JOURNAL OF COMPLEXITY 4, 1–11 (1988)

Probabilistic Satisfiability

GEORGE GEORGAKOPOULOS,* DIMITRIS KAVVADIAS,*
AND CHRISTOS H. PAPADIMITRIOU*,†

*Division of Computer Science, National Technical University of Athens, Athens, Greece; and †Departments of Computer Science and Operations Research, Stanford University, Stanford, California 94305

We study the following computational problem proposed by Nils Nilsson: Several clauses (disjunctions of literals) are given, and for each clause the probability that the clause is true is specified. We are asked whether these probabilities are consistent. They are if there is a probability distribution on the truth assignments such that the probability of each clause is the measure of its satisfying set of assignments. Since this problem is a generalization of the satisfiability problem for propositional calculus it is immediately NP-hard. We show that it is NP-complete even when there are at most two literals per clause (a case which is polynomial-time solvable in the non-probabilistic case). We use arguments from linear programming and graph theory to derive polynomial-time algorithms for some interesting special cases.

1. INTRODUCTION

Most approaches to decision-making in a complex, realistic environment involve the notion that a fact (or logical sentence) is true with a certain probability. These approaches are either informal, empirical, and non-rigorous as in expert systems (Nau, 1983; Prade, 1985), or rigorous in a way that diverges from probability theory (Zadeh, 1971). Recently, Nilsson proposed a rigorous framework for dealing with the probability of logical sentences (Nilsson, 1986). Repeating his example, the question we wish to formalize and answer is this: If the probability of proposition A is .8 and the probability of B is .3, can the probability of $A \rightarrow B$ be .6?[1] And what does this mean exactly?

[1] It follows from the appropriate formalism explained below that the answer is "no."

0885-064X/88 $3.00

In general, we have a set $S = \{C_1, \ldots, C_m\}$ of m logical sentences, with each sentence a *clause,* that is the disjunction of one or more literals. Each literal is one of n propositional variables $x_1, \ldots, x_n$, or its negation. We are also given m probabilities $0 \leq p_1, \ldots, p_m \leq 1$, one for each clause. These probabilities may be *inconsistent* (for example, if all probabilities are 1, and the conjunction of the clauses is unsatisfiable). To define consistency, Nilsson considers the event space consisting of all 2^n truth assignments. A *probability distribution* is a set of 2^n probabilities $\pi_j \geq 0$, $j = 1, \ldots, 2^n$ adding up to 1. We say that a probability distribution π satisfies the given clauses and probabilities if the following holds: *for each* $i = 1, \ldots, m$, *the sum of the* π_j's *over all truth assignments* a_j *that satisfy* C_i *equals* p_i. This can be written algebraically by defining an $m \times 2^n$ matrix A such that the i,jth element is 1 if a_j satisfies C_i, and zero otherwise. Then the requirement above can be rewritten as

$$A\pi = p$$

$$\pi \geq 0.$$

The problem of *probabilistic satisfiability* (PSAT) is to determine whether such a π exists.

In the linear program above, as in the rest of this paper, we have omitted the requirement that $\sum_{j=1}^{2^n} \pi_j = 1$, by assuming that we add to S the extra clause $C_0 =$ **true** (or $C_0 = (x_1 \vee \bar{x}_1)$), satisfied by all truth assignments, and with probability $p_0 = 1$. It is clear that we can omit from A columns that are all zero (truth assignments that falsify all clauses) or are repeated (as is done in Nilsson, 1986), but this does not in general result in substantial savings. *In this paper we study PSAT from the point of view of efficient algorithms and computational complexity.* Note that in practice we may wish to solve a different problem, such as finding upper and lower bounds on the admissible values for the probability of a clause, fixing the other probabilities (this is called *probabilistic entailment* in Nilsson, 1986). However, it is clear that these variants can be trivially reduced to PSAT, and thus the latter problem deserves some serious study.

Since PSAT is stated in terms of the existence of a vector which is exponentially long (and with entries of unspecified precision), its a priori complexity seems disappointingly high (for example, it is not immediate that the problem is in NP). However, it follows from the theory of linear programming (from the fact that if a solution to a linear program exists, then a basic feasible solution exists), or equivalently from Caratheodory's theorem, that PSAT is in NP. Also, since PSAT is a generalization of the satisfiability problem for propositional logic (called SAT (Garey and Johnson, 1976), which is the special case with all p_j's equal to 1), it is immediate that PSAT is NP-complete even in the special case in which all clauses

consist of three literals or less. In fact, by a reduction from graph coloring we show that PSAT remains NP-complete even in a special case in which the ordinary SAT problem is polynomially solvable, namely, when all clauses consist of one or two literals (this problem is called 2PSAT). In fact, our proof established the NP-completeness of another interesting formulation, in which we are given probabilities for certain variables, together with certain conditional probabilities. (This problem also has a linear programming formulation, via the quotient formula for conditional probabilities.)

