Proceedings of the 35th
Conference on Decision and Control
Kobe, Japan • December 1996



# Soft vs. Hard Bounds in Probabilistic Robustness Analysis

Xiaoyun Zhu Yun Huang John Doyle
California Institute of Technology, Pasadena, CA 91125

**Abstract**

The relationship between soft vs. hard bounds and probabilistic vs. worst-case problem formulations for robustness analysis has been a source of some apparent confusion in the control community, and this paper will attempt to clarify some of these issues. Essentially, worst-case analysis involves computing the maximum of a function which measures performance over some set of uncertainty. Probabilistic analysis assumes some distribution on the uncertainty and computes the resulting probability measure on performance. Exact computation in each case is intractable in general, and this paper explores the use of both soft and hard bounds for computing estimates of performance, including extensive numerical experimentation. We will focus on the simplest possible problem formulations that we believe reveal the difficulties associated with more general robustness analysis. More details and additional numerical experiments can be found from the web page: `http://hot.caltech.edu/~doyle`.

## 1 Motivation

In general, exact computation for robustness analysis, either worst-case or probabilistic, is intractable. It is quite easy to see intuitively why this might be so, even for linear systems with parametric uncertainty. Suppose that for fixed parameters we can compute some function $f$ to obtain a measure of performance and the cost to do so is $C$, and that we have $p$ uncertain parameters. Suppose that we want to compute $f$ for $r$ values in each parameter in order either to estimate the maximum (worst-case) or average (probabilistic) performance, or to estimate the probability that $f$ exceeds some threshold. The total number of possible combinations of parameter values is then $r^p$ and the cost of all the evaluations is $Cr^p$. Thus the growth rate in $p$ is exponential, which means that the addition of even a few parameters to a model can cause severe increases in computation. This intuition is supported by theoretical results that show that even for linear models with parametric uncertainty, evaluating virtually any robustness measure is NP hard in the number of parameters. If anything, a probabilistic framework makes this more difficult since the computation is more involved.

There are several approaches to overcoming this apparent intractability. An indirect approach which is well-known and has been the industry standard for decades is so-called Monte Carlo simulation. To experimentally estimate the distribution on $f$ given one on the parameters, we can compute $f$ at random values of the parameters. This can then be subjected to standard statistical tests like any experimental data to produce "soft" bounds on hypothesis tests. The beauty of the Monte Carlo approach is that the accuracy of the estimates trivially does not depend on the dimension of the parameter space, so there is no growth whatsoever in the computation cost with the number of parameters. The only cost is that of the function itself and the number of times it must be repeated to get a statistically significant sample size. Furthermore, Monte Carlo can be applied to any simulation or experiment, so is applicable to many problems that lack the mathematical structure for more systematic analysis.

The main difficulty with the Monte Carlo soft bounds approach is it doesn't actually compute the probability distribution of the performance, but only indirectly assess it. That is, we don't get hard bounds like "the model achieves the desired performance 99% of the time," but instead we get soft bounds like "we can be 95% confident based on the experimental data that the model achieves the desired performance 99% of the time." What this means more precisely is that a model that has acceptable peformance for 99% of the assumed uncertainty set would produce data as good as what we have observed for 95% of repeated experiments. The actual probability distributions remain unknown, and would require the prohibitive computation of multidimensional integrals with the inherent intractability described above, so it is still possible that our true probabilities are much worse than 99%. The need to use such confidence levels can be particularly annoying when they are not naturally motivated or when estimating rare events with high confidence levels, which requires an enormous number of Monte Carlo trials.

It may also be difficult to interpret the probabilities that describe both the assumptions and the results, although often the probability distributions have natural interpretations. Examples of this would be estimating the yield of chips as the result of some manufacturing process, or estimating the probability of failures for some system based on probabilities for component

0-7803-3590-2/96 $5.00 © 1996 IEEE

failure. On the other hand, we may simply want to know if anything bad can happen for some set of parameters, and there is no natural way to interpret the probability distribution of the parameters or the resulting probability distribution of the performance. This, however is an issue of motivating a probability distribution on the uncertainty, and should be separate from the issue of soft versus hard bounds. While it is possible, in principle, to use Monte Carlo to estimate soft bounds on worst-case performance, this is much more awkward than its conventional use and is unlikely to be practical.

