PERGAMON

Automatica 36 (2000) 1249–1274

www.elsevier.com/locate/automatica

Survey Paper

A survey of computational complexity results in systems and control☆

Vincent D. Blondel[a,*], John N. Tsitsiklis[b]

[a]Department of Mathematical Engineering, Center for Systems Engineering and Applied Mechanics (CESAME), University of Louvain, B-1348 Louvain-la-Neuve, Belgium

[b]Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Received 4 November 1998; revised 16 July 1999; received in final form 6 October 1999

Abstract

The purpose of this paper is twofold: (a) to provide a tutorial introduction to some key concepts from the theory of computational complexity, highlighting their relevance to systems and control theory, and (b) to survey the relatively recent research activity lying at the interface between these fields. We begin with a brief introduction to models of computation, the concepts of undecidability, polynomial-time algorithms, NP-completeness, and the implications of intractability results. We then survey a number of problems that arise in systems and control theory, some of them classical, some of them related to current research. We discuss them from the point of view of computational complexity and also point out many open problems. In particular, we consider problems related to stability or stabilizability of linear systems with parametric uncertainty, robust control, time-varying linear systems, nonlinear and hybrid systems, and stochastic optimal control.

Keywords: Control; Discrete-event systems; Discrete-time systems; Hybrid systems; Markov decision processes; Mathematical systems theory; Neural networks; Nonlinear systems; Time-varying systems; Turing machines

1. Introduction

In order to motivate the reader, and as a preview of the types of problems and "solutions" overviewed in this paper, we start with a deceptively simple problem.

We are given two $n \times n$ real matrices A_+ and A_-, and we consider products of the form

$$A_{i_T} \cdots A_{i_2} A_{i_1}, \tag{1}$$

where T is an arbitrary positive integer and $(i_1, i_2, \ldots, i_T)$ is an arbitrary finite sequence whose elements take values in the set $\{+, -\}$. We wish to decide whether there exists some T and a sequence of length T such that the corresponding product is a stable matrix (in the sense that all of its eigenvalues have magnitude strictly less than 1).

If we are given an upper bound T^* on the allowed values of T, the problem has an obvious solution: form all the possible products and check whether one of them is stable. The amount of computation required by this solution grows exponentially with T^*. We will see in Section 3.5 that the exponential growth of the computational resources needed for a solution is probably unavoidable.

In the absence of a bound T^*, we have to form and check an infinite set of matrix products, which cannot be done in finite time under any reasonable model of computation. We may wonder whether a shortcut is possible that only involves a finite amount of computation. It turns out that none exists, and the problem is unsolvable in a precise mathematical sense; see Section 3.5.

By turning around the question of the preceding paragraph, we could ask whether *all* products (1) are stable. As will be discussed in Section 3.5, this is related to some major open problems: an algorithmic solution is not known, but a proof of unsolvability is not available either.

In this paper, we introduce background material that will make statements such as the above precise, and also

☆This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised from by Editor M. Morari. This work was supported by the NATO under grant CRG-961115, by the European Community Framework IV program through the research network ALAPEDES, by the AFOSR under grant F49620-99-10320 and by the ARO under the MURI grant DAAH04-96-1-0341. It was partially carried out while the second author was visiting the EECS department at the University of California at Berkeley.

* Corresponding author. Tel.: + 32-10-472381; fax: + 32-10-472180.
E-mail addresses: blondel@inma.ucl.ac.be (V. D. Blondel), jnt@mit.edu (J. N. Tsitsiklis).

0005-1098/00/$ - see front matter
PII: S0005-1098(00)00050-9

provide a review of existing results of this type. The key concepts that we will be working with refer to the algorithmic solvability of different problems and to the amount of resources (e.g., computation time) required. The premise behind our development is that a problem has been "solved" only if an algorithm for that problem is available. This viewpoint is an outgrowth of the pioneering work of Church, Gödel, and Turing earlier in this century, and has been refined through an additional focus on computation time requirements. It meshes very well with the availability of powerful computing machines, which has brought sophisticated systems and control approaches into the realm of the practical.

The remainder of the paper is organized as follows. In Section 2, we present models of digital computation and develop the notion of undecidability. We also introduce the concepts of polynomial-time algorithms and NP-completeness. We present some basic problems that are known to be undecidable or NP-complete, and conclude by discussing the consequences of NP-completeness results from a pragmatic point of view. In Section 3, we survey complexity-theoretic results that relate to linear systems, with a focus on time-invariant or time-varying uncertainty. In Section 4, we consider several classes of nonlinear systems, such as systems with a single nonlinearity, linear systems with componentwise nonlinearities, and hybrid systems. In Section 5, we move to stochastic optimal control problems and Markov decision theory, together with some pointers to related literature on discrete-event systems. We end with some brief conclusions in Section 6. Throughout the paper, several open problems are pointed out.

The literature on these subjects is large and growing, and our presentation could not be exhaustive. Additional information can be found in a number of more focused reviews such as Sontag (1995) (nonlinear control), Blondel and Tsitsiklis (1998) (elementary nonlinear systems), Littman, Dean and Kaelbling (1995), Rust (1996), Mundhenk, Goldsmith, Lusena and Allender (1997), (Markov decision theory), Alur et al. (1995) (hybrid systems), Bournez and Cosnard (1996), Moore (1998) (Turing machine as dynamical systems) and Orponen (1994) (recurrent neural networks). Some open problems related to the computational complexity of control questions are proposed in Blondel, Sontag, Vidyasagar and Willems (1999c, Problems 11, 14, 22, and 43).

2. Complexity theory

In this section, we introduce the main concepts from the theory of computability and complexity, with a focus on models of digital computation. An accessible introduction to the concepts of decidability/undecidability is given in Jones (1974). For textbook treatments, we refer the reader to Minsky (1967), Aho, Hopcroft and Ullman (1974), Garey and Johnson (1979), Hopcroft and Ullman (1979), Davis (1982) and Papadimitriou (1994). There is also an extensive literature on algebraic and continuous models of computation, and on problems with continuous data, which we do not cover; we refer the reader to Nemirovsky and Yudin (1983), Traub, Wasilkowski and Wozniakowski (1988) and Blum, Cucker, Shub and Smale (1998).

2.1. Problems, instances, and the size of an instance

In a typical *instance* of a computational problem, we are given input data $x_1, \ldots, x_d$ and we are asked to compute some function of them. For example, in the DETERMINANT problem, an instance consists of the entries of a given matrix and the desired outcome is its determinant. We focus on digital computation and constrain the input data to be given with a finite number of bits. For example, we will typically assume that the inputs take integer or rational values. As far as negative complexity results are concerned, this is hardly restrictive. If a function is difficult or impossible to evaluate on integer or rational inputs, then it certainly remains so with arbitrary real inputs.

Different instances of the same problem can have different "sizes". We follow the standard convention of mainstream complexity theory and define the *size* of an instance as the number of bits used to encode the input data according to a certain prespecified format. In particular, any (nonzero) integer i can be viewed as having size (bit length) approximately equal to $\log|i|$, since this is roughly the number of bits in a binary representation of i. For example, the size of an instance of the DETERMINANT problem involving an $n \times n$ integer matrix can be upper bounded by $O(n^2 \log U)$, where U is an upper bound on the magnitude of the entries of the matrix.[1]

2.2. Algorithms

Loosely speaking, an algorithm is described by a program, i.e., a finite sequence of instructions of the type encountered in common programming languages. For a more precise description, we need to specify a model of computation. A rather simple such model is provided by a *random access machine* (RAM) (Aho et al., 1974; Papadimitriou, 1994). A RAM consists of a read-only tape that contains the input data $x_1, \ldots, x_d$, an output tape on which the outcome of the computation is written, an unlimited sequence of registers used to store intermediate quantities generated in the course of the computation, and a program. Each register and each memory

[1] We use the $O(\cdot)$ notation, whose precise meaning is the following. Let f and g be functions that map positive numbers to positive numbers. We write $f(n) = O(g(n))$ if there exist positive numbers n_0 and c such that $f(n) \leq cg(n)$ for all $n \geq n_0$.

location on the input and output tapes may contain an arbitrary, possibly negative, integer. The program is a sequence of instructions, some of which may be associated with labels that are used in "jump to" instructions. The instruction set can be assumed to contain the halting instruction, the four arithmetic operations, instructions for reading from a location on the input tape into a register (respectively, writing the contents of a register to a location on the output tape), indirect addressing (e.g., read the contents of the register whose address is stored in register i, and store them in register 1), and jump instructions that move the program to a next instruction, possibly depending on the outcome of the comparison of some register to zero. The exact nature of the allowed instructions is not important, because alternative choices of the instruction set lead to computing machines with equivalent computing capabilities, as long as all critical abilities, such as jumping, are present.

In a typical execution of the algorithm (or "computation"), the input data are loaded on the input tape, and the algorithm starts carrying out the program instructions. For any given input, the computation may or may not halt (that is, reach a halting instruction). We say that the algorithm solves a particular problem if it always halts (for every instance of the problem, that is, for every choice of the input data) and produces the correct answer on the output tape. We also say that a problem is unsolvable if there exists no algorithm (under our model of computation) that will always halt with the correct answer.

One may wonder whether the class of solvable problems depends on the choice of a model of computation. According to the so-called *Church–Turing thesis*, all reasonable models of *digital* computation lead to the same class of solvable problems, and are therefore equally powerful. This thesis is supported by the fact that all reasonable models that have been proposed and studied lead indeed to the same class of solvable problems. Still, it only remains a thesis — rather than a theorem — because we do not have a precise definition of "all reasonable models".

2.3. Decidable and undecidable problems

We will now focus on so-called *decision problems*, that is, problems where the desired output is binary, and can be interpreted as "yes" or "no". An example is the problem of deciding whether a given (integer) matrix is nonsingular, which can be solved by, say, computing its determinant and comparing it to zero. This makes MATRIX NONSINGULARITY a *decidable* problem, one for which there exists an algorithm that always halts with the right answer. Similar to MATRIX NONSINGULARITY, many problems in linear algebra are decidable. But there also exist *undecidable* problems, for which there is no algorithm that always halts with the right answer. We give below one additional example of a decidable problem and then describe three examples of undecidable problems.

Let us consider the following problem. We are given a set of multivariable polynomials $p_1, \ldots, p_m$ in n real variables, with rational coefficients. Let S be the set of all $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$ that satisfy equalities and inequalities of the form $p_i(x_1, \ldots, x_n) = 0$, $p_j(x_1, \ldots, x_n) > 0$, $p_k(x_1, \ldots, x_n) \geq 0$. The problem of deciding whether S is nonempty (i.e., whether there exists a solution) is known to be decidable. More generally, we may consider questions such as whether the following statement is true:

$$\exists x_1, x_2 \forall x_3 \exists x_4, x_5 [(x_1, x_2, x_3, x_4, x_5) \in S].$$

This problem can be reduced to the previous one by a technique known as *quantifier elimination* and is therefore decidable as well; see Tarski (1951) and Seidenberg (1954).

The amount of computation required by the original methods of Tarski and Seidenberg was very high and more efficient methods have been devised since then. Consider a system of m polynomial (in)equalities, each of degree at most d, in n real variables. The problem of deciding whether the system has a solution can be solved with $m^{(n+1)}d^{O(n)}$ arithmetic operations; see Basu, Pollack and Roy (1996). Similar bounds are possible when alternations of quantifiers are present (Basu et al., 1996); see also Blum et al. (1988, Section 18.6) and the references therein.

We now turn to undecidable problems. The usual technique for proving that a problem is undecidable involves the idea of a *reduction*. Suppose that some decision problem A can be reduced to problem B, in the sense that any algorithm for B leads to an algorithm for A. Suppose, furthermore, that A is known to be undecidable. It follows that B is undecidable as well. In this proof schema, we need a prototype problem whose undecidability can be established by some other means, and which can play the role of problem A. One such problem is the *halting* problem, described below. Its undecidability can be proved by a relatively simple "diagonalization" argument (Papadimitriou, 1994).

An instance of the halting problem consists of a description of a computing machine M, e.g., a particular RAM program, together with a particular input x to M. The question is whether M will eventually halt when started with input x. A possible approach to this problem might be to run M with input x and assert YES if it halts. But if M does not halt on input x, it is not apparent how to detect this in finite time and assert that the answer is NO. One could hope to infer the answer through some form of syntactic analysis of the program of machine M, but this is ruled out by the undecidability result.

A second example is provided by Hilbert's famous tenth problem on *Diophantine equations*, which is the

following. We are given a polynomial in several variables, with integer coefficients, and we wish to decide whether it has an integer solution. Building on previous work by Davis and Robinson, this problem was proved undecidable in Matiyasevich (1970). An elementary exposition of Matiyasevich's Theorem can be found in Davis (1973).

A third example, which is remarkable in its simplicity, is *Post's correspondence* problem. We are given finitely many pairs of words[2] $(x_1, y_1), \ldots, (x_n, y_n)$, and we wish to decide whether there exists some $n \geq 1$ and a finite sequence $(i_1, \ldots, i_m)$ of integers in the range $1, \ldots, n$, such that

$$x_{i_1} x_{i_2} \cdots x_{i_m} = y_{i_1} y_{i_2} \cdots y_{i_m}.$$

This problem, proved undecidable in Post (1946), remains undecidable even if there are only seven pairs of words (Matiyasevich & Sénizergues, 1996). The problem is decidable for two pairs of words (Salomaa, 1985). The decidability of the intermediate cases $(3 \leq n \leq 6)$ is unknown but is likely to be difficult to settle. The undecidability of the Post correspondence problem can be established by reducing the halting problem to it; for a proof see Hopcroft and Ullman (1969, Section 14.2).

2.4. Time complexity

If an algorithm halts, we define its *running time* to be the sum of the "costs" of each instruction carried out. Within the RAM model of computation, arithmetic operations involve a single instruction and could be assumed to have unit cost. Realistically, however, arithmetic operations take time that increases with the size (bit length) of the integers involved. This leads us to the *bit model* in which the cost of an arithmetic operation is taken to be the sum of the sizes (bit lengths) of the integers involved.

Of course, the running time of an algorithm will generally depend on the size of the instance. Furthermore, the running time can be different for different instances of the same size. We define the running time $T(s)$ of an algorithm, as a function of size, to be the *worst-case* running time over all instances of size s.

We say that an algorithm runs in *polynomial time* if there exists some integer k such that

$$T(s) = O(s^k)$$

and we define P as the class of all (decision) problems that admit polynomial-time algorithms. Due to a combination of practical and theoretical reasons, P is generally viewed as the class of problems that are efficiently solvable. It is most often the case that a computational problem can be efficiently solved in practice if and only if it belongs in P. The emphasis on worst-case performance is not fully satisfactory, as an algorithm may be fast on most instances but take exponential time only on a small minority of pathological instances. For example, the simplex method is known to have polynomial running time, *on the average* (Schrijver, 1986). However, it is often difficult to give a precise and reasonable meaning to the term "on the average". For this reason, the vast majority of complexity-theoretic research has focused on worst-case complexity.

Note that under the bit model that we have adopted, an algorithm that uses a polynomial number of arithmetic operations is not necessarily a polynomial-time algorithm. To appreciate this distinction, notice that the number 2^{2^n} can be computed with n multiplications, by successive squaring, but this takes at least 2^n steps (just to write the output). On the other hand, this distinction disappears for algorithms that produce intermediate and final results that are integers of size (bit length) bounded by a polynomial in the size of the original instance.

To make the above statement more concrete, let us consider the inversion of integer matrices. It can be carried out using Gaussian elimination, with $O(n^3)$ arithmetic operations, but more work is needed before asserting that we have a polynomial-time algorithm. The size of an instance involving an $n \times n$ matrix is $O(n^2 \log U)$, where U is the magnitude of the largest entry of the matrix. It can be verified (see, e.g., Schrijver, 1986) that Gaussian elimination can be implemented with $O(n^3)$ arithmetic operations and so that every intermediate quantity produced is an integer of magnitude $O((nU)^{n^k})$ for some integer k. Thus, intermediate quantities have size $O(n^k \log(nU))$. The cost of each arithmetic operation is bounded by that latter quantity, and the overall running time of the algorithm is $O(n^{k+3} \log(nU))$. This is seen to be polynomial in the instance size $O(n^2 \log U)$ and we conclude that matrix inversion can be done in polynomial time.