On the positive side, we develop a general technique for attacking PSAT. We show that, under very general conditions, PSAT can be reduced to another important generalization of SAT, in which we are given weights for the clauses, and we are required to find the truth assignment which maximizes the total weight of all satisfied clauses; this problem is called *MAXSAT*. Our main observation is that *MAXSAT is a naturally dual problem to PSAT*. The reduction is via the ellipsoid algorithm for linear programming (see, for example, Khachian, 1979; Papadimitriou and Steiglitz, 1982). Naturally, MAXSAT is known to be NP-complete even in the case of two literals per clause (2MAXSAT) (Garey and Johnson, 1979), but still this result is of some value. The reason is that MAXSAT is a problem which is much more elementary and intuitive than PSAT, a more suitable target for solution by specialized algorithms and heuristic approaches. In this paper we describe one success along these lines: We give a polynomial-time algorithm for the special case of 2MAXSAT in which the graph with nodes as the literals and with edges standing for the clauses is *outerplanar*. Our algorithm employs (i) the ellipsoid algorithm, (ii) a reduction from 2PSAT to MAXCUT (the problem of finding the maximum cut in a weighted graph), and (iii) a polynomial-time algorithm (appropriately modified) for the MAXCUT problem for planar graphs (Orlova and Dorfman).

The rest of this paper is organized as follows: In Section 2 we show that PSAT is in NP, and that even 2PSAT is NP-complete. In Section 3 we show the reduction to MAXSAT, and in Section 4 the algorithm for the outerplanar case of 2PSAT. Finally, in Section 5 we discuss further open questions, as well as a promising algorithmic approach to 2PSAT using linear programming and heuristics for the maximum cut problem.

2. NP-Completeness

Given a set $S = \{C_1, \ldots, C_m\}$ of clauses on n propositional variables $x_1, \ldots, x_n$, let us define $A(S)$ to be the $m + 1 \times 2^n$ 0-1 matrix with 1's on the zeroth row, and for $i > 0$, $A_{ij} = 1$ iff the jth truth assignment on x_1,

. . . , x_n satisfies C_i. PSAT is the following computational problem: Given a set $S = \{C_1, \ldots, C_m\}$ of clauses and m probabilities $0 \leq p_i \leq 1$, $i = 1, \ldots, m$, is there a 2^n-vector $\pi \geq 0$ such that $A\pi = p$? (Here we took $p_0 = 1$.)

PROPOSITION 1. PSAT *is in* NP.

Proof. If a given instance of PSAT has a solution, then by linear programming theory (e.g., Papadimitriou and Steiglitz) there is a set of $m + 1$ columns of A (truth assignments) such that the resulting $m + 1 \times m + 1$ system has a positive solution. These columns can serve as a certificate. ■

COROLLARY 1. *If a satisfying probability distribution for an instance of* PSAT *exists, then there is one with at most* $m + 1$ *non-zero probabilities, and with entries rationals with total precision* $O(m^2)$.

Also, since the satisfiability problem for propositional logic is a special case of PSAT, we have:

COROLLARY 2. PSAT *is* NP-*complete*.

Interestingly, we can show that PSAT remains NP-complete in the special case, called 2PSAT, in which each clause involves at most two literals:

THEOREM 1. 2PSAT *is* NP-*complete*.

Proof. We shall reduce the problem of *graph 3-colorability* (Garey and Johnson, 1979) to PSAT.[2] In that problem we are given a graph $G = (V, E)$ and we are asked whether there is a function $c: V \to \{1, 2, 3\}$ such that for each edge $[u, v] \in E$ we have $c(u) \neq c(v)$. Given such a graph we construct an instance of 2PSAT as follows: We have three variables u_1, u_2, u_3 for each vertex $u \in V$; intuitively, u_j stands for the statement that vertex u is colored by color j. As for clauses, there are $6|V| + 3|E|$ of them. In particular, for each vertex u we have a set $S(u)$ of three clauses (u_1), (u_2), (u_3), each with probability $\frac{1}{3}$, and a set $T(u)$ of three more $((\bar{u}_1 \vee \bar{u}_2)$, $(\bar{u}_2 \vee \bar{u}_3)$, and $(\bar{u}_3 \vee \bar{u}_1))$ with probability 1. Finally, for each edge $[u, v] \in E$ we have a set $Q(e)$ of three clauses $((\bar{u}_1 \vee \bar{v}_1)$, $(\bar{u}_2 \vee \bar{v}_2)$, and $(\bar{u}_3 \vee \bar{v}_3))$ with probability also 1. This completes the construction of the instance of 2PSAT.

