A Stochastic Branch–and–Bound Approach to Activity Crashing in Project Management

W.J. Gutjahr / *Univ. of Vienna, Dept. of Statistics, OR and Comp. Sci., A–1010 Vienna, walter.gutjahr@univie.ac.at*
C. Strauss / *Univ. of Vienna, Dept. of Management Science, A–1210 Vienna, strauss@pom.bwl.univie.ac.at*
E. Wagner / *Andersen Consulting, A–1030 Vienna, eric.w.wagner@ac.com*

Many applications such as project scheduling, workflow modeling, or business process re-engineering incorporate the common idea that a product, task, or service consisting of interdependent time–related activities should be produced or performed within given time limits. In real–life applications, certain measures like the use of additional man-power, the assignment of highly–skilled personnel to specific jobs, or the substitution of equipment are often considered as means of increasing the probability of meeting a due date and thus avoiding penalty costs. This paper investigates the problem of selecting, from a set of possible measures of this kind, the combination of measures that is the most cost–efficient. Assuming stochastic activity durations, the computation of the optimal combination of measures may be very expensive in terms of runtime.

In this article, we introduce a powerful stochastic optimization approach to determine a set of efficient measures that crash selected activities in a stochastic activity network. Our approach modifies the conventional Stochastic Branch–and–Bound, using a heuristic — instead of exact methods — to solve the deterministic subproblem. This modification spares computational time, and by doing so provides an appropriate method for solving various related applications of combinatorial stochastic optimization. A comparative computational study shows that our approach not only outperforms standard techniques, but also definitely improves conventional Stochastic Branch–and–Bound.

Subject classifications: Activity Crashing, Stochastic Branch–and–Bound, Project Management.
Other key words: Combinatorial Optimization, Importance Sampling, Project Compression, Simulation, Stochastic Optimization, Stochastic Scheduling, Stochastic Discrete Time–Cost Problem.

Most projects, production activities, and service activities in a competitive environment must fulfill global time requirements (e.g., penalties must be paid if a project is not completed on time, production rates decline if some parts take longer than expected to be finished) and fit into a rough framework. This framework consists of customer requirements that demand short production, lead and service times, as well as of the need to use resources economically.

The basic problem structure that a project manager faces often takes the form of a *time–cost tradeoff*: finishing the project on time necessitates crashing certain activities by exploiting extra resources (using additional manpower, assigning highly–skilled personnel to specific jobs, improving machines or equipment, subcontracting of certain tasks, etc.; see, e.g., [11], [12]). On the other hand, such crashing measures increase the costs of the project.

In practice, the problem is even more complicated as the durations of activities usually follow *probability distributions*; they are not known in detail in advance. Therefore, whether certain due dates can be met or not cannot be predicted with certainty; one can only estimate a *probability* for terminating a project before a given date.

In activity planning methodology, the well–established CPM– or PERT–based approaches deal with the time–cost problem in detail (see, [5], pp. 237). However, these approaches usually assume that resource–duration alternatives for activities can be expressed as a continuous duration–cost function, which is unrealistic because of the discrete nature of most resources (see, e.g., [8], [4]).

In certain situations, activities can only be crashed or left un-crashed, thus making zero-one decisions necessary. The selection of those activities which need to be crashed in order to achieve the most cost–efficient solution involves two interlinked problems: (1) estimating project completion time by using stochastic activity durations, and (2) solving a hard combinatorial optimization problem by determining the efficient measures.

Section 1 explains the Stochastic Discrete Time–Cost Problem (SDTCP) in formal terms. Section 2 presents the Stochastic Branch–and–Bound technique (developed by Norkin, Ermoliev, Pflug, and Ruszczynski [6, 7]), and its application to the given problem. Stochastic Branch–and–Bound requires the determination of cost values estimates by *sampling*. We apply two sampling techniques, namely a straightforward one and a more sophisticated one known as Importance Sampling, which is briefly outlined in section 3. In section 4, we introduce a computationally more efficient modification of the Stochastic Branch–and–Bound method. Finally, in section 5, the quality of the results produced by our new approach is evaluated on the basis of a computational study: a comparison with two general approaches ("Brute Force Simulation" and the application–specific approach "Complete Enumeration combined with modified PNET"), as well as with two conventional Stochastic Branch–and–Bound versions is presented. Our approach yields surprisingly good results within reasonable runtime limits. Section 6 summarizes these results, suggests other applications of the new solution approach presented in this article, and provides a brief outlook on further research activities.

1 The Stochastic Discrete Time–Cost Problem

This section provides a formally precise definition for the issue being investigated in the present paper. Let us represent the given network as an activity–on–arcs network, the nodes of which are numbered by $1, \ldots, n$. As a general convention, node 1 will always correspond to the unique initial event ("project start"), and node n will always correspond to the unique terminal event ("project termination"). We exclude multiple arcs (note that by the introduction of dummy activities, it is always possible to replace multiple arcs by single arcs), so each arc (activity) can be represented in the form (i, j), where i and j are nodes (events). $D(i, j)$ denotes the time required for activity (i, j). Since $D(i, j)$ is considered as not known with certainty before the termination of (i, j), it is modeled by a random variable. As in classic PERT, we assume that the distribution of each $D(i, j)$ can be estimated, and that the random variables $D(i, j)$ are independent.

Next, let us assume that penalty costs occur if the project is terminated after its pre–specified due date, and that they can be described by a *loss function* Λ, where $\Lambda(t)$ indicates the loss occurring if the project is terminated at time t. The start time of the project will always be set equal to zero. For the purpose of estimating Λ, it is convenient to assume that Λ is a step function, i.e.,

$$\Lambda(t) = \Lambda_k \quad \text{if } t_k < t \leq t_{k+1} \quad (k = 0, \ldots, K-1), \quad \Lambda(t) = \Lambda_K \quad \text{if } t > t_K,$$

where $0 = \Lambda_0 < \Lambda_1 < \ldots < \Lambda_K$ and $0 = t_0 < t_1 < \ldots < t_K$ (K denotes the number of steps of the function Λ). This special form of the loss function implies that no penalty occurs if the project is completed on time, i.e., before the (first) due date t_1. The restriction to step functions does not limit the applicability, as each non–decreasing loss function Λ with $\Lambda(0) = 0$ can be approximated to any desired degree of accuracy by step functions of the type above.

