Matrices in Shop Scheduling Problems

Heidemarie Bräsel

Faculty of Mathematics
Otto-von-Guericke University Magdeburg

Es ist für mich eine ehrenvolle Aufgabe, einen Beitrag für dieses Buch einzubringen. Gleichzeitig ist es ein herzliches Dankeschön für Herrn Prof. Klaus Neumann für seine wissenschaftlichen Arbeiten, deren Ergebnisse ich sehr gern nutze, und für seine Unterstützung und sein stetes Interesse an der Entwicklung unserer Forschungsgruppe. Ich verbinde dies mit allen guten Wünschen für einen gesunden Ruhestand der Familie Neumann, der - dessen bin ich mir sicher - öfter auch in einen Unruhestand ausarten wird.

1 Introduction

In this paper shop scheduling problems are modeled by matrices. Initially we assume that each job is processed at most once on each machine. It is shown how the model can be extended to shop problems with more than one operation on each machine and to the case that preemption is allowed.

Modelling shop problems by matrices is a very natural approach of modelling such scheduling problems. At first it was presented by BRÄSEL [1]. The model is easy comprehensible and can be applied to simplify the description of algorithms in this field, for instance the block-approach idea for job shop problems and algorithms in the case of unit processing times.

Moreover, this model gives rise to new theoretical results. We give a brief review on such papers. The complexity question of some open shop problems with unit processing times was solved, see for instance BRÄSEL ET AL. [7], [8] and [9], TAUTENHAHN [16] and [17]. The insertion technique (cf. [1]) was developed for enumeration algorithms and beam search strategies, see for instance BRÄSEL ET AL. [10], WERNER AND WINKLER [18] and SOTSKOW ET AL. [15]. Theoretical results were obtained for counting problems, see BRÄSEL AND KLEINAU [5], HARBORTH [14] and BRÄSEL ET AL. [2] and [3]. Moreover, the model was applied for structural investigations of sequences and schedules: Shop scheduling spaces were characterizised algebraically by DHAMALA [12]. The irreducibility theory was developed, introduced by BRÄSEL AND KLEINAU [6]. Here especially the papers of BRÄSEL ET AL. [2], [3] and WILLENIUS [19] has to be mentioned. Furthermore, the software package LiSA works with this model succesfully.

However, there is no article in English to explain the basic model in detail. This paper closes this gap. It is organized like an introductory lecture on shop problems. We start with basic notations, give an overview on the used graphs and their description by matrices and present simple algorithms concerning the defined matrices. The insertion technique for construction of sequences is introduced and some properties of sequences are charakterized. We next show how the model can be modified for other classes of shop problems. Finally, the software package LiSA - A Library of Scheduling Algorithms is presented which contains the introduced matrices and their visualization as graphs and Gantt charts.

2 Basic Notations

In a *shop scheduling problem* a set of n *jobs* A_i, $i \in I = \{1, \ldots, n\}$, has to be processed on a set of m *machines* M_j, $j \in J = \{1, \ldots, m\}$, in a certain machine environment α under certain additional constraints β such that an objective function γ is optimized. Such a problem is called deterministic if all parameters are fixed and given in advance. Various optimization problems concerning allocation of restricted resources can be modeled as scheduling problems. We use the standard $\alpha \mid \beta \mid \gamma$ classification of deterministic scheduling problems developed by GRAHAM ET AL. [13].

At first we consider so-called *classical* shop problems, i.e., each job is processed on each machine at most once.

Processing of job A_i on machine M_j is called an *operation* (ij). PT denotes the matrix of processing times: $PT = [p_{ij}]$. The set of all operations SIJ is given by $SIJ = \{(ij) \mid p_{ij} > 0\}$. We assume that each job is processed on at most one machine at a time and each machine processes at most one job at a time. For certain shop problems, a *release time* $r_i \geq 0$, a *due date* $d_i \geq 0$ or a weight $w_i > 0$ for job A_i, $i \in I$, are requested. Let u_i and v_j be the number of operations for job A_i and on machine M_j, respectively. Then we define:

The *machine order of the job* A_i is the order of machines on which this job has to be processed: $M_{j_1} \rightarrow M_{j_2} \rightarrow \ldots \rightarrow M_{j_{u_i}}$.

The *job order on machine* M_j is the order of the jobs which this machine processes: $A_{i_1} \rightarrow A_{i_2} \rightarrow \ldots \rightarrow A_{i_{v_j}}$.

In a *job shop problem* ($\alpha = J$) the machine order of each job is given in advance. In a *flow shop problem* ($\alpha = F$) the machine orders of each job are the same , w.l.o.g. in the case of $SIJ = I \times J$: $M_1 \rightarrow M_2 \rightarrow \ldots \rightarrow M_n$. In an *open shop problem* ($\alpha = O$) both machine orders and job orders can be chosen arbitrarily. Other precedence constraints on the operations can be easily integrated into the model.

In a shop problem a combination of machine orders and job orders is to determine such that a time table of processing (schedule) can be constructed, which satisfies the additional constraints and minimizes the given objective function.

Let C_i be the completion time of job A_i. An objective function $\gamma = F(C_1, \ldots, C_n)$ is called *regular* if it has the following property:
If for two schedules S and S^* the inequality $C_i^* \geq C_i$ holds for all $i \in I$ then $F(C_1^*, \ldots, C_n^*) \geq F(C_1, \ldots, C_n)$ is satisfied.

The makespan C_{max}, the weighted sum of completion times $\sum w_i C_i$, the maximum lateness L_{max}, the weighted tardiness $\sum w_i T_i$ and the weighted number of late jobs $\sum w_i U_i$ are regular, where:
$C_{max} = \max_{i \in I}\{C_i\}$, $L_{max} = \max_{i \in I}\{d_i - C_i\}$, $\sum w_i T_i = \sum_{i \in I} w_i \max\{0, d_i - C_i\}$

and $U_i = \begin{cases} 1, \text{ if } C_i > d_i \\ 0, \text{ otherwise} \end{cases}$ for all $i \in I$. Often $w_i = 1$ for all $i \in I$ holds.

3 Graphs and Matrices for Shop Problems

This chapter starts with a model of shop problems where preemption of the operations is not allowed.

3.1 Partial Orders and Schedules

We define the following digraphs where in each case the set of vertices is the set SIJ of operations:

- The digraph of machine orders $G(MO) = (SIJ, A_{mo})$ contains all arcs which describe the direct precedence constraints in all machine orders.
  $((ij),(kl)) \in A_{MO} \Longleftrightarrow \begin{cases} i = k \ \wedge \text{ after the processing of job } A_i \text{ on} \\ M_j \text{ job } A_i \text{ is processed on machine } M_l \end{cases}$
- The digraph of job orders $G(JO) = (SIJ, A_{JO})$ contains all arcs which describe the direct precedence constraints in all job orders.
  $((ij),(kl)) \in A_{JO} \Longleftrightarrow \begin{cases} j = l \ \wedge \text{ after the processing of job } A_i \text{ on} \\ \text{machine } M_j \text{ machine } M_j \text{ processes } A_k. \end{cases}$
- The digraph $G(MO, JO) = (SIJ, A)$ contains all arcs of $A = A_{MO} \cup A_{JO}$.

A combination (MO, JO) of machine orders and job orders is called *feasible*, if the corresponding digraph $G(MO, JO)$ does not contain a cycle. In this

case $G(MO, JO)$ is called a *sequence graph.* The described acyclic graphs are partial orders on the set of all operations.

