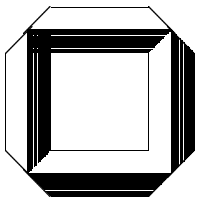
INSTITUT FÜR WIRTSCHAFTSTHEORIE
UND OPERATIONS RESEARCH
UNIVERSITÄT KARLSRUHE

Job Shop Scheduling
with
Stochastic Job Precedence Constraints

Welf G. Schneider

Report WIOR-448

February 1995



TECHNICAL REPORT

University of Karlsruhe

Kaiserstraße 12, D-76128 Karlsruhe, Germany

Abstract

Within the last years, big progress has been made in cracking job shop scheduling problems. However, those models have all been deterministic with respect to job's precedence constraints.

In this paper we reveal a job shop model with stochastic job precedence constraints given by GERT networks. This model appears to be highly interesting for mixed model production scheduling, where product types are customer ordered - and therefore to be produced - with a certain probability. We also show the way to heuristically solve it through a disjunctive graph concept applying shifting bottleneck procedures developed by Adams, Balas, and Zawack (1988) [1]. Our objective hereby is to minimize the expected makespan. Finally, extensions and generalizations are discussed.

Keywords

stochastic job shop scheduling, shifting bottleneck procedure, mixed model production scheduling, GERT networks, heuristics

Contents

Chapter 1

Classical Deterministic Job Shop Scheduling

This chapter is meant to briefly sketch a rough overview upon classical deterministic job shop scheduling. Since notations in scheduling theory are not normed yet, it shall either get us acquainted to the notation we want to stick to throughout this booklet. If you are not at all familiar with (job shop-) scheduling problems, you might want to check with Blazewicz et al. (1994) [6], Dempster et al. (1982) [10], or Domschke et al. (1993) [12] first.

1.1 The static job shop model

We summarize in short what makes a static job shop scheduling model.
We are given

- $n, m \in I\!N; n, m \geq 2$,
- $j = 1, \ldots, n$ jobs J_j,
- $i = 1, \ldots, m$ machines M_i or processors, respectively,
- job J_j consists of m operations $\{o_{ij}\}_{i=1}^{m}$, that are to be processed in a deterministically given order of priority through all of the machines $i = 1, \ldots, m$, i.e., for any job J_j a machine sequence is given,
- d_{ij}: processing time of o_{ij}, $d_{ij} \geq 0$,

and we obey the following capacity restrictions:

- On any machine at most one job can be processed at the same time.
- Any job can be processed on at most one machine at the same time.

Furthermore, we come upon that,

- no preemption is allowed when processing $o_{ij} \quad \forall i, j$, and
- J_j is completed :$\Leftrightarrow$ all o_{ij}, j fixed, are done.

1.2 $(\alpha|\beta|\gamma)$-notation for scheduling problems

More and more often, a triple-notation is used to characterize scheduling problems. In this $(\alpha|\beta|\gamma)$-notation, α stands for the machine environment, β for the job characteristics, and γ abbreviates the objective which is to be minimized. A detailed list of many different possible problem features can be found in Domschke et al. (1993) [12] or in Dempster et al. (1982) [10].

For the purposes of this report, the following explanations should be suffcient:

- $\alpha = J$: We are given a job shop model as described above in section 1.1.
- $\beta = prec$: We are given job precedence constraints, normally by an ordinary directed graph where nodes are jobs. In the case of stochastic scheduling, we want to signal stochastic job precedence constraints, as modeled in chapter 2, by the symbols *GERT* or *acyclEOR*, standing for stochastic networks. Specialities on that are outlined in chapter 2. In addition to constraints, we also flag information about stochastic job durations in the β-field. In general, processing times of job operations are allowed to have any distribution we can imagine. In this case, β shows $D \sim general$. But this could get that much complicated for scheduling theory we can't imagine any more. We therefore start off with constant durations, i.e., job and operation durations are supposed to be degenerated, meaning, not random. In this case, we signal $D \sim deg$.
- γ: We only want to consider regular criteria. Regular criteria are non-decreasing functions of the job's completion times C_j, $j = 1, \ldots, n$. We mainly will consider the objective "expected project makespan", i.e., $E(C_{max})$. We also derive results for $E(L_{max})$, the expected maximal lateness, and $E(T_{max})$, the expected maximal tardiness of jobs. For considering lateness and tardiness, we need to be given job due dates and, optionally, job release dates.

  Clearly, it is the "γ" that makes our model to a problem, since generally, we are able to admissibly schedule the operations in various ways. But normally, schedules differ in the objective and we are interested in finding schedules that minimize γ.

1.3 Schedules

Our interest in this section will be a classification of schedules which is used to limit the set of possibly optimal solutions. Basically, these considerations have been overtaken from Schwindt (1994) [22].

A *feasible* or *admissible* schedule is an assignment of operations to start times which respects the machine sequence conditions for each job, whose job sequences on the machines are finite (i.e. acyclic), and for which the operations of a job do not overlap in time. The set of all feasible schedules is infinite.

Generally, we will not be interested in the set of all optimal schedules. An obvious approach will be to only consider sequences where no operation can be set to start earlier without permutation of a job sequence on a machine or violation of any machine sequence condition. Each schedule can be transformed into such an equivalent *semi-active* plan by left-shifting the bars of the corresponding Gantt chart as far as possible (local left-shift). In an *active* schedule, it is not possible to bring forward the start of any operation without delaying the start of another operation, even by switching the job sequence (global left-shift). It can be shown both that, a semi-active schedule cannot be better with respect to any regular criterion than the corresponding active schedule and that, the set of all active schedules always contains at least one optimal schedule. In a *non-delay* schedule, operations are dispatched as soon as a machine becomes available for processing the next job. The set of all non-delay schedules forms a subset of the set of all active schedules. However, non-delay schedules do not dominate active schedules in the same manner as active plans dominate semi-actives. The set of non-delay schedules does not necessarily contain an optimal schedule. Figure 1.1 summarizes the relationship between the different classes of schedules.

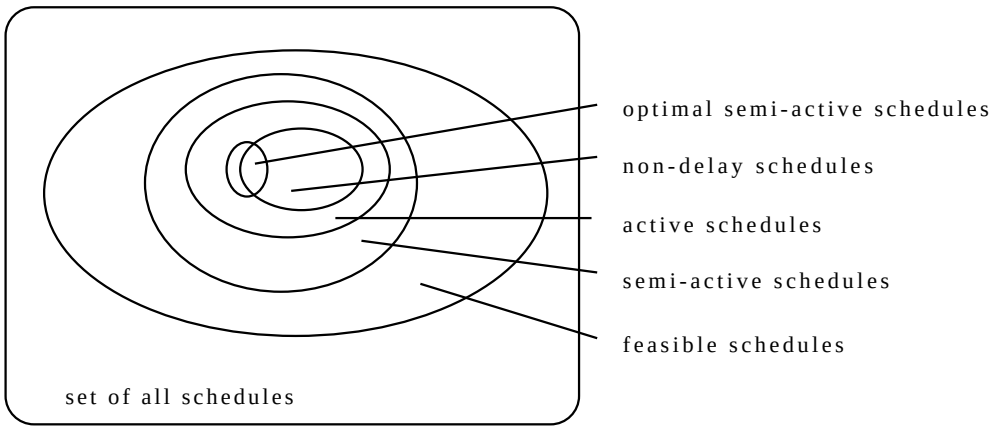


Figure 1.1: Venn diagram for the classification of schedules in the case of regular criteria

1.4 The job shop's disjunctive graph representation

Followingly, we will illustrate the disjunctive graph concept. For profound definitions, compare Domschke et al. (1993) [12] or Schwindt (1993) [23].

