The Sample Average Approximation Method Applied to Stochastic Routing Problems: A Computational Study

Bram Verweij[*], Shabbir Ahmed[†], Anton J. Kleywegt[*],
George Nemhauser[*], and Alexander Shapiro[*]

Georgia Institute of Technology
School of Industrial and Systems Engineering
Atlanta, GA 30332-0205

July 30, 2002

Abstract

The sample average approximation (SAA) method is an approach for solving stochastic optimization problems by using Monte Carlo simulation. In this technique the expected objective function of the stochastic problem is approximated by a sample average estimate derived from a random sample. The resulting sample average approximating problem is then solved by deterministic optimization techniques. The process is repeated with different samples to obtain candidate solutions along with statistical estimates of their optimality gaps.

We present a detailed computational study of the application of the SAA method to solve three classes of stochastic routing problems. These stochastic problems involve an extremely large number of scenarios and first-stage integer variables. For each of the three problem classes, we use decomposition and branch-and-cut to solve the approximating problem within the SAA scheme. Our computational results indicate that the proposed method is successful in solving problems with up to 2^{1694} scenarios to within an estimated 1.0% of optimality. Furthermore, a surprising observation is that the number of optimality cuts required to solve the approximating problem to optimality does not significantly increase with the size of the sample. Therefore, the observed computation times needed to find optimal solutions to the approximating problems grow only linearly with the sample size. As a result, we are able to find provably near-optimal solutions to these difficult stochastic programs using only a moderate amount of computation time.

1 Introduction

This paper addresses two-stage Stochastic Routing Problems (SRPs), where the first stage consists of selecting a route, i.e., a path or a tour, through a graph subject to arc failures or delays, and the second stage involves some recourse, such as a penalty or a re-routing decision. The overall objective is to minimize the sum of the first-stage routing cost and the expected recourse cost. A generic formulation of this class of problems is

$$z^* \;=\; \min_{x \in X} c^T x + \mathbb{E}_{\mathcal{P}}[Q(x, \xi(\omega))], \tag{1}$$

[*]Supported by NSF grant DMI-0085723.
[†]Supported by NSF grant DMI-0099726.

where

$$Q(x, \xi(\omega)) \;=\; \min_{y \geq 0} \{q(\omega)^T y \mid Dy \geq h(\omega) - T(\omega)x\}, \tag{2}$$

x denotes the first-stage routing decision, X denotes the first-stage feasible set involving route defining constraints, i.e., $X = R \cap \{0,1\}^d$ for some polyhedron R of dimension d, $\omega \in \Omega$ denotes a *scenario* that is unknown when the first-stage decision x has to be made, but that is known when the second-stage recourse decision y is made, Ω is the set of all scenarios, and c denotes the routing cost. We assume that the probability distribution $\mathcal{P}$ on Ω is known in the first stage. The quantity $Q(x, \xi(\omega))$ represents the optimal value of the second-stage recourse problem corresponding to the first-stage route x and the parameters $\xi(\omega) = (q(\omega), h(\omega), T(\omega))$. Note that the first-stage variables x are binary in this problem. Hence, SRP belongs to the class of two-stage stochastic programs with integer first-stage variables.

The difficulty in solving SRP is two-fold. First, the evaluation of $\mathbb{E}[Q(x, \xi(\omega))]$ for a given value of x requires the solution of numerous second-stage optimization problems. If Ω contains a finite number of scenarios $\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$, with associated probabilities p_k, $k = 1, 2, \ldots, |\Omega|$, then the expectation can be evaluated as the finite sum

$$\mathbb{E}[Q(x, \xi(\omega))] \;=\; \sum_{k=1}^{|\Omega|} p_k Q(x, \xi(\omega_k)). \tag{3}$$

However, the number of scenarios grows exponentially fast with the dimension of the data. For example, consider a graph with 100 arcs, each of which has a parameter with three possible values. This leads to $|\Omega| = 3^{100}$ scenarios for the parameter vector of the graph, making a direct application of (3) impractical. Second, $\mathbb{E}[Q(x, \xi(\omega))]$, regarded as a function defined on $\mathbb{R}^d$ (and not just on the discrete points $X = R \cap \{0,1\}^d$), is a polyhedral function of x. Consequently, SRP involves minimizing a piecewise linear objective over an integer feasible region and can be very difficult, even for problems with a modest number of scenarios.

Early solution approaches for stochastic routing problems were based on heuristics [35] or dynamic programming [1, 2]. These methods were tailored to very specific problem structures and are not applicable in general. Exact mathematical programming methods for this class of problems have mostly been restricted to cases with a small number of scenarios, to allow for the exact evaluation of the second-stage expected value function $\mathbb{E}[Q(x, \xi(\omega))]$. Exploiting this property, Laporte *et al.* [18, 19, 20, 21] solved various stochastic routing problems using the integer L-shaped algorithm of Laporte and Louveaux [17]. Wallace [38, 39, 40] has studied the exact evaluation of the second-stage expected value function for problems in which the first stage feasible set is convex and the second stage involves routing decisions.

For stochastic programs with a prohibitively large number of scenarios, a number of sampling based approaches have been proposed, for example to estimate function values, gradients, optimality cuts, or bounds for the second-stage expected value function. We classify such sampling based approaches into two main groups: interior sampling and exterior sampling methods. In interior sampling methods, the samples are modified during the optimization procedure. Samples may be modified by adding to previously generated samples, by taking subsets of previously generated samples, or by generating completely new samples. For example, methods that modify samples within the L-shaped algorithm for stochastic linear programming have been suggested by Van Slyke and Wets [36], Higle and Sen [12] (stochastic decomposition) and Infanger [14] (importance sampling). For discrete stochastic problems, an interior sampling branch-and-bound method was developed by Norkin *et al.* [24, 25]. In exterior sampling methods, a sample $\omega^1, \omega^2, \ldots, \omega^N$ of N *sample scenarios*

is generated from Ω according to probability distribution $\mathcal{P}$ (we use the superscript convention for sample values), and then a deterministic optimization problem specified by the generated sample is solved. This procedure may be repeated by generating several samples and solving several related optimization problems. For example, the Sample Average Approximation (SAA) method is an exterior sampling method in which the expected value function $\mathbb{E}[Q(x, \xi(\omega))]$ is approximated by the sample average function $\sum_{n=1}^{N} Q(x, \xi(\omega^n))/N$. The *Sample Average Approximation problem*

$$z_{_N} \;=\; \min_{x \in X} c^T x + \frac{1}{N} \sum_{n=1}^{N} Q(x, \xi(\omega^n)), \tag{4}$$

corresponding to the original two-stage stochastic program (1) (also called the "true" problem) is then solved using a deterministic optimization algorithm. The optimal value $z_{_N}$ and an optimal solution $\hat{x}$ to the SAA problem provide estimates of their true counterparts in the stochastic program (1). Over the years the SAA method has been used in various ways under different names, see e.g. Rubinstein and Shapiro [30], and Geyer and Thompson [8]. SAA methods for stochastic linear programs have been proposed, among others, by Plambeck *et al.* [28], Shapiro and Homem-de-Mello [32, 33], and Mak, Morton, and Wood [22]. Kleywegt, Shapiro, and Homem-de-Mello [16] analyzed the behavior of the SAA method when applied to stochastic discrete optimization problems. However, computational experience with this method is limited.

In this paper, we present a detailed computational study of the application of the SAA method combined with a decomposition based branch-and-cut framework to solve three classes of stochastic routing problems (SRPs). The first SRP is a shortest path problem in which arcs have deterministic costs and random travel times, and we have to pay a penalty for exceeding a given arrival deadline. In the second SRP, we are again looking for a shortest path, but now arcs in the graph can fail and the recourse action has to restore the feasibility of the first-stage path at minimum cost. In the third SRP, the recourse is the same as in the first SRP, but now the first-stage solution should be a tour instead of a path. The first and third SRPs have continuous recourse decisions, whereas the second SRP has discrete recourse decisions and a recourse problem with the integrality property, so that for all three SRPs it is sufficient to solve continuous recourse problems.

Integrating the branch-and-cut method within the SAA method allows us to solve problems with huge sample spaces Ω. We have been succesful in solving problems with up to 2^{1694} scenarios to within an estimated 1.0% of optimality. Furthermore, a surprising observation is that the number of optimality cuts required to solve the SAA problem to optimality does not significantly increase with the sample size. We suggest an explanation of this phenomenon.

The remainder of this paper is organized as follows. Section 2 gives a general description of our solution methodology for two-stage stochastic routing problems. In Section 3 we introduce the specific problem classes and discuss their computational complexity. We proceed in Section 4 by describing how the method of Section 2 applies to the specific problem classes of Section 3. Our computational results are reported in Section 5. Section 6 discusses the empirically observed stability of the number of optimality cuts. Finally, our conclusions are presented in Section 7.

2 Methodology

Our solution methodology for problem (1) consists of the integration of the SAA method, to deal with the extremely large sets Ω, and a decomposition based branch-and-cut algorithm, to deal with the integer variables in the approximating problems. We assume that the problem has relatively complete recourse (a two-stage stochastic program is said to have relatively complete recourse if

for each feasible first-stage solution $x \in X$ and each scenario $\omega \in \Omega$, there exists a second-stage solution $y \geq 0$ such that $Dy \geq h(\omega) - T(\omega)x$).

2.1 The Sample Average Approximation Method

As mentioned earlier, the SAA method proceeds by solving problem (4) repeatedly. By generating M independent samples, each of size N, and solving the associated SAA problems, objective values $z_N^1, z_N^2, \ldots, z_N^M$ and candidate solutions $\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^M$ are obtained. Let

$$\bar{z}_N \;=\; \frac{1}{M}\sum_{m=1}^{M} z_N^m \tag{5}$$

denote the average of the M optimal values of the SAA problems. It is well-known that $\mathbb{E}[\bar{z}_N] \leq z^*$ [25, 22]. Therefore, $\bar{z}_N$ provides a statistical estimate for a lower bound on the optimal value of the true problem.

For any feasible point $\hat{x} \in X$, clearly the objective value $c^T\hat{x} + \mathbb{E}[Q(\hat{x}, \xi(\omega))]$ is an upper bound for z^*. This upper bound can be estimated by

$$\hat{z}_{N'}(\hat{x}) \;=\; c^T\hat{x} + \frac{1}{N'}\sum_{n=1}^{N'} Q(\hat{x}, \xi(\omega^n)), \tag{6}$$

where $\{\omega^1, \omega^2, \ldots, \omega^{N'}\}$ is a sample of size N'. Typically N' is chosen to be quite large, $N' > N$, and the sample of size N' is independent of the sample, if any, used to generate $\hat{x}$. Then we have that $\hat{z}_{N'}(\hat{x})$ is an unbiased estimator of $c^T\hat{x} + \mathbb{E}[Q(\hat{x}, \xi(\omega))]$, and hence, for any feasible solution $\hat{x}$, we have that $\mathbb{E}[\hat{z}_{N'}(\hat{x})] \geq z^*$. The variances of the estimators $\bar{z}_N$ and $\hat{z}_{N'}(\hat{x})$ can be estimated by

$$\hat{\sigma}^2_{\bar{z}_N} \;=\; \frac{1}{(M-1)M}\sum_{m=1}^{M}\left(z_N^m - \bar{z}_N\right)^2 \tag{7}$$

and

$$\hat{\sigma}^2_{\hat{z}_{N'}(\hat{x})} \;=\; \frac{1}{(N'-1)N'}\sum_{n=1}^{N'}\left(c^T\hat{x} + Q(\hat{x}, \xi(\omega^n)) - \hat{z}_{N'}(\hat{x})\right)^2, \tag{8}$$

respectively.

Note that the above procedure produces up to M different candidate solutions. It is natural to take $\hat{x}^*$ as one of the optimal solutions $\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^M$ of the M SAA problems which has the smallest estimated objective value, that is,

$$\hat{x}^* \;\in\; \arg\min\left\{\hat{z}_{N'}(\hat{x}) \mid \hat{x} \in \{\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^M\}\right\}. \tag{9}$$

One can evaluate the quality of the solution $\hat{x}^*$ by computing the optimality gap estimate

$$\hat{z}_{N'}(\hat{x}^*) - \bar{z}_N, \tag{10}$$

where $\hat{z}_{N'}(\hat{x}^*)$ is recomputed after performing the minimization in (9) with an independent sample to obtain an unbiased estimate. The estimated variance of this gap estimator is

$$\hat{\sigma}^2_{\hat{z}_{N'}(\hat{x}^*) - \bar{z}_N} \;=\; \hat{\sigma}^2_{\hat{z}_{N'}(\hat{x}^*)} + \hat{\sigma}^2_{\bar{z}_N}.$$

The above procedure for statistical evaluation of a candidate solution was suggested in Norkin *et al.* [25] and developed in Mak *et al.* [22]. Convergence properties of the SAA method were studied in Kleywegt *et al.* [16].

2.2 Solving SAA problems using Branch-and-Cut

The SAA method requires the solution of the approximating problem (4). This problem can also be written as

$$\min_{x \in X} \quad c^T x + \frac{1}{N} Q_N(x), \tag{11}$$

where $Q_N(x) = \sum_{n=1}^{N} Q(x, \xi(\omega^n))$, and Q is given by (2).

It is well-known that the functions $Q(x, \xi(\omega^n))$, and hence $Q_N(x)$, are piecewise linear and convex in x. Thus, because of relatively complete recourse, $Q_N(x) = \max\{a_i^T x - a_{i0} \mid i \in \{1, 2, \ldots, p\}\}$ for all $x \in X$, for some $\{(a_{i0}, a_i)\}$, $i = 1, 2, \ldots, p$, and some positive integer p. Hence problem (11) can be reformulated as a mixed-integer linear program with p constraints representing affine supports of Q_N, as follows.

$$\min \; c^T x + \theta/N \tag{12a}$$

$$\text{subject to } \theta \geq a_i^T x - a_{i0} \qquad \text{for } i \in \{1, 2, \ldots, p\} \tag{12b}$$

$$x \in X \tag{12c}$$

For a given first-stage solution x, the optimal value of θ is equal to $Q_N(x)$. The coefficients a_{i0}, a_i of the optimality cuts (12b) are given by the values and extreme subgradients of Q_N, which in turn are given by the extreme point optimal dual solutions of the second-stage problems (2). The number p of constraints in (12b) typically is very large. Instead of including all such constraints, we generate only a subset of these constraints within a branch-and-cut framework. For a given first-stage solution x, the second-stage problem of each sample scenario can be solved independently of the others, providing a computationally convenient decomposition, and their dual solutions can be used to construct optimality cuts. This is the well-known Benders or L-shaped decomposition method [36].

The standard branch-and-cut procedure is an LP-based branch-and-bound algorithm for mixed-integer linear programs, in which a separation algorithm is used at each node of the branch-and-bound tree. The separation algorithm takes as input a solution of an LP relaxation of the integer program, and looks for inequalities that are satisfied by all integer feasible points, but that are violated by the solution of the LP relaxation. These violated valid inequalities (cuts) are then added to the LP relaxation. For more details on branch-and-cut algorithms we refer to Nemhauser and Wolsey [23], Wolsey [43], and Verweij [37].

Our branch-and-cut procedure for (12) starts with some initial relaxation that does not include any of the optimality cuts (12b), and adds violated valid inequalities as the algorithm proceeds. This includes violated valid inequalities for X, and the optimality cuts (12b), which are found by solving the second-stage subproblems. As in the integer L-shaped method by Laporte and Louveaux [17], we only generate optimality cuts at solutions that satisfy all first-stage constraints, including integrality. Our approach to generating optimality cuts is similar to the one used in the earlier exact L-shaped methods (see e.g. Wets [42]), except that we use the sample average function instead of (3), and embedding it in a branch-and-cut framework to deal with integer variables in the first stage.

3 Problem Classes

Here we introduce the problem classes on which we perform our computational study. Sections 3.1, 3.2, and 3.3 deal with the shortest path problem with random travel times, the shortest

path problem with arc failures, and the traveling salesman problem with random travel times, respectively. The computational complexity of these problems is discussed in Section 3.4. In Section 3.5 we analyze the mean value problems corresponding to the specific problem classes at hand. This is of interest because it justifies the use of stochastic models.

Each of the problems considered next involves a graph $G = (V, A)$ or $G = (V, E)$. For each node $i \in V$, $\delta_G^+(i)$ ($\delta_G^-(i)$) denotes the set of arcs in G leaving (entering) i, and $\delta_G(i)$ denotes the set of edges in G incident to i. For any $S \subseteq V$, $\gamma_G(S)$ denotes the set of arcs or edges in G with both endpoints in S. For any $x \in \mathbb{R}^{|A|}$ and $A' \subseteq A$, $x(A')$ denotes $\sum_{a \in A'} x_a$.

3.1 Shortest Path with Random Travel Times

In the Shortest Path problem with Random travel Times (SPRT), the input is a directed graph $G = (V, A)$, a source node $s \in V$ and a sink node $t \in V$, arc costs $c \in \mathbb{R}^{|A|}$, a probability distribution $\mathcal{P}$ of the vector of random travel times $\xi(\omega) \in \mathbb{R}_+^{|A|}$, and a deadline $\kappa \in \mathbb{R}$. The problem is to find an $s - t$ path that minimizes the cost of the arcs it traverses plus the expected violation of the deadline.

