INSTITUT FÜR WIRTSCHAFTSTHEORIE
UND OPERATIONS RESEARCH
UNIVERSITÄT KARLSRUHE

Heuristic Algorithms for Job–Shop Scheduling Problems with Stochastic Precedence Constraints

Klaus Neumann
Welf G. Schneider

Report WIOR–498

Juni 1997

TECHNICAL REPORT

Universität Karlsruhe

Kaiserstraße 12, D–76128 Karlsruhe

INSTITUT FÜR WIRTSCHAFTSTHEORIE
UND OPERATIONS RESEARCH
UNIVERSITÄT KARLSRUHE

Heuristic Algorithms for Job–Shop Scheduling Problems with Stochastic Precedence Constraints

Klaus Neumann
Welf G. Schneider

Report WIOR–498

Juni 1997

submitted to Annals of Operations Research 6/97

This research was partially supported by the Deutsche Forschungsgemeinschaft (Grant Ne 137/4).

Abstract:

This paper deals with job–shop scheduling with stochastic precedence constraints given by so–called OR networks. At first, a job–shop problem with stochastic OR network precedence constraints is described where the expected makespan is to be minimized. An example shows where such a problem occurs in practice. Next, the concept of an aggregate schedule is discussed, which represents a deterministic static scheduling policy for our stochastic problem. The construction of an appropriate aggregate disjunctive graph permits us to adapt the shifting bottleneck heuristic. After that, a priority–rule method is proposed for finding an approximate aggregate schedule. An experimental performance analysis shows that both heuristics provide good approximate solutions. Finally, we briefly discuss a flow–shop problem with OR network precedence constraints.

Keywords:

stochastic scheduling, job–shop and flow–shop scheduling, GERT networks, heuristics

Contents

1 Introduction

During the last ten years, literature on job–shop scheduling has grown enormously. In particular, a large number of exact and heuristic algorithms for solving the "classic" problem $J||C_{\max}$ have been proposed (for an overview compare Błażewicz et al. 1996, Brucker 1995, and Pinedo 1995). Job–shop problems with additional (deterministic) precedence constraints for the jobs, however, have been discussed extremely rarely (see, for example, Strusevich 1997). In job–shop scheduling with stochastic processing times not much research has been done, either (cf. Pinedo & Schrage 1982 and Pinedo 1995). Job–shop scheduling with stochastic precedence constraints has not been treated at all in the open literature so far.

In this paper, we are going to deal with job–shop scheduling where the precedence constraints are given by special stochastic project networks, so–called *GERT networks*. GERT networks have been introduced to model, schedule, and control projects whose evolution in time is not uniquely determined in advance and where feedback is allowed. That is, we have a stochastic evolution structure of the project in question (for a detailed discussion of GERT networks we refer to Neumann & Steinhardt 1979 and Neumann 1990). Each GERT network is associated with a random experiment, which consists of executing the corresponding project. The sample space Ω represents the set of all possible outcomes of the project, also called *project* or *network realizations*. The individual activities of the project are assumed to be carried out without interruption, i.e. we only consider the nonpreemptive case.

Single–machine scheduling problems with GERT network precedence constraints have been discussed by Neumann (1989, 1990), Bücker (1992), and Bücker et al. (1994). Parallel–machine scheduling problems with GERT precedence constraints have been studied by Zimmermann (1995) and Neumann & Zimmermann (1997). First results on job–shop problems with precedence constraints given by special GERT networks can be found in Schneider (1997).

In Section 2 of this paper we briefly sketch the basics from GERT networks needed in what follows. In Sections 3 and 4 we present a job–shop model with stochastic precedence constraints, where the expected makespan is to be minimized, and discuss some application. In Section 5 we introduce an appropriate concept of a deterministic scheduling policy for our stochastic job–shop problem, the so–called *aggregate schedule*. Section 6 is devoted to the construction of an *aggregate disjunctive graph* for the stochastic job–shop problem in question. This permits us to apply the well–known shifting bottleneck heuristic, which provides an approximate aggregate schedule. In Section 7 we propose a second approach to solving the stochastic job–shop problem by adapting a priority–rule–based heuristic of the Giffler–Thompson type. In Section 8 we present an experimental performance analysis for both heuristics. Section 9 contains some supplements such as stochastic flow–shop problems and cyclic stochastic precedence constraints.

2 Basics from the theory of GERT networks

We present a short review of some basic material from the theory of GERT networks in an intuitive way (for more details see Neumann 1990). For the basic concepts from the theory of graphs and networks needed in the following, we refer to Neumann (1990) and Ahuja et al. (1993).

For GERT networks, we use the activity–on–arc representation of projects known from CPM and PERT networks (see, for example, Elmaghraby 1977). A GERT network has exactly one source (corresponding to the beginning event of the project) and at least one sink (corresponding to terminal events of the project). As compared to CPM and PERT networks, GERT networks possess more general arc weights, several different types of nodes, and cycles to represent feedback. Because of the latter property, some activities may be carried out several times during a single project realization.

Each arc $< k, l >$ of a GERT network with initial node k and final node l is assigned a *weight vector* (p_{kl}, F_{kl}). $p_{kl} > 0$ is the *conditional execution probability* of the corresponding activity $< k, l >$ given that project event k has occurred. F_{kl} is the *conditional distribution function* of the nonnegative *duration* of activity $< k, l >$ given that activity $< k, l >$ is executed. p_{kl} and F_{kl} are assumed to be independent of how many times project event k has occurred or activity $< k, l >$ has been carried out before, respectively.

GERT networks possess six different node types resulting from combination of three possible entrance sides and two exit sides of a node. For simplicity, we consider only one entrance side of a node, the so–called *OR entrance*, which says that the node is "activated" (i.e. the corresponding project event occurs) every time when an incoming activity has been terminated. We speak of a *deterministic exit* of a node if all outgoing activities are carried out when the node has been activated. If exactly one outgoing activity is carried out when the node has been activated, we have a *stochastic exit*. A node with at most one successor is supposed to have a stochastic exit (if we speak of a successor or predecessor of a node in what follows, we always mean an immediate successor or predecessor, respectively). Hence, the special GERT networks we are going to discuss in this paper and which are called *OR networks* possess two types of nodes: *deterministic nodes* with OR entrance and deterministic exit and *stochastic nodes* with OR entrance and stochastic exit.

Each realization $\omega \in \Omega$ of an OR network or the corresponding project, respectively, begins with the activation of its source at time zero. Subsequently, for each stochastic node k activated, project realization ω specifies exactly one activity with initial node k to be carried out. Note that if stochastic node k is activated several times during project realization ω, different activations of node k may result in the execution of different outgoing activities. Moreover, ω specifies the realized duration of each activity execution.

An OR network is supposed to satisfy two assumptions. *Assumption A1* refers to the stochastic evolution structure of the corresponding project and expresses some Markov and independence properties. An OR network can then be associated with several Markov renewal processes, whose states represent the nodes and which evolve in time independently of one another and cease to exist and give birth to new processes once deterministic nodes are activated (for a precise formulation of *A1* see Bücker et al. 1994 and Neumann 1990).

Assumption A2 says that for each deterministic node k and any two distinct successors l_1 and l_2 of k, it must hold that $\mathcal{R}(l_1) \cap \mathcal{R}(l_2) = \emptyset$ where $\mathcal{R}(l)$ denotes the set of nodes reachable from node l. From *A2* it follows that every node which belongs to a cycle is stochastic and that each activity outside any cycle is carried out at most once during a single project realization.

Let q_k be the probability that node k is activated (during a single project execution). Exploiting *A1*, the activation probabilities q_k of the nodes k of an OR network can be computed by solving a system of linear equations (cf. Neumann 1990). The probability π_{kl} that activity $< k, l >$ is carried out (during a single project execution) is

$$\pi_{kl} = q_k p_{kl} \tag{1}$$

For simplicity, we only consider acyclic OR networks all of whose activities $< k, l >$ have deterministic durations in what follows. How to deal with cyclic OR networks will be discussed in Section 9.

3 Job–shop scheduling with OR network precedence constraints

We assume that the reader is familiar with the "classic" job–shop problem $J||C_{\max}$ (e.g. see Błażewicz et al. 1996, Brucker 1995, or Pinedo 1995) and we only review some notation. We consider a job–shop problem with job set $\mathcal{J} = \{J_1, \ldots, J_n\}$ and machine set $\mathcal{M} = \{M_1, \ldots, M_m\}$. Job J_j consists of m operations $O_{i_1 j}, \ldots, O_{i_m j}$ that have to be processed in this order on the machines $M_{i_1}, \ldots, M_{i_m}$, respectively, where $(i_1, \ldots, i_m)$ is some given permutation of $(1, \ldots, m)$. The following investigations also hold for the cases where some jobs are not processed on all machines $M_1, \ldots, M_m$ and where some jobs are processed on a machine more than once. Let $t_{ij} > 0$ be the (deterministic) processing time of operation O_{ij}. Then $t_j := \sum_{i=1}^{m} t_{ij}$ is the *processing time* or duration of job J_j.

