Scheduling under Uncertainty: Bounding the Makespan Distribution

Rolf H. Möhring*

Technische Universität Berlin, 10623 Berlin, Germany
moehring@math.tu-berlin.de,
http://www.math.tu-berlin.de/~moehring/

Abstract. Deterministic models for project scheduling and control suffer from the fact that they assume complete information and neglect random influences that occur during project execution. A typical consequence is the underestimation of the expected project duration and cost frequently observed in practice. This phenomenon occurs even in the absence of resource constraints, and has been the subject of extensive research in discrete mathematics and operations research.
This article presents a survey on the reasons for this phenomenon, its complexity, and on methods how to obtain more relevant information. To this end, we consider scheduling models with fixed precedence constraints, but (independent) random processing times. The objective then is to obtain information about the distribution of the project makespan. We will demonstrate that this is an $\#\mathcal{P}$-complete problem in general, and then consider several combinatorial methods to obtain approximate information about the makespan distribution.

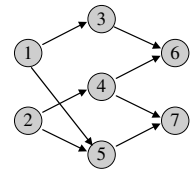
1 Uncertainty in Scheduling

In real-life projects, it usually does not suffice to find good schedules for fixed deterministic processing times, since these times mostly are only rough estimates and subject to unpredictable changes due to unforeseen events such as weather conditions, obstruction of resource usage, delay of jobs and others.

In order to model such influences, the processing time of a job $j \in V$ is assumed to be a random variable $\mathbf{p}_j$. Then $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ denotes the (random) vector of processing times, which is distributed according to a joint probability distribution Q. This distribution Q is assumed to be known, though sometimes, also partial information may suffice. In general, Q may contain stochastic dependencies, but the methods discussed in this paper require that the job processing times are stochastically independent (Methods for dependent processing times are quite different, see [14]). Furthermore, like in deterministic models, we have precedence constraints given by a directed acyclic graph $G = (V, E)$ but no resource constraints. (See [16] for an overview about the more involved case

* Supported by Deutsche Forschungsgemeinschaft under grant Mo 346/3-3 and by German Israeli Foundation under grant I-564-246.06/97

H. Alt (Ed.): Computational Discrete Mathematics, LNCS 2122, pp. 79–97, 2001.



$$C_{\max} = \max \{ \mathbf{p}_1 + \mathbf{p}_3 + \mathbf{p}_6, \mathbf{p}_1 + \mathbf{p}_5 + \mathbf{p}_7, \mathbf{p}_2 + \mathbf{p}_4 + \mathbf{p}_6, \mathbf{p}_2 + \mathbf{p}_4 + \mathbf{p}_7, \mathbf{p}_2 + \mathbf{p}_5 + \mathbf{p}_7 \}$$

Fig. 1. An example project network and its makespan $C_{\max}$.

with resource constraints.) We refer to G also as the *project network* and to the pair $(G, \mathbf{P})$ as *stochastic project network*. In the literature, they are also referred to as PERT networks, since PERT was one of the first techniques to analyze the stochastic behavior of such networks.

Now consider a particular realization $p = (p_1, \ldots, p_n)$ of the random processing time vector $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$. Since there are no resource constraints, every job j can complete at its earliest possible completion time $C_j = C_j(p)$, which is equal to the length of a longest path in G that ends with j, where the length of a job j is its processing time p_j.

The *earliest project completion* or *makespan* for the realization p is then $C_{\max}(p) := max_j C_j(p) = \max_P \sum_{j \in P} p_j$, where P ranges over all inclusion-maximal paths of G. Since the processing times $\mathbf{p}_j$ are random, the makespan C_{max} is also a random variable, and it may be written as $C_{\max} = \max_P \sum_{j \in P} \mathbf{p}_j$, i.e., as the maximum of sums over subsets of a common set of random variables. An example is given in Figure 1.
Our objective is to obtain information about the distribution of this random variable C_{max}.

The necessity to deal with uncertainty in project planning becomes obvious if one compares the "deterministic makespan" $C_{\max}(E(\mathbf{p}_1), \ldots, E(\mathbf{p}_n))$ obtained from the expected processing times $E(\mathbf{p}_j)$ with the expected makespan $E(C_{\max}(\mathbf{p}))$. Even in the absence of resource constraints, there is a systematic underestimation $C_{\max}(E(\mathbf{p}_1), \ldots, E(\mathbf{p}_n)) \leq E(C_{\max}(\mathbf{p}_1, \ldots, \mathbf{p}_n))$ which may become arbitrarily large with increasing number of jobs or increasing variances of the processing times [9]. Equality holds if and only if there is one path that is the longest with probability 1, see Theorem 1 below. This systematic underestimation of the expected makespan has already been observed by Fulkerson [6]. The error becomes even worse if one compares the deterministic value $C_{\max}(E(\mathbf{p}_1), \ldots, E(\mathbf{p}_n))$ with quantiles t_q such that $Prob\{C_{\max}(\mathbf{p}) \leq t_q\} \geq q$ for large values of q (say $q = 0.9$ or 0.95). A simple example is given in Figure 2 for a project with n parallel jobs that are independent and uniformly distributed on [0,2]. Then the deterministic makespan $C_{\max}(E(\mathbf{p}_1), \ldots, E(\mathbf{p}_n)) = 1$, while $Prob(C_{\max} \leq 1) \to 0$ for $n \to \infty$. Similarly, all quantiles $t_q \to 2$ for $n \to \infty$ (and $q > 0$).

This is the reason why good practical planning tools should incorporate stochastic methods.

We conclude this introduction with a proof of the underestimation error.

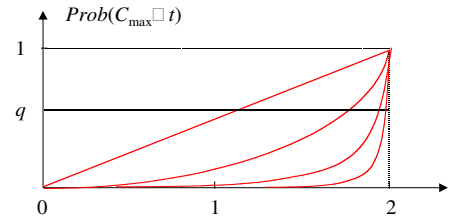


Fig. 2. Distribution function of the makespan for $n = 1, 2, 4, 8$ parallel jobs that are independent and uniformly distributed on [0,2].

Theorem 1. *Let $G = (V, E)$ be a project network with random processing time vector* $\mathbf{p}$. *Then*

$$C_{\max}(E(\mathbf{p}_1), \ldots, E(\mathbf{p}_n)) \leq E(C_{\max}(\mathbf{p}_1, \ldots, \mathbf{p}_n)).$$

Equality holds iff there is one path that is the longest with probability 1.

