Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks

by

Arfst Ludwig Rolf H. Möhring
Frederik Stork

No. 609/1998

A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks

Arfst Ludwig* Rolf H. Möhring‡ Frederik Stork‡ §

June 30, 1998, revised April 14, 2000

Abstract

Due to the practical importance of stochastic project networks (PERT-networks), many methods have been developed over the past decades in order to obtain information about the random project completion time. Of particular interest are methods that provide (lower and upper) bounds for its distribution, since these aim at balancing efficiency of calculation with accuracy of the obtained information.

We provide a thorough computational evaluation of the most promising of these bounding algorithms with the aim to test their suitability for practical applications both in terms of efficiency and quality. To this end, we have implemented these algorithms and compare their behavior on a basis of nearly 2000 instances with up to 1200 activities of different test-sets. These implementations are based on a suitable numerical representation of distributions which is the basis for excellent computational results. Particularly a distribution-free heuristic based on the Central Limit Theorem provides an excellent tool to evaluate stochastic project networks.

1 Introduction

In order to cope with stochastic influences when executing projects, planners naturally assume activity durations to be random. In combination with precedence constraints among activities this leads to so-called *stochastic project networks* or *PERT networks* $D = (N, A, \boldsymbol{p})$. Here, $A = \{1, ..., n\}$ is the set of activities and (N, A) is a digraph (or network) characterizing the precedence constraints as activity-on-arc representation. The nodes $v \in N$ of D are called *project events*. The distribution of activity durations is given by a vector $\boldsymbol{p} = (\boldsymbol{p}_1, ..., \boldsymbol{p}_n)$ of independent random variables, where $\boldsymbol{p}_j$ denotes the distribution associated with activity $j \in A$. In principle, these distributions are assumed to be known (though, as will become clear later, there are methods that can deal with incomplete information).

*Lufthansa Systems Berlin GmbH, Fritschestraße 27–28, D-10585 Berlin, Germany, Arfst.Ludwig@LHSystems.COM, http://www.lhsystems.com

‡Technische Universität Berlin, Department of Mathematics, Sekr. MA 6–1, Straße des 17.Juni 136, D-10623 Berlin, Germany, {moehring,stork}@math.tu–berlin.de, http://www.math.tu–berlin.de/coga.

§The author is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Mo 446/3-3

Typical tasks in the evaluation of stochastic project networks are the computation of "most critical" paths, "most critical" activities, as well as the calculation of the distribution function of the project completion time and/or some of its moments. Unfortunately, these problems are very hard to solve in general. With respect to determining the distribution function, researchers have therefore focused on computing both upper and lower bounds for the exact, but unknown distribution function. However, none of these approaches has been systematically tested and empirically analyzed within a comprehensive computational study.

Contribution of the paper. The purpose of this paper is to provide an extensive computational evaluation of stochastic bounds on the makespan distribution with the aim to test their suitability for practical applications both in terms of efficiency and quality. We have selected the most promising ones from the literature, which have been developed by Kleindorfer [11] (lower and upper bound), Dodin [7] (upper bound), and Spelde [22] (lower bound and heuristic procedure). They all rely on structural modifications of the underlying network and presume independence of activity durations. Their common idea is to modify the given network into a series-parallel project network, for which the makespan distribution can be computed more easily.

We have implemented these algorithms and compare their behavior on a basis of nearly 2000 instances with up to 1200 activities of different test-sets. These implementations are based on a suitable numerical representation of distributions which allows a fast and stable implementation of the required numerical operations. The bounds are compared with the exact distribution which is computed by exhaustive simulation. Our computations show that both quality and computation time is excellent. Networks with 1200 activities can be evaluated within less than 10 seconds on a desktop computer. We also consider a heuristic procedure that is based on the Central Limit Theorem. It is particularly suitable for coping with incomplete information since only mean $E[\boldsymbol{p}_j]$ and variance $V[\boldsymbol{p}_j]$ of the distributions are required. It provides results of similar quality within fractions of time when compared to the bounding routines. We here require less than one second of computation time.

Related work. Since most planning processes in industry and commerce bear uncertainty, there is a vast literature that considers problems being related to stochastic project networks. Early work is compiled in [4, 13, 9]; more recent reviews have been contributed by Adlakha and Kulkarni [1] and Soroush [21].

Hagstrom [10] has shown that some of the most fundamental problems in this respect are $\#P$-complete in general. Assuming two-point distributions, she has shown that the computation of even a single point of the distribution function of the project completion time is $\#P$-complete. The same holds for the computation of the mean. For more complicated distributions the complexity status of computing the mean is open, since the longer encoding of the distributions may possibly admit a polynomial algorithm. But such an efficient algorithm seems very unlikely to exist, since, assuming discrete distributions, Hagstrom has also shown that the mean cannot be computed in time polynomial in the number of possible values of the makespan, unless $P = NP$. Intuitively, the possibly exponential number of paths in the network and the fact that distributions of path lengths are correlated are the reasons for the hardness of these problems.

Methods for computing stochastic bounds on the makespan distribution have been proposed by Robillard and Trahan [17], Kleindorfer [11], Dodin [7], Spelde [22], Shogan [20], and Meilijson and Nadas [15]. While the first four approaches require independent activity durations, the method proposed by Shogan [20] partially allows dependencies among the distributions of the activity durations. However, in the independent case Shogan's bounds reduce to those of Kleindorfer. Möhring and Müller [16] introduced a unified model for such bounding results in terms of a *chain–minor* notion for project networks that covers and generalizes all approaches which are based on the transformation of the given project network into series-parallel networks. The bound proposed in [15] covers arbitrary dependencies among activity durations but is based on a weaker stochastic ordering of random variables.

Organization of the paper. In the next section, we will establish the stochastic basis for the bounding procedures which are stated in Section 3. Section 4 is then concerned with the computational study, where we also discuss the implementation of a suitable representation of distributions for our purpose. We conclude with some remarks on future research.