Now, let us suppose that the distributions of the random variables $D(i, j)$ might be changed by certain (crashing) *measures*, indexed by $1, \ldots, M$. Typically, measure m reduces the expected time required for one or several activities by a certain amount. Thus, the duration of (i, j) becomes dependent of the vector $x = (x_1, \ldots, x_M)$, where x_m is the indicator variable denoting whether measure m is set or not: $x_m = 1$ if measure m is chosen, and $x_m = 0$ else. $D(i, j, x)$ will denote the time required for activity (i, j) on the condition that a measure combination described by the vector x has been chosen. It is assumed that each measure m causes additional costs of c_m currency units.

For fixed x, the project completion time $t(x)$ can be computed on the basis of the values $D(i, j, x)$ by using of the usual Critical Path Method (CPM) approach. Since $t(x)$ depends on the numbers $D(i, j, x)$, it is also a random variable, as is the loss $\Lambda(t(x))$ by late termination of the project. The overall loss results from the sum of $\Lambda(t(x))$ and of the costs $\sum_m c_m x_m$ for the chosen measures. The aim is to minimize the *expected* overall loss, i.e., to solve the following stochastic optimization problem, which we call the *stochastic discrete time–cost problem* (SDTCP):

$$\begin{aligned} \text{Minimize} \quad & F(x) = \mathrm{E}(\Lambda(t(x))) + \sum_{m=1}^{M} c_m x_m \\ \text{s. t.} \quad & x_m \in \{0,1\} \ (m = 1, \ldots, M), \end{aligned} \tag{1}$$

where $\mathbf{E}$ denotes the mathematical expectation.

The SDTCP reduces to the *deterministic discrete time–cost problem* (DDTCP) in the special event that each (i, j, x) has a distribution of $D(i, j, x)$ that is a point mass in some $d(i, j, x)$. Techniques such as Dynamic Programming have been applied to produce practical solutions to problems similar to the DDTCP (see [8], [4]). Also (ordinary) Branch–and–Bound can be used for solving the DDTCP.

Our solution approach for the SDTCP will be based on *Stochastic Branch–and–Bound*, a specific technique for the solution of combinatorial stochastic optimization problems. In the next section, this technique will be outlined shortly, following its presentation in [2].

2 Stochastic Branch–and–Bound

2.1 General Description of the Algorithm

Let us start by representing a general combinatorial stochastic optimization problem in the form

$$\text{Minimize } F(x) = \mathrm{E}(f(x, \omega)) \quad \text{for } x \in \mathcal{X}, \tag{2}$$

where $\mathcal{X}$ is a finite set of possible decisions or actions and $\omega \in \Omega$ denotes the influence of randomness, formally described by a probability space (Ω, Σ, P). The Stochastic Branch–and–Bound method for solving (2), as developed by Norkin, Ermoliev, and Ruszczynski in [6], consists of

- partitioning the feasible set $\mathcal{X}$ into smaller subsets, and
- estimating lower bounds of the objective function $F(x)$ within the subsets.

As in most implementations of ordinary (deterministic) Branch–and–Bound, the well–known *lowest-bound rule* is applied as the selection rule for the "branch" part of the algorithm: at each step of the algorithm, the subset with minimum estimated lower bound is selected for a further partition into smaller subsets. This approach ensures that "promising" subsets are investigated in more detail. Contrary to ordinary Branch-and-Bound, this approach involves no definite step in which the algorithm terminates with the exact solution; instead, the computation can be aborted according to some stopping criterion selected by the user (e.g., time window or number of iterations), yielding an approximate solution for (2).

For a more explicit description, let us denote by $\mathcal{X}^p$ $(p = 1, 2, \ldots)$ the current subsets into which the original set $\mathcal{X}$ has been divided. In total, the sets $\mathcal{X}^p$ form a partition $\mathcal{P}$ of $\mathcal{X}$. Correspondingly, the original problem (2) is divided into subproblems

$$\text{Minimize } F(x) = \mathrm{E}(f(x, \omega)) \quad \text{for } x \in \mathcal{X}^p,$$

where $\mathcal{X}^p \in \mathcal{P}$. Let us set

$$F^*(\mathcal{X}^p) = \min_{x \in \mathcal{X}^p} F(x).$$

The following assumptions are made:

1. There is a *lower bound function* L mapping the set of subsets of $\mathcal{X}$ into the set $I\!R$ of real numbers, such that for all $\mathcal{X}^p \in \mathcal{P}$, $L(\mathcal{X}^p) \leq F^*(\mathcal{X}^p)$, and $L(\mathcal{X}^p) = F^*(\mathcal{X}^p)$ if $\mathcal{X}^p$ is a singleton set.

2. There exists a sequence $\xi^l(\mathcal{X}^p)$, $l = 1, 2, \ldots$ of *random estimates* of $L(\mathcal{X}^p)$, such that with probability one
$$\xi^l(\mathcal{X}^p) \to L(\mathcal{X}^p) \text{ as } l \to \infty.$$
Note that the estimates $\xi^l(\mathcal{X}^p)$ are random variables, while the bounds $L(\mathcal{X}^p)$ are deterministic quantities.

With these assumptions and notations, we are now able to present the Stochastic Branch–and–Bound algorithm. (In the original form of the algorithm, Norkin et al. [6] use *upper bounds* additionally to lower bounds. We have slightly simplified the algorithm by working with lower bounds only.) In the formulation below, subsequences of the sequences of estimates $\xi^l(\mathcal{X}^p)$ will be used. These subsequences will be indexed by $l_r = l_r(\mathcal{X}^p)$ $(r = 0, 1, \ldots)$; this notation makes clear that the choice of the current estimate is made dependent on the current subset $\mathcal{X}^p$.

Stochastic Branch-and-Bound:

Set $\mathcal{P}_0 := \{\mathcal{X}\}$, and $\xi_0(\mathcal{X}) := \xi^{l_0(\mathcal{X})}(\mathcal{X})$;
set $r := 0$;
until stopcriterion satisfied {
    select the lowest-bound subset $\mathcal{Y}^r \in \text{argmin}\,\{\xi_r(\mathcal{X}^p) \mid \mathcal{X}^p \in \mathcal{P}_r\}$;
    select an approximate solution $x^r \in \mathcal{Y}^r$;

if ($\mathcal{Y}^r$ is a singleton)

set $\mathcal{P}_{r+1} := \mathcal{P}_r$;

else {

construct a partition of the lowest-bound subset:

$\mathcal{P}'_r(\mathcal{Y}^r) = \{\mathcal{Y}^r_i \mid i = 1, \ldots, n_r\}$;

construct the new full partition:

$\mathcal{P}_{r+1} = (\mathcal{P}_r \setminus \{\mathcal{Y}^r\}) \cup \mathcal{P}'_r(\mathcal{Y}^r)$;

}

set $r := r + 1$;

for (all subsets $\mathcal{X}^p \in \mathcal{P}_r$)

determine estimates $\xi_r(\mathcal{X}^p) = \xi^{l_r(\mathcal{X}^p)}(\mathcal{X}^p)$;

}

Norkin, Ermoliev, and Ruszczynski [6] proved the following convergence result: Suppose the indices $l_r(\mathcal{X}^p)$ are chosen in such a way that whenever a subset $\mathcal{X}'$ is element of $\mathcal{P}_r$ for infinitely many r, then $\lim_{r\to\infty} l_r(\mathcal{X}') = \infty$. Then with probability one there exists a number r_0 such that for all $r \geq r_0$, the lowest-bound subsets $\mathcal{Y}^r$ are singletons and contain optimal solutions only.