Proof. Since $C_{\max}$ is the maximum of sums of processing times, it is obviously a convex function of p. Thus the inequality is a special case of Jensen's inequality for convex functions. We give here an elementary proof for C_{max}.

Let $P_1, \ldots, P_k$ be the inclusion-maximal paths of G and let $Y_1, \ldots, Y_k$ denote their (random) length, i.e., $Y_i := \sum_{j \in P_i} \mathbf{p_j}$. Then $C_{\max} = \max_i Y_i$, and

$$\begin{aligned} C_{\max}(E(\mathbf{p})) &= \max_i \sum_{j \in P_i} E(\mathbf{p}_j) = \max_i E(\sum_{j \in P_i} \mathbf{p}_j) = \max_i E(Y_i) \\ &= E(Y_{i_0}) \quad \text{assume that the maximum is attained at } i_0 \\ &\leq E(\max_i Y_i) \quad \text{since } Y_{i_0} \leq \max_i Y_i \quad \text{as functions of } p \\ &= E(C_{\max}(\mathbf{p})). \end{aligned}$$

Now assume that Y_1 is the longest path with probability 1. Then, with probability 1, $C_{\max} = Y_1 \geq Y_i$. Hence $E(C_{\max}) = E(Y_1) \geq E(Y_i)$ and the above calculation yields $C_{\max}(E(\mathbf{p})) = \max_i E(Y_i) = E(Y_1) = E(C_{\max})$.

In the other direction assume that $E(C_{\max}(\mathbf{p})) = C_{\max}(E(\mathbf{p}))$. Let w.l.o.g. P_1 be the longest path w.r.t. expected processing times $E(\mathbf{p}_j)$. Then $E(Y_1) = E(C_{\max}(\mathbf{p}))$ and

$$\begin{aligned} 0 &= E(C_{\max}(\mathbf{p}) - C_{\max}(E(\mathbf{p})) = E(\max_i Y_i - \max E(Y_i)) \\ &= E(\max E(Y_i) - Y_1) = \int (\max E(Y_i) - Y_1) dQ. \end{aligned}$$

Since the integrand in non-negative, it follows that it is 0 with probability 1. Hence $Y_1 = \max E(Y_i) = C_{\max}$ with probability 1. □

2 The Complexity of Evaluating the Makespan Distribution

Due to the practical importance of stochastic scheduling, many methods have been developed over the last 35 years. For stochastically independent processing times, these methods can be roughly grouped into *simulation* methods, methods for *bounding or calculating the expected makespan*, methods for *analyzing the "most critical" path*, and methods for *bounding the whole distribution function of the makespan*. We refer to [1] for a general overview, and to [18] for an overview on bounding methods and methods for independent processing times.

Most of the early contributions have not been aware of the enormous inherent complexity of the problem, which was formally analyzed only 1988 by Hagstrom [8]. She considers the following two problems:

MEAN: Given a project network with discrete, independent processing times $\mathbf{p}_j$, compute the expected makespan $E(C_{\max}(\mathbf{p}))$.

DF: Given a project network with discrete, independent processing times $\mathbf{p}_j$ and a time t, compute the probability $Prob\{C_{\max}(\mathbf{p}) \leq t\}$ that the project finishes by time t.

Hagstrom shows that DF and the 2-state versions of MEAN, in which every processing time $\mathbf{p}_j$ has only two discrete values, are $\#\mathcal{P}$-complete. (Any $\#\mathcal{P}$-complete problem is polynomially equivalent to counting the number of Hamiltonian cycles of a graph and thus in particular $\mathcal{NP}$-complete). This result is derived from a fundamental result of Provan and Ball [20] on the $\#\mathcal{P}$-completeness of reliability problems, see Theorem 2 below.

The complexity status of the general version of MEAN is open (only the 2-state version, which has a short encoding, is known to be $\#\mathcal{P}$-complete). If the processing times $\mathbf{p}_j$ may take more than 2 values, the problem has a longer encoding that in principle could admit a polynomial algorithm for solving MEAN. However, Hagstrom provides some evidence that problems with a long encoding may still be difficult, since MEAN and DF cannot be solved in time polynomial in the number of values of $C_{\max}(\mathbf{p})$ unless $\mathcal{P} = \mathcal{NP}$.

These results show that efficient methods for calculating the expected makespan or quantiles of the distribution function of the makespan are very unlikely to exist, and thus (although in retrospect) justify the great interest in approximate methods such as bounds, simulation etc.

We will show the $\#\mathcal{P}$-completeness of the 2-state version of MEAN below in Theorem 2. To this end, we make use of a construction that envisions the jobs of our project as arcs of a directed graphs instead of vertices. This construction will also be useful for the bounding algorithms below.

This construction uses an s,t-*dag*, i.e., a directed acyclic graph $D = (N, A)$ with a unique source s and a unique sink t. Every job j of G is represented by an arc of D such that precedence constraints are preserved, i.e., if (i, j) is an edge of G, then there is a path from the end node of i to the start node of j in D. Figure 3 gives an example. Such a representation is called an *arc diagram*

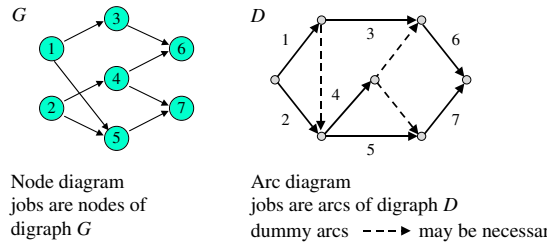


Fig. 3. Arc diagram of the project network of Figure 1.

(sometimes also *PERT network*) of the project. In general, one needs additional arcs (so-called *dummy arcs*) to properly represent the precedence constraints. A simple way to obtain an arc diagram is to take the original edges of G as dummy arcs, blow up every vertex of G to an arc with the same incoming and outgoing edges as in G, identify sources and sinks, and contract dummy arcs that are "superfluous" (i.e., the contraction does not change the original precedence constraints given by G). The arc diagram of Figure 3 is obtained in this way. Of course, arc diagrams are not unique, and it is $\mathcal{NP}$-hard to construct an arc-diagram with the fewest possible dummy arcs [11]. For our purposes, the given polynomial construction suffices.

Theorem 2. *[8] The* 2*-state version of* DF *is* $\#\mathcal{P}$*-complete.*

Proof. We will use the fact that the problem

> Reliability: Given an s,t-dag D with arc failure probabilities q_j (the *reliability network*), determine the probability that the s and t are joined by a path without arc failures (the *reliability*).

is $\#\mathcal{P}$-complete [20]. From a given instance I of Reliability, we will construct in polynomial time an instance I' of the 2-state version of DF such that the reliability of I equals 1 minus the probability that $C_{\max} \leq L-1$ for some appropriately chosen value L. This obviously proves the theorem.

