Am. J. Hum. Genet. 65:885–901, 1999

Blocking Gibbs Sampling for Linkage Analysis in Large Pedigrees with Many Loops

Claus Skaanning Jensen[1] and Augustine Kong[2]

[1]Department of Mathematics and Computer Science, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark; and [2]Department of Statistics, University of Chicago, Chicago

Summary

We apply the method of "blocking Gibbs" sampling to a problem of great importance and complexity—linkage analysis. Blocking Gibbs sampling combines exact local computations with Gibbs sampling, in a way that complements the strengths of both. The method is able to handle problems with very high complexity, such as linkage analysis in large pedigrees with many loops, a task that no other known method is able to handle. New developments of the method are outlined, and it is applied to a highly complex linkage problem in a human pedigree.

Introduction

For linkage analysis—the problem of the estimation of the relative positions of the genes on the chromosomes—many methods have been developed during recent years. Fast and exact methods for computation in Bayesian networks (e.g., pedigrees) (Cannings et al. 1976; Pearl 1986; Lauritzen and Spiegelhalter 1988; Shenoy and Shafer 1990; Lauritzen 1992) handle only small problems, since the computation is "NP-hard" (the complexity is exponential; i.e., there are no practical algorithms that will always solve the problem in polynomial time). Markov chain Monte Carlo (MCMC) methods (Gelfand and Smith 1990; Gelman and Rubin 1992; Geyer 1992; Thomas et al. 1992; Smith and Roberts 1993) have provided a good alternative, since they are able to handle very difficult problems. In these methods, computation time often exceeds any acceptable level, when one is considering very large networks (e.g., pedigrees of thousands of individuals), and it is often difficult to decide whether the desired precision has been reached. What may be even worse is that, in the general case, it is impossible to guarantee irreducibility, and thus, in many cases, it can never be known whether convergence has been reached.

Linkage analysis represents a problem of high complexity that has been particularly hard to handle. Existing methods such as those implemented in the LINKAGE (Lathrop and Lalouel 1984; Lathrop et al. 1985) and FASTLINK software packages (Cottingham et al. 1993; Schäffer et al. 1994) are unable to handle pedigrees with even a moderately low number of loops (i.e., ≈10), since the computation time increases exponentially with the number of loops. Sequential imputation (Kong et al. 1993; Irwin et al. 1994), which essentially is also a blocking scheme, handles multiple loci very well, but only when there are no or very few loops. Simulated tempering/annealing MCMC (Geyer and Thompson 1993) is a promising approach that handles these cases, although it seems to require a difficult choice of initial parameters and may suffer from problems of low acceptance rates.

Blocking Gibbs sampling allows general inference in very large complex Bayesian networks and is a particularly promising method for linkage analysis as well as for many other problems requiring inference in large Bayesian networks. The method combines exact local computations and Gibbs sampling (Geman and Geman 1984), such that, instead of sampling a single variable at a time ("single-site Gibbs sampling"), a very large proportion of the variables (usually >90%) are sampled jointly by exact local computations. Joint sampling of many variables (i.e., block updating) hinges on the fact that conditioning on certain variables breaks loops in the network, creating a network with fewer loops. This network, if enough loops are broken, then becomes feasible for exact computation, allowing us to sample the variables of the network jointly. Since blocking Gibbs sampling operates on general Bayesian networks in which the general unit of information is a variable, this notation will be used throughout the present article. A

Received January 14, 1997; accepted for publication June 2, 1999; electronically published July 29, 1999.

Address for correspondence and reprints: Dr. Claus Skaanning Jensen, Aalborg University, Institute for Electronic Systems, Department of Mathematics and Computer Science, Fredrik Bajers Vej 7E–DK 9220 Aalborg, 0–Denmark. E-mail: claus@cs.auc.dk

0002-9297/1999/6503-0033$02.00

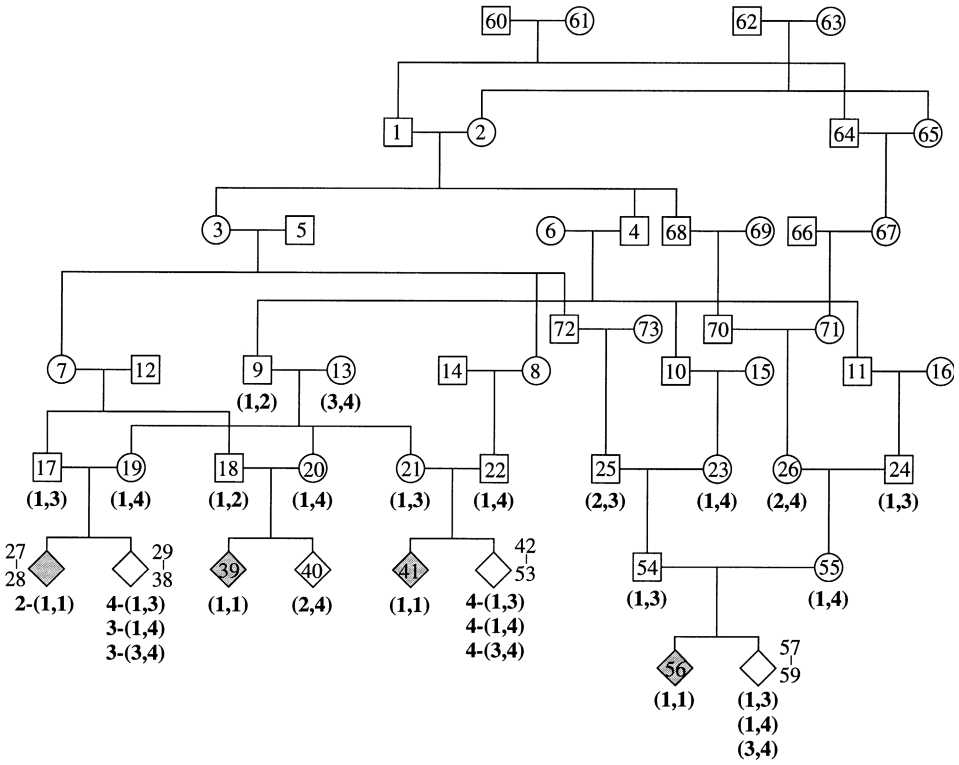


Figure 1 LQT pedigree. Marker genotypes are in parentheses. Gray-shaded diamonds denote affected sibs. Some diamonds represent several individuals; for example, diamond 29–38 represents the 10 individuals 29–38, 4 of whom have genotype (1,3), 3 of whom have genotype (1,4), and 3 of whom have genotype (3,4).

variable thus can represent any chosen unit of information, such as a genotype, an allele, a phenotype, etc.

Jensen et al. (1995) compared blocking Gibbs sampling with single-site Gibbs sampling. They applied both methods to three pedigrees, consisting of 455, 704, and 1,894 individuals. It was shown that, in all cases, blocking Gibbs sampling performs better than does single-site Gibbs sampling. In general, blocking Gibbs sampling mixes fast, whereas single-site Gibbs sampling often mixes very slowly or even gets completely stuck. The example pedigrees used by Jensen et al. (1995) were small (blocking Gibbs sampling can handle much larger pedigrees) but highly inbred, with hundreds if not thousands of loops. No other known method can handle pedigrees this large—except for simulated tempering, if suitable starting parameters can be found.

In the present article, we apply blocking Gibbs sampling to a 73-individual linkage problem (see fig. 1) (Kong 1991) concerning a rare heart disease called the "long QT syndrome" (LQT). We perform a two-point linkage analysis of this pedigree and analyze the behavior and performance of blocking Gibbs sampling. This example contains many loops; however, it is still possible to use exact computation on it, which provides us with the correct result. Although exact computation is possible, this example is just within the limits of what is currently possible to do with exact methods. Still, the blocking Gibbs method is able to handle much larger examples.

In the next section we describe the blocking Gibbs method. Then the method is applied to the LQT pedigree, and a discussion follows. A more detailed description of the blocking Gibbs method can be found in the work of Jensen (1997).

Blocking Gibbs Sampling

Although blocking Gibbs sampling is an MCMC method, it employs an exact inference method, which we describe in the following.

Exact Belief Updating

It has been shown (Shachter et al. 1991) that all exact methods for belief updating in Bayesian networks can be viewed as variations on a single, general algorithm involving clustering of variables in a structure called a "junction tree" (Jensen et al. 1990). (For a definition of Bayesian networks, see the work of Pearl [1991].) In this category also falls "peeling" (Elston and Stewart 1971;

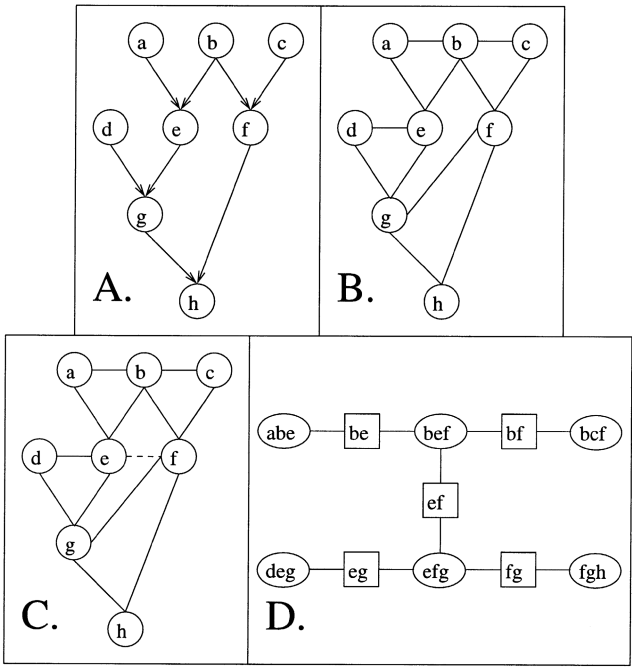


Figure 2 Transformation of a Bayesian network into a junction tree. In panel *D*, ovals represent cliques, and boxes represent separators.

Lange and Elston 1975; Cannings et al. 1978), which is an exact method developed for belief updating in a particular type of Bayesian networks—namely, pedigrees. Peeling can not be used for the blocking Gibbs algorithm, for two reasons: (i) it returns only the marginal probabilities of a single variable (or cluster of variables) and has to be run once for each variable (or cluster), thereby causing extreme inefficiency for this application, and (ii) it is not able to perform exact sampling of all variables, something that is crucial in a Gibbs sampler with block updating.

