

Dynamic MAP Calculations for Abduction

Eugene Charniak and Eugene Santos Jr.

Department of Computer Science

Brown University

Providence RI 02912

ec@cs.brown.edu and esj@cs.brown.edu

Abstract

We present a dynamic algorithm for MAP calculations. The algorithm is based upon Santos's technique (Santos 1991b) of transforming minimal-cost-proof problems into linear-programming problems. The algorithm is dynamic in the sense that it is able to use the results from an earlier, near by, problem to lessen its search time. Results are presented which clearly suggest that this is a powerful technique for dynamic abduction problems.

Introduction

Many AI problems, such as language understanding, vision, medical diagnosis, map learning etc. can be characterized as abduction — reasoning from effects to causes. Typically, when we try to characterize the computational problem involved in abduction, we distinguish between (a) finding possible causes and (b) selecting the one most likely to be correct. In this paper we concentrate on the second of these issues. We further limit ourselves to probabilistic methods. It seems clear that “the most likely to be correct” is that which is most probable. The arguments against the use of probability therefore usually concentrate on the difficulty of computing the relevant probabilities. However, recent advances in the application of probability theory to AI problems, from Markov Random Fields (Geman & Geman 1984) to Bayesian Networks (Pearl 1988) have made such arguments harder to maintain.

The problems of our first sentence are not just abduction problems, they are *dynamic* abduction problems. That is, we are not confronted by a single problem, but rather by a sequence of closely related problems. In medical diagnosis the problem changes as new evidence comes in. In vision, the picture changes slightly over time. In story understanding, we get further portions of the story.

Of course we can reduce the dynamic case to the static. Each scene, medical problem, or story comprehension task sets up a new probabilistic problem which the program tackles afresh. For example, in our

previous work on the Wimp3 story understanding program (Charniak & Goldman 1989; Charniak & Goldman 1991; Goldman & Charniak 1990) story decisions are translated into Bayesian Networks. The network is extended on a word-by-word basis. As one would expect, most of the network at word n was there for word $n - 1$. Yet the algorithm we used for calculating the probabilities at n made no use of the values found at $n - 1$. This seemed wasteful, but we knew of no way to do better. This paper corrects this situation.

Previous Research

Roughly speaking, probabilistic algorithms can be divided up into exact algorithms and ones which are only likely to return a number within some δ of the answer. Considering exact algorithms, we are not aware of any which are dynamic. The story is different for the approximation schemes since most are stochastic in nature and have the property that if we can start with a better guess of the correct answer they will converge to that answer more quickly. (See for example (Shachter & Peot 1989; Shwe & Cooper 1990) .) Thus such algorithms are inherently dynamic in the sense that the results from the previous computation can be used to improve the performance on the next one. However, even for such algorithms we are unaware of any study of their dynamic properties.

The particular (exact) algorithm we propose here did not arise from a search for a dynamic algorithm, but rather from an effort to improve the speed of probabilistic calculations in Wimp3. One scheme we considered was the use of Hobbs and Stickel's (Hobbs *et al.* 1988) work on minimal-cost proofs for finding the best explanations for text. Their idea was to find a set of facts which would allow the program to prove the contents of the textual input. Since there would be many possible sets of explanations, they assigned costs to assumptions. The cost of a proof was the sum of the cost of the assumptions it used. (Actually, we are outlining “cost-based abduction” as defined in (Charniak & Shimony 1990) which is slightly simpler than the scheme used by Hobbs and Stickel. However, the two are sufficiently close that we will not bother to distinguish

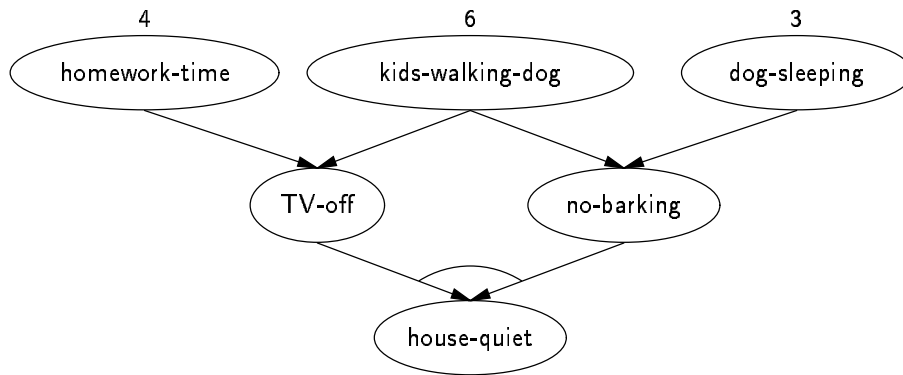


Figure 1: An and-or dag showing several proofs of house-quiet

between them.)