Assume that *stochastic precedence constraints* are prescribed for the jobs given by an acyclic OR network where each job J_j corresponds to exactly one arc or activity $< k_j, l_j >$ of the network with duration t_j and *conditional execution probability* p_j. We also speak of the *initial node* k_j and *final node* l_j of job J_j. Recall that by (1), the *execution probability* of job J_j is

$$\pi_j = q_{k_j} p_j \tag{2}$$

where q_{k_j} is the activation probability of node k_j.

Each realization of the underlying OR network corresponds to a job–shop problem with deterministic precedence constraints where in general not all jobs are performed. Job J_j is said to *precede* job $J_{j'}$ and $J_{j'}$ is said to *follow* J_j (in symbols, $J_j \prec J_{j'}$ or $J_{j'} \succ J_j$) precisely if there is a path in the OR network from final node l_j of job J_j to initial node $k_{j'}$ of job $J_{j'}$, where l_j and $k_{j'}$ may coincide. If J_j and $J_{j'}$ are carried out in one and the same network realization, then J_j has to be carried out before $J_{j'}$. The strict order $\prec$ in the job set $\mathcal{J}$ induces a corresponding strict order in the set $\mathcal{O}$ of all operations also denoted by $\prec$, where in addition, the operation sequences of the individual jobs have to be observed.

Let k be a *deterministic node* of the OR network with successors $l_1, \dots, l_r$. Then the emanating arcs $< k, l_1 >, \dots, < k, l_r >$ correspond to r jobs, say jobs $J_1, \dots, J_r$, all of whose conditional execution probabilities equal one and which can be performed simultaneously (see Fig. 1). That is, the jobs compete with each other for the machines (*job competition*). Now let k be a *stochastic node*. Then the emanating arcs $< k, l_1 >, \dots, < k, l_r >$ correspond to jobs $J_1, \dots, J_r$ whose conditional execution probabilities sum up to unity and which are performed exclusively (Fig. 1). That is, there is *no job competition* between $J_1, \dots, J_r$.

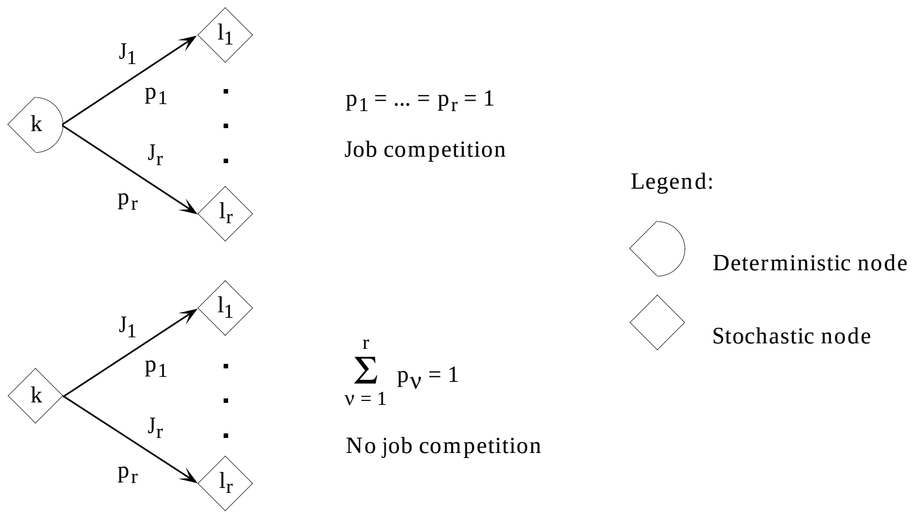


Figure 1: Deterministic and stochastic nodes

The *objective function* we want to minimize is the *expected makespan* $E(C_{\max})$, i.e. the expectation of the completion time of the last job to be performed. Using the three–field notation known from deterministic scheduling, our job–shop scheduling problem with precedence constraints given by an acyclic OR network is designated by $J|acyclOR|E(C_{\max})$. If the OR network contains only stochastic nodes, the jobs to be performed in any network realization have to be processed one after another and there is no optimization problem. The problem

$J|acyclOR|E(C_{\max})$ and its time complexity will be discussed more detailed in Section 5.

4 Application of job–shop scheduling with OR network precedence constraints

Job–shop problems with OR network precedence constraints occur in *mixed–model make–to–order production*, where several variants of some product are manufactured. We present an example, where several variants of some sporting car with respect to engine, transmission, and differential are manufactured.

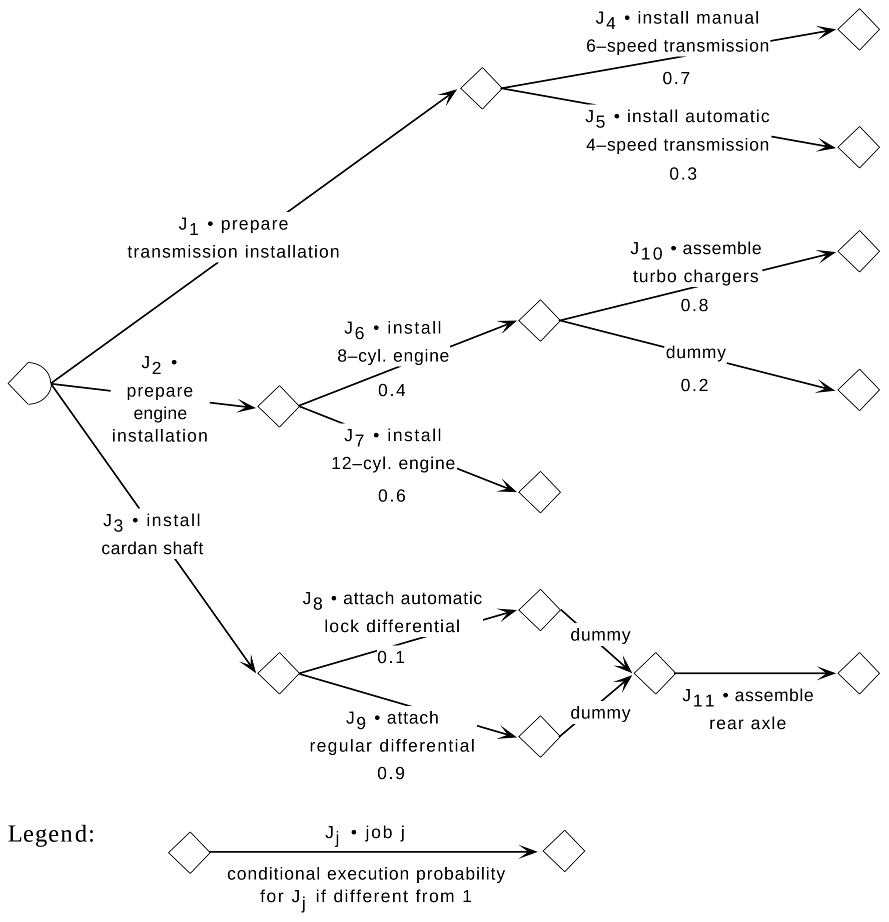
Example 1:

We consider the simplified automotive propulsion manufacturing for a certain kind of sporting car, where the customer has the choice to individually order a pattern of car–propulsion.

The customer is allowed to choose one of the following engines for his sporting car: 8–cyl. engine, bi–turbo–charged 8–cyl. engine, or 12–cyl. engine. Independent of the selected engine, the customer may choose either manual 6–speed transmission or automatic 4–speed transmission. Finally, the customer is allowed to order an automatic lock differential instead of a regular one, independent of engine and transmission preferences (an empirical investigation by Furmans (1995) shows that in production of individually ordered cars, the order probabilities of extra features are "highly uncorrelated"). For those order wishes, we are given corresponding order probabilities, e.g. we know that 60 % of the customers want to have the smooth–running 12–cyl. engine.

We assume that due to technological reasons, transmission, engine, and shafts with differential and rear axle can be assembled independently. Other technological production restrictions can be extracted from the OR network in Fig. 2, where additional "dummy activities" provide for consistent representation of the OR network.

Fig. 2 makes clear the meaning of a deterministic node in an OR network, namely, that all outgoing jobs are performed independently and, possibly, at the same time. A stochastic node indicates that exactly one outgoing job is executed. E.g., we want to assemble either a 12–cyl. engine or an 8–cyl. engine. The order probability of an 8–cyl. engine equals 0.4. We notice that the order probability of an 8–cyl.–turbo engine is 0.4×0.8. Turbo chargers are assembled (job J_{10}) with a probability of 0.8 given that an 8–cyl. engine has already been installed.