We claim that the instance of 2PSAT is satisfiable iff the given graph is 3-colorable. Suppose that the instance is satisfiable. Recall that $p(u_1 \vee u_2 \vee u_3) = p(u_1) + p(u_2) + p(u_3) + p(\bar{u}_1 \vee \bar{u}_2) + p(\bar{u}_2 \vee \bar{u}_3) + p(\bar{u}_3 \vee \bar{u}_1) - 3 + p(u_1 \wedge u_2 \wedge u_3)$. From the probabilities for $T(u)$ it follows that the last term

[2] The reduction is analogous to the one used in Honeyman *et al.* (1980) to show that the universal instance problem for relational databases is NP-complete.

is 0, and thus the right-hand side is 1. It follows that in any truth assignment with positive π_j (of which there must be at least one), exactly one of the a_j variables u_1, u_2, u_3, for each node u, is **true**. Suppose then that we define $c(u)$ to be k, whenever a_j assigns **true** to u_k. We claim that this is a legal coloring, that is, $c(u) \neq c(v)$ for each $[u, v] = e \in E$. However, this is immediate because of the clauses in $Q(e)$ with probability 1.

Conversely, if a legal 3-coloring c of G exists, define the truth assignment a in which u_k is **true** iff $c(u) = k$. Assign to this assignment a π of $\frac{1}{3}$. Consider now the colorings c' and c'' resulting by cyclically rearranging the colors of c, and the corresponding assignments a' and a'', also with probability $\frac{1}{3}$. All other assignments have zero probability. It is clear that the resulting vector π satisfies the given instance of PSAT. ■

In another variant of PSAT we are given certain *conditional probabilities* $p(x|y)$ for literals x and y; the problem is, again, to determine whether these are consistent. It is clear that this problem (called CONDSAT) can also be formulated as a linear program, using the formula $p(x|y) = p(x \wedge y)/p(y)$ and multiplying by the (non-zero) $p(y)$. Our proof of the theorem also establishes the following fact (by expressing $p(\bar{x} \vee \bar{y}) = 1$ as $p(x|y) = 0$):

COROLLARY 3. CONDSAT *is* NP-*complete.*

3. REDUCTION TO MAXSAT

PSAT is the problem of determining whether the linear program $A\pi = p$, $\pi \geq 0$ has a solution. The problem with this linear program is that it involves too many variables (an exponential number). However, linear programming duality (see, e.g., Papadimitriou and Steiglitz, 1982) shows the following important fact:

LEMMA 1. $A\pi = p, \pi \geq 0$ *has a feasible solution if and only if* $A^T x \leq 0$, $p'x \geq 1$ *has a feasible solution.*

The advantage of the dual formulation in the lemma is that the x sought is an $m + 1$-vector, subject to an exponential list of linear inequalities. This invites an attack via the ellipsoid alogrithm (Khachian, 1979; see Papadimitriou and Steiglitz, 1982, for an exposition). In that algorithm a feasible solution of a linear program is found by a sequence of iterations. In the beginning of the ith iteration we are at an infeasible point x^i. We then choose one of the inequalities violated by x^i, and based on it we compute another point x^{i+1}. After enough iterations, either a feasible point has been reached, or we know that none exists. The advantage of this algorithm is that *an explicit list of the inequalities is not necessary.* As

was pointed out by Karp and Papadimitriou (1982) among others, all we need is a procedure which, given a point x, returns an inequality of the linear program violated by x. Such a procedure is called a *generator of violated inequalities.*

LEMMA 2. *(Karp and Papadimitriou, 1982). A feasible solution of an $M \times N$ linear program $Ax \leq b$ with integer coefficients with maximum subdeterminant 2^L can be found in $O(N^3L)$ iterations, each involving a call of the generator of violated inequalities for the program, and $O(N^2L)$ more operations.*

In our case, since the dual of the linear program in PSAT is $2^n \times m$ and has $0 - 1$ coefficients, the number of iterations is $O(m^2 \log m)$, and the number of operations per iteration $O(m^3 \log n)$.

