Distributed Sampled Convex Problems

David Eckman (dje88) and Calvin Wylie (cjw278)

Abstract

It is well-known that chance constrained programs are in general intractable because of the non-convexity of the feasible region and the requirement of prior information about the distribution of the uncertain parameters. Solving a sampled convex program allows for finding solutions that are feasible for the chance constrained program with high probability with known bounds on the required number of samples. Sampled convex problems are more easily solved because they are characterized by a finite number of constraints and require only that one can obtain i.i.d. samples of the uncertain parameters. However, the required number of samples may still result in a sampled convex program that is expensive to solve exactly. We consider obtaining approximate solutions to these sampled convex programs by distributing the constraints across processors and solving smaller subproblems followed by a consensus on the active constraints. We consider a classical portfolio-optimization problem and study the tradeoffs that arise from this method: including wall-clock time, violation probability, and feasibility probability.

Introduction

Chance constrained problems model many real-word applications in areas as diverse as finance, energy, and emergency services [1]. For example, chance constraints can represent an acceptable level of risk in an investment or a fixed level of service that is guaranteed. A typical chance-constrained program CCP_ϵ is given by

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & \mathbb{P}\{f(x, \delta) \leq 0\} \geq 1 - \epsilon. \end{aligned} \tag{CCP_ϵ}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is compact, δ is the uncertain parameter drawn from the uncertainty set $\Delta \subseteq \mathbb{R}^d$ according to a probability measure $\mathbb{P}$, and $f : \mathcal{X} \times \Delta \mapsto \mathbb{R}$ is convex in x for any fixed δ. The parameter $0 \leq \epsilon \leq 1$ represents the acceptable probability with which the constraint $f(x, \delta) \leq 0$ can be violated. The problem CCP_ϵ is notoriously intractable for two reasons: (1) that the feasible region is, in general, non-convex, and (2) the distribution $\mathbb{P}$ must be known a priori.

One method for finding solutions with good performance which are also feasible for CCP_ϵ with high probability is to take N iid samples $(\delta_1, \ldots, \delta_N)$ from the uncertainty set Δ

according to $\mathbb{P}$ and solve the corresponding sampled convex program SCP_N:

$$\begin{array}{ll} \underset{x\in\mathcal{X}}{\text{minimize}} & c^T x \\ \text{subject to} & f(x,\delta_i)\leq 0 \quad i=1,\ldots,N. \end{array} \tag{SCP_N}$$

Notice that SCP_N is more tractable than CCP_ϵ since it has a finite number of constraints, a convex feasible region, and we need not explicitly know $\mathbb{P}$ but instead only need to be able to obtain iid samples from it.

Let x_N^* be the random solution to SCP_N obtained by taking N samples and let $V(x_N^*) = \mathbb{P}^N\{f(x_N^*,\delta)>0\}$ be the corresponding violation probability where $\mathbb{P}^N$ refers to the product probability measure. To clarify, if $V(x_N^*)\leq\epsilon$, then x_N^* is feasible for CCP_ϵ, otherwise it is infeasible.

Sampled convex programs do not provide a guarantee that $V(x_N^*)\leq\epsilon$ for a given sample size N. Instead, Calafiore and Campi [2] proved that one can bound the number of samples needed to ensure that $\mathbb{P}^N\{V(x_N^*)>\epsilon\}\leq\beta$. That is, the optimal solution to SCP_N is feasible for CCP_ϵ with probability greater than $1-\beta$. Their bound of $N\geq 1/(\epsilon\beta)-1$ is simple to calculate, but is quite loose in terms of asymptotics. The bound was further tightened to choosing N to satisfy

$$\mathbb{P}^N\{V(x_N^*)>\epsilon\}\leq\binom{N}{d}(1-\epsilon)^{N-d}.$$

in [3] and then later by Campi and Garatti in [4] to the solution to a binomial equation:

$$\mathbb{P}^N\{V(x_N^*)>\epsilon\}=\sum_{i=0}^{d-1}\binom{N}{i}\epsilon^i(1-\epsilon)^{N-i}.$$

Note that all of these bounds are distribution-free which means that N can be calculated without prior knowledge of $\mathbb{P}$.