J1 • prepare transmission installation
J2 • prepare engine installation
J3 • install cardan shaft
J4 • install manual 6–speed transmission
0.7
J5 • install automatic 4–speed transmission
0.3
J6 • install 8–cyl. engine
0.4
J7 • install 12–cyl. engine
0.6
J8 • attach automatic lock differential
0.1
J9 • attach regular differential
0.9
J10 • assemble turbo chargers
0.8
dummy
0.2
dummy
dummy
J11 • assemble rear axle
Legend:
Jj • job j
conditional execution probability for Jj if different from 1

Figure 2: Example of automotive propulsion in mixed–model production

5 Schedules for J|acyclOR|E($C_{\max}$)

We want to introduce a deterministic static scheduling policy Φ with the property that for each network realization, we obtain the sequences of jobs processed on the individual machines and their start times in a unique way. Since such a deterministic scheduling policy aggregates, in a sense, schedules for all individual network realizations, it is called an *aggregate schedule*. An aggregate schedule Φ is defined on the set $\mathcal{O}$ of all operations, where $\Phi(O)$ for $O \in \mathcal{O}$ can be viewed as some deterministic start time of operation O. The precise definition of an aggregate schedule is as follows.

Definition 1:

A mapping $\Phi : \mathcal{O} \to I\!R_+$ is called an *aggregate schedule* if it is

(a) precedence–consistent, that is,

$$\text{for all } O_{ij}, O_{i'j'} \in \mathcal{O} \text{ with } ij \neq i'j'$$

$$\Phi(O_{ij}) + t_{ij} \leq \Phi(O_{i'j'}) \text{ exactly if } O_{ij} \prec O_{i'j'}$$

(b) feasible, that is,

$$\text{for all } J_j, J_{j'} \in \mathcal{J}, j \neq j', \text{ and all } M_i \in \mathcal{M}$$

$$\Phi(O_{ij'}) \notin [\Phi(O_{ij}), \Phi(O_{ij}) + t_{ij}[$$

and

(c) semiactive.

Condition (b) says that there are no overlapping operations O_{ij} and $O_{ij'}$ on one and the same machine M_i if neither $J_j \prec J_{j'}$ nor $J_{j'} \prec J_j$ holds. As to condition (c), recall that a schedule is said to be semiactive if no local left–shift of any operation can be performed, i.e. no left–shift without altering the job sequence on any machine (cf. Pinedo 1995).

Let Φ be an aggregate schedule, $\omega \in \Omega$ be a network realization, and $\mathcal{J}_\omega \subseteq \mathcal{J}$ and $\mathcal{O}_\omega \subseteq \mathcal{O}$ be the sets of jobs and operations, respectively, carried out in network realization ω. Then we define the schedule Φ_ω for network realization ω belonging to aggregate schedule Φ as follows.

Definition 2:

A mapping $\Phi_\omega : \mathcal{O}_\omega \to I\!R_+$ is called a *network realization schedule belonging to* Φ if it is feasible, semiactive, and has the property that

$$\text{for all } J_j, J_{j'} \in \mathcal{J}_\omega, j \neq j', \text{ and all } M_i \in \mathcal{M}$$

$$\Phi_\omega(O_{ij}) \leq \Phi_\omega(O_{ij'}) \text{ exactly if } \Phi(O_{ij}) \leq \Phi(O_{ij'})$$

Remark 1: Given Φ, Φ_ω is uniquely determined for each network realization $\omega \in \Omega$. $\Phi_\omega(O)$ represents the start time of operation $O \in \mathcal{O}_\omega$ in realization ω.

Remark 2: Φ_ω is assumed to be semiactive. Hence, to obtain Φ_ω from Φ, after deleting all operations $O \in \mathcal{O} \setminus \mathcal{O}_\omega$, some local left–shifts of the remaining operations have to be performed in general.

Remark 3: The schedules Φ and Φ_ω are assumed to be semiactive (instead of being active) and thus no global left–shifts including changes of job sequences

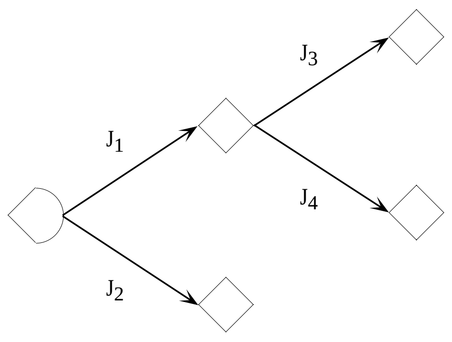


Figure 3: OR network

on the machines are permitted for the following reason: For one and the same aggregate schedule Φ and network realization ω, different global left–shifts might result in different network realization schedules Φ_ω, that is, the start times of operations would not necessarily be determined in a unique way. For a more detailed discussion we refer to Schneider (1997).

Example 2:

As an example, we consider a job–shop problem with four jobs and two machines where the precedence constraints are given by the acyclic OR network (without arc weights) depicted in Fig. 3. The order in which the operations have to be done on the jobs correspond to the order of the operations in Table 1, which also contains the processing times of the operations. An aggregate schedule for this job–shop problem is given by the Gantt chart in Fig. 4, where idle times are indicated by shaded areas. There are two possible network realizations. In realization ω_1 the jobs J_1, J_2, and J_3 are performed, in realization ω_2 the jobs J_1, J_2, and J_4 are performed. The network realization schedules Φ_{ω_1} and Φ_{ω_2} belonging to Φ are shown in Figs. 5 and 6. Note that to obtain Φ_{ω_2} from Φ, a left–shift of operation O_{14} has to be performed.

Table 1: Operation sequences and processing times

Job	Sequence of operations	Processing times
J_1	O_{21}, O_{11}	$t_{21} = 2, t_{11} = 1$
J_2	O_{12}, O_{22}	$t_{12} = 1, t_{22} = 2$
J_3	O_{13}, O_{23}	$t_{13} = 3, t_{23} = 2$
J_4	O_{24}, O_{14}	$t_{24} = 1, t_{14} = 1$

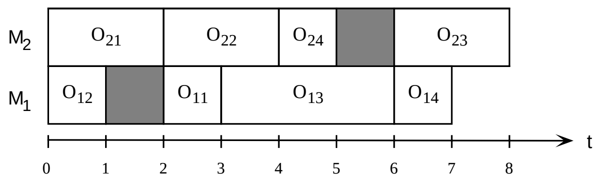


Figure 4: Aggregate schedule Φ

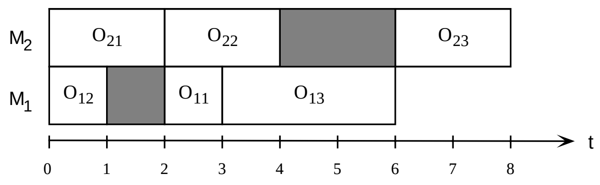


Figure 5: Network realization schedule Φ_{ω_1}

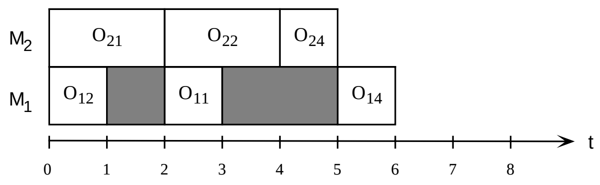


Figure 6: Network realization schedule Φ_{ω_2}

Let $P(\omega)$ be the probability that network realization ω occurs. $P(\omega)$ equals the product of the conditional execution probabilities p_j of all jobs $J_j \in \mathcal{J}_\omega$. Let $C_j(\Phi_\omega)$ be the completion time of job $J_j \in \mathcal{J}_\omega$ in network realization ω and $C_{\max}(\Phi_\omega)$ be the makespan when schedule Φ_ω is applied. Then the *expected makespan* when aggregate schedule Φ is applied is

$$E[C_{\max}(\Phi)] := \sum_{\omega \in \Omega} P(\omega) C_{\max}(\Phi_\omega) = \sum_{\omega \in \Omega} P(\omega) \max_{J_j \in \mathcal{J}_\omega} C_j(\Phi_\omega) \tag{3}$$

Let $\mathcal{S}$ be the set of all aggregate schedules. Then $\Phi^* \in \mathcal{S}$ is called an *optimal aggregate schedule* if its expected makespan is minimum:

$$E[C_{\max}(\Phi^*)] = \min_{\Phi \in \mathcal{S}} E[C_{\max}(\Phi)] \tag{4}$$

Let Ψ_ω^* be an optimal schedule for the deterministic job–shop problem corresponding to network realization ω.

Theorem 3:
It holds that

$$E[C_{\max}(\Phi^*)] \geq \sum_{\omega \in \Omega} P(\omega) C_{\max}(\Psi_\omega^*) \tag{5}$$

Proof:

Let Φ^*_ω be the schedule for network realization ω belonging to aggregate schedule Φ^*. Since Ψ^*_ω is optimal for network realization ω, we have $C_{\max}(\Psi^*_\omega) \leq C_{\max}(\Phi^*_\omega)$ and thus

$$\sum_{\omega \in \Omega} P(\omega) C_{\max}(\Psi^*_\omega) \leq \sum_{\omega \in \Omega} P(\omega) C_{\max}(\Phi^*_\omega) = E[C_{\max}(\Phi^*)]$$

□

The fact that the $>$ sign generally holds instead of $=$ in (5) means that we have to pay for restricting ourselves to (deterministic) aggregate schedules as scheduling policies with an increase in the minimum expected makespan. The quantity

$$LB := \sum_{\omega \in \Omega} P(\omega) C_{\max}(\Psi^*_\omega) \tag{6}$$

represents a lower bound on the minimum objective function value for aggregate schedules.

