BIOINFORMATICS

Vol. 18 Suppl. 1 2002
Pages S189–S198

Exact genetic linkage computations for general pedigrees

M. Fishelson and D. Geiger

Computer Science Department, Technion, Haifa, 32000, Israel

Received on January 24, 2002; revised and accepted on March 26, 2002

ABSTRACT

Motivation: Genetic linkage analysis is a useful statistical tool for mapping disease genes and for associating functionality of genes with their location on the chromosome. There is a need for a program that computes multipoint likelihood on general pedigrees with many markers that also deals with two-locus disease models.

Results: In this paper we present algorithms for performing exact multipoint likelihood calculations on general pedigrees with a large number of highly polymorphic markers, taking into account a variety of disease models. We have implemented these algorithms in a new computer program called SUPERLINK which outperforms leading linkage software with regards to functionality, speed, memory requirements and extensibility.

Availability: SUPERLINK is available at http://bioinfo.cs.technion.ac.il/superlink

Contact: fmaayan@cs.technion.ac.il; dang@cs.technion.ac.il

Keywords: Bayesian networks; Fastlink; Genehunter; linkage analysis; Vitesse.

INTRODUCTION

Multipoint linkage analysis has become an integral part of mapping disease genes and constructing genetic maps. Currently, there are two main approaches to computing pedigree likelihood exactly: Elston–Stewart (Elston and Stewart, 1971) and Lander–Green (Kruglyak *et al.*, 1995, 1996; Lander *et al.*, 1987). Both algorithms are variants of variable elimination methods that depend on different strategies to finding an elimination order (e.g., Dechter, 1998). The Elston–Stewart algorithm proceeds by 'peeling' one nuclear family at a time. The Lander–Green algorithm, which is based on a hidden Markov model (HMM), proceeds by 'peeling' one locus at a time.

The complexity of the Lander–Green algorithm is linear in the number of loci, but exponential in the number of non-founders in the pedigree (non-founders are individuals whose parents are in the pedigree). On the other hand, the complexity of the Elston–Stewart algorithm is linear in the number of individuals (for sufficiently simple pedigrees) and exponential in the number of loci. It is clear that each of these approaches is more suitable for a different class of linkage problems. The Elston–Stewart algorithm can handle large pedigrees with a few markers more efficiently, whereas the Lander–Green approach is better equipped for dealing with small to medium-sized pedigrees and a large number of markers. Over the years the computational boundaries of both algorithms have been extended. However, using only one of these algorithms still limits the class of problems that can be handled effectively.

In SUPERLINK we used the framework of Bayesian networks as the internal representation of linkage analysis problems. Using this representation allows us to give a unified treatment to the entire spectrum between these approaches and to handle a wide variety of linkage analysis problems. The choice of elimination order is made automatically according to the linkage problem at hand. This paper presents several algorithms that have been integrated in SUPERLINK to support efficient multipoint likelihood calculations on general pedigrees with a large number of highly polymorphic markers. We present experimental results for these algorithms on a variety of semi-artificial data sets and demonstrate the superior performance of SUPERLINK versus the performance of existing linkage software, FASTLINK (Cottingham *et al.*, 1993; Schäffer *et al.*, 1994; Becker *et al.*, 1998), GENEHUNTER (Kruglyak *et al.*, 1996) and VITESSE (O'connell and Weeks, 1995).

The paper is organized as follows. First we survey basic genetic terminology, elaborate on definitions and methods related to Bayesian networks, and explicate the representation of pedigrees as Bayesian networks. Then, we describe the main algorithmic principles behind SUPERLINK, highlight some special features of SUPERLINK, and report experimental results. Finally, we explain the differences between SUPERLINK and other leading linkage programs and outline future work.

BACKGROUND

Basic genetic terminology

Genes are the basic unit of genetic information. Each gene

resides at a different place, or *locus*, on the *chromosome*. Except for the sex chromosomes, there are two genes at every locus and these constitute the individual's *genotype* at that locus. Genotypes are not always observable. The expression of a genotype is termed a *phenotype*.

In the transmission of genes from parents to children, each parent contributes one allele from his genotype. The sequence of alleles at different genes that are received by an individual from one parent is called a *haplotype*. We say that a *recombination* occurred between two genes if the haplotype of an individual contains two alleles that resided in different haplotypes in the individual's parent. The measure that is used for estimating whether a recombination occurred is called the *recombination fraction* and is denoted by θ. The goal of linkage analysis is to estimate θ between a disease gene and known loci on the chromosome. This measure translates to an approximate tentative physical location of a disease gene on a chromosome. For more details we refer the reader to Terwilliger and Ott (1994); Lange (1997).

Bayesian networks

Consider a directed acyclic graph G, namely, a directed graph with no directed cycles, such that each vertex v corresponds to a variable X_v and is associated with a probability distribution $P(X_v = x_v \mid \mathbf{Pa_v} = \mathbf{pa_v})$ where $\mathbf{Pa_v}$ are the variables corresponding to vertices that have edges leading into v. Further, define the joint probability distribution for $X_1, \ldots, X_n$ via

$$P(x_1, \ldots, x_n) = \prod_{v=1}^{n} P(x_v \mid \mathbf{pa_v}) \tag{1}$$

The directed acyclic graph together with the joint probability distribution is called a *Bayesian network* (Pearl, 1988; Lauritzen, 1996).