We are interested in the case when SCP_N is still computationally expensive to solve due to the large number of samples (and therefore constraints) needed to reach a solution that is feasible for CCP_ϵ with high probability. For example, when $d=200$, $\epsilon=0.05$ and $\beta=1\times10^{-5}$ are small, the resulting sample size is $N=5312$. With several thousand constraints, existing solution methods for mixed integer programs or semi-definite programs might take too long to run to completion. For instances such as these, it is possible to decompose SCP_N into smaller sampled convex programs and solving them in parallel. Because the subproblems are of smaller dimension, they may be solved faster than SCP_N. However because each subproblem is only considering a subset of the constraints, there is a loss of optimality in solving these problems. Still the speedup in computational time may be worth the loss of CCP_ϵ feasibility. In particular, we would like to study how much is lost with respect to CCP_ϵ feasibility by taking the active constraints from the subproblems and solving a final sampled convex program with these constraints.

Even for problems for which N is not too large, but the number of variables n is far too large, this method of parallelism can be applied by taking the dual of SCP_N and then distributing the constraints of the dual across processors.

Method

A recent paper [5] studies solving large-scale convex programs in a distributed fashion. It assume some topology on the communication network connecting the processors. Each processor solves a convex program with a subset of the constraints. It then identifies the active constraints at that solution and passes them to other processors according to the communication network. Each processor then adds in the incoming constraints to their personal set and resolves their subproblem. It was proven that after a finite number of iterations, a.s., all processors will reach a consensus about which constraints are active for the solution to the original problem.

The objective in [5] is to recover the exact solution to the convex program. However in practice, this procedure would take far too long to run, especially since there is no bound on the number of iterations needed for consensus. The only instance you would fully run a procedure like this is if the master problem is too large to solve with existing software.

Instead, our method will be a variation of this procedure which only runs for two iterations. The topology on the processors is that all workers will pass their active constraints to a single master processor, who will then use these constraints to solve another convex program. Because there are only two iterations, the solutions returned will likely not be the exact solution to SCP_N. However this approximate solution may be calculated in less time, and thus it may be worth a loss in CCP_ϵ feasibility. We wish to explore this tradeoff to get a sense of what the best mix would be.

Another tradeoff comes from changing the number of processors. With more processors, the subproblems are smaller, but the total number of constraints in the second stage problem is larger. This will impact the wall-clock time of the procedure in two conflicting ways. The main question will be to determine what the optimal number of processors to use is.

Another way to view the problem is to assume with are willing to use the wall-clock time need to solve SCP_N. Using that time, what is the best number of processors to use and number of constraints to give to each processor. Because of the time savings from solving smaller problems, we may be able to use more than N samples.

We divide the N samples across p processors and solve each subproblem which has N/p constraints. We don't want to use so many processors that $N/p < n$. For each subproblem, we identify the active constraints (of which there should be n unless the problem is degenerate) by checking which δ_i satisfy $f(x_N^*, \delta_i) = 0$.

Motivated by the fact that a convex program in n dimensional space will have at most n support contraints at the optimal solution, the problem of finding an optimal solution to SCP_N can be viewed as the problem of identifying the n supporting constraints. Can we build a consensus on what the supporting constraints are by looking at solutions of smaller sampled convex programs?

For an example, suppose we determine that N samples are needed to reach our desirable CCP_ϵ feasibility probability, and suppose that we have available one "master" processor and p "slave" processors available. For simplicity, suppose that $N = mp$ for some positve integers $m > n$. Consider the following procedure:

Parallel Sampled Convex Programming

1. For each processor $1 \leq j \leq p$, solve the sampled convex program

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & f(x, \delta_i) \leq 0 \quad i = m(j-1)+1, \ldots, mj. \end{aligned}$$

   Let x_j^* be the corresponding solution.

2. For each slave processor $1 \leq j \leq p$, determine the supporting constraints of the optimal solution x_j^*. Assuming that each sampled convex program is fully-supported with probability one, label the sample corresponding to these constraints $\delta_1^j, \ldots, \delta_n^j$. Send these samples to the master processor.