If the OR network contains only stochastic nodes, there is no optimization problem as already mentioned. In that case, there is only one schedule Ψ_ω for each network realization ω. Moreover, for any two different aggregate schedules Φ^1 and Φ^2, it holds that $\Phi^1_\omega = \Phi^2_\omega = \Psi_\omega$ for all $\omega \in \Omega$. Trivially, the $=$ sign holds to be true in (5).

Problem $J|acyclOR|E(C_{\max})$ is $\mathcal{NP}$–hard since it represents a generalization of the classic job–shop problem $J||C_{\max}$: if all n jobs emanate from a deterministic source into sinks, this special OR network corresponds to the classic case. Even $J2|acyclOR|E(C_{\max})$ is $\mathcal{NP}$–hard since it is a generalization of the flow–shop problem $F2|tree|C_{\max}$ and the latter problem is $\mathcal{NP}$–hard, cf. Lenstra et al. (1977) or Strusevich (1997).

6 Disjunctive graphs and shifting bottleneck heuristic

Since the problem of finding an optimal aggregate schedule is $\mathcal{NP}$–hard, we are going to compute an approximate aggregate schedule by some heuristic. The first method we propose is the well–known shifting bottleneck heuristic based upon the disjunctive–graph concept. At first we consider disjunctive graphs for deterministic precedence constraints.

6.1 Disjunctive graph for deterministic precedence constraints

Let N be an acyclic OR network and N_ω be that subnetwork of N which belongs to network realization ω. We want to construct the disjunctive (directed) graph G_ω for the job–shop problem with deterministic precedence constraints given by network N_ω and objective function $C_{\max}$.

We briefly review the construction of the disjunctive graph for the job–shop problem $J||C_{\max}$ (for details we refer to Pinedo 1995). Each operation $O \in \mathcal{O}$ is assigned a node of the disjunctive graph, which additionally contains a source r and a sink s. If job J_j is processed consecutively on machines M_i and $M_{i'}$, there is a *conjunctive arc* from operation node O_{ij} to operation node $O_{i'j}$. For two different jobs J_j and $J_{j'}$ to be processed on the same machine M_i, there are two *disjunctive arcs* between operation nodes O_{ij} and $O_{ij'}$ going in opposite directions. Moreover, there are conjunctive arcs from source r to all the first operations of the jobs and from all the last operations of the jobs to sink s. Fig. 7 illustrates that construction of a disjunctive graph where conjunctive and disjunctive arcs are indicated by solid and broken arrows, respectively.

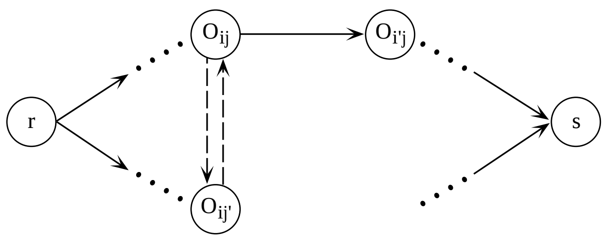


Figure 7: Disjunctive graph

Now we turn to deterministic precedence constraints given by network N_ω. First we consider the precedence relation given by two consecutive jobs J_j and $J_{j'}$. In that case, we introduce a conjunctive arc from the last operation of job J_j, say $O_{\beta_j j}$, to the first operation of job $J_{j'}$, say $O_{\alpha_{j'} j'}$ (see Fig. 8). Of course, there is no arc from last operation $O_{\beta_j j}$ to sink s in that case.

Second we consider several arcs emanating from a deterministic node in network N_ω, say, two arcs corresponding to jobs J_j and $J_{j'}$. Jobs J_j and $J_{j'}$ compete with each other for the machines. Thus, for each machine M_i, we have to introduce a pair of opposite disjunctive arcs between operations O_{ij} and $O_{ij'}$ (see Fig. 9). In addition, each job $J_{j''} \succ J_{j'}$ competes with job J_j for the machines. Hence, for each machine M_i, we have to introduce a pair of opposite disjunctive arcs between operations O_{ij} and $O_{ij''}$ as well. Analogously, each job $J_{j^*} \succ J_j$ competes with $J_{j'}$ and with each job $J_{j''} \succ J_{j'}$ for the machines, and corresponding pairs of disjunctive arcs must be introduced.

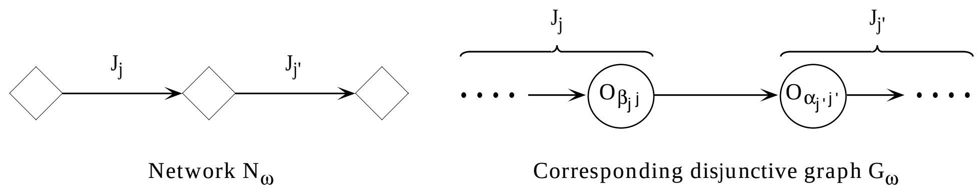


Figure 8: Network N_ω and corresponding disjunctive graph G_ω

Now let us return to Example 2 from Section 5 (see Fig. 3 and Table 1), where the network N_{ω_1} corresponding to network realization ω_1 is shown in Fig. 10. The associated disjunctive graph is depicted in Fig. 11.

As in the case without precedence constraints (see, for example, Pinedo 1995), a subset D of the arc set of the disjunctive graph is called a *feasible selection* if D contains all conjunctive arcs and one disjunctive arc from each pair of opposite arcs such that the induced directed graph is acyclic. Each feasible selection D corresponds to a feasible schedule Ψ_ω for the job–shop problem with precedence constraints given by N_ω.

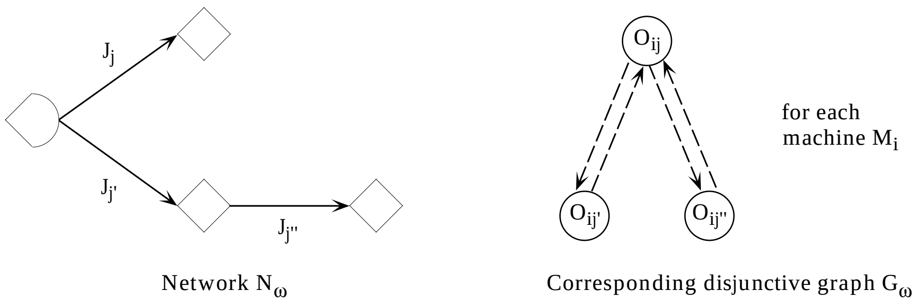


Figure 9: Network N_ω and corresponding disjunctive graph G_ω

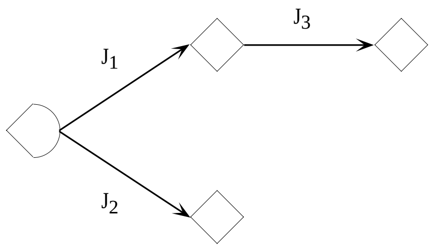


Figure 10: Network N_{ω_1}

Let G_ω be again the disjunctive graph corresponding to network N_ω and let $G_\omega(D)$ be the subgraph of G_ω with arc set D, where the weight of an arc

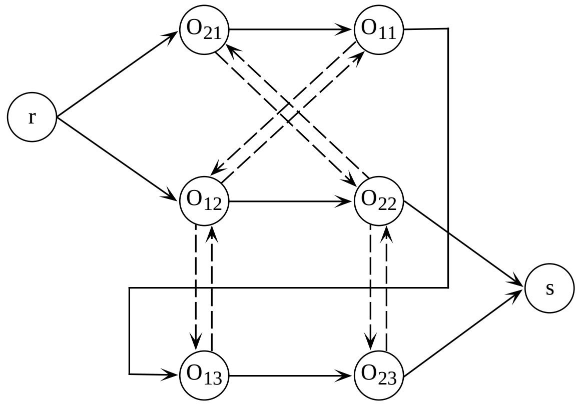


Figure 11: Disjunctive graph G_{ω_1}

emanating from operation node O_{ij} equals the processing time t_{ij} of O_{ij} (the arcs emanating from source r have weight zero). Then the length $L[G_\omega(D)]$ of a longest path in $G_\omega(D)$ from source r to sink s is equal to the makespan $C_{\max}(\Psi_\omega)$ for feasible schedule Ψ_ω. Moreover, if $\mathcal{D}$ is the set of all feasible selections and Ψ^*_ω is an optimal schedule, then

$$\min_{D \in \mathcal{D}} L[G_\omega(D)] = C_{\max}(\Psi^*_\omega). \tag{7}$$

An optimal schedule Ψ^*_ω for the job–shop problem corresponding to network realization ω can be computed by any branch–and–bound algorithm based upon the disjunctive–graph concept, for example, by the algorithms of Carlier and Pinson or of Brucker (see Carlier & Pinson 1989, Brucker et al. 1994, and Brucker 1995). To obtain an approximate (feasible) schedule Ψ_ω, the well–known shifting bottleneck heuristic can be used (cf. Adams et al. 1988, Balas et al. 1995, and Dauzère–Péres & Lasserre 1994). For small instances of the stochastic job–shop problem $J|acyclOR|E(C_{\max})$, the networks N_ω for all realizations $\omega \in \Omega$ of the underlying OR network can be generated and corresponding optimal schedules Ψ^*_ω can be computed. In addition, the lower bound LB on the minimum objective function value for aggregate schedules defined in (6) can be determined.

