AALBORG UNIVERSITY

Optimal Decomposition of Probabilistic Networks by Simulated Annealing

by

Uffe Kjærulff

December 1991

To be referenced as:

Kjærulff, U. (1992). Optimal decomposition of probabilistic networks by simulated annealing, *Statistics and Computing* **2**: 7–17.

INSTITUTE FOR ELECTRONIC SYSTEMS
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
Fredrik Bajers Vej 7E — DK 9220 Aalborg Ø — Denmark
Tel.: +45 98 15 85 22 — TELEX 69 790 aub dk

Optimal Decomposition of Probabilistic Networks by Simulated Annealing

Uffe Kjærulff
Department of Mathematics and Computer Science
Institute of Electronic Systems
Aalborg University
DK-9220 Aalborg, Denmark
uk@iesd.auc.dk

18 December, 1991

Abstract

This paper investigates the applicability of a Monte Carlo technique known as 'simulated annealing' to achieve optimum or sub-optimum decompositions of probabilistic networks under bounded resources. High-quality decompositions are essential for performing efficient inference in probabilistic networks. Optimum decomposition of probabilistic networks is known to be NP-hard (Wen 1990). The paper proves that cost-function changes can be computed locally, which is essential to the efficiency of the annealing algorithm. Pragmatic control schedules which reduce the running time of the annealing algorithm are presented and evaluated. Apart from the conventional temperature parameter, these schedules involve the radius of the search space as a new control parameter. The evaluation suggests that the inclusion of this new parameter is important for the success of the annealing algorithm for the present problem.

Keywords: Simulated annealing, global optimization, Monte Carlo algorithms, graph decomposition, probabilistic networks, NP-complete problems

1 Introduction and terminology

Major advances in probabilistic inference methods based on graphical representations have been realized in recent years (Jensen, Lauritzen & Olesen 1990, Lauritzen & Spiegelhalter 1988, Pearl 1988, Shachter 1988, Shafer & Shenoy 1988, Suermondt & Cooper 1988). However, probabilistic inference has the inherent disadvantage of being NP-hard (Cooper 1987). Hence probabilistic reasoning may require excessive computational resources. By exploiting the conditional independence relations among the discrete random variables of a probabilistic network the underlying joint probability space may be decomposed into smaller subspaces corresponding to cliques in a triangulated graph (Lauritzen

& Spiegelhalter 1988, Spiegelhalter 1986). Probabilistic networks are also frequently referred to as 'belief networks', 'Bayes networks', 'relevance diagrams', 'causal networks' and 'influence diagrams'.

A probabilistic network is built on a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$ of vertices $V = \{1, \dots, n\}$ interconnected by a set of edges $E \subseteq V \times V$ of ordered pairs of distinct vertices. If $(u, v) \in E$ and $(v, u) \notin E$, then the edge from u to v is *directed*. If both $(u, v) \in E$ and $(v, u) \in E$, then the edge between u and v is *undirected*. If there is an edge between u and v, u and v are *adjacent* or *neighbours*. The neighbours of u are denoted $\mathrm{adj}(u)$. If there is a directed edge from u to v, u is called a *parent* of v and v a *child* of u. A graph containing only undirected edges is said to be *undirected*; otherwise it is *directed*. Unless explicitly stated, graphs are assumed to be undirected.

A subset $A \subseteq V$ induces a subgraph $\mathcal{G}_A = (A, E_A)$ of $\mathcal{G}$, where $E_A = E \cap (A \times A)$. A graph is *complete* if all vertices are pairwise adjacent. A subset $A \subseteq V$ is complete if it induces a complete subgraph. A maximal complete subset is called a *clique*. A *path* $\langle u = u_1, \dots, u_k = v \rangle$ is an ordered sequence of distinct vertices such that $\{u_i, u_{i+1}\} \in E$ for $i = 1, \dots, k-1$. An m-cycle is a path of length m with the exception that $u = v$. A *chord* in an m-cycle is an edge $\{u_i, u_j\}$ such that $1 < |i-j| < m-1$ (i.e., $\{u_i, u_j\}$ connects the two non-consecutive vertices u_i and u_j in the cycle).

A graph is *triangulated* (also: *decomposable* or *chordal*) if any cycle of length greater than 3 has a chord. A vertex $v \in V$ is said to be *eliminated* from a graph $\mathcal{G} = (V, E)$ when the edges $(\mathrm{adj}(v) \times \mathrm{adj}(v)) \setminus E$ are added to E such that $\mathrm{adj}(v)$ becomes complete, the edges between v and its neighbours are removed from E, and v is removed from V. The set, say T, of edges added by eliminating all vertices in V in any order is called a *triangulation* of $\mathcal{G}$ as $\mathcal{G}' = (V, E \cup T)$ is triangulated (Rose 1970). The edges of T are called *fill edges* or *fill-ins*. Formally, an ordering of V is a bijection $\# : V \leftrightarrow \{1, \dots, n\}$. $\mathcal{G}_\#$ is an *ordered graph*. We shall refer to $\#$ as both the bijection and the actual ordering determined by it, and when it appears clearly from the context that a graph is ordered we shall often drop the $\#$-index. The triangulation $T(\mathcal{G}_\#)$ is the set of edges produced by eliminating the vertices of $\mathcal{G}$ in the order defined by $\#$.

A probabilistic network can be decomposed by sequentially performing three operations on the underlying DAG: First, transform the graph into its corresponding *moral graph* (Lauritzen & Spiegelhalter 1988) by adding undirected edges between pairs of parents with one or more common children. Next, drop the orientation of the edges; and finally, triangulate the moral graph. The cliques of the triangulated graph then define a decomposition of the probabilistic network.

As indicated above, triangulating a graph is essentially a problem of establishing an elimination ordering. Several algorithms have been suggested for establishing elimination orderings which have certain desired properties (Fujisawa & Orino 1974, Kjærulff 1990, Rose, Tarjan & Lueker 1976, Tarjan & Yannakakis 1984). As the task of establishing an elimination ordering may be perceived as a sorting problem, such algorithms are quite often referred to as *sorting algorithms*. Since most of the algorithms apply some sort of search strategy, the term *search algorithms* is also frequently used.

The present paper deals with the problem of achieving approximate optimum decompositions of probabilistic networks. In this context the notion of optimality relates to the sum total of the state-space sizes of the cliques of the decomposable network. Wen (1990) has proved this optimality task to be NP-hard.

In Section 2 the actual sorting task is formulated in terms of simulated annealing and it is proved that cost-function changes can be computed locally, which is essential to the efficiency of the annealing algorithm. Section 3 suggests and extensively evaluates some pragmatic control schedules to reduce the running time of the algorithm. Apart from the conventional temperature parameter these schedules involve the radius of the search space as a new control parameter. The evaluation is based on four different graphs. Finally, Section 5 summarizes the results and discusses the applicability of simulated annealing for decomposing probabilistic networks by comparing the results of the evaluation with those of other search algorithms.