Example 1 *Assume that three jobs have to be processed on four machines. The matrix PT of processing times is given by*

$$PT = \begin{bmatrix} 2 & 1 & 0 & 1 \\ 2 & 3 & 4 & 3 \\ 1 & 5 & 1 & 2 \end{bmatrix}, \text{ therefore } SIJ = I \times J \backslash \{(13)\} \text{ holds.}$$

We consider the following machine orders and job orders:

$$\begin{array}{ll} A_1: & M_1 \to M_2 \to M_4 \\ A_2: & M_2 \to M_4 \to M_1 \to M_3 \\ A_3: & M_4 \to M_1 \to M_2 \to M_3 \end{array} \qquad \begin{array}{ll} M_1: & A_1 \to A_2 \to A_3 \\ M_2: & A_2 \to A_3 \to A_1 \\ M_3: & A_3 \to A_2 \\ M_4: & A_3 \to A_1 \to A_2 \end{array}$$

The corresponding digraph $G(MO, JO)$, see Figure 1, contains vertical *arcs, which represent job orders on the machines and* horizontal *arcs representing machine orders of the jobs.*

The combination of machine orders and job orders is not feasible because $G(MO, JO)$ contains the cycle $(12) \to (14) \to (24) \to (21) \to (31) \to (32) \to (12)$. *Since we have a cycle, there can not exist any schedule of processing.*

If we choose the natural order of the machines in each machine order and of the jobs in each job order, the digraph $G(MO, JO)$ cannot contain any cycle, because all arcs are directed to the left or downwards. In this case a corresponding schedule can easily be constructed.

Now we assign the weight p_{ij} to each vertex (ij) of the sequence graph $G(MO, JO)$. Then a schedule can be constructed. Usually schedules are described by the start times or the completion times of all operations. There exist the following classes of schedules:
A schedule is called a *non-delay* schedule, if no machine is kept idle when there exists a job available for processing.
A schedule is called *active*, if no operation can be completed earlier by changing the job orders without delaying any other operation.
A schedule is called *semiactive*, if no operation can be completed earlier without changing the job order on any of the machines.

Note, that each non-delay schedule is also active and each active schedule is also semiactive, but not vice versa.

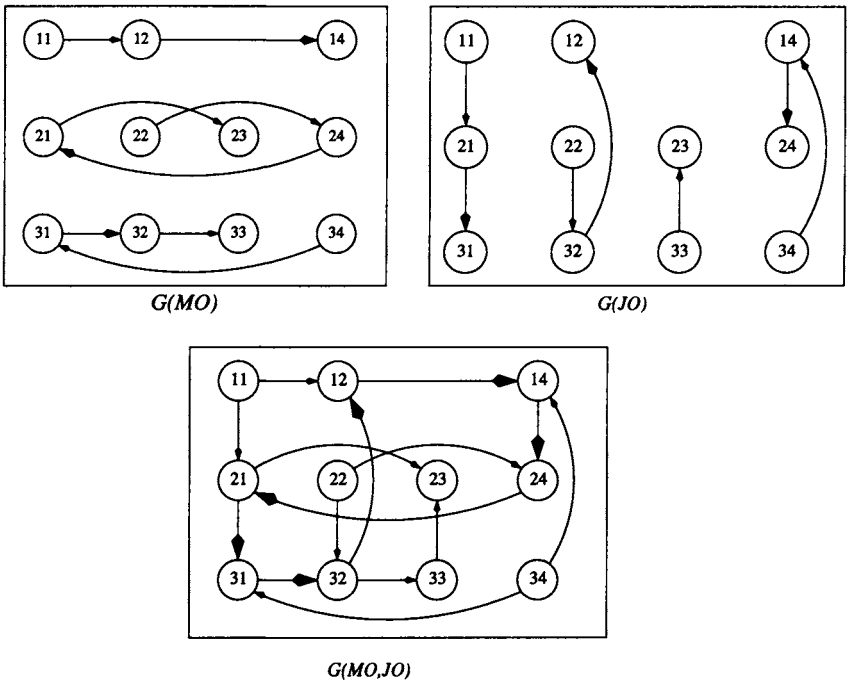

Figure 1: $G(MO)$, $G(JO)$ und $G(MO, JO)$ for Example 1

In the case of regular objective functions there always exists an optimal semiactive schedule and the computing of a longest path in $G(MO, JO)$ yields the makespan. We use the notation *longest path* with respect to the sum of the weights of the vertices contained in the path. Schedules are visualized by *Gantt charts*, which can be *machine oriented* or *job oriented.* In Figure 2 a job-oriented Gantt chart of a schedule with minimal makespan is given (see Example 1). There cannot be any better schedule because the longest job A_2 has no idle time within its processing.

In general, the set of schedules is infinite, but the set of sequences is finite. The binary relation R in the set of schedules:
"schedule 1 R schedule 2 if and only if both schedules have the same machine orders and job orders" is an equivalence relation. We can choose all semiactive schedules with unit processing times as representativs of the equivalence classes, whose number is finite.

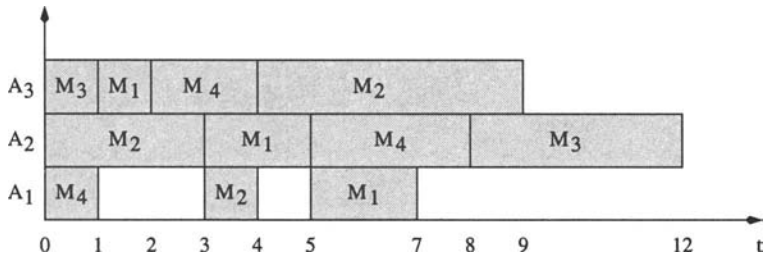

Figure 2: Job-oriented Gantt chart

3.2 Matrices in Shop Problems

In the literature the most commonly used model for shop problems is the well-known disjunctive graph model, see for instance BRUCKER [11]. We obtain the model used here by the following modifications:
- Cut the inserted source and sink and the corresponding incident arcs.
- Determine an acyclic orientation of the disjunctive graph.

We obtain the sequence graph (cf. Section 3.1) by deleting all transitive arcs which are not direct precedence constraints in the machine orders and in the job orders.

Now we define a set of matrices, where in each matrix an information of the operation (ij) on position (ij) is contained, this is the real advantage of the model. The digraphs $G(MO), G(JO)$ and $G(MO, JO)$ and in particular, the structure of the contained paths are visible by the matrices without drawing the digraphs. The number of vertices on a longest path with respect to unit weights of all vertices from a source to the vertex v is called *rank* of v: $rk(v)$. Now we define the following matrices:

- the *machine order matrix* $MO = [mo_{ij}]$: mo_{ij} is the rank of the operation $(ij) \in SIJ$ in the digraph $G(MO)$.
- the *job order matrix* $JO = [jo_{ij}]$: jo_{ij} is the rank of the operation $(ij) \in SIJ$ in the digraph $G(MO)$.
- the *sequence (matrix)* $PO = [po_{ij}]$: po_{ij} is the rank of the operation $(ij) \in SIJ$ in the sequence graph $G(MO, JO)$.

These matrices describe structural properties of a solution of a shop problem.