Note that in the above definition, and throughout this paper, we use capital letters for variable names and lowercase letters to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters, and assignments of values to the variables in these sets are denoted by boldface lower case letters. We also use $P(x)$ as a short hand notation for $P(X = x)$.

We define the *inference problem* as follows. The input is a Bayesian network along with a subset of vertices E. The output is the probability table $P(\mathbf{E} = \mathbf{e})$ for a given disjoint subset of variables $\mathbf{E} \subseteq \{X_1, \ldots, X_n\}$. Evaluating the pedigree likelihood in linkage analysis is a special case of the above inference problem.

Suppose that $X_1, \ldots, X_k$ are the variables not in E, then using Equation (1),

$$\Pr(\mathbf{e}) = \sum_{x_1} \cdots \sum_{x_k} \Pr(x_1, \ldots, x_k, \mathbf{e})$$

$$= \sum_{x_1} \cdots \sum_{x_k} \prod_{i} \Pr(x_i \mid \mathbf{pa_i}).$$

This inference problem can be abstracted into the problem of evaluating expressions of the form

$$\mathcal{E} = \sum_{x_1} \cdots \sum_{x_k} \prod_{l} f_l(\mathbf{Y_l}) \tag{2}$$

Each f_l is a *factor* (or a table) that contains an entry for each value of $\mathbf{Y_l} \subseteq \{X_1, \ldots, X_k\}$. Two ways to compute this expression are: *variable elimination* and *conditioning*.

In variable elimination we eliminate one variable at a time, by summing over all the possible values for the variable, until the expression does not contain any summations. For example, assume that we want to eliminate X_k from the expression. This is done in several steps. First, we rearrange the order of summation so that the sum over X_k is the innermost. Then, we move all the terms $f_l(\mathbf{Y_l})$ where $X_k \notin \mathbf{Y_l}$ outside the summation over X_k. Suppose that the factors $f_1, \ldots f_m$ remain in the scope of the summation over X_k. A new factor $f(\mathbf{Y})$ which is a product of these k factors, defined over $\mathbf{Y} = \bigcup_{j=1}^{m} \mathbf{Y}_j$, is constructed:

$$f(\mathbf{Y}) = \prod_{j=1}^{m} f_j(\mathbf{Y}_j). \tag{3}$$

In the last step, we marginalize X_k out of $f(\mathbf{Y})$ by summing over all possible values of X_k. We obtain a new factor $f'(\mathbf{Y}')$, where $\mathbf{Y}' = \mathbf{Y} - \{X_k\}$:

$$f'(\mathbf{Y}') = \sum_{x_k} f(\mathbf{Y}). \tag{4}$$

Note that with these steps we have rewritten $\mathcal{E}$ of Equation (2) as

$$\mathcal{E} = \sum_{x_1} \cdots \sum_{x_{k-1}} f'(\mathbf{Y}') \prod_{l>m} f_l(\mathbf{Y_l}).$$

The resulting expression has the same general form of Equation (2). Therefore, other variables can now be eliminated by repeating the same sequence of steps.

The complexity of variable elimination is dominated by the largest factor created during the computation (Equation (3)), and it depends on the order of elimination. The problem of finding an optimal elimination order is important in many applications and is known to be NP-complete (Arnborg, 1985; Arnborg *et al.*, 1987).

The second approach to compute $\mathcal{E}$ in Equation (2) is to perform the calculation using *conditioning*. We compute $\mathcal{E}$ by considering expressions of the form

$$\mathcal{E}_{x_1} = \sum_{x_2} \cdots \sum_{x_k} \prod f_l^*(\mathbf{Y}_l^*),$$

where f_l^* is f_l restricted to the case where $X_1 = x_1$ and $\mathbf{Y}_\mathbf{l}^* = \mathbf{Y} - \{X_1\}$. Note that

$$\mathcal{E} = \sum_{x_1} \mathcal{E}_{x_1}.$$

The motivation for this approach is that computing $\mathcal{E}_{x_1}$ is easier than computing $\mathcal{E}$, since it involves one less summation and some of the factors are smaller. The complexity of this procedure depends on the number of possible joint assignments to the variables that we condition on. This approach is called global conditioning in Shachter *et al.* (1994).

The advantage of conditioning over variable elimination is the lower memory overhead. Once the probability of the evidence for a particular assignment to the variables that we condition on has been computed, only a single number needs to be stored. The main disadvantage of this approach is that in different evaluations of $\mathcal{E}_{x_1}$, identical subexpressions are often recomputed several times.

Bayesian networks for linkage analysis

Bayesian networks allow us to represent pedigrees in a detailed manner. They also enable us to encode appropriate independence assumptions. A pedigree, which is the input to a genetic linkage problem, defines a joint distribution over the *genotypes* and *phenotypes* of the individuals represented in the pedigree.