2 Stochastic Background

A random variable $\boldsymbol{x}$ is said to be *stochastically smaller* than $\boldsymbol{y}$ if $E[g(\boldsymbol{x})] \leq E[g(\boldsymbol{y})]$ for all non-decreasing real functions g. For real-valued random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ this is equivalent to $F_{\boldsymbol{x}} \geq F_{\boldsymbol{y}}$, where $F_{\boldsymbol{x}}$ and $F_{\boldsymbol{y}}$ are the distribution functions of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Throughout the paper we compare distributions on the basis of this stochastic order (for details on this and related stochastic orders see [19]). We assume that activity durations are given by independent continuous distributions of finite range, represented by their distribution functions $F_j(t) = Prob(\boldsymbol{p}_j \leq t), j \in A$. Computationally, we handle all distribution functions as piecewise linear functions with sp supporting points. Details are given in Section 4.3 below.

The basic stochastic operations for computing the project makespan in a stochastic network are the *maximum* and the *sum* of random variables. For independent $\boldsymbol{x}$ and $\boldsymbol{y}$ it is well known that the distribution function of their maximum $F_{\max\{\boldsymbol{x},\boldsymbol{y}\}}$ can be computed as $F_{\boldsymbol{x}} \cdot F_{\boldsymbol{y}}$. The distribution function of their sum is the *convolution* $F_{\boldsymbol{x}} * F_{\boldsymbol{y}} := \int_{-\infty}^{\infty} F_{\boldsymbol{y}}(t-s) f_{\boldsymbol{x}}(s) ds$, where $f_{\boldsymbol{x}}$ denotes the density function of $\boldsymbol{x}$.

Since random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ of activity completion times are correlated in general, $F_{\boldsymbol{x}} \cdot F_{\boldsymbol{y}}$ is not the distribution function of their maximum. However, the following inequalities are valid.

$$F_{\boldsymbol{x}} \cdot F_{\boldsymbol{y}} \leq F_{\max\{\boldsymbol{x},\boldsymbol{y}\}} \leq \min\{F_{\boldsymbol{x}}, F_{\boldsymbol{y}}\} \tag{1}$$

Here, $\min\{F_{\boldsymbol{x}}, F_{\boldsymbol{y}}\}$ denotes the pointwise minimum of $F_{\boldsymbol{x}}$ and $F_{\boldsymbol{y}}$. For the validity of (1) we refer to Esary, Proschan, and Walkup [8] and Kleindorfer [11]. Note that we do not need bounding inequalities for the sum of random variables because this operation is applied to independent random variables only and can therefore be computed exactly (within reasonable numerical precision).

Finally, for notational convenience, we define δ_t as the characteristic function of the interval $[t, \infty)$, $t \geq 0$, i.e., $\delta_t(s) := 0$ if $s < t$ and $\delta_t(s) := 1$ if $s \geq t$.

3 Stochastic Bounds on the Project Makespan

In this section we give an algorithmic description of the implemented bounds. We have considered algorithms for two upper and two lower bounds as well as one heuristic approach which is based on the Central Limit Theorem. The common paradigm of all methods is to transform the given network D into a series-parallel network. It is well known that such networks can be evaluated more efficiently (see, e. g., Martin [14]). The reason for this fact is that, when performing all required $maximum$ operations on the random variables ($product$ operations on the corresponding distribution functions) in a suitable ordering, all involved distributions are independent.

In the following we denote the source and the target node of an activity j in D by $s(j)$ and $t(j)$, respectively. The nodes $\bar{s}$ and $\bar{t}$ are the (unique) global source and sink of D.

3.1 The Bounds of Kleindorfer

Kleindorfer [11] provided both a stochastic upper and a stochastic lower bound on the makespan distribution of stochastic project networks. The procedures traverse the network along a topological sort and assign a distribution function $F_{[v]}$ to each node v of the network (the brackets in $F_{[v]}$ indicates that the distribution is associated with an event of D instead of an activity). $F_{[v]}$ is computed as the product (upper bound) or minimum (lower bound) of all convolutions $F_{[s(j)]} * F_j$ with $t(j) = v$. Then, with (1), $F_{[v]}$ bounds the distribution function of the start time of all activities i with $s(i) = v$ from above (stochastically lower bound) or below (stochastically upper bound). Details for the upper bound are provided by Algorithm 1. For the lower bound simply replace the product operator by the minimum operator.

Input : A directed acyclic graph $D = (N, A)$; a distribution function F_j for each activity $j \in A$.
Output : A distribution function of an upper bound on the makespan.

$F_{[\bar{s}]} := \delta_0$;
for *nodes* $v \in N \setminus \{\bar{s}\}$ *along a topological sort* **do**
  $F_{[v]} := \prod_{j \in A, t(j)=v} [F_{[s(j)]} * F_j]$;
return $(F_{[\bar{t}]})$;

Algorithmus 1: Upper bound of Kleindorfer.

For both bounds, at most $|A|$ convolution and product (minimum) operations are performed. The running time is therefore $O(|A| \cdot (conv(sp) + prod(sp)))$ where $conv(sp)$ and $prod(sp)$ denote the complexity of the convolution and the product operation depending on the number sp of supporting points.

3.2 Dodin's Upper Bound

The upper bound of Kleindorfer is exact for each node v if the completion times of the incoming arcs (activities) j are stochastically independent. This is the case if the paths from $\bar{s}$ to v are pairwise disjoint. A bounding algorithm proposed by Dodin [7] is exact for a larger class of networks, namely *series-parallel* networks. It utilizes the classical series-parallel reductions to a single arc (see, e. g., [14]), and, if no such reduction is possible, a so-called *duplication* is performed.

1. **Series reduction:** If there exists a node $v \in N$ such that $Indegree(v) = 1$ and $Outdegree(v) = 1$, then the activities $i, j \in A, t(i) = s(j) = v$ are said to be in *series*. They are substituted by a single activity h, where $s(h) = s(i)$ and $t(h) = t(j)$. The distribution function F_h of the duration of h is $F_i * F_j$.