We extend this set by matrices with properties of the weighted sequence graph, i.e. the corresponding schedule (see Figure 3):

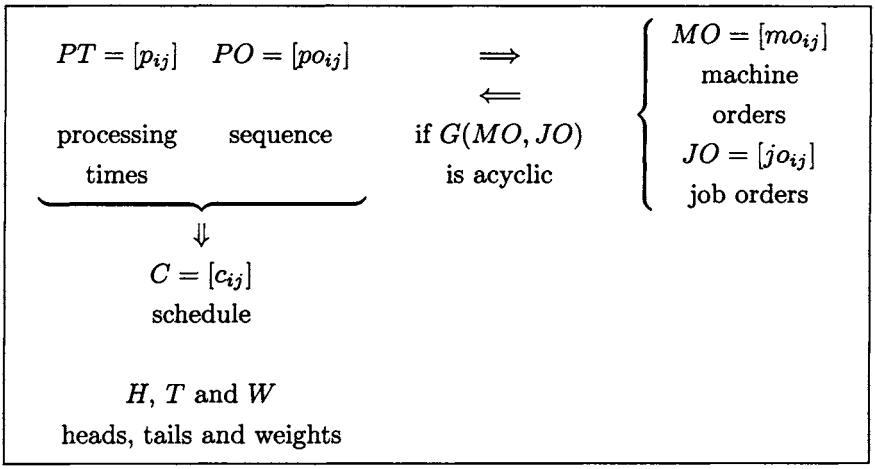

Figure 3: Matrices for shop problems

- the *completion time matrix* $C = [c_{ij}]$: c_{ij} is the completion time of the operation $(ij) \in SIJ$;

- the *head matrix* $H = [h_{ij}]$: the head h_{ij} is the lenght of the shortest time period which all predecessors of operation (ij) need for their processing, i.e. the weight of a longest path from a source to a direct predecessor of (ij) in the sequence graph;

- the *tail matrix* $T = [t_{ij}]$: the tail t_{ij} is the lenght of the shortest time period which all successors of (ij) need for their processing, i.e. the weight of a longest path from a direct successor of (ij) to a sink in the sequence graph;

- the matrix $W = H + PT + T$: the weight w_{ij} is the weight on a longest path across the operation (ij).

Clearly, the weight of a longest path can be obtained as maximal value in the completion time matrix C. By definition of W all operations with maximal value in W belong to at least one longest path.

Note, that in the case of a job shop problem with given matrix MO all sequences PO that contain the machine orders of MO are feasible.

3.3 Sequences, Schedules and Latin Rectangles

A *Latin rectangle* $LR[n,m,r]$ is a matrix with n rows, m columns and entries from the set $\{1,\ldots,r\}$, where each element from this set occurs at most once in each row and each column, respectively. If $n = m = r$ the matrix is denoted as latin square $LQ(n)$ of order n.

By definition of the rank of an operation, any sequence $PO = [p_{ij}]$ with $1 \leq po_{ij} \leq r$ has the following properties:
(a) Each entry k, $k \in \{1,\ldots,r\}$, is contained at most once in each row and in each column, respectively.
(b) For each $po_{ij} = k > 1$ the entry $k-1$ exists in row i or in column j.

Therefore, a sequence PO is a Latin rectangles, which satisfies in addition the so-called sequence condition (b). Now a feasible combination of MO and JO for given n and m can be easily constructed, if both matrices can be chosen arbitrarily or with repect to a given MO.

Example 2 *Consider a job shop problem with $n = 3$, $m = 4$ and the machine order matrix MO. If MO is a Latin rectangle, then a sequence is given by $PO = MO$. In the other case we determine a column where an (minimal) entry occurs more than once and construct a linear order on this operations, i.e., the corresponding arcs are inserted into $G(MO)$ and the rank matrix will be updated. The operations with fixed rank will be marked. This procedure is repeated until a Latin rectangle is obtained.*

$$MO = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \\ 4 & 1 & 3 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1* & 2* & \mathbf{3} & 4 \\ 2* & 4 & \mathbf{3} & 5 \\ 4 & 1* & \mathbf{3} & 2* \end{bmatrix} \quad PO = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 4 & 6 \\ 6 & 1 & 5 & 2 \end{bmatrix}$$

Note, that in the case $p_{ij} = 1$ for all operations the equality $PO = C$ holds for all shop problems without additional restrictions . Here the computing of an optimal schedule is equivalent to the construction of a certain Latin rectangle. Let us consider the problem $O \mid p_{ij} = 1 \mid C_{max}$. We have to compute a Latin rectangle $PO = C = LR[n,m,r]$ where r is minimal. Therefore, all schedules $C = LR[n,m,\max\{n,m\}]$ are optimal.

Such problems are related to edge coloring problems on graphs, which should be shortly described:

Each operation set SIJ of a shop problem can be assigned to a bipartite graph $G^b(SIJ) = (I \cup J, E)$ with $e = \{ij\} \in E$ iff $(ij) \in SIJ$. Therefore some open shop problems are equivalent to certain edge coloring problems on $G^b(SIJ)$:
(a) $O \mid p_{ij} = 1 \mid C_{max}$: Determine the chromatic index of $G^b(SIJ)$.

(b) $O \mid pmtn \mid C_{max}$: Solve a weighted edge coloring problem on $G^b(SIJ)$, where the edges are weighted by the processing times.
(c) $O \mid\mid C_{max}$: Solve an interval edge coloring problem on $G^b(SIJ)$, where the edges are weighted by the processing times.

3.4 Basic Algorithms in the Model

This section contains some basic algorithms for the modeling of shop problems by matrices. Note that the operations can be ordered in time $O(|SIJ|)$ by nondecreasing ranks: In a first loop we count the operations with rank k. Let n_k be the number of operations with rank k, $k = 1, \ldots, r$, where r is the maximal rank contained in PO. Next let $pointer(k) = \sum_{l=1}^{k-1} n_l + 1$ be the pointer to the first operation with rank k. Finally we sort in a second loop step by step the operations according to these pointers and update the applied pointer.

Algorithm 1 compute the sequence PO from given matrices MO and JO, if the combination (MO, JO) is feasible. The set MQ contains all operations, which are sources in both $G(MO)$ and $G(JO)$.

Algorithm 1: Computation of PO from MO and JO, if $G(MO, JO)$ is acyclic

Input: SIJ, MO and JO on the operation set SIJ;
Output: PO on the operation set SIJ, if $G(MO, JO)$ is acyclic;
BEGIN $k := 0$;
  REPEAT
    $k := k + 1$;
    Calculate $MQ = \{(ij) \in SIJ \mid mo_{ij} = jo_{ij} = 1\}$;
    IF $MQ = \emptyset$ **THEN** (MO, JO) is infeasible and **STOP**;
    FORALL $(ij) \in MQ$ **DO**
      BEGIN
        $po_{ij} := k$; Label row i in MO; Label column j in JO;
      END;
    $SIJ := SIJ \setminus MQ$;
    FORALL $(ij) \in SIJ$ in a labeled row in MO **DO**
      $mo_{ij} := mo_{ij} - 1$;
    FORALL $(ij) \in SIJ$ in a labeled column in JO **DO**
      $jo_{ij} := jo_{ij} - 1$;
    delete all labels;
  UNTIL $SIJ = \emptyset$;
END.

Algorithm 2 computes MO and JO from PO. Here a_i and b_j are the smallest integers, which are available for the rank of an operation of job A_i and of an operation on machine M_j, respectively. The maximal entry in sequence PO is denoted by r in the following algorithms.