We use the following types of random variables, as suggested in (Friedman *et al.*, 2000), for representing a pedigree:

- **Genetic Loci.** We number by $1, 2, \ldots$ the *loci* of interest in the genetic analysis. For each individual i and locus j, we define random variables $G_{i,jp}$, $G_{i,jm}$ whose values are the specific alleles of locus j in individual i's *paternal* and *maternal* haplotypes, respectively. That is, $G_{i,jp}$ was inherited from i's father, and $G_{i,jm}$ was inherited from i's mother.

- **Phenotypes.** For each individual i and phenotype j, we define a random variable $P_{i,j}$ that denotes the value of the phenotype for individual i.

- **Selector variables.** Similar to Lander and Green's approach, we use auxiliary variables that denote the inheritance pattern in the pedigree. We denote by $S_{i,jp}$ and $S_{i,jm}$ the *selection* made by the meiosis that resulted in i's genetic makeup at locus j. Formally, if a denotes i's father, then

$$G_{i,jp} = \begin{cases} G_{a,jp} & \text{if } S_{i,jp} = 0 \\ G_{a,jm} & \text{if } S_{i,jp} = 1 \end{cases}$$

  $G_{i,jm}$ is defined in a similar way.

The above notation has to be slightly modified in the case of sex-linked loci.

Each local probability table in the Bayesian network is of one of the following forms:

- **Transmission models:** $\Pr(G_{i,jp}|G_{a,jp}, G_{a,jm}, S_{i,jp})$, $\Pr(G_{i,jm}|G_{b,jp}, G_{b,jm}, S_{i,jm})$, where a and b are i's parents in the pedigree.

- **Penetrance models:** $\Pr(P_{i,j}|G_{i,jp}, G_{i,jm})$

- **Recombination models:** $\Pr(S_{i,1p}) = \Pr(S_{i,1m}) = 0.5$, $\Pr(S_{i,jp}|S_{i,j-1p}, \theta_{j-1})$ and $\Pr(S_{i,jm}|S_{i,j-1m}, \theta_{j-1})$, where θ_{j-1} is the known or unknown recombination fraction between locus $j-1$ and locus j.

- **General population allele probabilities:** $\Pr(G_{i,jp})$, $\Pr(G_{i,jm})$, when i is a founder.

The likelihood $\Pr(\mathbf{e} \mid \theta)$ of the pedigree data is the product of all the local probability tables of the Bayesian network, marginalized over all the variables of the network that are not assigned a value by $\mathbf{e}$.

For an example of a fragment of a network that describes parents-child interaction in a simple 3-loci analysis, consider Figure 1. The dashed boxes contain all variables that describe a single individual's genotype or phenotype. In this model it is assumed that loci are mapped in the order 1, 2, and 3. Figure 1 also shows the penetrance model for this simple 3-loci analysis. In this example we assume that each phenotype variable depends on the genotype at a single locus. This is reflected by the fact that only edges from the two haplotypes of a single locus point into each phenotype variable.

ALGORITHMS IN SUPERLINK

In SUPERLINK we represent linkage analysis problems using Bayesian networks, as just described. Using this representation allows us to give a unified treatment to the two extreme approaches to calculating pedigree likelihood exactly, the Elston–Stewart approach and the Lander–Green approach, and to the full spectrum of combinations of these approaches. Whenever feasible, we use variable elimination alone to calculate the likelihood of the pedigree data. Otherwise, our algorithm combines variable elimination with conditioning to achieve the best time-space tradeoff given the memory available for the linkage analysis problem. The choice of variable elimination order is made automatically according to the parameters of the specific linkage problem.

Some of the crucial features of our program are the preprocessing steps performed on the Bayesian network that often result in a substantial reduction in the time and

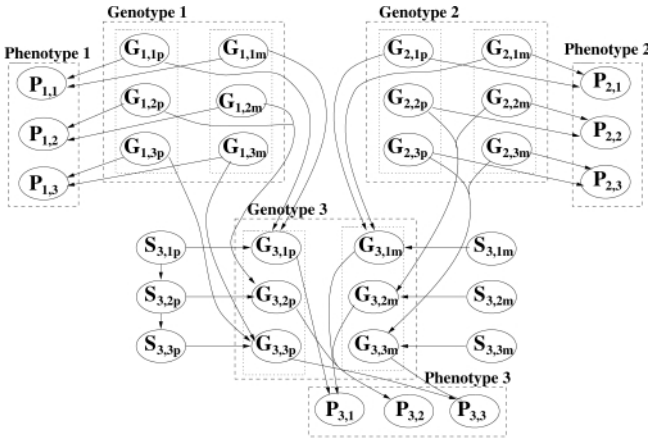

Fig. 1. A fragment of a Bayesian network representation of the transmission model and the penetrance model in a 3-loci analysis. Adapted from (Friedman *et al.*, 2000).

memory requirements. Preprocessing includes trimming redundant variables, merging some of the variables and reducing the range of values that are valid for each variable given the data.