The job shop scheduling problem $(J||C_{max})$ can be represented through a disjunctive graph, which is a specific directed graph. In this formulation, we define for any operation o_{ij} a node o_{ij} [1] and, for any pair of consecutive operations an arc between the corresponding nodes. Those arcs are called *conjunctive arcs*. Hence, conjunctive arcs represent the job's operation sequence regulations. Between any pair of operations that are to be proceeded through the same machine, we are to have a pair of opposing arcs. Those arcs represent machine competition between operations. I.e., the operations might want to be processed on the same machine at the same time, which we know is forbidden due to our statically given machine capacitating restriction. We call those arcs *disjunctive arcs*. Clearly then, all operations o_{ij}, i fixed, form a clique for all machines $i = 1, \ldots, n$. In addition, we introduce an artificial source q with artificial arcs to any job's starting operation. Analogously, we shall have an artificial sink s with arcs leading from any job's terminal operation into it.

Now, the admissible sequencing in our job shop problem is equivalent to the following:

For any pair of opposing disjunctive arcs, we need to select exactly one (and neglect the other one) such that the resulting directed graph is acyclic [2]. We call the set of selected disjunctive arcs leading to an acyclic directed graph a *selection*. With a selection we clarify the messy operation's machine competition situation by giving that operation precedence to another one where the selected disjunctive arc emanates from. Corresponding to an acyclic directed graph, there is an admissible, by local left-shift semi-active schedule obeying all precedence constraints - the order of priority between the job's operations through the constructed conjunctive arcs as well as machine capacity restrictions through the acyclic selection of disjunctive arcs.

If we weight every arc in the disjunctive graph by the processing time d_{ij} of operation o_{ij} being positively incident to the arc, and give the artificial source q the processing time 0, then, for a chosen selection, the length of a longest path from q to s is exactly equal to the makespan C_{max} of a corresponding semi-active schedule. This schedule is retrieved by the following: the starting time of

[1] Since normally it will be clear whether we mean operation o_{ij} or the corresponding node, we allow ourselves not to distinguish in notation.

[2] The name "disjunctive arc" comes from this "binary" selection of opposing arcs.

operation o_{ij} is exactly equal to the length of a longest path from q to node o_{ij}.

Minimizing C_{max}, we need to find a selection that minimizes the length of a longest path from q to s.

With this equivalence, we are able to exactly solve $(J||C_{max})$. Unfortunately, there generally are very many possible selections so that we tend to apply good heuristics in order to get a near-optimal solution in a reasonable amount of time. A "quite good" method exploiting equivalence to the disjunctive graph representation is that of Adams, Balas, and Zawack (1988) [1]. We will take advantage of it in our stochastic job shop problem later.

1.5 Ways to solve the job shop problem exactly

There is a representation of the job shop problem, origined by Manne in 1960, which is a mixed binary linear programming formulation. Variables are the real-valued starting times of operations and the occuring waiting times for operations, as well as binary variables regulating precedence constraints of operations due to pre-given job's operation sequences and machine capacity restrictions. For an exact model description we refer to Manne (1960) [15].

In the model of Manne, there occur $2 \cdot n \cdot m + \frac{n(n-1)m}{2}$ variables, where $\frac{n(n-1)m}{2}$ are binary. The number of restrictions is $n(m-1) + n(n-1)m + 2 \cdot n \cdot m$.

There have been developed improved integer formulations of the static job shop problem, too. A survey of those can be found in Seelbach (1975) [24]. Depending on the number of jobs, the number of machines, or on a considered time horizon, different models show advantages with respect to the number of binary variables.

However, even for moderate job and machine dimensions n or m, respectively, we have very many variables and, it gets quite difficult to solve practical problems therewith. Mixed integer formulations of job shop problems rather have explanatory purpose than practical applicability.

The branch-and-bound approach of Seelbach (1975) [24] is an effective way to solve the job shop problem exactly. It uses facility-based and job-based bounds. Powerful branch-and-bound procedures have been developed by Carlier & Pinson (1989) [9], Applegate & Cook (1991) [3], and Brucker et al. (1994) [7].

Recently, branch-and-cut techniques have been developed to solve the job shop problem, too. But those techniques couldn't show souvereign computing time advantages over branch-and-bound techniques yet. It is not trivial to describe a job shop polyhedron.

1.6 Complexity reflections

As a consequence of the above mentioned we get that, cracking job shop problems through exact mixed binary formulations via branch-and-bound or branch-and-cut techniques can cause exponentially increasing computing time in the number of jobs, or in the number of machines either. In addition, even for decent-sized problems with, e.g., 10 jobs and 5 machines we need plenty of time. For not much bigger problems we quickly get to the limits of our patience. Therefore, we tend to apply "good" heuristics to near-optimally solve the job shop problem for practically relevant problem sizes in an acceptable amount of time. "Good" shall mean: not too far optimum-off.

1.7 Ways to solve the job shop problem heuristically

There are many ways to solve the job shop problem by heuristics. We first mention the Giffler-Thompson approach. Next, we look at the shifting bottleneck procedure of Adams, Balas, and Zawack (1988) [1] which, we will apply to our stochastic job shop model. Finally, we don't forget to mention other heuristic types developed to solve job shop problems.

1.7.1 Giffler-Thompson's heuristic

A well-known approach for the job shop problem with regular objective functions is the algorithm of Giffler and Thompson. For details, check Giffler & Thompson (1960) [13], Neumann & Morlock (1993) [20], Schwindt (1994) [22], or Schwindt (1993) [23]. If we are only interested in the generation of a single schedule, we will employ priority rules in order to resolve dispatching conflicts. This technique allows appropriate adaptation of the basic heuristic to a specific regular objective function.

1.7.2 The shifting bottleneck procedure (SBP) of Adams, Balas, and Zawack

During the last years, several heuristics have been proposed for the job shop problem ($J||C_{max}$), which are based on the concept of disjunctive graphs. Compare section 1.4. Especially, the shifting bottleneck procedure of Adams, Balas, and Zawack (1988) [1] has found its way into job shop literature (cf. Blazewicz

et al. (1994) [6], Dauzere & Lasserre (1994) [11], Domschke et al. (1993) [12], and Morton & Pentico (1993) [16]).

In the following, we briefly sketch the basic idea of the shifting bottleneck algorithm. As we know, the set of vertices of a disjunctive graph contains the operations of the corresponding job shop problem, an artificial source, as well as an artificial sink. The given machine sequences of the jobs are mapped by conjunctive arcs whereas disjunctive arcs represent the job sequences on the machines, which are to be determined. In the course of the algorithm, the disjunctive graph is transformed by fixing or rejecting disjunctive arcs in an acyclic conjunctive directed graph. The equivalent semi-active schedule can then be generated from the latter directed graph. The procedure identifies that machine as bottleneck among the facilities not yet scheduled for which the algorithm of Carlier (1982) [8] yields the largest makespan. Carlier's algorithm is a very efficient branch-and-bound method for minimizing makespan for the following *single* machine constitution: Operations, which in this case are complete jobs, are to be sequenced, where heads and tails are given for each job or operation, respectively. Heads can be interpretated as pre-running times, tails as after-running times. Every time a new machine has been scheduled, the sequences previously determined are reoptimized. Besides this straight forward procedure (SBP), Adams, Balas and Zawack have implemented a version which applies the procedure on the nodes of a partial search tree (SBP2). For a more detailed presentation of the disjunctive graph modeling technique and the shifting bottleneck procedure we refer to Dauzere & Lasserre (1994) [11] or Domschke et al. (1993) [12].