For example, see Figure 1. There we want the minimal-cost proof of the proposition house-quiet (my house is quiet). This is an “and-node” so proving it requires proving both no-barking and TV-off. Figure 1 shows several possible proofs of these using different assumptions. One possible set of assumptions would be homework-time, dog-sleeping. Given the costs in the figure, the minimal cost proof would be simply assuming kids-walking-dog at a cost of 6.

We were interested in minimal-cost proofs since their math seems a lot simpler than typical Bayesian Network calculations. Unfortunately, the costs were ad-hoc, and it was not clear what the algorithm was really computing. Charniak and Shimony (Charniak & Shimony 1990) fixed these problems by showing how the costs could be interpreted as negative log probabilities, and that under this interpretation minimal-cost proofs produced MAP assignments (subject to some minor qualifications). (A MAP, Maximum A-Posteriori, assignment is the assignment of values to the random variables which is most probable given the evidence.) A subsequent paper (Shimony & Charniak 1990), showed that any MAP problem for Bayesian Networks could be recast as a minimal-cost-proof problem. In the rest of this paper we will talk about algorithms for minimal-cost proofs and depend on the result of (Shimony & Charniak 1990) to relate them to MAP assignments.

Unfortunately, the best-first search scheme for finding minimal-cost proofs proposed by Charniak and Shimony (and modeled after the Hobbs and Stickel’s work) did not seem to be as efficient as the best algorithms for Bayesian Network evaluation (Jensen *et al.* 1989; Lauritzen & Spiegelhalter 1988). Even improved admissible cost estimations (Charniak & Husain 1991) did not raise the MAP performance to the levels obtained for Bayesian Network evaluation.

Recently, Santos (Santos 1991a; Santos 1991c; Santos 1991b) presented a new technique different from the best-first search schemes. It showed how minimal-cost

proof problems could be translated into 0-1 programming problems and then solved using simplex combined with branch and bound techniques when simplex did not return a 0-1 solution. Santos reports that simplex returns 0-1 solutions on 95% of the random waodags he tried and 60-70% of those generated by the Wimp 3 natural language understanding program. This plus the fact that branch and bound quickly found 0-1 solutions in the remaining cases showed that this new approach outperformed the best-first heuristics. It actually exhibited an expected polynomial run-time growth rate where as the heuristics were exponential. This approach immediately suggested a dynamic MAP calculation algorithm.

Waodags as Linear Constraints

Santos, following (Charniak & Shimony 1990) formalized the minimal-cost proof problem as one of finding a minimal-cost labeling of a weighted-and-or-dag (waodag). Informally a waodag is a proof tree (actually a dag since a node can appear in several proofs, or in the same proof several times). We have already seen an example in Figure 1. Formally it is a 4-tuple $\langle G, c, r, S \rangle$, where

1. G is a connected directed acyclic graph, $G = (V, E)$,
2. c is a function from V to the non-negative reals, called the cost function (this is simplified in that we only allow assuming things true to have a cost),
3. r is a function from V to $\{\text{and, or}\}$ which labels each node as an and-node or an or-node,
4. S is a set of nodes with outdegree 0, called the evidence.

The problem is to find an assignment of true and false to every node, obeying the obvious rules for and-nodes and or-nodes, such that the evidence is all labeled true, and the total cost of the assignment is minimal.

In the transformation to a linear programming problem a node n becomes a variable x with values restricted to 0/1 (false/true). An and-node n which was

true iff its parents $n_0 \dots n_i$ were true would have the linear constraints

$$\begin{aligned} \text{AND1} \quad & x \leq x_0, \dots, x \leq x_i \\ \text{AND2} \quad & x \geq x_0 + \dots + x_i - i + 1. \end{aligned}$$

An or-node n which is true iff at least one of its parents $n_0 \dots n_i$ are true would have the linear constraints

$$\begin{aligned} \text{OR1} \quad & x \leq x_0 + \dots + x_i \\ \text{OR2} \quad & x \geq x_0, \dots, x \geq x_i. \end{aligned}$$

The constraints AND1 and OR1 can be thought of as “bottom up” in that if one thinks of the evidence as at the bottom of the dag, then these rules pass true values up the dag to the assumptions. The constraints AND2 and OR2 are “top down.” In our implementation we only bothered with the bottom-up rules.

