A Bayesian-Network Approach to Lexical Disambiguation

LEILA M. R. EIZIRIK
VALMIR C. BARBOSA
SUELI B. T. MENDES

Universidade Federal do Rio de Janeiro, Brazil

Lexical ambiguity can be syntactic if it involves more than one grammatical category for a single word, or semantic if more than one meaning can be associated with a word. In this article we discuss the application of a Bayesian-network model in the resolution of lexical ambiguities of both types. The network we propose comprises a parsing subnetwork, which can be constructed automatically for any context-free grammar, and a subnetwork for semantic analysis, which, in the spirit of Fillmore’s (1968) case grammars, seeks to fulfill the required cases of all candidates for verb of the sentence. Solving for the highest joint probability of the variables conditioned upon the evidences to the network yields the most likely candidate with its meaning, along with its cases and respective meanings. This is achieved by fixing the values of all evidence nodes concurrently, and then performing a stochastic simulation in which the remaining nodes are updated probabilistically with a high degree of parallelism. The process of disambiguation is directed neither by the syntax nor the semantics, but rather by the interrelation between the two subnetworks. The use of a Bayesian-network model allows us to express this interrelation between the two subnetworks and among their constituents in a rather direct and rigorous way that, in connection with the convergence properties of the stochastic simulation, reveals a very robust model.

1. INTRODUCTION

The resolution of lexical ambiguity occupies a position of great relevance in the processing of natural languages, and has received considerable attention in the literature (Allen, 1987; Charniak & Goldman, 1989; Cottrell, 1985; Fanty, 1985; Goldman & Charniak, 1992; McClelland & Kawamoto, 1986;...
Selman, 1985; Small, Cottrell, & Tanenhaus, 1988; Waltz & Pollack, 1985; Winograd, 1983). A well-accepted taxonomy of lexical ambiguity classifies it into *syntactic ambiguity* and *semantic ambiguity* (Small et al., 1988). Syntactic ambiguity refers to the ambiguity of grammatical categories, and often, as a consequence, a sentence can be parsed in various ways. By semantic ambiguity we mean that a word may have many different meanings, either related to one another or not.

Classically, the theory and practice of lexical ambiguity resolution (and, in fact, of all of natural language processing) has followed a path that today we might be tempted to label “algorithmic,” as opposed to the more recent, “nonalgorithmic’ connectionist approach. Whereas the former (as described in Allen, 1987; Winograd, 1983) is naturally suited to execution by sequential computers, the latter has found motivation in the very strong appeal of parallel computers, and in its (perhaps less agreeable) plausibility from the standpoint of brain function imitation (Cottrell, 1985; McClelland & Kawamoto, 1986; Waltz & Pollack, 1985). Regardless of the approach adopted, it is now a consensus that the processing of natural languages should rely on functional modules for syntactic and semantic analyses that interact intimately with each other (see, e.g., Lytinen, 1991; McRoy & Hirst, 1990; as well as their references). This seems, however, to be significantly harder to achieve under the constraints imposed by sequential computational models.

We perceive this consensus on the necessity of a close interaction of the modules for syntactic and semantic analyses as being motivated essentially by two factors. The first factor is related to the difficulty that early approaches to natural language processing had, because of their inherently sequential character, in constructing closely interacting modules. This seems to have been responsible for those approaches’ great inability to deal with lexical ambiguities, structural ambiguities, and garden-path sentences. Second, there seems to be ample evidence supporting the psychological plausibility of the fact that the processes of syntactic and semantic analysis, although modular, interact strongly with each other. This is apparently the dominant view concerning a variety of issues in the processing of natural languages, and, in particular, lies at the core of the model for lexical access that is currently most widely accepted. According to this model, every meaning of a word is retrieved initially, and then one of them is favored when the correct context is established (Simpson & Burgess, 1988).

In this article we deal with the problem of resolving lexical ambiguities of the syntactic and semantic types by adopting a model that allows the syntactic and the semantic processing to take place fully concurrently, therefore, in a highly cooperative fashion. Our point of departure has been to recognize that the process of disambiguation is essentially the process of resolving uncertainties in the syntactic and semantic domains, and that, in
Bayesian lexical disambiguation

Bayesian lexical disambiguation turns probabilistic models provide a principled way of dealing with uncertainties. Whether probability theory suffices to deal with uncertainties is still reason for great controversy, especially within more complex application areas, like the semantic interpretation of natural languages (Norvig & Wilensky, 1990). However, if probabilities are interpreted as measures of belief rather than frequencies, then there appear to be various reasons to support their use in the treatment of uncertainties (Cheeseman, 1985). The work we describe here has its origins in the report by Mendes and Eizirik (1989), where some of the ideas can be found in an embryonic stage. Independently, Charniak and Goldman (1989) also advocated the use of probabilistic models to treat ambiguity, and started investigating the use of such models for word-sense disambiguation at the story-understanding level (a more recent account of their work, which we shall discuss in more detail shortly, is given in Goldman & Charniak, 1992).

Our approach has been to employ Bayesian networks (Pearl, 1988) to model the uncertainties involved in lexical ambiguity. These networks are based on the use of probabilistic causal relationships among entities, and are no less attractive than connectionist systems when it comes to a parallel implementation. Moreover, with the use of Bayesian networks, the problem of ensuring the “lateral inhibition” or the “winning” in connectionist systems, so crucial to their use in the treatment of natural languages (and typically handled rather informally), becomes irrelevant, because this type of behavior is already embedded in the Bayesian formalism. Furthermore, as a Bayesian network can be solved via stochastic simulation (and by other methods as well), which, as we shall see, can be shown always to converge to the equilibrium distribution, one immediately has a clear advantage over connectionist systems like Boltzmann machines (Hinton, Sejnowski, & Ackley, 1984) and feedforward networks trained by the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986), where convergence to poor local minima is a constant problem. (We shall see, nevertheless, that a useful variation of the process of stochastic simulation of a Bayesian network can benefit greatly from some of the characteristics of a Boltzmann machine.)

The model we propose allows the following two functions to be fully performed in parallel: (1) Syntactic analysis, involving one or more equally probable parse trees (depending on whether the sentence is syntactically ambiguous); and (2) semantic analysis, based on a representation of knowledge that tries to fulfill the required cases for each verb meaning [in the sense of Fillmore’s (1968; Samlowski, 1976) case grammars]. When the syntactic analysis would result in an ambiguous decision, the semantic analysis helps decide which of the competing parse trees to select, depending on the cases that can be fulfilled. Otherwise, word-sense disambiguation is all that needs to be achieved. For example, when presented with the sentence “Time flies
like an arrow" (Winograd, 1983, p. 92), which is syntactically ambiguous, the network will at first tend to produce three parse trees during the process of stochastic simulation, and will eventually settle with the choice of flies as the most likely candidate for a verb with all cases fulfilled, at which time one single parse tree will then remain. This sentence will be used throughout this article as an example. Similarly, in the absence of multiple candidate parse trees, noun- and verb-sense disambiguation can be achieved, always relying on a mechanism for verb-case fulfillment.