1.7.3 Other heuristics

There have many other heuristics been developed for considering job shop scheduling problems. Those heuristics are of various origins such as

- simulated annealing or
- tabu search.

We don't want to stress on them since our way to solve the job shop problem with stochastic job precedence constraints is based on the SBP of Adams, Balas, and Zawack. For further research ideas we refer to the discussion in chapter 4.

1.8 Regular precedence constraints

If a job i necessarily, e.g. for technological reasons, has to be performed before a job j, we are given a precedence constraint "i before j". Within the context of job shop scheduling this means that, all processings of any operation of job i need

to be finished before starting the first operation of job j. In particular, there is no machine competition at any time between operations of job i and operations of job j.

Those precedence constraints normally are given by an appropriate directed graph, e.g., like shown in figure 1.2 for the precedence constraint "i before j".

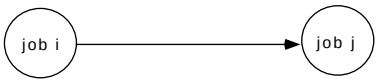


Figure 1.2: Regular precedence constraint

In the disjunctive graph representation of the job shop problem it is very easy to represent those precedence constraints "i before j". We just have to insert conjunctive arcs from the last operation of job i to the first operation of job j. Since there never is machine competition between jobs that have to be performed sequentially, we omit the corresponding disjunctive arcs and thereby avoid "killer"- cycles in the conjunctive graph for SBP. We're lucky on that!

Chapter 2

The Job Shop Model with Stochastic Job Precedence Constraints - $(J|GERT, D \sim deg|)$

Within chapter 2, we are guided into the theory of GERT networks first. Next, we are shown why this concept of stochastic networks is good for modeling stochastic precedence constraints. Finally here, we motivate the sense of stochastic job precedence constraints by considering mixed model production.

2.1 The concept of GERT networks

Modeling stochastic precedence constraints we want to stick to the concept of GERT [1] networks. GERT networks are activity-on-arcs project networks where stochastic elements can be described. Stochastics is meant with respect to project evolution as well as activity durations. The concept of GERT networks in general is quite powerful since aspects as feedback can either be taken into account. Feedback is expressed by cycle structures. For a detailed description of GERT networks and treatment of structural problems we refer to Neumann (1990) [18] and Neumann & Steinhardt (1979) [21].

For our purposes we do not need all the skills of GERT theory but only some extracts. In the following subsections we take a glance at the essentials.

Anyway, we keep in mind that with the forthcoming restriction to acyclic networks only containing special node types there is space for further extensions of our stochastic job shop model. At this stage, we are happy to have the corresponding discussion far ahead in chapter 4.

[1] GERT stands for **G**raphical **E**valuation and **R**eview **T**echnique.

2.1.1 Node types in GERT networks

We distinguish six different types of nodes, each composed by an entrance side and an exit side:

Entrance sides:

(1) AND-entrance ("and entrance"): Every activity leading into this node is to be terminated, then the node will be activated [2]. The symbol for the AND-entrance is shown in figure 2.1.

Figure 2.1: GERT node entrance type AND

(2) IOR-entrance ("inclusive-or entrance"): This node is activated at that time, when the first incoming activity is terminated. The node thereafter will never be activated again. We symbolize IOR-entrances as shown in figure 2.2.

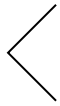


Figure 2.2: GERT node entrance type IOR

(3) EOR-entrance ("exclusive-or entrance"): This node will be activated whenever an incoming activity is terminated. The symbol for the EOR-entrance is mirrored in figure 2.3.

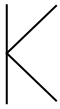


Figure 2.3: GERT node entrance type EOR

[2] We say a node is activated, if and only if a project execution realizes the state corresponding to that node.

Exit sides:

(1) DET-exit ("deterministic exit"): If this node is activated, all outgoing activities will be started. The symbol for the DET-exit is given in figure 2.4.

Figure 2.4: GERT node exit type DET

(2) ST-exit ("stochastic exit"): If this node is activated, exactly one of the outgoing activities will be started. The choice is by random. We hereby stick to the ST-exit symbol sketched in figure 2.5.

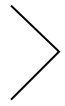


Figure 2.5: GERT node exit type ST

Conventions

- If there is at most one incoming activity, we let the entrance side be EOR.
- If there is at most one outgoing activity, we let the exit side be ST.

2.1.2 Stochastics in GERT networks and regularity assumptions

Whenever a project execution activates a node i with ST-exit, we are to choose exactly one outgoing activity. Thus, for any successor node j of node i the probability, that activity $< i, j >$ is chosen, given that node i is activated, is defined by p_{ij}. Of course, we require $\sum_{j \in S(i)} p_{ij} = 1$ for any ST-node i, where $S(i)$ denotes the set of all successors of i.

For the random duration of activity $< i, j >$, we need to be given a distribution function of the activity duration D_{ij} of activity $< i, j >$, given that node i is activated. It shall be denoted by F_{ij}.

Remarks

- We will assume the activity duration D_{ij} to have the same distribution regardless how often $< i, j >$ has already been activated. Especially, analysis of cycle structures is hereby made easier.
- Analogously, we define F_{ij} for DET-nodes. We are further not disturbed to speak of $p_{ij} = 1 \ \forall j \in S(i)$, if node i is a DET-node.
- Only positive durations are allowed.
- Expectation of any duration is supposed to be finite.
- We only want to consider projects having exactly one source. Generally, several sources are allowed and a project execution may start in a selection of sources. But in this case, we would have to cope with annoying structural problems. Treatment of this can be found in Neumann (1990) [18].

Definition

A *GERT network* is a network unifying the above properties.

Some further regularity assumptions

(A1) Markov property: The development of a project realization only depends on the current project state and not on former project behaviour. Especially,

- p_{ij} and F_{ij} do not depend on the number of previous activations of activity $< i, j >$ conditioned on the event, that node i is activated.
- For all activities $< i, j >$, the D_{ij}'s are independent.

(A2) Every node within a cycle structure is to be STEOR.
(A3) Every node not in a cycle structure having at least two possibly active predecessors is either to be AND or IOR.
(A4) In any project realization, a cycle structure is at most activated through one arc leading into it.

Properties (A1) to (A4) are nice to have and offer a good basis to work with. Therefore, we define the following.

Definition

A GERT network fulfilling (A1) to (A4) is called *admissible*.

Remarks

- Being sloppy, we always mean admissible GERT networks when speaking of GERT networks.
- For simplicity, we concentrate our reflections on GERT networks only having EOR typed nodes first. Those networks are called *acyclic EOR networks*. We abbreviate by "acyclEOR".

2.1.3 Acyclic EOR networks and node activation probabilities

For acyclic EOR networks, there is a very convenient way to calculate the probability, that a certain node will, in finite time, be activated in a project execution. We need those node activation probabilities, denoted by q_i, to state the execution probabilities of the following activities or, as we will see, jobs, respectively. In acyclic EOR networks namely, coincidence to appropriately overlapping Markov renewal processes has been found. Exploiting results from renewal theory, we can easily calculate the q_i's by the following linear system of equations, where, w.l.o.g, the node set shall be $V = \{1, \ldots, n\}$ and $i = 1$ is sentenced to be the only source:

$$q_j = \sum_{k=1}^{n} p_{kj} q_k; (j = 2, \ldots, n), q_1 := 1 \tag{2.1}$$

If you are more interested in renewal theory in general, you may check references Grimmet & Stirzaker (1982) [14], Barlow & Proschan (1975) [5], or, for experts, Alsmeyer (1991) [2].
Treatises of coincidence of acyclic EOR networks and Markov renewal processes can be found in Neumann (1986) [19] and Neumann & Steinhardt (1979) [21].