A generalized version of the Stochastic Branch–and–Bound algorithm, one which also covers continuous stochastic optimization problems, has been developed by Norkin, Pflug and Ruszczynski in [7]. This article also generalizes the above–mentioned convergence result.

Remark. In [6] and [7], the term "Stochastic Branch–and–Bound" has been chosen, although - strictly speaking - the word "bound" no longer refers to the same thing as in the well–known (deterministic) Branch–and–Bound algorithm: in deterministic Branch–and–Bound, certain branches are *cut* as soon as the corresponding lower bounds are higher than a current upper bound for the optimal cost value, and the evolution of the tree is "bounded" in this way. In contrast, the basic version of Stochastic Branch–and–Bound does not cut any branch forever, even a high lower bound assigned to some node is re–evaluated in the subsequent iteration steps, and it is therefore still possible that the corresponding branch will be expanded at some later time. What is "bounded" in Stochastic Branch–and–Bound is not the evolution of the tree, but rather the amount of simulation time spent for certain feasible solutions: while brute–force–simulation provides the same simulation effort for all solutions, Stochastic Branch–and–Bound restricts the simulation sample size in branches that have turned out as less promising. In spite of the fact that the analogy to deterministic Branch–and–Bound is not perfect, we have decided to keep the notion "Stochastic Branch–and–Bound" for our version of the algorithm, since it allows direct reference to the pioneering publications [6] and [7]. Of course, also in Stochastic Branch–and–Bound, branches *can* be cut, if it is possible to obtain exact (i.e., deterministic) lower and upper bounds. In the case of our SDTCP investigation, this indeed was possible, since the beta-distributed activity durations had strict, deterministic minimum and maximum values. We experimented with exploiting these deterministic bounds for cutting branches, but it turned out that they were too week to improve the performance of the algorithm.

2.2 Application to the SDTCP

Now we specify the Stochastic Branch–and–Bound algorithm for the intended application to the SDTCP described in section 1. Here,

$$f(x, \omega) = \Lambda(t(x)) + \sum_{m=1}^{M} c_m x_m,$$

ω being implicitly contained in $t(x)$, and $\mathcal{X} = \{0, 1\}^M$, the set of all possible measure combinations x. For applying Stochastic Branch–and–Bound, we have to indicate how the partitioning step is to be performed, and how estimates for lower bounds are determined.

1. **Partitioning**

   Partitioning is very simple in the case of the SDTCP. It is done in such a way that all subsets $\mathcal{X}^p$ occurring during the branch–and–bound process are of the form

   $$\mathcal{X}_{k_1 \ldots k_s} = \{x \in \{0, 1\}^M \mid x_1 = k_1, \ldots, x_s = k_s\}$$

   for some s, $0 \leq s \leq M$. The set $\mathcal{X}_{k_1 \ldots k_s}$ is partitioned into the two subsets $\mathcal{X}_{k_1 \ldots k_s 0}$ and $\mathcal{X}_{k_1 \ldots k_s 1}$, i.e., by the distinction whether measure $s + 1$ is chosen or not.

2. **Bound Estimation**

   For obtaining an estimate of a lower bound $L(\mathcal{X}^p)$ for $F(x) = \mathrm{E}(f(x, \omega))$, $x \in \mathcal{X}^p$, one may use interchange of minimization and expectation operator, as suggested in [6]: one can immediately verify that

   $$F^*(\mathcal{X}^p) = \min_{x \in \mathcal{X}^p} \mathbf{E}(f(x, \omega)) \geq \mathbf{E}(\min_{x \in \mathcal{X}^p} f(x, \omega)).$$

   Thus, it is possible to choose

   $$L(\mathcal{X}^p) = \mathrm{E}(\min_{x \in \mathcal{X}^p} f(x, \omega)).$$

   The last expression, however, can be estimated by sampling: draw N values $\omega_1, \ldots, \omega_N$ independently, according to distribution P. For fixed x, a particular ω_ν gives rise to a specific vector of activity durations $D(i,j,x)$ ((i,j) arc of the network), which allows the determination of $f(x,\omega_\nu)$. Now, consider x as variable, and determine, for each ω_ν,

$$\min_{x \in \mathcal{X}^p} f(x, \omega_\nu) \tag{3}$$

by solving the corresponding DDTCP. Then

$$\frac{1}{N} \sum_{\nu=1}^{N} \min_{x \in \mathcal{X}^p} f(x, \omega_\nu) \tag{4}$$

is an unbiased estimate of $L(\mathcal{X}^p)$ as defined above. Obviously, the variance of this estimate can be reduced to any desired degree by increasing the sample size N, such that condition 2 in section 2.1 on the lower-bound estimates is satisfied.

The reader should note that in order to produce the solution of the SDTCP according to our approach, corresponding DDTCPs must be solved as subproblems. This can be done by using Complete Enumeration or more sophisticated techniques, e.g., Dynamic Programming or ordinary Branch–and–Bound (cf. the remarks at the end of section 1). However, it turns out that for large problem instances (where these techniques outperform Complete Enumeration), *their* runtime too is already so high that, in combination with Stochastic Branch–and–Bound, the overall runtime becomes prohibitive. For this reason, we have resorted to a heuristic for providing an *approximate* solution for the DDTCP (see section 4).