Algorithm 2: Computing of MO and JO from PO

Input: r, I, J, SIJ, PO on the operation set SIJ;
Output: MO und JO on the operation set SIJ;
BEGIN
  FORALL $i \in I$ **DO** $a_i := 1$; **FORALL** $j \in J$ **DO** $b_j := 1$;
  FOR $k := 1$ **TO** r **DO**
    FORALL $(ij) \in SIJ$ with $po_{ij} = k$ **DO**
      BEGIN
        $mo_{ij} := a_i$ and $a_i := a_i + 1$;
        $jo_{ij} := b_j$ and $b_j := b_j + 1$;
      END;
END.

Algorithm 3 generates the corresponding semiactive schedule, i.e. the matrix C of completion times of all operations, from a corresponding pair PT and PO. Here r_i and $\overline{r}_j$ denote the currently smallest possible start times of operation (ij), i.e. for job A_i, and on machine M_j, respectively.

Algorithm 3: Computing of C from PT and PO

Input: r, I, J, SIJ, PT and PO on the operation set SIJ;
Output: C on the operation set SIJ.
BEGIN
  FORALL $i \in I$ **DO** $r_i := 0$; **FORALL** $j \in J$ **DO** $\overline{r}_j := 0$;
  FOR $k := 1$ **TO** r **DO**
    FORALL $(ij) \in SIJ$ with $po_{ij} = k$ **DO**
      BEGIN
        $c_{ij} := \max\{r_i, \overline{r}_j\} + p_{ij}$;
        $r_i := c_{ij}$; $\overline{r}_j := c_{ij}$;
      END;
END.

Algorithm 4 determines the matrices $H = [h_{ij}]$ and $T = [t_{ij}]$ from PT and PO.

Algorithm 4: Computing of H and T

Input: r, I, J, SIJ, PT and PO on the operation set SIJ;
Output: H and T on the operation set SIJ.
BEGIN
FORALL $i \in I$ **DO BEGIN** $r_i := 0$; $s_i := 0$; **END**;
FORALL $j \in J$ **DO BEGIN** $\overline{r}_j := 0$; $\overline{s}_j := 0$; **END**;
FOR $k := 1$ **TO** r **DO**
BEGIN
FORALL $(ij) \in SIJ$ with $po_{ij} = k$ **DO**
BEGIN
$h_{ij} := \max\{r_i, \overline{r}_j\}$; $r_i := h_{ij} + p_{ij}$; $\overline{r}_j := h_{ij} + p_{ij}$;
END;
FORAll $(ij) \in SIJ$ with $po_{ij} = r - k + 1$ **DO**
BEGIN
$t_{ij} := \max\{s_i, \overline{s}_j\}$; $s_i := t_{ij} + p_{ij}$; $\overline{s}_j := t_{ij} + p_{ij}$;
END;
END;
END.

We obtain the so-called reverse schedule S^{-1} to schedule S by reversing all arcs in the corresponding sequence graph. Clearly, $C_{max}(S) = C_{max}(S^{-1})$ is satisfied and the head of an operation (ij) in S is the tail of this operation in S^{-1} and vice versa. In Algorithm 4 $r_i, \overline{r}_j$ are again the earliest start times for job A_i and on machine M_j, respectively. $s_i, \overline{s}_j$ denote the earliest start times of job J_i and on machine M_j in the backwards calculation.

We close this section with an example illustrating the model.

Example 3 *Consider the matrix PT of processing times of Example 1, where due dates $d_1 = 6, d_2 = 12, d_3 = 8$ are given. The following combination of machine orders and job orders is feasible because the corresponding graph $G(MO, JO)$ is acyclic.*

Algorithm 1 computes the sequence PO and Algorithm 3 computes the schedule C. The matrices H and T are computed by Algorithm 4, therefore $W = H + PT + T$ can be computed. The longest path is unique: $(22) \rightarrow (21) \rightarrow (24) \rightarrow (23)$.
Schedule C yields $C_{max} = 12$ and $C_1 = 7$, $C_2 = 12$, $C_3 = 9$, therefore $\sum C_i = 28$, $L_{max} = 1$, $\sum T_i = 2$ and $\sum U_i = 1$ follow. Note, that this schedule is optimal in the open shop case with respect to the objective functions C_{max} and L_{max}. However, there exist better schedules with respect to $\sum C_i$, $\sum T_i$ and $\sum U_i$.

$A_1: \quad M_4 \to M_2 \to M_1$
$A_2: \quad M_2 \to M_1 \to M_4 \to M_3$
$A_3: \quad M_3 \to M_1 \to M_4 \to M_2$

$M_1: \quad A_3 \to A_2 \to A_1$
$M_2: \quad A_2 \to A_1 \to A_3$
$M_3: \quad A_3 \to A_2$
$M_4: \quad A_1 \to A_3 \to A_2$

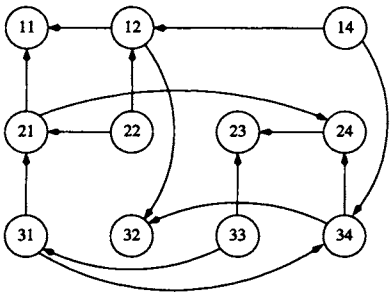

Figure 4: Machine orders, job orders and the sequence graph $G(MO, JO)$

We get the following information on operation (31)*: the processing time is 1, it is the second operation for job A_1 and it is the first operation on machine M_1, the completion time is 2, the head and the tail of* (31) *are 1 and 9, respectively. The longest path on this operation has weight 11.*

$$PT = \begin{bmatrix} 2 & 1 & 0 & 1 \\ 2 & 3 & 4 & 3 \\ 1 & 5 & 1 & 2 \end{bmatrix} \qquad PO = \begin{bmatrix} 4 & 2 & - & 1 \\ 3 & 1 & 5 & 4 \\ 2 & 4 & 1 & 3 \end{bmatrix}$$

$$\Leftrightarrow \quad MO = \begin{bmatrix} 3 & 2 & - & 1 \\ 2 & 1 & 4 & 3 \\ 2 & 4 & 1 & 3 \end{bmatrix} \qquad JO = \begin{bmatrix} 3 & 2 & - & 1 \\ 2 & 1 & 2 & 3 \\ 1 & 3 & 1 & 2 \end{bmatrix}$$

$$C = \begin{bmatrix} 7 & 4 & - & 1 \\ 5 & 3 & 12 & 8 \\ 2 & 9 & 1 & 4 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 3 & - & 0 \\ 3 & 0 & 8 & 5 \\ 1 & 4 & 0 & 2 \end{bmatrix} + \begin{bmatrix} 2 & 1 & - & 1 \\ 2 & 3 & 4 & 3 \\ 1 & 5 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 5 & - & 9 \\ 7 & 9 & 0 & 4 \\ 9 & 0 & 10 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 9 & - & 10 \\ 12 & 12 & 12 & 12 \\ 11 & 9 & 11 & 11 \end{bmatrix}$$

4 On Sequences in Shop Problems

In this section we discuss properties of sequences. We start with construction of sequences by the insertion idea, which is used in exact and heuristic algorithms for shop problems. It is shown in an example that we can exclude sequences because there is another sequence which has the same or a better objective function value for any choice of processing times .
The section closes with a distribution of open shop sequences to job shop classes.

4.1 Generating Sequences and Reducibility

Consider a sequence on the operation set $SIJ \subset I \times J$ and an operation $(ij) \in I \times J \setminus SIJ$. Now this operation is inserted into the given sequence graph with the following properties:
- a sequence on $SIJ \cup \{(ij)\}$ is generated and
- all precedence constraints of the old sequence graph are again contained in the new one.