The variables $x \in \{0, 1\}^{|A|}$ represent the path, where $x_a = 1$ if arc a is in the path and $x_a = 0$ otherwise. The SPRT can be stated as the following two-stage stochastic integer program:

$$\min_{x \in \{0,1\}^{|A|}} c^T x + \mathbb{E}_{\mathcal{P}}[Q(x, \xi(\omega))] \tag{13a}$$

$$\text{subject to } x(\delta_G^+(i)) - x(\delta_G^-(i)) = \begin{cases} 1, & \text{for } i = s, \\ -1, & \text{for } i = t, \\ 0, & \text{for } i \in V \setminus \{s, t\}, \end{cases} \tag{13b}$$

$$x(\gamma_G(S)) \;\leq\; |S| - 1, \qquad \text{for all } S \subset V \text{ with } |S| \geq 2, \tag{13c}$$

where

$$Q(x, \xi(\omega)) \;=\; \max\left\{\xi(\omega)^T x - \kappa, 0\right\}.$$

The same recourse function was used by Laporte *et al.* [19] in the context of a vehicle routing problem. Note that, if G does not contain negative-cost cycles, then the cycle elimination constraints (13c) can be omitted. That is, if for all cycles C in G and all ω, $\sum_{a \in C} c_a \geq 0$ and $\sum_{a \in C} \xi_a(\omega) \geq 0$; then there is an optimal solution of (13a)–(13b) that is a path.

3.2 The Shortest Path Problem with Random Arc Failures

Given a directed graph $G = (V, A)$, the set of *reverse arcs* A^{-1} is defined by $A^{-1} = \{a^{-1} \mid a \in A\}$, where for $(i, j) \in A$, the arc $(i, j)^{-1} = (j, i)$ has head i and tail j. Note that although (i, j) and $(j, i)^{-1}$ have the same head and tail, they are different (parallel) arcs, with $(i, j) \in A$ and $(j, i)^{-1} \in A^{-1}$. In the Shortest Path problem with random Arc Failures (SPAF), the input is a directed graph $G = (V, A)$, a source node $s \in V$ and a sink node $t \in V$, arc costs $c^1 \in \mathbb{R}^{|A|}$, and a probability distribution $\mathcal{P}$ of $\xi(\omega) = (c^2(\omega), u(\omega)) \in \mathbb{R}^{2|A|} \times \{0, 1\}^{|A|}$. Here, $c^2(\omega)$ is a vector of random arc and reverse arc costs, and $u(\omega)$ is a vector of random arc capacities, where $u_a(\omega) = 0$ denotes that arc a is not available and $u_a(\omega) = 1$ denotes that it is. Let $G' = (V, A \cup A^{-1})$.

In the first stage, an $s - t$ path P has to be chosen in G. After a first-stage path has been chosen, the values of the random variables $\xi(\omega) = (c^2(\omega), u(\omega))$ are observed, and then a second-stage decision is made. If an arc on the first-stage path fails, then the path becomes infeasible, and the first-stage flow on that arc has to be canceled. Flow on other arcs in the first-stage path may

be canceled too. First-stage flow on an arc $(i,j) \in A$ is canceled in the second stage by flow on its reverse arc $(i,j)^{-1}$. Such cancelation of a unit flow incurs a cost of $c^2_{(i,j)^{-1}}(\omega)$. (If $c^2_{(i,j)^{-1}}(\omega) < 0$, then a refund is received if the first-stage flow on arc $(i,j) \in A$ is canceled.) The second-stage decision consists of choosing flows in G' in such a way that the combination of the uncanceled flow in the first-stage path P and the second-stage flow on arcs in A forms an $s-t$ path in G. Specifically, let $x \in \{0,1\}^{|A|}$ represent the first-stage path, where $x_a = 1$ if arc a is in the path and $x_a = 0$ otherwise, and let $y \in \{0,1\}^{|A \cup A^{-1}|}$ respresent the second-stage flow. For any $y \in \{0,1\}^{|A \cup A^{-1}|}$, let $y^+, y^- \in \{0,1\}^{|A|}$ be defined by $y^+_a = y_a$ and $y^-_a = y_{a^{-1}}$ for all $a \in A$. Then let $\Delta y \in \{-1,0,1\}^{|A|}$ be defined by $\Delta y = y^+ - y^-$, that is, $\Delta y_a = y_a - y_{a^{-1}}$ for all $a \in A$. For (x,y) to be feasible, it is necessary that both x and $x + \Delta y$ be incidence vectors of $s-t$ paths in G. The objective of the SPAF is to minimize the sum of the cost of the first-stage path and the expected cost of the second-stage flow needed to restore feasibility.

Before we give a formulation of the SPAF, we characterize the structure of second-stage decisions.

Proposition 3.1. *The feasible second-stage decisions of the SPAF are circulations in G'.*

Proof. Because x and $x + \Delta y$ are both incidence vectors of $s-t$ paths in G, it follows that

$$\begin{aligned}
(x+\Delta y)(\delta^+_G(i)) - (x+\Delta y)(\delta^-_G(i)) &= x(\delta^+_G(i)) - x(\delta^-_G(i)) \\
\Rightarrow \quad \Delta y(\delta^+_G(i)) &= \Delta y(\delta^-_G(i)) \\
\Rightarrow \quad (y^+ - y^-)(\delta^+_G(i)) &= (y^+ - y^-)(\delta^-_G(i)) \\
\Rightarrow \quad y(\delta^+_{G'}(i)) &= y(\delta^-_{G'}(i))
\end{aligned}$$

for all nodes $i \in V$. □

However, all circulations in G' do not form paths when combined with a first-stage path P. Proposition 3.2 will establish conditions under which all circulations in G' can be allowed in the second stage.

The SPAF can be stated as the following two-stage stochastic integer program:

$$\min_{x \in \{0,1\}^{|A|}} (c^1)^T x + \mathbb{E}[Q(x, \xi(\omega))] \tag{14a}$$

$$\text{subject to } x(\delta^+_G(i)) - x(\delta^-_G(i)) = \begin{cases} 1, & \text{for } i = s, \\ -1, & \text{for } i = t, \\ 0, & \text{for } i \in V \setminus \{s,t\}, \end{cases} \tag{14b}$$

$$x(\gamma_G(S)) \;\le\; |S| - 1, \qquad \text{for all } S \subset V \text{ with } |S| \ge 2, \tag{14c}$$

$$\tag{14d}$$

where

$$Q(x, \xi(\omega)) = \min\ (c^2(\omega))^T y \tag{15a}$$

$$\text{subject to } y(\delta^+_{G'}(i)) - y(\delta^-_{G'}(i)) \;=\; 0, \qquad \text{for all } i \in V \tag{15b}$$

$$(x + \Delta y)\,(\gamma_G(S)) \;\le\; |S| - 1, \qquad \text{for all } S \subset V \text{ with } |S| \ge 2, \tag{15c}$$

$$y_a \in \{l_a(x, u(\omega)), r_a(x, u(\omega))\}, \qquad \text{for all } a \in A \cup A^{-1}, \tag{15d}$$

and $l_a(x, u(\omega)), r_a(x, u(\omega))$ are given by

$$l_a(x, u(\omega)) = 0, \qquad \text{and} \qquad r_a(x, u(\omega)) = u_a(\omega)(1 - x_a), \qquad \text{for all } a \in A$$

and

$$l_{a^{-1}}(x, u(\omega)) = x_a(1 - u_a(\omega)), \qquad \text{and} \qquad r_{a^{-1}}(x, u(\omega)) = x_a, \qquad \text{for all } a^{-1} \in A^{-1}.$$

Upper bound $r_a(x, u(\omega)) = u_a(\omega)(1 - x_a)$ for an arc $a \in A$ means that the second-stage decision y can have $y_a = 1$ only if $x_a = 0$, that is, an arc that was chosen in the first stage cannot be chosen in the second stage as well. If it is acceptable to choose an arc in the first stage, and then cancel the arc in the second stage and choose it again, then the upper bound is $r_a(x, u(\omega)) = u_a(\omega)$. This detail does not affect the rest of the development.

Proposition 3.2 shows that, under a particular condition, the cycle elimination constraints (15c) can be omitted from the second-stage problem (15).

Proposition 3.2. *Consider the second-stage problem (15) associated with a feasible first-stage solution x and a scenario ω. Suppose that for all cycles C in G with $u_a(\omega) = 1$ for all $a \in C$, and for all partitions $C = C_0 \cup C_1$, $C_0 \cap C_1 = \varnothing$, it holds that $\sum_{a \in C_1} c^2_a(\omega) - \sum_{a \in C_0} c^2_{a^{-1}}(\omega) \geq 0$ (it is cheaper to cancel the flow on the arcs in C_0 than it is to incur the costs for the arcs in C_1). Then, the cycle elimination constraints (15c) can be omitted from the second-stage problem (15) without loss of optimality.*

Proof. Consider any feasible solution y of the second-stage relaxation obtained by omitting the cycle elimination constraints (15c) from the second-stage problem (15). (Such a y always exists because of the assumption of relatively complete recourse.) Solution y still produces a circulation because of (15b). However, $x + \Delta y$ may not produce a path, but a path combined with a circulation. It remains to be shown that such a circulation can be removed without loss of optimality.

Let $C \subset A$ be any cycle in the circulation produced by $x + \Delta y$. It follows from the feasibility of y that $u_a(\omega) = 1$ for all $a \in C$. Let $C_0 = \{a \in C \mid y_a = 0\}$ and $C_1 = \{a \in C \mid y_a = 1\}$. For each $a \in A$, let $y^+_a = 0$ and $y^+_{a^{-1}} = 1$ if $a \in C_0$, $y^+_a = 0$ and $y^+_{a^{-1}} = y_{a^{-1}}$ if $a \in C_1$, and $y^+_a = y_a$ and $y^+_{a^{-1}} = y_{a^{-1}}$ if $a \notin C$. It is easy to check that y^+ eliminates C from the circulation, and thus satisfies circulation constraints (15b). Note that $r_{a^{-1}}(x, u(\omega)) = x_a = 1$ (and $y_{a^{-1}} = 0$) for all $a \in C_0$, and $l_a(x, u(\omega)) = 0$ for all $a \in C_1$, and thus y^+ also satisfies capacity and integrality constraints (15d). The difference in cost between y and y^+ is $\sum_{a \in C_1} c^2_a(\omega) - \sum_{a \in C_0} c^2_{a^{-1}}(\omega) \geq 0$.

Repeating the procedure a finite number of times, all cycles can be eliminated from the circulation, resulting in a second-stage solution y^+ with no more cost than y, and such that $x + \Delta y^+$ produces an $s - t$ path. □

Corollary 3.3 follows from the observation that, if the cycle elimination constraints (15c) are omitted from the second-stage problem (15), then the resulting relaxation is a network flow problem, which has the integrality property.

Corollary 3.3. *Suppose the conditions of Proposition 3.2 hold. Then the capacity and integrality constraints (15d) can be replaced with*

$$l_a(x, u(\omega)) \;\leq\; y_a \;\leq\; r_a(x, u(\omega)) \tag{16}$$

for all $a \in A \cup A^{-1}$, without loss of optimality. Also, $Q(x, \xi(\omega))$ is piecewise linear and convex in $x \in \mathbb{R}^{|A|}$ for all ω.

In the computational work, the conditions of Proposition 3.2 were satisfied.

Proposition 3.4 shows that, under some conditions, the cycle elimination constraints (14c) can also be omitted from the first-stage problem.

Proposition 3.4. *Suppose that for all $a \in A$ and all ω,*

(i) $c_a^1 + c_{a^{-1}}^2(\omega) \geq 0$ *(it is not profitable to choose an arc in the first stage and cancel the arc in the second stage), and*

(ii) $c_a^2(\omega) \leq c_a^1$ *(it is not profitable to choose an arc in the first stage in anticipation of needing the arc in the second stage).*

Then, the cycle elimination constraints (14c) can be omitted from the first-stage problem (14) without loss of optimality.

Proof. Consider any feasible solution x of the first-stage relaxation obtained by omitting the cycle elimination constraints (14c) from the first-stage problem (14). Solution x may produce a path combined with a circulation, because of (14b). It remains to be shown that such a circulation can be removed without loss of optimality.

Let $C \subset A$ be any cycle in the circulation produced by x. Let $x_a^+ = 0$ for each $a \in C$, and let $x_a^+ = x_a$ for each $a \notin C$. Note that x^+ satisfies (14b). Consider any scenario ω, and any feasible solution y of the resulting second-stage problem (15).

Let $C^+ = \{a \in C \mid y_{a^{-1}} = 0\}$ and $C^- = \{a \in C \mid y_{a^{-1}} = 1\}$. For each $a \in A$, let $y_a^+ = 1$ and $y_{a^{-1}}^+ = 0$ if $a \in C^+$, $y_a^+ = y_a$ and $y_{a^{-1}}^+ = 0$ if $a \in C^-$, and $y_a^+ = y_a$ and $y_{a^{-1}}^+ = y_{a^{-1}}$ if $a \notin C$. It is easy to check that (x^+, y^+) satisfies circulation constraints (15b). Also, $x_a^+ + y_a^+ - y_{a^{-1}}^+ \leq x_a + y_a - y_{a^{-1}}$ for all $a \in A$, and thus $(x^+ + \Delta y^+)(\gamma_G(S)) \leq (x + \Delta y)(\gamma_G(S)) \leq |S| - 1$ for all $S \subset V$ with $|S| \geq 2$, and hence (x^+, y^+) satisfies the second-stage cycle elimination constraints (15c). Note that $r_a(x^+, u(\omega)) = u_a(\omega) = 1$ for all $a \in C^+$, and $l_{a^{-1}}(x^+, u(\omega)) = 0$ for all $a \in C^-$, and thus (x^+, y^+) also satisfies capacity and integrality constraints (15d). The difference in cost between (x, y) and (x^+, y^+) is $\sum_{a \in C^+} \left(c_a^1 - c_a^2(\omega)\right) + \sum_{a \in C^-} \left(c_a^1 + c_{a^{-1}}^2(\omega)\right) \geq 0$. Thus for any scenario ω, there is a second-stage solution y^+ such that (x^+, y^+) satisfies the second-stage constraints and (x^+, y^+) has no more cost than (x, y).

Repeating the procedure a finite number of times, all cycles can be eliminated from the circulation, resulting in a first-stage solution x^+ with no more cost than x, and such that x produces an $s - t$ path. □

3.3 The Traveling Salesman Problem with Random Travel Times

In the Traveling Salesman problem with Random travel Times (TSPRT), the input is an undirected graph $G = (V, E)$, edge costs $c \in \mathbb{R}^{|E|}$, a probability distribution $\mathcal{P}$ of the vector of random travel times $\xi(\omega) \in \mathbb{R}^{|E|}$, and a deadline $\kappa \in \mathbb{R}$. The problem is to find a Hamiltonian cycle in G that minimizes the cost of the edges in the cycle plus the expected violation of the deadline.

We use $x \in \{0, 1\}^{|E|}$ to represent the tour, where $x_e = 1$ if edge e is in the tour and $x_e = 0$ otherwise. The TSPRT can be formulated as the following two-stage stochastic integer program:

$$\min\ c^T x + \mathbb{E}[Q(x, \xi(\omega))] \tag{17a}$$

$$\text{subject to } x(\delta_G(i)) = 2, \qquad \text{for all } i \in V, \tag{17b}$$

$$x(\gamma_G(S)) \leq |S| - 1, \qquad \text{for all } S \subset V \text{ with } |S| \geq 2, \tag{17c}$$

$$x \in \{0, 1\}^{|E|}, \tag{17d}$$

where $Q(x, \xi(\omega)) = \max\left\{\xi(\omega)^T x - \kappa, 0\right\}$.

3.4 Computational Complexity

Here we discuss the computational complexity of the models introduced in Sections 3.1, 3.2, and 3.3. We start by observing that in the case of the SPRT and the TSPRT, the stochastic part can be made redundant by choosing a sufficiently large κ, in which case both models reduce to their deterministic counterpart. This shows that both the TSPRT and its SAA are $\mathcal{NP}$-hard.

In the case of the SPRT, we show $\mathcal{NP}$-hardness by reducing the weight constrained shortest path problem, which is known to be $\mathcal{NP}$-hard [6], to a deterministic version of the SPRT (a deterministic shortest path problem with a deadline).

Proposition 3.5. *The deterministic shortest path problem with a deadline is $\mathcal{NP}$-hard.*

Proof. Consider an instance of the weight constrained shortest path problem on a graph $G = (V, A)$, with source and sink nodes $s, t \in V$, arc costs $c' \in \mathbb{R}^{|A|}$, arc weights $w \in \mathbb{N}^{|A|}$, and let $\kappa \in \mathbb{N}$ be the maximum allowed total weight of a path. Costs c' do not form negative-cost cycles in G.

Define $\bar{c}' = \max_{a \in A} |c'_a|$, and let $c = c'/(\bar{c}'|V|)$. Each arc $a \in A$ has travel time w_a. Then, (G, s, t, c, w, κ) defines an instance of the shortest path problem with a deadline. Note that costs c do not form negative-cost cycles in G.

Any feasible path P for the weight constrained shortest path problem is feasible for the shortest path problem with a deadline. Conversely, any path P with total travel time less than or equal to the deadline is feasible for the weight constrained shortest path problem.