## 2 Hard Bounds vs. Soft Bounds

There are alternatives to the Monte Carlo technique that provide different and complementary answers, and the development of such alternatives has been a driving force behind much research in robust control theory for the last 20 years. The difficulty is that if we want to avoid soft estimates we must overcome the worst-case intractability implied by the NP hardness of our problems. The approach that has proven to be most successful involves computing hard bounds, as opposed to the soft bounds available by Monte Carlo, and refining the bounds by using branch and bound.

The hard bounds approach has been primarily used to compute worst-case performance, but it could in principle be used to compute hard bounds on probabilistic measures as well. Similarly, Monte Carlo could be used to obtain soft bounds on worst-case performance, although it would require possibly unverifiable assumptions on the models. Nevertheless, it is somewhat artificial to exclusively associate Monte Carlo and probabilistic problem formulations, although historically most research has focused on computing hard bounds for worst case, and little attention has been focused on computing hard bounds for probabilistic measures.

The big advantage of hard bounds for either worst-case or probabilistic is that they are guaranteed and can be refined by branch and bound. The potential difficulty is that there exist examples for which this refinement may take prohibitively long. For many worst-case problems, it seems that such examples are extremely rare to the point that it is unlikely that anyone would ever encounter one without specifically constructing it. It is interesting to note that this is true of course with essentially all numerical algorithms, even those we think of as polynomial time, such as eigenvalue and singular value computation. For these problems, however, there is a much clearer picture of the nature of "hard" problems, whereas for the problems discussed here, the evidence is entirely numerical.

The problem of finding hard bounds on probability distributions has received almost no attention in the robust control literature. Since it involves essentially bounding the integral of some function, it appears harder than the corresponding worst-case problem of computing the maximum value of a function. This is explored in the remainder of the paper.

## 3 Computational Experience

### 3.1 Problem Description

We will begin with "$\mu$ on a box" ([4]), which is a standard robust analysis test. We will focus on perhaps the most elementary problem that is known to be NP hard, and then consider it in both worst-case and probabilistic forms. An interconnection structure is shown in Figure 1, where $M \in R^{n\times n}$ is a real square matrix and $\Delta$ contains parametric uncertainties.

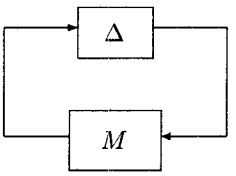


**Figure 1:** Standard Interconnected System

The uncertainty set $\mathbf{\Delta}$ has the following structure:

$$\mathbf{\Delta} := \{diag\{\delta_1, \cdots, \delta_n\} : \delta_i \in R\}$$

$$\mathbf{B\Delta} := \{\Delta \in \mathbf{\Delta} : \|\Delta\| \leq 1\}$$

The structured singular value $\mu$ for this problem is just the maximum real eigenvalue of $\Delta M$ on the box $\mathbf{B\Delta}$:

$$\mu_{\Delta}(M) = \max_{\Delta \in \mathbf{B\Delta}} \bar{\lambda}_r(\Delta M). \qquad (1)$$

Since both $M$ and $\Delta$ are real, it is easily shown that the maximum real eigenvalue must be achieved on the vertices of the parameter space $\mathbf{B\Delta}$. It is known that this problem is NP hard, which is generally taken to mean that it can not be computed in polynomial time for the worst case. The table below shows the growth rate of computation time versus problem size needed to check $\bar{\lambda}_r(\Delta M)$ on all the $2^n$ vertices of $\mathbf{B\Delta}$. Times are estimates for a SUN Ultra Sparc workstation.

| Problem Size ($n$) | | | | |
|---|---|---|---|---|
| 4 | 8 | 16 | 32 | 64 |
| 0.01 sec | 0.3 sec | 5 min | 3 years | $9 \times 10^{10}$ years |

Obviously this exponential growth rate is devastating for large problems and the known NP hardness supports this observation. Fortunately, there are upper

and lower bounds with polynomial time algorithms. A lower bound can be obtained immediately from (1) by local search. The actual algorithm we use here is a power iteration based on [1] and [2]. It is faster and has better global convergence than standard optimization, but cannot have guaranteed global convergence (or P=NP). An upper bound for this problem is the standard $\inf_D \bar{\sigma}(DMD^{-1})$ where $D$ is diagonal. The more sophisticated bounds typically used for real problems reduce to this because the matrix $M$ is real. The actual algorithm used in the tests in this paper is based on interior point methods for LMIs.