In general, there are $u_i + 1$ choices to insert (ij) in the machine order of job A_i and $v_j + 1$ ways to insert (ij) in the job order on machine M_j, where u_i and v_j are the number of operations of job A_i and on machine M_j, respectively. In addition, there exists $(u_i + 1)(v_j + 1)$ possibilities of insertion, in each case Algorithm 1 can decide wether the corresponding graph $G(MO, JO)$ is acyclic. But because of the sequence condition we know, that for open shop problems in the cases listed below a new sequence is generated:

1. Set $rk(ij) = 1$, i.e., insert (ij) as source in the sequence graph, i.e. as direct predecessor of the first operation of job A_i and of the first operation on machine M_j, and update the ranks of all successors of (ij).

2. Set $rk(ij) = k + 1$, i.e., insert (ij) as direct successor of an operation of job A_i or of an operation on machine M_j with rank k and update the ranks of all successors of (ij).

3. If there are two operations (il) for job A_i and (kj) on machine M_j with $rk(il) = rk(kj)$ then insert (ij) as direct successor of one of the two and as direct predecessor of the other one and update the corresponding ranks. In both cases a cycle cannot be generated because there exist no path between both operations in the sequence graph.

For a job shop problem we have in addition to take into account the given machine order of job A_i, i.e., some of the described cases are omitted. Other given precedence constraints requires a further modification. We illustrate the insertion idea by an example:

Example 4 *Consider the following sequence PO on SIJ. Here we write* $po_{ij} = .$ *if* $(ij) \notin SIJ$. *Now we construct sequences by insertion of the operation* (12) *such that the properties above described are satisfied. There are at most 6 new sequences on* $SIJ \cup \{(12)\}$. *Case 1 yields* PO_1, *case 2 yields* PO_2 *and* PO_3 *and by case 3 we obtain* PO_4 *and* PO_5. *The updated ranks are marked. Note, that the last option yields a new sequence* (PO_6) *as well, because there is no path from* (32) *to* (13) *in the sequence graph of PO.*

$$PO = \begin{bmatrix} 1 & . & 2 \\ 3 & . & . \\ 2 & 1 & . \end{bmatrix},$$

$$PO_1 = \begin{bmatrix} \mathbf{2} & \mathbf{1} & \mathbf{3} \\ \mathbf{4} & . & . \\ \mathbf{3} & \mathbf{2} & . \end{bmatrix}, PO_2 = \begin{bmatrix} 1 & \mathbf{2} & \mathbf{3} \\ 3 & . & . \\ 2 & 1 & . \end{bmatrix}, PO_3 = \begin{bmatrix} 1 & \mathbf{3} & 2 \\ 3 & . & . \\ 2 & 1 & . \end{bmatrix},$$

$$PO_4 = \begin{bmatrix} 1 & \mathbf{2} & \mathbf{3} \\ \mathbf{5} & . & . \\ \mathbf{4} & \mathbf{3} & . \end{bmatrix}, PO_5 = \begin{bmatrix} 3 & \mathbf{2} & \mathbf{4} \\ \mathbf{5} & . & . \\ \mathbf{4} & 1 & . \end{bmatrix}, PO_6 = \begin{bmatrix} 1 & \mathbf{3} & 2 \\ \mathbf{6} & . & . \\ \mathbf{5} & \mathbf{4} & . \end{bmatrix}.$$

Note, that among the new sequences PO_3 is the best one in the case of makespan minimization. For each choice of processing times $C_{max}(PO_3) \leq C_{max}(PO_k)$ holds for all $k = 1, \ldots, 6$. The reason is the structure of all paths in the correponding sequence graphs. We call a path trivial, if it contains only operations of a single job or of a single machine, respectively. In the sequence graph of PO_3 there exists only one nontrivial path: $(32) \rightarrow (31) \rightarrow (21)$. But $(32) \rightarrow (31) \rightarrow (21)$ is also a path or a part of a path in any other sequence graph because it is contained in the sequence graph of PO. A sequence PO is *reducible* to a sequence PO^*, if for all matrices PT the inequality $C_{max}(PO^*) \leq C_{max}(PO)$ holds. Therefore all sequences are

of interest which are not reducible to another one, we call them *irreducible* sequences. Among all irreducible sequences there must be an optimal one. WILLENIUS presents in [19] results in the irreducibility theory with respect to other regular objective functions.

4.2 Distribution of Sequences in Shop Problems

Because of its relevance for practical applications the flow shop problem was investigated more in detail than other job shop cases. For flow shop problems, each combination of MO and JO is feasible, i.e., there exist $(n!)^m$ sequences. In BRÄSEL ET AL. [3] it is shown, that the matrix MO strong influences the properties of a job shop problem. We say that MO_1 is isomorph to MO_2 if MO_2 is obtained from MO_1 by permuting the job numbers and machine numbers. The number of sequences is the same in both cases. In the case of makespan minimization, the problem with given MO has the same properties as MO^{-1}, i.e. the reverse machine order. From the mathematical point of view, each isomorphism class has its own properties.

Example 5 *A shop problem with 3 jobs and 3 machines is given. Then there are* $(m!)^n(n!)^m = 46.656$ *combinations (MO,JO), 27.492 are infeasible and 19.164 are feasible.*

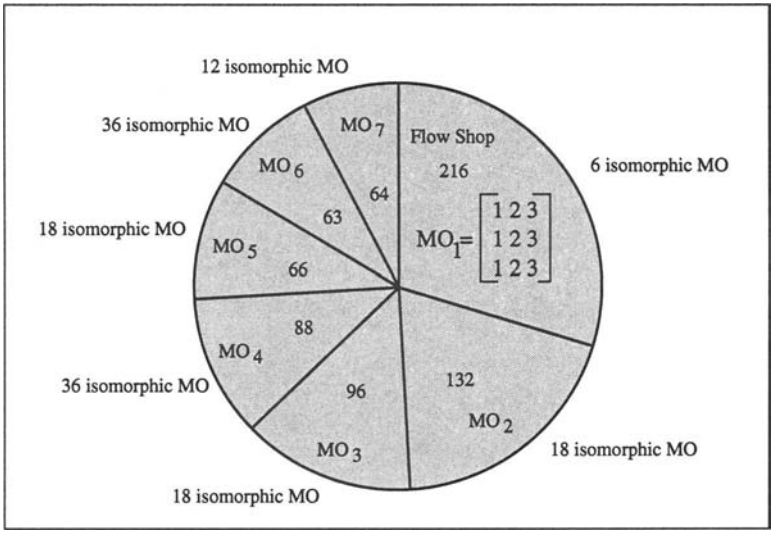

Figure 5: Number of sequences in different job shop problems

There exists 10 isomorphism classes with respect to the contained MO. If we consider makespan minimization, three of them can be deleted because MO is isomorphic to MO^{-1}. In [3] a formular for the number of isomorphism classes of the set of all machine order matrices is developed. Figure 5 shows the distribution of all sequences in the open shop to the different job-shops.

$$MO_2 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{bmatrix}, MO_3 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}, MO_4 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

$$MO_5 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix}, MO_6 = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 3 & 1 & 2 \end{bmatrix}, MO_7 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

For this small parameter the number of sequences range between 63 and 216. Note, that in the case $(n, m) = (3, 4)$, the smallest number is 473 and the greatest number is 13.824. It is interesting, that the class MO_6 has the minimum number of sequences, although it is not a Latin square.