First, we take the s,t-dag D of instance I as arc diagram of instance I'. Every arc (job) j may attain two processing times, 0 with the failure probability q_j, and $p_j > 0$ with probability $1-q_j$. The p_j are chosen in such a way that all paths from the source to the sink in D have the same length L. This may be obtained by starting with $p_j = 1$ for all j and then iteratively enlarging the p_j of those arcs j that are not yet on a path with the maximum length L until one path through j reaches this length L. Clearly, this is a polynomial algorithm that will output integer processing times $p_j \leq n$ such that all paths have length L. Then

$$\begin{aligned} Prob(\text{ some path is reliable }) &= 1 - Prob(\text{ all paths fail }) \\ &= 1 - Prob(\cap_i \{\text{ path } P_i \text{ fails}\}) \\ &= 1 - Prob(\cap_i \{\text{ path } P_i \text{ has length} < L\}) \\ &= 1 - Prob(\, C_{\max} \leq L-1\,). \quad \square \end{aligned}$$

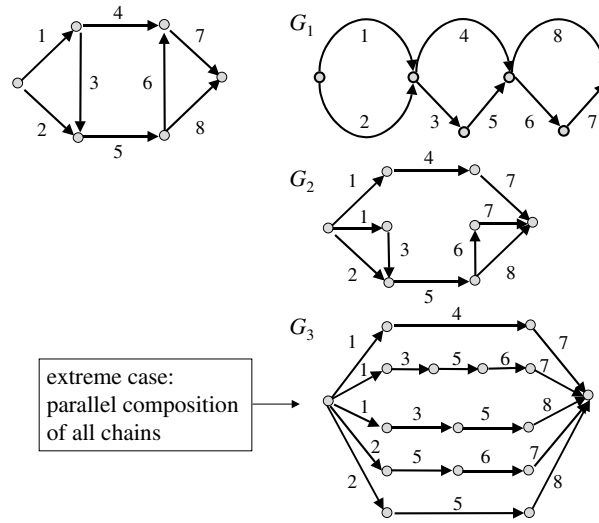


Fig. 4. Different networks G_i containing G as a minor in the arc diagram representation. Copies of a job have the same number.

3 A General Bounding Principle

Many of the methods that provide bounds for the distribution function of the makespan transform the given network (mostly represented as an arc diagram) into a network that is easier to evaluate. Typical examples in this respect are the bounds by Dodin [5], Kleindorfer [10], Shogan [22] and Spelde [23], see Section 5.

Möhring and Müller [17] give a unified model for such bounding results in terms of a *chain-minor* notion for project networks.

A network $G_1 = (V_1, E_1)$ is a *chain-minor* of a network $G_2 = (V_2, E_2)$ if (1) and (2) below hold.

(1) Every job $j \in V_1$ is represented in G_2 by a set of *copies* or *duplicates* $D(h)$, where $D(h) \cap D(j) = \emptyset$ if $h \neq j$ and $\cup_j D(j) = V_2$.
(2) Every chain C (the set of jobs on a path) of G_1 is "contained" in a chain C' of G_2 in the sense that, for every job $j \in C$, there is a duplicate $j' \in D(j)$ with $j' \in C'$. (These duplicates j may be different for different chains C of G_1.)

Figure 4 gives an example.

Theorem 3. *[17] Let G be a chain-minor of H. Then one obtains a lower bound for the distribution function F_G of the makespan of G if one gives every duplicate j' of a job j the same processing time distribution as job j, treats them as independent, and calculates the distribution function F_H of the makespan of H. In other words,*

$$Prob\{C_{\max} \leq t \text{ in } G\} \geq Prob\{C_{\max} \leq t \text{ in } H\} \text{ for every } t.$$

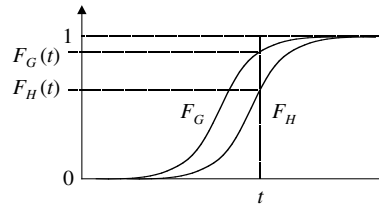


Fig. 5. An illustration of Theorem 3.

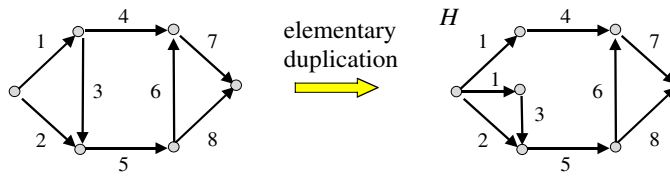


Fig. 6. The example networks for the proof of Theorem 3.

Figure 5 illustrates the relationship of F_G and F_H. Since $Prob\{C_{\max} \leq t \text{ in } G\} \geq Prob\{C_{\max} \leq t \text{ in } H\}$, F_H is called *stochastically greater* than F_G.

Proof. Let $C^G_{\max}$ and $C^H_{\max}$ denote the makespan in G and H, respectively. For a fixed realization p of processing times, the chain-minor property obviously ensures that $C^G_{\max}(p) \leq C^H_{\max}(p)$. It turns out that the independent variation of the processing time on duplicates of a job increases the makespan stochastically, i.e., $F_G \geq F_H$.

We will show this only for the special case of an "elementary duplication", i.e., one job is duplicated into two copies. It can be shown that if G is a chain-minor of H, then there is a finite sequence $\mathcal{P}_0, \ldots, \mathcal{P}_k$ of set systems such that $\mathcal{P}_0$ is the system $\mathcal{P}(G)$ of maximal paths of G, $\mathcal{P}_k$ is the system $\mathcal{P}(H)$ of maximal paths of H, and any two neighbored set systems $\mathcal{P}_i, \mathcal{P}_{i+1}$ differ by either an *elementary duplication* (i.e., only one job is duplicated, and in only two copies) or a *proper inclusion* without any duplication as G and G_1 in Figure 4. However, the intermediate set systems need not be systems of paths of a project network anymore, but only systems of pairwise incomparable subsets of V. This does, however, not matter for the proof.

The bounding property in the case of a proper inclusion is straightforward. So suppose that G and H differ by an elementary duplication as in Figure 6. The proof for this example will be generic for arbitrary elementary duplications. It follows an argument given by Dodin [5] for proving his bound, see Section 5.1.