In the sequel, the bit model will always be in effect when talking about polynomial-time algorithms. On the other hand, unless there is an explicit statement to the contrary, we will provide complexity estimates in terms of arithmetic operations, which is less tedious.

2.5. Turing machines and other models

Turing machines provide a model of digital computation which is more primitive, hence harder to "program" than random access machines. However, their primitiveness becomes an advantage when they are manipulated for the purpose of proving theoretical results.

A *Turing machine* (TM, for short) uses a finite symbol alphabet that includes a "blank" symbol. It uses a linear tape (memory), which is an infinite sequence of cells, each one containing a symbol. It also uses a "head" that has

[2] A word from a given alphabet (e.g., the alphabet $\{a, b\}$) is a concatenation of finitely many symbols from that alphabet (e.g., *aababb*), or the "empty" word. For any two words s and σ, $s\sigma$ stands for their concatenation.

a finite number of possible internal states and, at any time, scans a particular cell. When computation starts, the input is written on finitely many of the tape cells (the other cells are blank), the head is in a special "start" state, and it scans the first cell. At each time step, the machine reads the contents of the cell being scanned, and a transition is implemented. The effects of the transition are a function only of the current state and the contents of the scanned cell, and they result in (a) a new internal state, (b) a new symbol written in the scanned cell, and (c) a movement of the head one unit to the left or to the right. The computation terminates when the head enters a special "halting" state. At that point, the tape contents are interpreted as the output. A TM can be viewed as a representation of an algorithm, which is encoded in the transition function.

Most questions related to the operation of TMs are known to be undecidable (e.g., is a given TM guaranteed to halt on every input, does it halt on a particular input, etc.). Furthermore, the functions that can be computed by Turing machines are the same as those for the seemingly more powerful RAM model.

As far as time complexity is concerned, the most sensible notion of running time for TMs is to count the number of steps until the machine halts. Turing machines are capable of performing arithmetic operations on k-bit integers in time which is polynomial in k (verifying this is a simple but tedious exercise in low-level programming). Building on this observation, it can be shown that a TM can simulate a RAM with only polynomial slowdown. That is, if a RAM algorithm solves a certain problem in time $O(s^k)$ under the bit model, then there exists some l and an equivalent TM algorithm that takes time $O(s^{k+l})$. The converse is also true because RAMs are at least as powerful as TMs.

This and similar observations have led to the sometimes called *effective* Church–Turing thesis, which states that "any reasonable attempt to model mathematically computer algorithms and their time performance is bound to end up with a model of computation and associated time cost that is equivalent to Turing machines within a polynomial" (Papadimitriou, 1994).

2.6. Some polynomial-time solvable problems

We refer here to some important problems that are known to be polynomial-time solvable. We have already mentioned matrix inversion, and the same is true for problems such as solving systems of linear equations, computing determinants, or computing matrix rank. LINEAR PROGRAMMING (the problem of deciding whether a set of linear inequalities in real variables has a solution) can also be solved in polynomial time using interior point methods (Schrijver, 1986). The same is true for certain optimization problems such as convex quadratic programming (minimizing a convex quadratic function subject to linear equality and inequality constraints), or for solving linear matrix inequality (LMI) problems to within a prespecified error tolerance (Boyd, El Ghaoui, Feron & Balakrishnan, 1994).

Closer to control theory, consider the problem of deciding whether a matrix A is stable, i.e., whether its *spectral radius* $\rho(A)$ (the maximum of the magnitudes of its eigenvalues) satisfies $\rho(A) < 1$. A solution to this problem follows from the following result: the matrix A is stable if and only if the Lyapunov equation $APA^{\mathrm{T}} + I = P$ has a positive-definite solution P (Gantmacher, 1959). (In that case, the solution is unique.) Since the Lyapunov equation is linear in the unknown entries of P, we can compute a solution P (or decide it does not exist) in polynomial time. To check that P is positive definite, it suffices to compute the determinants of the n principal minors of P and verify that they are all positive. Since determinants can be computed in polynomial time, we have a polynomial-time solution to the stability problem. A similar polynomial-time solution is also possible for continuous-time stability (are all eigenvalues of A in the open-left half-plane?). The same questions about roots of *polynomials* can also be answered in polynomial time, since it is easy to construct a matrix A with a prespecified characteristic polynomial. An alternative method is to use the Routh test.

2.7. NP-completeness and beyond

There are many decidable problems of practical interest for which no polynomial-time algorithm is known, despite intensive research efforts. Many of these problems belong to a class known as NP (nondeterministic polynomial time), that includes all of P. A decision problem is said to belong to NP if every YES instance has a "certificate" of being a YES instance whose validity can be verified with a polynomial amount of computation. For example, consider the ZEROONE INTEGER PROGRAMMING problem (ZOIP). In this problem, we are given a number of zero–one variables and a system of linear equality and inequality constraints, and we wish to determine whether a solution exists. If we have a YES instance, any feasible solution is a certificate that testifies to this and whose validity can be checked in polynomial time. (Simply, check that all the constraints are satisfied.)

The ZOIP problem turns out to be a hardest problem within the class NP in the sense that every problem in NP can be reduced to ZOIP, in polynomial time (Cook, 1971). Such problems are said to be NP-complete. More precisely, if A is any problem in NP, there is an algorithm that runs in polynomial time and which, given an instance I of problem A produces an "equivalent" instance I' of ZOIP. Equivalence here means that I is a YES instance of A if and only if I' is a YES instance of ZOIP.

If we had a polynomial-time algorithm for ZOIP, then any problem in NP could also be solved in polynomial

time by first reducing it to ZOIP and then using a polynomial-time algorithm for ZOIP. It would then follow that NP is the same as the class P of polynomial-time solvable problems. Whether this is the case or not is a major open problem in theoretical computer science. It is widely believed that P $\neq$ NP, although a proof is not in sight. If indeed P $\neq$ NP, then ZOIP is not polynomial-time solvable, and the same is true for every other NP-complete problem. By now, NP-completeness has been established for thousands of problems (Karp, 1972; Garey & Johnson, 1979). We will have the opportunity of introducing several NP-complete problems later on. For now, let us mention QUADRATIC PROGRAMMING, the problem of minimizing a quadratic — but not necessarily convex — cost function over a polyhedron.

If a problem A is at least as hard as some NP-complete problem B (in the sense that B can be reduced to A in polynomial time) we will use the term NP-hard to describe this situation. In particular, a decision problem is NP-complete if and only if it is NP-hard and belongs to NP.

There are also classes of problems that are much harder than NP-complete. For example, a problem is said to belong to exponential time (EXP) if it can be solved in time $O(2^{s^k})$ for some k, where s is the instance size. Once more, EXP-complete problems are the hardest in this class and, in this case, they require *provably* exponential time (Papadimitriou, 1994).

Let us also define PSPACE, the class of all problems that can be solved by a Turing machine that uses a polynomial amount of memory (tape cells). We have NP $\subset$ PSPACE. (For example, ZOIP can be solved with polynomial memory by simply checking all possible solutions.) However, it is not known whether the containment is proper. Given the present state of knowledge, it is even possible that PSPACE = P, but this is considered highly unlikely. If a problem is shown to be PSPACE-complete, this is viewed as a very strong indication that it is not polynomial-time solvable. To summarize, we have P $\subset$ NP $\subset$ PSPACE $\subset$ EXP. It is known that P $\neq$ EXP but it is unknown which of the above inclusions are strict.

2.8. Coping with NP-completeness

NP-hardness of a problem means that it is about as difficult as ZOIP, and this is often interpreted as an indication of inherent intractability. Assuming that the conjecture P $\neq$ NP is true, an NP-hardness result eliminates the possibility of algorithms that run in polynomial time and that are correct for all problem instances. However, this is not reason enough for declaring the problem intractable and refraining from further research. Many NP-hard problems are routinely solved in practice, either exactly or approximately, using a variety of methods. For example, there are methods such as branch-and-bound that require exponential time in the worst case, but run fairly fast on many problems and instances of practical interest (Nemhauser & Wolsey, 1988).

Another point that needs to be appreciated is that not all NP-complete problems are equally hard. A standard example is provided by the 0–1 KNAPSACK problem (see, e.g., Bertsimas & Tsitsiklis, 1997)

$$\text{maximize} \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq K, \quad x_i \in \{0,1\},$$

where the data c_i, w_i, and K are nonnegative integers. The problem is NP-hard, but can be solved in *pseudo-polynomial* time, with $n^2 \max_i |c_i|$ arithmetic operations. This is not polynomial time, because the instance size is of the order of $\log K + \sum_i \log(c_i w_i)$, and the running time is an exponential function of the instance size. However, unless the numbers c_i are extremely large, this can be a practically viable algorithm.

In addition, many NP-complete (or NP-hard) problems become much easier if we are willing to settle for approximate solutions. For example, for any $\varepsilon > 0$, we can get a feasible solution of the 0–1 KNAPSACK problem, whose value is no less than $(1 - \varepsilon)$ times the optimal value, and this can be done in polynomial time, with $O(n^3/\varepsilon)$ arithmetic operations. More generally, a minimization (respectively, maximization) problem is said to admit a polynomial-time *approximation scheme* if for every $\varepsilon > 0$, there exists an algorithm (depending on ε) whose running time is polynomial in the instance size, and which outputs a solution which comes within a factor of $1 + \varepsilon$ (respectively, $1 - \varepsilon$) from the optimum. However, some NP-hard problems are not amenable to approximate solution, and there is a rich theory pertaining to this matter (Papadimitriou, 1994). We revisit this subject in Section 3.2.

Finally, let us mention the possibility of using polynomial-time randomized algorithms that provide solutions to arbitrarily high levels of accuracy and confidence (see Tempo et al. (1996), Khargonekar and Tikku (1996), Vidyasagar (1998) and Vidyasagar and Blondel (1999) for a description of such methods applied to control problems).

3. Linear systems

Many questions about linear systems are decidable, and can be decided efficiently. For example, controllability, stability, and observability of continuous- or discrete-time linear systems can be decided in polynomial time. One problem for which no algorithm is yet available is the "hyperplane hitting problem" which arises, among other situations, in sliding mode control. In this problem, we are given a discrete-time autonomous

system $x_{t+1} = Ax_t$, together with the initial state x_0, and we wish to determine whether the state x_t eventually lands on the hyperplane $c^{\mathrm{T}}x = 0$, that is, whether we have $c^{\mathrm{T}}A^t x_0 = 0$ for some $t \geq 0$. This problem is the system-theoretic transcription of a well-known problem in the theory of formal power series (Salomaa & Soittola, 1978). It is also a long-standing open problem in number theory. The problem is proved NP-hard in Blondel and Portier (1999a) but it is unknown whether it is decidable. The present consensus among number theorists is that an algorithm should exist. This has been proved to be the case for a large number of special cases, including the generic situation where the eigenvalues of A are all distinct; see Mignotte, Shorey and Tijdeman (1984), and also Samake (1996) for recent extensions.

3.1. Stable and unstable polynomials in polynomial families

Many computational complexity results in control theory arise when considering *families* of polynomials or matrices. The interest in the stability analysis of such families emerged in the early 1980s after the popularization of Kharitonov's theorem in the control engineering literature. In the following, a polynomial is said to be *stable* if all of its roots have negative real parts.[3]

In Kharitonov (1978), it is proved that every real polynomial in the interval family of polynomials

$$\{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \colon a_i^- \leq a_i \leq a_i^+, \; i = 0, \ldots, n\}$$

is stable if and only if four special members of the family with extreme coefficients (the *Kharitonov polynomials*) are stable.

The original proof of the theorem is in Russian. Simple geometrical proofs can be found in Dasgupta (1988) and Minnichelli, Anagnost and Desoer (1989). Using Kharitonov's theorem, it is easy to design a polynomial-time algorithm that decides whether a given interval family is stable: it suffices to apply stability tests to the four Kharitonov polynomials. The relevance of this result to robust stability analysis provided motivation for various attempts at generalizations but with limited success. This experience opened up the possibility that broad generalizations of Kharitonov's Theorem, of comparable simplicity, do not exist. Indeed, results from computational complexity show that, unless P = NP, a variety of other types of polynomial families do not admit stability criteria that can be tested efficiently. We review a number of such results.

[3] This definition of stability relates to continuous-time systems. We restrict the discussion in Sections 3.1–3.4 to this continuous-time setting.

Interval polynomial families are special cases of affine polynomial families. Let $p_i(x)$, $i = 1, \ldots, n$, be real polynomials. An *affine polynomial family* is a set of polynomials of the form

$$\{p_0(x) + a_1 p_1(x) + a_2 p_2(x) + \cdots + a_m p_m(x): \; a_i^- \leq a_i \leq a_i^+, \; i = 1, \ldots, m\}.$$

An *edge* of the family is a subset (i.e., a set of polynomials) for which all of the parameters a_i are fixed to their upper or lower bounds a_i^+ or a_i^-, with the exception of one parameter, which is allowed to vary within the corresponding interval. It is known (Bartlett, Hollot & Huang, 1988) that the stability of an affine polynomial family is equivalent to the stability of all of its edges. Furthermore, the stability of an edge can be checked in polynomial time by using an edge stability condition given in Bialas (1985). There are $m2^{m-1}$ edges in an affine polynomial family with m parameters. The elementary stability checking procedure that consists in checking all edges leads thus to a number of operations that is exponential in the number of parameters. This procedure can be improved by identifying critical edges. It is shown in Sideris (1991) that the stability of an affine polynomial family of degree n, with m parameters, can be checked in polynomial time, with $O(m^3 n^2)$ arithmetic operations.

Broader classes of polynomial families can be constructed as follows. A *multilinear polynomial family* (or, more precisely, a *multiaffine* polynomial family) is a set of polynomials of the form

$$\{b_0(a_0, \ldots, a_m) + b_1(a_0, \ldots, a_m)x + b_2(a_0, \ldots, a_m)x^2 + \cdots + b_n(a_0, \ldots, a_m)x^n : a_i^- \leq a_i \leq a_i^+, \; i = 0, \ldots m\},$$

where each coefficient $b_i(a_0, \ldots, a_n)$ is an affine function of each parameter a_i when the remaining parameters are held fixed. This structure arises, for example, when one considers the characteristic polynomial of an interval family of matrices (see the next section). Several contributions have dealt with the stability of such families. In contrast to the affine case, however, no polynomial-time stability checking algorithms are available. Indeed, using Proposition 2.1 in Nemirovskii (1993), it is easily concluded that, unless P = NP, no such algorithm exists. We present here an elementary proof of an improved version of this result. We define a *bilinear polynomial family* as a multilinear polynomial family in which each coefficient function depends only on two parameters. We show that, unless P = NP, the stability of bilinear polynomial families of *degree one* is not decidable in polynomial time. We provide a complete proof because of its simplicity and because it adequately illustrates the usual way that such negative results are proved. The proof is based on a reduction of the MAXIMUM 2-SATISFIABILITY problem,

which is known to be NP-complete (Garey & Johnson, 1979).

MAXIMUM 2-SATISFIABILITY

Instance: A set $U = \{u_1, \ldots, u_m\}$ of Boolean variables, a collection C of k clauses over U, such that there are at most two literals in each clause, and a positive integer $K \leq k$.[4]

Question: Is there a truth assignment for the variables in U that simultaneously satisfies at least K of the clauses in C?

Theorem. *The stability of bilinear polynomial families of degree one is* NP*-hard to decide.*

Proof. Given an arbitrary instance of MAXIMUM 2-SATISFIABILITY, we will construct an equivalent instance of the stability problem. Any clause $c \in C$ is of one of the following forms: $c = u_i$, $c = \bar{u}_i$, $c = u_i \vee u_j$, $c = u_i \vee \bar{u}_j$, or $c = \bar{u}_i \vee \bar{u}_j$, for some i,j. To $c = u_i$ we associate the affine function a_i, to $c = \bar{u}_i$ we associate $1 - a_i$, to $c = u_i \vee u_j$ we associate $a_i + a_j - a_i a_j$, to $c = u_i \vee \bar{u}_j$ we associate $1 - a_j + a_i a_j$, and to $c = \bar{u}_i \vee \bar{u}_j$ we associate $1 - a_i a_j$.