The important question is how can we develop a generator of violated inequalities for the dual of PSAT. One such inequality is $p'x \geq 1$, which can be easily treated separately. However, there are exponentially many other inequalities in $A^Tx \leq 0$, one for each truth assignment of the n variables. Such an inequality is violated by an $m + 1$-vector x iff *the sum of the x_i's corresponding to the clauses satisfied by the truth assignment is larger than $-x_0$*. That is, in order to find whether a violated inequality exists, we must do the following: We consider the components of x as weights (possibly negative) for the clauses. We wish to determine whether there is a truth assignment which satisfies a set of clauses with total weight greater than x_0. However, this is an instance of a well-known problem, called MAXSAT (Garey *et al.*, 1976). In MAXSAT we are given m clauses on n variables, for each clause C_i an integer weight w_i, and a goal W; we are asked whether there is a truth assignment a of the variables so that the sum of the weights of all clauses satisfied by a is at least W. Hence we have shown the following result:

THEOREM 2. *An instance of* PSAT *with m clauses and n variables can be solved in $O(m^2 \log m)$ iterations, each of which involves solving (a) an instance of* MAXSAT *on the same clauses and weights integers with $O(m)$ bits, and (b) $O(m^3 \log m)$ more operations.*

COROLLARY 4. PSAT *restricted to some class of clauses is polynomial-time reducible to* MAXSAT *on the same class of clauses.*

4. OUTERPLANAR 2PSAT

Consider a graph with weights on its edges. A *cut* in this graph is a partition of its nodes in two subsets S and T. The *weight* of the cut is the sum of the weights of the edges which have endpoints in both S and T.

MAXCUT is the problem of determining, given a weighted graph and an integer B, whether there is a cut with weight B or more.

2MAXSAT (maximum weight satisfiability with at most two literals per clause) can be polynomial-time reduced to MAXCUT.[3] The basic observation is that any triangle with unit weights on the edges is going to contribute to the maximum cut either zero (if all three nodes are in the same set) or two (in all other situations). Given an instance of 2MAXSAT (a set S of clauses $C_1, \ldots, C_m$ with at most two literals each, involving the variables $x_1 \ldots x_n$, weights $w_1, \ldots, w_m$ for the clauses, and a goal W), we construct a weighted graph $G = (V, E, c)$ and bound B as follows: The nodes of G are $\{x_i, \bar{x}_i : i = 1, \ldots, n\} \cup \{F\}$, that is, all literals and the node F (for "false"). As for edges, G has the edges $\{[x_i, \bar{x}_i] : i = 1, \ldots, n\}$, each with a large weight M (say, M is $4n^2$ times the largest absolute weight of a clause). Intuitively, this forces opposite literals to be on the opposite side of any sensible cut. Then, the side that contains F is going to be the "false" side. If (α) is a (one-literal) clause with weight w, we add to G an edge $[F, \alpha]$ with weight w. Finally, if $(\alpha \vee \beta)$ is a two-literal clause with weight w, we add to G the triangle $[F, \alpha]$, $[F, \beta]$, $[\alpha, \beta]$, all edges with weight $w/2$. We choose the bound B to be $nM + W$, where W is the desired weight in the instance of MAXSAT.

We claim that the constructed graph has a cut of weight B or more iff there is a truth assignment that satisfies a set of clauses of total weight W or more. Suppose that such a cut S, T exists, where $F \in S$. Since the bound B is attained, all n edges of weight M must contribute to the cut. The remaining weight of W is contributed by edges corresponding to clauses. Each clause, not all literals of which belong to S, contributes its weight to the total weight of the cut. Let us consider the truth assignment in which all nodes in S are **false**. It is clear that this truth assignment satisfies all clauses that contribute their weight to B. It follows that this assignment satisfies clauses with total weight at least $B - nM = W$. The converse is now trivial. Hence we have shown:

LEMMA 3. 2MAXSAT *is polynomially transformable to* MAXCUT.

MAXCUT is known to be NP-complete (Garey and Johnson, 1979). Another important fact about MAXCUT was first pointed out in (Orlova and Dorfman). They showed that the problem can be solved in polynomial time for *planar* graphs. We need here a slightly stronger version to cover the case of negative weights.