We must show that $Prob(C^G_{\max} \leq t) \geq Prob(C^H_{\max} \leq t)$ for every $t \geq 0$. Because of the stochastic independence of the processing times, both probabilities are measures in product spaces and can thus be computed using Fubini's theorem. Although both spaces are different, they share the last components belonging to $(p_2, \ldots, p_8)$. So both measures are obtained by measuring the "shadow"

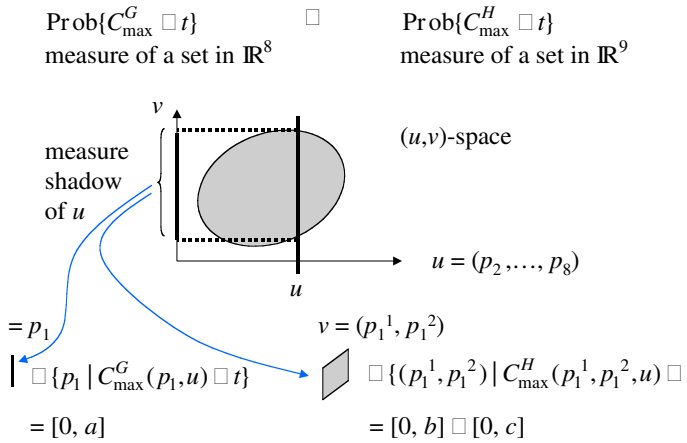


Fig. 7. Calculating probabilities by Fubini's theorem.

obtained by fixing $(p_2, \ldots, p_8)$ and integrating these shadows over $(p_2, \ldots, p_8)$. This is illustrated by Figure 7.
So it suffices to show that $Q_1(\{p_1 \mid C^G_{\max}(p_1, u) \geq t\}) \leq Q_1 \otimes Q_1(\{(p_1^1, p_1^2) \mid C^G_{\max}(p_1^1, p_1^2, u) \geq t\})$. Here Q_1 is the distribution of $\mathbf{p_1}$, and $Q_1 \otimes Q_1(\{\ldots\})$ measures the independent variation of $\mathbf{p_1}$ on the two copies of job 1 in H for fixed u. Let $A := \{p_1 \mid C^G_{\max}(p_1, u) \geq t\}$ and $B := \{(p_1^1, p_1^2) \mid C^G_{\max}(p_1^1, p_1^2, u) \geq t\}$. Obviously, A is a 1-dimensional interval, say $A = [0, a]$. Similarly, B is a 2-dimensional interval, say $B = [0, b] \times [0, c]$, since the two copies of job 1 can independently achieve their respective maxima b and c.

Assume w.l.o.g. that $b \geq c$. Then $a \geq c$. If not, then $a < c$, and the processing time vector (c, u) would give a makespan $C^G_{\max}(c, u) > t$ in G, while (c, c, u) would yield a makespan $C^H_{\max}(c, c, u) \leq t$ in H, since $(c, c) \in [0, b] \times [0, c]$. This contradicts $C^G_{\max}(c, u) = C^H_{\max}(c, c, u)$.

So $a \geq c$ and we obtain

$$\begin{aligned} Q_1(A) &= Q_1([0, a]) \geq Q_1([0, c]) \\ &\geq Q_1([0, b]) Q_1([0, c]) \\ &= Q_1 \otimes Q_1([0, b] \times [0, c]) = Q_1 \otimes Q_1(B). \quad \square \end{aligned}$$

Theorem 3 can be used to bound the distribution function of the makespan of a network from above and below. To this end, one must identify networks G_1, G_2 that "sandwich" the given network G in the sense that G_1 is a chain-minor of G and G is a chain-minor of G_2. Then the unknown makespan distribution function F_G of G is "sandwiched" by those of G_1 and G_2, i.e., $F_{G_1} \geq F_G \geq F_{G_2}$.

This brings up the question to identify networks G_1 and G_2 for which the distribution functions F_{G_1} and F_{G_2} are easier to evaluate. A proper choice is to consider series-parallel networks, since then the computation reduces to a sequence of convolutions and products of distribution functions. This is also the choice in the already quoted algorithms by Dodin, Kleindorfer, and Spelde.

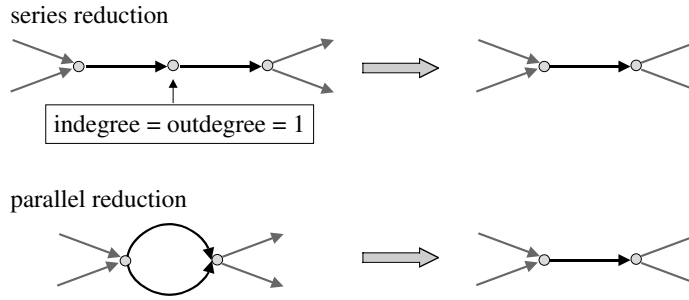


Fig. 8. Series and parallel reduction in arc diagrams.

4 Bounding with Series-Parallel Networks

A project network is *series-parallel*, if it can be reduced by a finite sequence of *series reductions* and *parallel reductions* to a network with only one job. In a series reduction, two jobs i, j that are "in series" in the arc diagram, i.e., the head of i is the tail of j and has indegree 1 and outdegree 1, are contracted to one job. In a parallel reduction, two jobs i, j that are "in parallel" in the arc diagram, i.e., have the same head and tail, are contracted to one job. These reductions are illustrated in Figure 8.

Series-parallel networks are a well-studied class of networks, see e.g. [24,15]. They can be recognized in linear time, and also a reduction sequence can be constructed in linear time, see [25].

For deterministic processing times, the makespan of a series-parallel network can be calculated along a reduction sequence by taking the sum of the processing times for a series reduction and the maximum for a parallel reduction, see Figure 9.

Since the processing times are stochastically independent, this favorable behavior generalizes to the stochastic case. One takes the distribution of the sum and the maximum of the processing times of the two jobs involved in a series and parallel reduction, respectively. In terms of distribution functions, these operations correspond to the *convolution* and the *pointwise product* of the distribution functions of the two jobs involved, see Figure 10.

It follows that the distribution function F_G of the makespan of a series-parallel network G can be calculated by convolutions and products "along" the reduction sequence of G. This approach belongs to the folklore in the field, see e.g. [13]. Although it suggests an easy and obvious algorithm, Möhring and Müller [17] show that the 2-state version of DF is still $\mathcal{NP}$-complete, but only in the weak sense. However, Mean can in this case be solved in time polynomial in the largest number of values of the makespan of a network encountered in any series-parallel reduction sequence. This number may of course be exponential in the number of jobs.

Let DF-SPdenote the restriction of DF to the class of series-parallel networks.