Let $A(a_1, \ldots, a_n)$ be the sum of the polynomials associated with the clauses in C and consider the bilinear polynomial family

$$\{x + (K - A(a_1, \ldots, a_n)) \colon 0 \leq a_i \leq 1\}.$$

All polynomials in the family are stable if and only if the root $A(a_1, \ldots, a_n) - K$ is negative for all choices of the a_i in [0,1]. That is, if and only if the maximum of $A(a_1, \ldots, a_n)$ is less than K. Note that the maximum of $A(a_1, \ldots, a_n)$ for $a_i \in [0,1]$ is attained for $a_i \in \{0,1\}$, and is therefore equal to the maximum (over all possible truth assignments) number of satisfied clauses. Thus, we have a stable family if and only if we have a NO instance of MAXIMUM 2-SATISFIABILITY. We therefore have a polynomial time reduction of an NP-complete problem to the problem of interest, and the result follows. □

We have seen that the stability of affine polynomial families can be checked in polynomial time, whereas the stability of bilinear polynomial families is NP-hard. Thus, the tractability borderline is somewhere between affine and bilinear families. Unless P = NP, efficient stability checking algorithms can only be derived for *particular* multilinear polynomial families, see, e.g., Fu, Dasgupta and Blondel (1995).

The problem of deciding whether a given family of polynomials is stable is equivalent to the problem of deciding whether the family contains an *unstable* polynomial. Both problems are *analysis* problems. A related problem is that of asking whether a family contains a *stable* polynomial, which is the same as asking whether all polynomials in the family are unstable. This is a *design* problem and, although its complexity is related to the complexity of the analysis problem, the relation is not as immediate as it may first seem.

Kharitonov's theorem does not hold for instability. In general, it is not true that instability of all four Kharitonov polynomials of an interval family of polynomials ensures instability of the entire family. Similarly, it is not true in general that instability of all edges of an affine polynomial family ensures instability of the entire family. Existence of a polynomial time algorithm for checking the presence of a stable polynomial in an affine polynomial family is an open problem. This problem is of interest because it is intimately related to basic control design problems. On the other hand, the proof given above can be easily adapted to prove that the problem of deciding whether there is a stable polynomial in a given bilinear family of polynomials is NP-hard.

3.2. Stability of matrix families

If A_- and A_+ are square matrices of the same dimensions, we define the interval matrix family $[A_-, A_+]$ as the set of all matrices A that satisfy the inequality $A_- \leq A \leq A_+$ componentwise, i.e., $(A_-)_{ij} \leq A_{ij} \leq (A_+)_{ij}$ for every i and j.

For interval matrix families, there are no results analogous to Kharitonov's theorem. In fact, the stability of all edges of an interval matrix family does not necessarily imply the stability of the entire family. There are however some results, for the symmetric case, in terms of the extreme points of the family. An interval matrix family $[A_-, A_+]$ is said to be *symmetric* if the matrices A_- and A_+ are symmetric. (Note, however, that in general, a symmetric interval matrix family $[A_-, A_+]$ also contains nonsymmetric matrices.) Necessary and sufficient conditions for stability of symmetric interval matrices, formulated in terms of the stability of a finite subset of matrices in the family, are given in Soh (1990), Hertz (1992), and Rohn (1994a). Still, in all of these references, the number of matrices that have to be tested is exponential in the dimension of the matrices. This seems to be unavoidable given the negative results that we describe next.

The computational complexity analysis of problems associated with interval matrix families was initiated in Poljak and Rohn (1993) and Nemirovskii (1993).[5] A com-

[4] A literal is either a Boolean variable u_i or its negation $\bar{u}_i$. A one-literal clause consists of a single literal. A two-literal clause is a disjunction $(x \vee y)$ of two literals x and y.

[5] The results presented in Poljak and Rohn (1993) and Nemirovskii (1993) were found independently. Poljak and Rohn (1993) was submitted about two years before Nemirovskii (1993), but for editorial reasons, the two papers appeared in the same issue of *Mathematics of Control, Signals, and Systems*.

bination of the results presented in these two papers is as follows. An elegant and economical presentation is provided in Rohn (1994b).

For interval matrix families, the following four problems are NP-hard:

(i) decide whether all matrices in a given family are *nonsingular* (INTERVAL MATRIX NONSINGULARITY);
(ii) decide whether all matrices in a given family are *stable* (INTERVAL MATRIX STABILITY);
(iii) decide whether all matrices in a given family have *spectral norm* (maximum singular value) less than one;
(iv) decide whether all symmetric matrices in a given family are *positive definite*.

The NP-hardness proof of (i) given by Poljak and Rohn involves the MAX CUT problem, which is defined as follows. We are given an undirected graph, and we wish to partition the set of nodes into two disjoint subsets, in a way that maximizes the number of edges with one endpoint in each subset. The decision version of this problem ("Is the maximum larger than a given number?") is NP-complete (Papadimitriou, 1994; Poljak & Rohn, 1993) shows that MAX CUT can be reduced to INTERVAL MATRIX NONSINGULARITY. Furthermore, INTERVAL MATRIX NONSINGULARITY remains NP-hard even if the size (bit length) of the entries of the $n \times n$ matrices A_- and A_+ is $O(\log n)$. Problems with such a property, that remain NP-hard even if the size of each number in the input is restricted to be small (of the order of a polynomial in the logarithm of the overall instance size), are said to be *strongly* NP-hard. The complexity of such problems is of an inherently combinatorial nature, rather than being due to excessively long numerical input.

We can rephrase the decision problem INTERVAL MATRIX NONSINGULARITY as an optimization problem, and look at the possibility of approximations. The *center* of the interval matrix family $[A_-, A_+]$ is $A_c = (A_+ + A_-)/2$ and its *error* is $A_e = (A_+ - A_-)/2$. The interval matrix family $[A_c - \lambda A_e, A_c + \lambda A_e]$ is equal to $\{A_c\}$ when $\lambda = 0$, and is equal to the interval family $[A_-, A_+]$ when $\lambda = 1$. The *radius of stability* $\lambda(A_c, A_e)$ associated with A_c and A_e is the smallest value of λ for which $[A_c - \lambda A_e, A_c + \lambda A_e]$ contains a singular matrix. Clearly, all matrices in $[A_-, A_+]$ are nonsingular iff $\lambda(A_c, A_e) > 1$ and, therefore, deciding whether $\lambda(A_c, A_e) > 1$ is also NP-hard. However, a stronger result that refers to the impossibility of approximating $\lambda(A_c, A_e)$ is also possible, as we discuss next.

It is known that unless P = NP, MAX CUT does not admit a polynomial-time approximation scheme as defined in Section 2.8 (Papadimitriou, 1994). It follows that the same is true for the optimization version of INTERVAL MATRIX NONSINGULARITY (Coxson & De Marco, 1994); see also Demmel (1992).

The NP-hardness of the other three problems in the above theorem was shown in Nemirovskii (1993), by using a reduction of the classical PARTITION problem,[6] which is NP-complete (Garey & Johnson, 1979). The reductions produce very particular classes of interval matrix families. Thus, restricted versions of the problems that encompass these classes are also NP-hard. For example, INTERVAL MATRIX STABILITY is NP-hard when all the entries of the matrices in the family are fixed, except for some of the entries of a single row and a single column, which are allowed to vary between -1 and 1.

Even though PARTITION is NP-complete, it can be solved in time polynomial in the number of integers and exponential in their size (bit length). (In fact, it is a special case of the 0–1 KNAPSACK problem discussed in Section 2.8.) Thus, the proof in Nemirovskii (1993) still leaves the possibility that problems such as INTERVAL MATRIX STABILITY can be solved in time polynomial in the dimensions of the matrices, when their entries have small bit length, e.g., logarithmic in the dimension of the matrices. However, this possibility is ruled out by adapting Nemirovskii's proof with the help of the reduction used in Poljak and Rohn (1993), and all of these problems are strongly NP-complete (Coxson, 1993).

An interval matrix family can always be expressed in the form

$$\{A_0 + a_1 A_1 + \cdots + a_m A_m : 0 \le a_i \le 1,\ 1 \le i \le m\},$$

where each matrix A_i has a single nonzero entry. A *rank-τ affine matrix family* is a family of the form above, where the matrices A_i have rank that is less than or equal to τ. The stability of rank-1 affine families is NP-hard to decide, and the special case $m = 1$ is solvable in polynomial time without any rank restrictions; see Fu and Barmish (1988). Besides these two special cases, the territory is unmarked. Exponential time branch-and-bound algorithms for the case of arbitrary m and τ are proposed in Kokame and Mori (1992). A solution in terms of finitely many linear programs is proposed in Barmish, Floudas, Hollot and Tempo (1994).

There are many more results on the complexity of *interval* linear algebra problems. A survey of some of these results is given in Kreinovich, Lakeyev, Rohn and Kahl (1995) and Rohn (1997). See also the recent book (Kreinovich et al., 1997).

Let us finally mention a different problem but with a somewhat similar flavor. Given a system with delays, of the form

$$\dot{x}(t) = A_0 x(t) + \sum_{k=1}^{p} A_k x(t - h_k),$$

we wish to determine whether the system is stable for all allowed values of the delays h_k. It turns out that this

[6] In the PARTITION problem, we are given integers $a_1, \ldots, a_n$, whose sum is some number K and we are asked whether there exists a subset of these integers whose sum is $K/2$.

problem is NP-hard for the case where each h_k is constrained to lie within an interval $[h_k^-, h_k^+]$, as well as for the unconstrained case, where $h_k \in [0, +\infty)$ (Toker & Ozbay, 1996).

3.3. Structured singular values

The results in the preceding subsection have direct implications for the computation of the *structured singular value* of a matrix, a fact that was hinted at in Poljak and Rohn (1993). The structured singular value generalizes the notion of singular value and arises in many robust control problems involving systems with uncertain parameters.

We start with a square, possibly complex, matrix M of dimensions $n \times n$ and we consider a set $\mathscr{D}$ of block-diagonal perturbation matrices Δ, of the form $\Delta = \mathrm{diag}\{\Delta_1, \ldots, \Delta_k\}$, where each Δ_i has dimensions $n_i \times n_i$ and $n_1 + \cdots + n_k = n$. Each diagonal block is constrained to be of a particular type, e.g., a real multiple of the identity matrix, a complex multiple of the identity matrix, or an arbitrary real (or complex) matrix. Having defined the set of matrices $\mathscr{D}$ in terms of such constraints, the corresponding *structured singular value* is defined by

$$\mu(M) = \begin{cases} 0 \\ \quad \text{if } \det(I - \Delta M) \neq 0 \text{ for all } \Delta \in \mathscr{D}, \\ (\min_{\Delta \in \mathscr{D}} \{\bar{\sigma}(\Delta) : \det(I - \Delta M) = 0\})^{-1} \\ \quad \text{otherwise,} \end{cases}$$

where $\bar{\sigma}(\Delta)$ stands for the largest singular value of Δ. When $\mathscr{D} = \{\delta I : \delta \in \mathbf{C}\}$, $\mu(M)$ is equal to the spectral radius of M, and when $\mathscr{D} = \mathbf{C}^{n \times n}$, $\mu(M)$ is equal to the maximum singular value of M. For an introduction to structured singular values and their importance in system analysis and design, see Doyle (1982), Safonov (1982) and Zhou, Doyle and Glover (1995).

Many researchers had worked on algorithms for computing μ exactly, under various uncertainty structures, until it was established that the problem of deciding whether $\mu(M) \geq 1$ is NP-hard for the following cases:

(i) If M is real and each block Δ_i is a real multiple of the identity (the "real μ" problem).
(ii) If M is complex and each block Δ_i is a multiple $\delta_i I$ of the identity, where some δ_i are constrained to be real and some are allowed to be complex (the "mixed μ" problem).
(iii) If M is complex and each block Δ_i is of the form $\delta_i I$ for some complex δ_i (the "purely complex μ" problem).

The result for the first two cases is a corollary of the results in Poljak and Rohn (1993). It is also derived in Braatz, Young, Doyle and Morari (1994), by showing that quadratic programming can be reduced to the problem of computing μ. The third result was shown later in Toker and Ozbay (1998) by establishing that a suitably defined quadratic programming problem involving complex variables is NP-hard, and then reducing it to the purely complex μ problem.

Partly as a consequence of such results, research efforts have shifted to the problem of finding approximations of μ which are easy to compute and that are as close as possible to the exact value of μ; see Fu (1999) and Toker and de Jager (1999) for some discussion on computational complexity problems related to the approximate computation of μ, and Trei (1999) for a negative result on an approach that had seemed promising.

Finally, let us mention that Fu and Dasgupta (1998) have studied the complexity of computing the real structured singular value with perturbations measured by a p-norm. For a set $\mathscr{D}$ of block diagonal matrices where each block Δ_i is a real multiple $\delta_i I$ of the identity, define

$$\mu_p(M) = \left(\min_{\Delta \in \mathscr{D}} \{ \|\Delta\|_p : \det(I - \Delta M) = 0 \} \right)^{-1},$$

where $\delta = (\delta_1, \ldots, \delta_k)$ and $\Delta = \mathrm{diag}\{\delta_1 I, \ldots, \delta_k I\}$. The quantity μ_p coincides with μ when $p = \infty$ and so the problem of deciding whether $\mu_\infty(M) \geq 1$ is NP-hard. Fu and Dasgupta show that the problem of deciding whether $\mu_p(M) \geq 1$ is NP-hard for any $p \in [1, \infty]$ when the size of the smallest block exceeds one. For blocks of size one and finite p, the question of NP-hardness is open.

3.4. Existence of a stable matrix in an interval family; static output feedback

We now turn our attention to design, rather than analysis, questions for interval matrix families, and consider the following problem.

STABLE MATRIX IN INTERVAL FAMILY
Instance: An interval family of matrices.
Question: Does the family contain a stable matrix?

This problem is NP-hard. It remains so even if the interval family is symmetric and all the entries of the family are fixed, except for some of the entries of a single row and a single column, which can take values between -1 and 1. The proof is given in Blondel and Tsitsiklis (1997) and involves again a reduction of the PARTITION problem. In contrast to the corresponding analysis question discussed in Section 3.2, it is not known whether STABLE MATRIX IN INTERVAL FAMILY is NP-hard in the strong sense. A variation of STABLE MATRIX IN INTERVAL FAMILY is the problem in which we only look at symmetric matrices. Consider the semidefinite programming problem of minimizing λ subject to the constraint that $\lambda I - A$ is nonnegative definite and subject to the interval constraints on the symmetric matrix A.[7] Stability is then

[7] This formulation has been suggested by M. Overton (personal communication).

equivalent to the optimal value of λ being negative. This semidefinite programming problem can be solved in polynomial time to any fixed desired accuracy. However, the complexity of an exact solution is unknown.

This problem bears some similarities to the widely studied, and still unsolved, *static output feedback problem*. We are given matrices A, B, and C, and we are interested in determining whether there exists a "gain" matrix K such that $A + BKC$ is stable. Despite numerous contributions since the early 1960s, a satisfactory answer to this problem has yet to be found. The problem is often cited as one of the difficult open problems in systems and control (Blondel, Gevers & Lindquist, 1995). See Syrmos, Abdallah and Dorato (1994) for a survey, and Rosenthal and Wang (1997) for genericity results on the related pole assignment problem. From a complexity point of view, the problem is algorithmically decidable using Tarski elimination, as pointed out in Anderson, Bose and Jury (1975).

Still, despite various attempts, it is unclear whether the problem is NP-hard. In Fu and Luo (1997), the authors show that a certain type of mixed LMI can be used to solve the static output feedback problem, but they then show that the proposed mixed LMI is NP-hard.

Using the NP-hardness result for STABLE MATRIX IN INTERVAL FAMILY, it is easily shown that the static output feedback problem is NP-hard if we also require the gain matrix K to satisfy interval constraints. However, the relevance of this observation is unclear, because the problem with interval constraints remains NP-hard for the special case of *state* feedback (given A and B, determine whether there exists some K within a given interval family such that $A + BK$ is stable), even though it can be solved in polynomial time in the absence of interval constraints (Blondel & Tsitsiklis, 1997a). Let us also mention some related problems that have been shown to be NP-hard in the same reference. These are the problems of simultaneous stabilization by output feedback (see also Toker and Ozbay, 1995), decentralized output feedback stabilization using a norm bounded controller, and decentralized stabilization using identical controllers.