3 Importance Sampling

The estimate (4) is based on straightforward simulation. In particular, as soon as a subset $\mathcal{Y}^r$ degenerates to a singleton $\{x\}$, an estimate for the corresponding function value $F(x)$ is obtained by the estimator

$$\frac{1}{N} \sum_{\nu=1}^{N} f(x, \omega_\nu). \tag{5}$$

However, in cases where the late termination of the given project is not very probable (although it would lead to high penalties), *Rare Event Simulation* methods should be applied, rather than simple Monte Carlo; otherwise, the required low variance of the estimate could only be achieved by a *very* large sample size. We have experimented with *Importance Sampling* as a variance–reducing technique of Rare Event Simulation (see, e.g., [10] or [3]). The key idea is to shift the probability distribution of the random variable, whose expected value will be estimated, towards the region of interest (in our case: the area where the project is late; see Fig. 1), to sample from the shifted distribution, and to compensate the shift afterwards by adjusting

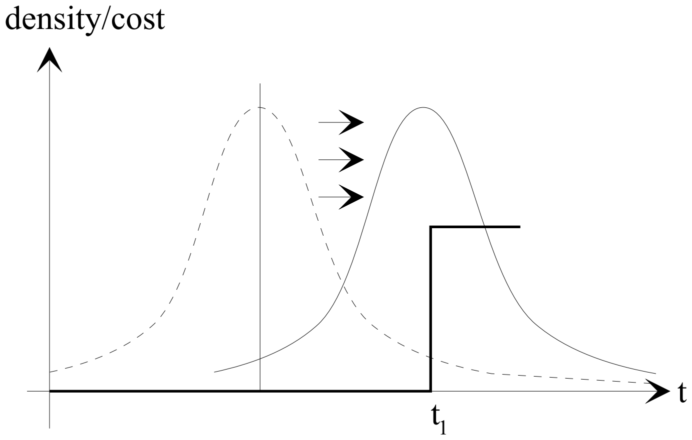

Figure 1: "Probability Shift" in Importance Sampling

suitable *weights* to the simulation results. These weights, usually called *likelihood ratios*, are simply the ratios between the densities of the original distribution and those of the shifted distribution.

Of course, the distribution of the random variable $f(x, \omega)$ cannot be controlled *directly* in our problem context, however it can be influenced by shifting the distributions of the single activity durations $D(i, j, x)$. Multiplying each $D(i, j, x)$ by a certain fixed factor c results in the multiplication of the project termination time t by c. Because the random variables $D(i, j, x)$ are regarded as being independent, the total likelihood ratio is then the product of the likelihood ratios of each $D(i, j, x)$. This yields the following procedure:

1. Shift the densities $g_{i,j,x}(u)$ of the random variables $D(i, j, x)$ to suitable *sampling densities* $\bar{g}_{i,j,x}(u)$.

2. For each arc (i, j) of the network, draw N independent sample values $\bar{D}^{(\nu)}(i, j, x)$ of activity durations according to the sampling densities $\bar{g}_{i,j,x}(u)$ $(\nu = 1, \ldots, N)$.

3. For the given measure combination x and for each ν, compute the resulting project time $\bar{t}^{(\nu)}(x)$ from the values $\bar{D}^{(\nu)}(i, j, x)$.

4. As an estimate of $\mathrm{E}(\Lambda(t(x)))$, take the arithmetic mean of weighted penalties,

$$\frac{1}{N} \sum_{\nu=1}^{N} \left(\prod_{(i,j)} \frac{g_{i,j,x}(\bar{D}^{(\nu)}(i, j, x))}{\bar{g}_{i,j,x}(\bar{D}^{(\nu)}(i, j, x))} \right) \Lambda(\bar{t}^{(\nu)}(x)). \tag{6}$$

It is not difficult to show that (6) is an unbiased estimate of $\mathrm{E}(\Lambda(t(x)))$, provided that $\bar{g}_{i,j,x}(u) > 0$ for all u where $g_{i,j,x}(u) > 0$.

The probability shift performed in the procedure above does not guarantee that the variance of the estimate decreases. In order to achieve such a variance reduction, the size of the shift must be chosen appropriately. For special cases, the optimal shift (i.e., the shift achieving maximal variance reduction) can be computed explicitly. Without proof, we state the following result: suppose a network is given in which all activities are switched in series, the durations $D(i, j, x)$ are normally distributed, the sum of their expected values is $d < t_1$, and the loss function Λ has only one step at t_1, i.e., $K = 1$. Furthermore, assume that the standard deviations of the variables $D(i, j, x)$ are small compared to the difference $t_1 - d$. Given these conditions, the optimal choice of the multiplication factor c is t_1/d. In other words, the maximal variance reduction is obtained by shifting the expected value of $t(x)$ from d to the due date t_1. Any value of c between 1 and t_1/d achieves a smaller variance reduction, but the variance still decreases in this case. On the other hand, if Λ has several steps at $t_1, \ldots, t_K$ $(K > 1)$, then a value of c larger than t_1/d is optimal — but the choice $c = t_1/d$ still guarantees an improvement in comparison to straightforward Monte Carlo simulation. For this reason, we have decided to always choose c in our experiments in such a manner that the estimated project time is shifted to the first step t_1 of the loss function.

4 Heuristic Solution of the Deterministic Subproblem

We have stated in subsection 2.2 that the application of the Stochastic Branch–and–Bound technique to the SDTCP necessitates solving special DDTCPs as subproblems (see (3)). Although these deterministic optimization problems no longer have the entire set $\mathcal{X}$ as their feasible set, but rather subsets $\mathcal{X}^p = \mathcal{X}_{k_1 \ldots k_s}$, their structure is that of the DDTCP. In general, the set $\mathcal{X}^p$ remains too large for the application of an exact optimization technique. Therefore, we have decided to apply a *Local Search* solution procedure (see [9], ch. 19) for providing approximate solutions for the DDTCPs.

The replacement of an exact solution to the deterministic subproblem by a heuristic one proved to be very advantageous in our experiments (Section 5 provides further details). Thus, it is reasonable to propose this modification of Stochastic Branch–and–Bound as a new, efficient general approach to combinatorial stochastic optimization.