2. **Parallel reduction:** If there exist activities $i, j \in A$ such that $s(i) = s(j)$ and $t(i) = t(j)$, then i and j are said to be in *parallel*. They are substituted by a single activity h, where $s(h) = s(i)$ and $t(h) = t(i)$. The distribution function F_h of the duration of h is $F_i \cdot F_j$.

3. **Duplication:** If there exists a node $v \in N$ such that $Indegree(v) = 1$ and $Outdegree(v) > 1$, then v and the activity $i \in N, t(i) = v$, are duplicated as follows: Insert a new node v' and a new activity i' such that $s(i') = s(i)$ and $t(i') = v'$. Then select an activity $j \in A, s(j) = v$, and set $s(j) := v'$. The distribution function $F_{i'}$ of the duration of i' is set to F_i. An analogous duplication can be performed if $Indegree(v) > 1$ and $Outdegree(v) = 1$.

Having reduced the network to a single arc j by repeatedly applying one of these network operations, the distribution function which is associated with j is returned. Dodin has proven that the algorithm always terminates and that the computed distribution function provides an upper bound on the distribution of the makespan of the original network. Moreover, Dodin has shown that the bound is tighter than the upper bound proposed by Kleindorfer. A detailed description of our implementation is given by Algorithms 2 and 3. Note that each duplication step is followed by a series reduction and, if applicable, by a parallel reduction. These operations are integrated in the duplication subroutine (Algorithm 2), see also Figure 1.

If neither a series reduction nor a parallel reduction is applicable, the node chosen for duplication is not uniquely determined. Since the correlation among involved path lengths increases with each duplication we would like to perform as few of them as possible. A measure in this direction is the number of *node reductions* introduced in [3]. A reduction on node v with $Indegree(v) = 1$ is defined as a sequence of $Outdegree(v) - 1$ duplications on v (an analogous reduction can be performed if $Outdegree(v) = 1$). Bein, Kamburowski, and Stallmann [3] have have shown that the minimal number of *node reductions* to obtain a series-parallel network can be computed in polynomial time by solving a vertex cover problem in an auxiliary, transitively orientable graph. However, we did not integrate their result because first, it requires that *all* duplications on a selected node are performed which is not necessarily advantageous for our application. Second, the quality of the result of Dodin's algorithm

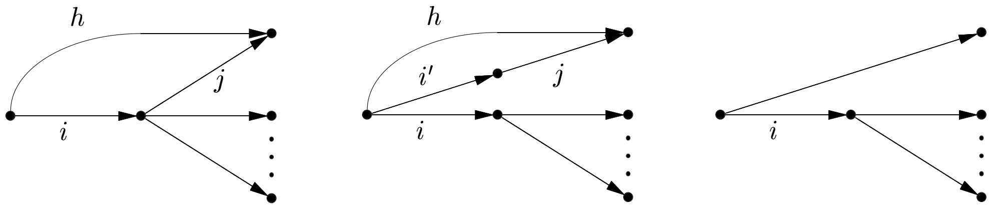

Figure 1: After a duplication step we immediately perform a series and, if applicable, a parallel reduction.

depends on the distribution of the duration of the activities which have been duplicated. For these reasons, the integration of their reduction algorithm will not lead to an improvement in general.

The worst case complexity of Algorithm 3 is $O(|A| \cdot (conv(sp) + prod(sp)) + d_{in}d_{out})$, where d_{in} (d_{out}) denotes the maximum in-degree (out-degree) of the nodes of D. Although it takes only linear time to decide whether a given network is series-parallel [23], Dodin's algorithm requires more effort: Once a reduction step has been performed it has to be checked if new parallel arcs have been created (first if-statement in Algorithm 2).

3.3 The Bounds of Spelde

Spelde [22] investigated sets of paths of the network in order to obtain bounds on the distribution function of the makespan. He derived both a lower and an upper bound by considering a set of pairwise disjoint paths and the set of all different paths, respectively. The bounds are subsequently obtained by computing the maximum over all (random) lengths of the considered paths.

When computing such a set of paths, we use the sum of the expectations of the activity durations to measure their length (it could also be valuable to choose other criteria such as the so-called criticality index, cf., e. g., Dodin [6]).

Lower bound. If we consider a subset of paths, such that no activity occurs on more than one path, then their lengths are stochastically independent. Thus, no error occurs when computing the maximum on these path lengths by the product operator. Since we have not taken all paths into account we obtain a stochastic lower bound on the makespan distribution. Note that the paths need not necessarily be $\bar{s}$-$\bar{t}$-paths, i.e. paths from the global source to the global sink of the network. It suffices to consider only subsets of activities of $\bar{s}$-$\bar{t}$-paths. Implementation details are given in Algorithm 4. The second condition $length(P) = 0$ of the repeat-loop is required in order to handle dummy activities of length zero which may occur in activity-on-arc networks. The worst case complexity is $O(|A| \cdot (conv(sp) + prod(sp)) + |A|^2)$.

Upper bound / heuristic approach / incomplete information. The upper bound is obtained by simply considering all $\bar{s}$-$\bar{t}$-paths. Clearly, we now have to deal with heavy dependencies among path lengths. Moreover, since the number of paths is exponential in the number of activities we cannot generate all paths in general and thus the straightforward approach to take the maximum of all path lengths is not applicable. Within our

Input : A directed acyclic graph $D = (N, A)$ (with associated distribution functions); an arc $j \in N$; the list L of nodes suitable for a series reduction.
Output : The graph D after the reduction step with respect to j.