Note that for any simple path P we have that $|P| < |V|$, and thus $c(P) < 1$ for all paths P. Consider any path with total travel time greater than the deadline. By integrality of w and κ, such a path incurs a penalty of at least one, so the objective value of such a path for the shortest path problem with a deadline is greater than one. Note that c is proportional to c'. Hence, if an optimal solution P^* of the shortest path problem with a deadline has objective value strictly less than one, then P^* is optimal for the weight constrained shortest path problem. Otherwise, if the optimal value of the shortest path problem with a deadline is greater than one, then the weight constrained shortest path problem is infeasible. □

By choosing $\mathcal{P}\{\xi_a(\omega) = w_a\} = 1$ for all $a \in A$, it follows that the SPRT is $\mathcal{NP}$-hard.

3.5 Associated Mean Value Problems

Given a stochastic programming problem, its associated *mean value problem* (MVP) is obtained by replacing all random variables by their means (see e.g. Birge and Louveaux [4]). The mean value problem is a deterministic problem, which can be solved using deterministic optimization algorithms, and usually this is much easier than solving a stochastic optimization problem. In the case of the SPRT and the TSPRT the MVP will always yield (first and second stage) feasible solutions as long as they exist. In the case of the SPAF, the MVP does not yield a sensible model due to the integrality constraints.

Here we show that in the case of the SPRT, the mean value problem can have an optimal solution that is arbitrarily bad compared with an optimal solution of the stochastic problem. A similar construction can be used to show that the mean value problem can have arbitrarily bad solutions in the case of the TSPRT.

Proposition 3.6. *There are instances of the SPRT in which the difference and the ratio between the expected cost of an optimal solution of the MVP and the optimal cost of the stochastic problem are arbitrarily large.*

Proof. Consider the graph $G = (V, A)$, where $V = \{s, t\}$, $A = \{a_1, a_2\}$, with $a_1 = a_2 = (s, t)$. Thus there are two distinct $s - t$ paths in G, with x^1 using arc a_1 and x^2 using arc a_2. Choose $\kappa > 0$ and $b \geq \kappa$. Let $c_{a_1} = 1$ and $c_{a_2} = 2$. Let $\mathcal{P}[\xi_{a_1}(\omega) = b] = \kappa/b$, $\mathcal{P}[\xi_{a_1}(\omega) = 0] = 1 - \kappa/b$, and $\mathcal{P}[\xi_{a_2}(\omega) = \kappa] = 1$. Note that

$$\mathbb{E}[\xi_{a_1}(\omega)] \;=\; \mathbb{E}[\xi_{a_2}(\omega)] \;=\; \kappa. \tag{18}$$

Also,

$$\begin{aligned} c_{a_1} + Q(x^1, \mathbb{E}[\xi(\omega)]) &= 1 \\ c_{a_2} + Q(x^2, \mathbb{E}[\xi(\omega)]) &= 2 \end{aligned}$$

Thus x^1 is the optimal solution of the MVP. However,

$$\begin{aligned} z(x^1) &= c_{a_1} + \mathbb{E}[Q(x^1, \xi(\omega))] = 1 + (b - \kappa)\kappa/b \\ z(x^2) &= c_{a_2} + \mathbb{E}[Q(x^2, \xi(\omega))] = 2. \end{aligned}$$

The result follows by choosing κ sufficiently large and $b \gg \kappa^2$. □

We conclude this section by observing that there are instances of the SPRT and the TSPRT for which optimal solutions of the MVP are optimal or almost optimal for the stochastic problem. When the deadline is either very loose or very tight, the recourse function is linear in the random variables near optimal solutions. By linearity of expectation, an optimal solution of the MVP is then an optimal solution of the stochastic problem.

4 Algorithmic Details

Here we complete the description of our branch-and-cut algorithm for solving the SAA problems for the three problem classes. Section 4.1 addresses initial relaxations and heuristics. Sections 4.2 and 4.3 discuss the separation of the optimality cuts.

4.1 Initial Relaxations and Heuristics

As mentioned before, we initialize our branch-and-cut framework by solving the LP-relaxation of formulation (12), without the optimality cuts (12b). For the SPRT, and for the SPAF, this means that we start with only the flow conservation constraints (13b), and (14b), respectively, and the bounds $0 \leq x \leq 1$. In the case of the TSPRT, this means we start with only the degree constraints (17b) and the bounds $0 \leq x \leq 1$, and we add subtour elimination constraints (17c) whenever they are violated. For all three problem classes we generate optimality cuts if and only if the LP relaxation at hand has an integer solution. For the SPRT and the TSPRT, the optimality cuts and their generation are discussed in detail in Section 4.2. Section 4.3 does the same for the SPAF.

The generation of subtour elimination constraints for the TSPRT reduces to finding a global minimum cut in a directed graph derived from the support of a fractional solution [26, 27]. For this we use Hao and Orlin's implementation [10] of the Gomory-Hu algorithm [9].

Solving the MIPs for the TSPRT turned out to be quite a computational challenge. For this reason, we decided to limit the number of nodes in the branch-and-bound tree to 10000. Note that the values of z_N and z_N^m from Section 2.1 are then taken to be the best lower bound obtained upon termination of the branch-and-cut algorithm. We also added a rounding heuristic to the

branch-and-cut algorithm, which is called occasionally starting from intermediate LP solutions. The heuristic tries to find a tour that is cheap with respect to the expected recourse in the support of the fractional solution, using edges that are not in the support only if they are needed to get a feasible solution. This is accomplished using the Lin-Kerninghan algorithm (see e.g. Johnson and McGeoch [15]).

4.2 Optimality Cut Generation for the SPRT and the TSPRT

Note that for the models with random travel times (the SPRT and the TSPRT), the second-stage recourse problem for sample scenario ω^n, with $n \in \{1, 2, \ldots, N\}$, is given by

$$Q_n(x) = \max\left(\xi(\omega^n)^T x - \kappa, 0\right).$$

It follows that

$$\sum_{n=1}^{N} Q_n(x) = \max\left\{\sum_{n\in S}(\xi(\omega^n)^T x - \kappa) \mid S \subseteq \{1, \ldots, N\}\right\},$$

where S is a subset of N. Hence, in the case of the SPRT, and the TSPRT, the sample average approximating problem is equivalent to the mixed-integer linear problem

$$\begin{aligned}
\min\ & c^T x + \theta/N && && (19a)\\
\text{subject to } & Ax \ \leq\ b && && (19b)\\
& \theta \ \geq\ \sum_{n\in S}(\xi(\omega^n)^T x - \kappa) && \text{for all } S \subseteq \{1, 2, \ldots, N\} && (19c)\\
& x \text{ binary}, && && (19d)
\end{aligned}$$

where the inequalities (19c) are the optimality cuts, and the system $Ax \leq b$ represents the constraints (13b), and (17b)–(17c), respectively. Note that (19c) represents a huge number of constraints. However, for any feasible first-stage solution in the course of our branch-and-cut iterations, we only add the maximally violated optimality cut. The maximally violated optimality cut at a feasible first-stage solution x corresponds to the subset $S(x) = \{n \in \{1, 2, \ldots, N\} \mid \xi(\omega^n)^T x > \kappa\}$.

4.3 Optimality Cut Generation for the SPAF

Assume that the conditions of Propositions 3.2 and 3.4 hold. Let $Q_n(x) = Q(x, \xi(\omega^n))$ be the optimal value of problem (15a), (15b), (16). By associating dual variables $\pi^n \in \mathbb{R}^{|V|}$, and $\gamma^n, \tau^n \in \mathbb{R}^{|A\cup A^{-1}|}$, with the flow conservation, lower, and upper bound constraints, respectively, we obtain the following LP dual of the recourse function

$$\begin{aligned}
Q_n(x) = \max\ & l(x, u(\omega^n))^T \gamma^n + r(x, u(\omega^n))^T \tau^n && && (20a)\\
\text{subject to } & \pi^n_i - \pi^n_j + \gamma^n_{(i,j)} + \tau^n_{(i,j)} \leq c_{(i,j)} && \text{for all } (i,j) \in A, && (20b)\\
& \pi^n_j - \pi^n_i + \gamma^n_{(i,j)^{-1}} + \tau^n_{(i,j)^{-1}} \leq c_{(i,j)^{-1}} && \text{for all } (i,j)^{-1} \in A^{-1}, && (20c)\\
& \gamma^n \geq 0, \tau^n \leq 0. && && (20d)
\end{aligned}$$

The optimal dual multipliers from (20) then provide the optimality cut

$$\theta \geq \sum_{n=1}^{N} l(x, u(\omega^n))^T \gamma^n + r(x, u(\omega^n))^T \tau^n. \qquad (21)$$

Recall that l and r are linear functions of x. We can simplify the expression for the cut by substituting the definitions of l and r, eliminating terms that evaluate to 0, and introducing the term $\alpha : \mathbb{R}^{|A \cup A^{-1}|} \times \mathbb{R}^{|A \cup A^{-1}|} \to \mathbb{R}^{|A|}$ defined as,

$$\alpha_a(\gamma^n, \tau^n, \omega^n) = \begin{cases} \tau^n_{a^{-1}} - \tau^n_a, & \text{if } u_a(\omega^n) = 1, \\ \tau^n_{a^{-1}} + \gamma^n_{a^{-1}}, & \text{if } u_a(\omega^n) = 0. \end{cases} \tag{22}$$

The optimality cut (21) can then be expressed as

$$\sum_{n=1}^{N} \alpha(\gamma^n, \tau^n)^T x \leq \theta - \sum_{n=1}^{N} \sum_{a \in A^n} \tau^n_a. \tag{23}$$

The resulting SAA problem for the SPAF is

$$\begin{aligned} &\min\ c^T x + \theta/N \\ &\text{subject to (14b), and (23) for all } (\pi^n, \gamma^n, \tau^n) \text{ satisfying (20b)–(20d), with } n \in \{1, 2, \ldots, N\} \\ &\qquad x \in \{0,1\}^{|A|}. \end{aligned}$$

The generation of the optimality cuts for the SPAF is implemented by maintaining one linear program of the form (15a), (15b), (16), which we will refer to as the *recourse LP* in the following discussion. Suppose we are given an integer solution x to the first stage of the SPAF. To generate an optimality cut, we iterate over the scenarios. Focus on any $n \in \{1, 2, \ldots, N\}$. First, we regenerate ω^n from a stored seed value. Next, we impose the bounds $l(x, u(\omega^n))$ and $r(x, u(\omega^n))$ on the recourse LP. Then, we use the dual simplex method to solve it, starting from the optimal dual associated with the previous recourse LP. In this way, we exploit similarities between consecutive recourse LPs. This could be further accelerated by techniques such as sifting [7] or bunching [41] that identify scenarios for which a particular recourse basis is optimal. However, we have not used such strategies in our implementation. We iteratively compute optimality cuts of the form (23) in each iteration by adding the terms that involve the optimal dual solution to the current recourse LP. Because the constraint matrix induced by the constraints (15b), (16) is totally unimodular (see e.g. Schrijver [31]), the recourse LPs yield integer optimal solutions (Corollary 3.3).

Note that because of the possibility of regenerating the random part of the constraint matrix, it is not necessary to have the entire constraint matrix in memory at any time during the execution of the algorithm. This aspect is of great practical significance, as it allows us to use large values of N. In this way we can solve SAAs that have a huge number of non-zeros in the constraint matrix.

5 Computational Results

The algorithmic framework of Section 4 has been implemented in C++ using the MIP solver of CPLEX 7.0 [13]. The implementation takes advantage of the callback functionality provided by CPLEX to integrate separation algorithms and heuristics within the branch-and-bound algorithm. This section presents our computational experiments. Section 5.1 discusses the generation of the problem instances and Section 5.2 provides the computational results. We only report on a subset of our experiments; tables with the full output can be found in Appendix A. The CPU times reported in this section were observed on 350MHz and 450MHz Sun Sparc workstations.

5.1 Generation of Test Instances

An instance of the SPRT is defined by a directed graph $G = (V, A)$, a source and sink node, a cost vector c, a probability distribution over the travel times on the arcs in G, and a deadline. Our approach is to generate instances from the TSP library [29] as follows.

A TSP library instance defines a set of p cities, $\{1, 2, \ldots, p\}$, together with an inter-city distance matrix $D = [d_{ij} \mid i, j \in \{1, 2, \ldots, p\}]$. We associate a node v_i in V with city i. Next, we iterate over the nodes in G. At iteration i, we connect node v_i to the δ closest nodes that it has not already been connected to in an earlier iteration. Connecting node v to node w is done by adding the arcs (v, w) and (w, v) to A. The cost of arc (v_i, v_j) is taken to be $c_{(v_i,v_j)} = d_{ij}$. To choose a source and sink node, we first find the set of pairs (u, v) with $u, v \in V$ that maximize the minimum number of arcs over all $u - v$ paths. From this set we choose one pair uniformly at random to be the source and sink. In order to generate instances for which the associated MVP is unlikely to yield optimal solutions, we use a probability distribution which discourages the use of short arcs. For $a \in A$ we let

$$\mathcal{P}\{\xi_a(\omega) = Fc_a\} = \begin{cases} p, & \text{if } c_a < \bar{c}, \\ 0, & \text{if } c_a \geq \bar{c}, \end{cases} \qquad \text{and} \qquad \mathcal{P}\{\xi_a(\omega) = c_a\} = \begin{cases} 1-p & \text{if } c_a < \bar{c}, \\ 1, & \text{if } c_a \geq \bar{c}, \end{cases}$$

where $\bar{c}$ denotes the median of the arc lengths, for some parameters $F > 1$ and $0 < p < 1$. Finally, the deadline κ is determined by computing a shortest path in G with respect to c, and taking the deadline to be the expected travel time on this path.

An instance of the SPAF is defined by a directed graph G, a source and sink node, arc costs c^1 and $c^2(\omega)$, and a probability distribution over $\xi(\omega) = (u(\omega), c^2(\omega))$. The graph G, the source and sink, and c^1 are generated as above from TSP library instances. For all arcs $a \in A$ we let $\mathcal{P}\{c_a^2(\omega) = c_a^1\} = 1$, and for all arcs $a^{-1} \in A^{-1}$ we let $\mathcal{P}\{c_a^2(\omega) = 0\} = 1$. The distribution we use for the upper bounds $u(\omega)$ is similar in spirit to the one used for the SPRT, namely,

$$\mathcal{P}\{u_a(\omega) = 0\} = \begin{cases} p, & \text{if } c_a < \bar{c}, \\ 0, & \text{if } c_a \geq \bar{c}, \end{cases} \qquad \text{and} \qquad \mathcal{P}\{u_a(\omega) = 1\} = \begin{cases} 1-p & \text{if } c_a < \bar{c}, \\ 1, & \text{if } c_a \geq \bar{c}, \end{cases}$$

with $\bar{c}$ and p as above. Note that assumptions of Propositions 3.2 and 3.4 are satisfied. Relatively complete recourse is ensured by inserting an arc (s, t) with very high cost.

An instance of the TSPRT is defined by an undirected graph $G = (V, E)$, a cost vector $c \in \mathbb{R}^{|E|}$, a distribution over the travel times, and a deadline. The graph G is generated as above, except that connecting nodes v and w is done by inserting an edge $\{v, w\}$ in E. The cost of edge $\{v_i, v_j\}$ is taken to be $c_{\{v_i,v_j\}} = d_{ij}$. The probability distribution of the travel time on edge $e \in E$ is defined by

$$\mathcal{P}\{\xi_e(\omega) = Fc_e\} = \begin{cases} p, & \text{if } c_e < \bar{c}, \\ 0, & \text{if } c_e \geq \bar{c}, \end{cases} \qquad \text{and} \qquad \mathcal{P}\{\xi_e(\omega) = c_e\} = \begin{cases} 1-p & \text{if } c_e < \bar{c}, \\ 1, & \text{if } c_e \geq \bar{c}, \end{cases}$$

where $\bar{c}$ denotes the median of the edge lengths, and $F > 1$ and $0 < p < 1$ are parameters. The deadline is taken to be the expected travel time on a shortest Hamiltonian cycle in G with respect to c.

In our experiments, we have used the following parameter settings for the probability distribution: $F = 10$ for the SPRT, $F = 20$ for the TSPRT, and $p = .1$ for all problems. We used $\delta = 10$ for generating graphs. We have used the following parameters for solving the SAAs (see Section 2.1): $M = 10$, and $N' = 10^5$. The dimensions of the graphs underling our experiments are summerized in Table 1.

name	$\|V\|$	$\|A\|$	$\|E\|$	name	$\|V\|$	$\|A\|$	$\|E\|$	name	$\|V\|$	$\|A\|$	$\|E\|$
burma14	14	172	86	eil76	76	1412	706	kroC100	100	1892	946
ulysses16	16	212	106	pr76	76	1412	706	kroD100	100	1892	946
ulysses22	22	332	166	gr96	96	1812	906	kroE100	100	1892	946
eil51	51	912	456	rat99	99	1872	936	rd100	100	1892	946
berlin52	52	932	466	kroA100	100	1892	946	eil101	101	1912	956
st70	70	1292	646	kroB100	100	1892	946				

Table 1: Dimensions of graphs generated from TSP library instances.

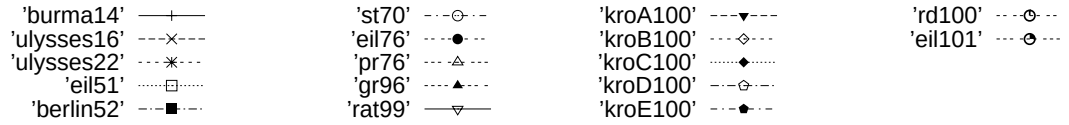

Figure 1: The key to Figures 2–10.