Thus, even though the use of peeling in the algorithm would make it more easily accessible to researchers in genetics, we must use the more general updating method provided by the junction-tree method, for the reasons noted above. The junction tree is obtained from the Bayesian network in the following way (see the example in fig. 2; as mentioned above, the variables given in figure 2 do not necessarily represent pedigree individuals but, rather, are data pertaining to an individual—for example, the paternal gene, the phenotype, etc.).

1. In figure 2*A*, we have the Bayesian network, a directed acyclic graph with a conditional distribution associated with each variable. Obviously, a pedigree can easily be represented as a Bayesian network.

2. The "moral graph" (fig. 2*B*) is obtained by addition of undirected edges between all pairs of disconnected variables with common children and by replacement of directed edges by undirected ones.

3. The "triangulated graph" (fig. 2*C*) is obtained by addition of edges to the moral graph, such that there are no cycles of length >3 that do not have a chord. These edges are denoted "fill-in links."

4. Finally, the "junction tree" (fig. 2*D*) is constructed by connecting the "cliques" C (maximal sets of pairwise connected variables) of the triangulated graph. The cliques are separated by the so-called separators, S. The junction tree has the property that the intersection $C_1 \cap C_2$ of any two cliques is a subset of all cliques on the path between C_1 and C_2 in the tree. (This is called "the junction-tree property.") The cliques of the junction tree essentially correspond to the cut sets of peeling.

The storage requirements of a junction tree JT can be expressed as the sum of the storage requirements of the cliques C and the separators S: $c(JT) = \Sigma_{A \in C \cup S}\, c(A)$, where c denotes complexity. If $\mathrm{Sp}(v)$ denotes the state space of variable v, then the storage requirements of $A \in C \cup S$ are given as $c(A) = \Pi_{v \in A} |\mathrm{Sp}(v)|$. In the current implementation of blocking Gibbs sampling, we have used an object-oriented version of the exact method of Lauritzen and Spiegelhalter (1988), which has been described by Jensen et al. (1990) and implemented in the expert-system shell HUGIN (Andersen et al. 1989). In this scheme, belief updating is implemented through message passing in a junction tree.

The message passing scheme is well described in the work of Jensen (1996*b*). Before message passing can be performed, the cliques and separators of the junction tree must be initialized as follows:

1. Initially all cliques and separators are given a probability table of all 1's.

2. Each variable v in the junction tree JT is assigned to a clique C that contains v and its parents, $\mathrm{pa}(v)$. Because of the moralization of the Bayesian network, such a clique always exists. The probability table for v, $P(v \mid \mathrm{pa}[v])$, is multiplied by C's probability table.

We can now insert observations—for example, that v is in state 1—into the junction tree, by finding a clique C containing v and then multiplying table f_v by C's probability table. f_v is a table for $\mathrm{Sp}(v)$ and has a 1 in the observed state and 0's in the other states.

The effect of observations on variables in the junction tree can now be propagated to other variables by message passing. Message passing is based on the "absorption" operator, which propagates information from a clique C_1 to one of its neighbors C_2 with separator S. It consists of the following steps, in which t_{C_1}, t_{C_2}, and t_S are the original probability tables:

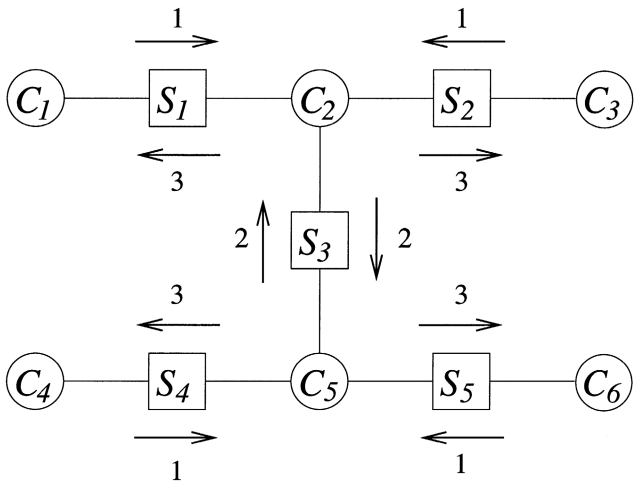


Figure 3 Example of message passing for a small junction tree

1. We calculate $t_S^* = \Sigma_{C_1 \setminus S} t_{C_1}$—that is, the information in the table for C_1 that concerns S;

2. then we assign to S the new table t_S^*;

3. and, finally, we assign to C_2 the new probability table $t_{C_2}^* = t_{C_2}(t_S^*/t_S)$.

We then say that C_2 has "absorbed" from C_1.

The message-passing scheme performs a series of absorptions in the junction tree, which passes information to all parts of the tree. Basically, a message is sent from C_1 to C_2 when C_2 absorbs from C_1. In the message-passing scheme, a clique C_1 can send exactly one message to each neighbor C_2, and it may be sent only when C_1 has received a message from all other neighbors.

An example of message passing is shown in figure 3. First, the leaf cliques with only one neighbor—that is, C_1, C_3, C_4, and C_6—can send a message to their neighbor. Then, C_2 can send a message to C_5, and C_5 can send a message to C_2. Then, finally C_2 can send messages to C_1 and C_3, and C_5 can send messages to C_4 and C_6.

It has been shown (e.g., see Jensen 1996*b*) that, when message passing has concluded, the junction tree is consistent, which means that all cliques containing v hold the same information about v. Furthermore, the marginal probabilities of each variable v, given the observations, can now be obtained by marginalization of an arbitrary clique containing v.

Thus, the benefit of the junction-tree method is that it organizes the variables in optimal clusters such that only the minimal number of variables have to be considered jointly. This is done by exploitation of the independencies between variables.

The junction-tree structure can be used as the basis for other kinds of computations—for example, a random joint sample of the unobserved variables, given observed variables (Dawid 1992). This is what is used for the sampling of large sets of variables jointly in blocking Gibbs sampling, and, as can be seen from the discussion above, it *requires* that exact computations can be performed on that pedigree part that contains these variables.

The random joint sample is basically obtained by modification of the absorption step in the message-passing scheme. Now, a random configuration has been selected for C_1 and has been entered as an observation such that the probability table of C_1 contains a single 1 at this configuration and contains 0's at all other places. Since S is contained within C_1, a random configuration for S is implicit in the table of C_1. The table for S containing a 1 at this configuration is multiplied by the table for C_2, and the resulting table for C_2 is used to draw a random joint sample for the remaining variables in C_2 that have not yet been fixed. When this modified absorption operator is used in the message-passing scheme, a joint sample is obtained for all variables in the junction tree. In the case of blocking Gibbs sampling, many variables in the junction tree will have fixed values, and the remaining variables will be sampled, given these—something that is possible in the method of Dawid (1992).

For further (and much more detailed) information on junction-tree algorithms, consult the work of Jensen (1996*b*).

Gibbs Sampling

As has been mentioned, exact methods such as those discussed above cannot be used for large problems with many loops. In these situations, MCMC methods can often be used.

A popular MCMC method is Gibbs sampling. The general Gibbs sampling method can be explained as follows: Given a Bayesian network with variables V, we can construct sets ("blocks") $B_1,\ldots,B_n$, such that $B_i \subseteq V$, $i = 1,\ldots,n$, $B_1 \cup \cdots \cup B_n = V$, and $B_1,\ldots,B_n$ are not necessarily disjoint. We then define $A_i = V \setminus B_i$, $i = 1,\ldots,n$ (i.e., A_i is the set obtained from V after exclusion of B_i). The Gibbs sampler then proceeds as follows:

0: Find the starting configuration of A_1, which is called "$A_1^{(1)}$."

1: Simulate the variables in B_1 conditional on $A_1^{(1)}$. This yields $B_1^{(1)}$. Now, all variables are realized, and the configuration $A_2^{(1)}$ is found directly from this realization.

2: B_2 is simulated conditional on $A_2^{(1)}$, yielding $B_2^{(1)}$. $A_3^{(1)}$ is extracted from the configuration $(A_2^{(1)},B_2^{(1)})$.

⋮

$n-1$: B_{n-1} is simulated conditional on $A_{n-1}^{(1)}$, yielding $B_{n-1}^{(1)}$. $A_n^{(1)}$ is extracted from the configuration $(A_{n-1}^{(1)},B_{n-1}^{(1)})$.

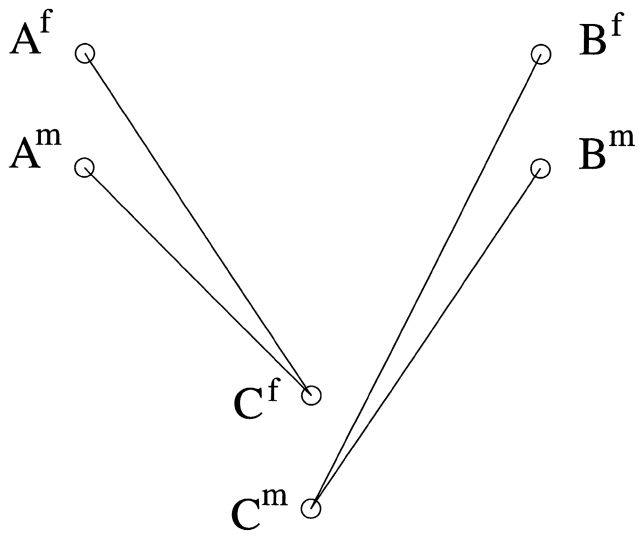


Figure 4 A and B and their offspring C. The notation is as follows: A has genes A^f and A^m, where A^f is the gene originating from the father of A and A^m is the one originating from the mother of A.

n: B_n is simulated conditional on $A_n^{(1)}$, yielding $B_n^{(1)}$. $A_1^{(2)}$ is extracted from the configuration $(A_n^{(1)},B_n^{(1)})$.

Steps 1,...,*n* are repeated until a sufficient number of samples—$(B_1^{(1)},\ldots,B_n^{(1)}),(B_1^{(2)},\ldots,B_n^{(2)}),\ldots,(B_1^{(m)},\ldots,B_n^{(m)})$—have been produced. These samples are then used to estimate the stationary distribution of the Markov chain induced by the Gibbs sampler.

In the scheme given above, it can be seen that a variable can be sampled multiple times in each iteration, since the blocks do not have to be disjoint.