Local Search requires the definition of a *neighborhood structure* (cf. [9], pp. 7-8) on the feasible set. For the DDTCP, a neighborhood may be defined in a straightforward way: for a given measure combination described by the vector $x = (x_1, \ldots, x_m)$, the set $N(x)$ of neighbors of x consists of all measure combinations x' obtained from x by swapping a single measure m (i.e., selecting measure m if it is not selected in x, and dropping it if it is selected in x):

$$N(x) = \{x' = (x_1, \ldots, x_{m-1}, 1 - x_m, x_{m+1}, \ldots, x_M) \mid 1 \leq m \leq M\}\,.$$

If the feasible set $\mathcal{X}$ has already been restricted to a subset $\mathcal{X}_{k_1 \dots k_s}$, the neighborhood structure on $\mathcal{X}_{k_1 \dots k_s}$ can be defined in an analogous way:

$$N_{k_1 \dots k_s}(x) = \{x' = (k_1, \dots, k_s, x_{s+1}, \dots, x_{m-1}, 1 - x_m, x_{m+1}, \dots, x_M) \mid s + 1 \leq m \leq M\} .$$

In summary, the Local Search procedure can now be described as follows:

(1) Choose an initial solution $x \in \mathcal{X}_{k_1 \dots k_s}$;

(2) Determine, for all neighbor solutions $x' \in N_{k_1 \dots k_s}(x)$, the overall costs $f(x', \omega_\nu)$, and choose a neighbor x^* with $f(x^*, \omega_\nu) = \min\{f(x', \omega_\nu) \mid x' \in N_{k_1 \dots k_s}(x)\}$;

(3) If $f(x^*, \omega_\nu) \geq f(x, \omega_\nu)$, stop; else set $x := x^*$ and go to (2).

One could conjecture that solving the deterministic subproblem only by means of a heuristic approach "destroys" the convergence property of the Stochastic Branch–and–Bound algorithm as proven by Norkin et al. (see subsection 2.1). Fortunately, this is *not* true: it is possible to extend the convergence results of [6] and [7] to cases in which the deterministic subproblem is approximately solved by a search heuristic with a random starting point keeping track of the best solution found so far. We omit the details, as they do not contribute to the main purpose of this paper.

5 Experimental Results

5.1 Generating Problem Instances

We generated 33 random problem instances on the basis of the following procedure, which is a modification of the random instance generation procedure reported in [1]:

1. Determine the graph size: we generated three types of graphs with $n = 25, 50,$ and 100 nodes, respectively.

2. Determine the inner arc density (an inner arc is an arc that is not connected with the source or the sink of the graph): for each graph size, we generated graphs with four different inner arc densities. The number r of inner arcs is chosen relative to the number of all possible arcs $a = n(n-1)/2$ by using an arc density of 1%, 10%, 20%, or approximately 30%. In the sequel, we use the term "sparse network" for a graph with 1% inner arc density, the term "medium network" for a graph with 10% or 20% inner arc density, and the term "dense network" for a graph with about 30% inner arc density. As a result, networks with 3 to 900 arcs (activities) are generated.

3. Select two events i and j at random; $i < j$.

4. If there is already an arc between i and j or if adding an arc (i,j) to the graph would produce a cycle, go to step 3.

5. Else connect i and j by generating an arc (i,j) and label the arc with three parameters characterizing a beta-distribution: an optimistic, most likely, and pessimistic estimate of the duration of the corresponding activity. The optimistic value and the differences between the optimistic and the most likely value, resp. between the most likely and the pessimistic value are randomly selected from geometric distributions.

6. Repeat steps 3 to 5 until r inner arcs are generated.

7. Eliminate "loose ends" by connecting each source node of the current graph (including each single node) with a randomly selected source node to provide the unique start node, as well as each sink node of the current graph (including each single node) with a randomly selected sink node as the unique end node. (This procedure leads to slight deviations in the number of inner arcs from the intended values specified by the inner arc densities.)

8. Generate $K = 10$ "step times" t_k for the penalty function (cf. section 1): the first step t_1 is assigned to the expected overall completion time in the event that all activities are performed with the most likely duration; the last step t_{10} is assigned to the expected overall completion time in the event that all activities are performed with the pessimistic duration; the distance $t_{i+1} - t_i$ between two step times is chosen as $(t_{10} - t_1)/10$.

9. Assign penalty costs of $const \cdot 2^k$ monetary units to step t_k.

10. Determine the number M of potential measures: for each generated network we produced three different problem instances by setting the number of potential measures $M = 10, 15$, or 20.

11. Generate M measures and assign each measure randomly to an arc. Although in principle one can assign several measures to one arc, we only considered single assignments for this problem. Furthermore, it should be pointed out that the assignment of measures is not restricted to inner arcs, but may also be performed on arcs that eliminate loose ends; cf. step 7. A measure is characterized by the measure costs and by its influence on the parameters of the beta-distribution:

(a) Generate measure costs: estimate expected penalty costs $\mathrm{E}(\Lambda(t(x)))$ without any measure using Brute Force Simulation with $N = 1000$; assign $\mathrm{E}(\Lambda(t(x)))/2$ monetary units as total costs of all measures; distribute these total costs randomly to the single measures.

(b) For each of the three values: in the event that a new random trial (with geometric distribution being used again) yields a lower value, reduce the current value to this lowered value.

12. Determine the time limit for the optimization run: if $m = 10, 15$, and 20, then a total runtime of $t = 2, 10$, and 60 minutes, respectively, is spent for optimization. The runs were

performed on a Pentium 133 MHz with 24 MB RAM using the operating system Windows 95; for programming, C++ was used.

13. Determine the number of different evaluation runs (each run is governed by another sequence of pseudo–random numbers): for $m = 10, 15$, and 20, $s = 10, 5$, and 3 evaluation runs, respectively, have been performed.

We refrained from generating the combination 100-node-networks with a 30% density because considerable runtime would have been necessary in order to make certain that the graph has no cycles. As a result, only 11 — and not 12 — combinations of graph sizes with network densities were generated. In light of the immense computational effort that it would have entailed, we did not attempt to reach exactly 30%; instead, we chose a time window for the generation procedure. The time limit was set so as to generate an inner arc density of approximately 30%; in some cases, more arcs than the exact value were generated within time limits, in other cases the number of generated arcs remained below the exact value.

5.2 Computational Results

Six different approaches are applied to each of the 33 problem instances. Two of the approaches are based on Complete Enumeration as an optimization strategy (i.e., Brute Force Simulation and modified PNET), while the other four approaches use Stochastic Branch–and–Bound. Two of the latter use exact methods to solve the deterministic subproblem (i.e., Complete Enumeration and deterministic Branch–and–Bound), whereas the remaining two approaches, which we subsume (for shorter reference) under the term *HS–Branch–and–Bound*, solve the deterministic subproblem heuristically, namely through the local search algorithm presented earlier in this paper. Figure 2 provides a graphical representation of the classification scheme described above and shows the position of the approaches used for comparative purposes. The shaded boxes indicate approaches that have been selected for our experimental comparison; the white boxes indicate possible alternative approaches. The functionality of the selected approaches, which are numbered (1) to (5b), is briefly described and their results are presented graphically.