3.5. Time-varying systems

We have been discussing so far problems related to uncertain systems but with time-invariant uncertain parameters. We now turn to a discussion of problems relevant to systems with time-varying uncertainty, and we will be focusing on discrete-time systems.

We consider a time-varying system of the form $x_{t+1} = A_t x_t$, where Σ is a finite set of matrices, and $A_t \in \Sigma$ for every $t \geq 0$. We do not impose any restrictions on the sequence of matrices A_t. Starting from the initial state x_0, we obtain

$$x_{t+1} = A_t \cdots A_1 A_0 x_0.$$

We wish to characterize how x_t grows with t. For this purpose, let $\|\cdot\|$ be an induced matrix norm. The *joint spectral radius* $\bar{\rho}(\Sigma)$ is defined in Rota and Strang (1960) by

$$\bar{\rho}(\Sigma) = \lim_{k \to \infty} \bar{\rho}_k(\Sigma),$$

where

$$\bar{\rho}_k(\Sigma) = \max\{\|A_k \cdots A_2 A_1\|^{1/k} : \text{each } A_i \in \Sigma\}$$

for $k \geq 1$. (See also Berger and Wang (1992) for another characterization of $\bar{\rho}(\Sigma)$ in terms of the spectral radius of the matrix products.) It turns out that the limit always exists and the value of $\bar{\rho}(\Sigma)$ is the same for every induced matrix norm. The same comment applies to the definitions of $\underline{\rho}(\Sigma)$ and $\rho_P(\Sigma)$ given later in this section.

For a given finite set of matrices Σ, we can now pose questions such as the following:

- Do we have $\bar{\rho}(\Sigma) < 1$?
- Are all possible matrix products $A_k \cdots A_1$ stable?
- Do we have $\bar{\rho}(\Sigma) \leq 1$?
- Do all infinite matrix products converge to zero?
- Do all periodic matrix products converge to zero?
- Do we have $x_t \to 0$ for every x_0 and any possible sequence of matrices?
- Do we have $x_t \to 0$ for every x_0 and any possible periodic sequence of matrices?
- Is the set of all possible matrix products $A_k \cdots A_1$ bounded?

The above questions are not equivalent, but they are closely related. Due to their relevance in many practical settings, they have been extensively studied in recent years. For example, certain inequalities proved in Daubechies and Lagarias (1992, Lemma 3.1) can be used to derive algorithms that compute arbitrarily precise approximations for $\bar{\rho}(\Sigma)$; see Gripenberg (1996) for one such algorithm. These approximation algorithms can then be used in procedures that decide, after finitely many steps, whether $\bar{\rho} > 1$ or $\bar{\rho} < 1$. Related procedures are given in Brayton and Tong (1980) and Barabanov (1988). The drawback of all of these procedures is that they need not terminate when $\bar{\rho}$ happens to be equal to 1. In fact, the ability to compute arbitrarily accurate approximations of $\bar{\rho}$ does not rule out the possibility that the problem of deciding whether $\bar{\rho} < 1$ is undecidable, and it is so far unknown whether this is the case or not; see Lagarias and Wang (1995) for a discussion of this issue and for a description of its connection with the so-called finiteness conjecture,[8] as well as the discussions in Gurvits (1995) and the NP-hardness result in Toker

[8] This conjecture can be restated as saying that the convergence to zero of all periodic products of a given finite set of matrices, implies the same for all possible products.

(1997). It turns out however that the slightly different problem of deciding whether $\bar{\rho} \leq 1$ (as well as the question of boundedness of the set of all matrix products) is undecidable (Blondel and Tsitsiklis, 1999b). Both problems remain undecidable in the case where Σ contains only two matrices. Another negative result is given in Kozyakin (1990), which establishes that several questions of the type introduced in this section cannot be answered using a restricted class of algebraic algorithms, because the set of all YES instances has infinitely many connected components, even for pairs of 2×2 matrices.

As a practical matter, efficient approximation algorithms for computing $\bar{\rho}(\Sigma)$ might suffice. However, the results of Tsitsiklis and Blondel (1997a, Theorem 1) place some limitations on possible approximation algorithms. In particular, it is shown that unless P = NP, there is no algorithm that can compute $\bar{\rho}$ with ε percent relative accuracy and which runs in time polynomial in $\log(1/\varepsilon)$ and the size of Σ.[9] Furthermore, this negative result continues to hold even if we restrict the set Σ to consist of only two matrices with 0–1 entries. Despite this negative result it is still conceivable that for any *fixed* $\varepsilon > 0$, there exists a polynomial-time algorithm that computes $\bar{\rho}$ with relative accuracy ε, but it is not known whether this is the case or not.

If a polynomial-time algorithm were available for checking the stability of all products of two given matrices, then the algorithm could be used in conjunction with binary search to approximate the joint spectral radius in polynomial time. As a consequence, it is NP-hard to decide whether all products of two given matrices (with rational entries) are stable. This is true even if all nonzero entries of the two matrices are constrained to be equal.[10] The decidability of this problem is not known. On the other hand, polynomial-time stability checking algorithms are available when all nonzero entries are equal to one; see Gurvits (1996).

Similar to the joint spectral radius, we can define the *lower spectral radius* $\underline{\rho}(\Sigma)$ by

$$\underline{\rho}(\Sigma) = \lim_{k \to \infty} \underline{\rho}_k(\Sigma),$$

where

$$\underline{\rho}_k(\Sigma) = \min\{\|A_1 A_2 \cdots A_k\|^{1/k} \text{: each } A_i \in \Sigma\}$$

for $k \geq 1$.

The joint spectral radius measures the maximal average norm of long products of matrices taken from a finite set, whereas the lower spectral radius measures the minimal average norm. The problem of computing $\underline{\rho}(\Sigma)$ is intimately related to the mortality problem. We say that the set of matrices Σ is *mortal* if the zero matrix can be expressed as the product of finitely many matrices from Σ. Reference Paterson (1970) establishes a relation between word concatenation and matrix multiplication, and reduces Post's correspondence problem to the problem of deciding whether Σ is mortal. In particular, it shows that the mortality problem is undecidable, even if Σ consists of only $2n_p + 2$ integer matrices of dimensions 3×3, as long as Post's correspondence problem with n_p pairs of words is undecidable. (As discussed in Section 2.3, we can take $n_p = 7$.) Building on this result (Blondel & Tsitsiklis, 1997b) establishes that the mortality problem remains undecidable even if Σ consists of only two integer matrices of dimensions $(6n_p + 6) \times (6n_p + 6)$. The number of 3×3 integer matrices involved in Paterson's proof can be reduced to $n_p + 2$ by considering the modified Post correspondence problem in which the initial pair of words used in the correspondence is predetermined and is part of the instance description (Hopcroft & Ullman, 1969); see Bournez and Branicky (1998) for a proof. An easy argument shows that, if mortality of d matrices of size $n \times n$ is undecidable, then the same applies to *pairs* of matrices of size $nd \times nd$ (Blondel & Tsitsiklis, 1997b). Thus, mortality of two integer matrices of size $3(n_p + 2)$ is undecidable. Moreover, all 3×3 matrices involved are diagonalizable, and so, by adapting the argument in Blondel and Tsitsiklis (1997b), one can conclude that the mortality problem for pairs of $3(n_p + 2)$ integer matrices, one of which is diagonal, is also undecidable. For the special cases of matrices with nonnegative entries, or when we restrict to products of the form $A_1 \cdots A_k$ for fixed k, the mortality problem can be solved (in exponential time), but is actually NP-hard (Blondel & Tsitsiklis, 1997b). See also Bournez and Branicky (1998) for a discussion of the mortality problem for 2×2 matrices, and Krom (1981), Krom and Krom (1989) and Klarner, Birget and Satterfield (1991) for more results on problems of similar flavor.

As long as the matrices have integer entries, we have either $\underline{\rho}(\Sigma) = 0$ (which is the case if and only if Σ is mortal) or $\underline{\rho}(\Sigma) \geq 1$. Thus the undecidability results for the mortality problem preclude the possibility of an exact, or even approximate computation of $\underline{\rho}(\Sigma)$ (Blondel & Tsitsiklis, 1997b). By the same argument, the problem of determining whether some product of two given matrices A_0 and A_1 is stable is also undecidable. Building on the comments made in the previous paragraph, it can be shown that this is true even if the two matrices have integer entries, are of size 27×27, and one of them is diagonal. On the other hand, if it is a priori known that $\bar{\rho}(\Sigma) = 1$, the question $\underline{\rho}(\Sigma) < 1$ can be decided in polynomial time (Gurvits, 1996).

While $\bar{\rho}(\Sigma)$ and $\underline{\rho}(\Sigma)$ refer to two extreme ways of choosing a sequence of matrices, we can define an

[9] In fact, a careful examination of the proof in that reference shows that a stronger result is true, ruling out algorithms that run in time which is polynomial in $1/\varepsilon$.

[10] This result is stated incorrectly in Tsitsiklis and Blondel (1997a); a correction appears in Tsitsiklis and Blondel (1997b).

intermediate quantity if we let the matrices A_t be generated randomly and independently, according to a probability distribution P on the set Σ. The *largest Lyapunov exponent* associated with P and Σ is then defined by

$$\lambda(\Sigma, P) = \lim_{k\to\infty} \frac{1}{k} E[\log(\|A_1 \cdots A_k\|)],$$

where $E[\cdot]$ stands for expectation. We define the *Lyapunov spectral radius* $\rho_P(\Sigma)$ by

$$\rho_P(\Sigma) = e^{\lambda(\Sigma, P)}$$

and it is easily verified that

$$\underline{\rho}(\Sigma) \le \rho_P(\Sigma) \le \bar{\rho}(\Sigma).$$

Similar to our earlier arguments, the undecidability of mortality can be used to establish that $\rho_P(\Sigma)$ cannot be computed exactly or approximately (Blondel & Tsitsiklis, 1997b).

It is interesting to contrast the latter result with what is known for analogous quantities defined in the max-plus algebra, where the usual operations of addition and multiplication are replaced by taking the maximum of two numbers, and addition, respectively (Baccelli, Cohen, Olsder & Quadrat, 1992). Products of matrices in the max-plus algebra correspond to performance measures for discrete-event systems, and the Lyapunov spectral radius is particularly interesting. In this context, the joint spectral radius can be computed in polynomial time, but the question whether $\rho < 1$ is NP-hard for both the lower spectral radius and for a quantity analogous to the Lyapunov spectral radius (Blondel, Gaubert & Tsitsiklis, 1998). In terms of computability, the latter quantity can be approximated (in exponential time) to any degree of accuracy (Baccelli et al., 1992), but it is unknown whether the question "is $\rho < 1$?" is decidable.

4. Nonlinear and hybrid systems

4.1. Introduction

Control problems are in general easier to solve for linear systems than for nonlinear ones. Even for systems that seem mildly nonlinear, simply stated control questions may become intractable or, in many cases, undecidable. We review in this section some of the available complexity results, including related results for hybrid systems. We look at four particular control problems.

STATE CAN BE DRIVEN TO THE ORIGIN

Input: A system $x_{t+1} = f(x_t, u_t)$ and an initial state x_0.

Question: Does there exist some $T \ge 1$ and controls u_t, $t = 0, \ldots, T-1$, such that $x_T = 0$?

NULLCONTROLLABILITY (All states can be driven to the origin.)

Input: A system $x_{t+1} = f(x_t, u_t)$.

Question: Is it true that for every initial state x_0, there exists some $T \ge 1$ and controls u_t, $t = 0, \ldots, T-1$, such that $x_T = 0$?

TRAJECTORY GOES TO THE ORIGIN

Input: A system $x_{t+1} = f(x_t)$ and an initial state x_0.

Question: Does there exist some $T \ge 1$ such that $x_T = 0$?

ALL TRAJECTORIES GO TO THE ORIGIN

Input: A system $x_{t+1} = f(x_t)$.

Question: Is it true that for every x_0 there exists some $T \ge 1$ such that $x_T = 0$?

Asymptotic versions of these definitions are possible by requiring the sequences to converge to the given state rather than reaching it in finite time. For example, in a problem ALL TRAJECTORIES CONVERGE TO THE ORIGIN, we have an autonomous system and the objective is to decide whether $x_t \to 0$ for any initial state x_0. The first two problems are control design problems, whereas the last two are analysis problems. When algorithms exist for the design problems, they can be used for the corresponding analysis problem as well. Thus, analysis problems are in general easier to solve. When looking for negative complexity-theoretic results, the reverse is true; analysis problems are harder to show undecidable or NP-hard than control problems.

In all of these problems, we need to assume that the system is "given" in terms of a finite-length description of the function f. For example, if f is polynomial with rational coefficients, it is "given" in terms of its coefficients. For linear systems the above questions are all decidable in polynomial time. On the other hand, no algorithms exist for general nonlinear systems. Stated at this level of generality, these problems are not interesting, because they are far too difficult. For example, as pointed in Sontag (1995), the null-controllability question for general nonlinear systems includes the problem of deciding whether a given arbitrary nonlinear equation $\phi(u) = 0$ has a solution. Indeed, for a given function ϕ, consider the system $x_{t+1} = \phi(u_t)$, which is null controllable if and only if ϕ has a zero. Thus, the null-controllability question for nonlinear systems is at least as hard as deciding whether a given nonlinear function has a zero, which is far too general a problem. For a problem to be interesting, we need to constrain the type of nonlinear systems considered. In the subsections that follow we consider several classes of systems, such as systems with a single nonlinearity, systems with componentwise nonlinearities, piecewise-linear systems, and hybrid systems.

4.2. Systems with a single nonlinearity

Even for systems that involve a *single* scalar nonlinearity, algorithms for deciding stability are inherently

inefficient. To illustrate this, let us fix an arbitrary nonconstant scalar function $v: \mathbf{R} \mapsto \mathbf{R}$ that satisfies

$$\lim_{x \to -\infty} v(x) \leq v(x) \leq \lim_{x \to +\infty} v(x), \quad \forall\ x \in \mathbf{R}. \tag{2}$$

Having fixed v, let us consider the problem in which we are given $A_0, A_1 \in \mathbf{Q}^{n \times n}$, and $c \in \mathbf{Q}^n$ as input, and we wish to determine whether all trajectories of the system

$$x_{t+1} = (A_0 + v(c^{\mathrm{T}} x_t) A_1) x_t \tag{3}$$

converge to the origin. If v were constant, system (3) would be linear and this question could be decided in polynomial time. On the other hand, the problem is known to be NP-hard for any fixed nonconstant v that satisfies (2) (Blondel & Tsitsiklis, 1999a).

If we focus on the particular case where v is the sign function, we obtain an NP-hardness result for a very simple class of nonlinear systems, namely systems that are linear on each side of a hyperplane:

$$x_{t+1} = \begin{cases} A_+ x_t & \text{when } c^{\mathrm{T}} x_t \geq 0, \\ A_- x_t & \text{when } c^{\mathrm{T}} x_t < 0. \end{cases} \tag{4}$$

Such systems could arise when a linear system is controlled using a "bang–bang" controller. They also represent one of the simplest types of hybrid systems.

It is not clear whether questions related to the stability of systems (3) are actually *decidable*. We do not know of any nonconstant function v for which ALL TRAJECTORIES CONVERGE TO THE ORIGIN has been proved decidable or undecidable.

System (4) is of the form $x_{t+1} = A_t x_t$ where each A_t belongs to the set $\{A_-, A_+\}$. This is a situation similar to the one considered in Section 3.5, except that now the sequence of matrices is not arbitrary, but is determined by the state sequence. Still, this does not help in reducing complexity as we now argue using an example taken from Blondel and Tsitsiklis (1998a). Consider a system described by a state vector (v_t, y_t, z_t), where v_t and y_t are scalars and z_t is a vector in $\mathbf{R}^n$, and the dynamics is of the form

$$\begin{pmatrix} v_{t+1} \\ y_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} -1/4 & 0 & 0 \\ -1/4 & 1/2 & 0 \\ 0 & 0 & A_+ \end{pmatrix} \begin{pmatrix} v_t \\ y_t \\ z_t \end{pmatrix} \quad \text{when } y_t \geq 0,$$

and

$$\begin{pmatrix} v_{t+1} \\ y_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} 1/4 & 0 & 0 \\ 1/4 & 1/2 & 0 \\ 0 & 0 & A_- \end{pmatrix} \begin{pmatrix} v_t \\ y_t \\ z_t \end{pmatrix} \quad \text{when } y_t < 0.$$

This system consists of two linear systems, each of which is enabled in one of two half-spaces, as determined by the sign of y_t. Given that y_0 can be any real number, it can be verified that the sequence $\mathrm{sign}(y_t)$ is completely arbitrary, which then implies that the matrices A_- and A_+ can be multiplied in an arbitrary order. Clearly, all trajectories converge to zero if and only if z_t always converges to zero and the problem is at least as hard the corresponding question from Section 3.5 in which arbitrary products were allowed. If the latter problem is ever shown to be undecidable, the same will be automatically true for the problem considered here. On the other hand, since the boundedness of all products of A_- and A_+ is undecidable (Blondel & Tsitsiklis, 1999b), it follows that the uniform boundedness of the trajectories of system (4), for all initial states in a bounded set, is also undecidable.