As our first numerical experiment, we computed these bounds for matrices of size 4, 8, 16, 32, and 64, with 50 matrices of each size. The elements of the matrices were zero mean normally distributed psuedorandomly generated floating point numbers. The bounds either achieved a normalized error of $10^{-3}$ or a maximum number of iterations were run. Figure 2 shows cumulative distributions of the bound error function $1 - \frac{lb}{ub}$. The $y$ axis is the cumulative percentage of problems that had $x \geq 1 - \frac{lb}{ub}$ versus the $x$ axis which is $x = 1 - \frac{lb}{ub}$ on a logarithmic scale. Note that the bounds degrade somewhat with problem size, but that even for $64 \times 64$ matrices, they are usually within .2. Unfortunately, as can be seen in the upper right hand corner of the figure, the bounds are occasionally quite poor for any size problem. These cases can be improved substantially using a very simple branch and bound scheme, as shown in Figure 3.

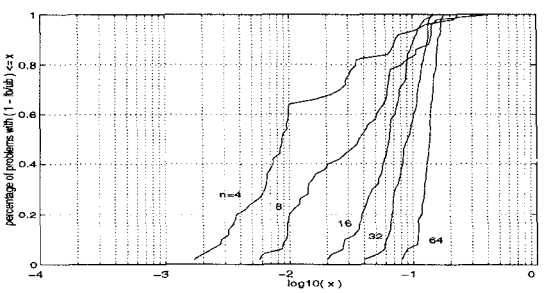


**Figure 2:** Cumulative distribution of $1 - \frac{lb}{ub}$ for dimension [4 8 16 32 64], with 50 random matrices of each dimension. No branching is performed.

### 3.2 Branch And Bound

Branch and Bound is a general technique for those optimization problems whose bounds depend on the domain of the problem. It has been proven to be quite useful in refining the bounds for problems such as the one considered here [3]. Our experience has shown that the average quality of the bounds themselves is critical. The intuition behind this is that there are occasionally bad problems where the bounds are poor, but that branching creates new problems where the bounds are good. For this to be successful, the bounds must be

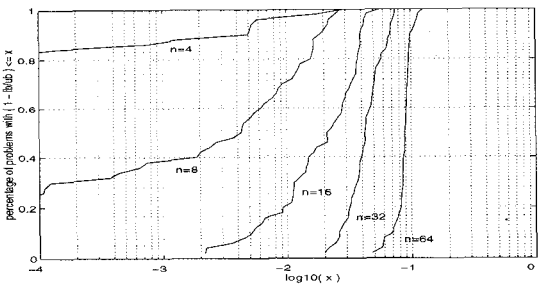


**Figure 3:** Cumulative distribution of $1 - \frac{lb}{ub}$ for the same 50 random matrices for each size. The bounds are obtained after branching 10 times.

good on average so that the branching process moves bad problems into easy problems. Interestingly, it has seemed less critical that the branching scheme be particularly clever.

These points can be readily illustrated on the problem we are considering. We used a very naive branching scheme which consisted of a simple heuristic to chose a branch variable, followed by splitting that variable into 2 equal parts, creating 2 new independent problems on which the bounds are computed. (A more sophisticated algorithm would optimize both the variable chosen and the location of the cut.) The global lower bound is the maximum over all the local lower bounds and the global upper bound is the maximum over all the local upper bounds. A branch can be pruned when its local upper bound is lower than the global lower bound. It is essential that branches be pruned effectively to avoid exponential growth. Just cutting each variable once produces $2^n$ subproblems unless some of them are pruned in the process.