3. On the master processor, solve the sampled convex program

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & f(x, \delta_i^j) \leq 0 \quad i = 1, \ldots, k \text{ and } j = 1, \ldots, p. \end{aligned}$$

   Let x^{**} be the corresponding solution.

Another possible direction to look is to take x^{**} as some convex combination of the optimals solutions from the subproblems. Intuitively, if each of the subproblems somehow approximate CCP_ϵ, then somehow "averaging" them should give us at least something no worse.

Example Problem

We consider the classical portfolio optimization problem of selecting an allocation of one dollar over n assets with uncertain payouts r_i for $i = 1, \ldots, n$. Of these assets, we assume that the nth asset has a fixed payout (it might, for example, represent a bond with a certain return). Our objective is to maximize the value-at-risk (VaR) at a risk level of ϵ. Here VaR is the ϵ-percentile of the distribution of returns for a particular allocation. Therefore we have the following chance constrained program:

$$\begin{aligned} \underset{y,t}{\text{maximize}} \quad & t \\ \text{subject to} \quad & \mathbb{P}\left\{ \sum_{i=1}^{n-1} r_i y_i + r_n y_n \geq t \right\} \geq 1 - \epsilon \\ & \sum_{i=1}^{n} y_i = 1 \\ & y_i \geq 0 \quad \forall i = 1, \ldots, n. \end{aligned} \tag{Portfolio CCP}$$

Here the chance constrained program models the acceptable level of risk in an investment. We will assume that the vector of payouts y is distributed according to a multivariate log-normal distribution with mean vector μ and covariance matrix Σ. A log-normal is typical

for asset payouts because it is unimodal with support on the positive real line. Even with knowledge of the probability distribution of the payouts, the feasible region is non-convex. (Is it really?)

From taking samples $r^{(j)}$ for $j = 1, \ldots, N$, the corresponding sampled convex program is

$$\begin{aligned} \underset{y,t}{\text{maximize}} \quad & t \\ \text{subject to} \quad & \sum_{i=1}^{n-1} r_i^{(j)} y_i + r_n^{(j)} y_n \geq t \quad j = 1, \ldots, N \\ & \sum_{i=1}^{n} y_i = 1 \\ & y_i \geq 0 \quad \forall i = 1, \ldots, n. \end{aligned} \tag{Portfolio SCP$_N$}$$

Portfolio SCP$_N$ has several nice properties:

- The objective function and constraints of Portfolio SCP$_N$ are linear. Therefore we can use the simplex method to solve the sampled convex subproblems.
- Because of the fixed asset n, it is possible to easily find a basic feasible solution by taking $y_n = 1$, $y_i = 0$ for $i = 1, \ldots, n-1$ and $t = r_n$.
- The feasible allocations y sit within a simplex.
- Is t also bounded by the some of the VaR for each asset.

Here we take $\epsilon = 0.05$ and $\beta = 1 \times 10^{-5}$ to get a required sample size of $N = 5312$.

Implementation in MPI

We use MPI for communication across processors and glpk as our simplex solver.

Figure 7 shows some experimental timings for the simplex and dual simplex methods applied to our problem with $n = 200$. It shows a superlinear increase in wall clock time with respect to the number of constraints. This trend is promising for us to decompose the Portfolio SCP$_N$ into smaller subproblems to achieve time savings through parallelism.

Experiments

However, $V(x_N^*)$ cannot be explicitly calculated unless we have an analytical form for $\mathbb{P}$. Instead we often resort to estimating $V(x_N^*)$ via Monte Carlo simulation: obtaining R i.i.d. samples of δ and calculating

$$\hat{V}(x_N^*) = \frac{1}{R} \sum_{i=1}^{R} \mathbf{1}\{f(x_N^*, \delta_i) > 0\}.$$