6.2 Aggregate disjunctive graph for $J|acyclOR|E(C_{\max})$

Our goal is to adapt the shifting bottleneck heuristic to the job–shop problem $J|acyclOR|E(C_{\max})$ with stochastic precedence constraints so that we can compute a (deterministic) aggregate schedule Φ. To this end, we are going to construct an appropriate disjunctive graph G, also called *aggregate disjunctive graph* because, in a sense, it aggregates the disjunctive graphs G_ω for all network realizations $\omega \in \Omega$. Applying the shifting bottleneck heuristic to G then provides an

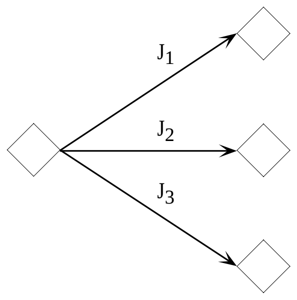


Figure 12: OR network N

Table 2: Operation sequences

Job	Operation sequence
J_1	O_{21}, O_{11}
J_2	O_{12}, O_{22}
J_3	O_{13}, O_{23}

aggregate schedule Φ.

We discuss some typical cases of stochastic precedence constraints given by an acyclic OR network N. *Case 1* deals with several jobs emanating from a stochastic node. As an example we consider the simple OR network of Fig. 12, where the operation sequences of the jobs are given in Table 2. Since the jobs J_1, J_2, and J_3 do not compete for the machines, we link them by introducing so–called *stochastic conjunctive arcs* from the last operation of J_1 to the first operation of J_2 and from the last operation of J_2 to the first operation of J_3 (in contrast to the *ordinary conjunctive arcs* introduced in Subsection 6.1). This provides the aggregate disjunctive graph G depicted in Fig. 13, where stochastic conjunctive arcs are indicated by bold arrows. To take into consideration that several jobs emanating from a stochastic node are carried out with a probability less than one each, the weight of an arc in an aggregate disjunctive graph emanating from operation node O_{ij} is to be equal to $\pi_j t_{ij}$, where π_j is again the execution probability of job J_j.

The length of the only path from source r to sink s in the aggregate disjunctive graph of Fig. 13 is $\sum_{j=1}^{3} \pi_j(t_{1j} + t_{2j}) = E[C_{\max}(\Phi)]$ for any aggregate schedule Φ. Later, we will see that the latter holds for any acyclic OR network with only stochastic nodes.

As *Case 2* we consider Example 2 from Section 5, where the OR network N is shown in Fig. 3 and the operation sequences of the jobs are given in Table 1. There are a deterministic and a stochastic node both with outgoing jobs. The

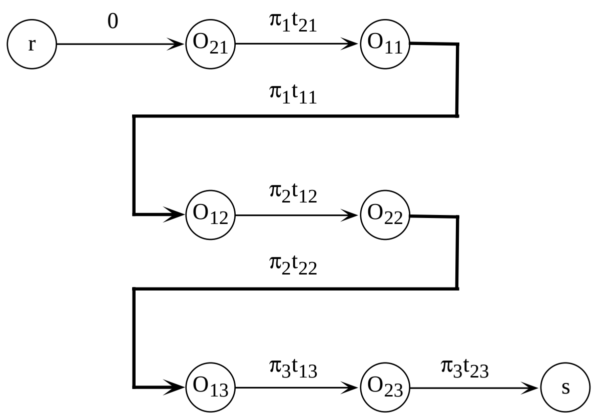


Figure 13: Aggregate disjunctive graph G for OR network N of Fig. 12

corresponding aggregate disjunctive graph G (without arc weights) is depicted in Fig. 14. The pairs of opposite disjunctive arcs result from the competition of job J_2 with jobs J_1, J_3, and J_4 for the machines. Since jobs J_1 and J_3 on the one hand and jobs J_1 and J_4 on the other hand can be carried out consecutively, there are ordinary conjunctive arcs $< O_{11}, O_{13} >$ and $< O_{11}, O_{24} >$. The stochastic node with outgoing jobs J_3 and J_4 in N leads to the stochastic conjunctive arc $< O_{23}, O_{24} >$ in G.

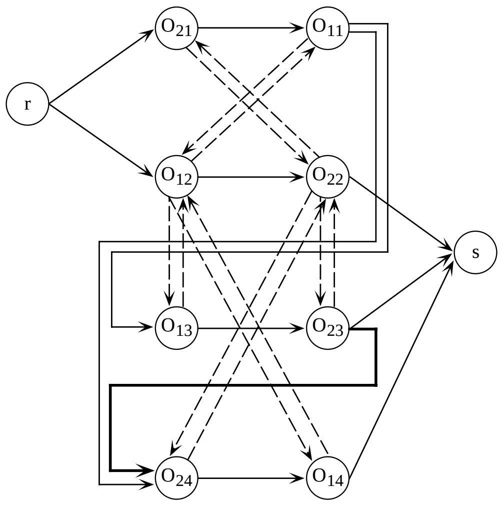


Figure 14: Aggregate disjunctive graph G for OR network N of Fig. 3

In *Case 3*, which is depicted in Fig. 15 and represents a generalization of Case 1, several job sequences emanate from a stochastic node k. We connect the

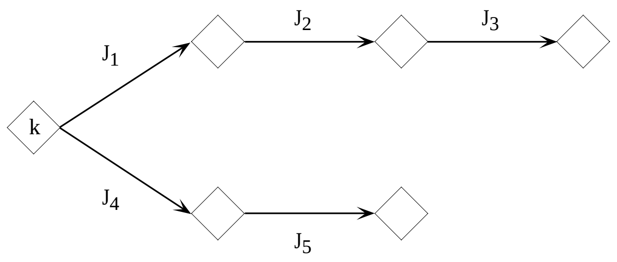


Figure 15: OR network

final job J_3 of job sequence (J_1, J_2, J_3) to the initial job J_4 of job sequence (J_4, J_5) by introducing a stochastic conjunctive arc from the last operation of J_3 to the first operation of J_4. A second possibility is to link jobs J_5 and J_1 in a similar way. In addition, we have to introduce ordinary conjunctive arcs connecting the consecutive jobs J_1, J_2, and J_3 on the one hand and jobs J_4 and J_5 on the other hand. Note that there may be several aggregate disjunctive graphs for one OR network or stochastic job–shop problem, respectively.

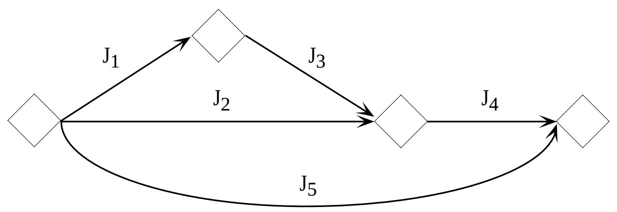


Figure 16: OR network

In general, the introduction of stochastic conjunctive arcs to link jobs beyond a stochastic node is neither easy nor unique. For example, consider the OR network of Fig. 16. A corresponding aggregate disjunctive graph contains three ordinary conjunctive arcs linking the consecutive jobs J_1 and J_3, J_3 and J_4, and J_2 and J_4, respectively. If in addition, we connect final job J_4 of job sequence (J_1, J_3, J_4) to job J_5 and link jobs J_5 and J_2, each by a stochastic conjunctive arc, we obtain a job cycle (J_2, J_4, J_5, J_2) and thus a cycle in the corresponding aggregate disjunctive graph all of whose arcs are conjunctive, which does not make sense. If, instead, we link jobs J_2 and J_1 as well as jobs J_5 and J_2 by a stochastic conjunctive arc each, we obtain an aggregate disjunctive graph which is acyclic with respect to conjunctive arcs.

Schneider (1997) has proposed an algorithm which constructs an *admissible aggregate disjunctive graph* G for each acyclic OR network N, that is, an aggregate disjunctive graph which is acyclic with respect to conjunctive arcs and where for jobs belonging to a subnetwork of N with only stochastic nodes, there is a sequence of conjunctive arcs in G representing all those jobs. The basic idea of that procedure is as follows.