One can easily adapt (3) to include the possibility of a control action, as in a system of the form

$$x_{t+1} = (A_0 + v(c^{\mathrm{T}} x_t) A_1) x_t + B u_t. \tag{5}$$

The question of NULLCONTROLLABILITY for system (5), subsumes the question of whether ALL TRAJECTORIES GO TO THE ORIGIN the autonomous system (3) — just consider the special case where $B = 0$ — and NP-hardness follows. One can in fact say more than that. When v is the sign function, then NULLCONTROLLABILITY and STATE CAN BE DRIVEN TO THE ORIGIN for systems (5) becomes undecidable (Blondel & Tsitsiklis, 1999a, Theorem 2).

Let us finally note that for functions v that have finite range, systems (5) become piecewise linear. We shall say more about the complexity of such systems in Section 4.4.

4.3. Systems with componentwise nonlinearities

Let us fix a scalar function $\sigma: \mathbf{R} \mapsto \mathbf{R}$, and consider systems of the form

$$x_{t+1} = \sigma(A x_t), \tag{6}$$

where the matrix A has size $n \times n$ and σ is applied componentwise, i.e.,

$$\sigma(q_1, \ldots, q_n) = (\sigma(q_1), \ldots, \sigma(q_n)).$$

Systems of this type arise in a wide variety of situations. When σ is linear, we have a linear system and most dynamical properties of interest can be checked efficiently. When σ has finite range, the entries of the state vector take values in a finite set, the system evolves on a finite state space after the first time step, and dynamical properties can be decided in time which is polynomial in the number of possible states. Note, however, that the number of states generally increases exponentially in n. If σ is the sign function and A is symmetric, system (6) is guaranteed to have a fixed point, but it is not known whether it can be found in polynomial time (Papadimitriou, 1994).

Recurrent artificial neural networks are commonly of the form (6) (Sontag, 1996b) and, in that context, σ is referred to as the *activation function*. Some of the activation functions that are common in the neural network literature are the logistic function $\sigma(x) = 1/(1 + \mathrm{e}^{-x})$, the trigonometric function $\sigma(x) = \arctan(x)$, and the

saturated linear function

$$\sigma(x) = \begin{cases} 0 & \text{when } x \le 0, \\ x & \text{when } 0 < x < 1, \\ 1 & \text{when } x \ge 1. \end{cases}$$

Note that all of these functions are continuous and have finite limits on both ends of the real axis, which is a common assumption in the context of artificial neural networks. Other motivations for studying systems of type (6) arise in the context of fixed-point digital filters and of linear systems with saturation limits on the state or control variables. See the book Liu and Michel (1994) for further motivation and many references related to neural networks and filter design. Finally, we will also discuss the *cut function*, defined by

$$\sigma(x) = \begin{cases} 0 & \text{when } x \le 0, \\ x & \text{when } x > 0, \end{cases}$$

which is probably the simplest nonlinear but piecewise linear function.

It turns out that systems of the form (6) have vastly different characteristics from linear systems. A result announced in Siegelmann and Sontag (1991) and completed in Siegelmann and Sontag (1995), shows that saturated linear systems are capable of simulating Turing machines. This is done by suitably encoding the transition rules of the Turing machine in the matrix A, while the tape contents and the machine's internal state are encoded on some of the states of the saturated linear system. Thus, as computational devices, linear saturated systems are as powerful as Turing machines. This abstract result has several system-theoretic implications. We describe a representative one. Recall that the halting problem is undecidable for Turing machines. Therefore, the problem of deciding whether a given initial state of a saturated linear system eventually leads to a state that encodes a halting configuration, is also undecidable. This halting state can be chosen to be the origin (see Sontag, 1995) and it follows that TRAJECTORY GOES TO THE ORIGIN is undecidable for saturated linear systems.

By using a universal Turing machine,[11] one can in fact prove that for saturated linear systems, TRAJECTORY GOES TO THE ORIGIN is undecidable for some *particular* matrix A. More concretely, there exists a particular integer matrix A (of size approximately equal to 1000×1000) for which there exists no algorithm that takes an initial state x_0 (with rational components) as input and decides whether the state will eventually hit the origin (Siegelmann & Sontag, 1995). The problem of finding the smallest possible such matrix is a difficult open problem that is related to the smallest possible size of universal Turing machines (Rogozhin, 1996).

The initial result by Siegelmann and Sontag has prompted efforts to understand the necessary properties of a function σ, under which Turing machine simulation is possible. The fact that such simulations are possible with the cut function is proved in an elementary and simple way in Hyotyniemu (1997).[12] In Koiran (1996), it is shown that Turing machines can be simulated if σ eventually becomes constant on both ends of the real line and is twice differentiable with nonzero derivative on some open interval. Note, however, that the function $\sigma = \arctan$ and other standard functions in the neural network literature do not satisfy these hypothesis. The conditions are further relaxed in Kilian and Siegelmann (1996). The authors offer a sketch of a proof that Turing machines can be simulated for any function σ in a certain class that includes, among others, the functions described above and all the functions that are classically used in artificial neural networks. The functions do not need to become ultimately constant but need to be monotone.

Using an argument similar to the one provided earlier for saturated linear systems, we arrive at the conclusion that TRAJECTORY GOES TO THE ORIGIN for systems (6) is undecidable when σ is the saturated linear function, the cut function, the logistic function, or any function that belongs to the classes defined in Koiran (1996) and Kilian and Siegelmann (1996). The undecidability of TRAJECTORY GOES TO THE ORIGIN immediately implies the same for STATE CAN BE DRIVEN TO THE ORIGIN for systems of the form

$$x_{t+1} = \sigma(Ax_t + Bu_t). \tag{7}$$

These results do not have any direct implications for the problem ALL TRAJECTORIES CONVERGE TO THE ORIGIN. For saturated linear systems, this problem was conjectured to be undecidable by Sontag in Sontag (1995). A proof that this is indeed the case is given in Blondel, Bournez, Koiran and Tsitsiklis (1999b) and establishes the undecidability of several related global properties such as ALL TRAJECTORIES GO TO THE ORIGIN. The main idea is to simulate a Turing machine by a saturated linear system so that all trajectories converge to the origin if and only if the Turing machine halts for *every* initial configuration. By a result of Hooper (Hooper, 1966), the latter Turing machine problem is undecidable, hence the proof of Sontag's conjecture.

4.4. Piecewise linear and hybrid systems

Hybrid systems involve a combination of continuous dynamics (e.g., differential or difference equations) and

[11] A universal Turing machine is a Turing machine M that receives as input the description of another machine M' and a string s, and then simulates the operation of machine M' on input s.

[12] Notice that the title of the reference Hyotyniemu (1997) involves the term "stability", but it is actually used in a sense different than the usual one in systems theory.

discrete dynamics. There is no consensus on a single general model of hybrid systems, but many discrete-time models include the class that we describe next.

We start with a partition of $\mathbf{R}^n$ into finitely many disjoint subsets $H_1, H_2, \ldots, H_m$. We assume that a different linear system is associated with each subset, and the overall system is described by

$$x_{t+1} = A_i x_t \quad \text{when } x_t \in H_i. \tag{8}$$

When the subsets H_i are defined in terms of a finite number of linear inequalities, the systems of the form (8) are the *piecewise linear* systems introduced by Sontag (1981) as a unifying model for describing interconnections between automata and linear systems. See Sontag (1985) for a characterization of the complexity of questions related to fixed time horizon properties of such systems, and Sontag (1996a) for a more recent account of available results.

Note that systems with componentwise nonlinearities, and with σ equal to the saturated linear function, are a special case of piecewise linear systems. Hence, using the results of the previous section, we deduce that piecewise linear systems can simulate Turing machines and TRAJECTORY GOES TO THE ORIGIN is undecidable for piecewise linear systems. It is known that simulation of Turing machines is possible by using piecewise *affine* systems in state dimension two (Moore, 1990), but not in dimension one (Koiran, Cosnard & Garzon, 1994). This result can be interpreted and rearranged by saying that Turing machine simulation is possible by piecewise linear systems in dimension three and, hence, TRAJECTORY GOES TO THE ORIGIN is undecidable for such systems. Similar to saturated linear systems, one can use a universal Turing machine to prove that there exists a *particular* piecewise linear system in $\mathbf{R}^3$ (with the state space being partitioned into fewer than 800 subsets) for which TRAJECTORY GOES TO THE ORIGIN is undecidable (Koiran et al., 1994).

Let us now introduce a control variable and consider systems of the form

$$x_{t+1} = A_i x_t + B u_t \quad \text{when } x_t \in H_i. \tag{9}$$

When $B = 0$, these systems are equivalent to autonomous piecewise linear systems and so STATE CAN BE DRIVEN TO THE ORIGIN is undecidable. This result is also obtained in Blondel and Tsitsiklis (1999a), with a different proof technique that has the advantage that it can be used to derive sharper estimates of the decidability limit (see also Toker, 1996 for a similar proof). There is an obvious tradeoff in piecewise linear systems between the state-space dimension n and the number of subsets m. When there is only one subset, or when the state dimension is equal to one, most properties are easy to check. In Blondel and Tsitsiklis (1999a, Theorem 2), it is shown that NULLCONTROLLABILITY is undecidable for piecewise linear systems of dimension $6n_p + 7$ on two regions, and that STATE CAN BE DRIVEN TO THE ORIGIN is undecidable for piecewise linear systems of dimension $3n_p + 1$ on two regions. Here, n_p is a number of pairs for which Post's correspondence problem is undecidable and, as explained in Section 2, we can take $n_p = 7$. This result can be improved by showing that, when $nm \geq 2 + 6n_p$, then the problem of deciding whether a given initial state x_0 can be driven to the origin, is undecidable. Thus, the problem of interest is undecidable when $nm \geq 44$. In particular, the problem remains undecidable for piecewise linear systems of state dimension 22 and with as few as two subsets.

Finally, the problem ALL TRAJECTORIES GO TO THE ORIGIN, as well as a number of other properties related to stability, are *undecidable* for piecewise *affine* systems in dimension two (Blondel, Bournez, Koiran, Papadimitriou & Tsitsiklis, 1999a). The proof makes use of two-counter machines, a model of computation equivalent to Turing machines, as far as decidability is concerned, and relies on the fact that the problem of determining whether a given two-counter machine halts for *every* initial configuration of the machine is undecidable. If the mapping $x_{t+1} = f(x_t)$ is piecewise affine but *continuous*, the problems related to stability are decidable in dimension 1, undecidable in dimension 3 (Blondel et al., 1999b), and open in dimension 2.

4.5. Other classes of systems

Piecewise linear systems are closely related to the continuous-time *piecewise constant derivative* systems analyzed in Asarin, Maler and Pnueli (1995). Once more, we are given a partition of $\mathbf{R}^n$ into finitely many disjoint subsets $H_1, \ldots, H_m$, and we consider a system of the form

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = b_i \quad \text{when } x \in H_i,$$

where the b_i are given vectors. Assuming that the sets H_i are described by linear inequalities, the trajectories of such systems are piecewise affine functions of time, with break points occurring on the boundaries of the regions. In Asarin et al. (1995) the authors show that, for given states x_b and x_e, the problem of deciding whether x_e is reached by a trajectory starting from x_b, is decidable for systems of dimension two, but is undecidable for systems of dimension three or more. Once more, this undecidability result is obtained by simulating Turing machines and by reducing the halting problem to a reachability problem.

Many of the undecidability results presented in this section are based on simulations of Turing machines using dynamical systems of a particular type. Similar Turing machine simulations are possible by other types of devices; see, e.g., Pollack (1987) for simulation by

high-order neural nets,[13] Bournez and Cosnard (1996) for simulation by analog automata, Garzon (1995) for simulation by cellular automata, Siegelmann and Sontag (1994) and Siegelmann (1998) for simulation by saturated linear systems that can involve arbitrary real (not necessarily rational) numbers, Branicky (1995b), Branicky (1995a), Ruohonen (1997) and Moore (1991) for simulation by differential equations, Moore (1990) for simulation by continuous-time physical devices in dimension three, and Koiran and Moore (1999) for simulation by analytic maps. In all of these constructions, the state of the system is used to encode the configuration (state and tape contents) of a Turing machine, and the dynamics of the system are used to represent the transition rules of the Turing machine.

The results discussed in this section are far from being exhaustive, and there are several topics that we have not touched. These include systems with polynomial nonlinearities (Sontag, 1988b), bilinear systems (Sontag, 1988a; Kawski, 1990), decidability results for hybrid systems (Henzinger, Kopke, Puri & Varaiya, 1998; Lafferierre et al., 1999), and systems evolving on finite groups (Golaszewski & Ramadge, 1989).

5. Stochastic control

We now turn our attention to problems of stochastic optimal control, and discuss exclusively the discrete-time case. Stochastic control problems can be addressed, in principle, using the methods of dynamic programming. However, their practical applicability is somewhat restricted because many natural problems lead to very large state spaces, a phenomenon that Bellman has termed the "curse of dimensionality." By approaching the subject from the point of view of computational complexity, we can get a lot of insight into the types of problems and formulations that can be efficiently solved. We will mainly focus on *Markov decision problems* (MDPs), that involve finite state and control spaces, but we will also discuss briefly the case of continuous state spaces.

5.1. Perfectly observed MDPs

A Markov Decision Problem is specified by a finite state space $S = \{1, \ldots, n\}$, a finite control space $U = \{1, \ldots, m\}$, a (possibly infinite) time horizon T, a discount factor $\alpha \in [0,1]$, transition probabilities $p_{ij}(u,t)$, and one-stage costs $g(i,u,t)$. At a typical time t, the state is equal to some i, a decision u is applied, and the cost $g(i,u,t)$ is incurred. The next state is chosen at random and is equal to j with probability $p_{ij}(u,t)$. We will pay special attention to *stationary* problems, in which the transition probabilities and costs per stage do not depend explicitly on time and can be written as $p_{ij}(u)$ and $g(i,u)$.

We define a *policy* π as a sequence of functions $\mu_t : S \mapsto U$ that specify the decision u_t at time t as a function of the current state i_t, that is, $u_t = \mu_t(i_t)$. If μ_t is the same for all t, the policy is called *stationary*, and we abuse terminology by referring to the corresponding function μ as a policy. Once a policy is fixed, the state of the system evolves as a Markov chain and we define the corresponding expected discounted cost-to-go, as a function of the initial time and state, by

$$J_t^{\pi}(i) = E\left[\sum_{k=t}^{T-1} \alpha^{k-t} g(k, i_k, u_k) \,\middle|\, i_t = i\right],$$

where $E[\cdot]$ stands for expectation with respect to the transition probabilities specified by the policy. When the time horizon T is infinite, we will always assume that the problem is stationary, and that $\alpha < 1$, so that the infinite sum converges and the expectation is well-defined. We define the optimal cost-to-go J_t^* by

$$J_t^*(i) = \inf_{\pi} J_t^{\pi}(i).$$

According to the standard theory of dynamic programming, there exists a policy π which is optimal, that is, $J_t^{\pi}(i) = J_t^*(i)$, for every state i. Furthermore, for infinite-horizon stationary problems, there exists an optimal policy which is stationary, and J_t^* is the same for all t. If J_t^* can be somehow computed, an optimal policy becomes immediately available by letting

$$\mu_t(i) = \arg\min_{u \in U} \left[g(i,u,t) + \alpha \sum_j p_{ij}(u,t) J_{t+1}^*(j) \right].$$

We will mostly refer to complexity results for decision problems where we wish to determine whether $J_t^*(i)$ is less than some given rational number, for some given i and t. Typically, the problem of computing an optimal decision $\mu_t(i)$ for some given i and t has similar complexity.