The costs of assumptions are reflected in the objective function of the linear system. As we are making the simplifying (but easily relaxed) assumption that we assign costs only to assuming statements true, modeling the costs $c_0 \dots c_i$ for nodes $n_0 \dots n_i$ is accomplished with the objective function

$$c_0 x_0 + \dots + c_i x_i$$

Santos then shows that a zero-one solution to the linear equations which minimizes the objective function must be the minimal-cost proof (and thus the MAP solution for the corresponding Bayesian Network.)

The Algorithm

Simplex is, of course, a search algorithm. It searches the boundary points of a convex polygon in a high dimensional space, looking for a point which gives the optimal solution to the linear inequalities. The search starts with an easily obtained, but not very good, solution to the problem. It is well known that simplex works better if the initial solution is closer to optimal. When we say “better” we mean that the algorithm requires fewer “pivots” – the traditional measure of time for simplex. To those not familiar with the scheme, the number of pivots corresponds to the number of solutions tested before the optimal is found.

In what follows we will assume that the following are primitive operations, and that it is possible to take an existing solution, perform any combination of them, and still use the existing solution as a start in finding the next solution. This is, in fact, standard in the linear-programming literature (see (Hillier & Lieberman 1967)).

- L1 Add a new variable (with all zero coefficients) to the problem.
- L2 Add a new equation to the problem.
- L3 Change the coefficient of a variable in an equation.
- L4 Change the coefficient of a variable in the objective function.

Moving down a level, we need to talk about operations on the woadags. In particular we need to do the following:

- W1 add a new node to the dag,
- W2 add an arrow between nodes,
- W3 set the value of a node,
- W4 remove a node (and arcs from it).

For example, when we get a new word of the text (e.g., “bank”) in a story comprehension program we need to add a new or-node corresponding to it (W1) and, since this is not a hypothesis, but rather a known fact, set the value of the node to true (W3). To add an explanation for this word, say that it was used because the author wanted to refer to a savings institution, we would add a second or-node corresponding to the “savings institution” hypothesis, add a coefficient corresponding to its cost to the objective function (L4), and add an arc between it and the node for the word (W2), possibly with an intermediate newly minted and-node.

Next we relate W1-W4 to L1-L4. W1-W3 are fairly simple.

W1 Adding a new node n_1 to the dag is accomplished by adding a new variable x_1 to the linear programming problem. If it has a cost c_1 , change the coefficient of x_1 in the objective function from 0 to c_1 (L4).

W2 Adding an arrow from node n_1 to n_2 is accomplished in one of three ways (we only deal with the bottom-up equations):

- If n_2 is an and-node, add the new equation $x_2 \leq x_1$ (L2)
- If n_2 is an or node with the equation $x_2 \leq x_3 + \dots + x_i$ already in the problem, change the coefficient of x_1 in this equation from zero to one (L3)
- Else, add a new equation $x_2 \leq x_1$ (L2).

W3 Setting the value of a node n_1 to true (false) is accomplished by adding the equation $n_1 = 1(0)$ to the linear programming problem (L2). (In cases where the node is about to be introduced, we can optimize by not explicitly adding its variable to any equations, and instead substituting its value into the equations for parents nodes.)

The only complicated modification to Woadags is W4, removing a node (and arcs from it). We need this so, as the text goes along, it will be possible to set values of nodes to be definitely true or false, thus removing them from further probabilistic calculations (except should they serve as evidence for other facts). If all we wanted to do was set the value, we could use the method in (W3), adding an equation setting the value. But this is not enough. We want to remove the variable from all equations as well so as to reduce the size of the simplex tableau and thus decrease the pivot time. We will assume in what follows that the variable in question is to be assigned the value it has in the current MAP. If this is not the case then a new equation must be added.