Another crucial feature is the compact representation of multilocus genotype information and the efficient representation of probability tables.

Genotype representation

There are several possible approaches to storing multilocus genotype information. One possible method, adapted by FASTLINK, is to allocate for each person a matrix of size $N(N+1)/2$ to represent all the possible multilocus genotypes (where N is the product of the number of alleles in all loci), and to keep track of valid genotypes during the likelihood calculations. Another approach, used by VITESSE, is to store single-locus genotype lists and build valid multilocus genotypes when needed. A third approach, implemented in SUPERLINK, is to store separate single-locus allele lists for the two haplotypes, one list for the maternal haplotype and one for the paternal haplotype, and to assemble valid single-locus and multilocus genotypes when necessary. This representation follows from the choice of variables in the Bayesian network.

These three approaches have very different memory requirements. For example, five five-allelic loci would require storage of 4884375 multilocus genotypes (N = 3125), or 75 single-locus genotypes (15 for each locus) and only at most 50 single-locus allele entries (5 entries per haplotype list per locus). We use the words 'at most' since SUPERLINK stores only valid alleles for each single-locus-haplotype (given the data), therefore reducing significantly the size of the single-locus allele lists and the time needed for the calculations.

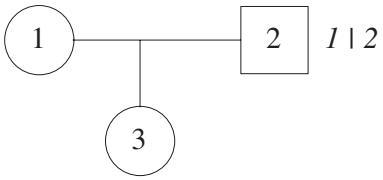

Fig. 2. An example for a possible *downward update*.

Value and allele exclusion

SUPERLINK performs a preprocessing phase that reduces the range of values that are valid for each variable of the Bayesian network given the data. This phase yields major savings in time and memory requirements of the likelihood calculations.

This phase is divided into two steps. The first step is performed directly on the graph representation of the pedigree, before transforming it into a Bayesian network, whereas the second step is performed on the local probability tables that annotate the nodes of the constructed Bayesian network.

The first step is performed on the graph representation of the pedigree. The nodes of this graph represent the people in the pedigree and the edges represent parental relations. This step is based on the well-known observation that the possible genotypes of an individual can be inferred from the genotypes of one's relatives (e.g., Lange and Goradia, 1987).

For example, if we know that some individual (a male) has the genotype *1 | 2* in some locus then his child can only have allele 1 or 2 in the paternal haplotype of this locus. The family in Figure 2 is drawn according to the convention that females are represented by circles and males are represented by squares in pedigree sketches. In this case a *downward update* is performed, the child is updated according to the parent.

Another possible update is an *upward update*, in which the parent is updated according to the children. For example, let us observe the family in Figure 3. The mother (2) is an homozygote for allele 1. Hence, both children (3 and 4) got allele 1 from their mother. Therefore, the father (1) must have transmitted allele 3 to his daughter (3) and allele 4 to his son (4), hence his genotype is *3 | 4*. These updates work in a *local* manner, by examining invalid parents-children joint assignments. However, each update is then propagated through the pedigree graph and therefore results in a *global* update.

When some information on the genotypes of an individual's grandparents is available, it sometimes becomes possible to rule out one of the two values of the relevant selector variable of the individual. In such a case, the variable reduces to a constant. We refer to such an update as a *selector update*.

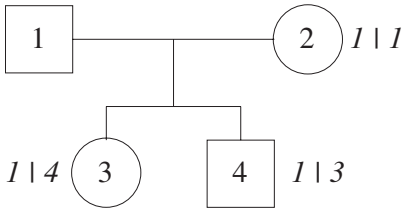

Fig. 3. An example for a possible *upward update*.

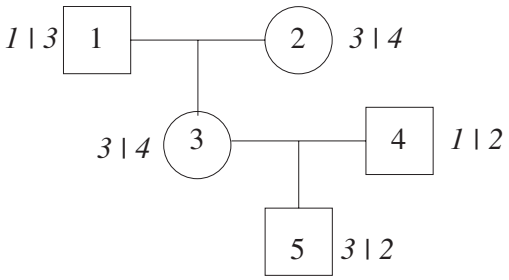

Fig. 4. An example for a possible *selector update*.

An example for a selector update is shown in Figure 4. Here, the algorithm infers that individual 5 received allele 3 from his mother (3). It also infers that this allele resided in the paternal haplotype of individual 3, since only her mother (2) could have transmitted allele 4 to her and this leaves allele 3 to have been transmitted from the father (1). From this the algorithm rules out the value 1 of the maternal selector variable of person 5 for this locus.

In the second step of value elimination the algorithm uses the fact that if all entries of some probability table that correspond to a specific value of one of its variables equal zero, then this value of the variable is invalid. For example, if $\Pr(x, y, z) = 0$ for all values y, z of Y, Z then the value x is not valid for the variable X.

This preprocessing phase often has a large impact on the computation time: instead of summing over all possible values for each variable, we only sum over the valid values and thus attain a reduction in the number of operations. The size of the tables that need to be stored is also reduced significantly. This reduction is realized due to the compact and flexible representation of probability tables used by our algorithm.