2.2 Job precedence constraints given by GERT networks

We model stochastic job precedence constraints by GERT networks, where any job with all its operations corresponds to uniquely one activity and hence to exactly one arc of the GERT network. A node partially reflects a project state. Activation probabilities q_i of node i indicate the likelihood that a following job will be executed.

We flag "GERT" in the β-field of scheduling theory's triple notation. See chapter 1.

Whenever we want to take advantage of the fine properties of acyclic EOR networks, where especially the q_i's are easily obtained, we signal "acyclEOR" in

our β-field.

2.3 Job durations

Job- and operation durations in our model are supposed to be constant. I.e., they are not random variables.

In short: $D_{ij} \sim deg(d_{ij})$ for some constants $d_{ij} \geq 0$.

More short: $D \sim deg$.

Being honest, we show this in our β-field, too.

Remark

Letting $D_{ij} \sim general$ would cause heavy trouble in solving the scheduling problem. The reflections we undergo in chapter 3 are not straight forwardly adaptable.

2.4 An example in mixed model production

We will now consider an example that shall illustrate the concept of GERT networks as well as the before defined concept of stochastic precedence constraints. Furthermore, it shall make us conscious that all our models are useful in mixed model production and operations management.

For basic theory upon production and operations management, we point out Askin & Standridge (1993) [4] as well as Neumann (1992) [17].

For simplicity and for later use, our example is of "$\beta = acyclEOR, D \sim deg$"-type.

Here it is:

We consider *fictitious* automotive propulsion manufacturing for a certain kind of Swabian quality car, where the customer is assumed to have the choice to individually order his personal pattern of propulsion and, the car thereafter is to be produced.

The customer shall be allowed to choose one of the following engines into his car: 4-cyl. engine, 6-cyl. engine, turbo-charged 6-cyl. engine, or turbo-charged+intercooled 6-cyl. engine. Independent from the selected engine, he is to choose either manual 5-speed transmission or automatic 4-speed transmission. Finally, he is allowed to wish an automatic lock differential instead of a regular one, independent from engine- and transmission preferences.

For those order wishes, we are given corresponding order probabilities, e.g., we know that 60 % of the customers want to have the economic 4-cyl. engine.

Now, due to technological production laws we assume that, transmission, engine, and shafts with differential and rear axle can be build in independently [3]. Other technological production restrictions can be read out of the GERT network given in figure 2.6.

Having another look at figure 2.6, we also see the sense of DET-nodes in the network, meaning, that all outgoing branches of jobs can be performed independently and, probably, at the same time. A ST-node indicates that, only one outgoing job has to be made. E.g., we only want to build in one 4-cyl. engine. The corresponding order probability is our given $0,6$. Beware that, the order probability of a 6-cyl.-turbo engine is $0,4 \cdot 0,8$. Since, in our example, a turbo charger is mounted at a regular 6-cyl. engine, the probability that a turbo charger has to be mounted conditioned on the event, that a 6-cyl. engine already has been built in, is $0,8$. This is the number we assign to the corresponding activity in the EOR network. But, the overall likelihood to execute J_{10} is $0,4 \cdot 0,8$.

2.5 Remarks to the model

- The order of priority within the operations o_{ij}, j fixed, is supposed to be deterministic. This ordering being stochastic either, the situation would get messy. Why so?
  - First, this would contradict the general model of job shop manufacturing, where a job's operation sequence is reasoned by technological circumstances and those are not stochastic. E.g., if a forest ranger wants to sell a tree-log, he first has to cut a tree and thereafter saw off its branches. There is no simple and normal way around that.
  - Secondly, o_{ij}-precedence relations given by GERT networks do cause trouble since, in this case, it is not for sure, if a job is being completed or not. A partly completed job just makes no sense in manufacturing.
- D_{ij} are not random, therefore $D \sim deg$. D_{ij} being real random variables would not directly work in the followingly suggested solution procedure.

[3] For car fetishists we again stress, that this is all fictitious. The author is aware of the naked fact, that real propulsion manufacturing looks different. The example was constructed to illustrate the above mentioned.

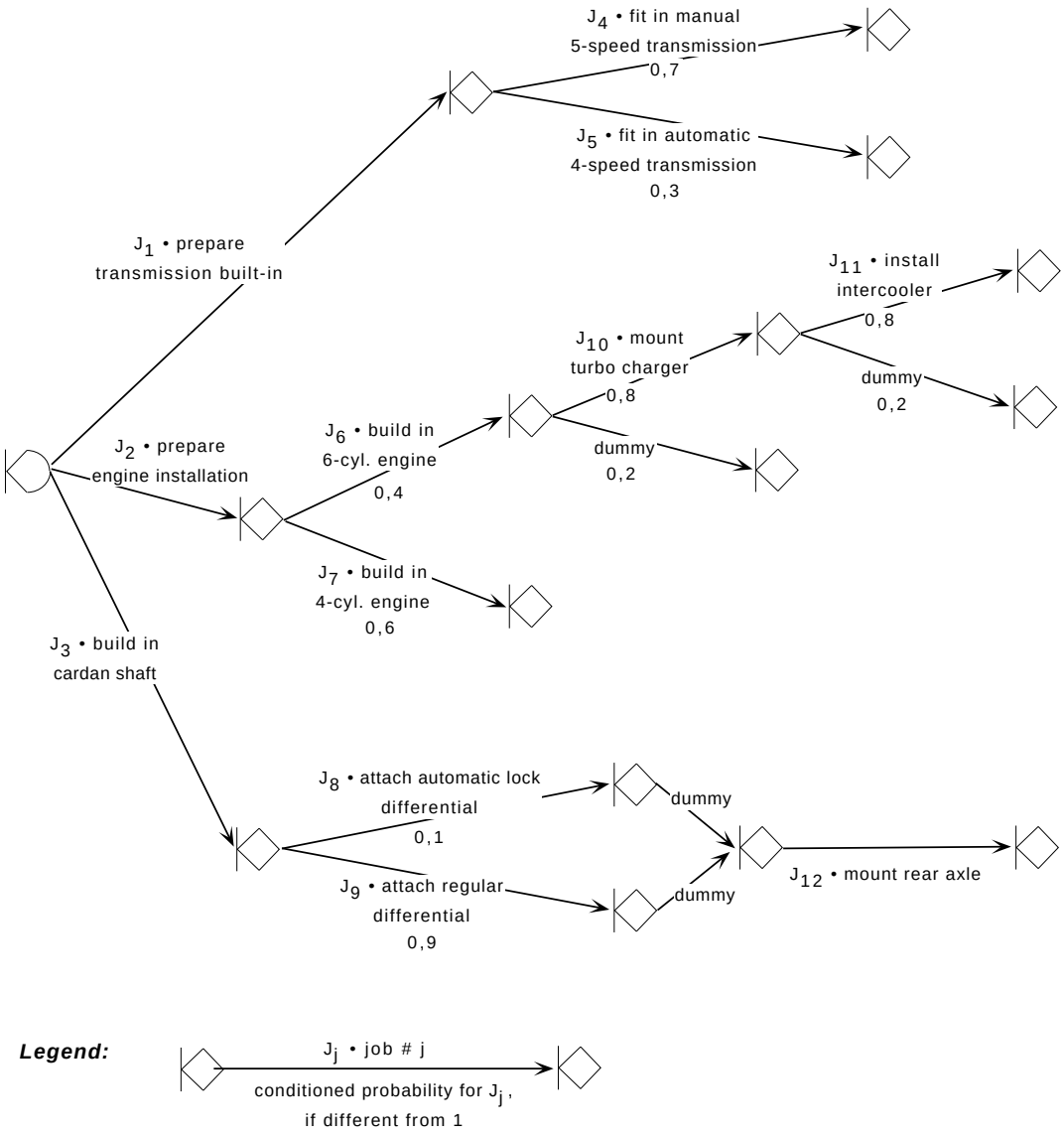


Figure 2.6: Our example for mixed model automotive propulsion

Chapter 3

A Way to Solve
$(J|acyclEOR, D \sim deg|EC_{max})$

This chapter shows us first how to get a proper disjunctive graph representation for our new stochastic job shop model. Secondly, we will see that the shifting bottleneck procedure of Adams, Balas, and Zawack decently works on the newly developed disjunctive graph and that, it gives us a semi-active schedule for all existing job operations. Finally, we learn how the obtained solution is to be understood.