$v := s(j)$;
$i :=$ incoming arc of v;
if *there exists an arc* h *with* $s(h) = s(i)$ *and* $t(h) = t(j)$ **then**
    $F_h := F_h \cdot (F_i * F_j)$;
    Remove arc j from D
else
    $F_j := F_i * F_j$;
    $s(j) := s(i)$;
if v *has no outgoing arcs* **then** remove v from D;
Update the list L;
return (D)*;*

Algorithmus 2: The Reduction Subroutine of Dodin's Algorithm

Input : A directed acyclic graph $D = (N, A)$; a distribution function F_j for each activity $j \in A$.
Output : A distribution function of an upper bound on the makespan.

Perform all possible parallel reductions in D;
Create a list L containing all nodes v with $Indegree(v) = Outdegree(v) = 1$;
while $|A| > 1$ **do**
    while $L \neq \emptyset$ **do**
        Extract node v from L;
        Reduction(D, outgoing arc j of v, L);
    if $|A| = 1$ **then**
        break;
    Find a node v with $Outdegree(v) > Indegree(v) = 1$ and a suitable outgoing arc j of v;
    Reduction(D, j, L);
return *distribution function of the remaining arc;*

Algorithmus 3: Upper bound of Dodin

Input : A directed acyclic graph $D = (N, A)$; a distribution function F_j for each activity $j \in A$.
Output : A distribution function of a lower bound on the makespan.

```
F := δ_0;
for j ∈ A do
  label(j) := true;
repeat
  F_P := δ_0
  P := Longest path among the activities j with label(j) = true;
  for j ∈ P do
    F_P := F_P * F_j;
  F := F · F_P;
  for j ∈ P do
    label(j) = false;
until all activities are labeled false or length(P) = 0;
return (F)
```

Algorithmus 4: Lower bound of Spelde (based on pairwise disjoint paths).

implementation, we therefore only consider the K longest paths for a suitable choice of K. Due to the reduced number of paths, we cannot guarantee the bounding property anymore, and we therefore refer to this approach as a heuristic. However, for a sufficiently large number of paths it is very likely that we still obtain a stochastic upper bound, at least for large quantiles. In order to find the K longest paths we use a variant of the algorithm proposed by Dodin [6].

In practice, it is often the case that distributions of activity durations are not completely known. Usually, only information about the expected activity duration and its variance is available. In the context of Spelde's bounds, this situation can be easily managed, since it follows from the Central Limit Theorem that the sum of the activity durations of the path is approximately normally distributed. Then, the product of the resulting normal distribution functions approximates the distribution function of the makespan. This issue has been studied by many authors, see, e. g., [9, 2, 18]. Anklesaria and Drezner [2] as well as Sculli and Shum [18] 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. With respect to the applicability of the Central Limit Theorem we observed that 10 activities on a path suffice to obtain an excellent approximation of the path length distribution. Algorithm 5 displays an implementation of the heuristic ($Normal(m, v)$ denotes the distribution function of a normally distributed random variable with mean m and variance v). Within our computations we have generated at most $\frac{1}{3}|A|$ paths for each instance.

Input : A directed acyclic graph $D = (N, A)$; an expectation $E[\boldsymbol{p}_j]$ and a variance $V[\boldsymbol{p}_j]$ for each activity $j \in A$.

Output : A distribution function approximating the makespan.

$F := \delta_0$;
repeat
  $P :=$ longest currently unconsidered path;
  $m := \sum_{j \in P} E[\boldsymbol{p}_j]$;
  $v := \sum_{j \in P} V[\boldsymbol{p}_j]$;
  $F := F \cdot Normal(m, v)$;
until $Prob(P$ *is longest path*$)$ *is sufficiently small*;
return *(F);*

Algorithmus 5: Heuristic computation based on the Central Limit Theorem.

4 Computational Study

4.1 Computing Environment

Our experiments were conducted on a Sun Ultra 1 with 143 MHz clock pulse operating under Solaris 2.6 with 64 MB of memory. The code has been written in C++ and is compiled with the EGCS g++ compiler version 1.1.2 using the O3 optimization option. All reported CPU times to compute the stochastic bounds are averaged over three runs.

4.2 Benchmark Instances

We have tested the algorithms stated in Section 3 on two different test-sets which have been generated randomly. All instances are represented by activity-on-arc-networks and the activity durations are given by their mean and their variance. In order to obtain appropriate distribution functions for the activity durations, we generate gamma, normal, uniform, and triangle distributions based on these parameters. For gamma and normal distributions (which do not have a finite range) we have computed suitable border points according to the precision given by the used number of supporting points.

Most of the experiments have been performed on a test-set which is generated by an approach inspired by [5] (Test-set A). This test-set consists of 1600 instances in total, which are obtained by systematically modifying four different parameters. Two of these parameters clearly are the number of activities (n =300, 600, 900, and 1200) and the number of nodes in the activity-on-arc network ($\frac{n}{10}$, $\frac{n}{8}$, $\frac{n}{6}$, $\frac{n}{4}$, and $\frac{n}{2}$). Then, recall that activity-on-arc networks do not contain any isolated nodes. To ensure this fact we initially create a certain number of paths (3rd parameter) such that all nodes except $\bar{s}$ and $\bar{t}$ are contained in exactly one path. We thus have two parameters to control the "degree of parallelism" of the network. Long paths as well as many nodes in the network indicate a *low* parallelism. Correspondingly, short paths and few nodes in the network indicate a *high* parallelism of the activities. So far, the constructed networks only contain activities (arcs) that guarantee that no isolated nodes exist. All remaining activities that must be generated according to the fixed parameter n are inserted by randomly choosing their source and target node. The method to perform these random choices is taken from [5]. Expected activity durations are chosen uniformly between 0 and 100 (activities of duration 0 can be seen as dummy activities which are required to model arbitrary precedence constraints in activity-on-arc networks). The fourth parameter controls the variance of the activity durations. We have chosen all variances uniformly out of one of the intervals $[0, 10]$ or $[0, 100]$, respectively. Thus we have instances with low variances and others with high variances.