Variable trimming

Variables that correspond to leaves in the Bayesian network for which no data exists (i.e., all their values are valid) can be trimmed without altering the likelihood computation. After trimming such variables, other variables become leaves and they could potentially be trimmed. The algorithm continues in this fashion until no further nodes can be trimmed.

An example for a possible trimming is when the affection status of an individual is unknown. In such a case, the relevant phenotype variable can be trimmed. Other variables that can be trimmed automatically, or, equivalently, not be included in the network, are selector variables of founders. These selector variables are omitted due to lack of information about founders' phase in a pedigree.

This trimming process speeds up the calculations and lowers storage requirements. It often results in a substantial reduction in the number of variables that SUPERLINK needs to sum over.

Merging variables

A pedigree contains no information about founder phase and consequently two genotypes which differ only by phase will have the same probability and can therefore be united when performing the calculations. Therefore, we can unite the two genetic-loci variables that describe the genotype of a founder for a specific loci into one variable (as suggested in Kruglyak *et al.*, 1996). The possible values for this united variable are all the valid value combinations of the two original variables.

Due to the fact that we cannot determine the phase in founders' genotypes, we also cannot identify recombination events in their children. Therefore, the selector variables of their children are redundant. We simply calculate the probability that the child would have a certain genotype given the genotype of his parent, without any consideration of recombination events.

This step reduces both time and memory requirements of the computations.

Time-space tradeoff

As mentioned before, there are two extreme approaches to computing the likelihood of a Bayesian network exactly, conditioning and variable elimination. If the data being analysed is simple enough, then it might be processed using variable elimination alone. Unfortunately, this is often not the case and memory limitations might be exceeded this way. SUPERLINK combines the two methods to achieve the best time-space tradeoff given the memory available for the linkage analysis problem. This approach is discussed in (Dechter, 1996). The crucial point of the algorithm is that conditioning is performed only after some steps of variable elimination, when the memory requirements are about to exceed the limitations. Such conditioning often applies only to parts of the Bayesian network. Thus, computations in other unrelated parts of the network are not repeated unnecessarily. The selection of variables to condition on often simplifies the Bayesian network sufficiently for processing with the available memory.

Variable elimination order

The order of variable elimination is critical. It greatly affects both time and memory requirements. The ordering is determined automatically according to the parameters of the specific linkage problem: the size of the pedigree, the number of loci, and the number of valid alleles at each locus. For small pedigrees with a large number of markers, the algorithm chooses a peeling order, based on the Lander–Green approach, that proceeds locus after locus. For large pedigrees with a few markers, the algorithm chooses an Elston–Stewart style elimination which peels one nuclear family at a time. Other linkage problems are handled by finding a good elimination order. Often the program chooses an elimination order that is a combination of these two extreme choices of ordering.

If the input of SUPERLINK is a small pedigree and many loci, the elimination is performed locus by locus, starting from one end of the genetic map and working in linear order towards the other end. That is, first all the phenotype and genetic loci variables that represent the locus on one end of the map are eliminated. Then all the selector variables that relate to this locus are eliminated. Then we continue to the next locus on the map, and so on. This heuristic defines groups of variables and an order in which to eliminate these groups. A greedy heuristic is used to determine the elimination order of the variables in each group.

If the input is an arbitrary sized pedigree and an arbitrary number of loci, SUPERLINK uses a greedy heuristic to determine the elimination order. The greedy heuristic being used assigns each variable an elimination cost and chooses to eliminate the variable with the smallest cost. The costs of the variables are updated dynamically, whenever necessary. The *elimination cost* of variable v is denoted by $EC(v)$ and is computed as follows:

- If the elimination of variable v would result in a function whose variables are already contained in an existing function, the cost of the variable is zero. By eliminating such a variable, we only reduce the memory requirements and therefore it is desirable to give such a variable the lowest possible cost.
- Otherwise, the cost of variable v is the size of the probability table of the function that would result from eliminating it. More formally, if $F(v)$ is the set of functions that use variable v, then the size of the probability table that would result from eliminating it is:

$$EC(v) = \prod_{X \in N(v)} \mid Val(X) \mid,$$

where $N(v) = \left(\bigcup_{f \in F(v)} Arg(f)\right) - \{v\}$ and $Arg(f)$ is the set of variables over which the function f is defined.

The variable chosen to be eliminated is:

$$\arg\min_{v} EC(v).$$

An intuitive explanation of this choice of heuristic is that table size constitutes a good measure for the complexity of eliminating a certain variable.

Finding the above minimum can be time consuming. Therefore, if the table that would result from eliminating a particular variable is below a certain threshold, then the search for the minimum is discontinued and this variable is eliminated.

Order for conditioning

As stated above, SUPERLINK combines variable elimination with conditioning. In the previous section we discussed how the variables to be eliminated are chosen. However, if the elimination of the chosen variable (which is the best variable found for elimination) would result in a function of size greater than a certain threshold, then the memory limitations might be exceeded. In such a case, a variable to condition on is selected. A greedy heuristic is used to choose a variable which fulfils the following condition:

$$\arg\max_{v} n(v)EC(v),$$

where $EC(v)$ is the elimination cost of v and $n(v)$ is the number of functions that use v.