Great portions of our Bayesian network for lexical disambiguation can be obtained automatically from a description of the syntactic and semantic domains under consideration. The syntactic domain is described by a context-free grammar in which some productions are singled out to indicate structural characteristics of relevance to the disambiguation process at the semantic level. Normally, these productions indicate how the coarse structure of the sentences generated by the grammar can be related to verb cases. In addition, the description of the syntactic domain comprises, for each word that the network is capable of handling, a set of the possible grammatical categories of that word. The semantic domain is, in turn, described by providing the possible meanings for each word. If the word has more than one possible grammatical category, then its possible meanings are related to those categories. In particular, the meanings of verbs are described by their required cases. A subnetwork for syntactic analysis can be obtained automatically from the context-free grammar, and a subnetwork for semantic analysis can be constructed from the set of words and their possible grammatical categories, as well as from their meanings. Once the two subnetworks have been built, they can be interconnected by an automatic procedure as well.

Lexical disambiguation in our Bayesian model is directed by neither the syntax nor the semantics alone, but emerges from the interaction of the two subnetworks, each devoted to one of the syntactic and semantic aspects of the problem. As a consequence, the model is at least, in principle, capable of handling sentences that are not grammatically correct, in the sense discussed by Charniak (1983) with his example "fire arson match hotel" (p. 174), where the absence of a grammatical structure only slightly disturbs one's understanding of the semantics (in this context, the appeal of a probabilistic model that issues a diagnostic with a measure of relative likelihood attached to it is even greater). In addition, and perhaps more importantly, this continual interplay between the two subnetworks accounts, based on our discussion earlier in this section, for our model's relative plausibility in terms of human psychology.

These main characteristics of our Bayesian-network model for lexical disambiguation indicate quite markedly the main differences between our model and others, both connectionist and Bayesian. For example, the ap-
proach proposed by Cottrell (1985) suffers from various idiosyncrasies of its underlying connectionist model and, as a consequence, is not guaranteed to yield the correct answer. Furthermore, Cottrell's network is critically dependent upon the timed sequencing of its input, which to us seems to be somewhat in disaccord with the aforementioned psychological plausibility of fully cooperative, concurrent processing. These problems are shared by the approach proposed by Waltz and Pollack (1985), which, in addition, seems to be very dependent upon the particular sentence at hand, and for this reason appears to be very hard to generalize. Another connectionist approach was pursued by McClelland and Kawamoto (1986), and differs from most of the others (including ours) in that it is essentially centered around the learning, distributed representation, and later generalization of concepts. In particular, McClelland and Kawamoto did not touch the issue of syntactic analysis at all.

Charniak and Goldman (1989) and Goldman and Charniak (1992) also employed Bayesian networks as part of an approach to interpret sentences aimed at the comprehension of a story comprising those sentences. The overall approach consists of performing, for each sentence, a syntactic analysis whose result is then used to construct, with the aid of a knowledge base, a Bayesian network for use in obtaining the most likely interpretation of the portion of the story encountered so far. Additional sentences, when input to the system, may elicit new information and then cause the network to be changed. In contrast to their approach, it seems that ours concentrates on employing Bayesian networks precisely when theirs employs traditional techniques (Charniak & Goldman, 1991; Goldman & Charniak, 1992).

As we remarked before, the model we describe for the resolution of lexical ambiguities relies on a process of verb-case fulfillment based on Fillmore's (1968) case grammars. Although, to some extent, lexical ambiguities can also be resolved at a lower level through grammatical agreement checking (not so much in English, but certainly in other languages, especially those of Latin origin), we do not explicitly describe that in this article. However, throughout the article the reader will notice that these tests of grammatical agreement in a Bayesian model are relatively simple and straightforward.

We also note that, although syntactic ambiguity is often related to the existence of multiple parse trees, the more general case in which this happens, known as structural ambiguity, is not treated here. So, resolving the ambiguity, in "I saw the man on the hill with a telescope" is outside the intended scope of this article. Likewise, we do not dwell on the topic of Bayesian knowledge representation, although it is of great importance to the process of lexical disambiguation. Rather, we show how to obtain the appropriate knowledge representation for our examples only.

The remainder of the article is organized in five additional sections: Section 2 is devoted to an outline of Bayesian networks; Sections 3 and 4
describe our network for lexical disambiguation (Section 3 describes the network for parsing, and Section 4 the network for semantic analysis, as well as how the two parts interact); Section 5 continues with the description of simulation strategies in both sequential and parallel machines, followed by a discussion of experimental results; and conclusions are given in Section 6. The material in Section 2 is based on the work of Pearl (1988), and Sections 3 through 5 are based on the work of Eizirik (1990). The Appendix provides more detailed information about the process of performing the stochastic simulation in parallel.

2. BAYESIAN NETWORKS

A Bayesian network is a probabilistic model in which an acyclic directed graph is used to represent probabilistic causal relationships among random variables. A node in a Bayesian network represents one of the n random variables in the set \( X = \{ X_1, \ldots, X_n \} \), each of which we take in this article to be a binary variable (i.e., it assumes value 0 or 1). A directed edge from \( X_i \) to \( X_j \) indicates the existence of a direct causal influence between \( X_i \) and \( X_j \). Given a probability distribution \( P \) on \( X \), a Bayesian network is formally defined as a minimal acyclic directed graph in which every "structural" node separation (according to a criterion that we need not specify here) has a corresponding conditional independence of events in \( P \). In the remainder of the article we simplify the notation by letting \( X_i \) denote the event \( X_i = 1 \) and \( \neg x_i \) denote \( X_i = 0 \). Likewise, we let \( x_i \) be either \( X_i = 1 \) or \( X_i = 0 \).

In a Bayesian network, a node \( X_i \) may have parent nodes (\( X_j \) such that the edge \( X_j \rightarrow X_i \) exists), child nodes (\( X_j \) such that the edge \( X_i \rightarrow X_j \) exists), and mate nodes (\( X_j \) such that the edges \( X_i \rightarrow X_k \) and \( X_j \rightarrow X_k \) exist for some \( X_k \)). We let \( P(X_i) \), \( C(X_i) \), and \( M(X_i) \) denote the sets of nodes of these three types, respectively. Also, variables whose values have been observed and are therefore known are called evidences, and may, in the context of a particular application, be regarded as constituting the input to the network. The set of evidences is denoted by \( E \).

