Computational Complexity of ?? Calculation

Conference Paper *in* IEEE Transactions on Automatic Control · July 1993

DOI: 10.1109/9.284879 · Source: IEEE Xplore

Richard D Braatz
Massachusetts Institute of Technology

Peter M. Young
Colorado State University

John Doyle
California Institute of Technology

Computational Complexity of μ Calculation

Richard P. Braatz, Peter M. Young, John C. Doyle, and Manfred Morari

***Abstract*— The structured singular value μ measures the robustness of uncertain systems. Numerous researchers over the last decade have worked on developing efficient methods for computing μ. This paper considers the complexity of calculating μ with general mixed real/complex uncertainty in the framework of combinatorial complexity theory. In particular, it is proved that the μ recognition problem with either pure real or mixed real/complex uncertainty is NP-hard. This strongly suggests that it is futile to pursue exact methods for calculating μ of general systems with pure real or mixed uncertainty for other than small problems.**

I. Introduction

Robust stability and performance analysis with real parametric and dynamic uncertainties can be naturally formulated as a structured singular value (or μ) problem, where the block structured uncertainty description is allowed to contain both real and complex blocks. It is assumed that the reader is familiar with this type of robustness analysis, as space constraints preclude covering this here. For a collection of papers describing the engineering motivation and the computational approaches, see [3] and the references contained within.

In this work, we determine the computational complexity of μ calculation with either pure real or mixed real/complex uncertainty. To apply computational complexity theory, we formulate μ calculation as a *recognition problem* (a "yes" or "no" problem). We show that this recognition problem is NP-hard, i.e., at least as hard as the NP-complete problems.

The exact consequences of a problem being NP-complete is still a fundamental open question in the theory of computational complexity, and we refer the reader to Garey and Johnson [5] for an in-depth treatment of the subject. However, it is generally accepted that a problem being NP-complete means that it cannot be computed in polynomial time in the worst case. It is important to note that being NP-complete is a property of the problem itself, not of any particular algorithm. The fact that the mixed μ problem is NP-hard strongly suggests that, given *any* algorithm to compute μ, there will be problems for which the algorithm cannot find the answer in polynomial time.

The terminology of computational complexity theory is used extensively in this note. The definitions for NP-complete, NP-hard, recognition problems, and other terms agree with those in the well-known textbooks by Garey and Johnson [5] and Papadimitriou and Steiglitz [8].

The proofs are simple. First, we show that indefinite quadratic programming can be cast as a μ problem of "roughly" the same size. Since the recognition problem for indefinite quadratic programming is NP-complete, the μ recognition problem must be NP-hard.

Nomenclature: Matrices are upper case; vectors and scalars are lower case. $\mathcal{R}$ is the set of real numbers; $\mathcal{C}$ is the set of complex numbers; $\mathcal{Z}$ is the set of integers; $\mathcal{Q}$ is the set of rationals. $\overline{\sigma}(A)$ is the maximum singular value of matrix A and I_r is the $r \times r$ identity matrix. Define the set $\boldsymbol{\Delta}$ of block diagonal perturbations by

$$\boldsymbol{\Delta} \equiv \Big\{ \operatorname{diag}\{\delta_1^r I_{r_1}, \cdots, \delta_k^r I_{r_k}, \delta_{k+1}^c I_{r_{k+1}}, \cdots, \delta_m^c I_{r_m}, \Delta_{r_{m+1}}, \cdots, \Delta_{r_l}\} | \; \delta_i^r \in \mathcal{R}, \delta_i^c \in \mathcal{C}, \Delta_{r_i} \in \mathcal{C}^{r_i \times r_i}, \sum_{i=1}^{l} r_i = n \Big\}. \tag{1}$$

Let $M \in \mathcal{C}^{n \times n}$. Then $\mu_\Delta(M)$ is defined as

$$\mu_\Delta(M) \equiv \begin{cases} 0 & \text{if there does not exist } \Delta \in \boldsymbol{\Delta} \text{ such that } \det(I - M\Delta) = 0, \\ \left[\min_{\Delta \in \boldsymbol{\Delta}} \{\overline{\sigma}(\Delta) | \det(I - M\Delta) = 0\}\right]^{-1} & \text{otherwise.} \end{cases} \tag{2}$$

Without loss of generality, we have taken M and each subblock of Δ to be square.