For finite-horizon problems, J_t^* can be computed by the standard backwards dynamic programming recursion

$$J_t^*(i) = \min_u \left[g(i,u,t) + \alpha \sum_j p_{ij}(u,t) J_{t+1}^*(j) \right],$$

which is initialized with $J_T^*(i) = 0$. This algorithm involves $O(Tmn^2)$ arithmetic operations, which is of the same order of magnitude as the length of the input (the specification of the transition probabilities $p_{ij}(u,t)$ for every i,j,u,t). An interesting twist arises if we deal with stationary problems, because the input now consists of only mn^2 transition probabilities and the time horizon T. Since it takes $O(\log T)$ bits to specify T, dynamic programming becomes an exponential time algorithm. Still, (Tseng, 1990) shows that for any fixed $\alpha < 1$, the problem

[13] In high-order neural nets, activations are combined using multiplication as opposed to just linear combinations.

can be solved with a number of arithmetic operations proportional to $\log T$, provided that the corresponding infinite horizon problem admits a unique optimal policy. (The complexity, however, increases as α approaches 1.)

For infinite-horizon, discounted, stationary problems, a solution boils down to solving the *Bellman equation*

$$J^*(i) = \min_u \left[g(i,u) + \alpha \sum_j p_{ij}(u) J^*(j) \right], \quad i = 1, \ldots, n, \quad (10)$$

which is a nonlinear system of n equations, in n unknowns. It is known that the (unique) solution to this equation can be obtained by solving an equivalent linear programming problem (Puterman, 1994; Bertsekas, 1995). Since linear programming can be solved in polynomial time, the same conclusion follows for such MDPs as well. Note, however, that the complexity of an exact solution will generally depend on the numerical values of the input data. No algorithm (neither for linear programming nor for MDPs) is known in which the number of arithmetic operations is a polynomial in n and m, and this is connected to major open problems in linear programming theory (Bertsimas & Tsitsiklis, 1997). This entire discussion applies equally to the average cost MDP, which is a related problem formulation that we will not discuss further. Suffice to say that for both discounted and average cost problems, linear programming is the only method known to solve them in polynomial time.

In practice, MDPs are often solved by special purpose methods, rather than linear programming. The simplest method, *value iteration* consists of iterating the Bellman equation (10). After k iterations, it is guaranteed to approximate J^* to within $O(\alpha^k)$, and also provides a policy which is within $O(\alpha^k)$ from being optimal. Because there is a finite number of policies, a policy which is ε-optimal, for sufficiently small ε, is guaranteed to be exactly optimal. Building on these observations and certain bounds derived in Tseng (1990), it is shown in Littman et al. (1995) that value iteration can be adapted to solve infinite-horizon discounted MDPs to exact optimality, with computational effort which is polynomial in m, n, $1/(1-\alpha)$, and the size (bit length) of the inputs $g(i,u)$, $p_{ij}(u)$. Thus, value iteration is a polynomial time algorithm as long as α is fixed to a constant value less than 1.

Some of the fastest methods for infinite-horizon MDPs are based on *policy iteration* and its refinements. Policy iteration is a method that produces policies with strict performance improvement at each step, until the algorithm terminates (which must eventually happen because there are only finitely many policies). Policy iteration is known to require no more iterations than value iteration, and the complexity analysis in the preceding paragraph still applies. This analysis still allows for the possibility that the number of iterations (policy improvements) increases as α approaches 1, but in practice the number of iterations seems to be fairly insensitive to the discount factor. We have just identified a major problem in this area; namely, to determine whether the number of iterations in policy iteration can be bounded by a polynomial in the instance size, or even better, whether the number of iterations is bounded by a polynomial in m and n. The answer to these questions is not known. The authors of Melekopoglou and Condon (1994) consider a variant of policy iteration and show, by means of an example, that the number of policy improvements is in the worst-case exponential in n.[14] However, this variant is rather weak in that it can be slower than value iteration, and need not have any implications for the performance of the standard version of policy iteration. Policy iteration bears close resemblance to Newton's method (Puterman & Brumelle, 1978) which, for smooth nonlinear systems of equations, is rapidly convergent in a local sense. Still, this observation is not enough to settle the complexity issue, first because Bellman's equations are not smooth (due to the minimization involved) and, second, because we are interested in convergence in a global sense.

The situation is different when we consider the special case of *deterministic* problems. In the finite-horizon case, deterministic MDPs are essentially equivalent to the much studied and efficiently solvable shortest path problem. For the infinite-horizon discounted case, the problem can be solved in polynomial time, with a number of arithmetic operations that depends on n but not on the discount factor (Papadimitriou & Tsitsiklis, 1987; Littman, 1996). For the infinite-horizon average cost case, the problem amounts to finding a cycle in a graph that has the smallest possible average arc cost. This is again a classical problem, that can be solved with $O(n^3)$ arithmetic operations (Karp, 1978).

5.2. Succintly described problems

Even though MDPs can be solved in time which increases polynomially in the number of states, many problems of practical interest involve a very large number of states, while the problem data (e.g., the transition probabilities) are succinctly described, in terms of a small number of parameters. For concreteness, let us consider the infinite-horizon discounted *multi-armed bandit* problem. We start with a Markov chain on an n-element state space S, with transition probabilities p_{ij} and costs per stage $g(i)$. In a problem involving N arms, the state is an N-tuple $(x_1, \ldots, x_N)$, where each x_k is an element of S and describes the state of the kth arm. The control set is $U = \{1, \ldots, N\}$. If a decision $u = k$ is made, the state of the kth arm makes a transition to a new state, according

[14] The example in Melekopoglou and Condon (1994) involves an undiscounted problem, but as pointed out in Littman et al. (1995), it applies to discounted problems as well.

to the transition probabilities p_{ij}, while the states of the remaining arms stay the same, and a cost $g(x_k)$ is incurred. We now have an MDP defined on a state space of cardinality n^N, but which is described using only $O(n^2)$ parameters (transition probabilities and costs). A straightforward solution using general purpose MDP tools requires at least n^N resources (time and memory). However, due to the ingenious solution by Gittins (Gittins, 1989; Bertsekas, 1995), there is an algorithm which given the current state performs a polynomial (in the instance size) amount of computation and determines an optimal action. (Note that there is no point in trying to precompute an optimal action for every state because there are exponentially many states.)

Unfortunately, the multi-armed bandit problem is a rare exception and for most problems of practical interest no alternatives have been found that would allow for computations that are polynomial in the size of a succinct problem description. In fact, a mild variation of the multi-armed bandit problem, the so-called *restless bandit* problem (Whittle, 1988), becomes PSPACE-hard, even if the problem is deterministic (Papadimitriou & Tsitsiklis, 1994). For a stronger result, the same reference considers a problem of routing and scheduling in closed queueing networks. An instance of this problem involving n customer classes and servers can be described in terms of $O(n^2)$ parameters (service rates, routing probabilities, etc.), but the state space in an MDP formulation is exponentially large. (If each one of n queues can be independently empty or nonempty, we already have 2^n states.) It is shown that the problem is EXP-complete, which implies that any algorithm that attempts a short cut — similar to the one for the multi-armed bandit problem — has *provably* exponential complexity. This result is developed for average cost problems, but the same proof also works for the finite-horizon case, as long as the time horizon is large (exponential in n). We also refer to Mundhenk et al. (1997, Theorem 6.7) and Littman (1997, Theorem 1) which study succinctly described exponential or infinite horizon MDPs, and provide EXP-completeness results consistent with the one discussed here. The interest on this subject partly stems from recent research activity on propositional planning systems within the field of artificial intelligence. It is shown in Littman (1997) that the EXP-completeness result applies to several classes of planning systems in the literature. Finally, note that if we restrict attention to deterministic succinctly described MDPs, the queueing network problem becomes PSPACE-complete (Papadimitriou and Tsitsiklis, 1994, Theorem 3), which is consistent with results on planning problems in deterministic domains (Bylander, 1994; Littman, Goldsmith & Mundhenk, 1998).

Another case of interest refers to succinctly described MDPs over a time horizon which is polynomial in the instance size, and therefore logarithmic in the number of states. This is motivated in the planning literature by the fact that excessively long plans cannot be practically useful. For this case, the problem becomes PSPACE-complete. This is proved in Littman (1997, Theorems 2 and 3) for several planning systems, and in Mundhenk et al. (1997, Corollary 6.14) for a related class of "compressed" MDPs; see also Littman et al. (1998).

5.3. Problems with continuous state spaces

Let us briefly discuss some related work on problems with continuous state spaces. Problems of this type do not admit closed-form or exact algorithmic solutions. A rare exception is the problem of optimal control of linear systems, when the cost per stage is a nonnegative quadratic function of the state and the control, and this is because the optimal cost-to-go function turns out to be quadratic. The finite horizon problem can be solved exactly using the standard dynamic programming recursion. The infinite horizon problem amounts to solving an algebraic Riccati equation, which cannot be done exactly, but there are rapidly convergent algorithms that can quickly produce solutions within any desired precision.

General continuous-state problems can be solved approximately, by discretizing them, as long as the problem data (transition probabilities and cost per stage) are sufficiently smooth functions of the state (Whitt, 1978a, b). References Chow and Tsitsiklis (1989) and Chow and Tsitsiklis (1991) establish that

$$O\left(\frac{1}{\varepsilon^{2n+m}}\right)$$

arithmetic operations are necessary and sufficient for uniformly approximating the function J^* within ε, for the case where the state and control spaces are the sets $[0,1]^n$ and $[0,1]^m$, respectively, under a Lipschitz continuity assumption on the problem data, and for a fixed discount factor α.[15] The $1/\varepsilon^m$ complexity arises because we need to search a grid with ε spacing in order to find a close-to-optimal decision for any given state. This issue disappears if the control set has finite cardinality. However, the curse of dimensionality remains in effect due to the $1/\varepsilon^{2n}$ term.

In some important work, Rust (1997a) has shown that the curse of dimensionality can be bypassed by allowing for randomized algorithms. The advantage of randomization can be best understood in terms of the multivariable integration problem. Deterministic methods require a fine discretization of the space of interest (which leads to the curse of dimensionality), whereas Monte Carlo

[15] Although these results are stated for the problem of uniform approximation, the proof of the lower bounds also applies to the problem of approximating J^* at a particular given state.

methods produce ε-approximate estimates of the value of an integral, with high probability, using a dimension-independent number $O(1/\varepsilon^2)$ of samples. Rust's method is a variant of value iteration which only looks at a relatively small number of randomly sampled representative states.

The complexity estimates in all of these references are proportional to the Lipschitz constant of the problem data. When one considers practical problems of increasing dimension, the Lipschitz constant will often increase exponentially. To see this, consider the probability density function $p(x) = 2x$ for a scalar random variable that takes values in [0,1]. If we introduce n independent copies of this random variable, the density becomes $2^n x_1 \cdots x_n$ on the unit cube and the Lipschitz constant increases exponentially with n. Thus, the curse of dimensionality is not fully eliminated. Nevertheless, the results of Rust suggest a definite advantage in using random sampling (or even deterministic but pseudo-random sampling (Rust, 1997b)).

5.4. Problems with imperfect information

We now return to finite-state systems. So far, we have allowed the decision u to be a function of the system state. In an alternative formulation, the controller only has access to partial observations $y_t = h(i_t)$, where h is a prespecified function on the state space, which we assume to have finite range. (One could augment this model by including observation noise. However, by suitably redefining the state of the system, the problem with observation noise can be reduced to a problem without it.) In the standard formulation of the partially observed Markov decision problem (POMDP), the decision u_t at time t is allowed to be a function of all past observations; that is, we are dealing with policies of the form $u_t = \mu_t(y_0, y_1, \ldots, y_t)$.

POMDPs can be reduced to perfectly observed MDPs, by redefining the state at time t to be the past observation history $(y_0, y_1, \ldots, y_t)$ (Bertsekas, 1995). With a time horizon T, the cardinality of the redefined state space (and the resulting dynamic programming algorithm) increases exponentially in T, but this is to some extent unavoidable. Indeed, if we assume that the original state space has cardinality n and let $T = n$, the problem is PSPACE-complete (Papadimitriou & Tsitsiklis, 1987, Theorem 6). Incidentally, the MDP obtained in this way is another example of a succinctly described MDP, with a time horizon which is polynomial in the size of the instance, and so this result is in agreement with the results mentioned in Section 5.2.

Another standard reformulation of an n-state POMDP to a perfectly observed MDP uses a redefined state (the "information state") which is the posterior distribution of the original state, given the available observations. This is now an MDP with a continuous state space (the unit simplex in $\mathbf{R}^n$), and smooth transition probabilities, and it can be solved approximately (cf. the discussion in Section 5.3). If the preimage of every possible observation has cardinality bounded by a constant k, and if $k \geq 3$, the problem remains NP-hard. (The case $k = 2$ is open.) But we are now dealing with an essentially k-dimensional state space and for any fixed k, the problem can be solved to within ε of optimality with a number of arithmetic operations which is polynomial in n, m, $1/\varepsilon$, and the time horizon (Burago, Rougement & Slissenko, 1996, Theorem 8).

Infinite-horizon POMDPs cannot be solved exactly by reducing them to perfectly observed MDPs, because the latter have an infinite state space. It turns out that infinite-horizon POMDPs are undecidable (under total, discounted, or average cost criteria) (Madani, Hanks & Condon, 1999), because they are closely related to the "emptiness" problem for probabilistic finite-state automata, which is known to be undecidable. There are also results for succinctly represented finite-horizon POMDPs. Generally speaking, the complexity (as a function of the instance size) is exponentially higher (Mundhenk et al., 1997).

Let us also mention an interesting special case of POMDPs, namely, the case where there are no observations (open-loop control). Assuming a time horizon T equal to the cardinality of the state space, the problem is NP-complete (Papadimitriou and Tsitsiklis, 1987, Corollary 2). See also Mundhenk et al. (1997) for results concerning the case of succinct problem descriptions, and Burago et al. (1996, Theorem 6) for nonapproximability results.

At this point, it is interesting to draw a connection with some of the results discussed in Section 3. For any decision u, let $P(u)$ be the corresponding transition probability matrix. Let π be a row vector with the probability distribution of the initial state, and let g be a column vector whose ith component is the cost if the system is found at state i at time T. (Assume that there are no costs at intermediate times.) Optimal open-loop control amounts to minimizing $\pi P(u_0)P(u_1)\cdots P(u_{T-1})g$ over all sequences $u_0, u_1, \ldots, u_{T-1}$, and as stated earlier, this is an NP-complete problem. But this problem can also be viewed as referring to the worst-case behavior of a time-varying linear system $x_{t+1} = P_t x_t$, where each matrix P_t is chosen from within a finite set of matrices. Indeed, this observation has been exploited in Tsitsiklis and Blondel (1997a) to establish NP-hardness results for problems related to time-varying linear systems (cf. Section 3.5).

5.5. Nonclassical information structures

In stochastic control problems with a nonclassical information structure, the decision at time t is made without knowledge of the full history of past

observations, and this setting includes problems of decentralized stochastic control and team decision theory. When arbitrary information structures are allowed, POMDPs are obtained as a special case and therefore, the complexity is at least as large. But negative results can be obtained even for seemingly simple special cases.

Consider the static team decision problem (Radner, 1962). Decision maker 1 (respectively, 2) observes a random variable y_1 (respectively, 2) and makes a decision according to a rule of the form $u_1 = \mu_1(y_1)$ (respectively, $u_2 = \mu_2(y_2)$), where u_1 and u_2 are restricted to lie in finite sets. We consider the objective

$$\min_{\mu_1,\,\mu_2} E[g(y_1, y_2, \mu_1(y_1), \mu_2(y_2))]$$

and ask the question whether it is less than a given rational number. Here, y_1 and y_2 are finite-valued random variables with a given (rational-valued) probability mass function, and g is a given (rational-valued) cost function. This problem was shown NP-complete in Papadimitriou and Tsitsiklis (1982), and this result remains valid for a special case that arises in the context of decentralized detection (Tsitsiklis, 1984; Tsitsiklis and Athans, 1985).