Representation of tables

Our algorithm defines all probability tables in a flexible size that depends on valid values for each variable. Each probability table, referred herein simply as a function, is represented by a one-dimensional array of double-precision numbers. In addition we store, the size of the probability table, the number of variables in the table and an array of pointers to the variables. For each variable, the number of valid values for it is also stored. The size of a probability table is the product of the number of valid values of each of its variables. We use a special indexing method that allows to quickly calculate the index of the array that corresponds to a certain set of values of the function's variables and also to determine the values of the function's variables that correspond to a certain entry of the array.

Each function also has a global scaling factor s. For an entry $T(v_1, v_2, \ldots, v_n)$ of a probability table with the value $p_T(v_1, v_2, \ldots, v_n)$, the value of the function corresponding to this entry is expressed via:

$$f(v_1, v_2, \ldots, v_n) = p_T(v_1, v_2, \ldots, v_n) \exp(s).$$

This method of storing tables is required in order to avoid underflow from occurring during the computations and to maintain accuracy. These problems arise because intermediate results are often very small numbers.

FEATURES OF SUPERLINK

SUPERLINK allows for analysis of autosomal as well as sex-linked traits and also allows for analysis with two bi-allelic disease loci. The description as a Bayesian network and the automatic optimization of computations yields a solid ground for efficient extensions.

Sex-linked traits

SUPERLINK allows for analysis of sex-linked traits, i.e., traits that are controlled by loci that reside on the X-chromosome. Such traits exhibit a different inheritance pattern from autosomal traits. Males have one X chromosome and one Y chromosome, while females have two X chromosomes. Therefore, in the case of sex-linked traits, recombination events can only be observed in the transmission from the mother, whereas in the case of autosomal traits they can be observed in the transmission from both parents.

Two-locus traits

SUPERLINK allows for a disease phenotype to be under the control of two loci. Each of the disease loci has two alleles: the disease allele and the normal allele. Thus, there are 3 possible genotypes per locus and 9 possible joint genotypes for the two loci. For each of these joint genotypes the susceptibility of being affected has to be provided.

This feature is not supported in FASTLINK and VITESSE. The LINKAGE programs, which are the origin of FASTLINK, have an extension, TLINKAGE (Lathrop and Ott, 1990; Risch, 1990; Schork *et al.*, 1993) which is slower than SUPERLINK. GENEHUNTER also has an extension that allows for analysis of two-locus traits, GENEHUNTER-TWOLOCUS (Strauch *et al.*, 2000). Its shortcoming is that it is not suitable for analysing pedigrees of large size. Our program is currently the only program suitable for analysis of two-locus traits, autosomal or sex-linked, in fully general pedigrees.

Extensibility

Bayesian networks provide us with a convenient language to describe and encode modelling assumptions about pedigrees. This language allows us to represent a wide range of possible alternatives that can arise in linkage analysis. So far, we implemented in SUPERLINK the option to analyse sex-linked traits and two-locus disease. SUPERLINK can be extended to efficiently handle multilocus disease models, to add environmental factors that affect disease onset, and to model chiasmata interference in the context of general pedigrees.

EXPERIMENTAL RESULTS

We have run several experiments to compare our program, SUPERLINK V1.0, to some of the leading linkage programs, FASTLINK V4.1, GENEHUNTER V2.1 and VITESSE V1.0. The running environment on which all experiments were performed was a Sun OS version 5.7 (sun4u) with 2624 MB RAM.

Experiment A. In the first experiment (Table 1) we used 12 data sets with a medium-sized topology, elicited for a coronary heart disease study (taken from Linkage User's guide) and artificially increasing complexity in terms of the number of loci being analysed. This pedigree contains no loops. As can be seen, the pedigree size exceeds the size that can be handled by GENEHUNTER and only the first few data sets can be run by FASTLINK and VITESSE before memory requirements are exceeded. SUPERLINK runs on all the data sets except for the last one on which the computation will require over a 100 hours in order to complete. Note also that for the data sets that run on FASTLINK and VITESSE, the running times of SUPERLINK are smaller.

Experiment B. In the second experiment (Table 2), we used 12 artificial data sets with increasing complexity in terms of the number of loci being analysed. In this experiment, the topology used included loops. VITESSE does not handle looped pedigrees and therefore cannot run on these data sets. GENEHUNTER fails due to the size of the pedigree, and FASTLINK cannot handle most data sets due to memory requirements. Our program can run on all the data sets.

Experiment C. In the third experiment (Table 3), we used 12 data sets with different topologies, different number of loci and different pedigree sizes. Some of the topologies contain loops and some do not. GENEHUNTER is currently faster on small pedigrees if and only if multiple likelihood computations (say, one between each pair of markers) are requested by the user. In the discussion we explain how SUPERLINK will be made faster also for this computation.