II. Computational Complexity of μ Calculation

We first show that indefinite quadratic programming is a special case of a μ problem. Let $x, p, b_l, b_u \in \mathcal{R}^n$, $A \in \mathcal{R}^{n \times n}$, and $c \in \mathcal{R}$. Define the quadratic programming problem

$$\max_{b_l \le x \le b_u} |x^T A x + p^T x + c| \tag{3}$$

where A can be indefinite. In the following theorem, we cast the aforementioned problem as a μ problem.

Theorem 2.1 (Quadratic Programming Polynomially Reduces to a μ Problem): Define

$$M = \begin{bmatrix} 0 & 0 & kw \\ kA & 0 & kA\overline{x} \\ \overline{x}^T A + p^T & w^T & \overline{x}^T A \overline{x} + p^T \overline{x} + c \end{bmatrix}, \tag{4}$$

$$\boldsymbol{\Delta} = \{\operatorname{diag}[\delta_1^r, \cdots, \delta_n^r, \delta_1^r, \cdots, \delta_n^r, \delta^c] | \delta_i^r \in \mathcal{R}; \delta^c \in \mathcal{C}\}, \tag{5}$$

$$\tilde{\boldsymbol{\Delta}} = \{\operatorname{diag}[\delta_1^r, \cdots, \delta_n^r, \delta_1^r, \cdots, \delta_n^r, \delta_{n+1}^r] | \delta_i^r \in \mathcal{R}\}, \tag{6}$$

$$\overline{x} = \tfrac{1}{2}(b_u + b_l), \tag{7}$$

$$w = \tfrac{1}{2}(b_u - b_l). \tag{8}$$

Then $\mu_\Delta(M) = \mu_{\tilde{\Delta}}(M)$, and

$$\mu_\Delta(M) \ge k \Leftrightarrow \max_{b_l \le x \le b_u} |x^T A x + p^T x + c| \ge k. \tag{9}$$

This implies that the indefinite quadratic program (3) polynomially reduces to both a real μ problem, and a mixed μ problem.

Proof: The proof is trivial for $k = 0$, so assume $k > 0$. The idea is to treat the constraints as uncertainty and the objective function as the performance objective of a robust performance problem (see [4] for a description of the robust performance problem). The constraint set is

$$\{x | b_l \le x \le b_u\} = \{x | x = \overline{x} + \Delta^r w; \; \Delta^r = \operatorname{diag}[\delta_1^r, \cdots, \delta_n^r]; \; \delta_i^r \in [-1, 1]\}. \tag{10}$$

For convenience, define an artificial output $y \in \mathcal{R}$ and an artificial input $d \in \mathcal{R}$. Then the quadratic programming problem can be

Manuscript received July 21, 1992; revised March 25, 1993. The work of R. D. Braatz was supported by the Fannie and John Hertz Foundation.

R. D. Braatz and M. Morari are with the Department of Chemical Engineering, California Institute of Technology, Pasadena, CA 91125 USA.

P. M. Young and J. C. Doyle are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA.

IEEE Log Number 9216456.

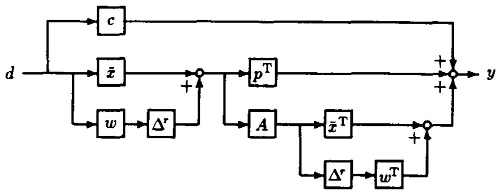

Fig. 1. Equivalent block diagram for quadratic programming problem.