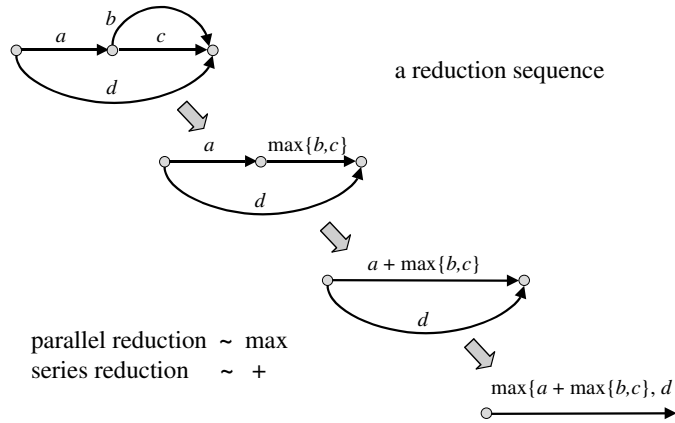


Fig. 9. Computing the makespan of a series-parallel network with deterministic processing times along a reduction sequence.

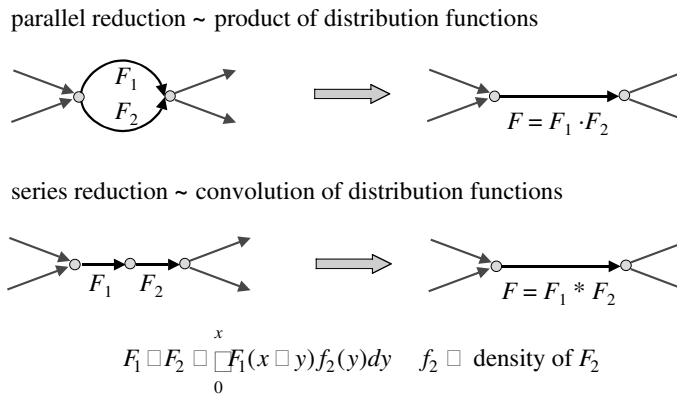


Fig. 10. Stochastic operations corresponding to series and parallel reductions.

Theorem 4. *[17]*

1. *The two-state version of* DF-SP *is $\mathcal{NP}$-hard in the weak sense.*
2. DF-SP *with* 0/1 *processing times can be solved in $O(n^3)$ time.*
3. DF-SP *with arbitrary discrete processing time distributions can be solved in $O(N^2 n)$ time, where N is the maximum number of distinct values that $C_{\max}$ attains along the reduction sequence.*

Proof. Statement 1 is proved by establishing a Turing reduction from PARTITION, which is known to be $\mathcal{NP}$-complete in the weak sense, to DF-SP. PARTITION is defined as

PARTITION: Given natural numbers $a_1, \ldots, a_n$ with $\sum a_i = 2b$, is there a partition I, J of $\{1, \ldots, n\}$ with $\sum_{i \in I} a_i = \sum_{i \in J} a_i = b$?

Now, given an instance of PARTITION as above, we construct a network G consisting of only one path with n jobs $1, \ldots, n$. Job j may attain the processing times 0 and a_j, each with probability 1/2. Obviously, this is a polynomial transformation and G is series-parallel.

Now suppose that there exists a polynomial algorithm A for calculating the value $F_G(t)$ for arbitrary t. Then apply it to the values $t_1 := b$ and $t_2 := b - 1$. If there is a subset $I \subseteq \{1, \ldots, n\}$ of the PARTITION instance with $\sum_{i \in I} a_i = b$, then the project duration $C_{\max}$ attains the value b with positive probability $Q(C_{\max} = b)$ (the realization p with $p_i = a_i$ if $i \in I$ and $p_i = 0$ otherwise has $C_{\max}(p) = b$ and probability $\frac{1}{2^n}$). Hence F_G will have a jump at $t = b$ and the values $F_G(t_1)$ and $F_G(t_2)$ differ. If there is no such subset I, there is no jump, and $F_G(t_1) = F_G(t_2)$. Hence PARTITION is Turing reducible to DP-SP.

To show statement 3., we will assume that every discrete distribution function F is represented by a sorted list of their jumps $t_1 < t_2 < \ldots < t_\ell$ with the associated values $F(t_1), \ldots, F(t_\ell)$. Then a value $F(t)$ can be computed by a simple scan through the list in $O(\ell)$ time.

Now observe that, by assumption, every distribution function, the given ones and those calculated along the reduction sequence have at most N jumps. Hence calculating the product $F_i \cdot F_j$ of two distribution functions takes $O(N)$ time, while calculating the convolution $F_i * F_j$ takes $O(N^2)$ time. Since there are $n-1$ reductions in the reduction sequence, F_G can be calculated in $O(N^2 \cdot n)$ time.

If all processing times are 0 or 1, we have $N \leq n + 1$ and thus $O(N^2 \cdot n) = O(n^3)$. This shows 2. □

Theorem 4 shows that even for series-parallel networks G, evaluating the makespan distribution function F_G is as least as difficult as PARTITION.

This is essentially due to the fact that the iterated convolutions and products along a reduction sequence may result in a discrete processing time distribution with a number of distinct values N that is exponential in the input size. However, even then it is possible to solve DF in pseudo-polynomial time, which is not possible for arbitrary partial orders, since DF is $\#\mathcal{P}$-complete in general.

We can conclude that series-parallel orders are easier to handle but still have a certain inherent complexity that—within the reduction approach—depends on the distributions generated on the way.

5 Specific Bounds

The very general bounding principle from the previous section covers the mentioned specific bounds of Kleindorfer, Spelde, Dodin and others. We will demonstrate this in two cases.

5.1 The Bound of Dodin

The algorithm of Dodin [5] works on the arc diagram D of the project network. A rough sketch is given in Figure 11 below. It carries out series and/or parallel

Input: Stochastic project network D as arc diagram
Output: A lower bound on the makespan distribution function of D

while D has more than one arc **do**
  if a series reduction is possible **then** apply one
  else if a parallel reduction is possible **then** apply one
  else find

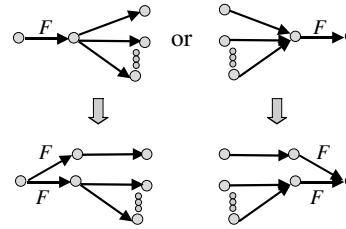


Fig. 11. The algorithm of Dodin.

reductions as long as possible. When this process stops and there are still more than one arc left, the algorithm performs an elementary duplication as depicted in Figure 11. That is, it identifies a node v of D with exactly one incoming arc i and more than one outgoing arc $j_1, \ldots, j_k$ (or, alternatively, a node with the dual situation). The incoming arc i is then used for an elementary duplication by making j_1 the only successor of the first copy of i and $j_2, \ldots, j_k$ the successors of the second copy of i (symmetrically in the alternative dual situation). The copies of i get the same distribution function as i.