For a distribution \( P \) and a total order \( \prec \) on \( X \), a parent set of \( X_i \), \( P(X_i) \), is a subset \( Y \subset X \) such that \( X_j \prec X_i \) for all \( X_j \in Y \) and \( Y \) is minimal with respect to the property that

\[
P(x_i | x_j; X_j \prec X_i) = P(x_i | x_j; X_j; \in Y).
\]

(This subset is unique if \( P \) is strictly positive.) So, \( P(x_1, \ldots, x_n) \) can be written in a Bayesian network as the product

\[
P(x_1, \ldots, x_n) = \prod_i P[x_i | x_j; X_j \in P(X_i)],
\]

which can then be used in the evaluation of

\[
P(x_i | x_j; X_j \in E),
\]
where one of the fundamental questions to be answered about the underlying probabilistic domain is summarized. The probability in Equation 3 is known as the posterior probability in the presence of E. Besides the computation of these posterior probabilities, another related problem of great relevance (in fact, of primary relevance in the context of this article) is the identification of the point in \( \{0, 1\}^{|\mathbf{X} - E|} \) at which

\[
P(x_i; X_i \in \mathbf{X} - E|x_j; X_j \in E)
\]

is highest. The distribution in Equation 4 is the so-called joint posterior distribution given E.

From the standpoint of utilizing Bayesian networks as a modeling tool in practical situations (i.e., when the causal relationships among variables are known but the distribution \( P \) is not known), what needs to be done is to specify for each \( X_i \in \mathbf{X} \) the set \( P(X_i) \) and the conditional probabilities appearing in Equation 2 for every possible combination of the values of the \( x_j; X_j \in P(X_i) \) [if \( P(X_i) = \emptyset \), then one single value, \( X_i \)'s prior probability, is needed]. This is done on an empirical basis, following one's intuition about the domain being modeled. Once these probabilities are available, the posterior probability of a particular variable (cf. Equation 3) or the point of maximum joint posterior probability (cf. Equation 4) can be found for a given set of evidences E. In this case, Equation 2 is still valid because the total order \( \prec \) in which every variable \( X_i \) precedes its children is consonant with Equation 1 for \( Y = P(X_i) \).

It turns out, however, that determining the probability in Equation 3 is, in general, computationally intractable (Cooper, 1990), and then, so is the problem of identifying the maximum of the distribution in Equation 4. In Section 5 we shall have more to say about this, and about importance of the sets \( C(X_i) \) and \( M(X_i) \) in helping to overcome this problem via stochastic simulation and variations thereof.

In the next section we begin to describe our Bayesian network for parsing, and in doing so, some of the network's nodes will have a special probabilistic structure, which we describe now. A node \( X_i \) for which \( P[X_i|x_j; X_j \in P(X_i)] \) has a very low value, except at the \(|P(X_i)| \) configurations \((1,0,\ldots,0), (0,1,\ldots,0), \ldots, (0,0,\ldots,1)\) of its parents, is an XOR node. Similarly, a node \( X_i \) for which \( P[X_i|x_j; X_j \in P(X_i)] \) has a very low value, except at the configuration \((1,1,\ldots,1)\) of its parents, and an AND node. At this point, we need not be more specific about this "very low value;" we note, however, that ideally, it would equal zero, were it not for convergence problems of the stochastic simulation process, to be discussed in Section 5.

### 3. THE PARSING NETWORK

The Bayesian network we describe for parsing can be built automatically for any context-free grammar, following the procedure outlined later in this
section. However, some productions in the grammar must be written in a way that synthesizes the structural information of relevance to the disambiguation process (this is not a requirement of the methodology for building the parsing network, but rather of the mechanism whereby the syntactic and semantic networks are interconnected, as we explain shortly). In addition, as a matter of practicality, there has to be a bound on the number of words in a sentence, so the language generated by the grammar can be thought of as being regular, although this fact is in no way used by the procedure employed to build the parsing network. In this article we use as an example the grammar $G$ whose productions are listed as follows:

1. $S \rightarrow W PP$
2. $S \rightarrow W NP$
3. $S \rightarrow W NP PP$
4. $W \rightarrow NP$ verb
5. $W \rightarrow$ verb
6. $PP \rightarrow$ prep NP
7. $NP \rightarrow$ det $N$
8. $NP \rightarrow N$
9. $N \rightarrow$ adj $N$
10. $N \rightarrow$ noun

According to $G$, our example sentence, "Time flies like an arrow," can be parsed in three different ways, as shown in Figure 1, where each of the parse tree corresponds to one of flies, like, and Time being chosen as verb in the sentence.

An important structural property of the grammar $G$ is that the first three productions (those with the initial symbol $S$ on the left) have been singled out to emphasize structural properties of the possible parse trees that will be of relevance in the disambiguation process at another level. This is something that will always have to be done, and has to be approached on a case-by-case basis, depending on the language at hand, in order to indicate the possible variations in the coarse structure of the sentences. In the case of $G$, the first three productions indicate the various ways whereby verb phrases, noun phrases, and prepositional phrases can be combined to yield sentences in the language. The importance of this explicit indication of a specific production for each combination will later become clearer, and has to do with the process of verb-case fulfillment. More specifically, each of these productions indicates a possibility for the verb's cases. If, for example, in $G$ a sentence can be parsed using Production 1, then its verb, in the corresponding semantic interpretation, must allow a case with a preposition (Fillmore, 1968), and similarly for Productions 2 and 3.

Another important characteristic of the grammar $G$ is that the nonterminal $S$ does not appear in the right-hand side of any production. The reason for this restriction is related to generalizing our model to treat structural ambiguities, and will be discussed in some more detail in Section 6.
Figure 1. Three different ways of parsing "Time flies like an arrow."
Although the language $L(G)$ generated by $G$ contains arbitrarily long sentences, in practice, a model for parallel syntactic analysis has to incorporate a parameter $\ell > 0$ indicating the maximum length to be allowed for a sentence. Once this parameter is specified, the network can be built to recognize the subset $L_t(G)$ that contains all of $L(G)$'s members whose length is no greater than $\ell$. In our discussion here, we take $\ell = 5$ when treating our example sentence.

The network for parsing $L_t(G)$ has $\ell$ groups of input nodes. The nodes in the $k$th group, $1 \leq k \leq \ell$, correspond to the grammatical categories that can occur in the $k$th position of a sentence in $L_t(G)$. For example, the node verb3 corresponds to the occurrence in the third group of a word whose grammatical categories include verb. One should note that determining a priori which categories may occur in which group is, in general, computationally costly; however, the general procedure we outline in the following for constructing the network renders this first step unnecessary.