In order to achieve a fair comparison, the same size of the runtime window was granted to each approach (see step 12 in subsection 5.1). Trimming the four approaches that are based on Stochastic Branch–and–Bound ((3) to (5b)) to the predefined runtime budgets was easy, since — as mentioned in section 2.1 — Stochastic Branch–and–Bound is well–suited for tailoring the runtime. However, for the two approaches based on Complete Enumeration, the runtime can only be influenced by tuning the sample size; of course, at least one sample has to be drawn, which results in a certain minimum overall runtime. We encountered cases in which the given runtime budget was not sufficient even when the sample size was reduced to $N = 1$; in these

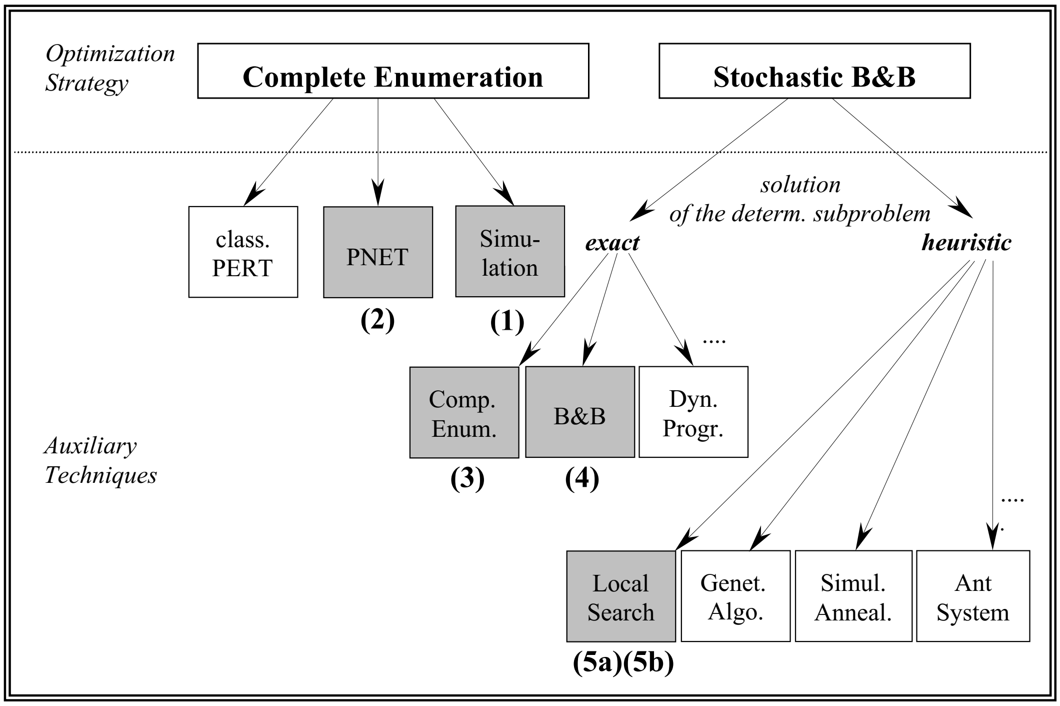

Figure 2: Classification of approaches

cases, the computation procedure had to be aborted without producing any results.

Results were evaluated as follows: for all approaches that have produced a solution x, we re–evaluated these solutions using Brute Force Simulation with sample size $N = 5000$. The best (average) value obtained in this way was used as a yardstick to quantify the relative deviation of the other values. (Of course, it would be desirable to compare the test results with the *optimal* results instead of with the best obtained solutions. Note, however, that the exact optima for our problem instances cannot be determined within reasonable runtime.)

In the Figures 3 to 8, the relative deviation of the costs from the cost value achieved by the best solution is used as an indicator for the quality of results. *Empty* spots in the figure indicate that *no* solution was found during the available runtime, whereas *flat* squares indicate the *best found* solution for a certain problem instance. In order to improve the visualization, relative deviations larger than 200 % have been truncated in the Figures.

Let us add a brief description of the six compared approaches and some comments on the obtained results.

(1) *Brute Force Simulation.* Brute Force Simulation sequentially simulates the costs for each measure combination x based on a sample of N vectors of random values for the activity durations. Contrary to Stochastic Branch–and–Bound, the *same* sample size N is applied to each measure combination x. The quality of the results rapidly declines with increasing number

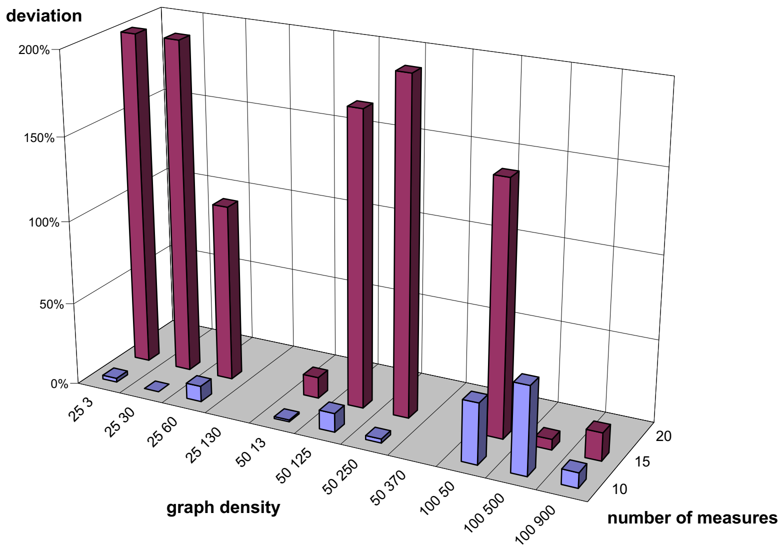

Figure 3: Complete Enumeration with Brute Force Simulation

of measures under consideration (Fig. 3).

(2) *Modified PNET.* The core idea of PNET lies in reducing the complexity of a network in order to simplify the evaluation process. We used the modified PNET algorithm ([5], p. 303), which first identifies major paths, then determines the correlation coefficients for each pair of major paths, and finally determines the representative path. Applying modified PNET to our generated networks, we observed that the method can handle only the sparse networks and some of the medium ones; even for these networks, this approach can result in unacceptable deviations from the best solutions found (Fig. 4). For most dense networks, no solutions were found within the given time limit.

(3) *Stochastic Branch–and–Bound using Complete Enumeration.* This Stochastic Branch–and–Bound approach solves the deterministic subproblem by Complete Enumeration. For networks with 20 measures under consideration, no results could be generated within the time limit (Fig. 5).