The variant of Gibbs sampling that is usually seen—that is, "single-site Gibbs sampling—samples only one variable at a time, rather than a set of variables. The single-site Gibbs sampler is very easily implemented but may have severe problems. If the problem at hand is large and contains many loops, mixing may be too slow for practical purposes, or, because of multimodality (i.e., the division of the configuration space into multiple almost disconnected subspaces), the sampler can get stuck in a subspace. These are problems particularly apparent in linkage analysis in which networks often are huge with large numbers of loops. Blocking Gibbs sampling usually avoids these problems by sampling the majority of variables jointly.

The Blocking Gibbs Sampling Scheme

Blocking Gibbs sampling is an attempt at implementation of the general Gibbs sampling scheme in which many variables are sampled jointly. As mentioned above, the blocks do not have to be disjoint, but their union should contain all variables. We have various criteria that we want the blocks to fulfill:

1. The blocks should contain as many variables as possible. Their size is governed by memory capacity. The larger the blocks, the faster the blocking Gibbs sampling will converge. In the limit for which the entire network is included within one block, we get exact simulation. In the opposite limit, in which only one variable is included in each block, we will get single-site Gibbs sampling.
2. All variables should be sampled approximately equally often.
3. The blocking Gibbs sampler should be irreducible.

Criterion 1.—To construct a block that contains as many variables as possible, we use the fact that conditioning on certain variables breaks loops in the network, creating a network with fewer loops. This network, given that enough variables are being conditioned on (and, thus, that enough loops are broken), then becomes feasible for exact computation, allowing us to sample the variables of the block jointly. The variables that are removed from the block in order to reduce the memory requirements will be termed the "optimal variables."

We construct a block *B* by selecting a set of variables *A* that should not be part of the block. When sampling *B,* we condition on the variables in *A,* breaking a sufficient number of loops. The variables in *B* are now sampled jointly by the "random propagation" method described by Dawid (1992). When loops are broken, the storage requirements of the junction tree are reduced.

We want *B* to contain as many variables as possible.

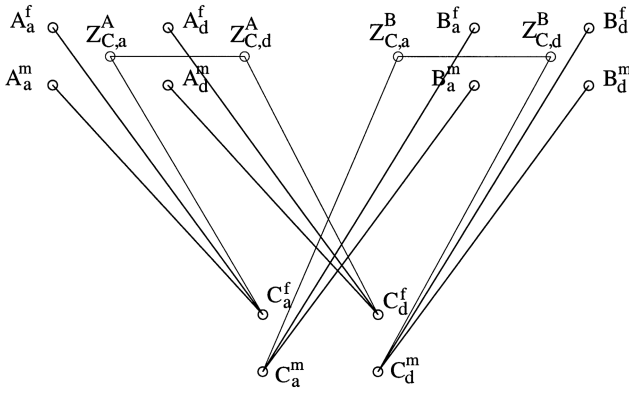


Figure 5 Representation of the two-locus linkage problem. As in figure 4, A^f represents the gene that individual A inherited from the father. The subscripts denote the locus of the gene; that is, "a" denotes the marker, and "d" denotes the disease. There are thus four gene variables for each individual. For each individual there are four indicator variables: $Z^A_{C,a}$, $Z^A_{C,d}$, $Z^B_{C,a}$, and $Z^B_{C,d}$. Only the indicator variables of C are shown.

This can be achieved by making A as small as possible, by selecting the variables for A that reduce the storage requirements of the junction tree the most. These variables are usually contained in many cliques in the junction tree, and they have many links in the Bayesian network.

In a pedigree setting, these variables would correspond to individuals who are involved in many loops. The variables in the blocking Gibbs sampler do not necessarily correspond with individuals, however. Depending on the type of pedigree analysis, each individual can correspond to one or more variables in the network. This will be further detailed in the Description of the Created Blocks section, below. In figure 4 we see an example in which an individual is represented by two variables, and in figure 5 we see an example in which individual C is represented by eight variables.

This situation also can be described by means of pseudo code. In the following program, S is the amount of storage (RAM, disk, etc.) available on the computer.

```
find optimal variables : {v1,v2,v3,v4,...}
JT1 ← initial junction tree
i ← 1
while storagereq(JTi) > S do :
    JTi+1 ← JTi with vi removed
    i ← i + 1
end while.
```

The optimal variable is found by computation, for each variable v, of the reduction of storage requirements that is caused by the removal of v from the junction tree.

The algorithm is iterative, such that, when an optimal variable has been found and removed from the junction tree, the reduced junction tree is used as basis for finding the next optimal variable. Thus, the pseudo code given above finds a conditionally ordered list of the variables.

Since the optimal variables are the variables that most reduce the sizes of the cliques in the junction tree, these variables will also, in general, be the variables that are located in the most cliques. Again, since cliques correspond to cut sets in the peeling algorithm, the optimal variables will correspond to variables that are located in many cut sets in the peeling.

The removal of a variable from the junction tree necessitates multiple changes in the junction-tree structure, to ensure that the junction-tree property remains valid. Here is a small piece of pseudo code illustrating the changes when the variable v is removed from a junction tree, JT:

```
delete v from all cliques and separators
    for each clique Ci in C do :
  for each neighboring clique Cj of Ci do :
      if Cj is a proper subset of Ci do :
    delete Cj, delete the separator connecting
                Cj and Ci, and reconnect all separators
                connected to Cj to Ci
          end if.
        end for.
    end for.
remove redundant fill-in links.
```

In this algorithm, the first 10 statements perform reductions in the junction tree that are made possible by the removal of v. When v is removed, it is possible that some cliques become subsets of others, and, since it is only necessary to have each clique represented once in

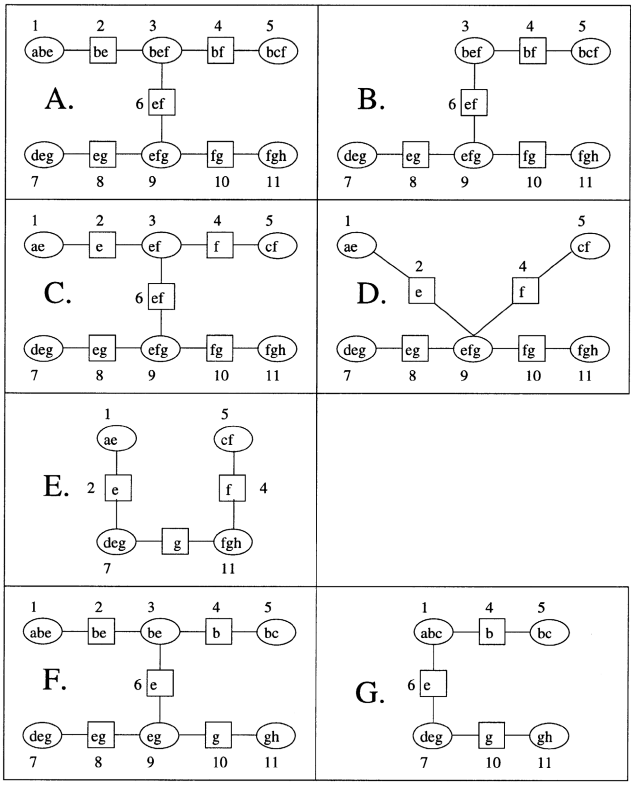


Figure 6 Examples of removal of a variable from a junction tree. Panel *A* shows the starting configuration. In panel *B*, "a" is removed; in panels *C–E*, "b" is removed; in panels *F* and *G*, "f" is removed.

the junction tree, these cliques that are subsets of others can be removed.

The last statement searches through the junction tree, for fill-in links that are rendered redundant by the prior reductions. If a variable is removed in a Bayesian network—for example, *b* in figure 2*A*—it is possible that fill-in links added during the triangulation become redundant. One fill-in link, *e–f,* is present in this situation, and, when *b* is removed, it becomes redundant. The reason for this is that it is no longer required for triangulation of the network, since there are no longer any cycles of length >3 that do not have a chord. A redundant fill-in link is characterized by being present in only one clique (Kjærulff 1993). Thus, the junction tree can quickly be searched for these links, and, once one has been removed, the next one can be found, until there are no more. The removal of the link in the Bayesian network is performed, in the corresponding junction tree, by splitting the single clique containing the variables of the link into two cliques, each of which contains only one of the two variables. These cliques may now be subcliques of other cliques and thus be subject to removal, etc.

The algorithm describes only the construction of a single block, constructed by removal of variable after variable until the junction tree representing the block can be stored within a reasonable space. To run the blocking Gibbs sampler, it would be necessary to run the algorithm several times, to construct a number of blocks. These blocks may be overlapping, but their union must contain all variables such that they are all sampled at least once in each iteration. How to do this is described the Criterion 2 subsection, below.

The algorithm can be illustrated with a few examples, as shown, for example, in figure 6 (again, it should be noted that the variables in figure 6 do not necessarily correspond to pedigree individuals).

Example A: In figure 6*A,* we see the initial junction tree also shown in figure 2.

Example B: In figure 6*B,* variable *a* has been removed. The removal of *a* causes clique 1 to become a proper subset of clique 3. Thus, clique 1 and separator 2 can be deleted.

Example C: In figure 6*C,* variable *b* has been removed. We see that clique 3 becomes a proper subset of clique 9. Thus, clique 3 and separator 6 can be removed. Separators 2 and 4 that were connected to clique 3 have now been connected to clique 9. This is shown in figure 6*D*.

Example E: The junction tree in figure 6*D* can be further reduced when we take into account that the fill-in link *e–f* is now redundant, since the variables *e* and *f* are present in only one clique, {*e,f,g*}. Thus, according to the last line of the block-construction algorithm, this clique can be split in two such that each of the variables occurs in one of the two new cliques. In this case, the two new cliques, {*e,g*} and {*f,g*} are both subsets of other cliques and can be removed. The resulting reduced junction tree is shown in figure 6*E*.

Example F: In figure 6*F,* variable *f* has been removed. This causes both clique 3 to become a proper subset of clique 1 and clique 9 to become a proper subset of clique 7. In figure 6*G,* first, clique 3 and separator 2 have been deleted, and separators 4 and 6 are connected to clique 1. Then, clique 9 and separator 8 are deleted, and separators 6 and 10 are connected to clique 7.

Thus, in figure 6, three blocks are constructed. These blocks are not optimal but are merely chosen to illustrate how blocks can be created from the initial junction tree. If three optimal variables were to be chosen in this junction tree, they would be chosen among *b, e, f,* and *g*.

With the blocks found in figure 6, a blocking scheme could be performed. One iteration in this scheme would consist of the following steps:

1. Update variables in example B, given the current value of *a*. Thus, a new value for *b* is found in this step.
2. Update variables in example E, given the current value of *b*. A new value for *f* is found in this step.
3. Update variables in example G, given the current value of *f*. A new value for *a* is found in this step.