3.1 Extracting a stochastic disjunctive graph out of the stochastic job precedence constraints

In this section we explain how we can get a disjunctive graph where given stochastic job precedence constraints can be modeled into. We again stress that we restrict ourselves to $\beta = acyclEOR, D \sim deg$ first. Generalizations to this will be discussed in chapter 4.

Let us be given an acyclic EOR network representing stochastic job precedence constraints as explained in chapter 2. We first read regular precedence constraints out of it. Therefore, we just ignore the individual node types for a while, and insert regular job's precedence constraints into the disjunctive job shop graph by inserting an arc from the final operation of any job to any start operation of its successor jobs. We remark that, due to the longest path concept, it is sufficient to insert only one arc to the immediate successor. Whenever we have regular job precedence constraints, we neglect the corresponding disjunctive arcs between the jobs since we know that those consecutive jobs never compete for the same

machine at any time. This is like we had in chapter 1 and does not surprise us so far.

Let's think of stochastics. We first will get a feeling for it by examining an example sequence. Thereafter, we formulate general regulations for extracting disjunctive graphs.

We open the example sequence by

Example 1:

Suppose, we are given $n = m = 2$ and precedence constraints as shown in figure 3.1. Those precedence constraints are deterministic. They mean that, job 1 and job 2 can be processed independently at the same time. This is like static job shop scheduling. The corresponding disjunctive graph is given in figure 3.2, where we can also read out the job's operation sequences for our problem.

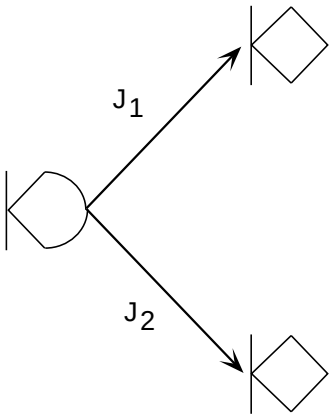


Figure 3.1: Stochastic job precedence constraints to example 1

Example 2:

Suppose, we are given a real stochastic situation with $n = m = 2$ as shown in figure 3.3. Example 2 is meant to illustrate stochastic components in our disjunctive graph. As a scheduling problem alone it is stupid, of course, since there is nothing to schedule.

We first recognize that either job 1 or job 2 is to be processed onto the machines. Job 1 is to be performed with probability r_1 and, the execution probability for job 2 is $1 - r_1$. As a consequence we notice that, there will never be machine competition between job 1 and job 2. Thus, there is no disjunction in the corresponding disjunctive graph, meaning, we don't insert disjunctive arcs between jobs 1 and 2. Next, we are concerned of taking expected processing times into our graph concept such that we get close to the objective $E(C_{max})$ as a longest

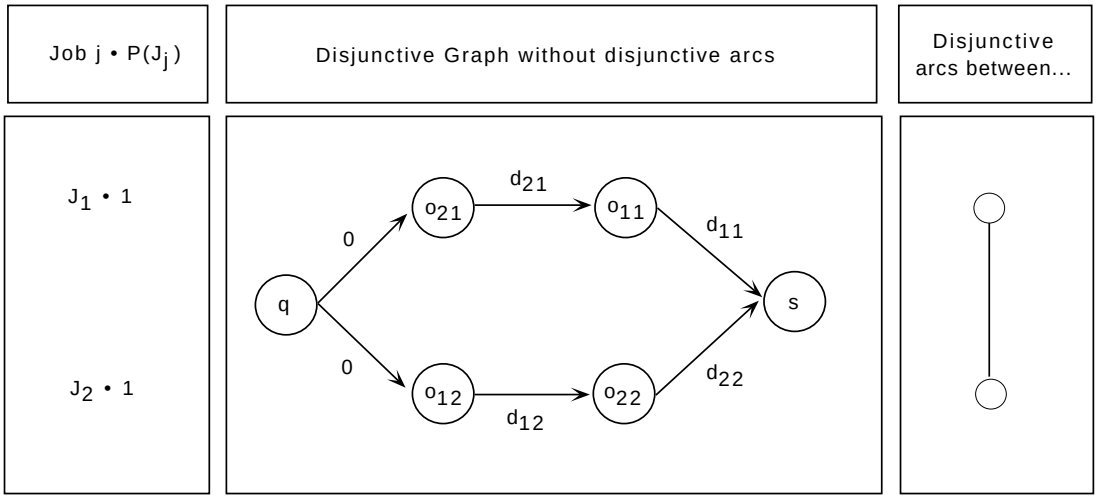


Figure 3.2: Disjunctive graph to example 1

path from q to s. Therefore, we weight the processing time d_{ij} of operation o_{ij} with its execution probability, which is exactly the probability, that we have to perform job j, which we will denote by $P(J_j)$. Remember from chapter 2, that in the case of $acyclEOR$ networks it is easy to get the node activation probabilities. For taking expected durations in a longest path from q to s, we apply a trick to consider simulated processing of both jobs, since both can influence $E(C_{max})$: We start off performing job 1 onto the machines taking the expected operation durations and thereafter make the process switch over to job 2 by an artificial (conjunctive) *stochastic arc*, going from the last operation of job 1 to the first operation of job 2. Of course, operation times of job 2 are weighted with its execution probabilities either, here, $1 - r_1$. Respecting our start and end conditions with an arrow from q to the first operation of job 1 and from the last operation of job 2 to s, respectively, we get the disjunctive graph as shown in figure 3.4 with the therein given job's operation sequences.
By symmetry of the precedence constraint it is possible, of course, to let the project start off with job 2, turn into job 1 and finish off with job 1. In this case however, we need to switch the stochastic conjunctive arc from the final operation of job 2 to the start operation of job 1. Clearly, we have to adapt the starting arc from q to job 2 and the terminating arc from the last operation of job 1 into s. From the mathematical point of taking expectation through the artificial stochastic arc, this makes no difference at all, and therefore, we tend to deal stochastic components in a lexicographical order.
In this example it is evident that the longest way in the new disjunctive graph is exactly the same than $E(C_{max})$. This is true for exclusively stochastic precedence constraints where we have no disjunctive arcs.