First we number the jobs from $\mathcal{J}$ such that $J_j \prec J_{j'}$ implies $j < j'$. To link two jobs J_j and $J_{j'}$ emanating from one and the same stochastic node, we insert a stochastic conjunctive arc from a job $J_{j''}$ to job $J_{j'}$, where $J_{j''} \succ J_j$, $J_{j''} \not\succ J_{j'}$, and j'' is maximum.

If the OR network N contains only stochastic nodes, there are no disjunctive arcs in any corresponding aggregate disjunctive graph G. There is only one feasible selection D in each graph G, which coincides with the arc set of G. Moreover, we have in analogy to (7)

Theorem 4:

If the OR network contains only stochastic nodes, the length of the only path in any admissible aggregate disjunctive graph from source r to sink s is

$$L(G) = \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_j t_{ij} = \sum_{j=1}^{n} \pi_j t_j = E[C_{\max}(\Phi)] \tag{8}$$

for each aggregate schedule Φ.

Proof:

Since G is admissible, it holds that

$$L(G) = \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_j t_{ij} = \sum_{\omega \in \Omega} P(\omega) \sum_{O_{ij} \in \mathcal{O}_\omega} t_{ij} = \sum_{\omega \in \Omega} P(\omega) C_{\max}(\Phi_\omega) = E[C_{\max}(\Phi)]$$

for any aggregate schedule Φ, where $\mathcal{O}_\omega$ is again the set of operation carried out in network realization ω. $\square$

If the OR network N contains stochastic and deterministic nodes, there is no counterpart to relation (7). Let G be an admissible aggregate disjunctive graph for the job–shop problem with precedence constraints given by N, $G(D)$ be the subgraph of G whose arc set is a feasible selection D, $L[G(D)]$ be the length of a longest path in $G(D)$ from source r to sink s, and $\mathcal{D}$ be again the set of all feasible selections. Then $\min_{D \in \mathcal{D}} L[G(D)]$ can be less or greater than $E[C_{\max}(\Phi^*)]$, where Φ^* is an optimal aggregate schedule (compare Schneider 1997). However, due to the introduction of stochastic conjunctive arcs and the specification of the arc weights in G, the deviation of $L[G(D)]$ with D obtained by the shifting bottleneck heuristic from lower bound LB is relatively small in general as we will see in Section 8.

An admissible aggregate disjunctive graph is a deterministic representation of the stochastic problem $J|acyclOR|E(C_{\max})$. The shifting bottleneck heuristic of Adams et al. (1988) or the modified version of Dauzère–Péres & Lasserre (1994) applied to that graph provides a sequence of operations on each machine.

Observing conditions (a), (b), and (c) of Definition 1, an aggregate schedule can then be determined.

7 A priority–rule–based procedure

The heuristic most frequently used in practice for approximately solving the classic job–shop problem $J||C_{\max}$ is the priority–rule method by Giffler and Thompson (cf. Giffler & Thompson 1960 or Neumann 1996). To adapt this heuristic to our stochastic problem $J|acyclOR|E(C_{\max})$, we replace the stochastic precedence constraints given by the underlying acyclic OR network N with deterministic ones. That is, each node k of N is replaced by a node with deterministic exit and so–called *AND entrance*, which says that node k is activated if all incoming jobs have been terminated. Then each node k corresponds to a node in a classical CPM network and the jobs emanating from k compete with each other for the machines. The stochastic character of the OR network precedence constraints will be taken into consideration by appropriate priority rules.

We briefly review one step of the *Giffler–Thompson algorithm*. Let $\tilde{\mathcal{O}}$ be the set of all schedulable operations, i.e. the set of all operations O with the property that all operations $O' \prec O$ have already been scheduled. Let S_{ij} and $C_{ij} = S_{ij} + t_{ij}$ be the start and completion time, respectively, of operation O_{ij}. Determine $O_{\mu\nu} \in \tilde{\mathcal{O}}$ such that $C_{\mu\nu} = \min_{O_{ij}\in\tilde{\mathcal{O}}} C_{ij}$. Choose an operation $O_{\mu j^*}$ from the conflict set $\mathcal{O}^* := \{O_{\mu j} \in \tilde{\mathcal{O}} \mid S_{\mu j} < C_{\mu\nu}\}$ by some priority rule and delete $O_{\mu j^*}$ from $\tilde{\mathcal{O}}$. Update $\tilde{\mathcal{O}}$ as well as S_{ij} and C_{ij} for $O_{ij} \in \tilde{\mathcal{O}}$.

An aggregate schedule Φ is then given by $\Phi(O_{ij}) := S_{ij}$ for all $O_{ij} \in \mathcal{O}$. The corresponding value of the objective function $E[C_{\max}(\Phi)]$ can be computed by (3) after having determined the network realization schedules Φ_ω for all $\omega \in \Omega$. Note that the computation of $E[C_{\max}(\Phi)]$ cannot be done in polynomial time in contrast to the determination of Φ.

A large number of priority rules for $J||C_{\max}$ have been studied in literature (for example, see Haupt 1989 and Neumann 1996). For $J|acyclOR|E(C_{\max})$ the following priority rules have been discussed by Schneider (1997):

SEPT (Shortest Expected Processing Time): Choose operation $O_{\mu j^*} \in \mathcal{O}^*$ such that

$$\pi_{j^*} t_{\mu j^*} = \min\{\pi_j t_{\mu j} \mid O_{\mu j} \in \mathcal{O}^*\}$$

MEWR (Most Expected Work Remaining): Choose operation $O_{\mu j^*} \in \mathcal{O}^*$ such that

$$\pi_{j^*} \sum_{M_i \in \mathcal{M}_{\mu j^*}} t_{ij^*} + \sum_{J_{j'} \in \mathcal{J}_{j^*}} \pi_{j'} t_{j'} = \min\left\{ \pi_j \sum_{M_i \in \mathcal{M}_{\mu j}} t_{ij} + \sum_{J_{j'} \in \mathcal{J}_j} \pi_{j'} t_{j'} \,\middle|\, O_{\mu j} \in \mathcal{O}^* \right\}$$

where $\mathcal{M}_{ij}$ is the set of machines on which job J_j has to be processed after its processing on M_i including M_i itself and $\mathcal{J}_j$ is the set of jobs $J_{j'}$ which follow job

J_j, i.e. $J_j \prec J_{j'}$. Note that in the MEWR rule we sum up not only the durations of the remaining operations of the currently schedulable job J_j weighted by its execution probability but also of all jobs J' with $J \prec J'$ in order to take the stochastic precedence constraints into account.

FCFS (First Come First Served): Choose operation $O_{\mu j^*} \in \mathcal{O}^*$ such that job J_{j^*} is the first element in the queue of jobs waiting for machine M_μ.

RND (Random): Choose operation $O_{\mu j^*} \in \mathcal{O}^*$ randomly where each $O_{\mu j} \in \mathcal{O}^*$ is equally likely.

In Section 8 we will see that the MEWR rule has turned out to be superior to the remaining three priority rules.

8 Experimental performance analysis

In this section we examine the performance of the two heuristics from Sections 6 and 7. First, we describe the test environment. Next, we analyze the performance of the shifting bottleneck and Giffler–Thompson heuristics. Finally, we compare both heuristics.

8.1 Test environment

Job–shop scheduling problems have been generated for two classes of OR networks. In Class 1, the OR networks have no outtree structure, i.e. nodes may have an indegree of more than one, and the networks generally have a stochastic source. Class 2 contains OR networks with outtree structure and a deterministic source. The latter networks have a larger portion of deterministic nodes and yield scheduling problems of a higher scheduling complexity. A detailed description of the network generator used can be found in Zimmermann (1995). Machine dimensions of $m = 5, 10, 15, 20$ have been chosen. The job dimensions in Class 1 are $n = 5, 10, 20, 30, 50, 75, 100$ and in Class 2 $n = 5, 10, 20, 30$ (due to higher scheduling complexity). All jobs are carried out on all m machines in a randomly determined machine sequence with operation durations uniformly chosen from $\{1, 2, \dots, 10\}$. For each pair (m, n) of the above ranges, 100 instances have been sampled. That is, 2800 problem instances of Class 1 and 1600 problem instances of Class 2 have been investigated.

All tests have been performed on an IBM™–compatible PC with an INTEL™-Pentium Processor 586 with clock speed of 133 MHz and 32 MB RAM. All procedures have been implemented in C^{++} under Microsoft™ Visual C^{++}, Release 4.2.

Table 3: Average relative percentage deviation Δ_G of $L[G(D)]$ from LB

Class 1	5 machines	10 machines	Class 2	5 machines	10 machines
5 jobs	0.2225	0.1358	5 jobs	1.123	1.011
10 jobs	-1.252	-0.9088	10 jobs	3.527	2.001
20 jobs	-1.008	-0.5715			

Table 4: Average percentage deviation Δ_Φ of $E[C_{\max}(\Phi)]$ from LB

Class 1	5 machines	10 machines	Class 2	5 machines	10 machines
5 jobs	0.2225	0.1358	5 jobs	1.618	1.823
10 jobs	1.107	0.7094	10 jobs	6.869	6.311
20 jobs	2.229	1.616			

8.2 Performance analysis of the shifting bottleneck heuristic

The computation time needed for computing an approximate aggregate schedule by the shifting bottleneck heuristic for an instance of a $(10, 10)$ problem of Class 1 is two seconds and for an instance of a $(10, 50)$ problem it is 30 seconds on the average. With increasing number of machines or jobs, the computation times increase remarkedly.