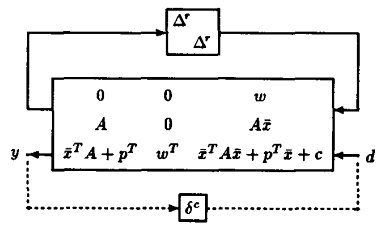

Fig. 2. Quadratic programming as a robustness problem.

written as the block diagram in Fig. 1. Block diagram manipulations give us the block diagram in Fig. 2, where we have augmented the block diagram with a performance block δ^c. The optimization objective is the input-output relationship between d and y. Define $\Delta_U = \text{diag}[\Delta^r, \Delta^r]$, N by

$$N = \left[\begin{array}{c|c} N_{11} & N_{12} \\ \hline N_{21} & N_{22} \end{array}\right] = \left[\begin{array}{cc|c} 0 & 0 & w \\ A & 0 & A\bar{x} \\ \hline \bar{x}^T A + p^T & w^T & \bar{x}^T A \bar{x} + p^T \bar{x} + c \end{array}\right], \tag{11}$$

and the linear fractional transformation (LFT) $F_u(N, \Delta_U)$ by

$$F_u(N, \Delta_U) = N_{22} + N_{21}\Delta_U(I - N_{11}\Delta_U)^{-1}N_{12}. \tag{12}$$

Since $\det(I - N_{11}\Delta_U) = 1$, the inverse in (12) is well defined. We have

$$\begin{aligned} \max_{b_l \le x \le b_u} |x^T A x + p^T x + c| &= \max_{\substack{x = \bar{x} + \Delta^r w \\ \|\delta^r\| \le 1 \\ \delta_i^r \in \mathcal{R}}} |x^T A x + p^T x + c| \\ &= \max_{\|\Delta_U\| \le 1} |F_u(N, \Delta_U)| \\ &= \max_{\|\Delta_U\| \le 1/k} \overline{\sigma}(F_u(M, \Delta_U)). \end{aligned} \tag{13}$$

Since $\mu_{\Delta_U}(M_{11}) = 0 < k$, we can apply the robust performance theorem of [4] to give (9). Since $F_u(M, \Delta_U)$ has no dynamics and is 1×1, the complex perturbation δ^c can be replaced by a real perturbation.

It can easily be shown that the μ problem in (9) is described by less than four times the number of parameters of the quadratic program. QED

Remark 2.2: Theorem 2.1 can be generalized to handle general linear constraints instead of the simple ones in (3). Any unbounded linear constraints can be converted through a bilinear transform to bounded linear constraints. All bounded linear constraints can be treated as uncertainty—the details are left to the reader. Unfortunately, for general linear constraints the resulting μ problem is impractically large. Theorem 2.1 can also be modified to solve the optimization problem that does not have the absolute value in the objective. The idea is simple: the maximizing x does not depend on c, so choose $c > 0$ very large. Then solve the resulting "absolute value" μ problem. The maximizing x for this problem will solve the original problem. Minimizations can be handled just as easily as maximizations—choose $c < 0$ very large in magnitude and solve the resulting "absolute value" μ problem. We do not show the details of these generalizations here because the generality is not needed to prove the main results of this work.

Remark 2.3: Any nonlinear programming problem with an LFT of x and x^T as an objective and general linear constraints can be written as a block diagram like that of Fig. 1. The block diagram can always be rearranged to be in the form of Fig. 2, where $y = F_u(N, \boldsymbol{\Delta}_U)d$, but with a different N and $\boldsymbol{\Delta}_U$. This block diagram has an equivalent μ problem. Therefore, any nonlinear programming problem, with an LFT of x and x^T as an objective and general linear constraints can be cast as a μ problem. It is not clear how to efficiently write a given nonlinear (e.g., polynomial) objective as an LFT in terms of x and x^T except for the specific cases of linear and quadratic programming. But we have good methods for solving linear and quadratic (at least in the definite and semi-definite cases) programs—what might be interesting in terms of computation would be to solve optimizations with more difficult objective functions. The well-known lower and upper bounds (see [11] for a summary) commonly used to approximate μ are bounds on the maximum of the LFT programming objective. The x that achieves the value of the lower bound can be calculated from the perturbation that achieves the lower bound from (7), (8), and (10). The error in the objective in using x from the lower bound algorithm instead of the optimal x is no greater than the difference between the upper and lower bounds.