When a sentence is input to the network for parsing, all input nodes are clamped (i.e., their values are fixed): Those that correspond to possible grammatical categories of the words are clamped to the value 1, the others to 0. For the sentence "Times flies like an arrow," the nodes clamped in 1 would be noun1, adj1, verb1, verb2, noun2, verb3, prep3, det4, and noun5. These input nodes have no parents, but nonetheless one need not worry about their prior probabilities, which would be meaningless in view of the fact that they are always clamped.

The remaining nodes of the parsing network are either AND or XOR nodes, explained in Section 2, depending on whether they represent the conjunction of more than one terminal or nonterminal symbol of $G$ rewriting a nonterminal symbol via exactly one production (this is the case of an AND node), or they represent the exclusive disjunction of many instances of a same nonterminal symbol appearing in the left-hand side of more than one production (XOR node).

In Figure 2 we show a portion of the parsing network for $L_t(G)$. The portion we show corresponds to the three possible ways of parsing our example sentence (incidentally, only AND nodes appear in the portion shown). The numbers appearing next to each noninput node's identification refers to the production in $G$ to which that node corresponds.

Of particular importance in a parsing network like that of Figure 2 are the AND nodes corresponding to the initial symbol $S$, in this case $S_1$, $S_2$, and $S_3$. These nodes function as "syntactic probes," and will be of fundamental importance in conveying the sentence's coarse syntactic structure to the network for semantic analysis. In fact, this is why the productions of $G$ rewriting the initial symbol $S$ have to be written carefully, as discussed previously. The remaining nodes of the parsing network belong to the possible parse trees. For example, those from which $S_1$ can be reached are in the parse tree obtained when Production 1 is applied first.
Figure 2. Portion of the Bayesian network for parsing sentences in $L(G)$. 

The diagram illustrates the relationships between different parts of a sentence, showing how the network processes and disambiguates the meaning of the sentence.
A Bayesian network for parsing can be constructed automatically for any context-free grammar, following the steps outlined in the following. We assume that the nonterminal $S$ does not appear in the right-hand side of any production, and that the value $\ell$ of the maximum sentence length allowed has been specified.

1. Eliminate all $e$-productions.
2. Create $\ell$ groups of nodes, each containing one node for each terminal symbol in the grammar.
3. For $1 \leq k \leq \ell$, do:
   a. Create one AND node for each production from which a sequence of $k$ terminal symbols can be eventually derived. Corresponding to each symbol in the right-hand side of this production there exists an AND node or an XOR node created in some iteration $k' \leq k$ of this step or in Step 2. Let these nodes be the parents of the newly added AND node.
   b. If more than one of the productions used in Step 3a rewrite the same nonterminal, and this nonterminal is not $S$, then create one XOR node for each such nonterminal, and let its parents be the corresponding AND nodes created in Step 3a.
4. Repeatedly eliminate all nodes without a child, except those that originate from a production rewriting $S$ and that correspond to a sequence of terminal symbols (cf. Step 3a) beginning at the leftmost position in the sentence (this process eventually eliminates input nodes corresponding to categories that cannot occur in a certain group).

For the sake of illustration, let us briefly examine the process of generating the network fragment of Figure 2 by the algorithm. Every node without a parent is generated in Step 2. For $k = 1$ in Step 3, the algorithm generates the nodes with labels $W_s$, $N_{1a}$, and $NP_s$ (with parent $N_{1a}$). For $k = 2$ in Step 3, nodes with labels $W_s$ (with parents $NP_s$ and verb2), $NP_s$, $N_s$, and $NP_s$ (with parent $N_s$) are generated, and similarly for the remaining values of $k$.

We should note that the overall structure of the network produced by this algorithm is very similar to that of the parsing (connectionist) network proposed by Fanty (1985). However, the two networks are functionally very different from each other, as a consequence of the models adopted.

As discussed by Eizirik (1990), the computational complexity of this algorithm is a function of the number of terminal and nonterminal symbols in $G$, the number of productions, the number of symbols in the lengthiest production, and $\ell$. Both the time and space complexities involve a term that depends on $\ell$ according to the middle binomial coefficient

\[
\binom{\ell}{(\ell/2)}.
\]
so the algorithm is impractical for arbitrary values of \( \ell \). However, the algorithm is to be used only once given \( G \), and typically \( \ell \) will be modestly sized, so in practice, this automated procedure for obtaining the parsing network can be applied. Even so, there is still the question of how costly the process of stochastic simulation of the network can be, in view of the fact that the number of nodes is exponential in the size of the grammar. As we have had the opportunity to allude to earlier, and will explain in more detail later on, this simulation can be parallelized with a high degree of concurrency in node updating, which in itself already significantly reduces the simulation time, provided enough processors are available (as we remarked before, the grammar \( G \) is fixed, so requiring an exponentially large number of processors is not a case of "exponential growth"). There is also, however, the question of how many times each node has to be updated. Although we still lack a good theoretical understanding of this issue, it seems that the number of times a node has to be updated in a stochastic simulation of a Bayesian network is much more dependent upon the desired accuracy than upon the size of the network (Pearl, 1988).

4. THE NETWORK FOR LEXICAL DISAMBIGUATION

The complete network for lexical disambiguation comprises the network described in the previous section for parsing, and the network for semantic analysis that we describe in this section. Initially, for simplicity, our discussion will be tailored to the construction of a network for the disambiguation of the example sentence we have been using. An outline of how to proceed in general will then follow.

The Bayesian network we propose appears in two different levels of details in Figures 3 and 4. These figures depict the general outline of the network, and show the nodes and interconnections for determining the correct grammatical category of each word, its meaning, and the verb's agent, object, and so on. [This terminology is derived from Fillmore's (1968; Sambowski, 1976) case grammars, which are—as was done by Cottrell (1985)—used relatively freely for the purpose of lexical disambiguation.]

The input nodes to the semantic portion of the network are, in the case of our example sentence, the nodes Time1, flies2, like3, an4, and arrow5, where the digits indicate the respective positions in the sentence. These nodes, together with the nodes for grammatical categories used previously as inputs to the parsing portion of the network, constitute the set \( E \) of evidences for the entire lexical disambiguation network.