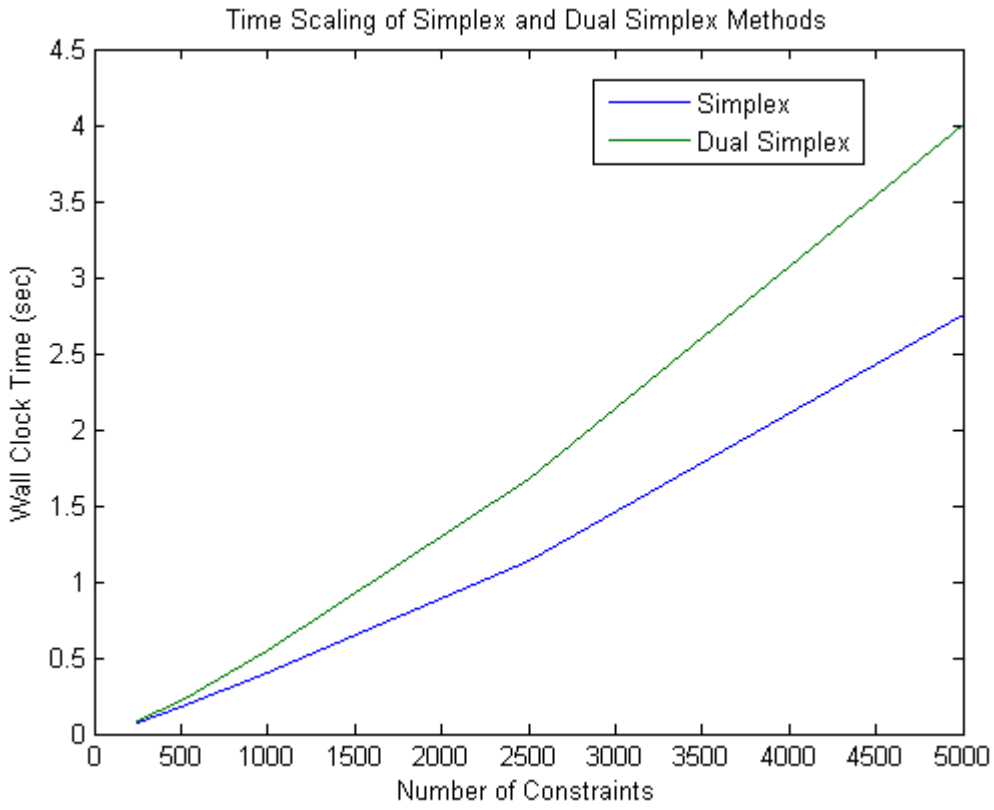

Figure 1: Wall-Clock Times for Simplex and Dual Simplex

We take M macro-replications of the procedure and calculate the following point estimates of $E[V(x^{**})]$ and $\mathbb{P}\{V(x^{**}) > \epsilon\}$:

$$\widehat{E[V(x^{**})]} := \frac{1}{M}\sum_{i=1}^{M} \hat{V}(x^{**}),$$

and

$$\widehat{\mathbb{P}\{V(x^{**}) > \epsilon\}} := \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}\left(\hat{V}(x^{**}) > \epsilon\right).$$

We use exact binomial tests (Clopper-Pearson) to form approximate confidence intervals for these two performance measures.

Sensitivity analysis (maybe).