A second test-set (Test-set B) we have considered has been generated by ProGen [12], a widely accepted instance generator for resource-constrained project scheduling problems. Since we do not consider a model with limited resources, we ignore all parameters that control resource constraints. We have systematically modified the parameters that control network characteristics, i.e. the maximum number of immediate successors and predecessors of an activity (1 – 7) and the so-called *network complexity*, which reflects the average number of immediate successors of an activity (1.0 – 3.9). The expected activity durations vary between 0 and 50. The whole test-set contains 320 instances each of which consists of 150 activities. We have added a variance for each activity duration which taken randomly from the interval $[5, 15]$. For further details on this instance generator we refer to [12].

4.3 Representation of Distributions and Numerical Operations

The representation of the probability distributions is among the most important components of a code for computing stochastic bounds. The decision on the representation has to be made 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, we decided to represent each random variable by a piecewise linear approximation of its distribution function.

It turned out that the arrangement of supporting points is a crucial issue in such an approximation. Within our first experiments we assumed them to be equidistant on the abscissa. Unfortunately, the associate procedures for computing the product and the convolution operation are rather complicated in this case. In addition, the results did not behave numerically stable in a reasonable manner. We therefore occasionally allowed to split certain intervals and added another supporting point but this leads to complicated case distinctions and the results are still unsatisfactory with respect to numerical stability.

We have then implemented a representation, in which the supporting points are arranged equidistantly on the ordinate. Thus, we represent distribution functions by their inverse. We require that all distribution function are represented by the same number of supporting points. This allows a simple implementation of the product and the convolution procedure when compared to implementations based on the previously mentioned representations. In addition, these procedures turn out to compute reasonably stable results when a suitable number of supporting points is chosen. The product operation can be performed in linear time in the number of supporting points sp whereas the convolution requires quadratic running time in sp. We represent all distribution functions F by an array d of size sp such that $Prob(\boldsymbol{x} \leq d(k)) = \frac{k}{sp-1}$, $k = 0, \ldots, sp-1$ (an example with $sp = 11$ is depicted in Figure 2). This particularly means that $F(x) = 0$ for $x \leq d(0)$ and $F(x) = 1$ for $x \geq d(sp-1)$.

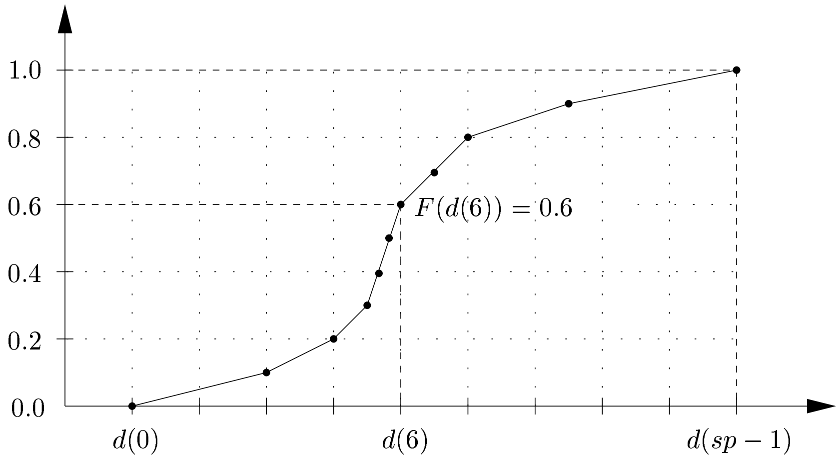

Figure 2: Distribution functions are represented by an array of supporting points that are arranged equidistantly on the ordinate.

The only numerical error which turns out to be of relevant magnitude occurs at the borders of distribution functions F obtained by the convolution operation. Here, a linear function is not an appropriate approximation of the exact distribution function. One solution to this problem would be to refine the resolution of supporting points in the intervals $[d(0), d(1)]$ and $[d(sp-2), d(sp-1)]$ (d denotes our representation of F) but this is not conform with our implementation which requires that all distribution functions have identical equidistant supporting points on the ordinate. Instead, we heuristically rearrange the border points $(d(0), 0)$ and $(d(sp-1), \frac{1}{sp-1})$ by slightly increasing $d(0)$ and decreasing $d(sp-1)$.

4.4 Computational Results

In order to measure the quality of the computed distribution functions, we have calculated approximate distribution functions by simulation. Allowing exhaustive computation time, we have simulated more than 1.5 million realizations for each instance. We compare distribution functions by means of 10 different quantiles p, namely $p \in \{0.01, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 0.95, 0.975, 0.99\}$. For each of these quantiles we measure the relative error (in percent) of the given distribution function from the simulated approximation (we always quote the average of these values). Experiments of three degrees of accuracy have been performed, namely for 50, 100, and 200 supporting points per distribution function.

We first analyze how the required computation times depend on the number of activities and the number of supporting points. For the instances of Test-set A, both dependencies are visualized in Figure 3 (a) and (b). They show a curve for each of the discussed procedures. We have plotted only one curve for the algorithms of Kleindorfer, since the running times of the upper and the lower bound are almost identical. Contrary to their theoretical running time, the computation time for Dodin's upper bound and Spelde's lower bound is minor when compared to those ones of Kleindorfer's bounds. For all network sizes and variations of the number of supporting points considered, the convolution operator consumed most of the computation time and therefore, network transformations (which cause larger theoretical running times) are not the bottleneck. Kleindorfer's procedures clearly require n convolutions. On the considered networks, Dodin's algorithm requires slightly fewer convolutions than Keindorfer's procedures, but for Spelde's lower bound suffice roughly $\frac{n}{2}$ convolutions. The reason here is that after having extracted paths with many activities from the network, there remains quite a number of paths with only one activity, for which no convolution need to be computed. Finally, since the heuristic procedure does not compute any convolutions, its computation time is negligible when compared with the bounding routines. Detailed results are provided in Table 1. Besides the average computation times we also include maximum computation times. The results for Test-set B coincide with the observations made for Test-set A. We therefore omit the details here.