2 Graph triangulation by simulated annealing

Simulated annealing is a stochastic optimization algorithm used for finding global minimum cost configurations of NP-complete combinatorial problems with cost functions having many local minima. The algorithm was first described by Kirkpatrick, Gelatt & Vecchi (1983) and may be thought of as a combination of deterministic descent search and a Monte Carlo method. While a strictly descent algorithm often terminates at a local minimum, the addition of Monte Carlo techniques provides a way of escaping local minima. That is, the annealing algorithm accepts cost-function increases with positive probability dependent on the state of the search process as described below. The act of accepting cost-function increases is often referred to as *hill climbing*. In the original formulation of the annealing process the evolution of the state of the search process is based solely on a control schedule involving exponential decrease of a 'temperature' parameter. In this paper we suggest extending this control schedule by an additional parameter such that the 'step size' of the search decreases along with the temperature. This approach has the advantages of narrowing the focus of attention as attractive subspaces are being identified, thereby avoiding large and obviously unprofitable transitions; and of significantly reducing the running time of the annealing algorithm.

For a given optimization problem to be suitable for the annealing algorithm it should be possible to describe it in terms of a *solution space*, S, the elements of which all relate to some cost-function value $c(s)$, $s \in S$, and which are connected by some appropriate neighbouring structure. That is, S has a certain topology which has the essential feature that for every pair of configurations $s_i, s_j \in S$, s_j should be reachable from s_i. Reachability is defined as a series of *transitions*, $s_k \to s_l$, each of which consists of a *perturbation* imposed on configuration s_k such that it changes to a neighbouring configuration s_l. The *distance* between two configurations $s_i, s_j \in S$ is defined as the smallest number of transitions required to reach s_j from s_i. The *radius* of a space S is defined as the maximum distance between two configurations in S. Of course, the radius depends on the perturbation scheme employed. In our case S is the set of all possible (elimination) orderings of V in a graph $\mathcal{G} = (V, E)$ (i.e., $|S| = n!$, where $n = |V|$); and a transition $s_i \to s_j$ is simply defined as the interchange of two vertices in ordering s_i, resulting in ordering s_j.

Henceforth we shall refer to the size of the total state-space of a decomposable probabilistic network as the *weight* of the corresponding triangulated graph $\mathcal{G}^t = (V, E \cup T(\mathcal{G}_\#))$, where $\mathcal{G}_\# = (V, E)$. For convenience the weight, $w(\mathcal{G}^t)$, of $\mathcal{G}^t$ is defined as the base-2 logarithm of the state-space size. As our endeavour is to minimize the weight of $\mathcal{G}^t$ the cost-function is defined as

$$c(\#) = w(\mathcal{G}^t) \tag{1}$$

and we should therefore find an optimum ordering $\#^*$ such that $c(\#^*) \leq c(\#)$ for all $\# \in S$.

The annealing algorithm starts at some, possibly random, configuration $\#_i$ $(i = 0)$, selects two vertices v and w (according to a perturbation scheme as discussed below), and computes the cost $c(\#_p)$ of the potential new configuration $\#_p$ obtained by interchanging v and w. Now, $\#_p$ is accepted if

$$c(\#_p) \leq c(\#_i) - t_i \log u, \tag{2}$$

where u is a uniformly distributed random number on the interval $]0; 1]$ and t_i is the *temperature*, a control parameter, at time i. This means that an uphill step of λt_i will be allowed with probability $e^{-\lambda}$. In other words, the new configuration $\#_{i+1}$ is set to $\#_p$ with probability

$$p = \min\left\{1, e^{-\Delta/t_i}\right\} \tag{3}$$

and to $\#_i$ with probability $1 - p$, where $\Delta = c(\#_p) - c(\#_i)$. Large values of t_i allow large cost-function increases, and small values allow only small increases.

The initial temperature, t_0, should be set to some suitable large value such that all perturbations are accepted (this ensures free motion in the search space, that is, all configurations can be reached). Thus, if $\#_i$ and $\#_j$ are neighbouring configurations in S and if we define

$$U = \max_{i,j} |c(\#_i) - c(\#_j)| \tag{4}$$

to be the maximum cost-function increase, then an uphill step of U should be accepted with probability close to 1. If $U = \alpha t_0$ we get the rule

$$e^{-\alpha} \approx 1 \tag{5}$$

meaning that t_0 should be large compared to U or $0 < \alpha \ll 1$.

Lundy & Mees (1986) concluded in their study on convergence properties of the annealing algorithm that the temperature should be lowered exponentially as

$$t_{i+1} = \frac{t_i}{1 + \beta t_i}, \tag{6}$$

where $\beta \ll 1/U$, and that the final temperature, t_f, should (essentially) obey the following inequality:

$$t_f \leq \frac{\varepsilon}{\log|S|}, \tag{7}$$

where ε is the acceptable error (i.e., $\varepsilon = c(\#_f) - c(\#^*)$).

An equivalent formulation of Equation 6 is

$$t_{i+1} = \frac{t_0}{1 + i\frac{\gamma}{\alpha}}, \tag{8}$$

where $\alpha = U/t_0$, $0 < \gamma \ll 1$ and $\gamma/\alpha = \beta t_0$. By solving Equation 8 for i it appears that the number of iterations required to reach a temperature t_{i+1} is less than $U/(\gamma t_{i+1})$ and hence to reach the final temperature t_f takes I iterations, where

$$I < \frac{U \log|S|}{\gamma\varepsilon}. \tag{9}$$

2.1 An additional control parameter

Initially, essentially free motion in the search space, S, should be allowed to avoid becoming trapped in local minima at an early stage. Apart from allowing large uphill steps it is also important that the configurations visited in the initial phase are uniformly distributed over S such that the topographic map of S (w.r.t. configuration costs) generated during the search process represents the entire search space. However, in addition the density of configurations visited in S should be sufficiently high to guarantee that all major basins are adequately represented. Obviously, assuming uniform distribution, the density of configurations visited depends solely on the number of iterations of the annealing algorithm in the initial phase (i.e., when virtually any uphill step is accepted). Of course, the uniformity of the distribution may be ensured by lowering the temperature sufficiently slowly (i.e., spending a long time in the initial phase). This may, however, not be the optimal schedule as the density of configurations visited may then become unnecessarily high. Thus we need an additional control parameter to ensure a uniform distribution of configurations without being forced to lower the temperature unnecessarily slowly.

An obvious candidate providing this feature is the radius of S, that is, by maintaining a small radius the distance between any two configurations becomes small, implying a more uniform distribution of configurations visited given a fixed number of iterations. Thus, in the initial phase the radius of the search space should be kept as small as possible to allow large steps in S. Later, when the algorithm has found an attractive subspace (basin) containing low-cost configurations, it seems more convenient to have a large radius (i.e., narrowing down the focus of attention by reducing the number of neighbours for each configuration allowing only small steps in S). The process of increasing the radius of S might be thought of as a magnification process allowing a more fine-grained search process (cf. the inflation of a balloon).