Each of the input nodes in Figure 3 has, in its parent set, the nodes \( S_i \), \( S_r \), and \( S_t \) (which summarize the parsing diagnostics), and nodes that describe the various possibilities of the corresponding word as \( \text{verb} \) or as part of an \( \text{NP} \). For example, for the node \( \text{flies2} \) we have
Figure 3. The network for lexical disambiguation.
Figure 4. Portion of the Bayesian network for semantic analysis.
\[ P(\text{flies2}) = \{S_1, S_2, S_3, \text{fliesNP0}, \text{fliesV}, \text{fliesNP1}\}, \]

where \( \text{fliesNP0} \) indicates the possibility of having \( \text{flies} \) in an NP before \( \text{verb} \), \( \text{fliesV} \) of having it as \( \text{verb} \), and \( \text{fliesNP1} \) in the first NP after \( \text{verb} \). The other input nodes are treated similarly.

The conditional probabilities associated with input nodes indicate the possible roles for each word, depending on the parsing diagnostics represented by \( S_1, S_2, \) and \( S_3 \). In the case of \( \text{flies2} \), these probabilities have significant values only at the combinations

\[ P(\text{flies2}|\text{fliesNP0}=s_1, \text{fliesV}=s_2, \text{fliesNP1}=s_3), \]

\[ P(\text{flies2}|\text{fliesNP0}, \text{fliesV}=s_1, \text{fliesNP1}=s_2, \rightarrow S_1, \rightarrow S_3), \]

and

\[ P(\text{flies2}|\text{fliesNP0}=s_1, \text{fliesV}=s_1, \text{fliesNP1}=s_2, \rightarrow S_1, \rightarrow S_2, S_3), \]

where \( s_1, s_2 \in \{0, 1\} \) (as in the end of Section 2, these “significant values” will be specified later; except for convergence problems, they would ideally equal one). In the first combination, for example, we state that with significant probability we have \( \text{flies2}=1 \), given that \( \text{fliesV}=1, S_1=1, \) and \( S_2=S_3=0 \), regardless of the values of \( \text{fliesNP0} \) and \( \text{fliesNP1} \). In other words, in order for \( \text{flies2} \) to be the sentence’s \( \text{verb} \), it suffices that at least one parsing diagnostic in which \( \text{verb} \) appears in the second position be present (\( S_1 \) in this case), whereas those that do not have \( \text{verb} \) in that position all be absent (\( S_2 \) and \( S_3 \) in this case). It is important to note that all combinations of \( \text{fliesNP0} \) and \( \text{fliesNP1} \) should be allowed in conjunction with \( \text{fliesV}=S_1=1 \), inasmuch as simply setting \( \text{fliesNP0}=\text{fliesNP1}=0 \) would forbid the occurrence of sentences like “\text{Flies fly}”, “\text{Men man}”, and so on, where basically the same word appears in different positions.

Similarly, the second combination indicates that \( \text{flies2} \) is in an NP before \( \text{verb} \) if at least one parsing diagnostic with \( \text{verb} \) in the third position (or farther to the right) is present (\( S_2 \) in this case), whereas those where \( \text{verb} \) is not in the third position (or farther to the right) are all absent (\( S_1 \) and \( S_3 \) in this case). This same reasoning applies to the remaining combination and to all other input nodes.

In general, the parent set of input nodes in the network for semantic analysis can be determined automatically, given the parsing network obtained from the grammar \( G \) and the sets of the possible grammatical categories of each word, following the procedure we outline next. Let \( D \) denote the set of parsing diagnostics (these are the nodes that correspond to productions in \( G \) rewriting the initial symbol \( S \)) obtained as part of the parsing network described in Section 3, and let \( C(w) \) be the set of possible grammatical categories of a word \( w \). For the example we have been discussing, \( D = \{S_1, S_2, S_3\}, C(\text{flies}) = \{\text{verb, noun}\}, \) and so on. Our goal now is to describe how to
obtain $P(w_k)$, and the corresponding conditional probabilities, for input node $w_k$, where $1 \leq k \leq \ell$ ($w_k$ here indicates the node identified by the string obtained by the concatenation of strings $w$ and $k$). The following steps achieve this, utilizing an auxiliary set $I(w_k)$.

1. For $S_p \in D$ and $cat \in C(w)$, check whether $S_p$ is reachable from $catk$ in the parsing network. In the affirmative case, do:
   a. Add $S_p$ to $P(w_k)$.
   b. Use the positions in the sentence of verb and of prep (as given by $S_p$) in choosing one of $NP_0$, $V$, $NP_1$, or $NP_2$ to indicate the phrase within the coarse structure of the sentence in which $w_k$ appears according to $S_p$. Let $Y$ be the phrase chosen, then add $wY$ to $P(w_k)$ and the pair $(S_p, wY)$ to $I(w_k)$. (Here, $NP_0$ indicates an NP before verb, $V$, a verb, $NP_1$ the first NP after verb, and $NP_2$ the second NP after verb.)

2. For $(T_1, T_2) \in I(w_k)$, let the conditional probabilities at node $w_k$ be such that a high value is assigned only to the combinations $P[w_k|T_1, T_2]$, all other members of $P(w_k) \cap D$ are 0, the remaining members of $P(w_k)$ are either 0 or 1].

It is relatively simple to see that this procedure's complexity is polynomial in the size of its inputs (the parsing network and the sets of grammatical categories for each word). As we remarked earlier, however, the size of the parsing network may depend exponentially on the size of the grammar $G$.

We now return to our example sentence. Aside from $S_1$, $S_2$, and $S_3$, which convey information about the sentence's parse tree, additional nodes to look at for a complete syntactic diagnostic (relating each word to its phrase within the sentence) in the network of Figure 3 are the other nodes in the parent sets of the input nodes. The joint probability of their variables in the presence of the evidences in $E$ will indicate the most likely candidate for the position of verb ($fliesV$ in this case), along with the accompanying $NP$'s before and after verb ($TimeNP_0$ and $arrowNP_1$, respectively, in this case). These nodes in turn have as parents nodes related to the various possible word senses of the candidates for the three $NP$'s and for verb. Networks for case hierarchies of the $NP$'s are then used to fulfill the required cases of the verbs. In different applications, it is conceivable that the entire parse tree would be needed, in which case the tree's nodes in the parsing network can be examined as part of the syntactic diagnostic.

Figure 4 shows in more detail the nodes for word-sense resolution and case fulfillment for verb and the $NP$ that precedes it. For example, the node $TimeV$ has, as a parent, the node $T_1$, which represents the possibility of having $Time$ used as a verb in the sense of timing athletic events, or timing the arrival of airplanes, and so on. Node $fliesV$ in its turn has two parents, $F1$ and $F2$, representing respectively the use of $flies$ as a verb in the sense of
time elapsing or of piloting an aircraft. Each of these nodes for verb senses has, as parents, nodes for the corresponding cases. For example, Fl has a parent FlObj corresponding to a required object (it would also have a "modifier case" to be fulfilled by arrow, but we have omitted that from the figure).