In each step, all variables except one are sampled jointly, and their new values are registered for later estimation of marginal probabilities.

We have been using the method for selection of blocks in many different networks, and empirical evidence suggests that, in almost all cases, the blocks contain >90% of the variables, allowing very fast mixing.

Criterion 2.—We want variables to be sampled approximately equally often, to ensure that we do not get situations in which some variables are sampled many times in each iteration whereas others are sampled only once. However, criterion 1 urges us to remove the optimal (with regard to reduction of storage requirements) variables from most of the blocks, and, as a result, these variables are then sampled only one or a few times in each iteration, whereas the majority of variables are included in all blocks and thus are sampled much more often. It seems that we *must* use the optimal variables in order to reduce storage requirements, letting criterion 1 take priority over criterion 2.

To ensure that the blocks have approximately the same size, we use a simple algorithm, here outlined with a piece of pseudo code. We want to construct N blocks, called "B_1,"…,"B_N." A counter, c_i, is associated with each variable, v_i, and c_i indicates how many times v_i has been removed from any block. We have tentatively decided that all c_i must not exceed $\gamma = \lfloor \frac{N}{2} \rfloor$ (where $\lfloor \rfloor$ de-

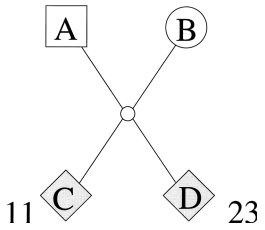


Figure 7 Simple pedigree in which A and B have offspring C and D. C has genotype (1,1), and D has genotype (2,3); thus, either (i) A has genotype (1,2) and B has genotype (1,3) or (ii) A has genotype (1,3) and B has genotype (1,2).

notes the integer part of the argument), thus forcing the optimal variables to be sampled at least $\lfloor \frac{N}{2} \rfloor$ times in each iteration. Again, S is the amount of storage available to the computer. In the following, the terms "junction tree" and "block" are used interchangeably; a block is created from the initial junction tree by removal of variables—that is, the block is also a junction tree.

```
for i = 1 to N do :
    B_i ← initial junction tree
end for.
while storagereq(B_1) > S ∨ ... ∨ storagereq(B_N) > S do :
    for i = 1 to N do :
        v_j ← variable in B_i with highest reduction
              of storage requirements for which c_j < γ
        c_j ← c_j + 1
        remove v_j from B_i
    end for.
end while.
```

Criterion 3.—Reducibility may often be a problem, especially when single-site Gibbs sampling is used. Only in simple cases, such as diallelic pedigree analysis, is irreducibility almost always guaranteed (Sheehan and Thomas 1993). However, for pedigree analysis with more than two alleles, irreducibility will often depend on the blocking of the variables. The simple pedigree-analysis example shown in figure 7 is reducible by the single-site Gibbs sampler, owing to the fact that it will be unable to switch between the configurations (A = 12,B = 13) and (A = 13,B = 12). Several similar examples and an extended investigation of the reducibility of Gibbs sampling in pedigree analysis is provided in he work of Jensen and Sheehan (1997). However, if blocking Gibbs sampling were applied and A and B were sampled jointly, the problem would be irreducible. The reducibility problem seen in figure 7 is very common in pedigrees; however, it is not seen in figure 1.

In the present implementation of blocking Gibbs sampling, the representation described by Kong (1991) has been used. In this representation, each variable represents the inherited allele of a genotype (see fig. 4.

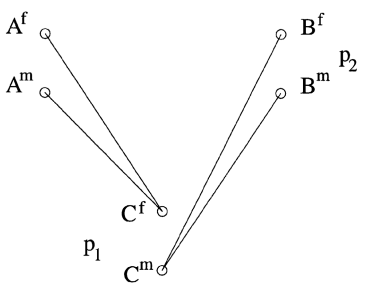
When this representation is used, reducibility can occur in a number of special cases. First, for example, if A has been typed to genotype (1,2), then A^f and A^m must be sampled jointly, to enable switching between configurations (A^f = 1,A^m = 2) and (A^f = 2,A^m = 1).

Second, if the probabilities $P(g|p)$ are defined such that (*a*) one phenotype p_1 can correspond to only one homozygous genotype (1,1) and (*b*) another phenotype p_2 can correspond only to a set of genotypes different from (1,1), then the pedigree shown in figure 8 will be reducible by single-site Gibbs sampling.

Again, the solution to the problem in figure 8 is to sample the variables B^f and B^m jointly, thereby allowing switching between the configurations (B^f = 1,B^m = 2) and (B^f = 2,B^m = 1). This problem appears in several individuals shown in figure 1; for example, individual 56, who has the disease phenotype (1,1), forces unaffected individual 54 to have one of the noncommunicating configurations—(54^f = 1,54^m = 2) or (54^f = 2,54^m = 1)—at the disease locus. (Even though we represent the possibility of both configurations of the genotype (1,2), the locus is *not* fully penetrant. For linkage analysis, it is necessary to know from which parent an allele has been inherited, and, even though this information has not been given to begin with, it can be obtained through sampling.)

In general, many other situations like that described above may occur, and they are not always easily detected. At this point, it is not clear whether a general method for finding these blocks will ever be found. The problem of finding the blocks corresponds to the very hard problem of finding the noncommunicating classes for MCMC methods, discussed by Lin et al. (1994).

Finding the starting configuration.—Finding the start-



Penetrance probabilities P(g|p)

	p_1	p_2
11	1	0
12	0	0.25
21	0	0.25
22	0	0.5

Figure 8 Offspring C with genotype (1,1), which requires that B with phenotype p_2 have either genotype (1,2) or genotype (2,1).

ing configuration is often a problem with MCMC methods. With single-site Gibbs sampling, one must find a starting configuration for the entire network (except one variable); however, with blocking Gibbs sampling, one must find a starting configuration for only the complement of the first block (A_1 in the Gibbs Sampling section, above). A_1 usually contains <10% of the variables, and these variables are usually located in different parts of the network, making the problem of finding a legal configuration easier. Even though one must find only a starting configuration for A_1, this starting configuration still must be legal with regard to the remainder of the variables, B_1. This is trivial, however, owing to the fact that the variables in A_1 are spread out over the network.

Jensen (1996*a*) has presented a method that swiftly finds the starting configuration in all cases tested so far. There is too little space to describe it here, but it is based on the exact sampling method of Dawid (1992) and attempts to sample parts that are as large as possible, until all variables have been fixed. It is sometimes necessary to backtrack (as with gene dropping/forward sampling and Gibbs sampling with relaxed probabilities), but this occurs only in rare cases, and even then it is necessary to backtrack only a few steps.

Linkage in the LQT Pedigree

In this section, we will apply the blocking Gibbs method to the LQT pedigree shown in figure 1. We will describe this particular linkage problem and explain how linkage-analysis problems are represented and handled by blocking Gibbs sampling. Finally, we will compare various methods for estimating the recombination fraction (θ) and will present results that show the fast convergence of blocking Gibbs sampling.

The LQT Pedigree

The LQT pedigree first identified by Brian Suarez is affected by the LQT syndrome. Blood samples have been collected from individuals 9, 13, and 17–59. Thus, marker data are available only for these individuals. The marker data shown in figure 1 have been simulated by Suarez to mimic close linkage ($\theta \sim 0$). The marker is assumed to have four alleles with equal population frequencies. The LQT syndrome is assumed to be determined by two alleles, one with population frequency .05 and the other with population frequency .95, with the disease allele being the rarest. The LQT pedigree has also been examined by Kong (1991).

Linkage Analysis with Blocking Gibbs Sampling

We represent the linkage problem in a way similar to the pedigree representations shown in figures 4 and 7. However, we now have two loci, as seen in figure 5. There are still four gene variables for each individual A–C, with an extra subscript—*a* (marker) or *d* (disease)—denoting the locus. In addition, for each individual, there are now four indicator variables: $Z^{\mathrm{A}}_{\mathrm{C},a}$, $Z^{\mathrm{A}}_{\mathrm{C},d}$, $Z^{\mathrm{B}}_{\mathrm{C},a}$, and $Z^{\mathrm{B}}_{\mathrm{C},d}$. The indicator variable $Z^{\mathrm{A}}_{\mathrm{C},a}$ takes on the value 0 if individual C inherits A^{f}_a from his or her father, A, and takes on the value 1 if individual C inherits A^{m}_a from his or her father. Similarly, $Z^{\mathrm{B}}_{\mathrm{C},a}$ takes on the value 0 if individual C inherits B^{f}_a from his or her mother, B, and takes on the value 1 if he or she inherits B^{m}_a instead. The other two indicator variables related to the disease gene are similarly defined. The joint distribution of $Z^{\mathrm{A}}_{\mathrm{C},a}$ and $Z^{\mathrm{A}}_{\mathrm{C},d}$ is

$$P_\theta(Z^{\mathrm{A}}_{\mathrm{C},a},Z^{\mathrm{A}}_{\mathrm{C},d}) = \begin{cases} (1-\theta)/2 & \text{if } (Z^{\mathrm{A}}_{\mathrm{C},a},Z^{\mathrm{A}}_{\mathrm{C},d}) = (0,0) \\ (1-\theta)/2 & \text{if } (Z^{\mathrm{A}}_{\mathrm{C},a},Z^{\mathrm{A}}_{\mathrm{C},d}) = (1,1) \\ \theta/2 & \text{if } (Z^{\mathrm{A}}_{\mathrm{C},a},Z^{\mathrm{A}}_{\mathrm{C},d}) = (0,1) \\ \theta/2 & \text{if } (Z^{\mathrm{A}}_{\mathrm{C},a},Z^{\mathrm{A}}_{\mathrm{C},d}) = (1,0) \end{cases} . \tag{1}$$

The indicator variables $Z^{\mathrm{B}}_{\mathrm{C},a}$ and $Z^{\mathrm{B}}_{\mathrm{C},d}$ have a joint distribution, as given in equation (1). Whenever two associated indicator variables have different values, a recombination has occurred.