To apply computational complexity theory, we must write the calculation of μ as a *recognition problem* (a "yes or no" problem). Consider μ with $M \in \mathcal{Q}^{n \times n}$, $k \in \mathcal{Q}$, and mixed real/complex uncertainty blocks. Define the recognition problem $\Phi :=$ "Is $\mu \ge k$?" = "Does there exist a perturbation of magnitude k^{-1} that destabilizes the system?"

The next lemma is essentially from [6]. This work is important because it was the first to use the techniques of discrete combinatorial complexity theory to study the computational difficulty of continuous optimization problems.

Consider $d_i \in \mathcal{Q}$ for $i = 0$ to n, and $k \in \mathcal{Q}$. Define the following nonconvex quadratic program:

$$q := \max_{0 \le x_i \le 1} \left(\sum_{i=1}^{n} d_i x_i - d_0 \right)^2 + \sum_{i=1}^{n} x_i(1 - x_i). \tag{14}$$

Lemma 2.4 (NP-Completeness of Indefinite Quadratic Programming): The recognition problem "Is $q \ge k$?" is NP-complete.

Proof: Murty and Kabadi [6] show that this problem is NP-hard. Vavasis [10] shows that the problem is in $\mathcal{NP}$. QED

The following theorem states that the μ recognition problem is NP-hard.

Theorem 2.5 (NP-Hardness of μ Recognition): Φ with general perturbation structure and general M is NP-hard.

Proof: The indefinite program (14) can be written as (3) through multiplications and additions ($\sim \mathcal{O}(n^2)$ operations). This problem is NP-complete by Lemma 2.4, and the quadratic program (3) polynomially reduces to a μ problem by Theorem 2.1. Thus Φ is in general at least as difficult as indefinite quadratic programming, and Φ is NP-hard. QED

Though the general μ recognition problem is NP-hard, special cases (i.e., with restrictions on the structure or field of M or Δ) may be simpler to compute. For example, when the M matrix is restricted to be rank one, the calculation of μ has sublinear growth in problem size, irrespective of the perturbation structure [1].

The case where μ has only real perturbations has received an especially large amount of attention in the μ calculation literature. The next result states that μ recognition is NP-hard for this case.

Theorem 2.6 (NP-Hardness of Real μ Recognition): Φ is NP-hard when M and the perturbations are restricted to be real.

Proof: Use the real μ problem of Theorem 2.1 in the proof of Theorem 2.5. QED

Models for real systems always have unmodeled dynamics associated with them. Unmodeled dynamics correspond to having at least one complex uncertainty *which enters nontrivially* in the μ problem. The next result states that μ recognition is NP-hard for this practically motivated class of problems.

Theorem 2.7 (NP-Hardness of Mixed μ Recognition): Let $\boldsymbol{\Delta}$ consist of both real and complex perturbations. Arrange the perturbations in $\boldsymbol{\Delta} = \operatorname{diag}\{\Delta_1, \Delta_2\}$ such that Δ_1 consists of pure real perturbations and Δ_2 consists of pure complex perturbations. Partition M compatibly, i.e.,

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{15}$$

where $\mu_{\Delta}(M)$, $\mu_{\Delta_1}(M_{11})$, and $\mu_{\Delta_2}(M_{22})$ are well-defined. Consider the class of μ problems for which $\mu_{\Delta_1}(M_{11}) < \mu_{\Delta}(M)$. Φ is NP-hard for this class of problems.

Proof: Use the mixed μ problem of Theorem 2.1 in the proof of Theorem 2.5. QED

The evaluation problem "What is μ?" is at least as difficult to solve as the recognition problem "Is $\mu \geq k$?" since the solution of the recognition problem immediately follows from the solution to the evaluation problem.

III. Comparison with Previous Results

It can be shown from results of Rohn and Poljak and Demmel [9], [2] that the recognition problem for a special case of computing μ with only real perturbations is NP-complete. This implies that the μ recognition problems for both the pure real and general cases are *NP-hard* (Theorems 2.5 and 2.6).