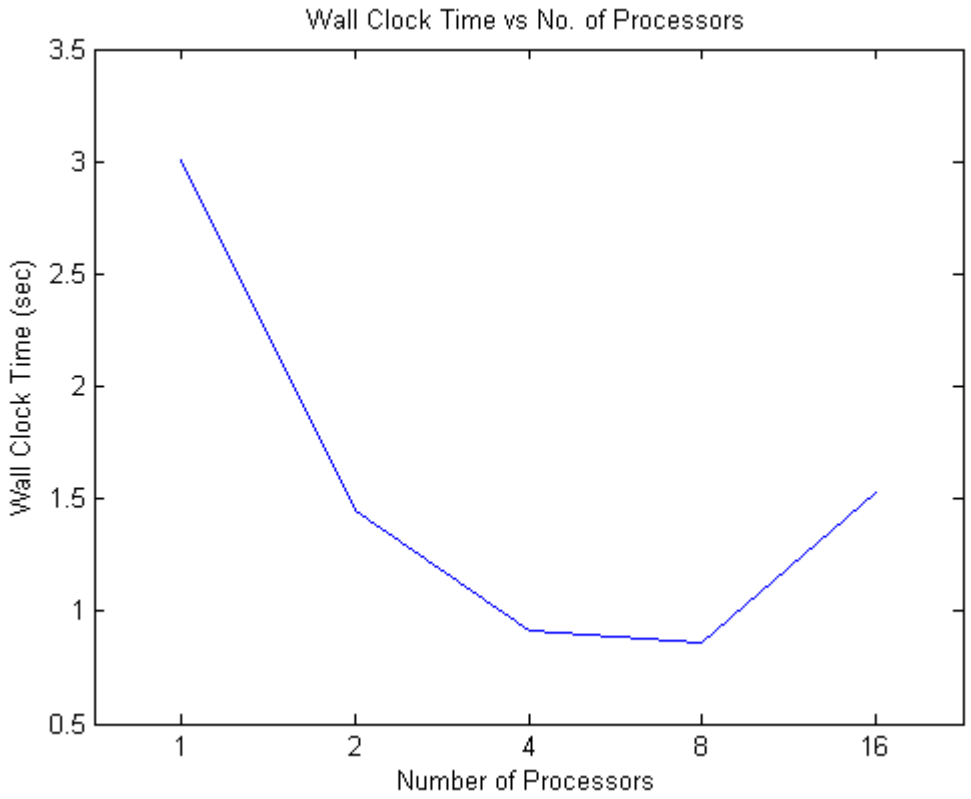

Figure 2: Wall-Clock Times by Number of Processors

Performance Analysis

Conclusions

References

[1] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, Princeton University Press, 2009.

[2] G. Calafiore and M. C. Campi, *Uncertain convex programs: randomized solutions and confidence levels*, Mathematical Programming, 102 (2005), pp. 25–46.

[3] ——, *The scenario approach to robust control design*, IEEE Transactions on Automatic Control, 51 (2006), pp. 742–753.

[4] M. C. Campi and S. Garatti, *The exact feasibility of randomized solutions of uncertain convex programs*, SIAM Journal on Optimization, 19 (2008), pp. 1211–1230.

[5] L. Carlone, V. Srivastava, F. Bullo, and G. C. Calafiore, *Distributed random convex programming via constraints consensus*, SIAM Journal on Control and Optimization, 52 (2014), pp. 629–662.