The joint distribution of the variables, A^{f}_a, A^{m}_a, A^{f}_d, A^{m}_d, $Z^{\mathrm{f(A)}}_{\mathrm{A},a}$, $Z^{\mathrm{f(A)}}_{\mathrm{A},d}$, $Z^{\mathrm{m(A)}}_{\mathrm{A},a}$, and $Z^{\mathrm{m(A)}}_{\mathrm{A},d} \in W$ (the set of variables in the extended pedigree) for all individuals $A \in V$ (the set of individuals) can be written as follows (where $V' \subset V$ is the set of individuals with no parents and where f(A) and m(A) denote the parents of A):

$$\begin{aligned} P_\theta(W) = & \prod_{A \in V'} P(A^{\mathrm{f}}_a)P(A^{\mathrm{m}}_a)P(A^{\mathrm{f}}_d)P(A^{\mathrm{m}}_d) \times \\ & \times \prod_{A \in V - V'} P_\theta(Z^{\mathrm{f(A)}}_{\mathrm{A},a},Z^{\mathrm{f(A)}}_{\mathrm{A},d})P_\theta(Z^{\mathrm{m(A)}}_{\mathrm{A},a},Z^{\mathrm{m(A)}}_{\mathrm{A},d}) \\ & \times \prod_{A \in V - V'} P(A^{\mathrm{f}}_a|\mathrm{f(A)}^{\mathrm{f}}_a,\mathrm{f(A)}^{\mathrm{m}}_a,Z^{\mathrm{f(A)}}_{\mathrm{A},a}) \\ & \quad \times P(A^{\mathrm{m}}_a|\mathrm{m(A)}^{\mathrm{f}}_a,\mathrm{m(A)}^{\mathrm{m}}_a,Z^{\mathrm{m(A)}}_{\mathrm{A},a}) \\ & \times \prod_{A \in V - V'} P(A^{\mathrm{f}}_d|\mathrm{f(A)}^{\mathrm{f}}_d,\mathrm{f(A)}^{\mathrm{m}}_d,Z^{\mathrm{f(A)}}_{\mathrm{A},d}) \\ & \quad \times P(A^{\mathrm{m}}_d|\mathrm{m(A)}^{\mathrm{f}}_d,\mathrm{m(A)}^{\mathrm{m}}_d,Z^{\mathrm{m(A)}}_{\mathrm{A},d}) \ . \end{aligned} \tag{2}$$

In the work of Kong (1991), the merits of the representation of equation (2), exemplified in figure 5, have been discussed. This representation results in more variables, but the conditional-probability tables of equation (2) are simpler and require less storage space. The representation is easily represented by a Bayesian network and can be handled immediately by blocking Gibbs sampling. Each of the components in equation (2) specifies a variable in a Bayesian network. The conditional distributions, furthermore, specify edges from parents (i.e.,

the conditioning variables) to a child. The pedigree in figure 5 thus shows an example of the Bayesian-network representation of equation (2).

For loci with incomplete penetrance, such as the disease locus shown in figure 1, terms for the penetrance probabilities must be multiplied by equation (2). For individual A with disease alleles A_d^f and A_d^m, the term would be $P_p(p|A_d^f,A_d^m)$ for phenotype p and penetrance probabilities P; thus, for all individuals, it would be $\Pi_{A\in V}P_p(p|A_d^f,A_d^m)$. The entire LQT pedigree is represented by a Bayesian network, exemplified in figure 5. To perform linkage analysis with blocking Gibbs sampling, we first pick a suitable value for (θ_0) such that all samples are produced conditional on this θ value. Then a starting configuration is found, and blocking Gibbs sampling can start. At each iteration, both the number of recombinations (n_r) and the number of nonrecombinations (n_{nr}) are counted. A recombination has occurred if a pair of associated indicator variables (e.g., $Z_{C,a}^A$ and $Z_{C,d}^A$) have different values, and nonrecombination is present if they have the same value. In the following discussion, this simple counting scheme will be referred to as "method 1."

However, this scheme can be refined. In figure 5, we examine the pair of indicator variables $Z_{C,a}^A$ and $Z_{C,d}^A$. If A_a^f is identical to A_a^m, we do not know which of these genes has been inherited by A. This means that we do not know whether a recombination has occurred. Counting this case as either a recombination or a nonrecombination corresponds to adding noise to the estimate. Therefore, leaving it out leads to a better estimate. In general, cases in which the parent is homozygous at one of the loci should be left out. In the following discussion, this refined counting scheme will be referred to as "method 2." Currently, linkage-analysis implementations using an MCMC method always use either method 1 or method 2. Another possibility (discussed in Thomas and Cortessis 1992) is to use $E(n_r,n_{nr}|\text{data},\theta,A)$.

However, method 2 can be refined even further. Consider the LQT pedigree shown in figure 1. For individuals with no offspring, it is relatively easy to directly estimate the probabilities of recombination in each iteration. With regard to individual 40, there are two θ values to consider—one relating to the inheritance from the father and one relating to the inheritance from the mother. For example, given that we know the values for the father's genes at both the disease locus (18_d^f,18_d^m) and the marker locus (18_a^f,18_a^m), we know the possible outcomes for the alleles inherited from the father of individual 40 (i.e., 40_a^f and 40_d^f). We can easily calculate the probability of each of the outcomes, as well as the probabilities of recombination and nonrecombination for each outcome. The probabilities of recombination for the outcomes are summed, and the probabilities of nonrecombination are summed. After a normalization, we have calculated the probability that a recombination has occurred in individual 40's inheritance from individual 18. This computation can be performed for all individuals without offspring. For individuals with offspring, the θ values are not independent, and the calculation cannot be performed easily. However, in figure 1, 42% of the individuals have no offspring, thereby making the benefit of this refined scheme great. In the following discussion, this counting scheme will be referred to as "method 3."

It is clear that each of the three methods defined above leads to an unbiased estimate of the full posterior distribution but that methods 2 and 3 will have a smaller Monte Carlo error, which is documented by the following results:

Example of method 3: In figure 9, an example configuration of individuals 18 and 20 is shown. In table 1, the computation leading us to the probabilities of recombination is shown.

First, given the configuration of the parents, the possible configurations for the offspring (individual 40) are found. We know that individual 40 has the genotype (2,4) at the marker locus, and, since we know that allele 2 (at the marker locus, *a*) must originate from the father (individual 18), that allele 4 (also at the marker locus) must originate from the mother (individual 20), and that individual 40 does not have the LQT disease, we have the three possible configurations shown in the second row of table 1; in the second column of the second row of this table, the format of the configurations is shown.

For each of these configurations, we check whether we have a recombination for the inheritance from the father and/or for the inheritance from the mother. With configuration 1, we have a recombination for inheritance

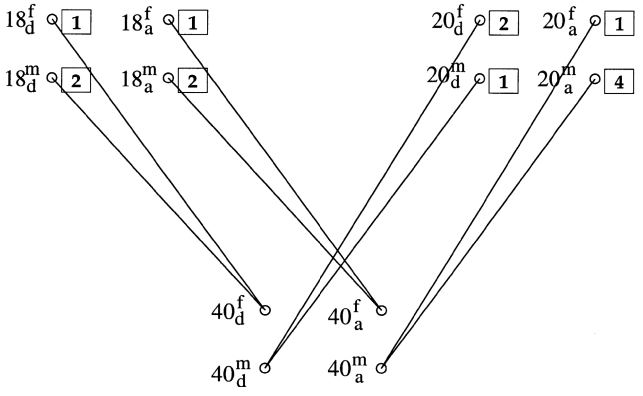


Figure 9 Sample configuration of individuals 18 and 20. Individual 18 has the disease genotype (1,2), meaning that he or she is unaffected but carries the disease allele (1). Also, individual 18 has the marker genotype (1,2). Individual 20 is unaffected as well, having the disease genotype (2,1) and the marker genotype (1,4).

Table 1

Status and Probabilities of Recombination Shown in Figure 9

	Configuration				Status		
Designation	40^{f}_{d}	40^{f}_{a}	40^{m}_{d}	40^{m}_{a}	From Father (Individual 18)	From Mother (Individual 20)	Probability
1	1	2	2	4	Recombinant	Recombinant	$(\frac{\theta_0}{2})^2$
2	2	2	1	4	Nonrecombinant	Nonrecombinant	$(\frac{1-\theta_0}{2})^2$
3	2	2	2	4	Nonrecombinant	Recombinant	$(\frac{1-\theta_0}{2}) - \frac{\theta_0}{2}$

from the father, since allele 1 (locus *d*) originates from the grandfather whereas allele 2 (locus *a*) originates from the grandmother. With configuration 1, we have a nonrecombination for inheritance from the mother, since both allele 2 (locus *d*) and allele 4 (locus *a*) originate from the grandfather.

Then, we compute the probability that each of these configurations will occur. For the first configuration, this probability is found simply by multiplication of the probability of one recombination and one nonrecombination. The probabilities of the other configurations are computed similarly.

Finally, the probability of recombination for inheritance from the father (individual 18) is calculated by first summing together the probability contributions from the configurations in which we have seen a recombination when inheritance is from the father and then dividing this number by the sum of the probability contributions.

In table 2, the three methods have been compared, for the LQT pedigree. For each method, 10 runs have been performed (at both 100 iterations and 1,000 iterations). The same seed for random numbers was used for the *i*th run of each method to make sure that the results are not coincidental. The complexities of the three methods are almost identical, since the more-complex computations performed in method 3 take negligible extra time compared with the remaining computations used in the blocking Gibbs sampler during each iteration. The average and SD of $\log_{10}[\widehat{L(\theta_1)}/L(\theta_0)]$, where $L(\theta)$ is the likelihood of θ, are shown in table 2; in this case, we use $\theta_0 = .2$ and $\theta_1 = .3$. In the following discussion, this value will be referred to as the "log-likelihood difference," since, in effect, it is the difference between the log-likelihoods for two values for θ. The LOD score corresponds to these log-likelihood differences in the following way: $\log_{10}[\widehat{L(\hat{\theta}_1)}/L(\theta_2)]$, where $\hat{\theta}$ is the θ value that has the maximum likelihood. The relationship between the log-likelihood difference shown in table 2 and the LOD score will be further elaborated later in this section. For now, it suffices to state that the log-likelihood differences shown in table 2 are used in the computation of the LOD score and that, as they become more precise, the LOD score becomes more precise.

The SD shown in the second column of table 2, SD(1), is computed by the autocorrelations method described by Geyer (1991). This SD expresses the variation within the dependent Markov-chain samples. Since the runs for the three methods are based on the same seed for random numbers, it is clearly seen that the SD of the estimate can be lowered significantly by use of the more advanced methods. Use of method 2 instead of method 1 basically corresponds to the removal of noise from the method 1 estimate and, thus, to obtaining a smaller SD. Further noise is removed when method 3 is applied. Since method 3 is clearly the optimal of the three methods, it has been used in all subsequent runs.

In the Appendix, we describe how to estimate LOD scores in a pedigree such as this, but here we will focus on the blocks created by the blocking Gibbs sampler.