We have determined the deviations of the lower bound LB from both the longest path length $L[G(D)]$ in the (respective subgraph of the) aggregate disjunctive graph G and the expected makespan $E[C_{\max}(\Phi)]$ of the aggregate schedule Φ computed. For different problem sizes (m, n), the relative percentage deviations $\Delta_G := 100 \times (L[G(D)] - LB)/LB$ averaged over the 100 instances are given in Table 3. We stress that the calculation of LB is only possible for small–sized problems, i.e. $m \leq 10$ and $n \leq 20$ in Class 1 or $m \leq 10$ and $n \leq 10$ in Class 2, respectively. The deviation of $L[G(D)]$ from LB is remarkably small. We also see that $L[G(D)]$ can be greater or less than LB. The relative percentage deviations $\Delta_\Phi := 100 \times (E[C_{\max}(\Phi)] - LB)/LB$ each averaged over the 100 instances are shown in Table 4. In summary, we notice a very good performance of the shifting bottleneck heuristic.

8.3 Performance analysis of the Giffler–Thompson procedure

In principle, the computing time for the Giffler–Thompson procedure increases in the number of jobs and machines in the same way as for the shifting bottle-

Table 5: Comparison of priority rules: portion of best and worst results yielded

Classes 1 and 2	MEWR	FCFS	SEPT	RND
portion best	0.670	0.187	0.0778	0.0652
portion worst	0.0229	0.0751	0.521	0.381

Table 6: Average percentage deviation Δ_Φ of $E[C_{\max}(\Phi)]$ from LB

Class 1	5 machines	10 machines	Class 2	5 machines	10 machines
5 jobs	1.891	1.212	5 jobs	2.661	1.083
10 jobs	1.936	1.087	10 jobs	6.336	3.300
20 jobs	2.698	1.792			

neck procedure. However, the Giffler–Thompson heuristic clearly outperforms the shifting bottleneck procedure in absolute computing times.

First, we have compared the four priority rules used. The portions of problem instances where each rule provides the best or worst result for $E[C_{\max}(\Phi)]$, respectively, are shown in Table 5 and visualized in Fig. 17. We see that the MEWR rule is markedly superior to all remaining rules.

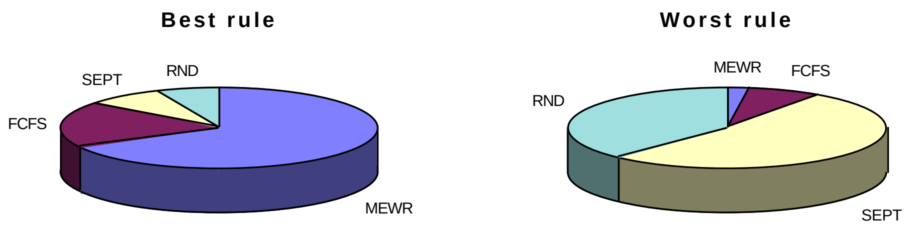


Figure 17: Portion of instances where each rule provides the best and worst result, respectively

Second, we have again determined the deviations of $E[C_{\max}(\Phi)]$ from LB. The relative percentage deviations $\Delta_\Phi := 100 \times (E[C_{\max}(\Phi)] - LB)/LB$ averaged over the 100 instances, where Φ is determined with the MEWR rule, are listed in Table 6. We see that the values of Δ_Φ are again small but mostly larger than for the shifting bottleneck procedure (cf. Table 4).

Table 7: Average percentage deviation $\Delta_{\Phi_{SG}}$ of shifting bottleneck from Giffler–Thompson, Class 1

Class 1	5 machines	10 machines	15 machines	20 machines
5 jobs	-1.552	-1.035	-0.8307	-0.4267
10 jobs	-0.007716	-0.3597	-0.1666	-0.3514
20 jobs	-0.4066	-0.1669	-0.1494	-0.1786
30 jobs	-1.198	1.820	1.715	1.567
50 jobs	0.2418	1.200	1.458	1.248
75 jobs	0.7960	2.878	2.764	3.300
100 jobs	0.9459	3.498	3.664	4.167

Table 8: Average percentage deviation $\Delta_{\Phi_{SG}}$ of shifting bottleneck from Giffler–Thompson, Class 2

Class 2	5 machines	10 machines	15 machines	20 machines
5 jobs	-0.8166	0.7679	0.4106	0.2225
10 jobs	0.6942	2.911	3.061	2.790
20 jobs	3.994	4.829	5.798	7.208
30 jobs	4.986	13.56	13.13	11.34

8.4 Comparison of shifting bottleneck heuristic and Giffler–Thompson procedure

We have computed $\Delta_{\Phi_{SG}} := 100 \times (E[C_{\max}(\Phi_{SBP})] - E[C_{\max}(\Phi_{GT})])/E[C_{\max}(\Phi_{GT})]$, the relative percentage deviations averaged over the 100 instances in order to compare the shifting bottleneck and Giffler–Thompson heuristics. Φ_{SBP} and Φ_{GT} are the aggregate schedules obtained by the shifting bottleneck procedure and the Giffler–Thompson heuristic with MEWR rule, respectively. The values of $\Delta_{\Phi_{SG}}$ are shown in Tables 7 and 8. Visualization is given by Fig. 18.

We see that, on the average, the shifting bottleneck procedure gives very good results for small–sized problem instances, whereas the Giffler–Thompson procedure yields better results for large–sized instances. The worse performance of the shifting bottleneck heuristic is due to the fact that stochastic conjunctive arcs cause additional precedence relations between jobs which actually do not exist.

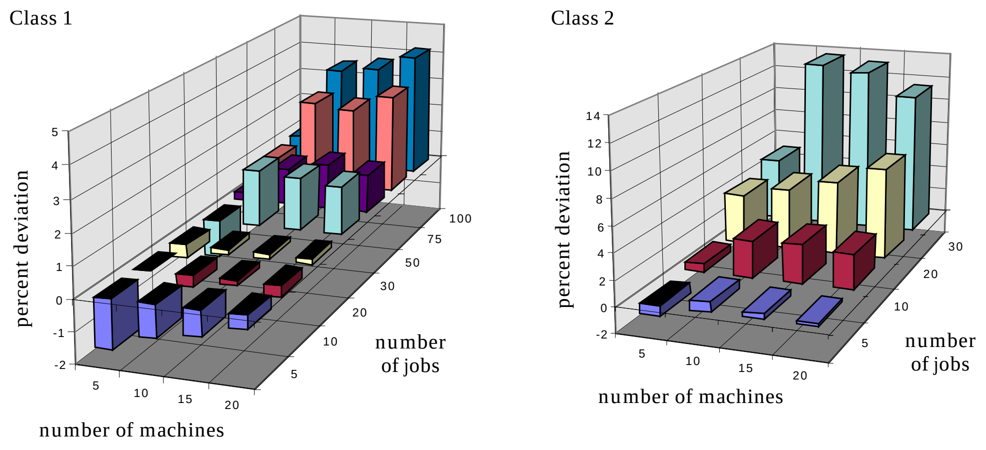


Figure 18: Average percentage deviation $\Delta_{\Phi_{SG}}$ of shifting bottleneck from Giffler–Thompson

9 Supplements

In this section we briefly discuss the flow–shop problem $F|acyclOR|E(C_{\max})$ with stochastic precedence constraints given by an acyclic OR network. After that, we show how to deal with OR networks containing cycles.

9.1 Flow–shop problem $F|acyclOR|E(C_{\max})$

In a flow–shop problem, all jobs have the same processing order through the machines, where we may assume that the machine sequence for each job J_j is $M_1, M_2, \ldots, M_m$. This permits us to aggregate certain operations beyond a stochastic node into so–called actions, which results in a reduction of the computational effort for solving the problem.

The concept of an action has been introduced for single–machine and parallel–machine scheduling with GERT network precedence constraints (cf. Neumann 1990 and Neumann & Zimmermann 1997). To adapt that concept to problem $F|acyclOR|E(C_{\max})$, let k be a stochastic node with outgoing jobs $J_1, \ldots, J_r$ (see Fig. 1). Then each set $\{O_{i1}, \ldots, O_{ir}\}$ $(i = 1, \ldots, m)$ is said to be a *stochastic action* with *beginning node* k and (expected) *duration* $\sum_{\nu=1}^{r} p_\nu t_{i\nu}$. If k is a deterministic node with outgoing jobs $J_1, \ldots, J_r$, then each set $\{O_{i\nu}\}$ $(i = 1, \ldots, m; \nu = 1, \ldots, r)$ is said to be a *deterministic action* with *beginning node* k and *duration* $t_{i\nu}$. In any network realization, an action is performed, i.e. exactly one of its operations is carried out, when its beginning node has been activated.