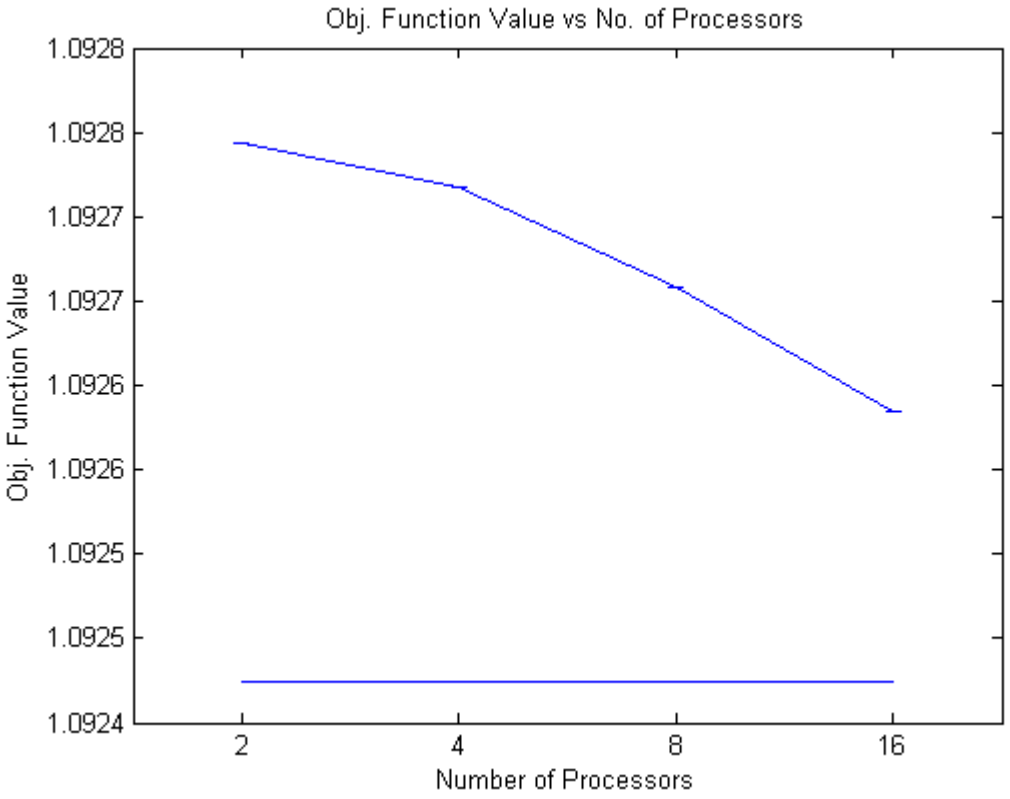

Figure 3: VaR t^* by Number of Processors

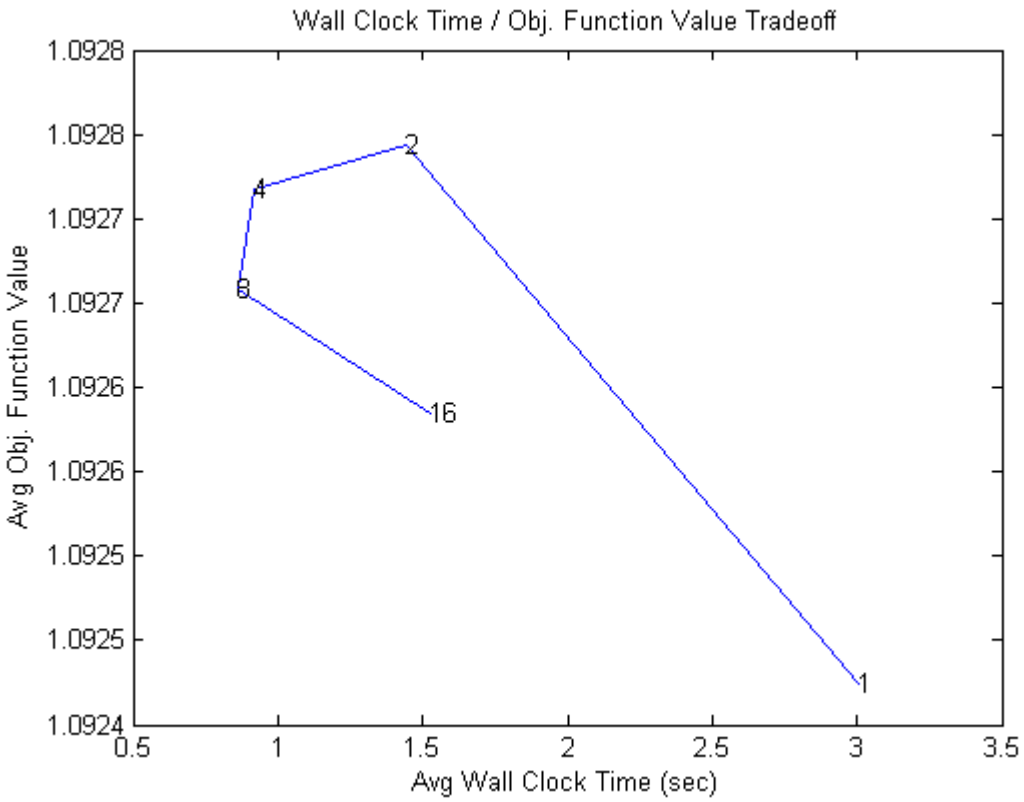

Figure 4: Trade-off between Time and Var t^*