Apart from narrowing the focus of attention, a large radius (small step size) has the additional advantage of significantly reducing the running time of the annealing algorithm, as will become clear in the following.

2.2 Perturbation scheme

The neighbouring structure of the search space, S, is determined by the perturbation scheme employed. That is, given a configuration, $\# \in S$, the set of neighbouring configurations, $N(\#) \subseteq S$, is defined as the set of all possible configurations which can be obtained by applying the perturbation scheme on $\#$. Thus the radius of S is determined by the perturbation scheme. Initially, the only constraint put on the perturbation scheme is that a perturbation is defined to be the interchange of only two vertices u and v in a configuration. With that constraint in mind, the scheme providing the smallest possible radius chooses the pair u, v completely arbitrarily from a configuration, $\#$, and the scheme providing the largest possible radius chooses a pair u, v for which $|\#(u) - \#(v)| = 1$ (i.e., u and v are adjacent in $\#$). For both schemes the pair of configurations $\#_i, \#_j \in S$ between which the distance equals the radius of S is characterized by the one being the reverse of the other (i.e., $\#_i(v) \neq \#_j(v)$ for all $v \in V$ except for at most one vertex). Thus for the two perturbation schemes the radii r_{min} and r_{max} of S are given by

$$r_{min} = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (n-1)/2 & \text{if } n \text{ is odd} \end{cases}, \tag{10}$$

$$r_{max} = \sum_{i=1}^{n-1} n - i = \frac{n^2 - n}{2}. \tag{11}$$

Apparently $r_{max} = (n-1)r_{min}$ if n is even and $r_{max} = nr_{min}$ if n is odd, and hence there is a substantial difference between r_{min} and r_{max} even for a moderate number of vertices. However, both of these perturbation schemes may be described in terms of the general scheme for which the interchange of u and v is restricted by the condition $|\#(u) - \#(v)| < \omega$, where $2 \leq \omega \leq n$. That is, we define a 'window' of width ω within which the perturbations take place. When the annealing process starts we set $\omega = n$ and decreases it gradually until it reaches the lower bound $\omega = 2$. A reasonable way of narrowing the window width (i.e., increasing the radius of the search space — decreasing the step size) could be to follow a reduction scheme similar to the one applied to the temperature, i.e., exponential reduction.

2.3 Local computation of cost-function change

It is crucial to the success of the annealing algorithm that the cost of a candidate new configuration, $\#_p$, can be computed efficiently. That is, with

$$c(\#_p) = c(\#_i) + \Delta$$

the cost-function increase, Δ, should be locally computable. For that requirement to be fulfilled we must develop a method for identifying the cliques not altered by a given perturbation. By Theorem 1, we find that only cliques produced by eliminating vertices $\#^{-1}(i), \ldots, \#^{-1}(j)$, $i < j$, need to be re-established when interchanging $\#^{-1}(i)$ and $\#^{-1}(j)$ in the elimination order given by $\#$.

In the following we denote by $\mathcal{G}_i = (V_i, E_i)$, where $i = 1, \ldots, n$, the ordered graph obtained by eliminating in order the vertices $\#^{-1}(1), \ldots, \#^{-1}(i) \in V$ from the ordered graph $\mathcal{G} = \mathcal{G}_0 = (V, E)$. By $C_i(\mathcal{G})$ we denote the clique, if any, produced by eliminating vertex $\#^{-1}(i)$ in $\mathcal{G}_{i-1}$. Let $C_i(\mathcal{G}) = \emptyset$ if the elimination of $\#^{-1}(i)$ does not produce a clique. In the same spirit we let $T_i(\mathcal{G})$ denote the fill edges produced by eliminating vertex $\#^{-1}(i)$ in $\mathcal{G}_{i-1}$.

Lemma 1 (Rose *et al.***)** *Let $\mathcal{G} = (V, E)$ be an ordered graph. Then $\{u, v\} \in T(\mathcal{G})$ if and only if there is a path $\langle u, u_1, \ldots, u_k, v\rangle$ in $\mathcal{G}$ such that $\#^{-1}(u_i) < \min(\#^{-1}(u), \#^{-1}(v))$ for all $1 \le i \le k$.*

For a proof of Lemma 1, see Rose et al. (1976).

Lemma 2 *Let $\mathcal{G}_{\#_a} = (V, E)$ and $\mathcal{H}_{\#_b} = (V, E)$ be ordered graphs, where $\#_a^{-1}(i) = \#_b^{-1}(i)$ for all $k < i \le n$ with k being an arbitrary non-negative integer. Then $\mathcal{G}_i = \mathcal{H}_i$ for all $k \le i \le n$.*

Proof: Clearly $\mathcal{G}_i$ and $\mathcal{H}_i$ consist of the same set of vertices, and from Lemma 1 it is equally clear that they also have identical sets of edges. □

Theorem 1 *Let $\mathcal{G}_{\#_a} = (V, E)$ and $\mathcal{H}_{\#_b} = (V, E)$ be ordered graphs such that for $u, v \in V$, $\#_a(u) = i < \#_a(v) = j$ and for $1 \le k \le n$*

$$\#_a^{-1}(k) = \begin{cases} \#_b^{-1}(j) & \textit{if } k = i, \\ \#_b^{-1}(i) & \textit{if } k = j, \\ \#_b^{-1}(k) & \textit{otherwise.} \end{cases}$$

Then $C_k(\mathcal{G}) = C_k(\mathcal{H})$ for $1 \le k < i$ and $j < k \le n$.

Proof: The proof consists of two parts: First, we shall prove that all cliques $C_k(\mathcal{G})$ for $1 \le k < i$ and $j < k \le n$, are unaffected by the interchange of u and v in the elimination ordering. Knowing that the triangulation of a graph determines how the cliques of the resulting triangulated graph are composed the problem is equivalent to the problem of proving that the fill edges $T_k(\mathcal{G}) = T_k(\mathcal{H})$ for all $1 \le k < i$ and for all $j < k \le n$. For $1 \le k < i$ the problem is trivial as the set of fill edges produced by eliminating a vertex solely depends on the original edges E and on the fill edges already produced. The validity of the equation for $j < k \le n$ follows immediately from Lemma 1.