An example is given in Figure 12. If one maintains an arithmetic expression with the graph operations ($+$ for a series reduction, $\cdot$ for a parallel reduction, and the identity for a duplication), then the resulting distribution function F can be read of directly from the final string $((1+4)+(6+8)\cdot 7)\cdot((2\cdot(1+3)+5)+8)$ as $F = ((F_1 * F_4) * (F_6 * F_8) \cdot F_7) \cdot ((F_2 \cdot (F_1 * F_3) * F_5) * F_8)$.

Theorem 5. *Given a project network $D = (N, A)$ with random processing times as arc diagram, Dodin's algorithm computes a lower bound for the makespan distribution function of D, which is exact if D is series-parallel [5].*

The algorithm can be implemented to run in $\mathcal{O}(|A| \cdot (conv() + prod()) + |A| \cdot d_{out})$ time, where d_{out} denotes the maximum indegree (outdegree) of the nodes of D, and $conv()$ and $prod()$ denote the time for the convolution and product of two distribution functions, respectively [12].

Proof. Fist observe that the cases in the algorithm are well defined, i.e., if no series or parallel reduction is possible, then an elementary duplication can be carried out. To see this, consider a topological sort $1, 2, \ldots$ of the nodes of D. Then node 2 has indegree 1 and outdegree > 1, so the arc from 1 to 2 qualifies for duplication.

Series and parallel reductions do not change the makespan distribution. So Dodin's algorithm is exact if only these operations occur, i.e., if the network is series-parallel. After each duplication step, the current network contains the

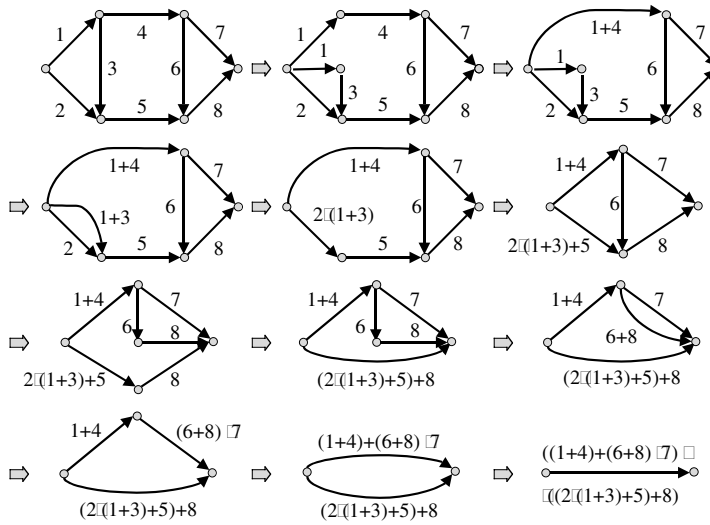


Fig. 12. An example of Dodin's algorithm. Series and parallel reductions are associated with the operations + and ·, respectively.

previous one as a chain-minor. Hence the bounding property follows directly from applying Theorem 3 in every duplication step.

To analyze the runtime of the algorithm, we define a potential function $\Phi(D) = \sum_{v \in N} d_{out}(v) \cdot h(v) + \sum_{v \in N} d_{in}(v) \cdot \bar{h}(v)$. Here $d_{in}(v)$ and $d_{out}(v)$ denote the indegree and the outdegree of node v, while $h(v)$ and $\bar{h}(v)$ are the *height* of v (the least number of arcs on a directed path from the source of D to v) and the *dual height* of v (the least number of arcs on a directed path from v to the sink of D), respectively.

Obviously, $\Phi(D)$ decreases in two successive iteration by at least 1 since every duplication is followed by a series reduction. This shows that, even for arbitrary choices of reductions and duplications, the algorithm terminates after at most $\mathcal{O}(|N|^2) = \mathcal{O}(|A|^2) = \mathcal{O}(n^2)$ iterations. Theorem 3 then yields that the distribution function F of the last arc is a lower bound for the makespan distribution function F_D. When D is series-parallel, no duplication is made, and $F = F_D$.

The better runtime of $\mathcal{O}(|A| \cdot (conv() + prod()) + |A] \cdot d_{out})$ results from a special choice of the arcs to duplicate and an "on the fly" handling of series and parallel reductions that are made locally possible by a duplication. In the first phase, all possible series and parallel reductions are performed in linear time by the series-parallel recognition algorithm of Valdes, Tarjan, and Lawler [25].

The second phase takes care of all duplications plus possible series and parallel reductions resulting from a duplication. To this end, one maintains a list L of nodes at which a series reduction becomes possible, which is initially empty. Every duplication is done with job j that is the arc from the source s to the first node v in a topological sort. The duplication is followed by the series re-

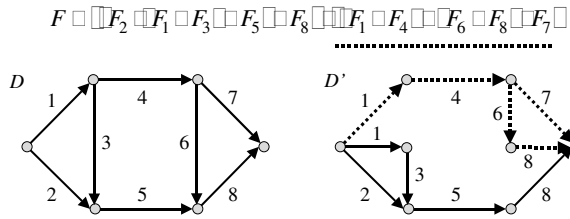


Fig. 13. Interpreting Dodin's bound as a chain-minor bound. The dotted part of D' corresponds to the dottily underlined part of F.

duction involving one of the copies of j and, if possible, a parallel reduction involving the arc resulting from this series reduction. Checking for this parallel reduction takes $\mathcal{O}(d_{out})$ time. Furthermore one updates the list L since a duplication plus the added reductions may create new possibilities for series reductions, and empties it before the next duplication by carrying out the corresponding series reductions. As a result, all series and parallel reductions occurring in the second phase are either handled in $\mathcal{O}(d_{out})$ time together with a duplication, or by emptying list L which takes $\mathcal{O}(|A|)$ steps altogether. Using the modified potential function $\Phi'(D) = \sum_{v \in N \setminus \{source\}} d_{out}(v)$, one easily sees that the second phase has $\mathcal{O}(|A|)$ duplication steps. So altogether, the runtime is $\mathcal{O}(|A| \cdot (conv() + prod()) + |A] \cdot d_{out})$. □

If one reverses the sequence of networks constructed by Dodin's algorithm by undoing all series and parallel reductions, but keeping all duplications, then the resulting network D' contains the given network D as a chain-minor, and the distribution function F computed by Dodin's algorithm equals the distribution function of the makespan of D', i.e., $F = F_{D'}$. This is illustrated in Figure 13 for the example from Figure 12. This provides another proof of the bounding property of Dodin's algorithm.