If the variable x_j to be removed is a non-basic variable the problem is easy. We just use technique L3 to modify all of x_i 's coefficients to be zero. We can then remove its column from the tableau. This does not work, however, if the variable is basic. Then there will be a row, say r_i which will look like this:

$$c_{i,1}, c_{i,2}, \dots, c_{i,j-1}, 1, c_{i,j+1}, \dots, c_{i,k} = b_r.$$

Changing the coefficient $c_{i,j}$ which is currently 1, to 0 will leave this equation with no basic variable, something not allowed. However, if any of the other coefficients $c_{i,1} \dots c_{i,k} \geq 0$ then we can pivot on that variable and make it basic. This leaves the case when none of the other variables are positive. The solution depends on the fact that we are setting the value of x_j to its current value in the MAP. This will be, of course, b_r . Thus we should, and will, subtract b_r from both sides of the equation. On the left-hand-side this is done by making $c_{i,j} = 0$. On the right we actually subtract $b_r - b_r$ to get $b'_r = 0$. Note now that since $b'_r = 0$ we can multiply both sides of the equation by -1 when all of the other coefficients are negative, thus making them positive. b'_r is still zero so the equation remains in standard form, and we now have a non-zero coefficient to pivot on. Finally, if there are no non-zero coefficients other than $c_{i,j}$ then the equation can be deleted, along with $x_{i,j}$.

Results

The algorithm described in the previous section was run on a set of probabilistic problems generated by the Wimp 3 natural-language understanding program. As we have already noted, Wimp 3 works by translating problems in language understanding into Bayesian Networks. Ambiguities such as alternative references for a pronoun become separate nodes in the network. The alternative with the highest probability given the evidence is selected as the "correct" interpretation. Wimp 3 used a version of Lauritzen and Spiegelhalter's algorithm (Lauritzen & Spiegelhalter 1988) as improved by Jensen (Jensen *et al.* 1989), to compute the probabilities. Wimp 3 generated 218 networks ranging in size from 3 to 87 nodes. As befits an inherently exponential algorithm, we have plotted the log of evaluation time against network size. (All times are compiled Lisp run on a Sun Sparc 1.) See Figure 2. A least squares fit of the equation $time = B \cdot 10^{A \cdot |dag|}$ gives $A = .03$ and $B = .275$. The total evaluation time for all of the networks was 575 seconds, to be compared with a total of 1235 seconds to run the examples (or 46.5% of the total running time). This illustrates that network evaluation time was the major component of the time taken to process the examples.

Wimp 3 was then modified slightly to use the dynamic MAP algorithm described in this paper. While there is a general algorithm for turning Bayesian networks into cost-based abduction problems (Shimony & Charniak 1990), it can be rather expensive in terms of

the number of nodes created. Instead we made use of the fact that the networks created by Wimp were already pretty much and-or dags, and fixed the few remaining places on a case-by-case basis. A graph of the resulting programs running time (or the log thereof) vs. dag size is given in Figure 3. The dag's sizes reported are, in general twice those for the Bayesian Networks. The major reason is that the dag's included explicit and-nodes while these were absorbed into the distributions in the networks. The total time to process the dags was 73 seconds. Thus the time spent on probabilistic calculations was decreased by a factor of 7.9.

Running time, of course, is not a very good measure of performance since so many factors are blended into the final result. Unfortunately, the schemes operate on such different principles that we have not been able to think of a more implementation independent measure for comparing them. It is, nevertheless, our belief that these numbers are indicative of the relative efficiency of the two schemes on the task at hand.

As should be clear from Figure 3, the connection between the size of the dag and the running time is much more tenuous for the linear technique. This suggested looking at the standard measure for the performance of the simplex method — number of pivots. Figure 4 shows the number of pivots vs DAG size. Note that we did *not* graph dag size against the *log* of the number of pivots. As should be clear, the number of pivots grows very slowly with the size of the dag, and the correlation is not very good. It certainly does not seem to be growing exponentially. If one does a least squares fit of the linear relation $pivots = A \cdot |dag| + B$ one finds the best fit $A = .091$ and $B = 7.1$. Since the time per pivot for simplex is order n^2 we have seemingly reduced the "expected time" complexity of our problem to low polynomial time.