Next, we shall prove that for every new clique $C_k(\mathcal{H})$, $i \le k \le j$, C_k cannot be a superset of a clique $C_l(\mathcal{G}) = C_l(\mathcal{H})$, $1 \le l < i$ and $j < l \le n$. For $1 \le l < i$ the problem is trivial as $\#_a^{-1}(l) = \#_b^{-1}(l) \in C_l$ and hence cannot be a member of C_k. To prove it for the case where $j < l \leq n$, we assume there is a clique C_l such that $C_l \subset C_k$. For this to be true the vertex $\#_b^{-1}(k)$ (the elimination of which created C_k) must have been adjacent to all vertices in C_l in $\mathcal{H}_{k-1}$. By Lemma 2 we see that there would also have been such a vertex in $\mathcal{G}_{k-1}$ and hence C_l would not have been a clique. $\square$

The consequence of Theorem 1 may be explained as follows. Let $\mathcal{G}_\#$ be an ordered graph, let $\mathcal{G}^t = (V, E)$ be the corresponding triangulated graph produced by eliminating the vertices in order $\#$, and let $\mathcal{G}_A^t = (A, E_A)$ be the subgraph of $\mathcal{G}^t$ induced by $A \subseteq V$ such that $\#^{-1}(i) \in A$ for all $a \leq i \leq a + k$, where $a = \min_{v \in A} \#(v)$ and $|A| = k + 1$. That is, A consists of a set of vertices comprising a consecutive subset of the ordering $\#$; or, more generally, A consists of a set of vertices comprising a consecutive subset of an ordered set defining an elimination order corresponding to the fill edges of $\mathcal{G}^t$. Note that $\mathcal{G}_A^t$ is triangulated. Now the fill edges of $\mathcal{G}_A^t$ are removed and the resulting graph is retriangulated resulting in the graph (A, E_A^*). By Theorem 1 we find that substituting E_A in $\mathcal{G}^t$ by E_A^* does not compromise the chordality of the graph nor alter the cliques produced by the elimination of vertices $V \setminus A$. This means that modifications of a triangulated graph may be carried out locally.

It is clear from the above discussion that the time complexity of the annealing algorithm is strongly dependent on the window width, ω. In the worst case the graph must be totally retriangulated for $\omega = n$. On the other hand, for $\omega = 2$ a perturbation affects at most two cliques. Thus an optimal control schedule under bounded resources involves a trade-off between mobility (ω large) and speed (ω small).

3 Annealing in practice

This section begins with a discussion of the need to compromise the control schedule presented in Section 2 due to excessive time complexity, and then turns to pragmatic ways of reducing the running time of the annealing algorithm. An extensive evaluation is carried out to identify an optimal control schedule under bounded resources.

3.1 The excessive time complexity of ideal annealing

To get a feeling for the time complexity of the annealing algorithm, given the control schedule in Section 2, assume $\alpha = 10^{-2}$ (satisfying Approximation 5) and $\gamma = 10^{-4}$ (so that $\gamma/\alpha \ll 1$). By accepting an error $\varepsilon = 10^{-3}U$ we find that $I \approx 10^7 n(\log n - 1)$ which for $n = 50$ is approximately 1.5×10^9. An actual implementation of the annealing algorithm in the C programming language gave a maximum speed ($\omega = 2$) of approximately 1000 iterations per second on a Sun SPARCstation SLC. Even at that speed the example above would still require a running time of more than 400 hours. Therefore, in order to obtain reasonable response times from the annealing algorithm, we have to compromise the convergence properties of the algorithm and adopt pragmatic control schedules which trade off efficiency against precision.

3.2 Pragmatic control schedules

There are three parameters which determine the number of iterations required: the initial temperature, t_0; the cooling rate controlled by γ; and the final temperature, t_f.

As discussed in Section 2, to allow essentially free motion in the search space, S, in the initial phase, the probability that any uphill step is accepted should be close to 1 implying the rule $t_0 \gg U$. The validity of this rule might, however, be highly dubious as it depends heavily on the topography of S. If, for example, the search space does not contain 'highland basins' with no configurations having costs close to the global minimum there is no need for a high initial temperature.

If the phenomenon of highland basins is scarce and the basin depths are moderate then the cooling rate may be increased without seriously compromising the convergence properties of the algorithm.

Specifying a reasonable acceptable error ε often leads to extremely small values of both t_f and temperature changes $t_i - t_{i+1}$ when t_i is close to t_f. This means that the algorithm may have reached configurations $\#$ for which $c(\#) - c(\#^*) \leq \varepsilon$ long before t_i has reduced to t_f. Further, by constantly keeping in mind the least-cost configuration visited, t_f can definitely be set at a considerably higher value than recommended by Inequality 7.

From the above discussion it appears that the total number of attempted configuration transitions before reaching the final temperature, given a fixed control schedule, depends on the size, $|S|$, of the search space and the maximum cost difference, U, between any two configurations. Concerning the problem of low-cost graph triangulations, this means that in practice there is a limit in terms of number of vertices and dispersion of vertex weights (loosely speaking) above which the annealing algorithm is inapplicable. This is, of course, a rather unsatisfactory situation which can be avoided by allowing more 'coarse-grained' solutions for large graphs. That is, if we adopt such a rather pragmatic approach the stopping criterion must be defined such that it becomes independent of $|S|$ and U. From Equation 8 we get

$$I = \frac{\alpha}{\gamma}\left(\frac{t_0}{t_f} - 1\right) = \frac{1}{\gamma}\left(\frac{U}{t_f} - \alpha\right), \tag{12}$$

which says that if the final temperature is defined in terms of U the number of iterations required depends solely on the cooling rate. One way of defining t_f is to say that it is the temperature at which an uphill step of cU, where $0 < c < 1$, is accepted with probability p, where $0 < p < 1$. Recalling that an uphill step of $\lambda t_f = cU$ is accepted with probability $e^{-\lambda} = p$ at temperature t_f, we get

$$I = -\frac{1}{\gamma}\left(\frac{\log p}{c} + \alpha\right), \quad p < e^{-\alpha c} \tag{13}$$

as desired. By fixing α and c we observe that the cooling rate may be given either explicitly via γ or implicitly via I and p. Similarly, I may be given explicitly or implicitly via γ and p.

The annealing algorithm performs a search in the graph of configurations (the solution space) which is normally controlled solely by the temperature parameter. However, by allowing a gradual increase of the radius of the graph during the search process we have introduced an additional control parameter as discussed in Section 2. Ideally the final temperature, t_f, should be low (relative to the acceptable error, ε) and the final radius should be large (i.e., ω should be small). However, to determine the effect of the two control parameters separately a series of test runs must be conducted. One of the parameters is kept at a fixed value which we define to be either small, medium or large.