This shows that Dodin's algorithm implicitly constructs a series-parallel network D' whose makespan distribution is taken as bound for F. In structural respect, the quality of this bound depends on how close D' is to the original network D. Bein, Kamburowski, and Stallmann [3] have introduced two measures that indicate the proximity of a network D to a series-parallel network D'. They are illustrated in Figure 14 below.

One of these measures, the number of *node reductions* is directly related to the process of elementary duplications. A reduction of a node v with $deg_{in}(v) = 1$ is defined as a sequence of $d_{out}(v) - 1$ elementary duplications of the arc ending in v as in Dodin's algorithm. (An analogous reduction can be performed if $d_{out}(v) = 1$). Bein, Kamburowski, and Stallmann [3] have shown that the least number $nr(D)$ of *node reductions* to obtain a series-parallel network from D can be computed in polynomial time by solving a vertex cover problem in an auxiliary, transitively orientable graph. So it provides an efficiently computable measure for the proximity to being series-parallel.

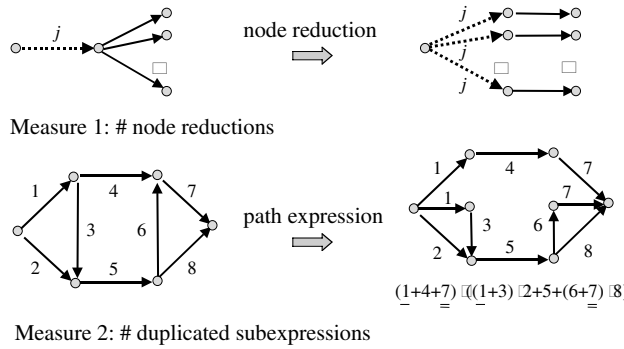


Fig. 14. Measuring the proximity to series-parallel networks.

The second measure they introduce, is the number of *duplicated subexpressions* in a "path expression" containing all the paths of D. In our terminology, the path expression is the arithmetic expression illustrated in Figures 12 and 13. Such an expression uniquely defines a series-parallel network, and, because of the required containment, the second measure is equivalent to the least number $ds(D)$ of duplicated subexpressions in the arithmetic expression of a series-parallel network containing D as a chain-minor. Kamburowski, and Stallmann [3] proved that $nr(D) \leq ds(D)$. Naumann [19] then showed that both measures are in fact equal, i.e., $nr(D) = ds(D)$ for every arc diagram D.

Of course, it is not only this distance that determines the quality of the bound, but also the actual processing time distributions.

5.2 The Bounds of Spelde

Spelde [23] presented both an upper and a lower bound that can be seen as direct applications of the chain-minor principle. The series-parallel networks D_1 and D_2 defining these bounds for D are very simple, they consist only of parallel chains as illustrated in Figure 15. D_1 defines the upper bound and consists of any system of pairwise disjoint chains of D covering all arcs of D, while D_2, which defines the lower bound, uses all maximal chains of D (the extreme case in Figure 4).

Although these bounds seem very course at first glance, they have an important advantage. This lies in the fact that most other methods (including the bound by Dodin) assume that the processing time distributions are known. This is usually not the case in practice and often inhibits the use of these methods.

Spelde's bounds offer a possibility to get around this information deficit. Since the bounding networks D_ℓ, $\ell = 1, 2$, are a parallel composition of paths, their makespan distribution functions F_{D_ℓ} are the product of the distribution functions of the lengths of these paths, say $F_{D_\ell} = F_1 \cdot F_2 \cdot \ldots \cdot F_{N_\ell}$, where F_i is the distribution function of the i-th path.

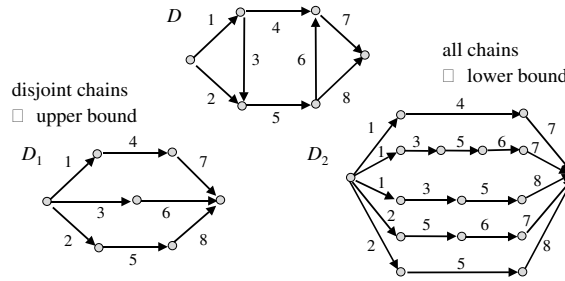


Fig. 15. The bounds of Spelde.

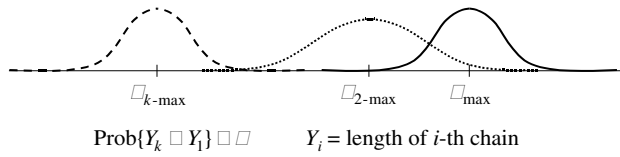


Fig. 16. The k longest paths in expectation suffice.

If the network D is large enough, i.e., all paths contain "enough" jobs, then, by the central limit theorem, every F_i is approximately a normal distribution function, whose mean μ_i and variance σ_i^2 are obtained as the sum of the means and variances of the processing times $\mathbf{p}_j$ of all jobs j contained in the i-th path. This has been studied by many authors, see, e. g., [7,2,21]. Anklesaria and Drezner [2] as well as Sculli and Shum [21] considered multi-variant normal distributions in order to model correlations among paths. They also report on computational experiences on some small networks and obtained very promising results. In practice, we have observed that 10 jobs on a path already yield an excellent approximation of the path length distribution.

Hence it suffices to know the expected processing time $E(\mathbf{p}_j)$ and the variance $V(\mathbf{p}_j)$ of every job in order to calculate Spelde's bounds. There is, however, a complication for D_2 since the number N_2 of all paths of D_2 may be exponential in the size of the given network D. This can be overcome by calculating the first k longest paths w.r.t. expected processing times $E(\mathbf{p}_j)$, until $Prob\{k\text{-th path is longer than 1st path}\} \leq \varepsilon$ for a given accuracy parameter ε (say $\varepsilon = 0.05$). If $F_1, F_2, \ldots, F_k$ are the normal distribution functions of these paths, then $F_{G_2} \approx F_1 \cdot F_2 \cdot \ldots \cdot F_k$, see Figure 16.

In fact, this method contains the traditional PERT as a special case, since PERT only analyzes the distribution of the path with the longest expected path length.