5.2 Computational Results

Figures 2, 5, and 8 give the average number of optimality cuts as a function of the sample size N for the SPRT, the SPAF, and the TSPRT, respectively. Figures 3, 6, and 9 give the average number of nodes in the branch-and-bound tree for the SPRT, the SPAF, and the TSPRT, respectively. Figures 4, 7, and 10 give the average CPU time (in seconds) for the SPRT, the SPAF, and the TSPRT, respectively. In these figures, each graph represents the execution of the SAA method with sample sizes $N \in \{200, 300, \ldots, 1000\}$ for a particular problem instance derived from a TSP library instance. The key is given in Figure 1. For each combination of a base instance and a sample size, the reported averages are taken over the M SAAs. Table 2 gives solution values, optimality gaps, and their standard deviations for all of the problems, as they were observed with $N = 1000$. Note, that some of the gaps in Table 2 are negative. As both the lower bound and the upper bound used to compute the gaps are random variables, and no gap is smaller than minus twice its estimated standard deviation, this is not unexpected. Also note from Table 2 that the MVP rarely yields a competitive solution for the generated instances.

In Figures 2, 5, and 8 we see that the average number of optimality cuts displays the same behavior for all generated instances of the SPRT, the SPAF, and the TSPRT. For each of the three problems under consideration, the average number of optimality cuts does not depend on the sample size used to formulate the SAA. From Figures 3, 6, and to a lesser extent from Figure 9 (for the TSPRT, all runs needed 10000 nodes in the branch-and-bound tree, except for the runs on the three smallest instances), it is clear that the same observation holds for the average number of nodes in the branch-and-bound tree. As a consequence, the running time grows only linearly in the sample size. This linear growth is clearly visible in Figures 4 and 7. From Figure 10 we conclude that for the TSPRT the running time is dominated by the branch-and-bound algorithm itself for $N \leq 1000$, therefore the average time spent in the generation of the optimality cuts is still a lower-order term in the average CPU time.

From Table 2 it is clear that the SAA produces provably close to optimal solutions. The reported solutions were within an estimated 1.0%, 0.3%, and 8.7% from the lower bounds for the SPRT, the SPAF, and the TSPRT, respectively (for $N = 1000$). Also, the estimated standard deviations of these estimated gaps are small; they are of the same order of magnitude as the estimated gaps

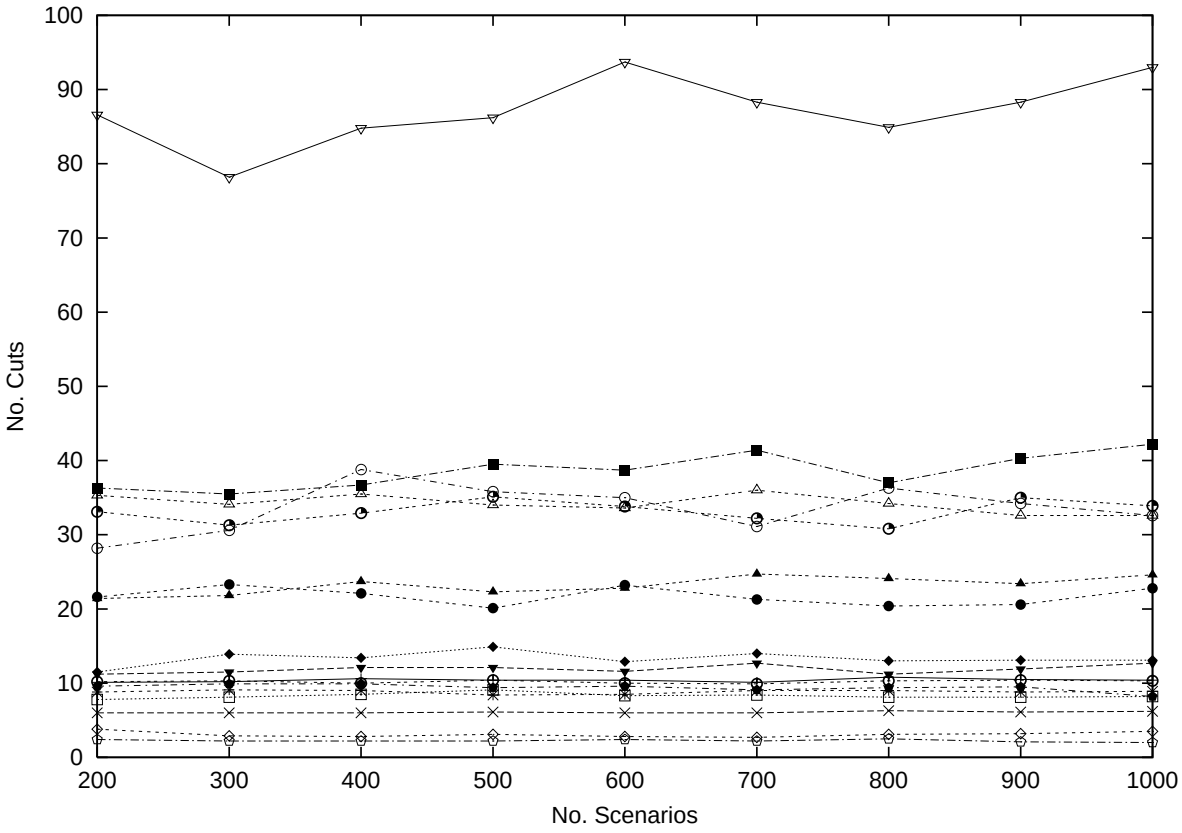

Figure 2: Average number of optimality cuts for the SPRT.

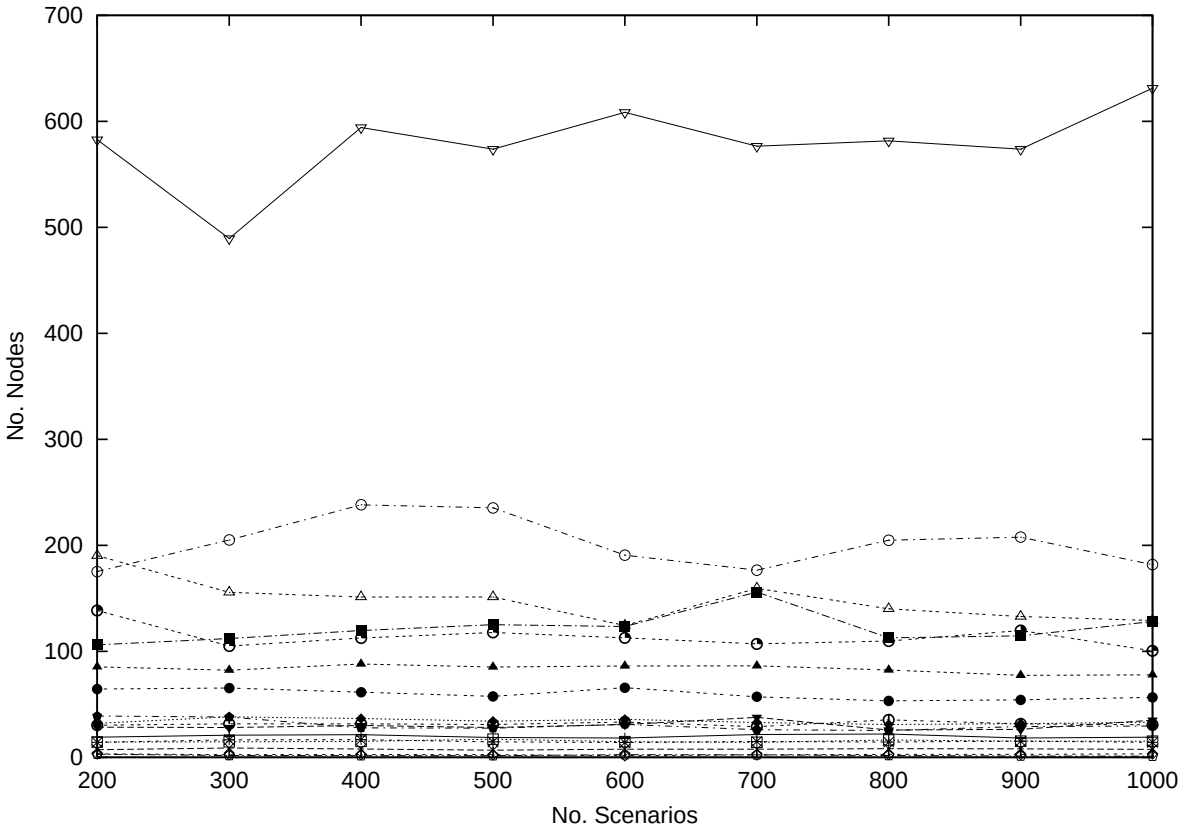

Figure 3: Average number of nodes in branch-and-bound tree for the SPRT.

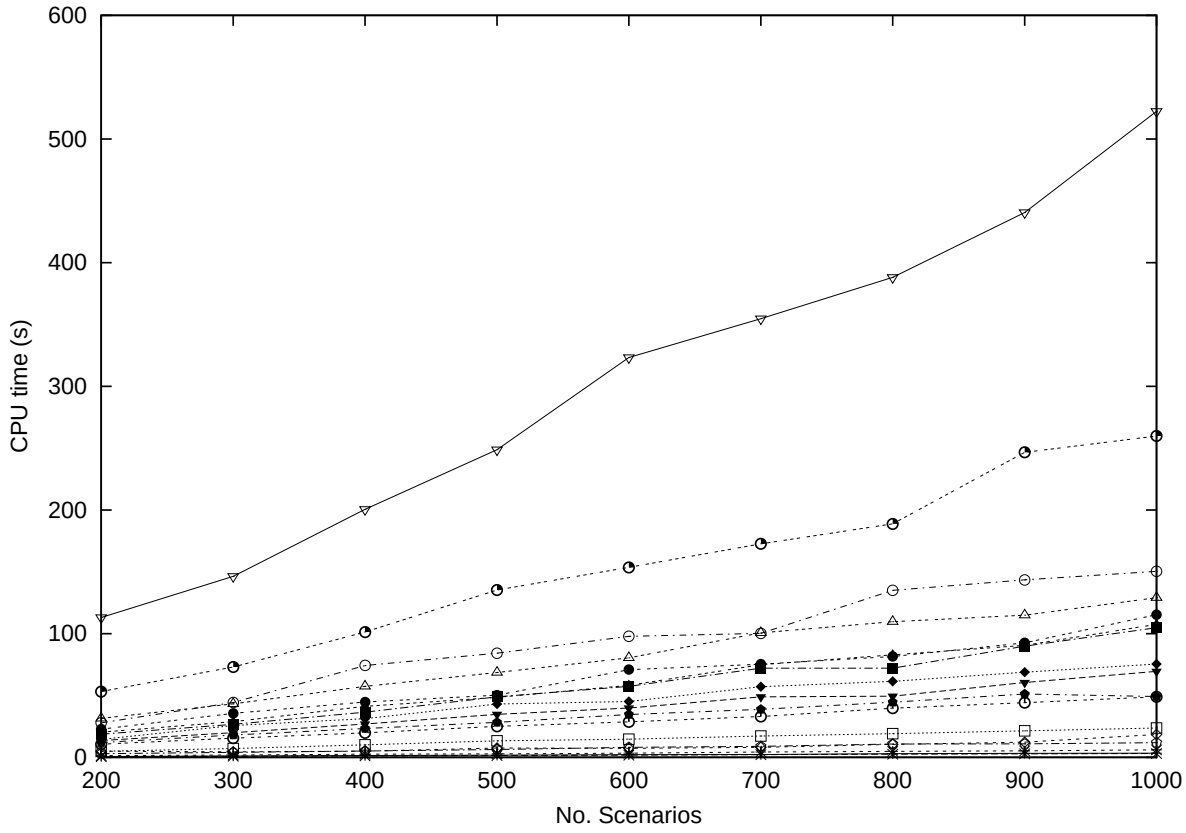

Figure 4: Average CPU times for the SPRT.

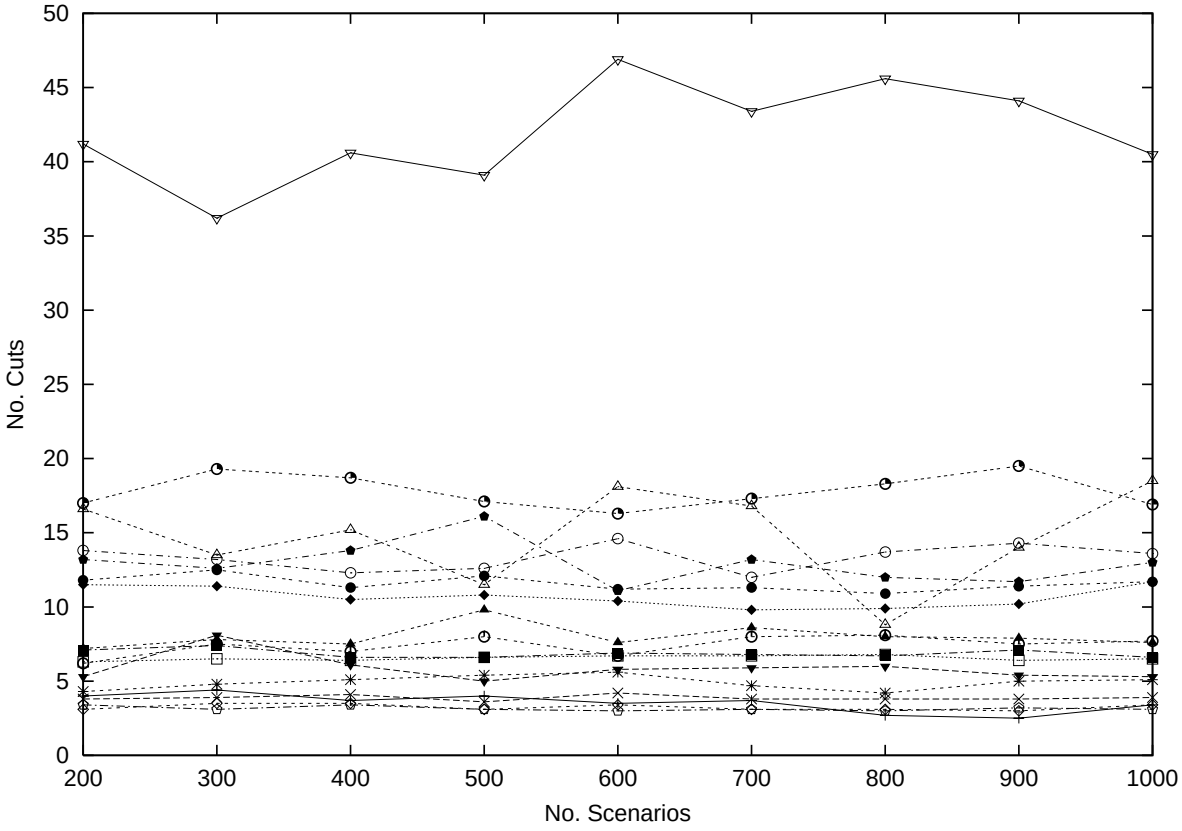

Figure 5: Average number of optimality cuts for the SPAF.

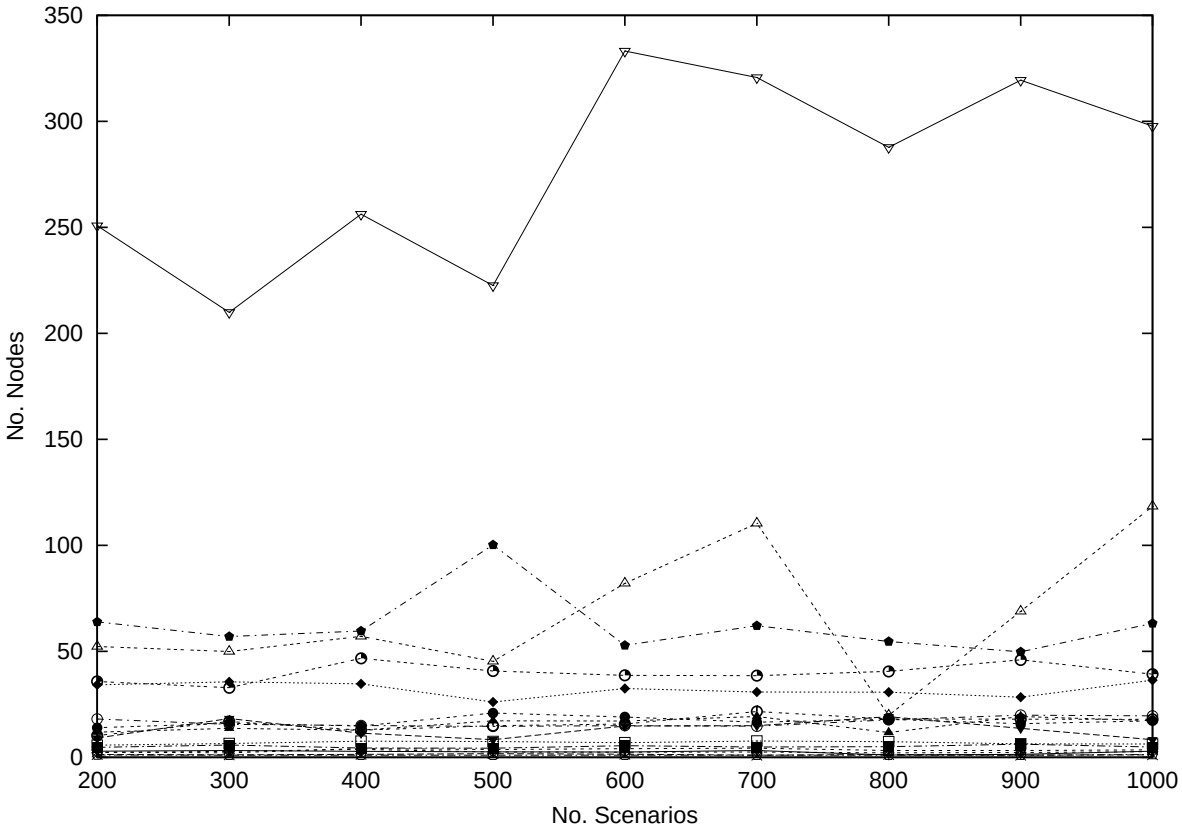

Figure 6: Average number of nodes in branch-and-bound tree for the SPAF.