The three temperature levels are defined in the same way as t_f above with $c = 0.1$ and probabilities $p_s = 0.05$, $p_m = 0.5$, and $p_l = 0.95$. Thus

$$t_s = \frac{U}{10 \log 20}, \tag{14}$$

$$t_m = \frac{U}{10 \log 2}, \tag{15}$$

$$t_l = \frac{U}{-10 \log 0.95}. \tag{16}$$

The radius levels are defined as

$$r_s = r_{\min}, \tag{17}$$

$$r_m = \frac{r_{\min} + r_{\max}}{2}, \tag{18}$$

$$r_l = r_{\max}. \tag{19}$$

As the radius of the search space is determined by ω, r is not an explicit control parameter. So we need an algebraic relationship between r and ω. This is, however, rather hard to find if we insist on employing the perturbation scheme described above. By accepting a slightly different scheme based on a 'move-and-push' procedure (i.e., an element $x = L[i]$ at position i in sequence L is moved to position j by pushing each element $L[k]$ to position $k + \delta$ for $k = i + \delta, i + 2\delta, \ldots, j$, where $\delta = -1$ if $i < j$ and $\delta = 1$ if $i > j$, and placing x at position j), it can be shown that ω can be expressed in terms of r and n as

$$\omega = \left\lceil \frac{(n-1)^2}{2r - n + 1} + 0.5 \right\rceil. \tag{20}$$

The interchange and the 'move-and-push' schemes are identical for $\omega = 2$. For $\omega = n$ 'move-and-push' yields $r_{\min} = n - 1$. Thus

$$r_m = \frac{r_{\min} + r_{\max}}{2} = \frac{n^2 + n - 2}{4}, \tag{21}$$

$$\omega_m = \left\lceil \frac{2(n-1)^2}{n(n-1)} + 0.5 \right\rceil = \left\lceil 2\left(1 - \frac{1}{n}\right) + 0.5 \right\rceil. \tag{22}$$

The free control parameter may be subjected to a slow, medium or fast reduction (temperature) or increase (radius) schedule.

The rate by which the temperature parameter is decreased is determined by the ratio γ/α. Recalling that $e^{-\alpha}$ expresses the probability of accepting uphill steps of U we select $\alpha = 0.1$ (i.e., with probability $p \approx 0.9$ an uphill step of U is accepted when $t = t_0$). The rule of essentially free motion in the initial phase is not violated, which is clearly demonstrated by the fact that at temperature t_i uphill steps of U are accepted with probability e^{-U/t_i} (i.e., even with the temperature lowered to $t_0/10$ more than one-third of the potential uphill steps of size U are accepted). So, with α fixed the cooling rate is determined by γ. A slow cooling rate is defined such that the temperature equals t_l when only 10% of the iterations are left, for a medium rate the final temperature is defined to be reached at the last iteration, and we define a fast cooling rate such that the temperature equals $t_{\,s}$ when only 10% of the iterations are done. In the case of a medium cooling rate we set $t_{\,f} = t_{\,s}$. Thus

$$\gamma_{\,s} = -\frac{1}{0.9I}\left(\frac{\log p_l}{c} + \alpha\right) = \frac{-10\log 0.95 - \alpha}{0.9I}, \tag{23}$$

$$\gamma_{\,m} = -\frac{1}{I}\left(\frac{\log p_{\,s}}{c} + \alpha\right) = \frac{10\log 20 - \alpha}{I}, \tag{24}$$

$$\gamma_{\,f} = -\frac{1}{0.1I}\left(\frac{\log p_{\,s}}{c} + \alpha\right) = \frac{10\log 20 - \alpha}{0.1I}. \tag{25}$$

Even though we argued above that ε can be set to a value which is considerably higher than the actual acceptable error, provided the least-cost configuration visited is remembered, it seems hard to justify $t_{\,f} = t_{\,s}$ as ε then becomes comparable with U. The discussion of whether $t_{\,f} = t_{\,s}$ is a reasonable choice is continued in Section 4.

Now we should assign values to either I or γ (recall that values of p have already been determined). As I is a derived parameter which is defined in terms of the cooling rate and the final temperature we naturally choose the latter. Knowing that $\gamma/\alpha \ll 1$ or $\gamma \ll \alpha$ a value of $\gamma = 10^{-3}$ seems reasonable, and we find from Equation 24 that

$$I = \frac{10\log 20 - \alpha}{\gamma_{\,m}} \approx 10^4 \log 20 \approx 3 \times 10^4 \tag{26}$$

corresponding to running times in terms of minutes instead of hours or even days (cf. Section 3.1).

In the spirit of Equation 8 the window width, ω, is decreased exponentially as

$$\omega_{i+1} = \frac{\omega_0}{1 + i\tau}. \tag{27}$$

Thus the rates by which r is increased are determined by

$$\tau_{\,s} = \frac{\omega_0/\omega_{\,m} - 1}{I}, \tag{28}$$

$$\tau_{\mathrm{m}} = \frac{\omega_0/\omega_{\mathrm{s}} - 1}{0.8I}, \tag{29}$$

$$\tau_{\mathrm{f}} = \frac{\omega_0/\omega_{\mathrm{s}} - 1}{0.2I}. \tag{30}$$

Note that ω_{m} corresponds to r_{m}, $\omega_0 = n$, and ω_{s} corresponds to r_l. Equations 28–30 suggest that, at a slow rate the radius of the search space is medium when the annealing terminates, at a medium rate the radius is maximum when the annealing is 80% done, and at a fast rate it is maximum when 20% done.

Now, the only undefined parameter is the maximum cost-function increase, U, which, depending on the problem at hand, can be found either analytically or experimentally. For the graph-triangulation problem it seems very hard to find U analytically as it depends on a number of parameters such as the size and density of the graph, the homogeneity of the graph w.r.t. the distribution of edges and number of states of the individual vertices, and the window width, ω. So we will stick to the experimental approach.

4 Evaluation

In order to find the optimal schedule among the pragmatic control schedules presented above, a selected set of schedules is first tested on a particular graph and then a subset of these schedules which appear to produce the best results is then applied to three other graphs. Then, hopefully, we should be able to determine which control schedule(s) is (are) generally preferable for the network decomposition problem. Also the accuracy of the approximate solutions is discussed and compared to results obtained by other graph-triangulation algorithms.

4.1 Evaluating the pragmatic control schedules

In the first test cycle we select the control schedules for which at least one of the control parameters is varying, giving a total of 27 distinct combinations. As the target graph we use the Medianus-I network known from the MUNIN expert system (Andreassen, Jensen, Andersen, Falck, Kjærulff, Woldbye, Sørensen, Rosenfalck & Jensen 1989, Jensen, Andersen, Kjærulff & Andreassen 1987, Olesen, Kjærulff, Jensen, Jensen, Falck, Andreassen & Andersen 1989). The Medianus-I network consists of 43 vertices and 110 edges and describes relations between disorders, pathophysiological features, and measurements for the human median nerve. For Medianus-I we find $U \approx 7$. We cannot expect, of course, to find least-cost configurations with identical costs every time the algorithm is run. We thus have to run the algorithm several times for each control schedule. Table 1 shows the minimum, median, and maximum least costs found for the 27 different control schedules with ten runs for each schedule. The schedules have been ordered according to the median least-cost value which is the quantity we shall use in our comparison of the schedules. Note that the median cost is defined as the logarithm of the average state-space size. As the state-space size is the quantity of interest and as $E(\log s) \leq \log E(s)$ (cf. Jensen's inequality), where s denotes the state-space size, we focus on the median cost rather than the average cost.