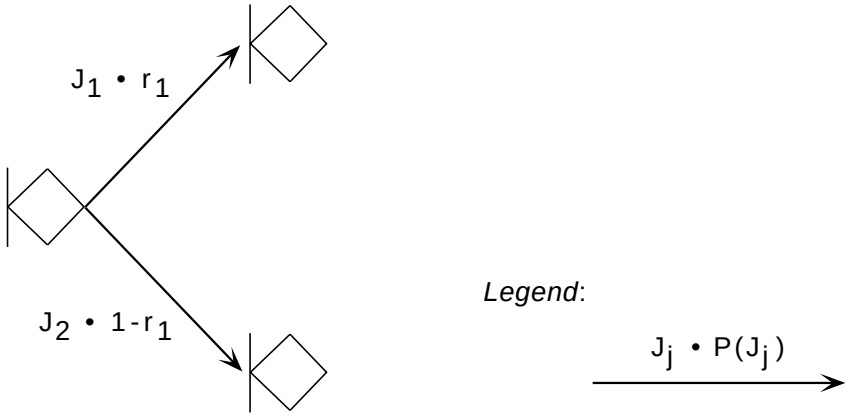


Figure 3.3: Stochastic job precedence constraints to example 2

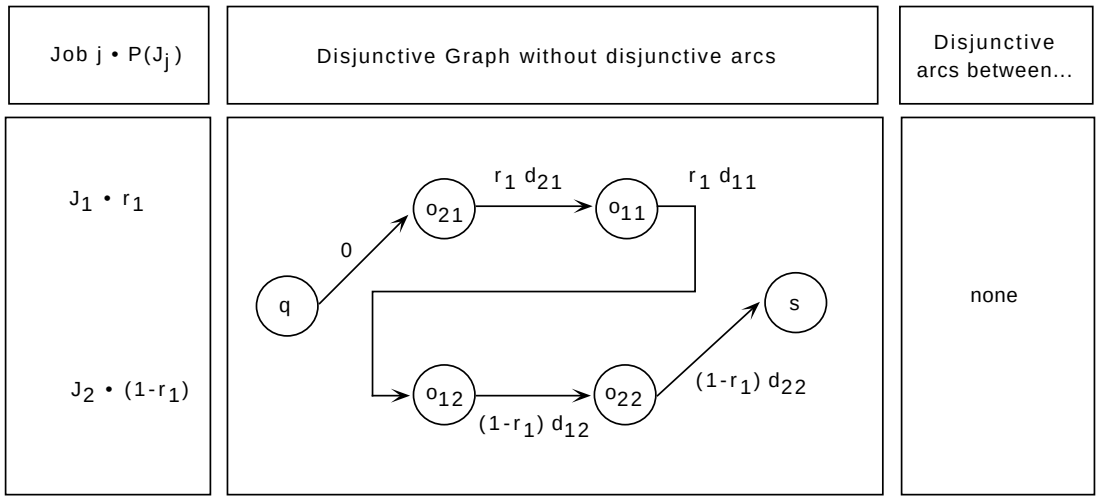


Figure 3.4: Disjunctive graph to example 2

Example 3:

This example is meant to show combination of deterministic and stochastic components.

We take a look at figure 3.5, where we are shown four jobs and their precedence relations. Job 1 and job 2 can be processed independently and therefore probably at the same time. Having done job 1, we either need to do job 3 (with probability r_1) or job 4 (with probability $1 - r_1$).

We construct a disjunctive graph as follows: Since the project can start off with either job 1 or job 2, we set starting arcs from q to the start operations of jobs 1 and 2. The project can either end with finishing job 2 or with job 3 or 4, respectively. We draw finishing arcs from end operations of the jobs 2, 3, and 4 to the sink s. Since job 1 is regular predecessor of jobs 3 and 4, we take care of that by regular precedence arcs from job 1 to jobs 3 and 4. Job 3 and 4 being in stochastic order, we insert an artificial stochastic arc from the final operation of job 3 to the start operation of job 4. We also have to weight the operation processing times of jobs 3 and 4 with the corresponding execution probabilities r_1 and $1 - r_1$, respectively. There clearly is machine competition between jobs 1 and 2. But also, job 3 and 2 could be done at the same time, namely, when job 1 is already finished and we are further supposed to perform job 3 and still need to do job 2, at least in parts. Analogous situation is true for job 4 and 2. We therefore need to express this through corresponding disjunctive arcs. These have been the essentials for constructing the disjunctive graph to example 3. We further remark that, a longest path from q to s will never run through the final operation of job 3, since, there are longer components leading through the stochastic arcs into job 4 and then heading out to s. Analogy is true for the regular precedence arc from job 1 to job 4. We can neglect those redundant arcs. Assuming to have two machines available for production, we are now able to fully understand figure 3.6 with the therein given job's operation sequences. However, we mention that, for a given selection a longest path from q to s does not always give $E(C_{max})$ anymore. At least, it is an approximation to it.

In the above example sequence, we got to know the basic ideas to establish a *stochastic disjunctive graph* out of given stochastic job precedence constraints. Generally, we need to take care a bit more. For instance, when having more than two emanating jobs out of a stochastic node in the given network, we have to sequence the stochastic arcs throughout all jobs. Or, when having consecutive components to stochastic network components, we need to build our stochastic graph iteratively for any stochastic sub-component, starting from the most interlocked one. Moreover, we have to multiply any operation processing time by the operation's execution probability, i.e., $d_{ij}^{new} := d_{ij} \cdot P(J_j)$, where $P(J_j)$ is the execution probability of job j. This is even true in deterministic components. In the worst, we multiply by 1. Taking care of real machine competition, we need to get disjunctive arcs into our graph. For acyclic EOR networks, we see that there is machine competition from any job of any branch emanating from any DET node to any job of another branch of the same node. This can happen interlockedly, too. Moreover, we can have m machines and surely need to insert the corresponding disjunctive arcs for all of the m machines.

We now will show a disjunctive graph to our automotive mixed model production example of chapter 2 as a more complex example. We will assume production to take place on two machines. We don't stress on job's operation sequences and

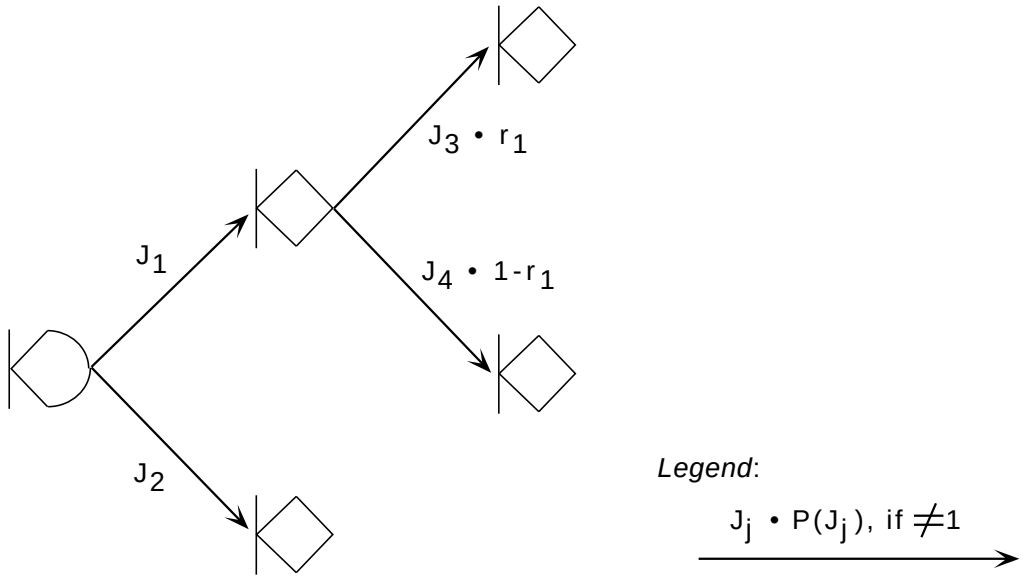


Figure 3.5: Stochastic job precedence constraints to example 3

leave the corresponding labeling in our stochastic disjunctive graph open. The stochastic disjunctive graph is given in figure 3.7.