We next consider the quality of the computed bounds compared with the crude simulation approach. As depicted in Figure 3 (c), for Test-set A the results are excellent. The average relative error of the computed bounds is less than 1% for most of the considered instances, number of supporting points and distributions. The relative error decreases slightly when the number of supporting points increases. The upper bound procedures show expected behavior: Dodin's approach outperforms Kleindorfer's algorithm. With respect to the lower bounds, we obtain better results for Spelde's algorithm based on disjoint paths. Detailed results are given in Table 2. The relative error also depends on the distribution type of the activity durations. When using uniform or triangle distributions, we obtain better results than for gamma or normal distributions, cf. Figure 3 (d). A similar behavior is observed when comparing instances with low and with high variances, respectively. For instances with variances in $[0, 10]$ we obtain an average relative error of 0.3% (max. 1.9%) while the corresponding values for variances in $[0, 100]$ is 0.9% (max. 5.7%). Note that the quality of the computed distributions does not seem to depend on the number of activities. We here observed a constant relative error.

The relative errors for Test-set B are still remarkably small, but they cannot compete with corresponding results for Test-set A, cf. Table 3. In particular, the average relative error varies 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 border 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%. Since the heuristic procedure does not require any computation of a convolution, we here obtain better results when compared to the bounding algorithms.

In summary, the used representation of distribution functions yields results of excellent quality within adequate computation times. This particularly holds for the results obtained by the heuristic procedure based on the Central Limit Theorem.

5 Concluding Remarks

Within this computational study we have empirically evaluated several procedures to bound or approximate the distribution function of the makespan of stochastic project networks that are based on network transformations. It turned out that the heuristic procedure based on the Central Limit Theorem provides excellent estimates at virtu-

Algorithm	#sp	Average CPU time (s)				Maximum CPU time (s)			
		300	600	900	1200	300	600	900	1200
Dodin	50	1.5	3.3	5.1	7.0	2.0	4.1	6.3	8.9
	100	6.0	12.9	19.9	26.9	7.9	16.0	24.2	33.1
	200	23.9	51.4	79.3	107.1	31.8	63.7	100.2	134.3
Kleindorfer (upper)	50	1.7	3.4	5.1	6.8	1.9	3.9	6.7	9.0
	100	6.8	13.6	20.5	27.5	8.5	15.8	24.1	37.9
	200	27.1	54.5	82.1	110.5	32.1	63.4	94.4	126.4
Kleindorfer (lower)	50	1.7	3.4	5.1	6.8	2.3	4.0	5.8	7.8
	100	6.7	13.6	20.9	27.8	8.1	16.1	26.7	36.5
	200	27.0	54.6	82.2	109.6	31.9	64.1	94.9	127.0
Disjoint paths	50	1.1	2.1	3.3	4.4	1.5	3.4	4.2	5.7
	100	4.2	8.5	12.8	17.3	5.6	11.5	16.8	24.8
	200	16.8	33.9	51.0	67.9	22.0	48.3	68.1	90.1
Spelde	50	0.1	0.3	0.4	0.6	0.1	0.3	0.5	0.8
	100	0.3	0.6	0.9	1.2	0.3	0.7	1.0	1.3
	200	0.7	1.3	2.0	2.7	0.7	1.4	2.1	2.8

Table 1: Test-set A: Average and maximum required CPU time of the algorithms, depending on the number of activities and the number of supporting points.

Algorithm	#sp	Average rel. error				Maximum rel. error			
		gam	nor	tri	uni	gam	nor	tri	uni
Dodin	50	0.9	0.8	0.5	0.5	4.6	4.4	3.9	4.1
	100	0.7	0.7	0.5	0.4	5.2	4.7	4.3	4.3
	200	0.6	0.6	0.5	0.4	5.0	4.5	4.4	4.3
Kleindorfer (upper)	50	1.0	1.0	0.6	0.6	5.1	4.8	4.4	4.6
	100	0.9	0.8	0.6	0.5	5.5	5.0	4.7	4.7
	200	0.8	0.7	0.6	0.5	6.3	5.7	4.8	4.7
Kleindorfer (lower)	50	0.6	0.6	0.7	0.5	9.5	9.9	9.2	7.8
	100	0.5	0.5	0.4	0.4	8.1	8.5	7.8	7.2
	200	0.4	0.4	0.3	0.3	7.6	7.9	7.4	7.2
Disjoint paths	50	0.6	0.6	0.6	0.4	6.5	7.0	6.1	4.7
	100	0.4	0.4	0.3	0.3	4.5	4.6	4.2	3.7
	200	0.3	0.3	0.2	0.2	3.6	3.8	3.4	3.3
Spelde	50	0.6	0.6	0.7	0.6	5.0	5.1	5.3	5.0
	100	0.7	0.7	0.8	0.7	6.1	6.1	6.3	6.1
	200	0.7	0.7	0.8	0.7	6.2	6.3	6.4	6.2

Table 2: Test-set A: Average and relative error from simulated distribution, depending on the type of distribution and the number of supporting points.