In this note, we use a *control approach* to studying the computational complexity of μ. The proofs use only simple linear algebra—the approach in [9], [2] involves transformation to the "max-cut problem." Theorem 2.7, which shows that including complex perturbations (which appear to be better behaved numerically, see [11]) in the μ problem does not remove the NP-hardness, follows naturally from the approach taken in this note. This result is important since practically-motivated μ problems are in this class.

Another immediate result (follows from [7]) of this note is that μ recognition remains NP-hard when the class of problems is restricted to those in which μ is a continuous function of M.

IV. Conclusion

The main results strongly suggest that it is futile to pursue exact methods for calculating μ of general systems with pure real or mixed uncertainty for other than small problems. In particular, one should not expect to find a polynomial time algorithm that calculates either real or mixed μ with general M exactly. These results do not mean, however, that practical algorithms are not possible. Practical algorithms for other NP-hard problems exist and typically involve approximation, heuristics, branch-and-bound, or local search [5], [8]. The results of Young *et al.* [11] strongly suggest that a combination of these techniques which takes into account the structure of the μ calculation problem can yield an algorithm which approximates μ in polynomial time for typical problems.

Acknowledgment

The authors thank Prof. J. Tsitsiklis at Massachusetts Institute of Technology, Cambridge, for his comments.

References

[1] J. Chen, M. K. H. Fan, and C. N. Nett, "The structured singular value and stability of uncertain polynomials: A missing link," *Control of Systems with Inexact Dynamic Models, ASME*, pp. 15–23, 1991.
[2] J. W. Demmel, "The component-wise distance to the nearest singular matrix," *SIAM J. Matrix Anal. Appl.*, vol. 13, pp. 10–19, 1992.
[3] P. Dorato and R. K. Yedavalli, eds., *Recent Advances in Robust Control.* New York: IEEE Press, 1990.
[4] J. C. Doyler, "Analysis of feedback systems with structured uncertainties," *IEE Proc. Part D*, vol. 129, pp. 242–250, 1982.
[5] M. R. Garey and D. S. Johnson, *Computers and Intractibility: A Guide to NP-Completeness.* New York: W. H. Freeman, Co., 1983.
[6] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Math. Programm.*, vol. 39, pp. 117–129, 1987.
[7] A. Packard and P. Pandey, "Continuity properties of the real/complex structured singular value," *IEEE Trans. Automat. Contr.*, vol. 38, pp. 415–428, 1993.
[8] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall, 1982.
[9] S. Poljak and J. Rohn, "Checking robust nonsingularity is NP-hard," *Math. Contr. Signals Syst.*, vol. 6, pp. 1–9, 1993.
[10] S. A. Vavasis, "Quadratic programming is in NP," *Informat. Process. Lett.*, vol. 36, pp. 73–77, 1990.
[11] P. M. Young, M. P. Newlin, and J. C. Doyle, "μ analysis with real parametric uncertainty," in *Proc. 30th IEEE Conf. Decision Contr.*, 1991, pp. 1251–1256.

On the General Solution of the State Deadbeat Control Problem

Vasfi Eldem and Hasan Selbuz

***Abstract*—In this note, the state deadbeat control problem is considered. It is shown that, after appropriate change of basis of input and state spaces, the general solution of the state deadbeat control problem can be expressed completely by the rows of the powers of system matrix. This result yields a very simple procedure for the calculation of a state feedback deadbeat control gain. It also provides the number of free parameters which could be used for further design purposes. The results are illustrated by an example at the end of the note.**

I. Introduction

The problem of constructing a constant state feedback control which derives any state to the origin in a minimum number of time steps is called the state deadbeat control problem. The interest in this problem goes back to early works of Kalman [1] on time-optimal control. Since then, the deadbeat control problem has been investigated by many researchers which has resulted in a rich variety of construction procedures. Ackerman [2], for instance, uses controllable canonical form, whereas Mullis [3] and Leden [4] employ an appropriate selection procedure for choosing n linearly independent

Manuscript received August 12, 1992; revised April 20, 1993.
The authors are with TUBITAK, Marmara Research Center, Division of Mathematics, Kocaeli 41470, Turkey.
IEEE Log Number 9216459.