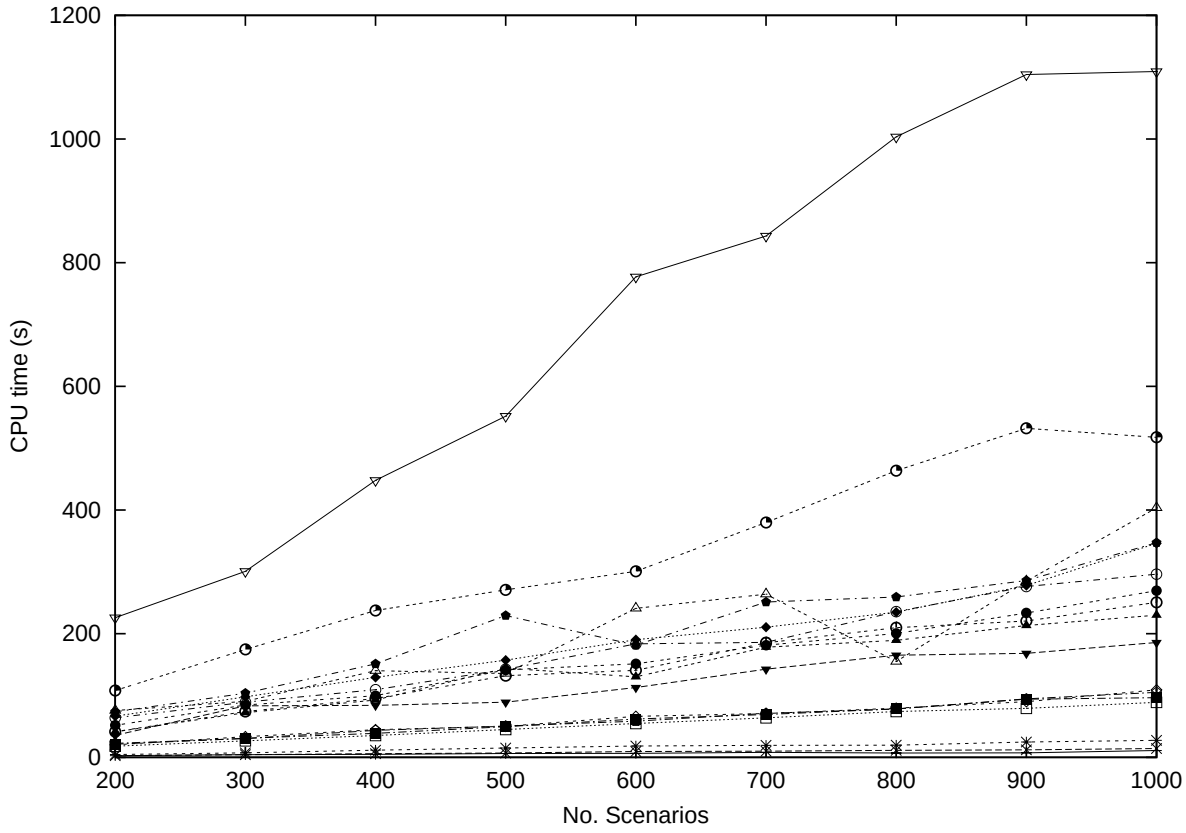

Figure 7: Average CPU times for the SPAF.

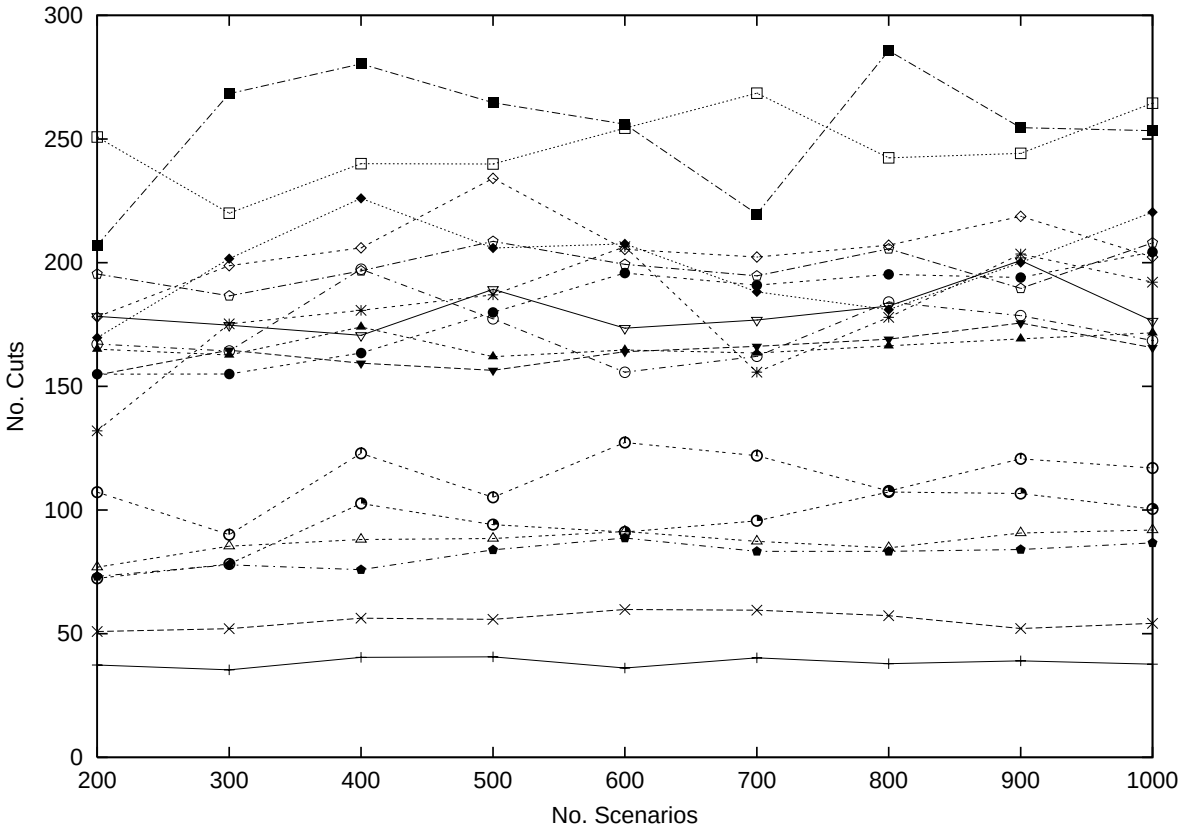

Figure 8: Average number of optimality cuts for the TSPRT.

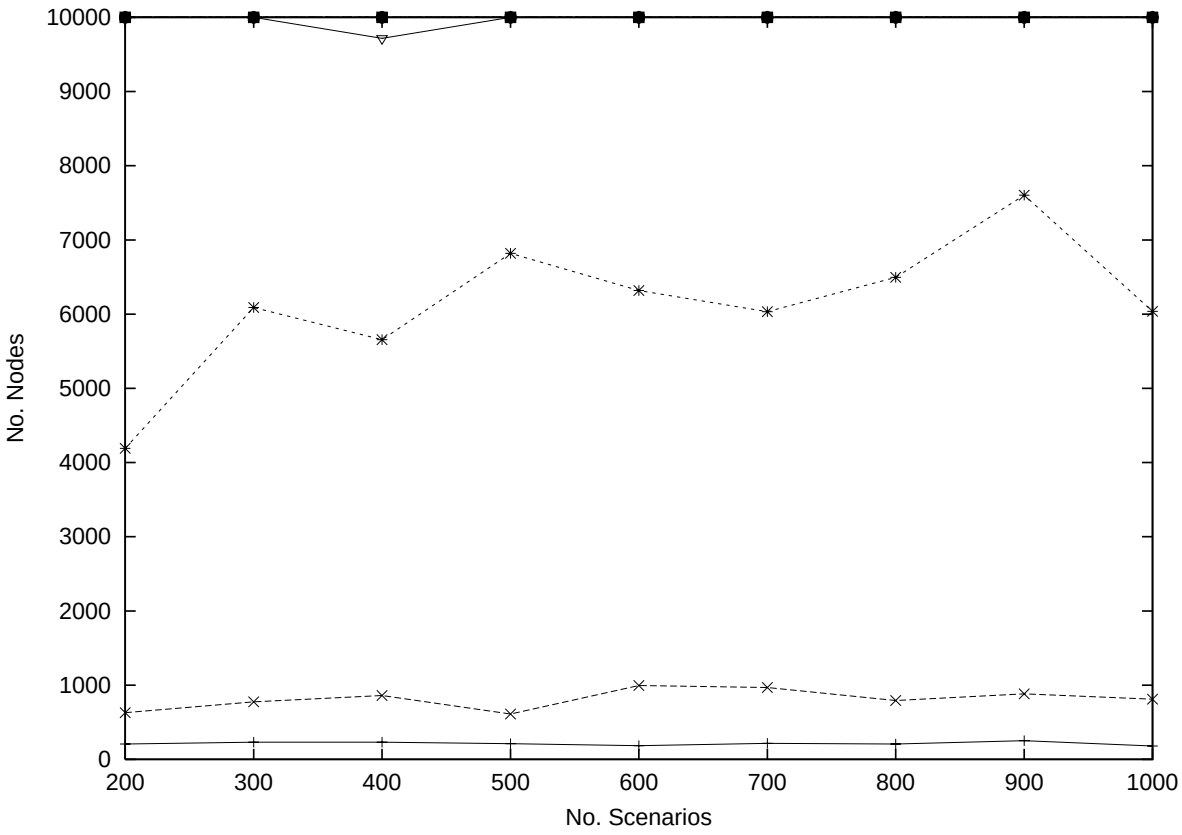

Figure 9: Average number of nodes in branch-and-bound tree for the TSPRT.

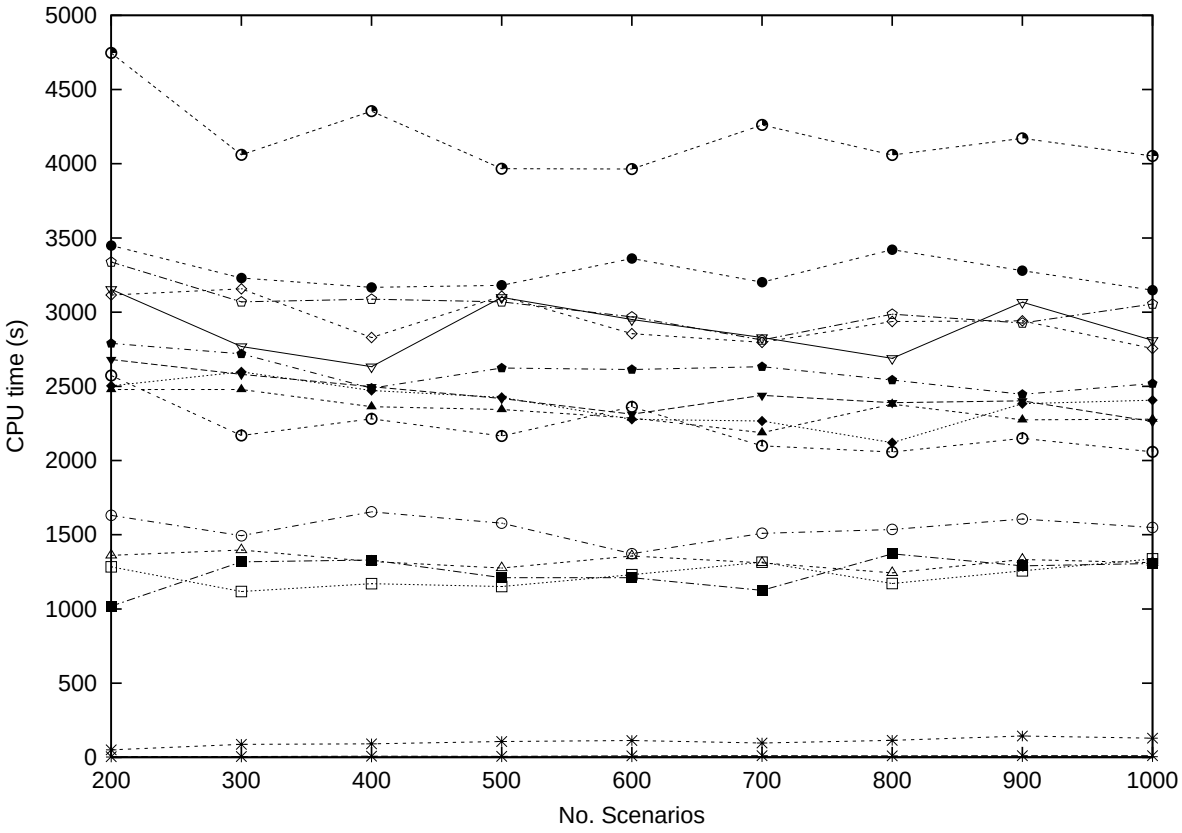

Figure 10: Average CPU times for the TSPRT.

	SPRT						SPAF			TSPRT					
	SAA			MVP			SAA			SAA			MVP		
instance	$\hat{z}_{10^5}(\hat{x}^\star)$	Gap	$\hat{\sigma}_{\text{Gap}}$	$\hat{z}_{10^5}(\hat{x}^\star)$	Gap	$\hat{\sigma}_{\text{Gap}}$	$\hat{z}_{10^5}(\hat{x}^\star)$	Gap	$\hat{\sigma}_{\text{Gap}}$	$\hat{z}_{10^5}(\hat{x}^\star)$	Gap	$\hat{\sigma}_{\text{Gap}}$	$\hat{z}_{10^5}(\hat{x}^\star)$	Gap	$\hat{\sigma}_{\text{Gap}}$
burma14	371.0	1.2	1.09	412.3	42.5	0.99	320.8	-0.6	0.43	1562.7	7.2	3.18	2129.6	574.0	4.12
ulysses16	2510.0	0.0	0.00	2642.1	132.1	2.82	2382.2	0.6	1.09	9506.2	34.7	24.30	11455.6	1984.1	17.47
ulysses22	1442.4	8.6	3.83	1623.0	189.2	3.33	1307.1	0.8	0.83	10000.9	80.0	28.68	10577.5	656.6	11.47
eil51	61.7	0.6	0.28	61.8	0.7	0.10	51.2	0.0	0.07	543.9	17.9	1.80	548.3	22.3	0.68
berlin52	839.7	-1.9	1.00	1240.7	399.2	2.87	812.5	1.6	0.73	9481.0	308.5	21.87	10202.0	1029.6	12.33
st70	114.9	0.3	0.51	143.3	28.7	0.34	102.7	0.2	0.19	834.5	66.9	2.48	852.8	85.2	1.01
eil76	49.2	0.1	0.21	64.3	15.2	0.12	46.5	0.1	0.05	661.0	30.0	1.55	674.1	43.1	0.74
pr76	18207.0	0.0	0.00	25214.3	7007.3	37.90	18463.5	-0.5	8.60	124419.0	9315.0	188.09	135868.0	20764.0	139.73
gr96	7601.0	0.0	0.00	8719.3	1118.3	10.30	7422.6	0.9	3.98	62837.4	2328.4	56.58	68136.5	7627.5	63.70
rat99	205.8	0.4	0.12	269.3	63.9	0.39	207.0	0.0	0.14	1464.4	47.4	2.50	1474.0	57.0	1.39
kroA100	3405.4	20.7	10.76	3657.3	272.6	5.07	3022.8	-2.0	2.10	25775.1	1233.8	65.33	26563.3	2022.0	26.24
kroB100	2468.2	-3.4	4.82	2469.4	-2.3	1.47	2366.6	2.4	1.45	26411.4	1862.4	50.90	27773.6	3224.6	27.54
kroC100	2941.0	0.6	1.39	3136.0	195.6	4.38	2649.3	0.5	3.27	24660.3	950.3	40.60	26129.0	2419.0	26.84
kroD100	2490.0	0.0	0.00	2598.4	108.4	2.44	2382.9	0.6	1.70	25468.7	1350.0	40.89	26802.9	2684.2	27.55
kroE100	3708.3	4.3	2.45	4014.2	310.2	3.51	3532.4	3.7	2.03	25775.5	1975.1	40.51	27799.1	3998.7	28.59
rd100	1197.0	0.0	0.00	1393.5	196.5	1.71	1166.5	0.4	0.45	9478.4	439.3	18.04	9856.2	817.1	9.84
eil101	52.2	-0.1	0.16	60.1	7.8	0.12	44.5	0.0	0.05	763.1	41.9	2.45	758.3	37.1	0.74

Table 2: Solution values and absolute gaps with lower bound.

themselves for the SPRT and the SPAF, and for the TSPRT they are significantly smaller. In the case of the TSPRT, a significant part of these estimated gaps can be attributed to the fact that the MIPs are not solved to optimality; the instance that realized the forementioned gap of 8.7% had an average (MIP) gap between the best integer solution and the best lower bound of 8.17% over the M SAAs. We believe that by taking advantage of sophisticated cutting-plane and branching techniques as applied by Applegate *et al.* [3], these SAA problems can be solved to optimality for the instances we considered, thereby reducing the gap to that caused by the quality of the stochastic bounds.