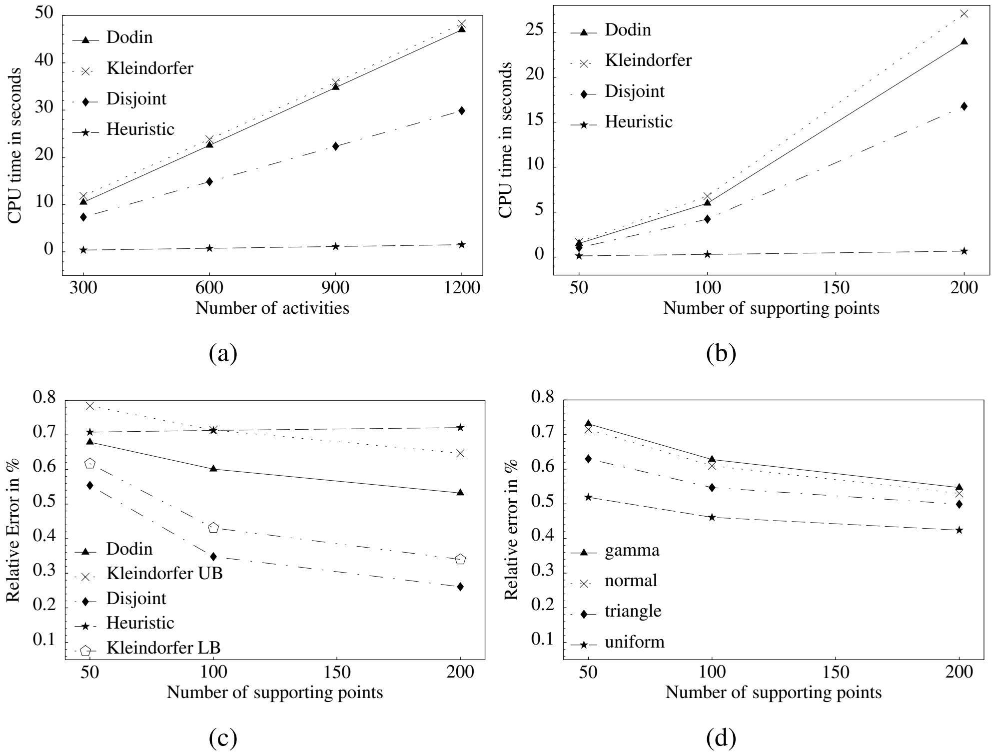

Figure 3: Plots (a) and (b) show the running time of the different algorithms depending on the number of activities and the number of supporting points. Plot (c) visualizes how the average relative error decreases when increasing the number of supporting points. Plot (d) shows the average relative error depending on the used type of distribution.

ally no computational expense. The approach is therefore a remarkable alternative to (computationally costly) simulation techniques which are mostly used in practice. The upper and lower bounding procedures consume more time but they also allow an efficient computation of distribution functions that "sandwich" the exact distribution function. The key to an efficient implementation is a suitable representation of the distributions and a numerically stable implementation of the product and convolution operations.

Future work will include the analysis of bounds that cover stochastic dependencies as provided by Meilijson and Nadas [15].

References

[1] V. G. Adlakha and V. G. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR*, 27:272–296, 1989.

[2] K. P. Anklesaria and Z. Drezner. A multivariate approach to estimating the completion time for PERT networks. *Journal of the Operational Research Society*, 40:811–815, 1986.

[3] W. W. Bein, J. Kamburowski, and M. F. M. Stallmann. Optimal reduction of two-terminal directed acyclic graphs. *SIAM Journal on Computing*, 21:1112–1129, 1992.

[4] C. G. Bigelow. Bibliography on project planning and control by network analysis. *Operations Research*, 10:728–731, 1962.

[5] E. Demeulemeester, B. Dodin, and W. Herroelen. A random activity network generator. *Operations Research*, 41:972–980, 1993.

[6] B. Dodin. Determining the k most critical paths in pert networks. *Operations Research*, 32:859–877, 1984.

[7] B. Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33:862–881, 1985.

[8] J. D. Esaray, F. Proschan, and D. W. Walkup. Association of random variables, with applications. *Ann. Math. Statist.*, 38:1466–1474, 1967.

[9] D. Golenko. *Statistische Methoden der Netzplantechnik*. Teubner Verlagsgesellschaft, 1972.

[10] J. N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18:139–147, 1988.

[11] G. B. Kleindorfer. Bounding distributions for a stochastic acyclic network. *Operations Research*, 19:1586–1601, 1971.

[12] R. Kolisch and A. Sprecher. PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 96:205–216, 1996.

[13] S. Lerda-Olberg. Bibliography on network-based project planning and control techniques. *Operations Research*, 14:925–931, 1966.

[14] J. J. Martin. Distribution of the time through a directed acyclic network. *Operations Research*, 13:46–66, 1965.

[15] I. Meilijson and A. Nadas. Convex majorization with an application to the length of critical paths. *J. Appl. Prob.*, 16:671–677, 1979.

[16] R. H. Möhring and R. Müller. A combinatorial approach to bound the distribution function of the makespan in stochastic project networks. Technical Report 610/1998, Technical University of Berlin, Department of Mathematics, Germany, 1998.

[17] P. Robillard and M. Trahan. Expected completion time in PERT networks. *Operations Research*, 24:177–182, 1976.

[18] D. Sculli and Y. W. Shum. An approximate solution to the PERT problem. *Computers and Mathematics with Applications*, 21:1–7, 1991.

[19] M. Shaked and J. G. Shanthikumar. *Stochastic orders and their applications*. Academic Press, 1994.

[20] A. W. Shogan. Bounding distributions for a stochastic PERT network. *Networks*, 7:359–381, 1977.

[21] H. Soroush. Risk taking in stochastic PERT networks. *European Journal of Operational Research*, 67:221–241, 1993.

[22] H. G. Spelde. *Stochastische Netzpläne und ihre Anwendung im Baubetrieb*. PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 1976.

[23] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series-parallel digraphs. *SIAM Journal on Computing*, 11:298–314, 1982.

Algorithm	#sp	Average rel. error				Maximum rel. error			
		gam	nor	tri	uni	gam	nor	tri	uni
Dodin	50	3.7	3.2	2.0	2.1	10.5	7.9	6.3	6.9
	100	3.4	2.8	2.2	1.9	12.3	9.3	7.5	7.5
	200	2.9	2.4	2.2	1.7	13.0	9.7	7.7	7.6
Kleindorfer (upper)	50	3.7	3.2	2.0	2.1	10.5	7.9	6.3	6.9
	100	3.4	2.8	2.2	1.9	12.3	9.3	7.5	7.5
	200	2.9	2.4	2.2	1.7	13.0	9.7	7.7	7.6
Kleindorfer (lower)	50	3.4	3.4	3.2	2.7	16.5	17.9	13.1	10.3
	100	2.7	2.7	2.4	2.4	12.0	13.2	10.2	9.4
	200	2.3	2.3	2.0	2.2	9.3	9.8	9.0	8.6
Disjoint paths	50	3.4	3.3	3.2	2.7	15.4	16.8	12.3	9.9
	100	2.7	2.6	2.4	2.3	10.1	11.3	9.2	8.5
	200	2.3	2.2	2.0	2.2	8.2	8.4	7.8	7.8
Spelde	50	1.6	1.6	2.1	1.6	8.0	7.7	8.4	7.7
	100	1.9	1.9	2.5	2.0	9.2	9.0	9.7	9.1
	200	1.9	1.9	2.6	2.0	9.4	9.3	9.9	9.4

Table 3: Test-set B: Average and maximum relative error from simulated distribution, depending on the type of distribution and the number of supporting points.