Description of the Created Blocks

The Bayesian network corresponding to the linkage analysis shown in figure 1 contains 915 variables. The

Table 2

Comparison of Three Counting Methods and the Exact Method, for Log-Likelihood Difference log[*L*.2)/*L*(.3)], with LQT Pedigree

No. of Iterations and Method	Average (SDs) for Log-Likelihood Difference $\log[L(.2)/L(.3)]$[a]
100:	
1	1.9 (.22, .3)
2	1.93 (.032, .07)
3	1.90 (.028, .04)
1,000:	
1	1.88 (.023, .05)
2	1.87 (.013, .03)
3	1.91 (.0037, .02)
Exact method	1.85 (..., ...)

Note.—A total of 20 runs were performed for each method—10 with 100 iterations and 10 with 1,000 iterations.

[a] The average is that over the 10 log-likelihood differences; the first value in parentheses is the SD of the Markov chain, found by use of the autocorrelations method of Geyer (1991), and the second value in parentheses is the SD over the 10 results.

Table 3

Results of Simple$_1$ Method, for Different Iterations and for the Exact Method

No. of Iterations	Average (SD) for Log-Likelihood Difference[a]					
	log[$L(.0)/L(.01)$]: $\theta_0 = .0$, $\theta_1 = .01$	log[$L(.01)/L(.1)$]: $\theta_0 = .01$, $\theta_1 = .1$	log[$L(.1)/L(.2)$]: $\theta_0 = .1$, $\theta_1 = .2$	log[$L(.2)/L(.3)$]: $\theta_0 = .02$, $\theta_1 = .3$	0 $\theta_0 = .3$, $\theta_1 = .4$	log[$L(.4)/L(.5)$]: $\theta_0 = .4$, $\theta_1 = .5$
10	.18 (.01)	1.6 (.2)	1.92 (.07)	1.8 (.2)	1.5 (.3)	.9 (.5)
100	.184 (.004)	1.72 (.04)	1.97 (.03)	1.90 (.04)	1.66 (.07)	1.0 (.2)
1,000	.186 (.001)	1.73 (.02)	1.966 (.008)	1.91 (.02)	1.63 (.07)	1.05 (.04)
10,000	.1856 (.0003)	1.73 (.01)	1.966 (.006)	1.912 (.008)	1.64 (.01)	1.02 (.02)
Exact method	.1859	1.70	1.912	1.851	1.592	1.02

[a] Data are those for 10 log-likelihood differences. θ_0 is the θ value used during each run.

reason for this is that several variables are created for each individual in the pedigree—for example, for individual 56, 56_d^{f}, 56_a^{f}, 56_d^{m}, 56_a^{m}, 56_a^{G}, 56_d^{G}, 56_d^{P}, $Z_{56,a}^{54}$, $Z_{56,d}^{54}$, $Z_{56,a}^{55}$, $Z_{56,d}^{55}$, Z_{56}^{54}, and Z_{56}^{55}. These variables have all been introduced in preceding sections.

A few more variables are created in order to take account of heterozygous individuals. The indicator variables discussed above are not created for the 12 nodes without parents.

All in all, there are a total of 915 variables in the Bayesian network, spread over 10 generations.

Obviously, there is plenty of redundant information in the variables discussed above, and, if some extra bookkeeping is performed in a practical implementation, it would be possible to leave out some of them. However, in this application, it was decided not to do this, since it was not necessary. With regard to the storage requirements of the junction trees, it makes little difference, since the same loops will be present even if the variables are reduced to the bare minimum. The enormous storage requirements of the junction trees are caused by the presence of hundreds of loops, and reductions in the variables associated with each individual will reduce the total storage requirements by only a small linear factor.

What is more important is the presence of several logically related variables in the set of variables for each individual. This usually severely decreases the mixing properties of single-site Gibbs samplers; however, for a blocking Gibbs sampler that updates almost all variables jointly, this should not pose a significant problem. It remains to be investigated whether reduction of the set of logical dependences would significantly improve the mixing properties of the blocking Gibbs sampler.

The junction-tree representation necessary for this Bayesian network that is necessary for belief updating requires 1.5 gigabytes, which is beyond the processing capacity of most contemporary computers. When 10 blocks are created in a random run with blocking Gibbs sampling, the average size of these junction trees will be 1.7 megabytes. The total memory requirement of these 10 blocks therefore will be 17 megabytes, which is ~100 times less than that of the initial junction tree.

As a rule of thumb, the number of blocks in a given blocking Gibbs sampling scheme should usually be 2–10, and often the optimal number of blocks is ~5. The *optimal* number of blocks in this respect is the number of blocks that yields both the lowest total storage requirement for the blocks and the fastest updating scheme. Usually, when the number of blocks is >4–5, the size of the blocks does not decrease very much, so the time required by each iteration is increased. Still, almost all variables are sampled in each block; thus, on the average, each variable is sampled N times each iteration if there are X blocks. In the work of Jensen et al. (1995), several rules of thumb have been provided regarding how to balance the number of blocks, with respect to the lowest total storage requirements, the speed of the updating scheme, and the mixing properties of the resulting Markov chain.

It is very difficult to provide a visual image of what these 10 blocks look like, but a few statistics can be provided. On average, the blocks consist of ~96% of the 915 variables in the network:

1. 880 variables
2. 880 variables
3. 882 variables
4. 882 variables
5. 878 variables
6. 871 variables
7. 871 variables
8. 874 variables
9. 874 variables
10. 872 variables

In general, a block tends to comprise variables of distantly related subjects and, usually, only one or a few of the variables associated with each individual. Usually, the variables for an individual are involved in the same loops, so inclusion of just one of them in the block is sufficient to break these loops. So, for some number of individuals, one or a few of the associated variables are chosen to break the loops; however, it is not the case that the same types of variables are chosen for each

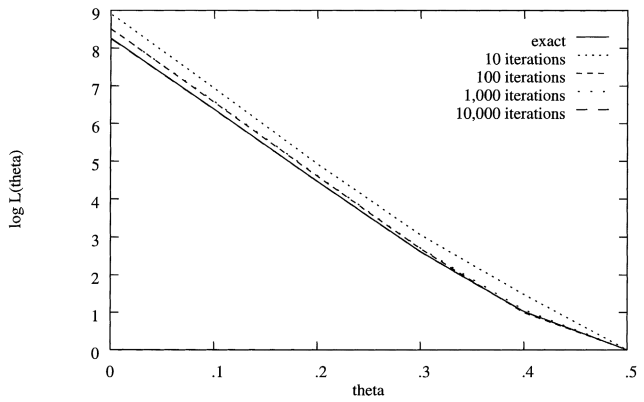


Figure 10 Log likelihood of θ (*log L*(*theta*)) plotted against θ. For 10, 100, and 1,000 iterations, the maximum-likelihood estimate for θ clearly is .0, and it can be seen that the graphs converge toward the exact curve.

individual. The type of the variable is of no consequence to the algorithm that selects the variables—all that matters is the reduction in storage requirements. This further substantiates the basic claim of the blocking Gibbs sampler—that is, that, by removing <10% of the variables, it can reduce to a manageable size the memory requirements of belief updating. In this case, the memory requirements of the junction tree have been reduced to 0.1% by removing only 4% of the variables.

Obviously, there must be a high degree of overlap among the blocks, since they all contain almost all the variables. There is also significant overlap between the variables that are not present in the blocks. Obviously, these are among the best variables to remove in order to reduce memory requirements, and it is optimal to remove, from as many blocks as possible, the very best of these best variables.

In the implemented version of blocking Gibbs sampling, the number of blocks from which a variable is allowed to be removed can be controlled. In this example, each variable was allowed to be removed from all blocks except one, to ensure that it would be sampled at least once every iteration. This results, on average, in >50% overlap between the variables removed from two blocks.

Results of Linkage Analysis

In this section, blocking Gibbs sampling is applied to the LQT pedigree; the results are presented in table 3. To better understand some of these results, it may be necessary to review the Appendix. Each element in table 3 represents the log-likelihood differences for a number of pairs of θ values—for example, $\log_{10}[\widehat{L(.2)}/L(.1)]$. A good estimate for the LOD score—that is, $\log_{10}[\widehat{L(\hat{\theta}_1)}/L(.5)]$ (with $\hat{\theta} = .0$, since the LQT-pedigree data are simulated to be in close linkage)—can be found by adding together the differences. Each of the differences shown in table 3—$\log[\widehat{L(.0)}/L(.01)]$, $\log[\widehat{L(.01)}/L(.1)]$, $\log[\widehat{L(.1)}/L(.2)]$, $\log[\widehat{L(.2)}/L(.3)]$, $\log[\widehat{L(.3)}/L(.4)]$, and $\log[\widehat{L(.4)}/L(.5)]$—corresponds to a piece of the graph shown in figure 10. The maximum-likelihood estimate of θ can be found in figure 10 by finding the highest point on the graph, and the LOD score can be read as the log-likelihood difference of this point. Furthermore, the exact results have been plotted in figure 10 and can be compared with the estimates. Clearly, the estimates converge toward the exact results. At each log-likelihood difference, 10 runs have been made, and the values shown in the table are the means and SDs of these runs. (On a SPARCstation-20, a 1,000-iteration run took ~9 h.)

It would be interesting to investigate whether multiplying together six independent log-ratio estimates is more efficient than a direct estimate based on a single series of six times the length.

On the basis of the data shown in table 3, it can be seen that the most likely θ value is indeed 0. This can be seen when we examine the cumulative sum of log-likelihood differences across the columns, plot them against θ, and find the maximum at 0. Adding the log-likelihood differences together, we find the LOD score to be ~8, thus providing strong evidence of tight linkage.

Furthermore, it can be seen that the accuracy of the estimates improves significantly when more iterations are performed. After 10,000 iterations, the SD is ~1%

Table 4

Comparison of Four Counting Methods and the Exact Method, for Log-Likelihood Difference log[*L*(.3)/*L*(.4)]

No. of Interactions and Method	Average (SD) for Log-Likelihood Difference log[*L*(.3)/*L*(.4)][a]
100:	
Simple_1	1.66 (.07)
Simple_2	1.50 (.05)
Square root	1.58 (.04)
Iterative	1.59 (.04)
1,000:	
Simple_1	1.63 (.07)
Simple_2	1.53 (.02)
Square root	1.59 (.01)
Iterative	1.60 (.01)
10,000:	
Simple_1	1.64 (.01)
Simple_2	1.534 (.005)
Square root	1.591 (.005)
Iterative	1.599 (.005)
Exact method	1.592 (...)