[3] This was the reduction first used in (Garey *et al.*, 1976) to show that the MAXCUT problem is NP-complete; it is a pleasure to employ in the pursuit of a more positive cause a reduction originally used to prove NP-completeness.

LEMMA 4. MAXCUT *for planar graphs* $G = (V, E, w)$ *with positive and negative weights can be solved in* $O(|V|^4)$ *time.*

Sketch. We consider the *dual* graph $G^D = (V^D, E, w)$ of G (the edges of G and G^D are in a natural one-to-one correspondence, hence the weights remain the same). Since nodes of G are faces of G^D, sets of nodes of G correspond to sets of faces of G^D. Thus, any cut of G corresponds in a natural way to a *set of edge–disjoint cycles* in G^D, or, equivalently, to a subgraph of G^D with even degree at all nodes. Thus, the task of finding the maximum cut in G is identical to finding the Eulerial subgraph of G^D which has the largest total weight. In graphs with positive weights this is equivalent to finding a minimum-weight complete matching of the odd-degree nodes (the "Chinese postman problem"), but negative weights complicate the task a little.

This latter problem can be solved in *any* weighted graph by solving the *b-matching problem* (see, e.g., Papadimitriou and Steiglitz) in a related graph. In the b-matching problem we are given a weighted graph and, for each node v, its *degree requirement* $b(v)$. In fact, the graph may have multiple edges and self-loops, where inclusion of a self-loop to the subgraph adds two to the degree of the node involved. We are asked to find the subgraph with the given degrees which has the largest possible total weight. This problem can be solved by matching techniques in $O([\sum_V b(v) \deg(v)]^2)$ time. (Note: Although the b-matching problem is usually posed as a minimization problem, this is hardly a difficulty, since any feasible subgraph has a fixed number of edges (the sum of all b's divided by two), and thus we can change the weights by subtracting from a suitably large number; for the same reason negative weights are no problem.)

Given a weighted graph $G = (V, E, w)$ in which we are to find the heaviest Eulerian subgraph, we construct a new one $G' = (V, E', w')$ and a degree requirement b as follows: For each node v of degree d we add $\lfloor d/2 \rfloor$ self-loops $[v, v]$ of weight zero. The degree requirement for the node is $b(v) = 2\lfloor d/2 \rfloor$; all weights are kept the same. It is quite easy to see that there is a one-to-one, weight-preserving correspondence between Eulerian subgraphs of G and b-matchings of G' (by adding an appropriate number of zero-weight loops for each node). It follows that the problem of finding the maximum-weight Eulerian subgraph of a graph can be solved in $O([\sum_V \deg^2(v)]^2)$ time, which is $O(|V|^4)$ for planar graphs. ■

A graph is *outerplanar* if it is planar, and furthermore it can be embedded in the plane so that all of its nodes lie on the same face. Suppose that we are given a set S of clauses, with at most two literals per clause. We define *the graph of* S, $G(S)$, as follows: $G(S)$ has as nodes the literals $\{x_i, \bar{x}_i : i = 1, \ldots, n\}$ of the clauses, and two kinds of edges: First, the edges $\{[x_i, \bar{x}_i] : i = 1, \ldots, n\}$; and finally, the edges of $[\alpha, \beta]$, where α and β

are literals appearing in the same clause. We say that a set S of clauses is *outerplanar* if $G(S)$ is outerplanar.

THEOREM 3. 2PSAT *on outerplanar sets of clauses can be solved in polynomial time.*

Proof. If $G(S)$ is outerplanar, then the graph constructed in the reduction from 2MAXSAT to MAXCUT in Lemma 3 is planar (we simply place the new node F on the Hamiltonian face and connect it with all literals). Therefore, the corresponding instance of MAXCUT can be solved in $O(n^4)$ time by Lemma 4. Finally, the ellipsoid algorithm which reduces 2PSAT to 2MAXSAT (Theorem 2) runs in $O(m^2 \log m(n^4 + m^3 \log m)) = O(n^6 \log n)$ time (since for outerplanar instances m is linear in n). ■

An interesting consequence of this result is that 2PSAT is polynomial also in the case in which there are no "circular connections" between the Boolean variables. Formally, define the *summary graph* of a set S of clauses to have the variables as nodes, and an edge between x and y if x (or its negation) appear in the same clause as y (or its negation). We say that an instance of PSAT is *acyclic* if its summary graph is a tree.