Figure 3 shows the results of this algorithm applied to the same problems as in Figure 2 with 10 branches performed on each problem. Note that this made substantial improvements in the bound quality, with only a small fraction of the $n = 64$ problems remaining with a gap of greater than .1. Figure 4 focuses on the worst problems for each problem size and plots the number of branches required to achieve a given error in the bound, for 15%, 10%, 5%, 2% and 1%. Since this is a log-log scale, straight lines indicate polynomial growth, and flat lines indicate no growth.

It is not clear what the growth rates are for small percentages, as the 2% and 1% cases where only done up to size 16. Beyond that, the computation time was too great (several hours per problem) to be practical on individual workstations. Note that the worst problem required exactly 3 branches for each problem size to reach 15%, and that 10% was easily achieved for all problem sizes. This further supports the notion that branch and bound can easily take the worst-case prob-

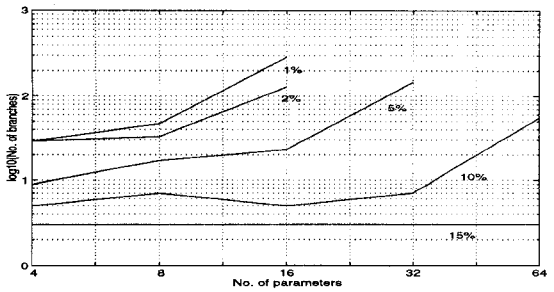


**Figure 4:** No. of branches versus problem size for various tolerances, for the worst problem out of 50 random matrices in each size. The vertical axis is on a $\log_{10}$ scale.

lems and get them to roughly the level that the bounds achieve on average, but not much better. It is possible that more sophisticated branching schemes would improve on this, and certainly the branch and bound algorithms are embarrasingly parallelizable, but we have not yet explored these possibilities. Of course, there must exist truly bad examples where even very clever branch and bound fails (or P=NP), but these seem so rare that they are very unlikely to be encountered in practice. This latter assertion must be supported by exactly the type of numerical experiments we are showing here.

The point of these numerical experiments is to underscore the point that branch and bound can be used effectively to overcome the inherent intractability of worst-case robustness analysis. The key seems to be to have good bounds, where good here means good on average. Even naive branching schemes can then be relatively effective in refining the bounds in those cases where they are poor.

## 4 Probabilistic Robustness Analysis

Our aim is to apply branch and bound techniques to estimate probability distributions for robustness analysis similar to the way it was used for worst-case. Clearly, computing the largest value of a function (worst-case) is easier than computing more general quantities as would arise in probabilistic analysis. Nevertheless, if we want more than soft bounds we must somehow address this issue. Simple modification can be carried out on the worst-case hard bound algorithm to evaluate the probability distribution of the maximum real eigenvalue function on the unit box $\mathbf{B}\boldsymbol{\Delta}$. Again the probability distribution itself can not be computed accurately. We can only get bounds on it. Two kinds of bounds are computed in our numerical test.

**Hard upper bound:** To estimate the probability of a function being above a certain level, say, $\gamma$, we can get an upper bound on it if we know that the upper bound of the function is less than $\gamma$ on some subregion. This bound is guaranteed, so it is a hard upper bound, which can be refined by the branch and bound algorithm.

**Soft bound:** In a standard statistical test, a large number of points are randomly picked from the parameter space according to a certain probability distribution and the function is evaluated on these points. Although more sophisticated statistical analysis can be done, we will simply plot the sorted values of the function obtained from this sampling and view this as a soft bound (perhaps *estimate* would be a better term since this is neither an upper nor a lower bound) on the true probability distribution of the function. Since we are usually interested only in large values of the function, hard bounds can be useful here in excluding regions of parameter space so that the soft bound can be generated more efficiently.