From the tables in Appendix A, it is clear that as N increases, the estimated values of the reported solutions does not change much. For the SPRT and the TSPRT, the stochastic lower bounds improve, for the SPAF there is not much change in the lower bounds either. In all cases the estimated variance goes down as N increases.

6 Stability of the Number of Optimality Cuts

It was observed empirically that the *average* number of optimality cuts generated when solving an SAA problem is fairly stable and hardly changes with increase in the sample size N. In this section we suggest an explanation of this empirical phenomenon.

Denote by $z(x)$ and $\hat{z}_N(x)$ the objective functions of the true problem (1) and the SAA problem (4), respectively. Note that for our problem classes

$$\hat{z}_N(x) = \frac{1}{N}\sum_{n=1}^{N} h(x, \omega^n), \tag{24}$$

where $h(x,\omega) = c^T x + Q(x, \xi(\omega))$, and that for every ω the function $h(x,\omega)$ is piecewise linear and convex in x. Thus both functions $z(x)$ and $\hat{z}_N(x)$ are convex, and the function $\hat{z}_N(x)$ is piecewise linear.

Since $\hat{z}_N(x)$ is convex, and piecewise linear, its subdifferential $\partial\hat{z}_N(x)$ is a convex polyhedron. To every element of $\partial\hat{z}_N(x)$, called a *subgradient*, there corresponds an optimality cut at the point x. It should be noted that our algorithm uses cuts corresponding to the *extreme* points of $\partial\hat{z}_N(x)$. Also note that z is differentiable at x if and only if $\partial z(x)$ is a singleton.

By convexity of $h(x,\omega)$ in x we have that, for a given point x, the subdifferential $\partial\hat{z}_N(x)$ converges with probability one (w.p.1) to $\partial z(x)$ as $N \to \infty$, that is

$$\lim_{N\to\infty} \mathcal{H}\left(\partial\hat{z}_N(x), \partial z(x)\right) = 0, \qquad \text{w.p.1} \tag{25}$$

(see Shapiro and Homem-de-Mello [34, Proposition 2.2]). Here $\mathcal{H}$ measures the Hausdorff distance between two sets. In particular, if $\partial z(x) = \{\nabla z(x)\}$ is a singleton, then any element of $\partial\hat{z}_N(x)$ converges to $\nabla z(x)$ w.p.1.

Suppose, further, that $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ is *finite*, i.e., the total number of scenarios is finite. Then the true objective function $z(x)$ is also piecewise linear, so that the linear space of vectors x can be partitioned into a finite number of convex polyhedral sets C_ℓ, $\ell = 1, 2, \ldots, L$, such that the function z is affine on every set C_ℓ. Moreover, it was shown [34] that if the function z is affine on a convex polyhedral set C, then $h(x,\omega)$ is also affine on C for every scenario ω. That is, every function $h_k(x) = h(x,\omega_k)$, $k = 1, 2, \ldots, K$, is affine on each set C_ℓ, $\ell = 1, 2, \ldots, L$. Of course, h_k can be affine on a union of a number of sets C_ℓ.

Since $\hat{z}_N$ is a linear combination of functions h_k (with some coefficients equal to a positive integer multiple of $1/N$ and some zeros), it follows that: (i) for any random sample, the SAA

function $\hat{z}_N$ is also affine on every C_ℓ, and for any x (ii) the subdifferentials $\partial z(x)$ and $\partial \hat{z}_N(x)$ are bounded convex polyhedra, (iii) there is a one-to-one correspondence between the extreme points of $\partial \hat{z}_N(x)$ and a subset of the extreme points of $\partial z(x)$. It follows from (iii) that the number of extreme points of $\partial \hat{z}_N(x)$ is always less than or equal to the number of extreme points of $\partial z(x)$. Since it may happen that $\hat{z}_N$ is affine on a union of some sets C_ℓ, the number of extreme points of $\partial \hat{z}_N(x)$ can be smaller than the number of extreme points of $\partial z(x)$. However, it follows from equation (25) that the extreme points of $\partial \hat{z}_N(x)$ converge w.p.1 to the extreme points of $\partial z(x)$ as $N \to \infty$. That is, in the limit, $\partial \hat{z}_N(x)$ and $\partial z(x)$ have the same number of extreme points close to each other.

In our algorithm, optimality cuts are obtained by computing extreme subgradients of the SAA function $\hat{z}_N$ at feasible points $x \in X$. These solutions satisfy all integrality constraints. The key observation is that since the set X is finite, the number of required optimality cuts does not change if at each iteration the subgradients of z are slightly perturbed. By the above convergence analysis we obtain that w.p.1 for N large enough the number of optimality cuts of a SAA problem should be equal to the number of optimality cuts required for the true problem.

If the subdifferentials $\partial z(x)$, of the true objective function, are singletons at each feasible (iteration) point x, then the number of optimality cuts required for the true problem is constant. In that case, w.p.1 for N large enough, the number of optimality cuts of SAA problems is equal to the number of optimality cuts of the true problem. That is, the number of optimality cuts of SAA problems converges w.p.1 to a constant as $N \to \infty$.

If $\partial z(x)$ are not singletons at some of the iteration points, then the number of optimality cuts of the true problem may depend on the chosen subgradients. In any case, for N large enough, the number of optimality cuts of a SAA problem should be equal to the corresponding number of optimality cuts of the true problem, and hence does not depend on N. It appears that the probability distribution of the number of optimality cuts of the generated SAA problems stabilizes as the sample size N increases. As a consequence the expected value of this number of optimality cuts converges to a constant. This in turn implies that the average estimate of the number of optimality cuts of the generated SAA problems stabilizes as the sample size N increases.

7 Conclusions

We have applied the sample average approximation method to three classes of 2-stage stochastic routing problems. Among the instances we considered were ones with a huge number of scenarios and first-stage integer variables. Our experience is that the growth of the computational effort needed to solve SAA problems is linear in the sample size used to estimate the true objective function. The reason for this is that in the cases that we examined, neither the number of optimality cuts, nor the sizes of branch-and-bound trees needed to solve SAA problems to optimality increase when the sample size increases. This allows us to find solutions of provably high quality to the stochastic programming problems under consideration.

References

[1] G. Andreatta. Shortest path models in stochastic networks. In G. Andreatta, F. Mason, and P. Serafini, editors, *Stochastics in Combinatorial Optimization*, pages 178–186. CISM Udine, World Scientific Publishing, Singapore, 1987.

[2] G. Andreatta and L. Romeo. Stochastic shortest paths with recourse. *Networks*, 18:193–204, 1988.

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. On the solution of traveling salesman problems. *Documenta Mathematica*, Extra Volume ICM 1998 III:645–656, 1998.

[4] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, Berlin, 1997.

[5] Y. Ermoliev and R.J-B. Wets, editors. *Numerical techniques for stochastic optimization*. Springer-Verlag, Berlin, 1988.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[7] S. Garstka and D. Rutenberg. Computation in discrete stochastic programmin with recourse. *Operations Research*, 21:112–122, 1973.

[8] C.J. Geyer and E.A. Thompson. Constrained Monte Carlo maximum likelihood for dependent data, (with discussion). *Journal of the Royal Statistical Society Series B*, 54:657–699, 1992.

[9] R.E. Gomory and T.C. Hu. Multi-terminal network flows. *SIAM Journal on Applied Mathematics*, 9:551–570, 1961.

[10] J. Hao and J.B. Orlin. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 165–174, 1992.

[11] J.L. Higle and S. Sen. *Stochastic Decomposition*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.

[12] J.L. Higle and S. Sen. Stochastic decomposition: An algorithm for two stage stochastic linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.

[13] ILOG, Inc., CPLEX Division, Incline Village, Nevada. *CPLEX, a division of ILOG*, 2001.

[14] G. Infanger. *Planning Under Uncertainty: Solving Large Scale Stochastic Linear Programs*. Boyd and Fraser Publishing Co., Danvers, MA, 1994.

[15] D.S. Johnson and L.A. McGeoch. The traveling salesman problem: a case study. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons Ltd, Chichester, 1997.

[16] A.J. Kleywegt, A. Shapiro, and T. Homem-de-Mello. The sample average approximation method for stochastic discrete optimization. To appear in *SIAM Journal on Optimization*, 2001.

[17] G. Laporte and F.V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[18] G. Laporte, F.V. Louveaux, and H. Mercure. Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research*, 39:71–78, 1989.

[19] G. Laporte, F.V. Louveaux, and H. Mercure. The vehicle routing problem with stochastic travel times. *Transportation Science*, 26:161–170, 1992.

[20] G. Laporte, F.V. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42:543–549, 1994.

[21] G. Laporte, F.V. Louveaux, and L. van Hamme. Exact solution of a stochastic location problem by an integer L-shaped algorithm. *Transportation Science*, 28:95–103, 1994.

[22] W.K. Mak, D.P. Morton, and R.K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24:47–56, 1999.

[23] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, 1988.

[24] V.I. Norkin, Y.M. Ermoliev, and A. Ruszczyński. On optimal allocation of indivisibles under uncertainty. *Operations Research*, 46:381–395, 1998.

[25] V.I. Norkin, G.Ch. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 83:425–450, 1998.

[26] M.W. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257, 1990.

[27] M.W. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[28] E.L. Plambeck, B.R. Fu, S.M. Robinson, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75:137–176, 1996.

[29] G. Reinelt. `http://softlib.rice.edu/softlib/tsplib`, 1995.

[30] R.Y. Rubinstein and A. Shapiro. Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation*, 32:373–392, 1990.

[31] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons Ltd, Chichester, 1986.

[32] A. Shapiro. Simulation-based optimization: Convergence analysis and statistical inference. *Stochastic Models*, 12:425–454, 1996.

[33] A. Shapiro and T. Homem-de-Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81:301–325, 1998.

[34] A. Shapiro and T. Homem-de-Mello. On rate of convergence of Monte Carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11:70–86, 2001.

[35] A.M. Spaccamela, A.H.G. Rinnooy Kan, and L. Stougie. Hierarchical vehicle routing problems. *Networks*, 14:571–586, 1984.

[36] R.M. Van Slyke and R.J-B. Wets. *L*-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[37] A.M. Verweij. *Selected Applications of Integer Programming: A Computational Study.* PhD thesis, Department of Computer Science, Utrecht University, Utrecht, 2000.

[38] S.W. Wallace. Solving stochastic programs with network recourse. *Networks*, 16:295–317, 1986.

[39] S.W. Wallace. Investing in arcs in a network to maximize the expected max flow. *Networks*, 17:87–103, 1987.

[40] S.W. Wallace. A two-stage stochastic facility-location problem with time-dependent supply. In Ermoliev and Wets [5], pages 489–513.

[41] R.J-B. Wets. Solving stochastic programs with simple recourse. *Stochastics*, 10:219–242, 1983.

[42] R.J-B. Wets. Large scale linear programming techniques. In Ermoliev and Wets [5], pages 65–93.

[43] L.A. Wolsey. *Integer Programming.* John Wiley and Sons, New York, 1998.

Appendix

Tables 3, 4, and 5 give our full computational results with the SAA method. Here, for each combination of a base instance and a number of scenarios, the Nodes, CPU, and Cuts rows give the average number of nodes, CPU time, and number of optimality cuts, respectively, over the M SAA problems solved to find the best reported solution and the lower bound estimate.

		No. Scenarios								
		200	300	400	500	600	700	800	900	1000
burma14	$\bar{z}_n$	363.0	365.0	367.4	368.7	369.5	369.3	371.8	369.6	369.8
	$\hat{\sigma}_{\bar{z}_n}$	2.60	1.03	2.40	1.55	1.34	1.04	1.67	1.57	1.00
	$\hat{z}_{10^5}(\hat{x}^*)$	372.0	372.7	371.2	371.9	371.5	371.3	371.2	371.3	371.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.44	0.45	0.44	0.44	0.44	0.44	0.44	0.44	0.44
	Nodes	19.0	20.9	21.4	18.7	18.3	21.5	22.0	18.3	19.1
	CPU	0.7	1.1	1.5	1.8	2.1	2.3	2.9	3.2	3.4
	Cuts	10.1	10.2	10.6	10.4	10.4	10.1	10.8	10.5	10.4
ulysses16	$\bar{z}_n$	2508.3	2508.0	2510.0	2510.0	2510.0	2510.0	2510.0	2510.0	2510.0
	$\hat{\sigma}_{\bar{z}_n}$	1.66	2.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$\hat{z}_{10^5}(\hat{x}^*)$	2510.0	2510.0	2510.0	2510.0	2510.0	2510.0	2510.0	2510.0	2510.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Nodes	7.3	8.8	7.9	6.7	7.7	7.7	8.1	8.0	7.6
	CPU	0.6	0.9	1.2	1.5	1.8	2.1	2.6	2.8	3.1
	Cuts	6.0	6.0	6.0	6.1	6.0	6.0	6.3	6.1	6.2
ulysses22	$\bar{z}_n$	1424.2	1440.8	1435.6	1431.1	1433.1	1433.1	1440.0	1444.9	1433.8
	$\hat{\sigma}_{\bar{z}_n}$	5.61	4.35	4.26	9.70	5.71	6.35	4.23	4.38	3.49
	$\hat{z}_{10^5}(\hat{x}^*)$	1440.1	1438.2	1440.5	1439.0	1439.3	1439.1	1438.6	1440.3	1442.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	1.57	1.56	1.57	1.57	1.57	1.57	1.56	1.57	1.58
	Nodes	14.2	16.5	16.6	14.7	14.0	15.0	14.4	15.2	14.1
	CPU	1.3	1.9	2.5	2.8	3.4	4.3	4.9	5.3	6.0
	Cuts	8.8	9.1	9.0	8.4	8.5	9.0	9.0	8.8	8.9
eil51	$\bar{z}_n$	61.0	61.1	61.3	61.4	60.9	61.0	61.6	61.4	61.1
	$\hat{\sigma}_{\bar{z}_n}$	0.46	0.26	0.24	0.36	0.30	0.33	0.26	0.19	0.26
	$\hat{z}_{10^5}(\hat{x}^*)$	61.6	61.6	61.8	61.8	61.8	61.7	61.6	61.6	61.7
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
	Nodes	14.4	14.6	15.1	17.1	14.8	14.2	16.3	15.3	15.2
	CPU	4.9	7.2	10.1	13.4	14.7	17.2	19.1	21.4	23.7
	Cuts	7.8	8.1	8.5	9.1	8.3	8.4	8.1	8.1	8.2
berlin52	$\bar{z}_n$	836.2	836.6	836.8	836.9	839.0	842.7	837.0	839.3	841.5
	$\hat{\sigma}_{\bar{z}_n}$	1.42	2.08	1.78	1.16	1.98	1.20	1.52	1.02	0.88
	$\hat{z}_{10^5}(\hat{x}^*)$	840.0	839.1	839.5	839.7	839.0	839.5	839.4	838.9	839.7
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.48	0.47	0.48	0.48	0.48	0.48	0.48	0.47	0.48
	Nodes	106.1	112.1	119.7	125.1	123.5	156.0	112.8	114.7	128.2
	CPU	18.8	26.7	36.5	48.9	57.4	72.1	72.1	89.9	104.9
	Cuts	36.3	35.5	36.7	39.5	38.7	41.4	37.0	40.3	42.2
st70	$\bar{z}_n$	112.1	113.5	115.9	115.4	113.9	113.4	115.2	115.2	114.6
	$\hat{\sigma}_{\bar{z}_n}$	0.90	0.72	0.67	0.53	0.39	0.42	0.41	0.36	0.49
	$\hat{z}_{10^5}(\hat{x}^*)$	114.8	115.0	114.9	114.7	114.8	114.7	114.8	114.8	114.9
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
	Nodes	175.3	205.0	238.2	235.3	190.7	176.6	204.9	207.7	181.8
	CPU	27.9	44.3	74.3	84.2	97.8	100.3	135.1	143.5	150.4
	Cuts	28.2	30.6	38.8	35.8	35.0	31.1	36.3	34.2	32.6
eil76	$\bar{z}_n$	49.1	49.0	49.5	49.4	49.0	49.1	49.2	49.4	49.1
	$\hat{\sigma}_{\bar{z}_n}$	0.38	0.26	0.27	0.14	0.12	0.12	0.23	0.07	0.20
	$\hat{z}_{10^5}(\hat{x}^*)$	49.3	49.2	49.2	49.2	49.3	49.3	49.2	49.2	49.2
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	Nodes	64.5	65.4	61.4	57.6	65.8	57.1	53.3	54.3	56.6
	CPU	22.8	35.6	44.7	50.3	70.9	75.4	81.6	92.6	115.4
	Cuts	21.6	23.3	22.1	20.1	23.2	21.3	20.4	20.6	22.8
pr76	$\bar{z}_n$	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0
	$\hat{\sigma}_{\bar{z}_n}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$\hat{z}_{10^5}(\hat{x}^*)$	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0	18207.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Nodes	190.3	155.7	151.2	151.3	124.4	159.5	140.1	132.9	129.1
	CPU	31.5	43.3	57.4	68.6	80.4	100.7	109.7	114.9	129.1
	Cuts	35.3	34.1	35.5	34.0	33.6	36.0	34.2	32.6	32.6

Table 3: Computational Results with SAA (SPRT).