5 Generalization of the Model

The model can be generalized for other shop scheduling problems. First we consider shop problems, where preemption is allowed, cf. BRÄSEL AND HENNES [4].
Let $z_{ij} - 1$, $z_{ij} \geq 1$, be the number of preemptions of operation (ij). Then each operation of *SIJ* is splitted in an ordered set of split operations $\{(ij)_k \mid k = 1, \ldots, z_{ij}\}$, where the split operation $(ij)_k$ has to be processed before the split operation $(ij)_{k+1}$ can start, $k = 1, \ldots, z_{ij} - 1$. We denote the operation set with all split operation *pSIJ*. Now we define analogously to shop problems without *pmtn* the digraphs $G^p(MO) = (pSIJ, A_{MO})$, $G^p(JO) = (pSIJ, A_{JO})$ and $G^p(MO, JO) = (pSIJ, A)$ where $A = A_{MO} \cup A_{JO}$ holds.

- The digraph $G^p(MO) = (SIJ, A_{MO})$ contains all arcs which are given by the direct precedence constraints between two split operations for the same job A_i, $i = 1, \ldots, n$, i.e. in all machine orders.
- The digraph $G^p(JO) = (SIJ, A_{JO})$ contains all arcs which are given by the direct precedence constraints between two split operations on the same machine M_j, $j = 1, \ldots, m$, i.e. in all job orders.

- The digraph $G^p(MO, JO) = (pSIJ, A)$ contains all arcs of $A = A_{MO} \cup A_{JO}$.

We call $G^p(MO, JO)$ a preemptive sequence graph, if it is acyclic. Now we define the following "preemptive" matrices whose elements are ordered sets:

- the preemptive machine order matrix $pMO = [\{mo_{ij}^1, \ldots, mo_{ij}^{z_{ij}}\}]$, where mo_{ij}^k is the rank of the split operation $(ij)_k$ in $G^p(MO)$;

- the preemptive job order matrix $pJO = [\{jo_{ij}^1, \ldots, jo_{ij}^{z_{ij}}\}]$, where jo_{ij}^k is the rank of the split operation $(ij)_k$ in $G^p(JO)$;

- the preemptive sequence (matrix) $pPO = [\{po_{ij}^1, \ldots, po_{ij}^{z_{ij}}\}]$, where po_{ij}^k is the rank of the vertex $(ij)_k$ in the sequence graph $G^p(MO, JO)$.

To obtain a *preemptive schedule* pC we have to split each processing time p_{ij} in z_{ij} parts: $p_{ij} = \sum_{k=1}^{z_{ij}} p_{ij}^k$. Then we get:

- the preemptive processing time matrix $pPT = [\{p_{ij}^1, \ldots, p_{ij}^{z_{ij}}\}]$, where p_{ij}^k is the processing time of the split operation $(ij)_k$.

- the preemptive schedule $pC = [\{c_{ij}^1, \ldots, c_{ij}^{z_{ij}}\}]$, where c_{ij}^k is the completion time of the split operation $(ij)_k$.

The matrices H, T and W can also be modified to the preemptive case. Algorithm 1- 4 can be adopted easily, cf. BRÄSEL AND HENNES [4].
The model is also applicable for flow shop problems, job shop problems and mixed shop problems with preemption, since further precedence constraints can be easily integrated.

If we consider shop problems where a job has to be processed more than once on a machine, this model can also be applied. In this case the split matrix Z and the corresponding matrix pPT are given.

For shop problems where preemption is allowed we have no fixed splitting of the operations. Thus additionally we have to find a splitting of the operations. The section closes with an example.

Example 6 *In Figure 6 a preemptive sequence graph with 3 jobs and 3 machines is shown. We have* $z_{ij} = 2$ *for* $(ij) \in \{(21), (31), (32)\}$. *All other operations are not splitted.*

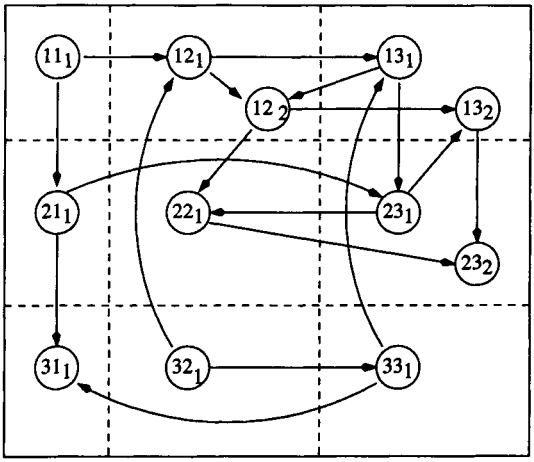

Figure 6: Preemptive sequence graph

The corresponding preemptive matrices are:

$$pMO = \begin{bmatrix} \{1\} & \{2,4\} & \{3,5\} \\ \{1\} & \{3\} & \{2,4\} \\ \{3\} & \{1\} & \{2\} \end{bmatrix}, pJO = \begin{bmatrix} \{1\} & \{2,3\} & \{2,4\} \\ \{2\} & \{4\} & \{3,5\} \\ \{3\} & \{1\} & \{1\} \end{bmatrix},$$

$$pPO = \begin{bmatrix} \{1\} & \{2,4\} & \{3,5\} \\ \{2\} & \{5\} & \{4,6\} \\ \{3\} & \{1\} & \{2\} \end{bmatrix}.$$

Using the preemptive processing time matrix pPT the preemptive schedule pC can be computed:

$$pPT = \begin{bmatrix} \{3\} & \{2,2\} & \{4,3\} \\ \{4\} & \{3\} & \{3,1\} \\ \{6\} & \{2\} & \{3\} \end{bmatrix}, pC = \begin{bmatrix} \{3\} & \{5,11\} & \{9,15\} \\ \{7\} & \{15\} & \{12,16\} \\ \{13\} & \{2\} & \{5\} \end{bmatrix}.$$

Again all objective functions $f(C_1, \ldots, C_n)$ can be easily computed from this schedule with $C_1 = 15$, $C_2 = 16$ and $C_3 = 13$.

The sequence pPO has the following properties, where r is the maximal rank in pPO:

(a) Each element of $\{1, \ldots, r\}$ occurs at most once in each row of PO and in each column of PO, respectively
(b) To any element $po_{ij}^k = l > 1$ there is an element $l-1$ in row i or column j of pPO (*sequence condition*).

6 LiSA: A Library of Scheduling Algorithms

The described model is applied in the software package LiSA - A Library of Scheduling Algorithms, which is developed for solving deterministic scheduling problems, especially for shop problems. All notations used in this paper are used in LiSA, too. We refer the interested reader to the LiSA-homepage: http://lisa.math.uni-magdeburg.de. The following example shows the using of LiSA. We consider an open shop problem with $m = 4$