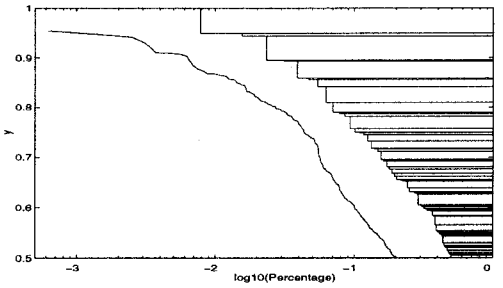


**Figure 5:** Hard upper bound and soft bound for a size 4 problem, after branching 100 times, with *ratio* = 0.5. The horizontal axis is the probability of the function value $> y$ on a $\log_{10}$ scale. The rectangles represent the hard upper bound while the solid line represents the soft bound.

In Figure 5 the hard upper bound and soft bound for a particular problem are shown. The width of the upper bound rectangles indicates the incremental volume for a subregion on which a hard upper bound was found equal to the lower boundary of the rectangle. As in the worst-case computation, there always exists a gap between the two kinds of bounds, which can be used to determine their quality. From the experience of a great number of numerical experiments we know that the extension of branch and bound techniques from the worst-case to probabilistic computation is not trivial, which is not surprising since essentially getting the probabilistic bounds involves estimating the shape of a function over the whole parameter space while worst-case computation involves finding the maximum of the function.

### 4.1 Two Special Cases

To see the difficulties we might anticipate in extending the worst-case hard bounds techniques to probabilistic formulations, let's consider two cases that are in a sense the extreme ends of the space of examples, but for

which we can analytically compute probability distributions. For simplicity, we always assume the parameters are uniformly distributed on the unit box $\mathbf{B\Delta}$. Let $P(\epsilon)$ denote the induced probability of $\bar{\lambda}_r(\Delta M)$ being no smaller than $1-\epsilon$.

For $M = I$, $\bar{\lambda}_r(\Delta M) = \max(\delta_i)$, so $\mu_\Delta(M) = 1$. It's easy to show that $P(\epsilon) = 1 - (1 - \frac{\epsilon}{2})^n$. Note that $P \to 1$ as $n \to \infty$. For worst-case the bounds are exact so there is no need to branch. It is hard to use our naive branch and bound techniques to get hard bounds on $P(\epsilon)$ though, as the function achieves its maximum on all the faces of the unit box $\mathbf{B\Delta}$ with any $\delta_i = 1$. Whichever parameter we choose to cut, both of the two new branches have the same bound.

As the opposite extreme suppose $M$ is rank one and has already been scaled by $DMD^{-1}$ so that $M = aa'$, where $a = [a_1\ a_2\ \cdots\ a_n] \in R^n$ and $\|a\| = 1$. Then

$$\bar{\lambda}_r(\Delta M) = \sum_{i=1}^{n} a_i^2 \delta_i.$$

So $\mu_\Delta(M) = 1$ with the worst-case achieved at the vertex where all $\delta_i = 1$. Again, the worst-case bounds are exact so no branching is required. It is easily checked, however, that our branch and bound scheme to bound the probability distribution is prohibitively expensive, even though we can compute the $P(\epsilon)$ analytically. For simplicity, assume $a_i^2 = \frac{1}{n}$, $i = 1, 2, \ldots, n$, then we have

$$P(\epsilon) \le \frac{(\frac{n\epsilon}{2})^n}{n!}, \tag{2}$$

The equality holds when $\epsilon \le \frac{2}{n}$. It is possible to get $P(\epsilon)$ in general but the formula is very messy. Note that in contrast to the $M = I$ case, $P \to 0$ very rapidly as $n \to \infty$. It's easy to show that this is the hardest case ($P$ is smallest for fixed $\epsilon$ and $n$) for all the rank-one matrices with $\mu = 1$. For general rank-one problems we have

$$P(\epsilon) = \frac{(\frac{\epsilon}{2})^n / \prod_{i=1}^{n} a_i^2}{n!} \ge \frac{(\frac{n\epsilon}{2})^n}{n!},$$

if $\frac{\epsilon}{2a_i^2} \le 1$, $i = 1, 2, \ldots, n$. The minimum is achieved when $a_i^2 = \frac{1}{n}$, which is exactly the simplest case we consider.