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

669/2000 *Michael Naatz:* A Note On a Question of C. D. Savage

667/2000 *Sándor P. Fekete and Henk Meijer:* On Geometric Maximum Weight Cliques

666/2000 *Sándor P. Fekete, Joseph S. B. Mitchell, and Karin Weinbrecht:* On the Continuous Weber and k-Median Problems

664/2000 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* A Note on Scheduling Problems with Irregular Starting Time Costs

661/2000 *Frederik Stork and Marc Uetz:* Resource-Constrained Project Scheduling: From a Lagrangian Relaxation to Competitive Solutions

658/1999 *Olaf Jahn, Rolf H. Möhring, and Andreas S. Schulz:* Optimal Routing of Traffic Flows with Length Restrictions in Networks with Congestion

655/1999 *Michel X. Goemans and Martin Skutella:* Cooperative facility location games

654/1999 *Michel X. Goemans, Maurice Queyranne, Andreas S. Schulz, Martin Skutella, and Yaoguang Wang:* Single Machine Scheduling with Release Dates

653/1999 *Andreas S. Schulz and Martin Skutella:* Scheduling unrelated machines by randomized rounding

646/1999 *Rolf H. Möhring, Martin Skutella, and Frederik Stork:* Forcing Relations for AND/OR Precedence Constraints

640/1999 *Foto Afrati, Evripidis Bampis, Chandra Chekuri, David Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Cliff Stein, and Maxim Sviridenko:* Approximation Schemes for Minimizing Average Weighted Completion Time with Release Dates

639/1999 *Andreas S. Schulz and Martin Skutella:* The Power of α-Points in Preemptive Single Machine Scheduling

634/1999 *Karsten Weihe, Ulrik Brandes, Annegret Liebers, Matthias Müller–Hannemann, Dorothea Wagner and Thomas Willhalm:* Empirical Design of Geometric Algorithms

633/1999 *Matthias Müller–Hannemann and Karsten Weihe:* On the Discrete Core of Quadrilateral Mesh Refinement

632/1999 *Matthias Müller–Hannemann:* Shelling Hexahedral Complexes for Mesh Generation in CAD

631/1999 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Insights from a Computational Study

629/1999 *Martin Skutella:* Convex Quadratic Programming Relaxations for Network Scheduling Problems

628/1999 *Martin Skutella and Gerhard J. Woeginger:* A PTAS for minimizing the total weighted completion time on identical parallel machines

627/1998 *Jens Gustedt:* Specifying Characteristics of Digital Filters with FilterPro

620/1998 *Rolf H. Möhring, Andreas S. Schulz, Frederik Stork, and Marc Uetz:* Resource Constrained Project Scheduling: Computing Lower Bounds by Solving Minimum Cut Problems

619/1998 *Rolf H. Möhring, Martin Oellrich, and Andreas S. Schulz:* Efficient Algorithms for the Minimum-Cost Embedding of Reliable Virtual Private Networks into Telecommunication Networks

618/1998 *Friedrich Eisenbrand and Andreas S. Schulz:* Bounds on the Chvátal Rank of Polytopes in the 0/1-Cube

617/1998 *Andreas S. Schulz and Robert Weismantel:* An Oracle-Polynomial Time Augmentation Algorithm for Integer Proramming

616/1998 *Alexander Bockmayr, Friedrich Eisenbrand, Mark Hartmann, and Andreas S. Schulz:* On the Chvátal Rank of Polytopes in the 0/1 Cube

615/1998 *Ekkehard Köhler and Matthias Kriesell:* Edge-Dominating Trails in AT-free Graphs

613/1998 *Frederik Stork:* A branch and bound algorithm for minimizing expected makespan in stochastic project networks with resource constraints

612/1998 *Rolf H. Möhring and Frederik Stork:* Linear preselective policies for stochastic project scheduling

609/1998 *Arfst Ludwig, Rolf H. Möhring, and Frederik Stork:* A computational study on bounding the makespan distribution in stochastic project networks

605/1998 *Friedrich Eisenbrand:* A Note on the Membership Problem for the Elementary Closure of a Polyhedron

596/1998 *Andreas Fest, Rolf H. Möhring, Frederik Stork, and Marc Uetz:* Resource Constrained Project Scheduling with Time Windows: A Branching Scheme Based on Dynamic Release Dates

595/1998 *Rolf H. Möhring Andreas S. Schulz, and Marc Uetz:* Approximation in Stochastic Scheduling: The Power of LP-based Priority Policies

591/1998 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Towards a Flexible Software Design

590/1998 *Stefan Felsner and Jens Gustedt and Michel Morvan:* Interval Reductions and Extensions of Orders: Bijections to Chains in Lattices

584/1998 *Alix Munier, Maurice Queyranne, and Andreas S. Schulz:* Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems

577/1998 *Martin Skutella:* Semidefinite Relaxations for Parallel Machine Scheduling

Reports may be requested from:

Hannelore Vogt-Möller
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: moeller@math.TU-Berlin.DE

Reports are also available in various formats from

`http://www.math.tu-berlin.de/coga/publications/techreports/`

and via anonymous ftp as

`ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-`*number*`-`*year*`.ps`