Let $\mathcal{A}$ be the set of all actions of the underlying OR network. Then, in analogy to Definition 1 of an aggregate schedule, the concept of an *action schedule* $\Upsilon : \mathcal{A} \to I\!R_+$ can be introduced, where in Definition 1 operations and their durations are replaced by actions and their durations. The concept of a *network realization schedule* Υ_ω *belonging to* Υ (cf. Definition 2 where each action is replaced by that of its operations which is carried out in realization ω), the *expected makespan* $E[C_{\max}(\Upsilon)]$ when action schedule Υ is applied (compare (3)), and the concept of an *optimal action schedule* Υ^* (compare(4)) can be introduced as well. Analogously, the shifting bottleneck heuristic and the priority–rule method of Giffler and Thompson (cf. Sections 6 and 7) can be used for computing an approximate action schedule. The following result has been proved by Schneider (1997):

Theorem 5:
For $F|acyclOR|E(C_{\max})$ it holds that

$$E[C_{\max}(\Phi^*)] \leq E[C_{\max}(\Upsilon^*)]$$

that is, aggregate schedules dominate action schedules.

9.2 Cyclic OR networks

For a cyclic OR network N, there are infinitely many network realizations ω. Jobs within a cycle structure $\mathcal{C}$ (i.e. a strong component with at least two nodes) of N may be executed more than once. In this case, we transform N into an acyclic network N' by eliminating cycles of each cycle structure $\mathcal{C}$ from the inside outwards. We begin with a cycle of $\mathcal{C}$ with minimum number of nodes, say $k_0, \ldots, k_{\mu-1}, k_\mu, \ldots, k_\nu, k_0$ in that sequence. Pick any node, say k_μ, let c_μ be a copy of node k_μ, and replace arc $< k_{\mu-1}, k_\mu >$ by arc $< k_{\mu-1}, c_\mu >$. Replace the cycle by the node sequence $k_\mu, \ldots, k_\nu, k_0, \ldots, k_{\mu-1}, c_\mu$. Let all arcs which entered the cycle now lead into k_μ and all arcs which left the cycle now emanate from c_μ. Fig. 19 illustrates such an elimination of a cycle. Proceed analogously with a next cycle (with minimum number of nodes) of the modified cycle structure $\mathcal{C}$. That procedure eventually results in an acyclic network N'. For more details we refer to Schneider (1997).

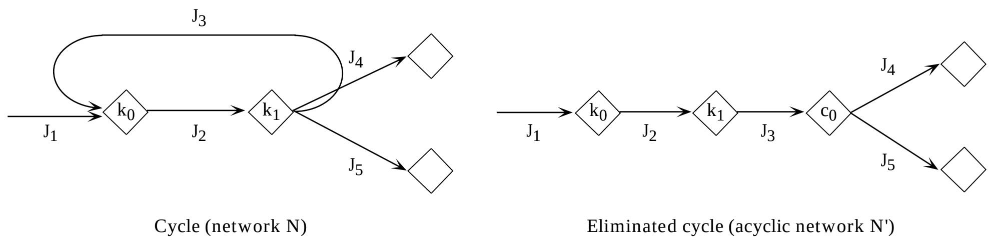


Figure 19: Eliminating a cycle in an OR network

There is no aggregate schedule Φ for the cyclic OR network N (condition (a) of Definition 1 cannot be satisfied). However, we can determine a schedule Φ_ω for each network realization ω of N as follows. At first, we compute the activation probabilities q_k for all nodes k of N and the execution probabilities π_j for all jobs J_j. Next we eliminate the cycle structures in N which results in an acyclic network N' as shown above. For N', we then determine an aggregate schedule Φ' by some heuristic (cf. Sections 6 and 7). From Φ' we derive a network realization schedule Φ_ω for realization ω of N in the following way. Given Φ' a sequence Q_i of operations is specified for each machine M_i. Let $\mathcal{O}_{\omega i} \subseteq \mathcal{O}_\omega$ be the set of operations carried out on M_i in network realization ω. Then every time a machine, say M_i, is freed, the next operation to be carried out on M_i is that one which has the foremost position in Q_i among all operations $O \in \mathcal{O}_{\omega i}$ that are ready to be performed (i.e. all operations $O' \in \mathcal{O}_{\omega i}$ with $O' \prec O$ have already been executed).

10 Conclusions

We have discussed a stochastic job–shop scheduling problem with objective function $E(C_{\max})$ where stochastic precedence constraints are given by so–called OR networks. We have introduced the concept of a deterministic aggregate schedule for that problem. Two heuristics have been presented, which provide good approximate aggregate schedules: the shifting bottleneck procedure and a priority–rule method.

An important area of future research is to develop similar heuristics for stochastic job–shop problems with different objective functions such as $E(L_{\max})$ and $E(T_{\max})$. Moreover, since the heuristics proposed represent only schedule construction procedures, schedule improvement procedures should be developed based upon local search. However, a comparison of different aggregate schedules by computing the corresponding expected makespans cannot be made in polynomial time.

References

[1] J. Adams, E. Balas, and D. Zawack: The shifting bottleneck procedure for job shop scheduling, Management Science 34 (1988), 391–401.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin: *Network Flows*, Prentice Hall, Englewood Cliffs, 1993.

[3] E. Balas, J. K. Lenstra, and A. Vazacopoulos: One machine scheduling with delayed precedence constraints, Management Science 41 (1995), 94–109.

[4] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Węglarz: *Scheduling Computer and Manufacturing Processes*, Springer, Berlin, 1996.

[5] P. Brucker: *Scheduling Algorithms*, Springer, Berlin, 1995.

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

[7] M. Bücker: Time complexity of single machine scheduling with stochastic precedence constraints, ZOR–Mathematical Methods of Operations Research 36 (1992), 211–225.

[8] M. Bücker, K. Neumann, and T. Rubach: Algorithms for single–machine scheduling with stochastic outtree precedence relations to minimize expected weighted flowtime or maximum expected lateness, ZOR–Mathematical Methods of Operations Research 39 (1994), 321–348.

[9] J. Carlier and E. Pinson: An algorithm for solving the job shop problem, Management Science 35 (1989), 164–176.

[10] S. Dauzère–Péres and J.–B. Lasserre: *An Integrated Approach in Production Planning and Scheduling*, Lecture Notes in Economics and Mathematical Systems 411, Springer, Berlin, 1994.

[11] S. E. Elmaghraby: *Activity Networks: Project Planning and Control by Network Models*, John Wiley, New York, 1977.

[12] K. Furmans: A discrete model of a car assembly system, Proceedings of the INRIA/IEEE Conference on Engineering Technologies and Factory Automation, Paris, 1995.

[13] B. Giffler and G. L. Thompson: Algorithms for solving production–scheduling problems, Operations Research 8 (1960), 487–503.

[14] R. Haupt: A survey of priority–rule based scheduling, OR Spektrum 11 (1989), 3–16.

[15] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker: Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1977), 343–362.

[16] K. Neumann: Scheduling of stochastic projects by means of GERT networks, in: Advances in Project Scheduling, R. Slowinski and J. Węglarz, eds., Elsevier, Amsterdam, 1989, 467–496.

[17] K. Neumann: *Stochastic Project Networks*, Lecture Notes in Economics and Mathematical Systems 344, Springer, Berlin, 1990.

[18] K. Neumann: *Produktions– und Operations–Management*, Springer, Berlin, 1996.

[19] K. Neumann and U. Steinhardt: *GERT Networks and the Time–Oriented Evaluation of Projects*, Lecture Notes in Economics and Mathematical Systems 172, Springer, Berlin, 1979.

[20] K. Neumann and J. Zimmermann: Heuristic procedures for parallel–machine scheduling problems with stochastic precedence constraints, Annals of Operations Research, to appear, 1997.

[21] M. Pinedo: *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Englewood Cliffs, 1995.

[22] M. Pinedo and L. Schrage: Stochastic job scheduling: a survey, in: *Deterministic and Stochastic Scheduling*, M. A. H. Dempster, J. K. Lenstra, and A. H. G. Rinnooy Kan, eds., D. Reidel, Dordrecht, 1982, 181–196.

[23] W. G. Schneider: *Job Shop Scheduling with Stochastic Precedence Constraints*, Ph. D. Thesis, Universität Fridericiana zu Karlsruhe, 1997.

[24] V. A. Strusevich: Shop scheduling problems under precedence constraints, Annals of Operations Research 69 (1997), 351–377.

[25] J. Zimmermann: *Mehrmaschinen–Schedulingprobleme mit GERT–Anordnungsbeziehungen*, Ph. D. Thesis, Universität Fridericiana zu Karlsruhe, 1995.