For both of these cases, it is trivial to compute the worst-case bounds, and apparently extremely difficult to compute hard bounds on the probability distribution by naive application of branch and bound. The rank one problem appears particularly problematic, and we'd expect general random matrices to have similar characteristics. As the first step in verifying this, we generate 20 random matrices of size 4 with 10 of them being rank-one and 10 of them not. All of them are normalized to have $\mu = 1$. Then we compute the soft bound of $P(\epsilon)$ for $0 < \epsilon \le 0.05$ in the neighborhood of the worst-case. The average values of the soft bounds are shown in Figure 6. The dotted line represents the theoretical value of $P(\epsilon)$ for the worst rank-one problem (2). The dashed line is from the rank-one random matrices, while the solid line is from the general random matrices. This plot clearly shows that in the neighborhood of the worst-case, the general matrices have very similar characteristics as the rank-one matrices, as we'd expect. Unfortunately, we know we can't easily compute accurate hard bounds on the rank-one problem, so we can reasonably anticipate having similar difficulties with general random matrices. This has been verified by numerical experimentaion, as is shown in the remainder of this paper.

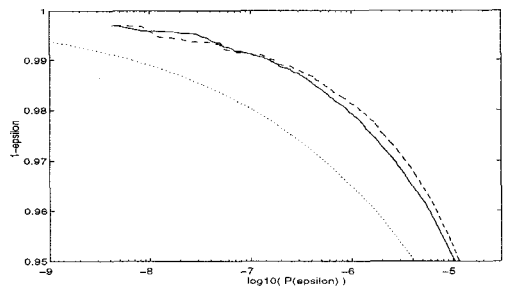


**Figure 6:** Soft bounds on $P(\epsilon)$ for different kinds of problems for $n = 4$. The horizontal axis is on a $\log_{10}$ scale.

### 4.2 Numerical Experiments

We will compute the hard upper bound and the soft bound of the probability of the function value being above a threshold $ratio * \mu$, that is $P(1 - ratio * \mu)$, using some of the same random matrices tested for worst-case. For size 4, 8 and 16, the worst-case bounds are already within 1%, so the worst-case upper bounds are used to normalize the matrices. Separate tests are performed with $ratio$ set at 0.5 and 0.9. The soft bounds for various problem size are illustrated in Figure 7 and Figure 8, where the vertical axis is $P(1 - ratio * \mu)$ on a $\log_{10}$ scale and each point is one matrix. As expected, the average of the soft bounds goes down with $n$. When $ratio = 0.9$ and $n = 16$ we found no points above the threshold for $10^6 \sim 10^8$ random points. So we can only get an "upper" bound on the soft bound from this test which is indicated by 'x' in Figure 8. More interesting is the ratio of the soft bound to the hard upper bound for various dimensions shown in Figure 9 and Figure 10. Different symbols 'x', 'o', and '+' represent the ratios after branching 100, 200, and 1000 times respectively. Note in Figure 9 that the ratios for $n = 16$ are independent of the number branches, because the hard upper bounds stay at 1 for all 10 matrices. This suggests that when the threshold is well below the maximum and the dimension is large, we get essentially no pruning of subregions, as predicted by our rank one analysis.

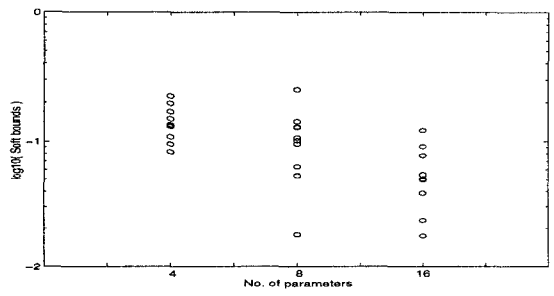


**Figure 7:** Soft bound versus problem size for 10 random matrices of each size, with $ratio = 0.5$.

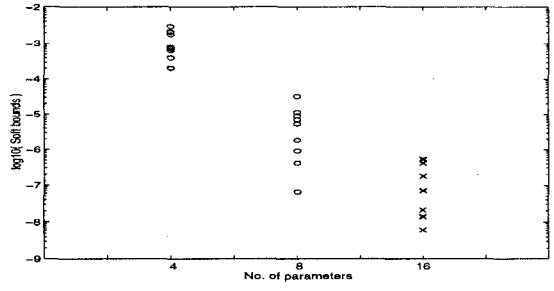


**Figure 8:** Soft bound versus problem size for the same 10 random matrices of each size, with $ratio = 0.9$. The symbol 'x' means a soft upper bound on the probability.