3.2 Applying shifting bottleneck procedure

The above made transformation from our stochastic job shop problem is one into a regular disjunctive graph. Therefore, all methods based on a disjunctive graph representation are directly applicable. The objective turns from C_{max} into an estimate for $E(C_{max})$ for the original problem. Of course, it still needs to be justified that our stochastic disjunctive graph generally does not bias $E(C_{max})$ too much. Simulation and exact evaluation for appropriate problems is up to show this. We are anyway of good hope, since durations are deterministic and stochastic arcs together with weighted durations take care of approximate discrete expectation calculus. SBP being extremely good for C_{max}-minimization, we think of approximately solving our problem therewith.

3.3 Interpreting the solution

Finding a good schedule for the stochastic job shop problem through our disjunctive graph approach, we get an order of priority of all involved operations. However, in one realization of our stochastic project, not all jobs generally need

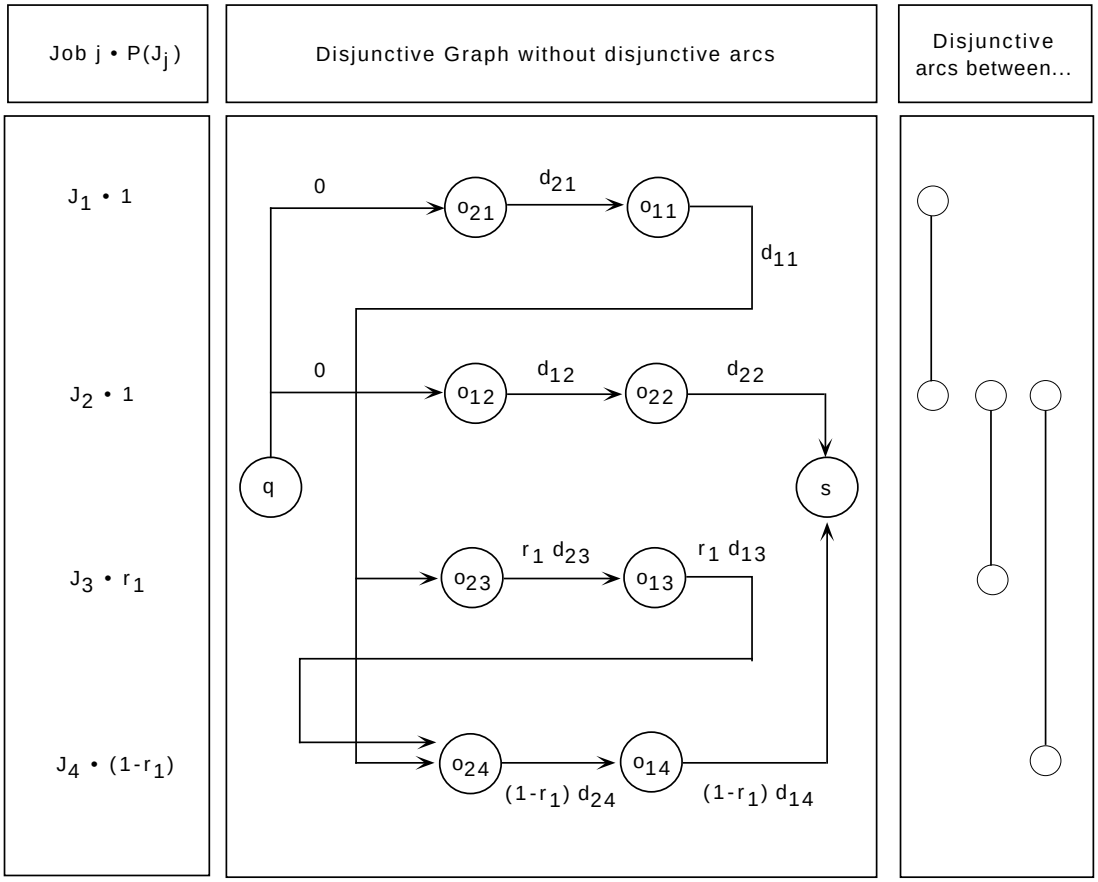


Figure 3.6: Disjunctive graph to example 3

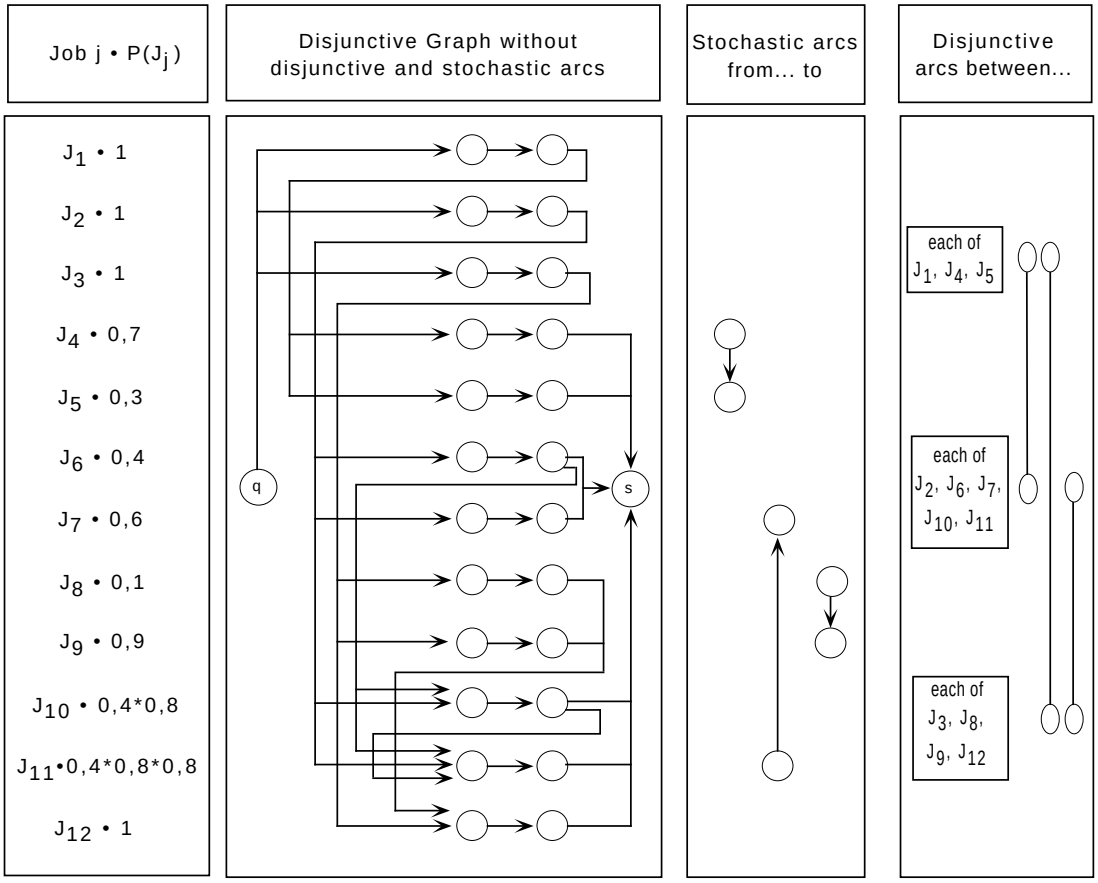


Figure 3.7: Disjunctive graph to automotive propulsion example of chapter 2

to be performed. When considering one special project outcome we therefore just drop the operations we need not to process and get a schedule for our realization. Thinking of mixed model production, we can quickly read out a schedule for any product type of our total schedule for all product types (operations, respectively). Of course, computer simulation has to show, that this schedule still is adequate. We are of good hope especially in mixed model production since product types do normally not differ too much in sequencing operations.