The semantic diagnostic is given by the network in the word-sense nodes and in the nodes that specify verb cases. For "Time flies like an arrow," a highly likely conjunction of events, in which the nodes Timeperiod, Fl, and FlObj have value 1, is part of the correct diagnostic. That this diagnostic can indeed be expected to be achieved comes from the realization that only the sense corresponding to node Fl will have its cases completely fulfilled. As a consequence, fliesV occurs with much higher probability than TimeV, and by the conditional probabilities in the nodes Time1 and flies2, TimeNP0 occurs with much higher probability than fliesNP0.

The nodes for word-sense resolution and for case fulfillment are connected to semantic networks comprising nodes for a Bayesian representation of knowledge. As we remarked in Section 1, we have limited ourselves to building only very modest segments of these networks, as needed for the experiments we report in Section 5. We believe that extensive work is still required before a thorough understanding and formalization of Bayesian knowledge representation is achieved, and that is outside the intended scope of this article. Initial steps have been taken by some researchers, for example, Bacchus (1991).

One observation is in order now concerning the conditional probabilities we have been adopting in the description of our Bayesian network for lexical disambiguation. Although in both the parsing network and in the nodes that interconnect this network with the network for semantic analysis we restricted ourselves to using binary conditional probabilities, this will not, in general, be true for the remaining parts of the network. For example, when it comes to specifying the conditional probabilities associated with the meanings of verbs and nouns, the use of nonbinary probabilities is essential for conveying the expected relative frequency of occurrence of the different meanings. In fact, this possibility accounts for an even greater interest in the use of a Bayesian model, as the use of exclusively binary conditional probabilities would render parts of the network (e.g., the parsing network, if used on a stand-alone basis) as simple as a straightforward classifier.

5. SIMULATION STRATEGIES AND RESULTS

The Bayesian network for lexical disambiguation described in this article has cycles in its underlying undirected graph. In addition to the general intractability (in the sense of NP-hardness) of evaluating the probability in Equation 3, the existence of these cycles accounts for further problems with
the employment of methods of solution otherwise feasible (albeit expensive). Pearl (1988) discussed the technique of *stochastic simulation* as an approach to overcome the difficulty of an exact analytic treatment, with the additional advantage of being inherently amenable to parallel processing. A great portion of this section, as well as all of the Appendix, are not specific to the problem of lexical disambiguation, but rather refer to any sufficiently complex Bayesian network.

In a stochastic simulation, each variable $X_i \in \mathbf{X}$ is updated a number of times according to the conditional probability

$$P(x_i | x_j; j \neq i),$$

and along the process, the frequency with which the event $X_i = 1$ happens is registered and taken at the end as an approximation to the probability in Equation 3.

Let $\mathcal{N}(X_i)$ be the set of *neighbors* of $X_i$ in the network, defined by

$$\mathcal{N}(X_i) = \mathbf{P}(X_i) \cup \mathbf{C}(X_i) \cup \mathbf{M}(X_i).$$

It turns out that for a Bayesian network, the probability in Equation 5 is given by

$$P(x_i | x_j; j \neq i) = P[x_i | x_j; X_j \in \mathcal{N}(X_i)],$$

which is an extension of the Markovian dependency beyond the usual one-dimensional setting. This probability can be shown to be computable using solely the probabilities (conditioned on parents’ values) corresponding to $X_i$ and to the nodes in $\mathbf{C}(X_i)$, along with the values of the variables in $\mathbf{P}(X_i)$, $\mathbf{C}(X_i)$, and $\mathbf{M}(X_i)$ (Pearl, 1988). Precisely,

$$P(x_i | x_j; j \neq i) = \alpha P[x_i | x_j; X_j \in \mathbf{P}(X_i)] \prod_{x_j \in \mathbf{C}(X_i)} P[x_j | x_k; X_k \in \mathbf{P}(X_j)],$$

where $\alpha$ is a normalizing constant [note that the members of $\mathbf{P}(X_i)$ inside the product are also in $\mathbf{M}(X_i)$]. As far as a distributed parallel simulation is concerned, this means that we can associate one concurrent process with each of the variables in $\mathbf{X}$, and have the processes communicate with one another through channels that coincide with the edges of the undirected graph underlying the Bayesian network. [There are additional issues related to the concurrent updating of two variables $X_i$ and $X_j$ such that $X_j \in \mathcal{N}(X_i)$, and more channels are usually added as a consequence, as we discuss in the Appendix.]

If the probability distribution $\mathbf{P}$ is strictly positive (i.e., if the probabilities conditioned on parents’ values for each variable do not strictly forbid any combination of those values), then the stochastic simulation necessarily converges to an equilibrium solution. So, as long as we refrain from assigning probability zero to some variable conditioned on some combination of its parents’ values, the approach of stochastic simulation is guaranteed to yield an approximation to the equilibrium probabilities of the Markov process.
underlying the simulation, and consequently to the probability in Equation 3. The conditional probabilities of \( X_i \) are then constrained by

\[
0 < P[X_i | x_j; X_j \in P(X_i)] < 1,
\]

and evidently by

\[
P[X_i | x_j; X_j \in P(X_i)] + P(-X_i | x_j; X_j \in P(X_i)) = 1,
\]

for all \( X_i \in \mathbf{X} \) and all combinations of the \( x_j \)'s.

Stochastic simulation is not the only method of approximate solution of a Bayesian network and, in fact, is not even the most efficient one. In particular, if some of the conditional probabilities are set to extreme values (as we do, in some cases), then the process of stochastic simulation is rather slow, and is sometimes far outweighed by some of the others. Hemion (1990) provided a good comparative survey of these methods, and gave numerous references that can be looked up for more details. However, as we remarked earlier in Section 2, and has now become clear from the material in Section 4, solving for the posterior probability of single variables does not help us much. What we really need is to solve for the value assignments to the variables in \( \mathbf{X} - \mathbf{E} \) for which the joint posterior probability (cf. Equation 4) is maximum. In this case, the basic stochastic simulation approach described so far is insufficient, for its primary purpose is to solve for the posterior probabilities of single variables (as given by Equation 3). A slight modification to the basic stochastic simulation can, however, be shown to yield excellent approximations to the overall assignment of values at which the joint posterior probability is maximum. This modification is based on the incorporation to the basic method of the simulated annealing heuristic of Kirkpatrick, Gelatt, and Vecchi (1983), which not only elicits the possibility of reaching approximate maxima of the joint posterior distribution, but also seems to render the stochastic simulation method relatively insensitive to the presence of extreme conditional probabilities (although Equation 7 still has to be satisfied).