The same problem can be recast as an MDP with a nonclassical information structure, and a time horizon $T = 3$. We view (y_1, y_2) as the state after the first transition. At stages 1 and 2, y_1 and y_2 are observed, respectively, but we require the decisions at any time t to be functions of the present observation only, of the form $u_t = \mu_t(y_t)$, $t = 1,2$. Thus, imperfectly observed MDPs with a nonclassical information structure are NP-complete even when the time horizon is $T = 3$. (With a time horizon of two, and a fixed initial state, we are back to the case of a classical information structure and such problems are easily seen to be polynomial-time solvable.)

Problems in which we restrict to policies of the form $u_t = \mu_t(y_t)$ are referred to as "time-dependent partially observable" in Mundhenk et al. (1997). This reference characterizes the complexity of several related variants; see also Littman (1994). We also refer the reader to Papadimitriou and Tsitsiklis (1986) which establishes the NP-completeness of a discrete variant of Witsenhausen's counterexample (Witsenhausen, 1968).[16]

5.6. Supervisory control of discrete-event systems

There is an extensive literature on supervisory control of discrete-event systems, within the framework of Ramadge (1983) and Ramadge and Wonham (1987). Many of the problem formulations in this area lead to problems similar to MDPs, except that the transitions are not determined stochastically, but may be chosen by an adversary. The difference between the two model types is rather minor, because a variant of dynamic programming, applicable to min–max problems, can be used. In particular, the standard formulation of the supervisory control problem can be solved in polynomial time if the supervisor has perfect information (Ramadge and Wonham, 1987), but becomes PSPACE-hard for the case of imperfect information, because it has essentially the same structure as POMDPs (Tsitsiklis, 1989). See also Rudie and Willems (1995) for some related results.

5.7. Zero-sum Markov games

The setting here is similar to MDPs except that there are two competing decision makers, one acting at odd times, the other at even times. One decision maker wants to maximize the total expected cost, and the other attempts to minimize it. The finite-horizon version of the problem is similar to MDPs: with perfect information, it can be solved by a simple adaptation of the dynamic programming recursion. For the infinite-horizon discounted case, the basic theory, including a suitable Bellman equation is again the same as for MDPs (Shapley, 1953; Raghavan and Filar, 1991). However, a reformulation based on linear programming is not known, and unlike ordinary MDPs, a polynomial-time algorithm is not available. On the other hand, a negative complexity result is not available either, and this problem's complexity is an important open problem. We also refer the reader to Condon (1989, 1992) and Zwick and Paterson (1996) for complexity results on more general classes of games.

6. Conclusions

The subject of complexity in systems and control is multifaceted and interesting in many different ways. It can help the practitioner in choosing problem formulations, and in calibrating expectations of what can be algorithmically accomplished. For the systems theorist or the applied mathematician, it raises a variety of challenging open problems that require a diverse set of tools from both discrete and continuous mathematics. Finally, for the theoretical computer scientist, the problems in systems and control theory provide the opportunity to relate abstractly defined complexity classes with specific problems of practical interest.

Acknowledgements

We gratefully acknowledge the input of several people, including Eugen Asarin, Olivier Bournez, Michael

[16] This counterexample showed that optimal controllers for LQG problems with a nonclassical information structure are not necessarily linear, thus eliminating the hope of extending LQG theory to decentralized control.

Branicky, Laurent El Ghaoui, Minyue Fu, Stéphane Gaubert, Leonid Gurvits, Pascal Koiran, Michael Littman, Oded Maler, Maurice Mignotte, Cristopher Moore, Jiri Rohn, John Rust, Rodolphe Sepulchre, Eduardo Sontag, Roberto Tempo, Jan van Schuppen, M. Vidyasagar and Louis Wehenkel. Also, the second author has greatly benefitted from collaborating with Christos Papadimitriou over the years.

References

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. Reading, MA: Addison-Wesley.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., & Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, *138*, 3–34.

Anderson, B. D. O., Bose, N. K., & Jury, E. I. (1975). Output feedback stabilization and related problem — solutions via decision methods. *IEEE Transactions on Automatic Control*, *20*, 53–66.

Asarin, E., Maler, O., & Pnueli, A. (1995). Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, *138*, 35–66.

Baccelli, F., Cohen, G., Olsder, G.-J., & Quadrat, J.-P. (1992). *Synchronization and linearity*. New York: Wiley.

Barabanov, N. E. (1988). Lyapunov indicators of discrete inclusions, parts I, II and III, *Avtomatika i Telemekhanika*, *2*, 40–46, *3*, 24–29 and *5*, 17–24. Translation in *Automatic remote control* (1988) part I, *49*(3), 152–157, part II, *49*(5), 558–565.

Barmish, B. R., Floudas, C. A., Hollot, C. V., & Tempo, R. (1994). A global linear programming solution to some open robustness problems. *Proceedings of the American control conference*, Seattle, USA (pp 3871–3877).

Bartlett, A. C., Hollot, C. V., & Huang, L. (1988). Root locations of an entire polytope of polynomials: It suffices to check the edges. *Mathematics of Control, Signals, and Systems*, *1*, 61–71.

Basu, S., Pollack, R., & Roy, M.-F. (1996). On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, *43*, 1002–1046.

Berger, M., & Wang, Y. (1992). Bounded semigroups of matrices. *Linear Algebra Applications*, *166*, 21–27.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*. Belmont, MA: Athena Scientific.

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization*. Belmont, MA: Athena Scientific.

Bialas, S. (1985). A necessary and sufficient condition for the stability of convex combinations of stable polynomials or matrices. *Bulletin of the Polish Academy of Sciences, Technical Sciences*, *33*, 473–480.

Blondel, V. D., Bournez, O., Koiran, P., Papadimitriou, C., & Tsitsiklis, J. N. (1999a). Deciding stability and mortality of piecewise affine dynamical systems. Preprint.

Blondel, V. D., Bournez, O., Koiran, P., & Tsitsiklis, J. N. (1999b). The stability of saturated linear dynamical systems is undecidable. Preprint.

Blondel, V. D., Gaubert, S., & Tsitsiklis, J. N. (1998). Approximating Lyapunov exponents of matrices in the max-algebra is NP-hard. Preprint.

Blondel, V. D., Gevers, M., & Lindquist, A. (1995). Survey on the state of systems and control. *European Journal of Control*, *1*, 5–23.

Blondel, V. D., & Portier, N. (1999a). The presence of a zero in an integer linear recurrent sequence is NP-hard to decide. Preprint.

Blondel, V. D., & Tsitsiklis, J. N. (1997a). NP-hardness of some linear control design problems. *SIAM Journal of Control and Optimization*, *35*, 2118–2127.

Blondel, V. D., & Tsitsiklis, J. N. (1997b). When is a pair of matrices mortal?. *Information Processing Letters*, *63*, 283–286.

Blondel, V. D., & Tsitsiklis, J. N. (1998). Overview of complexity and decidability results for three classes of elementary nonlinear systems. In Y. Yamamoto, & S. Hara, *Learning, control and hybrid systems* (pp. 46–58) Heidelberg: Springer.

Blondel, V. D., & Tsitsiklis, J. N. (1999a). Complexity of stability and controllability of elementary hybrid systems. *Automatica*, *35*, 479–489.

Blondel, V. D., & Tsitsiklis, J. N. (1999b). Boundedness of finitely generated matrix semigroups is undecidable. Preprint.

Blondel, V. D., Sontag, E. D., Vidyasagar, M., & Willems, J. C. (1999c). *Open problems in mathematical systems and control theory*. London: Springer, http://www.inma.ucl.ac.be/ ~ blondel/openprobs.html.

Blum, L., Cucker, F., Shub, M., & Smale, S. (1998). *Complexity and real computation*. New York: Springer.

Bournez, O., & Cosnard, M. (1996). On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, *168*, 417–459.

Bournez, O., & Branicky, M. (1998). *On matrix mortality in low dimensions*. Technical Report 98-01, Verimag, France.

Boyd, S., El Ghaoui, L., Feron, E., & Balakrishnan, V. (1994). Linear matrix inequalities in system and control theory. *SIAM Studies in Applied Mathematics,* vol. 15. Philadelphia, PA: SIAM.

Braatz, R., Young, P., Doyle, J., & Morari, M. (1994). Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, *39*, 1000–1002.

Branicky, M.S. (1995a). *Studies in hybrid systems: Modeling, analysis, and control*. Ph.D. thesis, Center for Intelligent Control Systems, Cambridge, MA: MIT Press.

Branicky, M. S. (1995b). Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, *138*, 67–100.

Brayton, R., & Tong, C. (1980). Constructive stability and asymptotic stability of dynamical systems. *IEEE Transactions on Circuits and Systems*, *27*, 1121–1130.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *69*, 165–204.

Chow, C. -S., & Tsitsiklis, J. N. (1989). The complexity of dynamic programming. *Journal of Complexity*, *5*, 466–488.

Chow, C. -S., & Tsitsiklis, J. N. (1991). An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, *36*, 898–914.

Condon, A. (1989). *Computational models of games*. Cambridge, MA: The MIT Press.

Condon, A. (1992). The complexity of stochastic games. *Information and Computation*, *96*, 203–224.

Cook, S.A. (1971). The complexity of theorem proving procedures. *Proceedings of the third ACM symposium on the theory of computing* (pp. 117–128).

Coxson, G. E., & De Marco, C. L. (1994). The computational complexity of approximating the minimal perturbation scaling to achieve instability in an interval matrix. *Mathematics of Control, Signals, and Systems*, *7*, 279–291.

Coxson, G. E. (1993). *Computational complexity of robust stability and regularity in families of linear systems*. Ph.D. thesis, Electrical and Computer Engineering Department, University of Wisconsin at Madison.

Dasgupta, S. (1988). Kharitonov's theorem revisited. *Systems and Control Letters*, *11*, 381–384.

Daubechies, I., & Lagarias, J. C. (1992). Sets of matrices all infinite products of which converge. *Linear Algebra Applications*, *162*, 227–263.

Davis, M. (1973). Hilbert's tenth problem is unsolvable. *American Mathematical Monthly*, *80*, 233–269.

Davis, M. (1982). *Computability and unsolvability*. New York: Dover.

Demmel, J. W. (1992). The component-wise distance to the nearest singular matrix. *SIAM Journal of Matrix Analysis and Applications*, *13*, 10–19.

Doyle, J. C. (1982). Analysis of feedback systems with structured uncertainties. *IEE Proceedings*, *129*, 240–250.

Fu, M. (1999). Approximation of complex μ. In V. D. Blondel, E. D. Sontag, M. Vidyasagar, & J. C. Willems, *Open problems in mathematical systems and control theory*, Problem 22. London: Springer.

Fu, M., & Luo, Z.-Q. (1997). Computational complexity of a poblem aising in fixed order output feedback design. *Systems and Control Letters*, *30*, 209–215.

Fu, M., & Barmish, B. R. (1988). Maximal unidirectional perturbation bounds for stability of polynomials and matrices. *Systems and Control Letters*, *11*, 173–179.

Fu, M., & Dasgupta, S. (1998). Computational complexity of real structured singular value in ℓ_p setting. Preprint.

Fu, M., Dasgupta, S., & Blondel, V. (1995). Robust stability under a class of nonlinear parametric perturbations. *IEEE Transactions on Automatic Control*, *40*, 213–223.

Gantmacher, F.R. (1959). *Matrix theory,* vol. 2. New York: Chelsea Publishing Company.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman and Co.

Garzon, M. (1995). *Models of massive parallelism. Analysis of cellular automata and neural networks*. Berlin: Springer.

Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. New York: Wiley.

Golaszewski, C., & Ramadge, P. J. (1989). The complexity of some reachability problems for a system on a finite group. *Systems and Control Letters*, *12*, 431–435.

Gripenberg, G. (1996). Computing the joint spectral radius. *Linear Algebra Applications*, *234*, 43–60.

Gurvits, L. (1995). Stability of discrete linear inclusions. *Linear Algebra Applications*, *231*, 47–85.

Gurvits, L. (1996). *Stability of linear inclusions — Part 2*. Technical Report TR 96-173, NEC Research.

Henzinger, T., Kopke, P., Puri, A., & Varaiya, P. (1998). What's decidable about hybrid automata. *Journal of Computer and System Science*, *57*, 97–124.

Hertz, D. (1992). The extreme eigenvalues and stability of real symmetric interval matrices. *IEEE Transactions on Automatic Control*, *37*, 532–535.

Hooper, P. (1966). The undecidability of the Turing machine immortality problem. *The Journal of Symbolic Logic*, *2*, 219–234.

Hopcroft, J. E., & Ullman, J. D. (1969). *Formal languages and their relation to automata*. Reading, MA: Addison-Wesley.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation*. Reading, MA: Addison-Wesley.

Hyotyniemu, H. (1997). On unsolvability of nonlinear system stability. *Proceedings of the fourth European control conference*, Brussels, vol. 636 (pp. 1–6).

Jones, J. P. (1974). Recursive undecidability. *American Mathematical Monthly*, *81*, 724–738.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. Miller, & J. Thatcher, Complexity of computer computations. New York: Plenum Press (pp. 85–104).

Karp, R. M. (1978). A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, *23*, 309–311.

Kawski, M. (1990). The complexity of deciding controllability. *Systems and Control Letters*, *15*, 9–14.

Khargonekar, P. P., & Tikku, A. (1996). Randomized algorithms for robust control analysis have polynomial complexity. *Proceedings of the conference on decision and control*.

Kharitonov, V. L. (1978). Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. *Differentsial'nye Uravneniya*, *14*, 2086–2088.

Kilian, J., & Siegelmann, H. (1996). The dynamic universality of sigmoidal neural networks. *Information and Computation*, *128*, 48–56.

Klarner, D. A., Birget, J.-C., & Satterfield, W. (1991). On the undecidability of the freeness of integer matrix semigroups. *International Journal of Algebra and Computation*, *1*, 223–226.

Koiran, P. (1996). A family of universal recurrent networks. *Theoretical Computer Science*, *168*, 473–480.

Koiran, P., & Moore, C. (1999). Closed-form analytic maps in 1 or 2 dimension can simulate universal Turing machines. *Theoretical Computer Science*, *210*, 217–223.

Koiran, P., Cosnard, M., & Garzon, M. (1994). Computability properties of low-dimensional dynamical systems. *Theoretical Computer Science*, *132*, 113–128.

Kokame, H., & Mori, T. (1992). A branch-and-bound method to check the stability of a polytope of matrices. In M. Mansour, S. Balemi & W. Truoel, *Robustness of dynamics systems with parameter uncertainties* (pp. 125–137). Basel: Birkhauser.

Kozyakin, V. S. (1990). Algebraic unsolvability of problem of absolute stability of desynchronized systems. *Automation and Remote Control*, *51*, 754–759.

Kreinovich, V., Lakeyev, A., & Rohn, J. (1995). Computational complexity of interval algebraic problems: Some are feasible and some are computationally intractable — a survey. Scientific computing and validated numerics (Wuppertal, 1995) (pp. 293–306). Mathematical Research, vol. 90. Berlin: Akademie Verlag.

Kreinovich, V., Lakeyev, A., Rohn, J., & Kahl, P. (1997). Computational complexity and feasibility of data processing and interval computation. Applied Optimization Series, vol. 10. Dordrecht: Kluwer Academic.

Krom, M. (1981). An unsolvable problem with products of matrices. *Mathematical Systems Theory*, *14*, 335–337.

Krom, M., & Krom, M. (1989). Recursive solvability of problems with matrices. *Z. Math. Logik Grundlagen Math.*, *35*, 437–442.

Lafferierre, G., Pappas, G. J., & Yovine, S. (1999). A new class of decidable hybrid systems. *Lecture Notes in Computer Sciences,* vol. 1596. Heidelberg: Springer.

Lagarias, J. C., & Wang, Y. (1995). The finiteness conjecture for the generalized spectral radius of a set of matrices. *Linear Algebra Applications*, *214*, 17–42.

Littman, M. L. (1994). Memoryless policies: Theoretical limitations and practical results.. In D. Cliff, P. Husbands, J.-A. Meyer, & S. W. Wilson, *From animals to animats 3: Proceedings of the third International Conference on simulation of adaptive behavior*. Cambridge, MA: The MIT Press.

Littman, M. L. (1996). *Algorithms for sequential decision making*. Ph.D. thesis, Department of Computer Science, Brown University.