	Min.	Med.	Max.		Min.	Med.	Max.
1. t_s, τ_m	22.63	22.71	22.92	15. γ_m, τ_f	22.71	23.41	24.40
2. γ_m, r_s	22.68	22.73	22.81	16. γ_m, r_l	22.69	23.46	24.23
3. t_s, τ_s	22.66	22.78	23.14	17. t_m, τ_m	23.17	23.63	24.37
4. t_s, τ_f	22.65	22.95	23.87	18. t_m, τ_f	22.98	23.81	24.43
5. γ_f, r_s	22.63	22.96	23.44	19. γ_s, r_s	23.99	24.95	25.83
6. γ_m, τ_m	22.70	22.99	23.47	20. t_l, τ_s	24.07	25.16	26.03
7. γ_f, τ_s	22.63	23.00	23.44	21. γ_s, τ_s	24.35	25.23	26.18
8. γ_f, τ_m	22.70	23.03	24.06	22. t_l, τ_m	24.12	25.34	26.33
9. γ_m, τ_s	22.74	23.07	23.63	23. γ_s, τ_m	24.07	25.66	27.05
10. γ_f, r_m	22.63	23.13	24.06	24. γ_s, τ_f	25.36	26.43	27.28
11. γ_f, τ_f	22.63	23.19	23.78	25. t_l, τ_f	25.06	26.67	28.41
12. γ_m, r_m	22.68	23.25	24.19	26. γ_s, r_m	25.64	27.18	30.46
13. γ_f, r_l	22.63	23.35	24.06	27. γ_s, r_l	25.07	27.44	29.12
14. t_m, τ_s	22.79	23.36	23.92				

Table 1: Minimum, median, and maximum least costs found for 27 different control schedules applied ten times to the Medianus-I graph.

A varying control parameter is indicated by a γ for the temperature and a τ for the radius and the rate is indicated by indices s, m, f (i.e., slow, medium, fast). A fixed control parameter is indicated by a t for the temperature or an r for the radius and the level is indicated by indices s, m, l (i.e., small, medium, large).

	Medianus-I			Medianus-II			Sparse			Dense		
	Min.	Med.	Max.	Min.	Med.	Max.	Min.	Med.	Max.	Min.	Med.	Max.
1. t_s, τ_m	22.63	22.89	23.82	20.46	20.82	22.45	22.67	22.92	24.10	50.89	51.94	53.98
2. γ_m, r_s	22.67	22.79	23.47	20.46	20.57	20.88	22.70	22.96	23.39	50.90	51.41	53.81
3. t_s, τ_s	22.63	22.80	23.46	20.47	20.67	22.44	22.66	22.84	23.33	50.88	51.80	54.94
4. t_s, τ_f	22.63	23.12	24.11	20.49	20.95	22.88	22.69	23.34	25.21	50.90	53.62	56.78
5. γ_f, r_s	22.63	22.96	24.02	20.45	20.64	22.42	22.61	22.68	23.29	50.88	51.42	53.18
6. γ_m, τ_m	22.69	23.22	24.18	20.50	21.30	22.96	22.81	23.89	25.70	51.17	54.25	56.87
7. γ_f, τ_s	22.63	22.97	24.08	20.45	20.61	21.50	22.61	22.77	24.12	50.88	51.84	54.44
8. γ_f, τ_m	22.63	22.99	24.07	20.45	20.83	22.86	22.61	22.99	25.37	50.88	52.34	54.86
9. γ_m, τ_s	22.67	23.04	23.60	20.48	20.87	21.97	22.67	23.38	25.00	50.96	52.80	54.55

Table 2: Minimum, median, and maximum least costs found for nine different control schedules applied 100 times to the four test graphs.

Considering the results provided by the top third control schedules in Table 1 we observe that for all of these the final temperature is low (i.e., $t_f \leq t_s$) and that the initial radius is small (i.e., $r_0 = r_s$). Three additional test graphs are employed: Medianus-II, a later version of Medianus-I containing 56 vertices and 110 edges; Sparse, a computer generated graph of 50 vertices and 100 edges; and Dense, a computer generated graph of 50 vertices and 359 edges. The results obtained by applying the top third control schedules in Table 1 100 times to all four graphs are displayed in Table 2. In Table 1 the most striking observation is that the three schedules with a constant low temperature belong to the top five schedules. Therefore an interesting experiment would be to test a strictly descent version of the annealing algorithm (i.e., $t = 0$). However, before doing that we will add a few more comments on Tables 1 and 2. The last third of the schedules is characterized by a high final temperature. For the middle third the final temperature is either low or medium. Note that the average state-space size is more than doubled from the 18th best to the 19th best schedule (i.e., the boundary between the middle and last third).

By calculating the average relative deviation of the median values from the least median for each control schedule in Table 2 we find that the ranking of the schedules is 5, 2, 3, 7, 1, 8, 9, 4, 6 (i.e., schedule number 5 is the best), where the differences between the results of schedules 2 and 5 are insignificant (see Figure 1). Schedules 4, 6 and 9

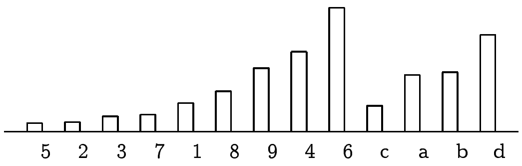

Figure 1: Average relative deviation from least medians for control schedules 1–9 and the four descent schedules a–d for the four test graphs.

are significantly worse than the rest: for all four graphs these three schedules are the three worst. The reason why schedule 4 performs badly appears to be the fast radius increase: For schedules 1, 3 and 4 the temperature is constantly low and the radius is increased at medium, slow and fast rates, respectively. Thus, fast radius increase is not recommendable and not even medium-rate radius increase seems to be recommendable as all of the control schedules including medium-rate radius increase (i.e., schedules 1, 6 and 8) perform relatively badly, though with schedule 1 as a possible exception. It is interesting to compare schedules 2, 6 and 9 as they are based on medium-rate temperature decrease (relatively high final temperature), but only schedule 2 produces acceptable results. From that comparison we must conclude that if the final temperature is (relatively) high only a constantly small radius is recommendable. Similarly, by comparing schedules 5, 7 and 8 we must conclude that with a high initial temperature a constant small radius is preferable.

So far we have not seen much evidence that the introduction of the new control parameter has resulted in improved annealing schedules. However, the differences between schedules 3 and 7 and between schedules 2 and 5 are very small, but due to the constantly small radius in schedules 2 and 5 the algorithm runs significantly slower with these control schedules than it does with schedules 3 and 7: with schedules 3 and 7 it runs approximately 2.6 times as fast as it does with schedules 2 and 5. Thus, seen in the light of bounded time resources, schedules 3 and 7 would be preferable to schedules

2 and 5. Even schedule 1 might be of interest as the annealing algorithm then runs an additional 30% faster without seriously compromising the median least cost.