The modified stochastic simulation approach has been described by Hrycej (1990), and is based on results of Geman and Geman (1984) on the updating of Markov random fields, of which Bayesian networks can be thought of as being a particular case, in view of the property summarized in Equation 6. Specifically, let \( T \) be the temperature-like parameter of simulated annealing, that is, it is a parameter that starts off at a relatively high value and decreases slowly as the process goes on. The key observation is that, if stochastic simulation is performed based on a variation of Equation 6 in which all conditional probabilities are raised to the \((1/T)\)th power, and if \( T \) is not decreased any faster than the rate given by Geman and Geman (1984), then a maximum of the joint posterior distribution is guaranteed to be identified. This maximum rate of reduction for \( T \)'s, never-
theless, impractical, as it requires an exponentially large number of updates per variable. Much faster rates are usually employed, instead, apparently without much harm to the quality of solution obtained. It might be mentioned, in addition, that this modified version of stochastic simulation bears some resemblance to the process of updating the variables in Boltzmann machines (Hinton et al., 1984), although luckily it does not appear to have inherited their less fortunate characteristics as well.

We have built a distributed parallel simulator in the Occam language, which is tailored to be executed on a network of transputer microprocessors (Burns, 1988). The design of this simulator has followed closely the general guidelines described by Barbosa and Lima (1990) for the simulation of some neural networks and is outlined in the Appendix. We have performed simulation experiments with our parallel software on a network of nearly 400 nodes (and consequently 400 concurrent processes) with very stimulating results. For a network of this size, capable of handling sentences of up to five words, all clamped simultaneously in the beginning of the simulation, convergence to an approximate maximum of the joint posterior distribution happens in a few thousand updates per variable. This is to be taken as a very satisfactory performance, especially when contrasted with earlier (connectionist) proposals equally amenable to parallel processing. Take, for example, the work of Selman (1985), where the connectionist parsing, although parallelizable, suffers from all the idiosyncrasies of Boltzmann machines (Hinton et al., 1984), often trading performance for quality of solution.

In addition to the disambiguation of sentences that can be parsed correctly by $G$, the use of a probabilistic model allows us to treat, to some extent, sentences whose grammatical structures are not entirely correct. For example, let the set of evidences be given by $flies_1$, $airplane_2$, and $time_3$, for the semantic analysis subnetwork, and by $noun_1$, $verb_1$, $noun_2$, $noun_3$, and $verb_3$, for the syntactic subnetwork. This sentence cannot be correctly parsed in $G$, but, nonetheless, the network indicates a parse tree with verb in the first position as the most likely parsing diagnostic. Similarly, a semantic diagnostic that indicates piloting an aircraft as the most likely meaning of flies, together with airplane as an object case, is obtained. Apparently, then, our approach is relatively insensitive to the absence in the sentence of items such as determiners and prepositions, as the sentence's verb is still correctly identified and its cases reasonably fulfilled. However, if the sentence is ungrammatical to the point that more vital items are missing, as, for example, its verb, then we expect that only a more thorough investigation of the knowledge-representation portion of our network can clarify the extent to which such sentences can still be handled.

One issue of particular importance is the choice of prior probabilities for those variables $X_i$ for which $P(X_i) = 0$. Our experiments have indicated that
values of the order of $10^{-2}$ are generally suitable, and that the network's performance is not especially sensitive to small changes in these values. Similarly, recall that throughout this article, we have, on various occasions, referred to conditional probabilities as having "significantly high" or "significantly low" values, instead of the ideal 1 or 0, respectively. Whenever these extreme probabilities were needed (as, e.g., in the specification of AND and XOR nodes), we have found that, although Equation 7 has to be complied with, those values should not be too far apart from the ideal extremes, lest the equilibrium probabilities no longer provide a means for a clear choice between conflicting diagnostics. In our experiments we have chosen to utilize .995 and .005, respectively, in place of 1 and 0.

As we mentioned earlier, these tight probability values tend to seriously impair the performance of the stochastic simulation approach in its original version. Although the inclusion of simulated annealing accounts, as we have explained, for far more than just a speed-up under extreme probabilities, it might now be appropriate to provide some intuition as to how this speed-up is effected. Essentially, the incorporation of simulated annealing has the effect of distorting the probability in Equation 6 in such a way that, in a few first steps of the simulation, variable $X_i$ receives value 1 or 0 with equal probabilities (i.e., .5). As the simulation progresses, this distortion becomes less and less effective, and from a certain step onward Equation 6 is followed until, at the end, $X_i$ is assigned values with extreme probabilities. Intuitively, what this procedure does is to make equally probable in the first steps all transitions in the Markov chain that underlies the stochastic simulation, thereby avoiding "barriers" that would otherwise delay convergence.

6. CONCLUSIONS

We have dealt with the problem of lexical ambiguity resolution, and have proposed the use of a Bayesian-network model. As we see it, the approach of Bayesian networks is appealing in general for a number of reasons, particularly for allowing a rigorous description of how the various constituents of the problem interrelate (as opposed to connectionist systems), and for its inherent amenability to speed-up via distributed parallel processing (like connectionist systems).

Also relevant in the use of a Bayesian approach is its very nature as a probabilistic model, whose handling of uncertainties often implies some extra flexibility in treating situations that do not completely fit a preestablished template. As we remarked and exemplified earlier, to some extent our model can handle sentences whose syntactic structures are not entirely correct.

Our model is capable of handling ambiguities of both the syntactic and semantic types, and parts of the network can be constructed automatically.
from a context-free grammar and a set of words from a dictionary, along with the possible grammatical categories of each word. The remainder of the network is built from the possible meanings of each word, and comprises nodes for knowledge representation. For a given sentence, the input to the network consists of a sequence of words, each with its possible grammatical categories. Disambiguation is achieved by solving the model for the joint probability of the diagnostic nodes given the input.

Owing to its complexity, the model can only be solved by approximation techniques. We have employed parallel stochastic simulation to this end with good performance results, which together with the simulation’s convergence properties makes the model and the method of solution very promising. The stochastic simulation techniques we have employed are general, and apply to any Bayesian-network model. Nevertheless, during the simulation, certain patterns occur much more frequently than others: For example, the nodes in the parsing network can, in some cases, begin to give hints of the final parse tree much earlier than the eventual convergence of the entire network. This could be used to accelerate the simulation by updating certain portions of the network less and less frequently, and constitutes the subject of current research, involving investigations in the area of distributed parallel algorithms.