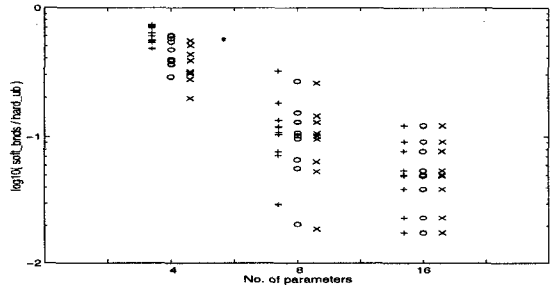


**Figure 9:** Ratio of soft bound to hard upper bound versus problem size for 10 random matrices of each size, with $ratio = 0.5$.

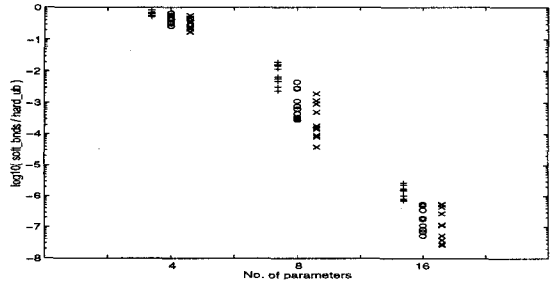


**Figure 10:** Ratio of soft bound to hard upper bound versus problem size for 10 random matrices of each size, with $ratio = 0.9$.

The ratio of the bounds gets smaller as the dimension goes up, expecially when $ratio = 0.9$. A natural suspicion is that the quality of the worst-case bounds are not accurate enough to give a good estimation of the probability bounds, but further numerical experiments suggest that this is not the case. First, we can get the worst-case bounds quite tight (within 1%) after branching a certain number of times for problems of the size we are testing (up to 16). Second, recall that we can compute exact bounds by checking all vertices, which is feasible through $n = 8$. Thus we can recompute the probability bounds on the same matrices used above but with exact bounds and compare the results. We did this and it made no significant difference.

## 5 Conclusion

There is strong motivation for developing hard bounds for probabilistic analysis to augment the conventional Monte Carlo soft bounds approach. We have explored the direct application of branch and bound algorithms of the type that have proven successful in worst-case analysis to the more computational challenging problem of probabilistic analysis. Numerical experiments have confirmed our speculation based on special cases that for certain classes of problems, such algorithms are unable to break the intractability of probabilistic analysis. The experiments also suggest that good bounds are not enough, and more clever branching schemes will be necessary.

Interestingly, the essential difficulties in this extension seem to be present in rank-one problems, which are both trivial from a worst-case perspective and can be treated analytically. It would probably be fruitful for short term research to focus on rank-one problems. Perhaps more importantly, we need additional thinking on what are the most sensible problem formulations for probabilistic robustness analysis, since there are many more choices than for worst-case.

## References

[1] Young, Peter M., Newlin, Matthew P. and Doyle, John, 1992, "Practical Computation of the Mixed $\mu$ Problem", *Proc. American Control Conference*, pp. 2190-2194.

[2] Newlin, Matthew P. and Glavaski, Sonja, 1995, "Advances in the Computation of the $\mu$ Lower Bound", *Proc. American Control Conference*, pp. 442-446.

[3] Newlin, Matthew P. and Young, Peter M., 1992, "Mixed $\mu$ Problems and Branch and Bound Techniques", *Proc. 31st IEEE Conference on Decision and Control*, pp. 3175-3180.

[4] Newlin, Matthew P., 1996, Model Validation, Control, and Computation, Ph.D dissertation, California Institute of Technology.