		No. Scenarios								
		200	300	400	500	600	700	800	900	1000
gr96	$\bar{z}_n$	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0
	$\hat{\sigma}_{\bar{z}_n}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$\hat{z}_{10^5}(\hat{x}^*)$	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0	7601.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Nodes	85.3	82.3	88.0	85.2	86.1	86.3	82.4	77.4	77.8
	CPU	20.3	28.8	41.3	48.5	58.2	74.9	82.9	90.5	107.5
	Cuts	21.4	21.8	23.7	22.3	22.8	24.7	24.1	23.4	24.6
rat99	$\bar{z}_n$	205.3	204.8	205.3	205.6	205.6	205.4	205.3	205.5	205.4
	$\hat{\sigma}_{\bar{z}_n}$	0.22	0.25	0.23	0.18	0.16	0.19	0.17	0.10	0.10
	$\hat{z}_{10^5}(\hat{x}^*)$	205.8	205.8	205.7	205.8	205.7	205.6	205.9	205.9	205.8
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.06	0.06	0.06	0.06	0.06	0.05	0.06	0.06	0.06
	Nodes	582.7	489.5	594.2	573.8	608.5	576.6	581.6	573.8	631.3
	CPU	113.1	146.4	200.5	248.8	323.3	354.7	388.2	440.6	522.4
	Cuts	86.6	78.2	84.8	86.2	93.7	88.3	84.9	88.3	93.0
kroA100	$\bar{z}_n$	3355.6	3378.9	3374.7	3386.8	3395.3	3389.9	3390.9	3376.6	3384.7
	$\hat{\sigma}_{\bar{z}_n}$	17.54	8.55	9.74	7.96	6.80	9.19	7.32	9.18	10.07
	$\hat{z}_{10^5}(\hat{x}^*)$	3414.8	3416.2	3419.6	3413.6	3419.6	3418.2	3419.6	3414.9	3405.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	3.84	3.85	3.86	3.84	3.67	3.86	3.86	3.84	3.80
	Nodes	28.4	28.2	30.1	28.2	31.0	37.8	25.8	26.4	35.1
	CPU	13.6	19.9	27.0	34.8	39.9	48.9	49.5	60.5	69.5
	Cuts	11.2	11.5	12.1	12.1	11.6	12.7	11.2	11.9	12.7
kroB100	$\bar{z}_n$	2466.4	2466.4	2462.9	2470.1	2466.4	2460.6	2469.1	2475.0	2471.7
	$\hat{\sigma}_{\bar{z}_n}$	7.16	7.34	6.08	6.29	6.81	5.71	4.42	5.38	4.59
	$\hat{z}_{10^5}(\hat{x}^*)$	2467.9	2470.0	2469.2	2469.7	2470.7	2472.1	2470.8	2468.9	2468.2
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	1.47	1.48	1.47	1.48	1.48	1.49	1.48	1.47	1.47
	Nodes	3.3	2.5	2.6	2.5	1.6	2.3	2.9	2.9	3.3
	CPU	5.0	4.4	5.0	7.6	7.3	8.1	10.5	12.3	18.4
	Cuts	3.8	2.9	2.8	3.1	2.8	2.7	3.1	3.2	3.5
kroC100	$\bar{z}_n$	2920.0	2944.5	2946.7	2939.1	2939.6	2938.9	2938.0	2938.2	2940.4
	$\hat{\sigma}_{\bar{z}_n}$	11.40	3.11	3.46	3.45	2.63	2.89	0.98	0.89	1.25
	$\hat{z}_{10^5}(\hat{x}^*)$	2940.5	2941.1	2940.6	2940.7	2941.2	2939.8	2940.3	2939.9	2941.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.60	0.60	0.60	0.60	0.60	0.59	0.60	0.60	0.60
	Nodes	32.1	38.2	36.7	34.1	35.8	32.9	30.9	31.7	33.1
	CPU	14.5	25.7	31.2	43.1	45.2	57.2	61.5	68.9	75.5
	Cuts	11.5	13.9	13.4	14.9	12.9	14.0	13.0	13.1	13.1
kroD100	$\bar{z}_n$	2486.6	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0
	$\hat{\sigma}_{\bar{z}_n}$	3.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$\hat{z}_{10^5}(\hat{x}^*)$	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0	2490.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Nodes	3.0	1.8	1.6	1.6	2.5	2.3	1.8	1.2	1.1
	CPU	2.9	4.0	5.1	6.4	8.1	8.9	10.8	10.9	11.9
	Cuts	2.4	2.2	2.2	2.2	2.4	2.2	2.5	2.1	2.0
kroE100	$\bar{z}_n$	3718.3	3716.2	3705.2	3702.2	3714.6	3719.0	3710.8	3716.0	3704.0
	$\hat{\sigma}_{\bar{z}_n}$	9.24	7.39	6.05	5.70	5.64	5.29	6.16	3.15	2.05
	$\hat{z}_{10^5}(\hat{x}^*)$	3710.4	3709.5	3712.1	3709.8	3710.3	3710.2	3709.7	3710.3	3708.3
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	1.36	1.35	1.37	1.36	1.36	1.36	1.35	1.35	1.34
	Nodes	39.0	38.0	27.8	27.5	31.2	26.1	25.3	29.1	29.6
	CPU	12.0	18.1	23.3	28.3	34.7	38.8	44.6	51.2	49.2
	Cuts	9.6	9.9	9.9	9.4	9.6	9.1	9.4	9.5	8.2
rd100	$\bar{z}_n$	1195.2	1196.5	1197.0	1196.8	1197.0	1197.0	1197.0	1197.0	1197.0
	$\hat{\sigma}_{\bar{z}_n}$	1.64	0.47	0.00	0.23	0.00	0.00	0.00	0.00	0.00
	$\hat{z}_{10^5}(\hat{x}^*)$	1197.0	1197.0	1197.0	1197.0	1197.0	1197.0	1197.0	1197.0	1197.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Nodes	30.4	31.6	31.6	31.0	33.3	29.0	35.3	31.6	30.6
	CPU	10.6	15.5	19.8	25.1	28.7	33.0	39.7	44.2	48.9
	Cuts	10.2	10.3	10.0	10.4	10.0	9.9	10.3	10.4	10.3
eil101	$\bar{z}_n$	51.7	52.1	51.9	52.6	52.3	52.1	52.2	52.1	52.3
	$\hat{\sigma}_{\bar{z}_n}$	0.39	0.23	0.33	0.16	0.29	0.16	0.21	0.20	0.15
	$\hat{z}_{10^5}(\hat{x}^*)$	52.2	52.2	52.2	52.2	52.2	52.2	52.2	52.2	52.2
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.05	0.05	0.05	0.05	0.05	0.06	0.05	0.05	0.05
	Nodes	138.5	105.0	112.6	117.8	112.8	107.0	109.8	119.6	100.5
	CPU	53.3	73.0	101.3	135.3	153.7	172.7	188.8	246.6	259.9
	Cuts	33.1	31.3	32.9	35.1	33.8	32.2	30.8	35.0	33.9

Table 3: Computational Results with SAA (SPRT). (Cnt'd)

		No. Scenarios								
		200	300	400	500	600	700	800	900	1000
burma14	$\bar{z}_n$	321.6	322.2	321.0	321.5	320.4	320.4	321.0	320.3	321.4
	$\hat{\sigma}_{\bar{z}_n}$	0.69	0.64	0.65	0.63	0.85	0.61	0.37	0.41	0.41
	$\hat{z}_{10^5}(\hat{x}^*)$	320.9	320.9	320.9	320.8	320.7	320.9	320.8	321.0	320.8
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
	Nodes	2.9	3.2	2.8	2.6	2.5	3.2	1.1	1.4	2.9
	CPU	2.6	4.4	4.9	6.2	6.6	8.2	7.1	7.4	11.0
	Cuts	4.0	4.4	3.7	4.0	3.5	3.7	2.7	2.5	3.4
ulysses16	$\bar{z}_n$	2376.8	2379.7	2380.5	2381.4	2383.0	2379.8	2381.7	2380.3	2381.6
	$\hat{\sigma}_{\bar{z}_n}$	1.77	1.30	1.59	1.15	1.45	0.89	1.33	0.94	1.03
	$\hat{z}_{10^5}(\hat{x}^*)$	2381.8	2382.1	2382.2	2381.3	2382.4	2382.1	2381.8	2381.8	2382.2
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.37
	Nodes	1.3	1.2	1.4	1.7	1.5	1.0	1.3	1.0	1.2
	CPU	2.8	4.3	5.8	7.0	9.4	10.0	11.4	12.2	14.3
	Cuts	3.8	3.9	4.1	3.6	4.2	3.8	3.8	3.8	3.9
ulysses22	$\bar{z}_n$	1302.0	1302.3	1303.9	1302.5	1303.9	1305.9	1306.2	1305.1	1306.3
	$\hat{\sigma}_{\bar{z}_n}$	2.14	1.02	1.65	0.81	0.97	1.08	0.52	0.74	0.77
	$\hat{z}_{10^5}(\hat{x}^*)$	1306.2	1306.7	1306.6	1307.0	1306.3	1306.9	1306.5	1306.7	1307.1
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
	Nodes	2.5	2.1	4.2	3.7	4.1	4.3	3.2	3.2	3.5
	CPU	4.7	7.4	11.7	15.0	18.4	19.4	19.8	24.8	27.6
	Cuts	4.3	4.8	5.1	5.4	5.6	4.7	4.2	5.0	5.1
eil51	$\bar{z}_n$	51.2	51.1	51.3	51.2	51.4	51.2	51.2	51.3	51.2
	$\hat{\sigma}_{\bar{z}_n}$	0.19	0.10	0.09	0.09	0.08	0.06	0.09	0.06	0.07
	$\hat{z}_{10^5}(\hat{x}^*)$	51.2	51.2	51.2	51.2	51.2	51.2	51.2	51.2	51.2
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	Nodes	5.7	6.5	7.6	7.5	6.9	7.7	7.4	6.4	6.3
	CPU	18.0	26.8	35.2	45.1	54.8	64.0	74.0	79.2	88.9
	Cuts	6.3	6.5	6.4	6.6	6.7	6.7	6.8	6.4	6.5
berlin52	$\bar{z}_n$	810.8	812.0	811.0	810.5	811.9	812.4	811.2	812.0	811.0
	$\hat{\sigma}_{\bar{z}_n}$	1.00	0.94	0.80	0.51	0.59	0.45	0.54	0.26	0.70
	$\hat{z}_{10^5}(\hat{x}^*)$	812.8	812.9	812.3	812.5	812.4	812.5	812.6	812.6	812.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.18	0.19	0.18	0.18	0.18	0.18	0.18	0.18	0.18
	Nodes	4.7	5.8	4.4	4.3	5.5	4.9	5.0	6.2	4.9
	CPU	21.3	30.7	38.8	49.7	61.9	69.4	78.5	94.0	96.6
	Cuts	7.1	7.4	6.6	6.6	6.9	6.8	6.7	7.1	6.6
st70	$\bar{z}_n$	102.4	101.9	102.4	102.4	102.6	102.5	102.4	102.7	102.4
	$\hat{\sigma}_{\bar{z}_n}$	0.22	0.31	0.29	0.22	0.20	0.17	0.20	0.12	0.18
	$\hat{z}_{10^5}(\hat{x}^*)$	102.8	102.6	102.8	102.6	102.8	102.8	102.8	102.7	102.7
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
	Nodes	18.1	15.6	15.0	14.6	14.9	14.7	17.7	19.9	19.6
	CPU	63.8	89.3	109.3	142.3	183.5	186.1	235.6	276.4	296.1
	Cuts	13.8	13.2	12.3	12.6	14.6	12.0	13.7	14.3	13.6
eil76	$\bar{z}_n$	46.1	46.4	46.4	46.4	46.4	46.4	46.4	46.4	46.4
	$\hat{\sigma}_{\bar{z}_n}$	0.10	0.07	0.07	0.10	0.07	0.05	0.03	0.05	0.05
	$\hat{z}_{10^5}(\hat{x}^*)$	46.5	46.5	46.5	46.5	46.5	46.5	46.5	46.5	46.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	Nodes	14.0	16.0	14.8	20.9	19.0	16.6	17.9	15.9	17.2
	CPU	51.7	84.7	99.8	141.4	151.2	181.4	200.3	233.4	269.5
	Cuts	11.8	12.5	11.3	12.1	11.2	11.3	10.9	11.4	11.7
pr76	$\bar{z}_n$	18457.6	18452.6	18451.5	18453.3	18459.6	18467.4	18461.4	18460.2	18464.0
	$\hat{\sigma}_{\bar{z}_n}$	10.88	11.20	11.54	9.39	9.39	10.49	7.37	8.30	8.22
	$\hat{z}_{10^5}(\hat{x}^*)$	18468.4	18465.7	18470.1	18467.2	18462.8	18469.4	18472.5	18466.1	18463.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	2.54	2.53	2.56	2.54	2.52	2.55	2.56	2.53	2.53
	Nodes	52.3	49.9	57.1	45.3	82.1	110.4	19.8	68.9	118.4
	CPU	75.7	90.5	140.2	135.6	241.0	263.9	154.9	284.8	403.6
	Cuts	16.6	13.5	15.2	11.5	18.1	16.8	8.8	14.0	18.5
gr96	$\bar{z}_n$	7400.2	7420.4	7426.7	7420.0	7419.5	7423.9	7422.5	7426.0	7421.6
	$\hat{\sigma}_{\bar{z}_n}$	2.24	4.28	2.72	5.32	2.93	4.14	3.41	3.56	3.88
	$\hat{z}_{10^5}(\hat{x}^*)$	7420.1	7422.0	7422.4	7421.9	7422.0	7422.0	7421.6	7421.4	7422.6
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.90	0.91	0.91	0.90	0.90	0.90	0.90	0.90	0.91
	Nodes	11.9	13.7	12.8	17.2	17.1	19.3	11.7	19.3	17.3
	CPU	41.2	72.2	92.7	144.1	130.3	178.3	189.2	213.2	230.0
	Cuts	7.2	7.8	7.5	9.8	7.6	8.6	8.0	7.9	7.6

Table 4: Computational Results with SAA (SPAF).

