences D and E are both irreducible, D is not structure isomorphic to E (denoted by $D \not\simeq_S$), but D and E are similar ($D \simeq E$):

Table 6

Number of structure isomorphism classes of irreducible $n \times m$-sequences having certain maximal ranks and numbers of implication classes.

Format	Maximum rank of an operation								
	4	5	6	7	8	9	10	11	12
3 × 4	4	40	75	4	0	0	0	0	0
3 × 5	0	38	541	1153	334	7	0	0	0
3 × 5	0	0	658	8428	19744	10844	1248	11	0
4 × 4	4	88	1847	5845	3932	355	2	0	0

Format	Number of implication classes								
	1	2	3	4	5	6	7	8	9
3 × 4	93	24	5	1	0	0	0	0	0
3 × 5	987	700	190	101	60	23	9	3	0
3 × 6	10050	11972	6761	4501	2762	2082	1161	708	462
4 × 4	7905	2564	1148	290	118	39	6	2	1

$$D = \begin{pmatrix} 1 & 6 & 7 & 8 \\ 2 & 5 & 6 & 7 \\ 3 & 4 & 5 & 1 \\ 4 & 7 & 1 & 2 \end{pmatrix}, \quad E = \begin{pmatrix} 2 & 6 & 7 & 8 \\ 1 & 5 & 6 & 7 \\ 3 & 4 & 5 & 1 \\ 4 & 7 & 1 & 2 \end{pmatrix}.$$

Bräsel and Kleinau [7] implicitly raised the question whether an irreducible sequence can be reduced only to itself and to its reversed sequence. By the example of sequences D and E, the answer is no. It turns out that there is a proper subset $\mathcal{M}$ of the set of irreducible sequences such that for each instance of processing times, the set $\mathcal{M}$ contains an optimal sequence. In the case $n = m = 4$, we can exclude one of the sequences D and E from $\mathcal{M}$.

9. Concluding remarks

Using the described algorithms, we were able to compute the number of all sequences up to the formats 4 × 5 and 3 × 7. It turns out that the number of irreducible sequences is indeed a very small fraction of the number of all sequences, which becomes even smaller as n and m increase. Besides these enumerational results, which are mainly of combinatorial interest, the possibility to actually enumerate the irreducible sequences already leads to some theoretical results about irreducibility because it gives a good basis to derive promising conjectures. For example, the results in section 6 were found during implementation of the program.

Some of the open questions left in this area are:

- Find good upper and lower bounds for the number of $n \times m$ sequences.
- Is it possible to decide in polynomial time whether a given sequence A is irreducible?
- What is the maximal rank and the maximal number of implication classes of an irreducible sequence?

Acknowledgements

We would like to thank the anonymous referees for the detailed hints to improve the quality of the paper.

References

[1] S.B. Akers and J. Friedman, A non-numerical approach to production scheduling problems, Operations Research 3(1955)429–442.

[2] S. Ashour, *Sequencing Theory*, Springer, 1972.

[3] H. Bräsel, Latin rectangle and scheduling theory (in German), Habilitationsschrift, Technical University Magdeburg, Germany, 1990.

[4] H. Bräsel, M. Harborth, T. Tautenhahn and P. Willenius, On the hardness of the classical job shop problem, Working Paper, University of Magdeburg, Germany, 1997.

[5] H. Bräsel, M. Harborth and P. Willenius, Isomorphism for digraphs and sequences of shop scheduling problems, Preprint 21/1996, University of Magdeburg, Germany, 1996.

[6] H. Bräsel and M. Kleinau, On the number of feasible schedules of the open shop problem – an application of special Latin rectangles, Optimization 23(1992)251–260.

[7] H. Bräsel and M. Kleinau, New steps in the amazing world of sequences and schedules, Mathematical Methods of Operations Research 43(1996)195–214.

[8] W. Burnside, *Theory of Groups of Finite Order*, Cambridge University Press, 1897.

[9] R.W. Conway, W.L. Maxwell and L.W. Miller, *Theory of Scheduling*, Addison Wesley, 1967.

[10] D. de Werra, Almost nonpreemptive schedules, Annals of Operations Research 26(1990)243–256.

[11] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[12] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[13] T. Gonzalez and S. Sahni, Open-shop scheduling to minimize finish time, Journal of the Association for Computing Machinery 23(1976)665–680.

[14] A. Goralcikova and G. Koubek, A reduct and closure algorithm for graphs, Mathematical Foundations of Computer Science 74 (1979) 301–307.

[15] M. Kleinau, On the structure of shop scheduling problems: Number problems, reducibility and complexity (in German), Dissertation, Technical University Magdeburg, Germany, 1993.

[16] T. Tautenhahn, Irreducible sequences – an approach to interval edge colouring trees, Preprint OR83, Faculty of Mathematical Studies, University of Southampton, UK, 1996.