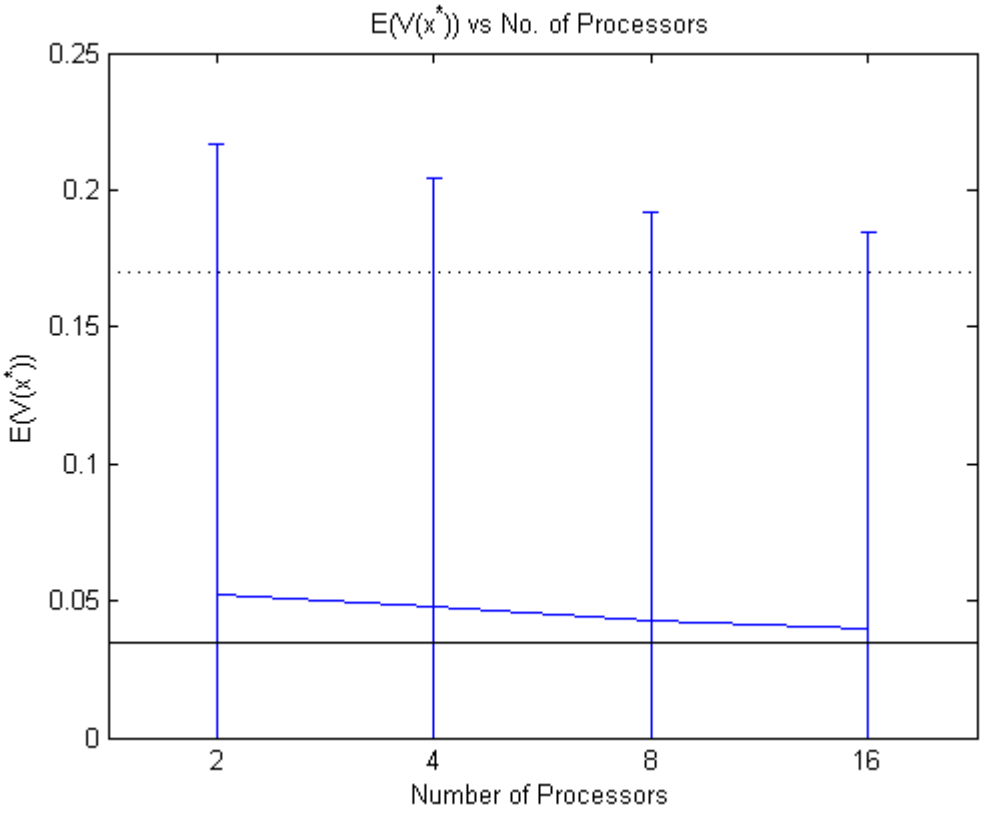

Figure 5: $E[V(x^{**})]$ by Number of Processors

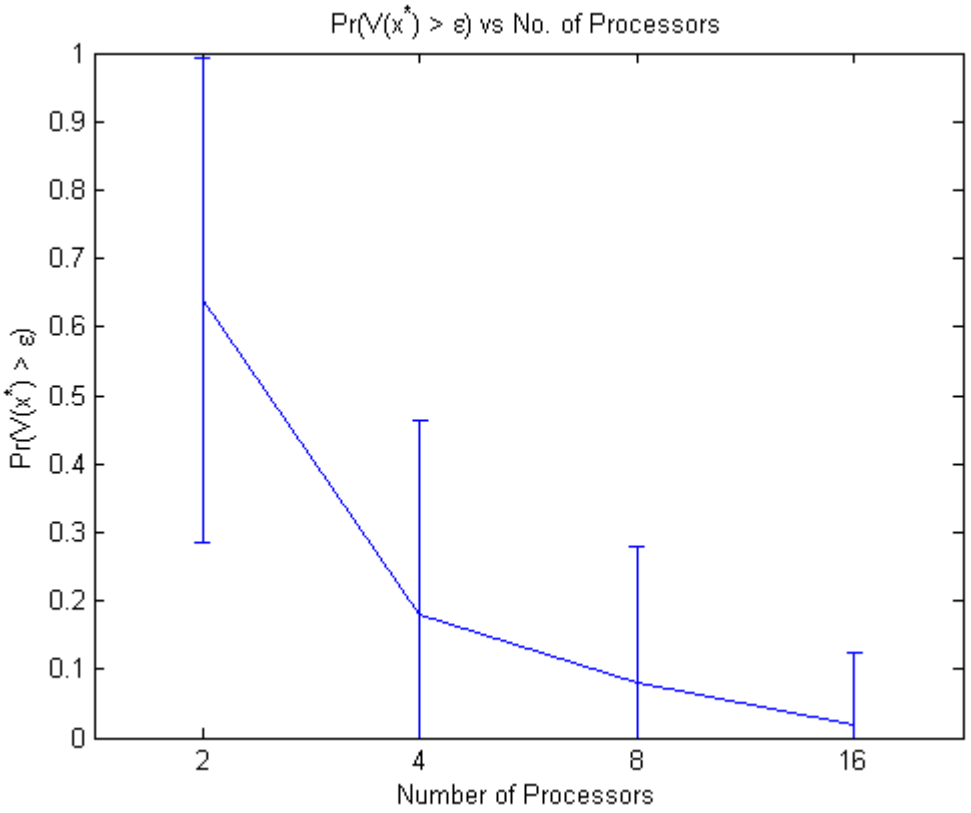