Conclusion

While the results of the previous section are impressive to our eyes, it should be emphasized that there are limitations to this approach. The most obvious is that even an n^2 algorithm is only feasible for smaller n . Our algorithm performs acceptably for our networks of up to 175 nodes. It is doubtful that it would work for problems in pixel-level vision, where there are, say 10^6 pixels, each a random variable. Furthermore, if the problem is not amenable to a cost-based abduction formulation it is unlikely that the general transformation from Bayesian Networks to woadags will produce acceptably small dags.

But this still leaves a lot of problems for which this technique *is* applicable. In several previous papers (Charniak 1991; Charniak & Goldman 1991) on the use of Bayesian Networks for story understanding we have emphasized that the problem of network evaluation was the major stumbling block in the use of these networks. The results of the previous section would indicate that this is no longer the case. Putting the

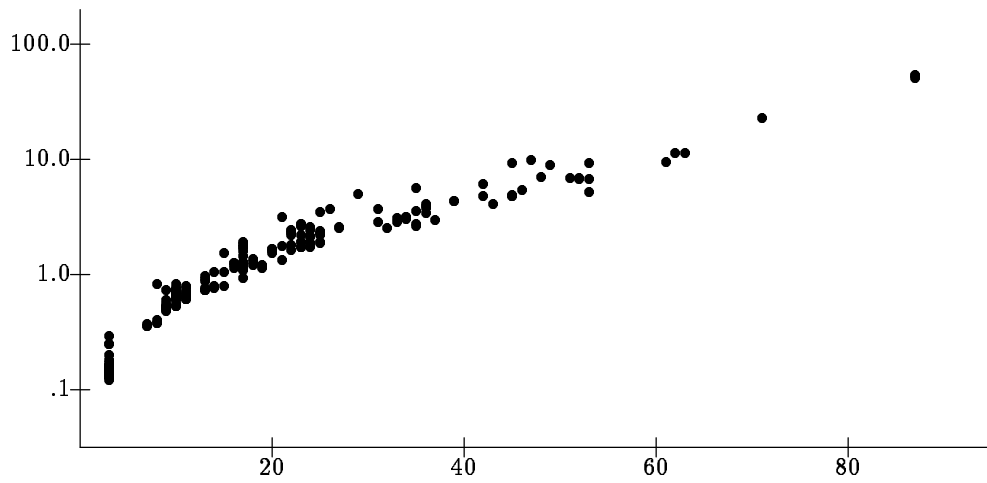


Figure 2: Bayesian-Network-Evaluation Time (in Sec.) vs. Number of Nodes

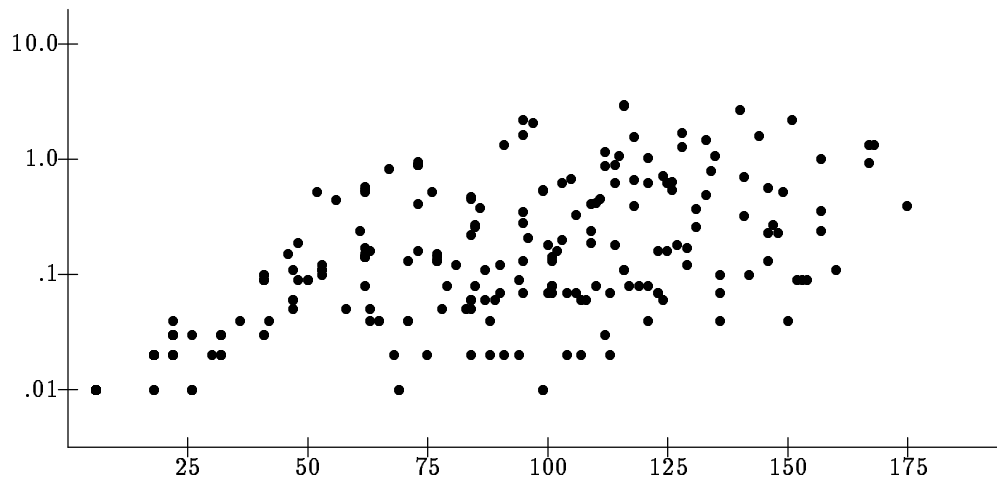


Figure 3: Linear Cost-Based Abduction Time (in Sec.) vs. Number of Nodes

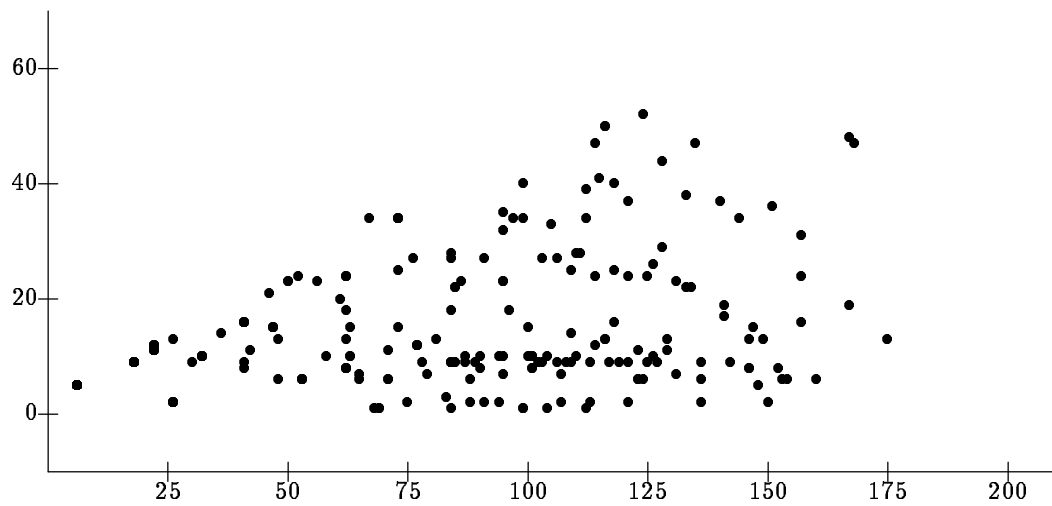


Figure 4: Number of Pivots vs. Number of Nodes

data in perspective, we are spending about .3 seconds per word on probabilistic calculations. Furthermore, we are using a crude version of simplex which one of the authors wrote. It includes none of the more sophisticated work which has been done to handle larger linear programming problems. It seems clear that a better simplex could handle order of magnitude larger problems in the same time (or less, with the very much more powerful machines already on the market.) Thus network evaluation is no longer a limiting factor. Now the limiting factor is for us, as it is for pretty much the rest of "traditional" AI, knowledge representation.

Acknowledgements

This research was supported in part by NSF contract IRI-8911122 and ONR contract N0014-91-J-1202.

References

- Charniak, E. and Goldman, R. 1989. A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding. In *Proceedings of the IJCAI Conference*.