Due to the reduced number of paths, one cannot guarantee the bounding property anymore. However, for a sufficiently large number of paths it is very likely that one still obtains a stochastic upper bound, at least for large quantiles.

In order to find the k longest paths, a variant of the algorithm proposed by Dodin [4] can be used.

5.3 Computational Experience

Ludwig, Möhring, and Stork [12] have implemented several bounds based on the chain-minor principle (Kleindorfer [10], Dodin [5], and Spelde [23]), and have made an extensive computational study of their bounding behavior on benchmark networks with up to 1200 jobs and a variety of different distributions.

In the code, the representation of the processing time distributions was a crucial task, as it had to be chosen with respect to an efficient and numerically stable computation of the sum and the maximum of random variables. Since the maximum operation is performed easily on distribution functions (it correspond to the product), it was decided to represent each random variable by a piecewise linear approximation of its distribution function. A good approximation was obtained by choosing the supporting points arranged equidistantly on the ordinate. Thus, every distribution function is represented by its inverse. Furthermore, all distribution functions were represented by the same number of supporting points. This allows a simple implementation of the product and the convolution procedure. The product operation can be performed in linear time in the number sp of supporting points whereas the convolution requires quadratic running time in sp. In addition, these procedures turn out to compute reasonably stable results when a suitable number of supporting points is chosen.

In order to measure the quality of the computed bounds, they were compared with approximate distribution functions that were computed by extensive simulation with more than 1.5 million realizations for each instance.

Concerning runtime, the convolution operator consumed most of the computation time and therefore network transformations (which cause larger theoretical running times) are not the bottleneck. Spelde's lower bound based on k longest paths does not need any convolutions at all, and requires negligible computation times compared with the others (a factor of 20 to 50 times faster).

The quality of the computed bounds depended on the two test sets used. For one of them, the average relative error of the computed bounds is less than 1% for most of the considered instances, number of supporting points and distributions. For the second test set, the relative errors are a little larger, but still vary only between 1.6% and 3.7%. The maximum error occasionally exceeds 15%. However, this only occurs for the quantiles which are directly affected by the computation of boundary points at the computation of convolutions, i.e., the 0.01 and the 0.99 quantile. When disregarding these extreme quantiles, the maximum relative error reduces to 11%. In the case of Spelde's bound, $k \leq n/3$ paths were sufficient for this accuracy.

Perhaps the most important conclusion from this study is that Spelde's heuristic procedure based on the Central Limit Theorem provides excellent estimates at virtually no computational expense, and with only partial information about the processing time distributions. The approach is therefore a remarkable

alternative to (computationally costly) simulation techniques which are mostly used in practice.

References

1. V. Adlakha and V. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR*, 27(3):272–296, 1989.
2. K. P. Anklesaria and Z. Drezner. A multivariate approach to estimating the completion time for PERT networks. *J. Opl. Res. Soc.*, 40:811–815, 1986.
3. W. W. Bein, J. Kamburowski, and M. F. M. Stallmann. Optimal reduction of two-terminal directed acyclic graphs. *SIAM J. Comput.*, 21:1112–1129, 1992.
4. B. Dodin. Determining the k most critical paths in PERT networks. *Oper. Res.*, 32(859-877), 1984.
5. B. Dodin. Bounding the project completion time distribution in PERT networks. *Oper. Res.*, 33:862–881, 1985.
6. D. R. Fulkerson. Expected critical path lengths in PERT networks. *Oper. Res.*, 10:808–817, 1962.
7. D. I. Golenko. *Statistische Methoden der Netzplantechnik.* B. G. Teubner, Stuttgart, 1972.
8. J. N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18:139–147, 1988.
9. U. Heller. On the shortest overall duration in stochastic project networks. *Methods Oper. Res.*, 42:85–104, 1981.
10. G. B. Kleindorfer. Bounding distributions for a stochastic acyclic network. *Oper. Res.*, 19:1586–1601, 1971.
11. M. S. Krishnamoorthy and N. Deo. Complexity of the minimum-dummy-activities problem in a PERT network. *Networks*, 9:189–194, 1979.
12. A. Ludwig, R. H. Möhring, and F. Stork. A computational study on bounding the makespan distribution in stochastic project networks. Technical Report 609, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998. To appear in and Annals of Oper. Res.
13. J. J. Martin. Distribution of the time through a directed acyclic network. *Oper. Res.*, 13:46–66, 1965.
14. I. Meilijson and A. Nadas. Convex majorization with an application to the length of critical paths. *J. Appl. Prob.*, 16:671–677, 1979.
15. R. H. Möhring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, Nato Advanced Study Institutes Series, pages 105–193. D. Reidel Publishing Company, Dordrecht, 1989.
16. R. H. Möhring. Scheduling under uncertainty: Optimizing against a randomizing adversary. In K. Jansen and S. Khuller, editors, *Approximation Algorithms for Combinatorial Optimization, Proceedings of the Third International Workshop APPROX 2000, Saarbrücken*, pages 15–26. Springer-Verlag, Lecture Notes in Computer Science, vol. 1913, 2000.
17. R. H. Möhring and R. Müller. A combinatorial approach to bound the distribution function of the makespan in stochastic project networks. Technical Report 610, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
18. R. H. Möhring and F. J. Radermacher. The order-theoretic approach to scheduling: The stochastic case. In R. Słowiński and J. Węglarz, editors, *Advances in Project Scheduling*, pages 497–531. Elsevier Science B. V., Amsterdam, 1989.

19. V. Naumann. Measuring the distance to series-parallelity by path expressions. In E. W. Mayr, G. Schmidt, and G. Tinhofer, editors, *Proceedings 20th International Workshop on Graph-Theoretic Concepts in Computer Science WG'94*, pages 269–281. Springer-Verlag, Lecture Notes in Computer Science, vol. 903, 1995.
20. J. S. Provan and M. O. Ball. The complexity of counting cuts and of the probability that a graph is connected. *SIAM J. Comput.*, 12:777–788, 1983.
21. D. Sculli and Y. W. Shum. An approximate solution to the PERT problem. *Computers and Mathematics with Applications*, 21:1–7, 1991.
22. A. W. Shogan. Bounding distributions for a stochastic PERT network. *Networks*, 7:359–381, 1977.
23. H. G. Spelde. *Stochastische Netzpläne und ihre Anwendung im Baubetrieb.* PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 1976.
24. K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *J. Assoc. Comp. Mach.*, 29:623–641, 1982.
25. J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series-parallel digraphs. *SIAM J. Comput.*, 11:298–314, 1982.