Figure 6: $\mathbb{P}\{V(x^{**}) > \epsilon\}$ by Number of Processors

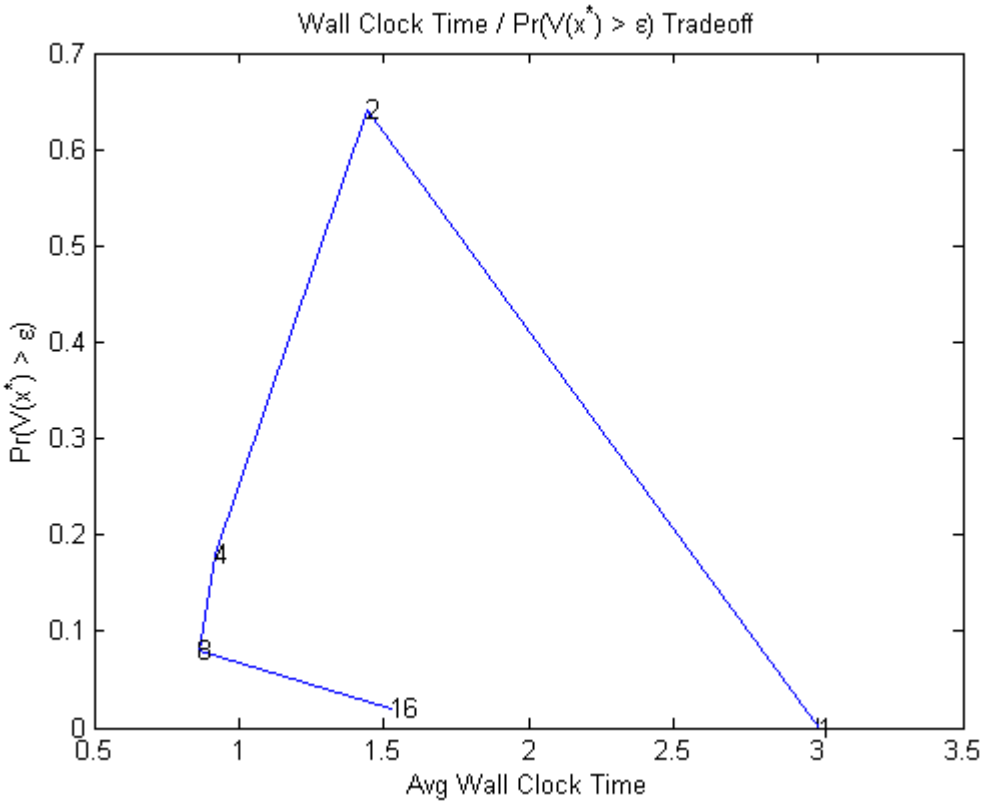

Figure 7: Trade-off between Time and $\mathbb{P}\{V(x^{**}) > \epsilon\}$