To test the utility of a descent algorithm we perform four additional tests: constant small radius, slow increasing radius, medium increasing radius, and fast increasing radius. Table 3 shows the resultant minimum, median and maximum least costs obtained by these four descent schedules applied 100 times to each of the four graphs. Not surprisingly,

	Medianus-I			Medianus-II			Sparse			Dense		
	Min.	Med.	Max.	Min.	Med.	Max.	Min.	Med.	Max.	Min.	Med.	Max.
a. $t = 0, \tau_s$	22.63	23.08	24.25	20.45	20.85	22.41	22.61	22.85	24.66	50.88	53.27	56.86
b. $t = 0, \tau_m$	22.63	23.09	24.25	20.45	20.98	22.41	22.61	23.03	24.93	50.88	52.93	55.37
c. $t = 0, r_s$	22.63	23.06	24.25	20.45	20.74	22.41	22.61	22.74	24.51	50.88	51.99	54.55
d. $t = 0, \tau_f$	22.63	23.16	24.25	20.45	21.10	22.97	22.61	23.57	26.73	50.88	53.78	57.29

Table 3: Minimum, median, and maximum least costs found for four different descent control schedules applied 100 times to the four test graphs.

constant small radius (schedule c) is found to be superior, although schedules a and b are not significantly worse. By comparing the median values in Table 3 by the least medians in Table 2 we calculate the average relative deviation as above (see Figure 1). It is interesting to compare schedules a, b, c and d with schedules 3, 1, 5 and 4, respectively, as they have pairwise identical radius schedules. Note that the temperature for schedules 1, 3 and 4 are constantly low and that for schedule 5 the temperature becomes low quite rapidly. So the comparison provides a means of evaluating the effect of hill climbing. The comparison quite clearly demonstrates the superiority of the annealing algorithm over the descent algorithm. Thus the phenomenon of highland basins appears to have a significant effect on the results of the descent algorithm.

4.2 Accuracy of solutions

Having run the annealing algorithm several hundred times with different control schedules on each of the four test graphs, we can be pretty sure that we have found the global minimum cost or at least one very close to the global minimum. Each of the 13 control schedules was run 100 times giving a total of $1300 \times 3 \times 10^4 = 3.9 \times 10^7$ iterations for each of the four graphs. A histogram of the resulting 1300 least costs for the Medianus-I graph is shown in Figure 2, where the height of each bar corresponds to the number of runs producing minimal configurations with exactly the same cost. This shows clearly a unique lower value for these runs which was obtained in roughly one out of every eight runs. Histograms with this characteristic appearance are also obtained for the other three graphs. In order to further substantiate the assertion of the four lower values (i.e., one for each graph) being the global minimum costs, the annealing algorithm was run with a control schedule complying much better with the criteria for convergence than do the pragmatic ones of Section 3. For none of the graphs did the algorithm with this 'almost

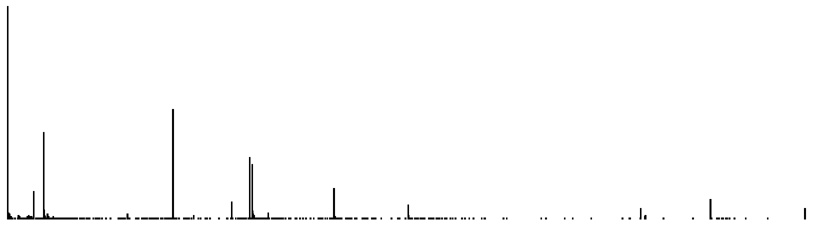

Figure 2: Distribution of the least costs for all runs on the Medianus-I graph.

ideal' control schedule (10^7 iterations) give better results.

4.3 Comparison with other methods

To get a fair and useful evaluation of the performance of the annealing algorithm we should compare it with other graph-triangulation algorithms. In the following discussion it is implicitly assumed that the term *annealing* means simulated annealing applied with control schedule 3 (t_s, τ_s).

An extensive evaluation of various heuristic graph-triangulation algorithms has been carried out by Kjærulff (1990). This evaluation was based on the same four graphs as used in Section 4.1. Some of the heuristic algorithms are well-known: maximum cardinality search (Tarjan & Yannakakis 1984), lexicographic search (Rose et al. 1976), minimum degree and minimum deficiency heuristics (Rose 1973). Kjærulff (1990) found a heuristic called the *minimum weight heuristic* to be superior.

By comparing the figures from Kjærulff (1990) with those of Table 2 it is found that the median cost produced by annealing for the Medianus-I graph is slightly better than the median cost produced by the minimum weight heuristic, whereas a similar comparison for Medianus-II shows that the heuristic algorithm is slightly better than annealing. In both cases the difference is insignificant. For the Sparse and Dense graphs the comparison clearly turns out in favour of annealing (the average state-space sizes obtained by the minimum weight heuristic are 5.5 and 3 times larger than those obtained by annealing, respectively).

It should be emphasized that the comparisons are based on average values from a large number of tests. This introduces another important parameter, namely the dispersion of the results. In Kjærulff (1990) an extremely small dispersion was found for the results of the minimum weight heuristic. A detailed study of the test runs for the annealing algorithm showed that more than 80% of the runs resulted in costs equal to or less than the median costs for the Medianus graphs. A similar study for the Sparse and Dense graphs gave 63% and 68%, respectively. Comparisons of worst-case behaviours of the annealing algorithm and the minimum weight heuristic showed the following. For Medianus-I, Medianus-II and Dense, annealing produced worst-case state-space sizes which are 2.0, 4.8 and 1.8 times as large as the corresponding worst-case results of the minimum weight heuristic, respectively. For the Sparse graph, however, the heuristic algorithm produced results which are 4.3 times as large as the worst-case results of the annealing algorithm. Similar comparisons of best-case behaviours showed that annealing produces state-space sizes which are 81%, 88%, 19% and 16% of those of the heuristic algorithm for Medianus-I, Medianus-II, Sparse and Dense, respectively.

5 Conclusion

This paper has investigated practical issues of applying simulated annealing to achieve approximate optimum decomposition of probabilistic networks and suggested control schedules, including the radius of the search space as an additional control parameter. Although the annealing algorithm has been successfully applied to provide high-quality approximate solutions to many NP-complete problems (Bohachevsky, Johnson & Stein 1986, Geman & Geman 1984, Lundy 1985, Thomas 1986) convergence of the algorithm in less than exponential time cannot be guaranteed for general problems (Lundy & Mees 1986). The purpose of this paper, however, has been to investigate the applicability of the algorithm under bounded resources. The specific aim was to develop an optimal control schedule such that the running time of the algorithm becomes independent of the size and topography of the search space. Of course, an important consequence of this is more coarse-grained solutions for large networks, i.e., larger absolute distances between adopted and optimum solutions. However, by assuming independence between the relative rate of convergence and the problem size this strategy implies identical relative distances. Further, keeping the relative distance at a constant level is quite acceptable for the purpose of the present problem.