One should bear in mind that, although in principle a parsing network can be constructed for any context-free grammar, we have indicated throughout this article that some structural features of the grammar are of fundamental importance in our approach to lexical disambiguation. Specifically, a group of productions rewriting \( S \) indicating the possible arrangements of verb cases in the coarse structure of the sentence is needed. It is conceivable that, depending on the types of ambiguity we want to treat, this group of productions may become too large to be handled efficiently, as, for example, when we consider the problem of structural disambiguation. In such cases, a more suitable approach would be to allow the nonterminal \( S \) to appear in the right-hand side of productions, and then extract semantically relevant information from intermediate levels in the parsing network. The approach we described in this article does not handle such cases, which constitute the subject of ongoing research.

REFERENCES


APPENDIX
THE PARALLEL ALGORITHM FOR STOCHASTIC SIMULATION

In a parallel algorithm for stochastic simulation, a concurrent process $p_i$ is associated with each variable $X_i$, and assigned the responsibility of updating that variable periodically. Each $p_i$ stores the conditional probabilities $P[X_i|x_j; X_j \in P(X_i)]$ locally. A variable $X_i$ is updated according to Equation 6, where the values of the variables in $P(X_i)$, $C(X)$, and $M(X)$ are used. This dependency of $X_i$'s value on the values of variables in its neighborhood $N(X)$ implies the constraint that no two variables $X_i$ and $X_j$ may be updated concurrently if $X_j \in N(X_i)$ (Barbosa & Gafni, 1989).

Let $H=(X, E)$ be the undirected graph obtained by creating edges in $E$ for all pairs of neighbors. This graph is simply an extension of the Bayesian network's underlying undirected graph to include edges between mates. So the constraint on the concurrent updating of variables is equivalent to requiring that no two variables be concurrently updated if they are neighbors in $H$.

A distributed protocol that prevents this constraint from being violated is the edge-reversal protocol given by Barbosa and Gafni (1989), which functions as follows. Initially, the edges in $E$ are turned into directed edges by an arbitrary acyclic orientation (these directions imposed on the edges have
nothing to do with the directions of edges in the Bayesian network). Sinks in this acyclic orientation are necessarily nonneighbors of one another, and may be updated concurrently. Once this is done, the direction of all edges incident to these sinks is reversed; they become sources, but new sinks are created, which can then be updated. This protocol is totally asynchronous, and messages between processes are used to indicate the reversal of edge directions. So, after \( X_i \) has been updated by \( p_i \), reversal messages are sent to all processes whose variables are in \( N(X_i) \). These messages need not be the same for all neighbors: to those in \( P(X_i) \), conditional probabilities are sent; those in \( C(X_i) \) receive \( X_i \)'s value; those in \( M(X_i) \) receive a signal that merely indicates edge reversal.

In the following we outline the main steps to be taken by \( p_i \) in this parallel stochastic simulation, assuming that an initial acyclic orientation is already available. According to this orientation, variables are "upstream" or "downstream" with respect to one another, and this changes as the orientation changes. For simplicity of notation, we let a variable's identification, say \( X_j \), denote a local variable at \( p_i \) where \( X_j \)'s value is stored as well.

1. Mark the upstream neighbors of \( X_i \).
2. Send the value of \( X_i \) to all \( p_j \) such that \( X_j \in C(X_i) \).
3. Wait to receive from all \( p_j \) such that \( X_j \in P(X_i) \) a message, to be stored in \( X_j \).
4. For \( X_j \in P(X_i) \), let
   \[ \pi^i(j) = P[X_i | X_j, x_k; k \neq j, X_k \in P(X_i)] \]
   and
   \[ \pi^o(j) = P[X_i | X_j = \neg X_j, x_k; k \neq j, X_k \in P(X_i)] \].

If \( X_i = 1 \), then send \( \pi^i(j) \) and \( \pi^o(j) \) to all \( p_j \) such that \( X_j \in P(X_i) \) and \( X_j \) is unmarked. Send \( 1 - \pi^i(j) \) and \( 1 - \pi^o(j) \), otherwise.
5. Wait to receive from all \( p_j \) such that \( X_j \in C(X_i) \) and \( X_j \) is marked two consecutive messages, to be stored in \( \rho_j^i \) and \( \rho_j^o \), respectively.
6. Repeat \( K \) times:
   6.1. Wait to receive from all \( p_j \) such that \( X_j \in N(X_i) \) and \( X_j \) is downstream from \( X_i \):
      - one message, to be stored in \( X_j \), if \( X_j \in P(X_i) \);
      - two consecutive messages, to be stored in \( \rho_j^i \) and \( \rho_j^o \), respectively, if \( X_j \in C(X_i) \);
      - one message, to be ignored, if \( X_j \in M(X_i) \).
   6.2. Let
   \[ a = P[X_i | x_j; X_j \in P(X_i)] \prod_{X_j \in C(X_i)} \rho_j^i \]
   and
   \[ b = \{1 - P[X_i | x_j; X_j \in P(X_i)]\} \prod_{X_j \in C(X_i)} \rho_j^o. \]
If $X_i \notin \mathbf{E}$, then assign value 1 to $X_i$ with probability $a/(a + b)$, otherwise assign value 0.

6.3. Let $\pi_i^1(j)$ and $\pi_i^2(j)$ be given as in Step 4. If $X_i = 1$, then send $\pi_i^1(j)$ and $\pi_i^2(j)$ to all $p_j$ such that $X_j \in \mathbf{P}(X_i)$; send $1 - \pi_i^1(j)$ and $1 - \pi_i^2(j)$, otherwise. Send the value of $X_i$ to all $p_j$ such that $X_j \in \mathbf{C}(X_i)$, and a signal to all $p_j$ such that $X_j \in \mathbf{M}(X_i)$.

7. Wait to receive a message from all $p_j$ such that $X_j \in \mathbf{N}(X_i)$ and $X_j$ is marked, and ignore it.

Step 6 is the main iterative step of the simulation, where $K$ is the number of times each variable is to be updated. Steps 1–5 are initialization steps, and Step 7 ensures proper termination.

The procedure we have used for incorporating simulated annealing into the simulation applies to Step 6.2 of the algorithm. Instead of assigning value 1 to $X_i$ with probability $a/(a + b)$, do it with probability

$$r = \frac{a^{1/T}}{a^{1/T} + b^{1/T}}$$

$$= \frac{1}{1 + \exp[\ln(b/a)/T]}$$

where $T = T_0$ (a very high value) for the first step of the simulation, and for the $k$th step, $1 < k \leq K$, $T = \beta^{k-1}T_0$, where $0 < \beta < 1$ and $K$ is, as we have discussed, the number of iterations. Clearly, $r \approx .5$ for the first step, then tends to $a/(a + b)$ as $T \rightarrow 1$, and then acquires an extreme value, depending on the sign of $\ln(b/a)$, as $T$ is reduced toward zero.