- Charniak, E. and Goldman, R. 1991. A probabilistic model of plan recognition. In *Proceedings of the AAAI Conference*.
- Charniak, E. and Husain, S. 1991. A new admissible heuristic for minimal-cost proofs. In *Proceedings of the AAAI Conference*.
- Charniak, E. and Shimony, S. E. 1990. Probabilistic semantics for cost based abduction. In *Proceedings of the AAAI Conference*. 106-111.
- Charniak, E. 1991. Bayesian networks without tears. *AI Magazine* 12(4):50-63.
- Geman, S. and Geman, D. 1984. Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6:721-41.
- Goldman, R. and Charniak, E. 1990. Dynamic construction of belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
- Hillier, F. S. and Lieberman, G. J. 1967. *Introduction to Operations Research*. Holden-Day, Inc.
- Hobbs, J. R.; Stickel, M.; Martin, P.; and Edwards, D. 1988. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*.
- Jensen, F. V.; Lauritzen, S. L.; and Olesen, K. G. 1989. Bayesian updating in recursive graphical models by local computations. Technical Report Report R 89-15, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg, Denmark.
- Lauritzen, S. L. and Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their applications to expert systems. *J. Royal Statistical Society* 50(2):157-224.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.
- Santos, E. Jr. 1991a. Cost-based abduction and linear constraint satisfaction. Technical Report CS-91-13, Department of Computer Science, Brown University.
- Santos, E. Jr. 1991b. A linear constraint satisfaction approach to cost-based abduction. Submitted to *Artificial Intelligence Journal*.
- Santos, E. Jr. 1991c. On the generation of alternative explanations with implications for belief revision. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
- Shachter, R. D. and Peot, M. A. 1989. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
- Shimony, S. E. and Charniak, E. 1990. A new algorithm for finding map assignments to belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.
- Shwe, M. and Cooper, G. 1990. An empirical analysis of likelihood-weighting simulation on a large multiply-connected belief network. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.