(4) *Stochastic Branch–and–Bound using ordinary Branch–and–Bound.* Here, the deterministic subproblem was solved by ordinary (deterministic) Branch–and–Bound. This approach could be executed solving all problem instances within the given runtime limit. Very good results were generated when 10 measures were being considered; for the 15 and 20 measures, the deviations to the best solutions increased. Although the optimum was found twice in the case of 15 measures, some deviations were considerably large (Fig. 6).

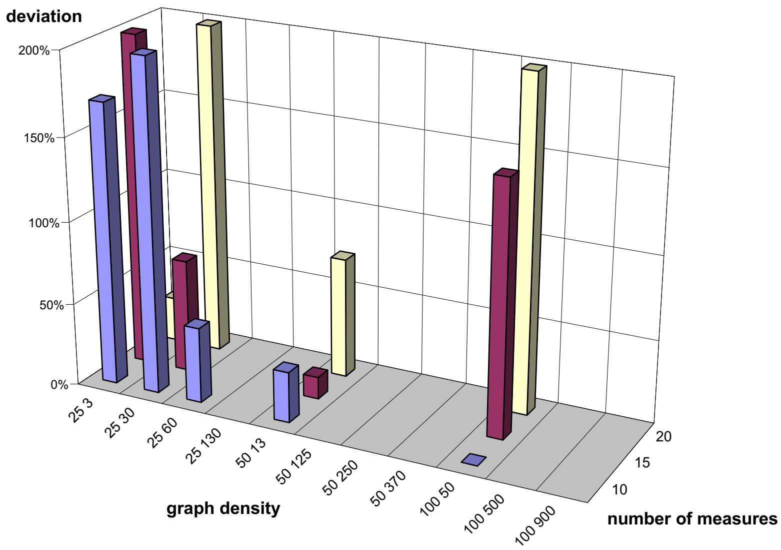

Figure 4: Complete Enumeration with modified PNET

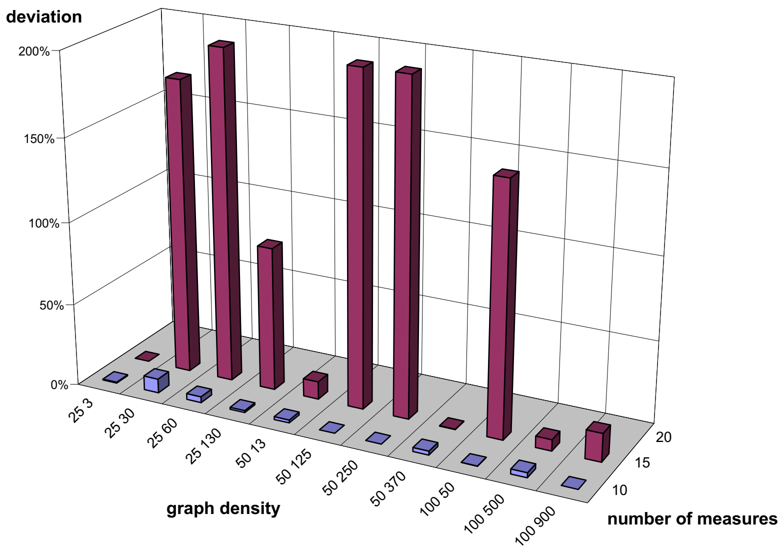

Figure 5: Stochastic Branch–and–Bound using Complete Enumeration for the subproblem

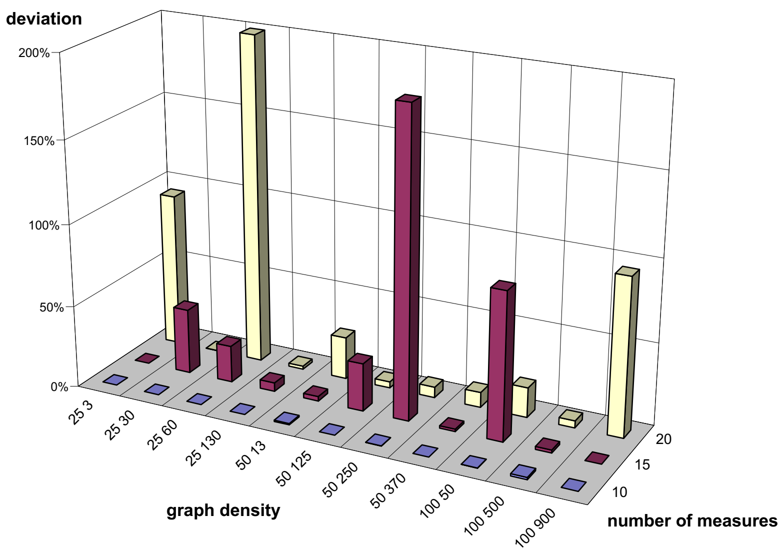

Figure 6: Stochastic Branch–and–Bound using ordinary Branch–and–Bound for the subproblem

(5a) *HS–Branch–and–Bound with Ordinary Sampling.* This Stochastic Branch–and–Bound approach uses Local Search to provide a heuristic solution for the deterministic subproblem. The approach proved to be very successful: it found the best solutions in 13 out of 33 cases (39%), only three deviations exceeded 10%, and the worst result had a deviation of only 17.25% (Fig. 7).

(5b) *HS–Branch–and–Bound with Importance Sampling.* Encouraged by the success of HS–Branch–and–Bound, we tried to further improve this technique by applying the more sophisticated sampling procedure, Importance Sampling, which is described in section 3. This advanced approach found the best solutions in 11 out of 33 cases (33%) (Fig. 8); it seems to work particularly well when 15 or 20 measures are being considered.

In summary, the computational experiments cn be said to show three facts:

(1) Classic approaches based on an optimization strategy of Complete Enumeration fail to generate results within reasonable runtime limits. This difficulty is already encountered when dealing with medium-size problems.

(2) The application of traditional Stochastic Branch–and–Bound, which uses exact methods to solve the deterministic subproblem, generates results that are definitely better than those generated by classic approaches (even for small problem instances, classic approaches are outperformed).