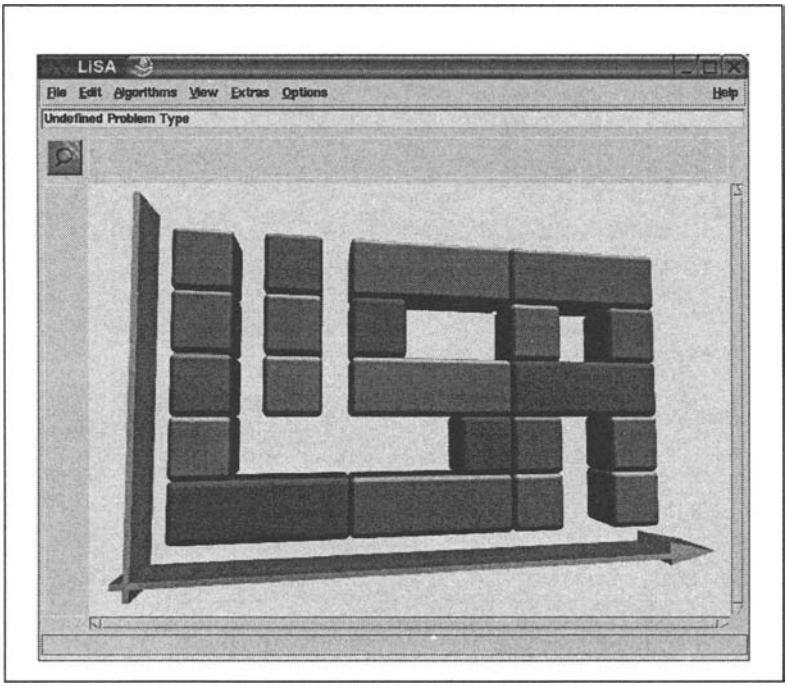

Figure 7: LiSA - A Library of Scheduling Algorithms

machines, $n = 4$ jobs and makespan minimization without any additional

constraints. The processing times are given by the following matrix PT:

$$PT = \begin{bmatrix} 12 & 6 & 15 & 7 \\ 13 & 6 & 7 & 13 \\ 3 & 14 & 8 & 7 \\ 10 & 13 & 9 & 7 \end{bmatrix}$$

What steps are necessary? Figure 8 shows a snapshot of the corresponding LiSA windows. After starting LiSA we choose the `New` button in the `File` menue. The problem type window is opened and we enter our problem in the $\alpha \mid \beta \mid \gamma$ denotation, namely $O \mid\mid C_{max}$. Moreover, enter $n = 4$ and $m = 4$. Now LiSA provides us with all modules for the considered problem. We start with the input of the processing times (buttons `Edit`, `Parameter`). We can do it by hand or by using the random generator. Note that all input data can also be read from a file.

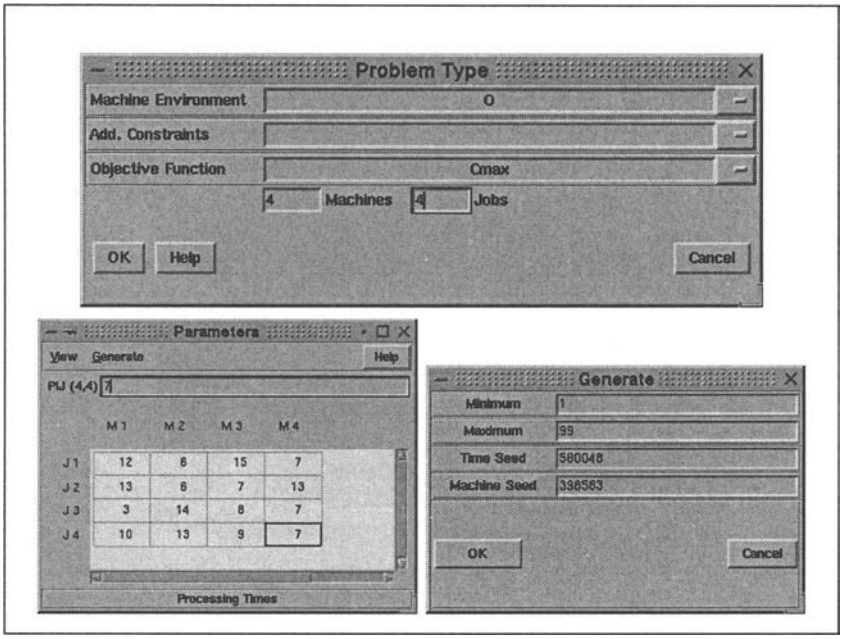

Figure 8: Input of the example

In the `Algorithms` menue exact and heuristic algorithms are available. Figure 9 shows some possibilities for the considered example. For instance,

the LPT-Rule (Longest Processing Time First) can be applied. After the construction of a first schedule we are able to use neighbourhood search algorithms, here simulated annealing with the 3_CR neighbourhood is chosen. In LiSA simulated annealing, iterative improvement, threshold acceptance and tabu search are available. Finally, matching heuristics can be used for the problem, here with minimal bottleneck objective function.

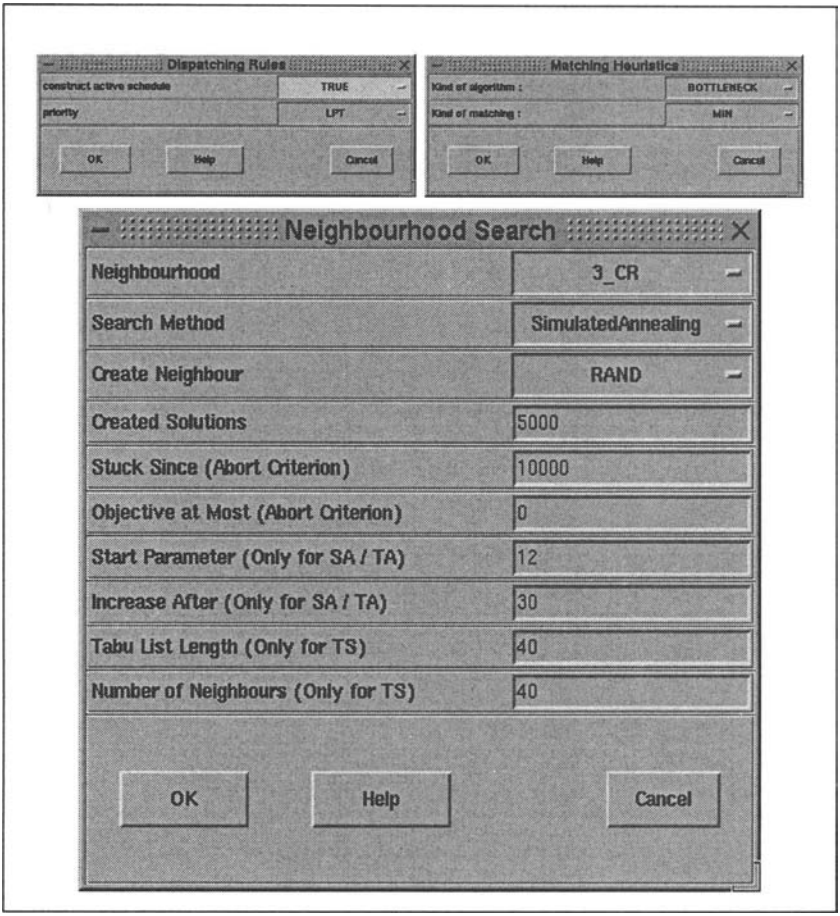

Figure 9: Heuristic algorithms for the considered problem

Usually a call of an algorithm produces the Gantt chart of the generated schedule , cf. Figure 11, but there are also other output options, cf. Figure 10. Here we can choose the sequence matrix or the sequence graph in the

`View` menue and the schedule described as the matrix of completion times or as a Gantt chart. Note, that under `Options` some options of the Gantt chart can be chosen: for instance, it can be machine oriented or job oriented or the critical path may be highlighted. If the number of jobs or machines is large, the Gantt chart is complex and the use of the zoom makes sense.