COROLLARY 5. 2PSAT *for acylic sets of clauses can be solved in polynomial time.*

Proof. The key observation is that any acyclic set of clauses is outerplanar, or can be made so easily. If a variable x has degree $d > 1$ in the summary graph, we replace it with the variables $x_1, \ldots, x_d$, with the additional clauses $(\bar{x}_1 \vee x_2), (\bar{x}_2 \vee x_3), \ldots, (\bar{x}_d \vee x_1)$, all with probability one, and from now on variable x_j is going to appear in the same clause as the jth neighbor of x in the summary graph. The graph of the resulting set of clauses has the same tree structure as the summary graph, except that each internal node of the tree is replaced by a cycle of length $2d$, and each edge (pair of variables appearing in a clause together) is replaced by edges joining all pairs of literals that appear together in a clause. However, we can assume that there are going to be at most three such appearances. The reason is the following: If all four possible clauses involving the two variables appear, then the probability of the fourth is simply 3 minus the sum of the others, and thus the fourth clause can be omitted. The three edges which replace the tree edge can then be easily drawn so that the resulting graph is outerplanar. ■

5. DISCUSSION

Early results indicate that our approach can be the basis of truly practical algorithms for a wide range of 2PSAT instances. The idea is to apply

simplex to the linear program. Naturally, we cannot store and manipulate the whole tableau. We can store, however, the $m \times m$ basis, and bring each time in the basis the column with most negative marginal cost (this technique is called *column generation*). Interestingly, finding the column with a negative marginal cost is again an instance of MAXSAT. For outerplanar or acyclic problems we can do this in polynomial time. The algorithm is no longer guaranteed to be polynomial, but our early computational experience shows considerable promise. We can even attack more general instances of PSAT, by using heuristics for MAXSAT to prove inconsistency. Our computational methods and experience will be reported in (Kavvadias, 1987).

There are many other open questions: Under what conditions is matrix A totally unimodular? Can we solve *planar* (not outerplanar) 2PSAT in polynomial time? Can we use the same approach to solve CONDSAT, or even mixed sets of clauses and conditionals? Can we apply this approach to the problem of inferring an n-fold distribution from a set of marginals? Interestingly, the acyclic case of this problem is known to be more well-behaved in that problem as well, and a linear programming formulation is still possible. The case in which the random variables are binary and the marginal distributions are defined on pairs of variables is a special case of 2PSAT, not known to be NP-complete.

ACKNOWLEDGMENTS

We thank Foto Afrati, Nils Nilsson, and John Tsitsiklis for helpful discussions on this and related problems. This research was partially supported by the National Science Foundation.

REFERENCES

GAREY, M. R., AND JOHNSON, D. S. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, New York.

GAREY, M. R., JOHNSON, D. S., AND STOCKMEYER, L. J. (1976), Some simplified NP-complete problems, *Theor. Comput. Sci.* **1,** 237–267.

GRÖTSCHEL, M., LOVÀSZ, L., AND SCHRIJVER, S. (1980), "The Ellipsoid Method and Its Consequences in Combinatorial Optimization," Technical Report, University of Bonn.

HONEYMAN, P., LADNER, R. E., AND YANNAKAKIS, M. (1980), Testing the universal instance assumption, *Inf. Process. Lett.* **10**(1), 14–19.

KARP, R. M., AND PAPADIMITRIOU, C. H. (1982), On linear characterization of combinatorial optimization problems, *SIAM J. Comput.* **11**(4), 620–632.

KAVVADIAS, D. (1987), Ph.D. dissertation in preparation, University of Patras.

KHACHIAN, L. G. (1979), A polynomial algorithm for linear programming, *Dokl. Akad. Nauk. SSSR* **244:5,** 1093–1096.

NAU, D. S. (1983), Expert computer systems, *Comput. Surv.*, 63–85.

NILSSON, N. (1986), Probabilistic logic, *Artif. Intell.*

PAPADIMITRIOU, C. H., AND STEIGLITZ, K. (1982), "Combinatorial Optimization: Algorithms and Complexity," Prentice–Hall, Englewood Cliffs, N.J.

PRADE, H. (1985), A computational approach to approximate and plausible reasoning with applications to expert systems, *IEEE Trans. PAMI* **7**(3).

TSITSIKLIS, J. N. (1986), Personal communication, February.

ZADEH, L. A. (1971), Fuzzy sets, *Inform. Contr.* **8,** 338–353.