(3) HS–Branch–and–Bound which uses heuristic methods to solve the deterministic sub-

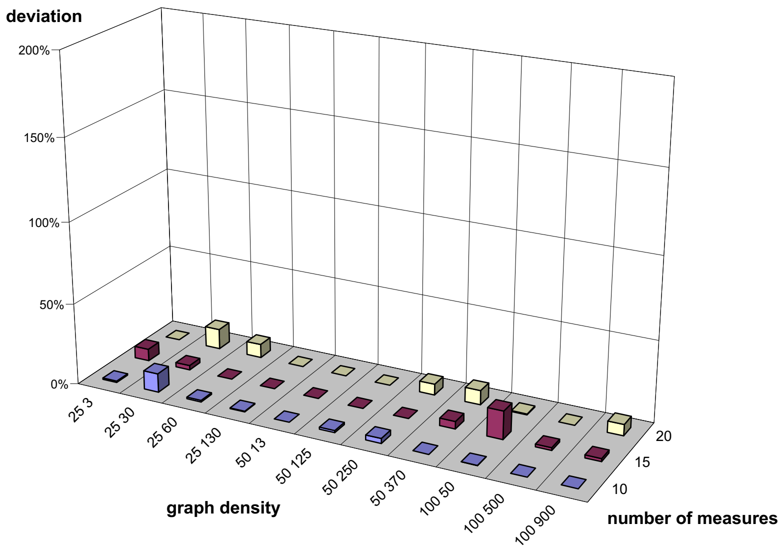

Figure 7: HS–Branch–and–Bound without Importance Sampling

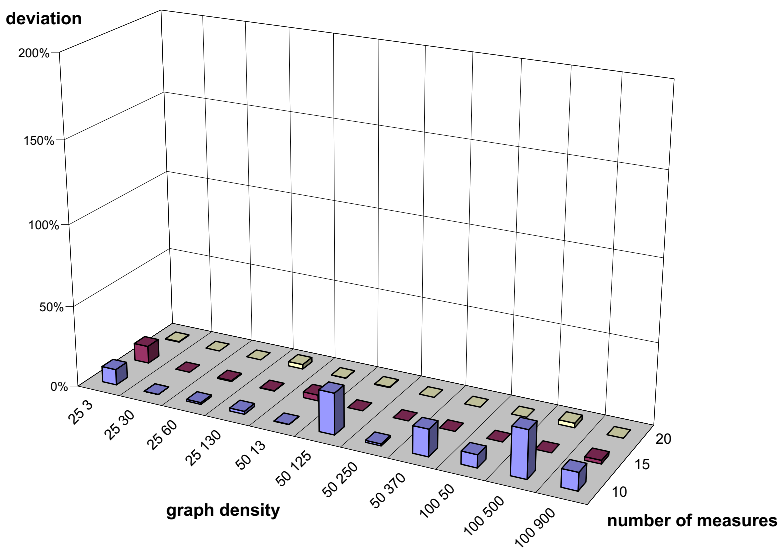

Figure 8: HS–Branch–and–Bound with Importance Sampling

problem outperforms traditional Stochastic Branch–and–Bound.

6 Conclusion

HS–Branch–and–Bound is a new version of Stochastic Branch–and–Bound that solves the deterministic subproblem by a heuristic procedure. To explain the approach and evaluate its quality, we apply it to the Stochastic Discrete Time–Cost Problems (SDTCP), because SDTCP is a core problem occuring (sometimes with additional constraints) in many applications, such as production, R&D, software development, project scheduling, workflow modeling, resource allocation, real time planning, re-engineering, and business process management. Comparative computational tests showed that neither Brute Force Simulation, nor Complete Enumeration using PNET were able to generate results of acceptable quality. HS–Branch–and–Bound outperformed not only these classic methods, but also traditional Stochastic Branch–and–Bound, which uses exact methods for the subproblem.

The encouraging computational results provide impetus for further research activities. Besides experimenting with HS–Branch–and–Bound in various related applications of combinatorial stochastic optimization (e.g., sequencing, stochastic partitioning, routing problems, reliability optimization etc.), one should experiment with incorporating other heuristics like Random Search, Simulated Annealing, Ant Systems, and particular variants of Genetic Algorithms. Moreover, improvements on the level of Stochastic Branch–and–Bound itself (e.g., by developing a priority rule which directs an even quicker navigation in the solution space) ought to be investigated.

Also, as far as the specific problem treated in this paper, the SDTCP, is concerned, additional topics for further research remain: the next step involves a generalization to measures that are not described by 0-1-values, but rather allow several discrete modes. The extension of our technique to such a situation is rather straightforward, but nevertheless deserves a computational study. A further generalization would include a mix of measures with discrete and continuous modes. This extension would require a combination of our approach with established classic techniques for solving continuous time–cost tradeoff problems.

References

[1] R. Etgar, A. Shtub, and L.J. LeBlanc, 1996. Scheduling projects to maximize net present value - the case of time–dependent, contingent cash flows. *European Journal of Operational Research* 96, 90-96.

[2] W.J. Gutjahr, A. Hellmayr, and G.Ch. Pflug, 1999. Optimal Stochastic Scheduling by Stochastic Branch–and–Bound. *European Journal of Operational Research* 117, 396-413.

[3] P. Heidelberger, 1995. Fast Simulation of Rare Events in Queuing and Reliability Models. *ACM Transactions on Modeling and Computer Simulation* 5, 43-85.

[4] T.J. Hindelang and J.F. Muth, 1979. A Dynamic Programming Algorithm for Decision CPM Networks. *Operations Research* 27(2), 225-241.

[5] J.J. Moder, C.F. Phillips, and E.W. Davis, 1983. *Project Management with CPM, PERT and Precedence Diagramming*, 3rd ed., Van Nostrand.

[6] V.I. Norkin, Y.M. Ermoliev, and A. Ruszczynski, 1998. On optimal allocation of indivisibles under uncertainty. *Operations Research* 46(3), 381-395.

[7] V.I. Norkin, G.Ch. Pflug, and A. Ruszcynski, 1998. A branch and bound method for stochastic global optimization. *Mathematical Programming* 83, 425-450.

[8] D. Panagiotakopoulos, 1977. A CPM time–cost computational algorithm for arbitrary activity cost functions. *INFOR* 15(2), 183-195.

[9] C.H. Papadimitriou and K. Steiglitz, 1982. *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ.

[10] R.Y. Rubinstein, 1981. *Simulation and the Monte Carlo Method*, Wiley, New York.

[11] L. Sunde and S. Lichtenberg, 1995. Net-present-value cost/time tradeoff. *International Journal of Project Management* 13(1), 45-49.

[12] C. Yau, and E. Ritchie, 1990. Project compression: A method for speeding up resource constrained projects which preserve the activity schedule. *European Journal of Operational Research* 49, 140-152.