Experiment D. At the time of writing this paper we were unable to run VITESSE V2.0 (O'Connell, 2001). However, we ran data sets 1 and 5 in (O'Connell, 2001) and the results are as follows. On data set 1, our algorithm runs 4000 times faster than VITESSE V1.0 while VITESSE V2.0 runs 1800 times faster. Note that this example was constructed to show the advantage of VITESSE V2.0 over its previous version while we used SUPERLINK without any adjustments. Data set 5 runs over 3600 times faster on VITESSE V1.0 compared to VITESSE V2.0. With SUPERLINK, it runs three time faster than VITESSE V1.0, again with no adjustments. VITESSE V2.0 does not accept general pedigrees and is not designed to handle more efficiently the data sets of experiments A, B and C.

Table 1. Experiment A: The table shows run times (in seconds) of SUPERLINK, FASTLINK, VITESSE and GENEHUNTER for various data sets

Data Sets	♯People	♯Loci	♯Alleles	Loops	Superlink	Fastlink	Vitesse	Genehunter
EA0	57	2	4-5	NO	0.03	0.12	0.27	****
EA1	57	5	4-5	NO	0.1	3.77	0.31	****
EA2	57	6	4-5	NO	0.14	79.32	0.39	****
EA3	57	7	4-5	NO	0.42	*	0.69	****
EA4	57	8	4-5	NO	0.36	*	2.81	****
EA5	57	10	4-5	NO	1.19	*	84.66	****
EA6	57	12	4-5	NO	4.65	*	*	****
EA7	57	14	4-5	NO	3.01	*	*	****
EA8	57	18	4-5	NO	20.98	*	*	****
EA9	57	37	4-5	NO	8510.15	*	*	****
EA10	57	38	4-5	NO	10446.27	*	*	****
EA11	57	40	4-5	NO	*****	*	*	****

Table 2. Experiment B: Run times for additional data sets

Data Sets	♯People	♯Loci	♯Alleles	Loops	Superlink	Fastlink	Vitesse	Genehunter
EB0	100	5	5-10	YES	2.56	3933.7	***	****
EB1	100	6	5-10	YES	2.63	*	***	****
EB2	100	10	5-10	YES	82.56	*	***	****
EB3	100	12	5-10	YES	437.55	*	***	****
EB4	100	13	5-10	YES	17.29	*	***	****
EB5	100	14	5-10	YES	278.8	*	***	****
EB6	100	15	5-10	YES	935.86	*	***	****
EB7	100	16	5-10	YES	902.8	*	***	****
EB8	100	17	5-10	YES	288.2	*	***	****
EB9	100	18	5-10	YES	113.96	*	***	****
EB10	100	19	5-10	YES	2901.25	*	***	****
EB11	100	20	5-10	YES	143640.22	*	***	****

DISCUSSION

The model of Bayesian networks, with the variables defined as described herein (due to Friedman *et al.*, 2000), enabled us to represent linkage problems in sufficiently fine detail to allow efficient exact likelihood computations for more complex pedigrees than was previously possible. One reason for outperforming previous software is that previous software was restricted to specific Bayesian networks (implicitly), and consequently was unable to utilize an optimal, or close to optimal, order of computation. For example, FASTLINK represents multilocus genotypes as one variable which is equivalent to summing variables person by person, VITESSE represents genotypes by pairs at each locus, rather than locus by locus, and GENEHUNTER represents inheritance vectors as a single variable with exponential number of values, rather than by individual selector variables, each with two values. Such choices limit the performance. These choices should all be made automatically by the linkage software on a case-by-case sufficiently fast input analysis.

There are several easy ways to further improve speed which will be incorporated in the near future. When asked to compute the likelihood of data assuming the disease locus can lie between any two consecutive markers, it is often the case, for small pedigrees, that using the order locus by locus, keeping some intermediate results, and multiplying special probability tables more efficiently, as done by GENEHUNTER, is the most efficient computation route. These speedups are natural to incorporate into SUPERLINK, and due to the selector variables that we use, which replace inheritance vectors of GENEHUNTER, the incorporation of these ideas will make SUPERLINK also outperform GENEHUNTER for repeated likelihood computations on small pedigrees, which is the criterion that GENEHUNTER is designed to meet best.

ACKNOWLEDGEMENTS

We thank Alejandro Schäffer from NIH for several years of continued help and support. We thank Nir Friedman from the Hebrew University for many fruitful discussions and for writing with the second author a research proposal to support this work. This research was supported by the Israel Science Foundation.