[a] Data for the methods are as follows: simple_1, results at $\theta_0 = .3$; simple_2, results at $\theta_0 = .4$; square root, results at $\theta_0 = .3$ and $\theta_0 = .4$, for the square-root method; iterative, combined results at $\theta_0 = .3$ and $\theta_0 = .4$, for the iterative method.

Table 5

Comparison of Results with Different Estimation Methods, at 10,000 Iterations

	Average (SD[a]) for the Log-Likelihood Difference				
Method	$\log[L(.01)/L(.1)]$: $\theta_1 = .01$, $\theta_0 = .1$	$\log[L(.1)/L(.2)]$: $\theta_1 = .1$, $\theta_0 = .2$	$\log[L(.2)/L(.3)]$: $\theta_1 = .2$, $\theta_0 = .3$	$\log[L(.3)/L(.4)]$: $\theta_1 = .3$, $\theta_0 = .4$	$\log[L(.4)/L(.5)]$: $\theta_1 = .4$, $\theta_0 = .5$
Simple_1	1.73 (.01)	1.966 (.006)	1.912 (.008)	1.64 (.01)	1.02 (.02)
Simple_2	1.664 (.001)	1.856 (.002)	1.783 (.002)	1.524 (.005)	.968 (.007)
Simple_3	1.694 (.002)	1.910 (.003)	1.849 (.004)	1.591 (.006)	1.02 (.01)
Square root	1.700 (.001)	1.912 (.002)	1.850 (.003)	1.591 (.005)	1.019 (.005)
Iterative	1.7035 (.0009)	1.914 (.002)	1.854 (.003)	1.599 (.005)	1.031 (.006)
Exact	1.6990	1.912	1.851	1.592	1.024

[a] Average over 10 runs.

of the estimate for almost all of them, showing that very high precision can be obtained.

However, on the basis of the data shown in table 3, it can be seen that the estimates do not seem to converge toward the exact results; between 1,000 and 10,000 iterations, the estimates have not moved farther toward the exact results, indicating that the former do not converge toward the latter. The reason for this is that, as has been explained above, the estimator used in table 3—which is called "simple_1" in table 4—is not optimal. Thus, even if an infinite number of iterations are performed, we will not be able to obtain better results by means of the method given in table 3. However, better results can be obtained in several ways, as can be seen in table 5. Here, we compare the results for the simple_1 and simple_2 methods with those for the "square-root" and "iterative" estimation methods and with those for a new method, denoted "simple_3." The simple_3 method uses the runs that also have been used in table 3, but in a more clever way. For example, instead of estimating $\log[\widehat{L(.2)}/L(.1)]$ only by using the results at $\theta_0 = .1$, we can use the results at $\theta_0 = .1$, to compute $\log[\widehat{L(.1)}/L(.15)]$, and the results at $\theta_0 = .2$, to compute $\log[\widehat{L(.15)}/L(.2)]$, and then

$$\log\frac{\widehat{L(.2)}}{L(.1)} = \log\frac{\widehat{L(.1)}}{L(.15)} + \log\frac{\widehat{L(.15)}}{L(.2)} \ .$$

Using this method allows us to get better estimates than can be made with either simple_1 or simple_2, as is seen clearly in table 5. In fact, the results obtained with simple_3 are almost as good as those obtained with the square-root method. As expected, the square-root method is still the optimal method and is significantly better than the iterative method.

Discussion

We have described a general method for inference in very large, complex Bayesian networks and have applied it successfully to a particularly hard problem in genetics—namely, linkage analysis. The results shown in table 3 document that the algorithm converges toward a distribution that is close to the correct distribution and that it mixes fast. Single-site Gibbs sampling would have been totally useless if applied to a problem of this size. The size and complexity of the LQT pedigree is just within the limits of exact methods, so we have been able to check the accuracy of the results.

The blocking Gibbs method has been shown to be successful in this very difficult case, and it can easily be applied to larger problems. Because of the way in which the blocks are selected, the method scales well, and we expect multipoint linkage analysis to pose no further theoretical problems. It will merely make the storage requirements greater.

The major problem still remaining with the method is that we cannot yet prove that it is irreducible in the general case. This requires the construction of a general method for finding the noncommunicating classes of the Gibbs sampler, such as has been discussed by Lin et al. (1994).

At this point, however, it is uncertain whether such a general method can be found. Furthermore, the problem of detecting these classes may be NP-hard. If these classes were identified, it would be possible to design blocks tailored to allow the blocking Gibbs sampler to jump between the classes, thus guaranteeing irreducibility of the sampler.

In practice, it is often possible to design these blocks by hand, and the built-in robustness of the blocking Gibbs sampler—which lets it usually sample >90% of variables jointly—will render it irreducible in most cases. Of course, this still cannot be guaranteed in the general case. In the LQT pedigree, we experience only reducibility problems that are due to such gene representation as has been described in the Criterion 3 subsection, above. These problems can be handled automatically by the blocking Gibbs software.

Acknowledgments

C.S.J. would like to thank the other members of the decision-support-systems group at Aalborg University, for providing a stimulating environment, and one of the anonymous referees, for tiresome work on improvement of this article. This research was supported by the Danish Research Councils, through the Program for Informatik i Forskning og Teknologi.

Appendix

Estimation of LOD Scores

The θ is estimated by use of the number of recombinations n_r, the number of nonrecombinations n_{nr}, and the calculated θ for bottom-level individuals $r_1,\ldots,r_k$, by the means of equation (A1). If M iterations have been performed with the blocking Gibbs sampler at a fixed recombination fraction θ_0, then the likelihood ratio can be estimated by the following expression:

$$\frac{\widehat{L(\theta_1)}}{L(\theta_0)} = \frac{1}{M}\sum_{k=1}^{M}\left(\frac{\theta_1}{\theta_0}\right)^{n_r^{(k)}}\left(\frac{1-\theta_1}{1-\theta_0}\right)^{n_{nr}^{(k)}} \times\left\{\prod_{i=1}^{n_r^{(k)}}\left[\frac{\theta_1}{\theta_0}r_i + \frac{1-\theta_1}{1-\theta_0}(1-r_i)\right]\right\}, \tag{A1}$$

where $n_r^{(k)}$ is the number of calculated probabilities of recombination in bottom-level individuals at iteration k.

In general, let y be the simulated data and let R_i be the 0-or-1 variable denoting whether there is a recombination with meiosis i. Then, in general, the contribution of a meiosis is the term in the curly braces in equation (A1), where

$$r_i = E(R_i = 1|y,\theta_0) = P(R_i = 1|y,\theta_0) .$$

When the data determine R_i, so that r_i is either 0 or 1, then the term in the curly braces reduces to either θ_1/θ_0 or $(1-\theta_1)/(1-\theta_0)$. When the data are not informative at all, then $r_i = P(R_i = 1|\theta_0) = \theta_0$, and the term in the curly braces reduces to 1 and can be ignored.

The LOD score can be found by maximizing the ratio ofequation (A1) over θ_1 and applying $\log_{10}$.

By means of methods presented in the work of Meng and Wong (1996), it is now possible to combine the results from two runs that have different θ values. We present the first results of use of these methods in a practical application. The theory underlying the methods is described very well in the work of Meng and Wong (1996) and will not be discussed in detail here. We will present only the formulas that have been used in the present article, and the notation has been changed to better suit this application. Assume that we have run 100 iterations at $\theta_0 = .1$ and 100 iterations at $\theta_1 = .3$. Now, if we want to compute $\log[\widehat{L(.3)}/L(.1)]$, instead of usingequation (A1) and only the run at .1, we can use both runs. As more information is included in the computation, it is to be expected that by using this method we will get better results. In the work of Meng and Wong (1996), various methods for performing this estimation are discussed. We will compare two of these methods versusequation (A1).

Equation (A1) can be expressed as follows:

$$\frac{\widehat{L(\theta_1)}}{L(\theta_0)} = E_0\left[\frac{q_1(w_0|\theta_1)}{q_0(w_0|\theta_0)}\right] , \tag{A2}$$

where w_i is the number of observations, given θ_i; where E_0 implies that we average over all observations, given $\theta = \theta_0$; and where $q_i[w_0|\theta_i]$ is a function that estimates some quantity from samples w_0 generated with $\theta = \theta_0$—but at the value of $\theta = \theta_i$, which may or may not be equal to θ_0.

In our case,

$$\frac{q_1(w_0^k|\theta_1^k)}{q_0(w_0^k|\theta_0)} = \left(\frac{\theta_1}{\theta_0}\right)^{n_r^{(k)}}\left(\frac{1-\theta_1}{1-\theta_0}\right)^{n_{nr}^{(k)}} \times\left\{\prod_{i=1}^{n_r^{(k)}}\left[\frac{\theta_1}{\theta_0}r_i + \frac{1-\theta_1}{1-\theta_0}(1-r_i)\right]\right\},$$

where the samples $n_r^{(k)}$, $n_{nr}^{(k)}$ and r_i have been generated with $\theta = \theta_0$, and thus $w_0^k = \{n_r^{(k)}, n_{nr}^{(k)}, r_i\}$.

Equation (A1) is actually a generalization of an equation that states

$$\frac{\widehat{L(\theta_1)}}{L(\theta_0)} = \frac{E_0[q_1(w_0|\theta_1)\alpha(w_0)]}{E_1[q_0(w_1|\theta_0)\alpha(w_1)]} , \tag{A3}$$

where $\alpha(w)$ is an arbitrary function. Different choices of $\alpha(w)$ are discussed in the work of Meng and Wong (1996). If $\alpha(w) = 1/[q_0(w|\theta_0)]$, then equation (A1) reduces toequation (A2).