Chapter 4

Applications of the Model and Discussion

In this chapter we discuss our model in various facets. We first care about generalization with respect to the objective. Next, we keep our eyes onto stochastics and show in what kind of trouble we run into when relaxing "$\beta = acyclEOR$" or "$\beta = D \sim deg$". We then discuss what other heuristic attempts could be made to solve stochastic job shop scheduling. Finally, we are encouraged thinking of applicability to mixed model production.

4.1 Generalization and forthcoming problems ...

4.1.1 ... concerning the objective ...

... $\gamma = E(L_{max})$

If we subtract job J_j's weighted due date of the weighted processing time at job J_j's final operation at the corresponding arc in our stochastic disjunctive graph for all jobs, the length of a longest path form q to s approximates $E(L_{max})$. This is straight forward adaption from deterministic scheduling via disjunctive graphs.

... $\gamma = E(T_{max})$

Every schedule minimizing L_{max} minimizes T_{max}, too. Analogous is true in expectations.

... $\gamma = other$

Other objectives, such as $maxE(C)$, $maxE(L)$, or $maxE(T)$ can be considered in disjunctive graphs either. By how far approximation can be made in expectations we need to test. There is plenty of space for further research.

4.1.2 ... to $\beta = acyclGERT, D \sim deg$

Considering acyclic GERT networks, we are able to model job precedence constraints where AND nodes are involved. This is like deterministic AND logic, where regular precedence arcs are overtaken into the disjunctive graph. The problem of an AND node is, that, it is generally not yet possible to get the corresponding activation probabilities in an easy manner. There have been made attempts to estimate those probabilities, but the results have been poor so far. No citation on that work here. The situation even is more hopeless for IOR nodes. There is no straight forward way to get those conditions modeled into arcs in the stochastic disjunctive graph. Moreover, the activation probabilities of IOR nodes even seem to be more difficult to get than those of AND nodes, in general.

4.1.3 ... to $\beta = acyclEOR, D \sim general$

Letting $D \sim general$, we get more problems. Expectation calculus surely gets more biased by adding expectations of operations through machine disjunction than in the "$D \sim deg$"-case. We additionally have to cope with the same problems occured in $PERT$-project analysis.

4.1.4 ... to $\beta = GERT, D \sim deg$ or $general$

Modeling feedback by allowing circles in job's precedence constraints is the total killer of the stochastic- or even deterministic disjunctive graph concept. We namely get cycles into the disjunctive graph either, and there is no schedule to be read out of it no matter how selection is made. If we want to consider job shop scheduling where the same job has to be made several times, we can't apply methods based on disjunctive graphs. Even in deterministic job shop scheduling, there is no literature on feedback.

4.2 Other attempts to crack the new model

It might be worth to adapt other heuristics. The Giffler-Thompson approach could be adaptable. We might be able to determine the performable operations sequentially in time, elegantly planning one operation after another onto a machine using an appropriate dispatching rule.

4.3 Applications to mixed model production

As we saw in chapters 2 and 3, our new stochastic job shop model is highly interesting in mixed model production. It is possible to calculate an overall

schedule for all product types, and then read out a schedule for any of the product types. The shifting bottleneck procedure is a very good heuristic for the objective C_{max}. Even if we trade in long computing times, we need to calculate only once and get a result for the total variety of products we can manufacture. This is a clear advantage to deterministic methods which need to clarify the scheduling order for each product type seperately. Stochastic job shop scheduling makes perfect sense in mixed model production. Since product types do not differ too much in sequencing order, we are of good hope that our total mixed model production schedule gives good results for any product type. Simulation is up to prove this.

Bibliography

[1] Adams, Balas, Zawack: *The Shifting Bottleneck Procedure for Job Shop Scheduling*, Management Science 34, Vol. 3, p. 391-401, 1988

[2] Alsmeyer: *Erneuerungstheorie*, Teubner Verlag, 1991

[3] Applegate, Cook: *A computational Study of the Job Shop Scheduling Problem*, ORSA Journal of Computation 3, No. 2, p. 149-156, 1991

[4] Askin, Standridge: *Modeling and Analysis of Manufacturing Systems*, John Wiley & Sons, Inc., 1993

[5] Barlow, Proschan: *Statistical Theory of Reliability and Life Testing*, Holt, Rinehart and Winston, INC., 1975

[6] Blazewicz, Ecker, Schmidt, Weglarz: *Scheduling in Computer and Manufacturing Systems*, Springer Verlag, 1994

[7] Brucker, Jurisch, Sievers: *A Branch and Bound Algorithm for the Job Shop Scheduling Problem*, Discrete Applied Mathematics 49, p. 107-127, 1994

[8] Carlier: *The one-machine Sequencing Problem*, European Journal of Operations Research 11, p. 42-47, 1982

[9] Carlier, Pinson: *An Algorithm for Solving the Job Shop Problem*, Management Science 35, Vol. 2, p. 164-176, 1989

[10] Dempster, Lenstra, Rinnooy Kan: *Deterministic and Stochastic Scheduling*, D. Reidel Publishing Company, 1982

[11] Dauzere-Peres, Lasserre: *An Integrated Approach in Production Planning and Scheduling*, Springer Verlag, 1994

[12] Domschke, Scholl, Voß: *Produktionsplanung - Ablauforganisatorische Aspekte*, Springer Verlag, 1993

[13] Giffler, Thompson: *Algorithms for Solving Production-Scheduling Problems*, Operations Research, Vol. 8, p. 487-503, 1960

[14] Grimmet, Stirzaker: *Probability and Random Processes*, Oxford University Press, 1982

[15] Manne: *On the Job Shop Scheduling Problem*, Operations Research, Vol. 8, p. 219-233, 1960

[16] Morton, Pentico: *Heuristic Scheduling Systems*, John Wiley & Sons, Inc., 1993

[17] Neumann: *Production and Operations Management*, WIOR-Report 425, Universität Karlsruhe, 1992

[18] Neumann: *Stochastic Project Networks*, Springer Verlag, 1990

[19] Neumann: *Stochastic Project Networks: Temporal Analysis, Scheduling, and Cost Optimization I*, WIOR-Report 280, Universität Karlsruhe, 1986

[20] Neumann, Morlock: *Operations Research*, Hanser Verlag, 1993

[21] Neumann, Steinhardt: *GERT-Networks*, Springer Verlag, 1979

[22] Schwindt: *Vergleichende Beurteilung mehrerer Varianten der Heuristik von Lambrecht und Vanderveken zur Lösung des integrierten Losgrößen- und Ablaufplanungsproblems*, WIOR-Report 437, Universität Karlsruhe, 1994

[23] Schwindt: *Vergleichende Bewertung mehrerer Varianten der Heuristik von Lambrecht und Vanderveken zur sukzessiven Lösung des Losgrößensequenzproblems*, Diplomarbeit, Institut für WIOR, Universität Karlsruhe, 1993 Operations Research, Vol. 8, p. 219-233, 1960

[24] Seelbach: *Ablaufplanung*, Physica-Verlag, 1975