With the pragmatic control schedules developed in Section 3 it appeared that the annealing algorithm performs best (under bounded resources) with a constant low temperature and a slowly increasing search-space radius. Strictly speaking, it is not correct to call it simulated annealing then, but to avoid confusion we will stick to that term anyway. Using the radius of the search space as an additional control parameter has a significant effect on the time consumption of the annealing algorithm and does not significantly compromise the quality of the solutions if the radius is increased only slowly.

The lesson to be learnt from the comparisons in Section 4.3 is that if time is not a crucial parameter the use of annealing might be recommended provided it is run more than once. The clearest example in which annealing is superior appears with the Sparse graph, where even the worst-case state-space size obtained by annealing is only 30% of the best-case size obtained by the minimum weight heuristic. On the other hand, if triangulation has to be fast we can by no means recommend the annealing algorithm which is orders of magnitude slower than the heuristic algorithms.

Thomas (1986) applied the annealing algorithm to establish optimum peeling sequences in pedigree analysis. Interestingly, his finding is that annealing performs significantly better than a greedy algorithm (which appeared to be identical to the minimum weight heuristic) even though the two applications are fairly similar. The main differences lie in a different cost-function definition (maximum clique weight instead of sum of clique weights) and the special structures of the pedigree networks.

Since annealing performs best with a constant low temperature (under bounded resources) there is no reason for starting with an arbitrary vertex ordering, which would waste a lot of time testing useless orderings. Instead, it is highly recommended to apply one of the heuristic ordering algorithms first and then let the annealing process start with the ordering produced by the heuristic. This might, first, stabilize the performance of the algorithm (i.e., reduce dispersion), and second, reduce the running time.

Kjærulff (1990) also evaluated a branch-and-bound algorithm, which in a way is the deterministic equivalent of the annealing algorithm. Comparisons with Medianus-I as the target graph showed that branch-and-bound could not at all compete with annealing. A test was conducted by running the annealing and the branch-and-bound algorithms for fixed amounts of time on a heuristically ordered start configuration. In nine out of the ten cases annealing improved the initial ordering down to a level very close to the global minimum cost. Branch-and-bound, on the other hand, made modest improvements in just two of the ten cases.

Acknowledgements

I wish to thank Finn V. Jensen and Steffen L. Lauritzen for their valuable comments and suggestions during the process of producing this paper. I am also grateful to Frank Jensen who gave me his source code of the branch-and-bound algorithm. In my implementation of the simulated annealing algorithm I benefited from reusing some of the ideas from this code.

The research was conducted mainly while I was employed by Judex Datasystemer A/S, Aalborg, Denmark. It was financially supported by Judex Datasystemer A/S and The Danish Academy of Technical Sciences.

References

Andreassen, S., Jensen, F. V., Andersen, S. K., Falck, B., Kjærulff, U., Woldbye, M., Sørensen, A. R., Rosenfalck, A. & Jensen, F. (1989). MUNIN — an expert EMG assistant, *in* J. E. Desmedt (ed.), *Computer-Aided Electromyography and Expert Systems*, Elsevier Science Publishers B. V. (North-Holland), chapter 21.

Bohachevsky, I. O., Johnson, M. E. & Stein, M. L. (1986). Generalized simulated annealing for function optimization, *Technometrics* **28**(3): 209–217.

Cooper, G. F. (1987). Probabilistic inference using belief networks is NP-hard, *Research Report KSL-87-37*, Knowledge Systems Laboratory, Medical Computer Science, Stanford University, California.

Fujisawa, T. & Orino, H. (1974). An efficient algorithm of finding a minimal triangulation of a graph, *IEEE International Symposium on Circuits and Systems*, pp. 172–175.

Geman, S. & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(6): 721–741.

Jensen, F. V., Andersen, S. K., Kjærulff, U. & Andreassen, S. (1987). MUNIN – on the case for probabilities in medical expert systems – a practical exercise, *in* J. Fox, M. Fieschi & R. Engelbrecht (eds), *Proceedings of First Conference of the European Society for Artificial Intelligence in Medicine (AIME)*, Springer-Verlag, Heidelberg, pp. 149–160.

Jensen, F. V., Lauritzen, S. L. & Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computations, *Computational Statistics Quarterly* **4**: 269–282.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing, *Science* **220**: 671–680.

Kjærulff, U. (1990). Triangulation of graphs — algorithms giving small total state space, *Research Report R-90-09*, Department of Mathematics and Computer Science, Aalborg University, Denmark.

Lauritzen, S. L. & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems, *Journal of the Royal Statistical Society, Series B* **50**(2): 157–224.

Lundy, M. (1985). Applications of the annealing algorithm to combinatorial problems in statistics, *Biometrika* **72**: 191–198.

Lundy, M. & Mees, A. (1986). Convergence of an annealing algorithm, *Mathematical Programming* **34**: 111–124.

Olesen, K. G., Kjærulff, U., Jensen, F., Jensen, F. V., Falck, B., Andreassen, S. & Andersen, S. K. (1989). A MUNIN network for the median nerve — a case study on loops, *Applied Artificial Intelligence*. Special issue: Towards Causal AI Models in Practice.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Series in Representation and Reasoning, Morgan Kaufmann Publishers, Inc.

Rose, D. J. (1970). Triangulated graphs and the elimination process, *Journal of Mathematical Analysis and Applications* **32**: 597–609.

Rose, D. J. (1973). A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, *in* R. C. Read (ed.), *Graph Theory and Computing*, Academic Press, New York, pp. 183–217.

Rose, D. J., Tarjan, R. E. & Lueker, G. S. (1976). Algorithmic aspects of vertex elimination on graphs, *SIAM Journal on Computing* **5**: 266–283.

Shachter, R. D. (1988). Probabilistic inference and influence diagrams, *Operations Research* **36**: 589–604.

Shafer, G. & Shenoy, P. P. (1988). Bayesian and belief-function propagation, *Working Paper 121*, School of Business, University of Kansas, Lawrence, Kansas.

Spiegelhalter, D. J. (1986). Probabilistic reasoning in predictive expert systems, *in* J. F. Lemmer & L. M. Kanal (eds), *Uncertainty in Artificial Intelligence*, North-Holland.

Suermondt, H. J. & Cooper, G. F. (1988). Updating probabilities in multiply-connected belief networks, *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence*, Association for Uncertainty in Artificial Intelligence, pp. 335–343.

Tarjan, R. E. & Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM Journal on Computing* **13**(3): 566–579.

Thomas, A. (1986). Optimal computation of probability functions for pedigree analysis, *IMA Journal of Mathematics Applied in Medicine & Biology* **3**: 167–178.

Wen, W. X. (1990). Optimal decomposition of belief networks, *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence*, Association for Uncertainty in Artificial Intelligence, pp. 245–256.