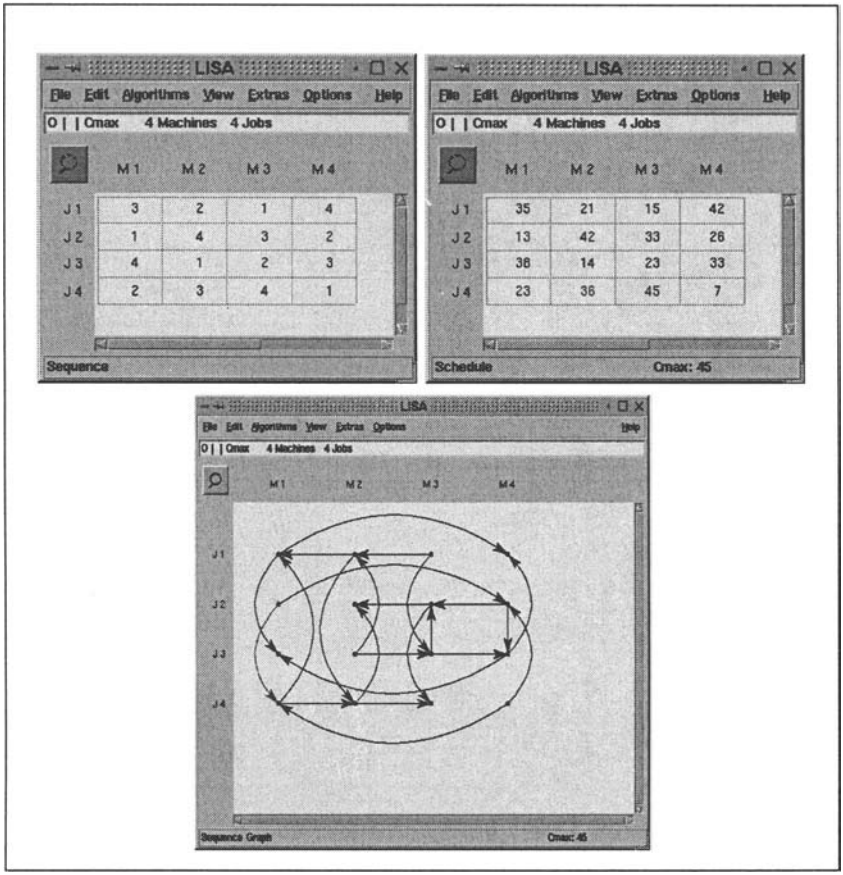

Figure 10: Output of LiSA: PO, C and G(MO, JO)

LiSA has some special modules, the most important one is the complexity module. Whenever LiSA has the $\alpha \mid \beta \mid \gamma$ notation of a problem, it determines the complexity status and gives, in addition, a complete reference. This module uses the data base of the scheduling group in Osnabrück. The software LiSA has a modular structure. The main part of LiSA con-

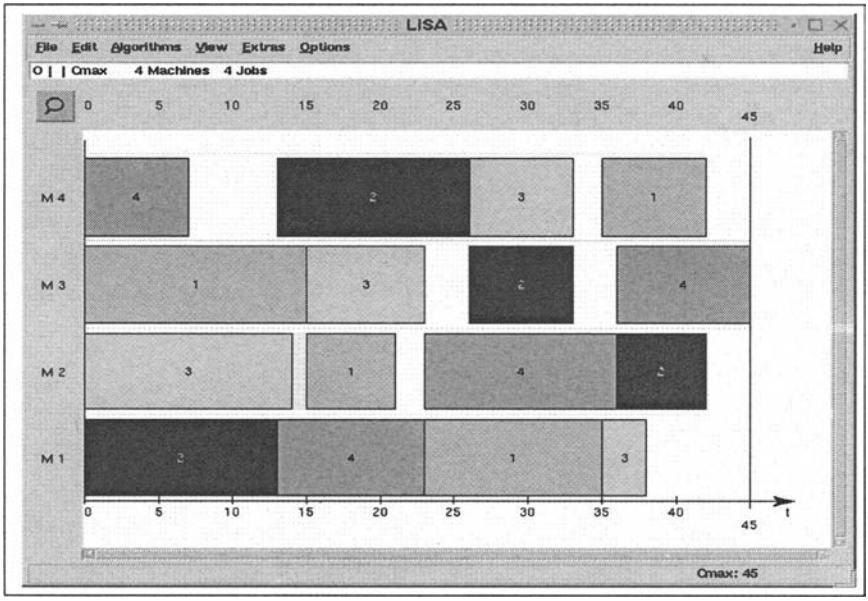

Figure 11: Output of LiSA: Gantt chart

tains all basic algorithms related to the model, the input and output procedures and the modules of the graphical user interface (GUI). Additionally, the program data is managed here and the work with external algorithms is coordinated.

In LiSA all algorithms used for solving scheduling problems are encapsulated in external modules. These modules are autonomous binaries with a common command line interface. They communicate with the main program via files. So they can be used both from within the LiSA GUI and independently without any GUI in a batch mode. The inclusion of new algorithms in LiSA is done by an algorithm description file and a corresponding help file in HTML format.

References

[1] Bräsel H (1990) Latin Rectangle in Scheduling Theory. Habilitation (in German), University Magdeburg, Germany

[2] Bräsel H, Harborth M, Tautenhahn T, Willenius P (1999) On the set of solutions of the open shop problem. Annals of Operations Research 92:241–263

[3] Bräsel H, Harborth M, Tautenhahn T, Willenius P (1999) On the hardness of the classical job shop problem. Annals of Operations Research 92:265–279

[4] Bräsel H, Hennes H (2004) On the open-shop problem with preemption and minimizing the average completion time. European Journal of Operational Research 157:607–619

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

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

[7] Bräsel H, Kluge D, Werner F (1994) A polynomial algorithm for the $n|m|O, t_{ij} = 1, tree|C_{max}$ open-shop problem. European Journal of Operational Research 72:125–134

[8] Bräsel H, Kluge D, Werner F (1995) A polynomial algorithm for an open shop problem with unit processing times and tree constraints. Discrete Applied Mathematics 59:11–21

[9] Bräsel H, Kluge D, Werner F (1996) Polynomial time algorithms for special open shop problems with precedence constraints and unit processing times. RAIRO Operations Research 30:65–79

[10] Bräsel H, Tautenhahn T, Werner F (1993) Constructive heuristic algorithms for the open shop problem. Computing 51:95–110

[11] Brucker P (2001) Scheduling Algorithms, Third Edition. Springer Verlag, Berlin, Heidelberg, New York

[12] Dhamala TN (2002) Shop Scheduling Solution-Spaces with Algebraic Characterizations. Dissertation, Shaker Verlag, Mathematics

[13] Graham RE, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. 4:287–326

[14] Harborth M (1999) Structural Analysis of Shop Scheduling Problems: Counting Problems, Potential Optimality and New Enumeration Algorithms. Dissertation (in German), GCA-Verlag, Forschen und Wissen: Mathematik

[15] Sotskow Y, Tautenhahn T, Werner F (1999) On the application of insertion techniques for job shop problems with setup times. RAIRO 33(2):209–245

[16] Tautenhahn T (1993) Open-Shop Problems with Unit Processing Times. Dissertation (in German), University Magdeburg

[17] Tautenhahn T (1996) On unit-time open shops with additional restrictions. ZOR 43(2):215–231

[18] Werner F, Winkler A (1995) Insertion techniques for the heuristic solution of the job shop problem. Discrete Applied Mathematics 58(2):191–211

[19] Willenius P (2000) Irreducibility Theory for Shop Scheduling Problems. Dissertation (in German), Shaker Verlag, Mathematics