		No. Scenarios								
		200	300	400	500	600	700	800	900	1000
rat99	$\bar{z}_n$	206.5	206.5	206.7	206.8	206.9	206.9	207.1	207.0	207.0
	$\hat{\sigma}_{\bar{z}_n}$	0.20	0.17	0.17	0.13	0.09	0.08	0.09	0.09	0.14
	$\hat{z}_{10^5}(\hat{x}^*)$	207.1	207.1	207.1	207.2	207.1	207.2	207.1	207.1	207.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03
	Nodes	250.8	209.9	256.2	222.6	333.2	320.7	287.7	319.4	297.7
	CPU	225.9	300.8	447.9	551.7	777.0	843.0	1003.6	1104.2	1109.2
	Cuts	41.2	36.2	40.6	39.1	46.9	43.4	45.6	44.1	40.5
kroA100	$\bar{z}_n$	3021.7	3024.5	3030.5	3022.9	3025.8	3021.9	3024.1	3027.1	3024.8
	$\hat{\sigma}_{\bar{z}_n}$	4.59	4.14	2.23	2.91	3.78	3.16	4.11	2.30	1.97
	$\hat{z}_{10^5}(\hat{x}^*)$	3023.5	3023.3	3023.8	3023.6	3022.4	3023.5	3023.7	3023.1	3022.8
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.73
	Nodes	9.1	18.3	11.4	8.3	14.8	14.9	19.2	13.7	8.3
	CPU	35.6	83.3	84.2	89.2	113.0	142.7	165.2	168.3	185.7
	Cuts	5.3	8.1	6.1	5.0	5.8	5.9	6.0	5.4	5.3
kroB100	$\bar{z}_n$	2372.3	2364.7	2367.5	2366.4	2365.9	2366.4	2366.1	2368.6	2364.2
	$\hat{\sigma}_{\bar{z}_n}$	2.41	3.29	1.68	2.36	1.50	1.94	2.21	1.53	1.36
	$\hat{z}_{10^5}(\hat{x}^*)$	2366.4	2366.7	2366.3	2366.6	2366.6	2366.6	2366.8	2366.5	2366.6
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
	Nodes	2.3	2.8	3.8	2.5	2.4	2.5	2.1	2.3	2.7
	CPU	20.3	33.4	44.8	50.0	66.2	70.8	79.4	89.9	109.2
	Cuts	3.1	3.5	3.5	3.1	3.4	3.1	3.1	3.0	3.4
kroC100	$\bar{z}_n$	2650.3	2646.5	2650.2	2645.1	2643.2	2648.5	2650.3	2656.9	2648.9
	$\hat{\sigma}_{\bar{z}_n}$	4.78	3.84	4.96	3.18	2.83	3.66	2.26	3.32	3.15
	$\hat{z}_{10^5}(\hat{x}^*)$	2647.8	2649.6	2648.3	2649.9	2647.7	2648.3	2649.2	2648.0	2649.3
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87	0.87
	Nodes	34.2	35.6	34.7	26.1	32.5	30.8	30.7	28.4	36.4
	CPU	66.5	97.9	129.5	157.1	190.3	210.5	234.7	277.9	346.0
	Cuts	11.5	11.4	10.5	10.8	10.4	9.8	9.9	10.2	11.7
kroD100	$\bar{z}_n$	2384.1	2379.8	2387.7	2383.7	2381.8	2381.0	2382.9	2382.4	2382.4
	$\hat{\sigma}_{\bar{z}_n}$	2.67	2.33	2.02	1.70	1.04	1.51	1.14	1.65	1.64
	$\hat{z}_{10^5}(\hat{x}^*)$	2382.9	2383.0	2382.7	2382.9	2382.8	2383.5	2382.3	2383.5	2382.9
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.44	0.44	0.44	0.44	0.44	0.44	0.43	0.44	0.43
	Nodes	1.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.3
	CPU	22.7	30.4	43.8	50.8	58.6	71.2	78.1	94.6	105.1
	Cuts	3.4	3.1	3.4	3.1	3.0	3.1	3.0	3.2	3.1
kroE100	$\bar{z}_n$	3531.5	3532.2	3536.8	3537.5	3532.5	3532.1	3535.3	3532.2	3528.8
	$\hat{\sigma}_{\bar{z}_n}$	6.32	2.74	2.71	2.94	2.70	2.97	1.85	2.21	1.91
	$\hat{z}_{10^5}(\hat{x}^*)$	3533.1	3533.5	3532.9	3532.8	3532.8	3532.9	3533.8	3532.4	3532.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69
	Nodes	63.9	57.0	59.6	100.2	52.8	62.1	54.6	49.8	63.2
	CPU	73.6	103.8	151.1	229.4	181.5	251.2	259.2	286.0	347.3
	Cuts	13.2	12.6	13.8	16.1	11.1	13.2	12.0	11.7	13.0
rd100	$\bar{z}_n$	1165.5	1166.0	1165.0	1166.6	1167.6	1166.8	1166.8	1166.3	1166.2
	$\hat{\sigma}_{\bar{z}_n}$	1.11	0.73	0.94	0.65	0.61	0.58	0.50	0.74	0.42
	$\hat{z}_{10^5}(\hat{x}^*)$	1166.3	1166.4	1166.3	1166.5	1166.5	1166.2	1166.4	1166.4	1166.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	Nodes	10.3	16.9	12.8	14.9	15.7	21.6	18.1	18.0	17.9
	CPU	41.4	74.2	95.1	132.2	140.7	185.3	209.6	220.6	250.8
	Cuts	6.2	7.5	7.0	8.0	6.7	8.0	8.1	7.5	7.7
eil101	$\bar{z}_n$	44.2	44.4	44.6	44.5	44.5	44.4	44.5	44.7	44.5
	$\hat{\sigma}_{\bar{z}_n}$	0.15	0.15	0.07	0.10	0.10	0.06	0.09	0.06	0.04
	$\hat{z}_{10^5}(\hat{x}^*)$	44.5	44.5	44.5	44.5	44.5	44.6	44.5	44.5	44.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	Nodes	35.8	32.9	46.7	40.7	38.7	38.5	40.6	46.0	39.2
	CPU	108.0	174.4	237.6	270.9	300.8	379.8	463.5	532.0	517.5
	Cuts	17.0	19.3	18.7	17.1	16.3	17.3	18.3	19.5	16.9

Table 4: Computational Results with SAA (SPAF). (Cnt'd)

		No. Scenarios								
		200	300	400	500	600	700	800	900	1000
burma14	$\bar{z}_n$	1559.1	1551.7	1565.4	1554.0	1550.4	1559.2	1556.9	1555.8	1555.5
	$\hat{\sigma}_{\bar{z}_n}$	8.57	6.43	3.94	5.57	5.30	4.06	3.24	2.86	2.90
	$\hat{z}_{10^5}(\hat{x}^*)$	1565.0	1565.0	1564.0	1561.3	1564.0	1563.7	1564.7	1564.2	1562.7
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	1.32	1.32	1.32	1.30	1.32	1.32	1.33	1.33	1.31
	Nodes	206.2	230.6	232.0	211.2	183.6	216.3	205.9	250.7	180.7
	CPU	1.7	2.0	2.7	3.0	3.1	4.0	4.2	4.8	4.9
	Cuts	37.3	35.4	40.4	40.6	36.1	40.2	37.9	39.0	37.7
ulysses16	$\bar{z}_n$	9290.9	9374.3	9408.1	9439.8	9457.7	9474.0	9457.3	9466.4	9471.5
	$\hat{\sigma}_{\bar{z}_n}$	31.10	36.80	35.54	37.19	27.46	30.93	19.30	27.78	23.08
	$\hat{z}_{10^5}(\hat{x}^*)$	9488.2	9501.4	9529.2	9493.6	9498.7	9498.2	9500.6	9497.4	9506.2
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	7.50	7.58	6.87	7.50	7.58	7.53	7.59	7.55	7.61
	Nodes	629.0	774.2	859.4	610.7	996.1	968.1	791.8	882.4	811.6
	CPU	4.5	5.9	7.7	7.1	10.1	10.8	10.6	11.0	11.6
	Cuts	50.9	52.0	56.3	55.8	59.8	59.5	57.3	52.1	54.2
ulysses22	$\bar{z}_n$	9804.1	9883.3	9921.6	9934.1	9929.3	9883.8	9894.0	9961.9	9920.9
	$\hat{\sigma}_{\bar{z}_n}$	47.97	24.34	23.11	18.98	18.20	30.12	14.86	21.84	27.93
	$\hat{z}_{10^5}(\hat{x}^*)$	10023.6	10024.1	9985.6	10000.3	10009.6	9989.3	9986.9	9996.5	10000.9
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	6.50	6.49	6.50	6.56	6.47	6.51	6.46	6.52	6.55
	Nodes	4190.3	6089.5	5656.8	6818.6	6317.7	6033.4	6496.3	7602.9	6038.7
	CPU	50.6	88.0	90.5	106.7	112.9	97.1	114.1	143.9	129.3
	Cuts	132.0	175.3	180.7	187.0	206.5	155.7	177.9	203.5	192.1
eil51	$\bar{z}_n$	512.4	519.5	519.1	521.1	522.2	523.5	522.7	523.9	526.0
	$\hat{\sigma}_{\bar{z}_n}$	3.58	1.90	2.62	2.41	2.27	1.78	1.95	2.26	1.69
	$\hat{z}_{10^5}(\hat{x}^*)$	550.2	544.1	547.9	547.0	543.8	547.0	544.7	546.7	543.9
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.65	0.64	0.64	0.64	0.63	0.64	0.64	0.64	0.64
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	1283.6	1117.4	1170.1	1150.9	1230.8	1315.0	1171.6	1256.2	1337.7
	Cuts	250.8	220.0	240.0	239.9	254.4	268.5	242.4	244.2	264.5
berlin52	$\bar{z}_n$	9036.3	9141.6	9148.0	9186.4	9188.0	9183.1	9221.7	9200.2	9172.4
	$\hat{\sigma}_{\bar{z}_n}$	19.36	33.09	24.08	32.23	18.85	33.27	18.23	24.77	20.14
	$\hat{z}_{10^5}(\hat{x}^*)$	9496.3	9495.3	9496.8	9476.0	9469.4	9453.1	9482.9	9482.0	9481.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	8.57	7.83	8.56	8.48	8.45	8.46	8.51	8.47	8.53
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	1018.8	1317.9	1329.9	1210.4	1210.9	1125.2	1371.7	1290.0	1305.6
	Cuts	207.1	268.3	280.4	264.7	256.0	219.7	285.7	254.6	253.4
st70	$\bar{z}_n$	747.1	760.6	761.4	762.4	759.6	764.5	769.3	768.0	767.6
	$\hat{\sigma}_{\bar{z}_n}$	3.84	4.52	1.97	3.01	3.31	2.66	2.32	2.37	2.31
	$\hat{z}_{10^5}(\hat{x}^*)$	860.6	840.8	839.8	837.0	841.0	836.7	842.5	841.3	834.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.97	0.92	0.92	0.92	0.93	0.92	0.93	0.92	0.91
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	1630.8	1494.2	1654.8	1578.5	1372.8	1508.9	1535.2	1606.5	1549.0
	Cuts	167.1	164.3	197.4	177.3	155.7	162.1	184.1	178.6	168.5
eil76	$\bar{z}_n$	619.4	620.6	624.2	627.8	626.4	630.5	629.8	632.3	631.0
	$\hat{\sigma}_{\bar{z}_n}$	3.45	1.88	2.11	1.42	1.83	3.31	2.03	1.63	1.44
	$\hat{z}_{10^5}(\hat{x}^*)$	662.6	667.9	661.8	662.7	659.9	660.1	661.0	661.1	661.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.59	0.70	0.59	0.59	0.59	0.59	0.59	0.59	0.59
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	3448.6	3229.6	3167.3	3181.0	3361.5	3201.7	3421.0	3279.7	3147.5
	Cuts	154.9	155.0	163.5	179.9	195.8	190.9	195.3	194.0	204.3
pr76	$\bar{z}_n$	114408.0	114314.0	114500.0	114491.0	114851.0	114930.0	114932.0	114944.0	115104.0
	$\hat{\sigma}_{\bar{z}_n}$	371.11	305.30	148.97	378.42	177.70	284.87	200.05	200.64	171.58
	$\hat{z}_{10^5}(\hat{x}^*)$	125067.0	124206.0	124144.0	124426.0	123658.0	123586.0	123609.0	123480.0	124419.0
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	77.59	74.98	75.91	76.86	74.59	72.42	75.06	74.39	77.06
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	1360.8	1397.0	1319.3	1275.6	1356.4	1311.1	1243.0	1333.6	1312.2
	Cuts	76.9	85.4	88.1	88.5	91.3	87.3	84.7	90.7	91.9
gr96	$\bar{z}_n$	59866.4	59975.4	60301.5	60068.8	60387.4	60457.5	60443.3	60409.5	60509.0
	$\hat{\sigma}_{\bar{z}_n}$	141.43	133.45	120.02	79.93	102.25	97.40	98.30	65.73	44.94
	$\hat{z}_{10^5}(\hat{x}^*)$	62799.3	63090.6	62857.1	63050.7	63092.9	63086.8	63155.8	62844.3	62837.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	37.12	35.65	33.84	35.41	37.52	33.11	34.63	37.34	34.39
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	2478.5	2478.5	2363.2	2343.8	2286.7	2187.4	2379.4	2274.8	2278.7
	Cuts	165.0	162.7	174.0	162.0	164.7	163.5	166.4	169.2	171.6

Table 5: Computational Results with SAA (TSPRT).

		No. Scenarios								
		200	300	400	500	600	700	800	900	1000
rat99	$\bar{z}_n$	1391.5	1388.0	1405.2	1410.1	1414.8	1408.2	1412.5	1413.8	1417.0
	$\hat{\sigma}_{\bar{z}_n}$	13.35	4.45	4.73	6.06	3.51	5.25	4.44	2.99	2.18
	$\hat{z}_{10^5}(\hat{x}^*)$	1469.1	1465.2	1470.3	1468.5	1466.8	1462.4	1461.0	1462.3	1464.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	1.25	1.28	1.20	1.24	1.35	1.33	1.28	1.29	1.24
	Nodes	10000.0	10000.0	9716.7	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	3153.4	2768.7	2632.0	3099.2	2949.8	2828.6	2688.9	3067.2	2809.8
	Cuts	178.3	174.7	170.6	189.2	173.5	176.8	182.5	200.9	176.4
kroA100	$\bar{z}_n$	24141.1	24155.7	24381.7	24388.7	24369.9	24335.0	24448.1	24580.2	24541.3
	$\hat{\sigma}_{\bar{z}_n}$	112.48	87.66	68.61	73.86	58.01	73.18	57.04	42.44	62.13
	$\hat{z}_{10^5}(\hat{x}^*)$	25807.5	25860.3	26004.5	25697.8	25918.0	25810.7	25793.2	25803.5	25775.1
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	19.23	21.23	20.57	19.10	19.36	17.95	20.03	20.21	20.19
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	2681.1	2581.2	2497.5	2418.5	2311.9	2439.4	2390.2	2401.8	2263.0
	Cuts	154.6	164.6	159.4	156.5	164.0	166.2	169.1	175.6	165.6
kroB100	$\bar{z}_n$	24043.4	24506.2	24353.1	24475.2	24407.7	24457.3	24453.6	24655.2	24549.0
	$\hat{\sigma}_{\bar{z}_n}$	106.38	53.81	63.92	47.58	53.02	40.70	42.31	41.62	47.49
	$\hat{z}_{10^5}(\hat{x}^*)$	26319.7	26732.1	26418.0	26466.2	26286.0	26470.1	26551.4	26356.6	26411.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	18.57	18.21	19.77	20.53	15.80	17.47	21.37	19.26	18.32
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	3114.9	3156.8	2828.6	3107.4	2854.8	2796.4	2936.4	2941.2	2754.5
	Cuts	178.1	198.8	206.0	234.1	205.4	202.3	207.1	218.7	202.3
kroC100	$\bar{z}_n$	23372.6	23556.9	23516.8	23679.1	23656.5	23680.3	23680.1	23649.7	23710.0
	$\hat{\sigma}_{\bar{z}_n}$	97.70	69.81	66.86	83.78	59.53	49.01	80.80	33.33	36.78
	$\hat{z}_{10^5}(\hat{x}^*)$	24792.5	24902.9	24732.4	24640.2	24655.1	24681.6	24692.3	24628.5	24660.3
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	19.42	19.71	18.33	17.07	18.21	18.19	18.13	17.18	17.19
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	2503.8	2597.9	2471.9	2426.4	2278.0	2266.1	2120.2	2383.2	2406.5
	Cuts	169.7	201.6	226.0	205.9	207.6	188.2	181.0	200.1	220.4
kroD100	$\bar{z}_n$	23713.4	23716.1	23930.4	23977.2	23952.7	24043.7	24096.5	24136.7	24118.7
	$\hat{\sigma}_{\bar{z}_n}$	79.48	81.50	64.31	69.41	45.74	39.65	37.34	32.03	35.72
	$\hat{z}_{10^5}(\hat{x}^*)$	25719.1	25896.5	25774.7	25538.6	25570.3	25500.6	25629.6	25499.0	25468.7
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	18.00	19.42	20.04	17.54	18.72	16.52	18.67	18.52	19.91
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	3338.1	3069.2	3087.8	3069.9	2967.9	2812.1	2985.9	2927.2	3055.0
	Cuts	195.4	186.6	196.7	208.6	199.3	194.7	205.6	189.6	208.0
kroE100	$\bar{z}_n$	23424.6	23712.8	23644.5	23745.5	23826.0	23744.6	23766.0	23856.1	23800.4
	$\hat{\sigma}_{\bar{z}_n}$	64.11	67.07	57.41	30.93	60.42	45.91	38.93	35.65	35.55
	$\hat{z}_{10^5}(\hat{x}^*)$	26007.8	26134.5	25894.7	25773.3	25933.4	25963.4	25766.3	25843.9	25775.5
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	17.93	18.23	19.75	17.60	19.82	18.04	17.57	19.43	19.43
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	2790.3	2719.1	2487.3	2624.0	2613.7	2632.9	2542.5	2446.4	2518.9
	Cuts	73.2	77.8	75.9	83.9	88.7	83.3	83.3	84.0	86.7
rd100	$\bar{z}_n$	8903.9	8930.1	9002.5	8958.3	9037.0	9000.0	9044.6	9015.0	9039.1
	$\hat{\sigma}_{\bar{z}_n}$	42.51	20.26	31.60	25.14	29.54	22.30	28.43	19.47	16.50
	$\hat{z}_{10^5}(\hat{x}^*)$	9557.9	9526.7	9502.9	9486.1	9517.4	9486.0	9500.6	9463.5	9478.4
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	7.01	7.33	7.68	6.59	6.62	6.59	7.35	7.24	7.31
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	2573.2	2168.6	2281.4	2165.2	2362.2	2099.5	2057.1	2149.1	2058.1
	Cuts	107.2	90.0	122.9	105.1	127.3	122.0	107.7	120.7	117.0
eil101	$\bar{z}_n$	701.7	708.7	714.7	712.8	718.5	721.0	720.9	720.6	721.2
	$\hat{\sigma}_{\bar{z}_n}$	3.37	0.89	1.94	2.58	3.27	2.34	2.61	1.44	2.33
	$\hat{z}_{10^5}(\hat{x}^*)$	771.7	766.5	768.5	761.3	764.4	761.1	759.6	763.8	763.1
	$\hat{\sigma}_{\hat{z}_{10^5}(\hat{x}^*)}$	0.77	0.76	0.76	0.75	0.75	0.75	0.74	0.75	0.75
	Nodes	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0	10000.0
	CPU	4745.7	4060.1	4354.6	3966.6	3964.8	4261.1	4058.6	4170.6	4051.9
	Cuts	72.4	78.2	102.6	94.1	91.1	95.6	107.3	106.7	100.4

Table 5: Computational Results with SAA (TSPRT). (Cnt'd)