As mentioned, we will look at two choices for $\alpha(w)$:

1. $\alpha = 1/\sqrt{q_0 q_1}$. With this value of α, equation (A3) looks likeequation (A4). In the following discussion, this method (which, in the work of Meng and Wong [1996], is called the "geometric-mean method") will be referred to as the "square-root method":

$$\frac{\widehat{L(\theta_1)}}{L(\theta_0)} = \frac{E_0\left(\sqrt{\frac{q_1(w_0|\theta_1)}{q_0(w_0|\theta_0)}}\right)}{E_1\left(\sqrt{\frac{q_0(w_1|\theta_0)}{q_1(w_1|\theta_1)}}\right)} . \tag{A4}$$

2. $\alpha = c/s_1q_1 + s_0rq_0$, where c is a constant and where, if n_0 is the number of observations with θ_0, n_1 is the number of observations with θ_1, and $n = n_0 + n_1$, then $s_0 = n_0/n$ and $s_1 = n_1/n$. α furthermore depends on the ratio r, which is computed in an iterative fashion and is defined in the following discussion. Inequation (A5), the iterative estimator is shown:

$$\frac{\widehat{L(\theta_1)}}{L(\theta_0)} = \frac{E_0\left[\frac{q_1(w_{0i}|\theta_1)}{s_1q_1(w_{0i}|\theta_1)+s_0rq_0(w_{0i}|\theta_0)}\right]}{E_1\left[\frac{q_0(w_{1i}|\theta_0)}{s_1q_1(w_{1i}|\theta_1)+s_0rq_0(w_{1i}|\theta_0)}\right]} \quad . \tag{A5}$$

Starting with an initial guess of the value of r, $\hat{r}^{(0)}$, we calculate the estimate of r iteratively, by using the previous estimate of r. Specifically, at the $(t + 1)$st iteration, we compute

$$\hat{r}^{(t+1)} = \frac{\frac{1}{n_0}\sum_{i=1}^{n_0}\left[\frac{q_1(w_{0i}|\theta_1)}{s_1q_1(w_{0i}|\theta_1)+s_0\hat{r}^{(t)}q_0(w_{0i}|\theta_0)}\right]}{\frac{1}{n_1}\sum_{i=1}^{n_1}\left[\frac{q_0(w_{1i}|\theta_0)}{s_1q_1(w_{1i}|\theta_1)+s_0\hat{r}^{(t)}q_0(w_{1i}|\theta_0)}\right]}$$

$$= \frac{\frac{1}{n_0}\sum_{i=1}^{n_0}\left[\frac{l_{0i}}{s_1l_{0i}+s_0\hat{r}^{(t)}}\right]}{\frac{1}{n_1}\sum_{i=1}^{n_1}\left[\frac{1}{s_1l_{1i}+s_0\hat{r}^{(t)}}\right]} \quad ,$$

where $l_{0i} = [q_1(w_{0i}|\theta_1)/q_0(w_{0i}|\theta_0)]$ and $l_{1i} = [q_1(w_{1i}|\theta_1)/q_0(w_{1i}|\theta_0)]$. These values need only be calculated once, at the beginning of the algorithm.

There is one problem with this method. When the samples are independent, we know the effective sample sizes, n_0 and n_1; however, with dependent samples, such as those of the blocking Gibbs sampler, n_0 and n_1 are no longer the true sample sizes, since the dependence between successive samples typically reduces the "effective" sample sizes, and thus the use of n_0 and n_1 may lead to simulation errors.

The square-root method is a new method, which first was presented in the work of Meng and Wong (1996), but the iterative method is not new. It has been discussed, in the area of physics, in the work of Bennet (1976). The square-root method is an interesting addition, since it is sometimes desirable to have simple, noniterative procedures that have good—albeit not necessarily optimal—properties. Such a noniterative estimator can be used, for example, as a starting value for the iterative method. As can be seen in table 4, a noniterative estimator can be better than the iterative estimator, when the samples are not independent. The potential for the simple identity ofequation (A3) has been further investigated in the work of Gelman and Meng (1994, 1996) and Meng and Schilling (1996).

In table 4, results from runs with the three previously described methods are shown. A total of 10 new runs have been performed for each of the methods and for 100 and 1,000 iterations. It is seen that the log-likelihood differences found by combining the results from two runs with different θ_0 values are consistently better than those obtained by using the results of only a single run.

It is also interesting to note that the noniterative square-root method seems to give results that are significantly closer to the exact value than are those given by the iterative method. This is probably due to the fact that (*a*) the optimality of the iterative method was derived under the independence assumption but (*b*) the samples of the blocking Gibbs sampler are dependent. In such cases, it has been speculated (in Meng and Wong 1996) that the square-root method can be better. This is a useful result, showing that the simple, noniterative estimator can be better than the more complex, iterative method, in cases in which samples are dependent. Furthermore, with the blocking Gibbs sampler, the samples are much less dependent than they are with the single-site Gibbs sampler, indicating that, in this case, the square-root method may be significantly better than the iterative method.

References

Andersen SK, Olesen KG, Jensen FV, Jensen F (1989) HUGIN—a shell for building Bayesian belief universes for expert systems. In: Sridharan NS (ed) Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, pp 1080–1085

Bennett CH (1976) Efficient estimation of free energy differences from Monte Carlo data. J Comput Physics 22: 245–268

Cannings C, Thompson EA, Skolnick MH (1976) The recursive derivation of likelihoods on complex pedigrees. Adv Appl Prob 8:622–625

——— (1978) Probability functions on complex pedigrees. Adv Appl Prob 10:26–61

Cottingham RW Jr, Idury RM, Schäffer AA (1993) Faster sequential genetic linkage computations. Am J Hum Genet 53: 252–263

Dawid AP (1992) Applications of a general propagation algorithm for probabilistic expert systems. Stat Comput 2: 25–36

Elston RC, Stewart J (1971) A general model for the genetic analysis of pedigree data. Hum Hered 21:523–542

Gelfand AE, Smith AFM (1990) Sampling-based approaches to calculating marginal densities. J Am Stat Assoc 85(410): 398–409

Gelman A, Meng XL (1994) Path sampling for computing normalizing constants: identities and theory. Tech rep 376. Department of Statistics, University of Chicago, Chicago

——— (1998) Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. Stat Sci 13(2): 163-185

Gelman A, Rubin DB (1992) Inference from iterative simu-

lation using single and multiple sequences. Stat Sci 7: 457–511

Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans Pattern Anal Machine Intell 6:721–741

Geyer CJ (1991) Markov chain Monte Carlo Maximum likelihood. In: Keramidas E (ed) Computing science and statistics: proceedings of the 23d Symposium on the Interface. Inteface Foundation of North America, Fairfax Station, VA, pp 156–163

——— (1992) Practical Markov Chain Monte Carlo. Stat Sci 7(4): 473–511

Geyer CJ, Thompson EA (1993) Annealing Markov chain Monte Carlo with applications to pedigree analysis. Res rep 589, School of Statistics, University of Minnesota, Minneapolis

Irwin M, Cox N, Kong A (1994) Sequential imputation for multilocus linkage analysis. Proc Natl Acad Sci USA 91: 11684–11688.

Jensen CS (1996*a*) A simple method for finding a legal configuration in complex Bayesian networks. Tech rep R-96-2046, Department of Computer Science, Aalborg University, Aalborg, Denmark

——— (1997) Blocking Gibbs sampling for interference in large and complex Bayesian networks with applications in genetics. PhD thesis, Department of Computer Science, Aalborg University, Aalborg, Denmark

Jensen CS, Kong A, Kjærulff U (1995) Blocking-Gibbs sampling in very large probabilistic expert systems. Int J Hum Comput Stud 42:647–666

Jensen CS, Sheehan N (1998) Problems with determination of noncommunicating classes for Monte Carlo Markov chain applications in pedigree analysis. Biometrics 54:416–425

Jensen FV (1996*b*) An introduction to Bayesian networks. UCL Press/Springer-Verlag, New York

Jensen FV, Lauritzen SL, Olesen KG (1990) Bayesian updating in causal probabilistic networks by local computations. Comput Stat Q 4:269–282

Kjærulff U (1993) Approximation of Bayesian networks through edge removals. Res rep IR-93-2007, Department of Computer Science, Aalborg University, Aalborg, Denmark

Kong A (1991) Efficient methods for computing linkage likelihoods of recessive diseases in inbred pedigrees. Genet Epidemiol 8:81–103

Kong A, Cox N, Frigge M, Irwin M (1993) Sequential imputation and multipoint linkage analysis. Genet Epidemiol 10:483–488

Lange K, Elston RC (1975) Extensions to pedigree analysis. I. Likelihood calculations for simple and complex pedigrees. Hum Hered 25:95–105

Lathrop GM, Lalouel JM (1984) Easy calculations of lod scores and genetic risks on small computers. Am J Hum Genet 36:460–465

Lathrop GM, Lalouel JM, Julier C, Ott J (1985) Multilocus linkage analysis in humans: detection of linkage and estimation of recombination. Am J Hum Genet 37:482–498

Lauritzen SL (1992) Propagation of probabilities, means and variances in mixed graphical association models. J Am Stat Assoc 87:1098–1108

Lauritzen SL, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems. J R Stat Soc [B] 50(2): 157–224

Lin S, Thompson E, Wijsman E (1994) Finding noncommunicating sets for Markov chain Monte Carlo estimations on pedigrees. Am J Hum Genet 54:695–704

Meng XL, Schilling S (1996) Fitting full-information item factor models and an empirical investigation of bridge sampling. J Am Stat Assoc 91:1254–1267

Meng XL, Wong WH (1996) Simulating ratios of normalizing constants via simple identity: A theoretical exploration. Stat Sin 6:831–860

Pearl J (1986) Fusion, propagation and structuring in belief networks. Artificial Intell 29:241–288

——— (1991) Probabilistic reasoning in intelligent systems: networks of plausible inference series in representation and reasoning, 2d ed. Morgan Kaufmann, San Mateo, CA

Schäffer AA, Gupta SK, Shriram K, Cottingham RW Jr (1994) Avoiding recomputation in linkage analysis. Hum Hered 44: 225–237

Shachter RD, Andersen SK, Szolovits P (1994) Global conditioning for probabilistic inference in belief networks. In: Lopez de Mantaras R, Poole D (eds) Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, pp 514–522

Sheehan N, Thomas A (1993) On the irreducibility of a Markov chain defined on a space of genotype configurations by a sampling scheme. Biometrics 49:163–175

Shenoy PP, Shafer GR (1990) Axioms for probability and belief-function propagation. In: Shachter RD, Levitt TS, Kanal LN, Lemmer JF (eds) Uncertainty in artificial intelligence. Vol 4. Elsevier Science (North-Holland), Amsterdam, pp 169–198

Smith AFM, Roberts GO (1993) Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. J R Stat Soc [B] 55(1): 5–23

Thomas A, Spiegelhalter DJ, Gilks WR (1992) BUGS: A program to perform Bayesian inference using Gibbs sampling. In: Bernardo JM, Berger JO, Dawid AP, Smith AFM (eds) Bayesian statistics. Vol 4. Clarendon Press, Oxford, pp 837–842

Thomas DC, Cortessis V (1992) A Gibbs sampling approach to linkage analysis. Hum Hered 42:63–76