Littman, M. L. (1997). Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th national conference on artificial intelligence* (pp. 748–754). Cambridge, MA: AAAI Press/The MIT Press.

Littman, M. L., Dean, T. L., & Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. *Proceedings of the 11th annual conference on uncertainty in artificial intelligence*, Montreal, Québec, Canada (pp. 394–402).

Littman, M. L., Goldsmith, J., & Mundhenk, M. (1998). The computational complexity of probabilistic plan existence and evaluation. *Journal of AI Research*, *9*, 1–36.

Liu, D., & Michel, A. (1994). *Dynamical systems with saturation nonlinearities: Analysis and design*. London: Springer.

Madani, O., Hanks, S., & Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems, *Proceedings of AAAI99*, to appear.

Matiyasevich, Yu. (1970). Enumerable sets are diophantine. *Doklady Akademii Nauk SSSR*, *191*, 279–282.

Matiyasevich, Y., & Sénizergues, G. (1996). Decision problem for semi-Thue systems with a few rules. *Proceedings of LICS96* (pp. 523–531). Silver Spring, MD: IEEE Computer Society Press.

Melekopoglou, M., & Condon, A. (1994). On the complexity of the policy improvement algorithm for markov decision processes. *ORSA Journal on Computing*, *6*, 188–192.

Mignotte, M., Shorey, T. N., & Tijdeman, R. (1984). The distance between terms of an algebraic recurrence sequence. *Journal fur die Reine and Angewandte Mathematik*, *349*, 63–76.

Minnichelli, R. J., Anagnost, J. J., & Desoer, C. A. (1989). An elementary proof of Kharitonov's theorem with extensions. *IEEE Transactions on Automatic Control*, *34*, 995–998.

Minsky, M. (1967). *Computation. Finite and infinite machines*. Englewood Cliffs, NJ: Prentice-Hall.

Moore, C. (1990). Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, *64*, 2354–2357.

Moore, C. (1991). Generalized shifts: Undecidability and unpredictability in dynamical systems. *Nonlinearity*, *4*, 199–230.

Moore, C. (1998). Finite-dimensional analog computers: Flows, maps, and recurrent neural networks. *Proceedings of the first international conference on unconventional models of computation*, Auckland, New Zealand.

Mundhenk, M. J., Goldsmith, J. Lusena, C., & Allender, E. (1997). *Encyclopedia of complexity results for finite horizon Markov decision process problems*. Technical Report 273-97, Computer Science Department, University of Kentucky.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. New York: Wiley.

Nemirovskii, A. (1993). Several NP-Hard problems arising in robust stability analysis. *Mathematics of Control. Signals, and Systems*, *63*, 99–105.

Nemirovsky, A. S., & Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization*. New York: Wiley.

Orponen, P. (1994). Computational complexity of neural networks: A survey. *Nordic Journal of Computing*, *1*, 94–100.

Papadimitriou, C. H. (1994). *Computational complexity*. Reading, MA: Addison-Wesley.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1982). On the complexity of designing distributed protocols. *Information and Control*, *53*, 211–218.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1986). Intractable problems in control theory. *SIAM Journal of Control and Optimization*, *24*, 639–654.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, *12*, 441–450.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1994). The complexity of optimal queueing network control. *Proceedings of the nineth annual conference on struture in complexity theory*. Amsterdam, The Netherlands. Full version available from: http://web.mit.edu/jnt/www/publ.html.

Paterson, M. (1970). Unsolvability in 3×3 matrices. *Studies in Applied Mathematics*, *49*, 105–107.

Poljak, S., & Rohn, J. (1993). Checking robust nonsingularity is NP-hard. *Mathematics of Control, Signals, and Systems*, *6*, 1–9.

Pollack, J. B. (1987). *On connectionist models of natural language processing*. Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana.

Post, E. L. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, *52*, 264–268.

Puterman, M. L. (1994). *Markov decision processes*. New York: Wiley.

Puterman, M. L., & Brumelle, S. L. (1978). The analytic theory of policy iteration. In M.L. Puterman, *Dynamic programming and its applications.* New York: Academic Press.

Raghavan, T. E. S., & Filar, J. A. (1991). Algorithms for stochastic games — a survey. *Zeitschrift für O. R.*, *35*, 437–472.

Radner, R. (1962). Team decision problems. *Annals of Mathematical Statistics*, *33*, 857–881.

Ramadge, P. J. (1983). *Control and supervision of discrete event processes*. Ph.D. thesis, Department of Electrical Engineering, University of Toronto.

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control Optimization*, *25*, 206–230.

Rogozhin, Y. (1996). Small universal Turing machines. *Theoretical Computer Science*, *168*, 215–240.

Rohn, J. (1994a). Positive definiteness and stability of interval matrices. *SIAM Journal on Matrix Analysis and Applications*, *15*, 175–184.

Rohn, J. (1994b). Checking positive definiteness or stability of symmetric interval matrices is NP-hard. *Commentationes Mathematicae Universitatis Carolinae*, *35*, 795–797.

Rohn, J. (1997). Complexity of some linear problems with interval data. *Reliable Computing*, *3*, 315–323.

Rosenthal, J., & Wang, X. (1997). Inverse eigenvalue problems for multivariable linear systems. In C. I. Byrnes, B. N. Datta, D. Gilliam, & C. F. Martin, *Systems and control in the twenty-first century* (pp. 289–311). Basel: Birkhauser.

Rota, G.-C., & Strang, G. (1960). A note on the joint spectral radius. *Nederlandse Koninklÿke Akademie & van Wetenschappen Indagationes Mathematicae*, *22*, 379–381.

Rudie, K., & Willems, J. C. (1995). The computational complexity of decentralized discrete-event control problems. *IEEE Transactions on Automatic Control*, *40*, 1313–1319.

Ruohonen, K. (1997). Undecidable event detection problems for ODEs of dimension one and two. *RAIRO Information and Theoretical Applications*, *31*, 67–79.

Rust, J. (1996). Numerical dynamic programming in economics. In H. Amman, D. Kendrick, & J. Rust, *Handbook of computational economics* (pp. 619–729). Amsterdam: Elsevier (Chapter 14).

Rust, J. (1997a). Using randomization to break the curse of dimensionality. *Econometrica*, *65*(3), 487–516.

Rust, J. (1997b). A comparison of policy iteration methods for solving continuous-state, infinite-horizon Markovian decision problems using random, quasi-random, and deterministic discretizations. Available from the Economics Working Paper archive: http://econwpa.wustl.edu/eprints/comp/papers/9704/9704001.abs

Safonov, M. G. (1982). Stability margins of diagonally perturbed multivariable feedback systems. *Proceedings of IEE-D*, *129*, 251–256.

Salomaa, A. (1985). *Computation and automata*. Cambridge: Cambridge University Press.

Salomaa, A., & Soittola, M. (1978). *Automata-theoretic aspects of formal power series*. New York: Springer.

Samake, K. (1996). *Suite récurrentes linéaires, problèmes d'effectivité*. Ph.D. thesis, Institut de Recherche Mathématique Avançée, Université Louis Pasteur, Strasbourg, France.

Schrijver, A. (1986). *Theory of linear and integer programming*. New York: Wiley.

Seidenberg, A. (1954). A new decision method for elementary algebra. *Annals of Mathematics*, *60*, 365–374.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, *39*, 1095–1100.

Sideris, A. (1991). An efficient algorithm for checking the robust stability of a polytope of polynomials. *Mathematics of Control, Signals, and Systems*, *4*, 315–337.

Siegelmann, H. T. (1998). *Neural networks and analog computation. Beyond the Turing limit*. Boston: Birkhauser.

Siegelmann, H. T., & Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, *4*, 77–80.

Siegelmann, H., & Sontag, E. (1994). Analog computation via neural networks. *Theoretical Computer Science*, *131*, 331–360.

Siegelmann, H., & Sontag, E. (1995). On the computational power of neural nets. *Journal of Computer and System Sciences*, 132–150.

Soh, C. B. (1990). Necessary and sufficient conditions for stability of symmetric interval matrices. *International Journal of Control*, *51*, 243–248.

Sontag, E. D. (1981). Nonlinear regulation: the piecewise linear approach. *IEEE Transactions on Automatic Control*, *26*, 346–358.

Sontag, E. D. (1985). Real addition and the polynomial hierarchy. *Information Processing Letters*, *20*, 115–120.

Sontag, E. D. (1988a). Controllability is harder to decide than accessibility. *SIAM Journal of Control and Optimization*, *26*, 1106–1118.

Sontag, E. D. (1988b). Some complexity questions regarding controllability. *Proceedings IEEE conference on decision and control*, San Antonio, TX (pp. 1326–1329).

Sontag, E. D. (1995). From linear to nonlinear: Some complexity comparisons. *Proceedings of the IEEE conference decision and control*, New Orleans (pp. 2916–2920).

Sontag, E. D. (1996a). Interconnected automata and linear systems: A theoretical framework in discrete-time. In R. Alur, T. Henzinger, & E. D. Sontag, *Hybrid systems III: Verification and control* (pp. 436–448). Berlin: Springer.

Sontag, E. D. (1996b). Recurrent neural networks: Some learning and systems-theoretic aspects. Preprint.

Syrmos, V., Abdallah, C., & Dorato, P. (1994). Static output feedback: A survey. *Automatica*, *33*, 125–137.

Tarski, A. (1951). *A decision method for elementary algebra and geometry*. Berkeley, USA: University of California Press.

Tempo, R., Bai, E. W., & Dabbene, F. (1996). Probabilistic robustness analysis: Explicit bounds for the minimum number of sampling points. *Proceedings on IEEE conference decision and control* (pp 2412–2420).

Toker, O. (1997). On the complexity of the robust stability problem for linear parameter varying systems. *Automatica*, *33*, 2015–2017.

Toker, O. (1996). On the algorithmic unsolvability of some stability problems for discrete event systems. *Proceedings of IFAC world congress* (pp. 353–358).

Toker, O., & de Jager, P. (1999). Conservatism of the standard upper bound test. In V. D. Blondel, E. D. Sontag, M. Vidyasagar, & J. C. Willems, *Open problems in mathematical systems and control theory*, Problem 20. London: Springer.

Toker, O., & Ozbay, H. (1998). On the complexity of purely complex μ computation and related problems in multidimensional systems. *IEEE Transactions on Automatic Control*, *43*, 409–414.

Toker, O., & Ozbay, H. (1996). Complexity issues in robust stability of linear delay-differential systems. *Mathematics of Control, Signals, and Systems*, *9*, 386–400.

Toker, O., & Ozbay, H. (1995). On the NP-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback. *Proceedings of the IEEE American control conference*, Seattle, WA (pp. 2525–2526).

Traub, J., Wasilkowski, G., & Woźniakowski, H. (1988). *Information-based complexity*. New York: Academic Press.

Trei, S. (1999). The gap between complex structured singular value and its upper bound is infinite. Preprint, http://www.math.msu.edu/$\sim$treil/research.html.

Tseng, P. (1990). Solving h-horizon, stationary Markov decision problems in time proportional to $\log h$. *Operations Research Letters*, *9*, 287–297.

Tsitsiklis, J. N. (1984). *Problems in decentralized decision making and computation*. Ph.D. thesis, Department of Electrical Engineering and Computer Science. Cambridge, MA: MIT Press.

Tsitsiklis, J. N. (1989). On the control of discrete-event dynamical systems. *Mathematics of Control, Signals, and Systems*, *2*, 95–107.

Tsitsiklis, J. N., & Athans, M. (1985). On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, *30*, 440–446.

Tsitsiklis, J. N., & Blondel, V. D. (1997a). The Lyapunov exponent and joint spectral radius of pairs of matrices are hard — when not impossible — to compute and to approximate. *Mathematics of Control Signals, and Systems*, *10*, 31–40.

Tsitsiklis, J. N., & Blondel, V. D. (1997b). Lyapunov exponents of pairs of matrices, a correction. *Mathematics of Control, Signals, and Systems*, *10*, 381.

Vidyasagar, M. (1998). *A theory of learning and generalization. With applications to neural networks and control systems*. London: Springer.

Vidyasagar, M., & Blondel, V. D. (1999). Probabilistic solutions to some NP-hard matrix problems. Preprint.

Whitt, W. (1978a). Approximations of dynamic programs — I. *Mathematics of Operations Research*, *3*, 231–243.

Whitt, W. (1978b). Approximations of dynamic programs — II. *Mathematics of Operations Research*, *4*, 179–185.

Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. In J. Gani, *A celebration of applied probability*, *Journal of Applied Probability*, *25A*, 287–298.

Witsenhausen, H. S. (1968). A counterexample in stochastic optimum control. *SIAM Journal of Control and Optimization*, *6*, 138–147.

Zhou, K., Doyle, J. C., & Glover, K. (1995). *Robust and optimal control*. Englewood Cliffs, NJ: Prentice Hall.

Zwick, U., & Paterson, M. (1996). The complexity of mean payoff games on graphs. *Theoretical Computer Science*, *158*, 343–359.

John N. Tsitsiklis was born in Thessaloniki, Greece, in 1958. He received the B.S. degree in Mathematics (1980), and the B.S. (1980), M.S. (1981) and Ph.D. (1984) degrees in Electrical Engineering, all from the Massachusetts Institute of Technology, Cambridge, Massachusetts.

During the academic year 1983–1984, he was an acting assistant professor of Electrical Engineering at Stanford University, Stanford, California. Since 1984, he has been with the Massachusetts Institute of Technology, where he is currently Professor of Electrical Engineering and Computer Science. He has served as acting co-director of the MIT Laboratory for Information and Decision Systems (Spring 1996 and 1997). He has also been a visitor with the Dept. of EECS at the University of California at Berkeley, and the Institute for Computer Science in Iraklion, Greece. His research interests are in the fields of systems, optimization, control, and operations research. He has written about 80 journal papers in these areas.

He is a coauthor of "Parallel and Distributed Computation: Numerical Methods" (with D. Bertsekas, 1989), "Neuro-Dynamic Programming" (with Dimitri Bertsekas, 1996), and "Introduction to Linear Optimization (with Dimitris Bertsimas, 1997). He has been a recipient of an IBM Faculty Development Award (1983), an NSF Presidential Young Investigator Award (1986), an Outstanding Paper Award by the IEEE Control Systems Society, the M.I.T. Edgerton Faculty Achievement Award (1989), the Bodossakis Foundation Prize (1995), the INFORMS/CSTS prize (1997), and is a Fellow of the IEEE (1999). He is an associate editor of *Applied Mathematics Letters* and has been an associate editor of the *IEEE Transactions on Automatic Control and Automatica.*

For a photograph of J.N. Tsitsiklis, see *Automatica* 35(3), March (1999).

Vincent D. Blondel received a first degree in Applied Mathematics from the University of Louvain (Louvain-la-Neuve, Belgium) in 1988, a Master Degree in Pure Mathematics from Imperial College (London) and a Ph.D. in 1992 from the University of Louvain.

In 1993 he was visiting scientist at Oxford University. During the academic year 1993–1994 he was Goran Gustafsson Fellow at the Department of Mathematics of the Royal Institute of Technology (Stockholm). In 1993–1994 he was a research fellow at the French National Research Center in Control and Computer Science INRIA (Paris). From 1994 to 1999 he was Associate Professor of Mathematics at the University of Liège (Belgium). He has recently moved to the University of Louvain where he is currently Professor of Applied Mathematics. He has been a visitor with the Australian National University, the Massachusetts Institute of Technology, the University of Paris VII and the Ecole Normale Supérieure in Lyon.

Dr. Blondel's major current research interests lie in several areas of mathematical systems theory and theoretical computer science. He is the author of "Simultaneous Stabilization of Linear Systems" (1994)

and is one of the editors of "Open Problems in Mathematical Systems Theory" (1999). He has been a recipient of a grant from the Trustees of the Mathematics Institute of Oxford University (1992), the Prize Agathon De Potter of the Belgian Royal Academy of Science (1993) and the Prize Paul Dubois of the Montefiore Institute (1993). He is the coordinator of a NATO collaborative grant with the Massachusetts Institute of Technology and is a partner in a European TRM network on discrete event systems. He is an associate editor of *Systems and Control Letters*, of the *European Journal of Control* and of *Mathematics of Control*, *Signals*, *and Systems*.

For a photograph of V.D. Blondel, see *Automatica* 35(3), March (1999).