Table 3. Experiment C: Run times for additional data sets

Data Sets	♯People	♯Loci	♯Alleles	Loops	Superlink	Fastlink	Vitesse	Genehunter
EC0	100	6	2	YES	27.62	382.88	***	****
EC1	100	7	2	NO	2.43	0.82	0.40	****
EC2	100	8	2	YES	0.56	12.59	***	****
EC3	100	10	2	NO	16.36	**	22.79	****
EC4	100	15	5-10	NO	1.00	*	*	****
EC5	20	15	2	NO	44.1	*	*	****
EC6	15	20	2	NO	35.32	*	*	****
EC7	15	22	2	NO	102.56	*	*	****
EC8	150	8	5-7	YES	0.87	*	***	****
EC9	150	10	5-7	YES	1.28	*	***	****
EC10	5	100	3-6	NO	0.06	**	*	0.41 (+)
EC11	5	110	3-6	NO	0.08	**	*	0.45 (+)

Table 4. Definitions of symbols used in the tables:

Symbol	Meaning
*	Out-of-Memory
*	Segmentation Fault or Bus Error
**	Not Applicable - VITTESE does not handle looped pedigrees.
***	Not Applicable - GENEHUNTER does not handle large pedigrees.
****	Over 100 hours.
+	In GENEHUNTER, the lod-score is calculated several times, for different positions of the disease gene on the marker map. In the runs above, the number of lod-score calculations equals the number of markers in the analysis (i.e, 99 for data sets EC10 and 109 for data sets EC11). Hence, GENEHUNTER is currently faster for multiple likelihood calculations.

REFERENCES

Arnborg,S. (1985) Efficient algorithms for combinatorial problems on graphs with bounded decomposibility. *BIT*, **25**, 2–23.

Arnborg,S., Corneil,D.G. and Proskurowski,A. (1987) Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, **8**, 277–284.

Becker,A., Geiger,D. and Schäffer,A.A. (1998) Automatic selection of loop breakers for genetic linkage analysis. *Hum. Hered.*, **48**, 49–60.

Cottingham,Jr,R.W., Idury,R.M. and Schäffer,A.A. (1993) Faster sequential genetic linkage computations. *Am. J. Hum. Genet.*, **53**, 252–263.

Dechter,R. (1996) Topological parameters for time-space tradeoff. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI)*. pp. 220–227.

Dechter,R. (1998) Bucket elimination: a unifying framework for probabilistic inference. In Jordan,M.I. (ed.), *Learning in Graphical Models*. Kluwer Academic Press, pp. 75–104.

Elston,R.C. and Stewart,J. (1971) A general model for the analysis of pedigree data. *Hum. Hered.*, **21**, 523–542.

Friedman,N., Geiger,D. and Lotner,N. (2000) Likelihood computation with value abstraction. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*.

Kruglyak,L., Daly,M.J. and Lander,E.S. (1995) Rapid multipoint linkage analysis of reccessive traits in nuclear families including homozygosity mapping. *Am. J. Hum. Genet.*, **56**, 519–527.

Kruglyak,L., Daly,M.J., Reeve-Daly,M.P. and Lander,E.S. (1996) Parametric and nonparametric linkage analysis: a unified multipoint approach. *Am. J. Hum. Genet.*, **58**, 1347–1363.

Lander,E.S. and Green,P. (1987) Construction of multilocus genetic maps in humans. *Proc. Natl Acad. Sci. USA*, **84**, 2363–2367.

Lange,K. (1997) *Mathematical and Statistical Methods for Genetic Analysis*. Springer, New York.

Lange,K. and Goradia,T.M. (1987) An algorithm for automatic genotype elimination. *Am. J. Hum. Genet.*, **40**, 250–256.

Lathrop,G.M. and Ott,J. (1990) Analysis of complex diseases under oligogenic models and intrafamilial heterogeneity by the linkage programs. *Am. J. Hum. Genet.*, **47**, A188.

Lauritzen,S.L. (1996) *Graphical Models*. Oxford University Press.

O'Connell,J.R. (2001) Rapid multipoint linkage analysis via inheritance vectors in the elston-stewart algorithm. *Hum. Hered.*, **51**, 226–240.

O'connell,J.R. and Weeks,D.E. (1995) The vittesse algorithm for rapid exact multilocus linkage analysis via genotype set-recoding and fuzzy inheritance. *Nature Genet.*, **11**, 402–408.

Pearl,J. (1988) *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA.

Risch,N. (1990) Linkage strategies for genetically complex traits. i. multilocus models. *Am. J. Hum. Genet.*, **46**, 222–228.

Schäffer,A.A., Gupta,S.K., Shriram,K. and Cottingham,Jr,R.W. (1994) Avoiding recomputation in linkage analysis. *Hum. Hered.*, **44**, 225–237.

Schork,N.J., Boehnke,M., Terwilliger,J.D. and Ott,J. (1993) Two

trait locus linkage analysis: a powerful strategy for mapping complex genetic traits. *Am. J. Hum. Genet.*, **53**, 1127–1136.

Shachter,R.D., Andersen,S.K. and Szolovits,P. (1994) Global conditioning for probabilistic inference in belief networks. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI)*. pp. 514–522.

Strauch,K., Fimmers,R., Kurz,T., Deichmann,K.A., Wienker,T.F. and Baur,M.P. (2000) Parametric and nonparametric multipoint linkage analysis with imprinting and two-locus-trait models: application to mite sensitization. *Am. J. Hum. Genet.*, **66**, 1945–1957.

Terwilliger,J.D. and Ott,J. (1994) *Handbook of Human Genetic Linkage*. Johns Hopkins University Press, Baltimore, Maryland.