Automatic Selection of Loop Breakers for Genetic Linkage Analysis

Ann Becker [*]
Computer Science Department
Technion
Haifa
ISRAEL

Dan Geiger [†]
Computer Science Department
Technion
Haifa
ISRAEL

Alejandro A. Schäffer [‡]
National Human Genome Research Institute
National Institutes of Health
Bethesda and Baltimore
U.S.A.

October 9, 1997

Keywords: Genetics, linkage analysis, algorithms, loops, probabilistic inference, Bayesian networks.
Running Head: Loop breakers.

Address for correspondence:
Alejandro A. Schäffer
NIH/NHGRI Suite 2000
333 Cassell Drive
Baltimore MD 21224
U.S.A.
FAX: (410) 550-7513

[*] anyuta@cs.technion.ac.il
[†] dang@cs.technion.ac.il
[‡] schaffer@helix.nih.gov

Abstract

Pedigree loops pose a difficult computational challenge in genetic linkage analysis. The most popular linkage analysis package, LINKAGE, uses an algorithm that converts a looped pedigree into a loopless pedigree, which is traversed many times. The conversion is controlled by user-selection of individuals to act as loop breakers. The selection of loop breakers has significant impact on the running time of the subsequent linkage analysis. We have automated the process of selecting loop breakers, implemented a hybrid algorithm for it in the FASTLINK version of LINKAGE, and tested it on many real pedigrees with excellent performance. We point out that there is no need to break each loop by a distinct individual because, with minor modification to the algorithms in LINKAGE/FASTLINK, a single individual that participates in multiple marriages can serve as a loop breaker for several loops. Our algorithm for finding loop breakers, called LoopBreaker, is a combination of: (1) a new algorithm that is guaranteed to be optimal on the special case of pedigrees with no multiple marriages and (2) an adaptation of a known algorithm for breaking loops in general graphs. The contribution of this work is the adaptation of abstract methods from computer science to a challenging problem in genetics.

1 Introduction

Pedigree loops pose a difficult computational challenge in genetic linkage analysis. In this paper we address a combinatorial optimization problem, which we call the *loop breaker selection* (LBS) problem that arises for looped pedigrees in the most popular genetic linkage analysis software package, LINKAGE [1, 2, 3]. We implemented our solution in FASTLINK [4, 5], a faster version of LINKAGE. We illustrate with real examples that our solution to the LBS problem translates to faster computation times.

Our algorithm for finding loop breakers is a combination of: (1) a new algorithm that is guaranteed to be optimal on the special case of pedigrees with no multiple marriages and (2) an adaptation of a known algorithm for breaking loops in general graphs. We also point out that there is no need to break each loop by a distinct individual because, with minor modification to the algorithms in LINKAGE/FASTLINK, a single individual that participates in multiple marriages can serve as a loop breaker for several loops. This idea is adapted from artificial intelligence [6].

Kong [7] pointed out in 1991 that the evaluation of the basic Elston-Stewart maximum likelihood linkage analysis formula is a special case of a problem known in statistics and artificial intelligence as "probabilistic inference in Bayesian networks". Kong applied mathematical and software tools developed for the probabilistic inference problem to derive approximate likelihood algorithms for pedigrees where the Elston-Stewart [8, 9] method is computationally infeasible. We use the relationship between linkage analysis and probabilistic inference to improve an existing implementation of the Elston-Stewart method.

Our work transcends three broad areas: combinatorial optimization, probabilistic inference, and genetics. The next two sections present the needed background information to enable the presentation of our results in proper detail. The remainder of the paper consists of a short methods section describing our implementation, a section describing our new algorithm, a results section describing our the performance of our algorithm on real data, and a discussion section.

2 Background

The essential input to a genetic linkage analysis computation is a *pedigree*. To illustrate some definitions, we will use a pedigree containing a first-cousin marriage with 3 offspring (Figure 1). The pedigree in Figure 1 is drawn according to the conventions advocated by the Pedigree Standardization Task Force [10]. For algorithmic purposes we prefer the *marriage graph* representation of pedigrees promoted by Lange and Elston [9] and Cannings, Thompson, and Skolnick [11]. A marriage graph drawing of the first-cousin-marriage pedigree is shown in Figure 2.

A marriage graph is an example of a *graph*. A *graph* consists of two sets, *vertices* and *edges*. A vertex can represent any object; in our usage vertices represent individuals and marriages. An edge is a pair v, w of vertices. In *undirected* graphs the edge pairs are unordered and an edge is drawn v — w. In *directed* graphs the edge pairs are ordered and an edge from v to w is drawn $v \rightarrow w$. Two vertices are *neighbors* if there exists an edge between them.

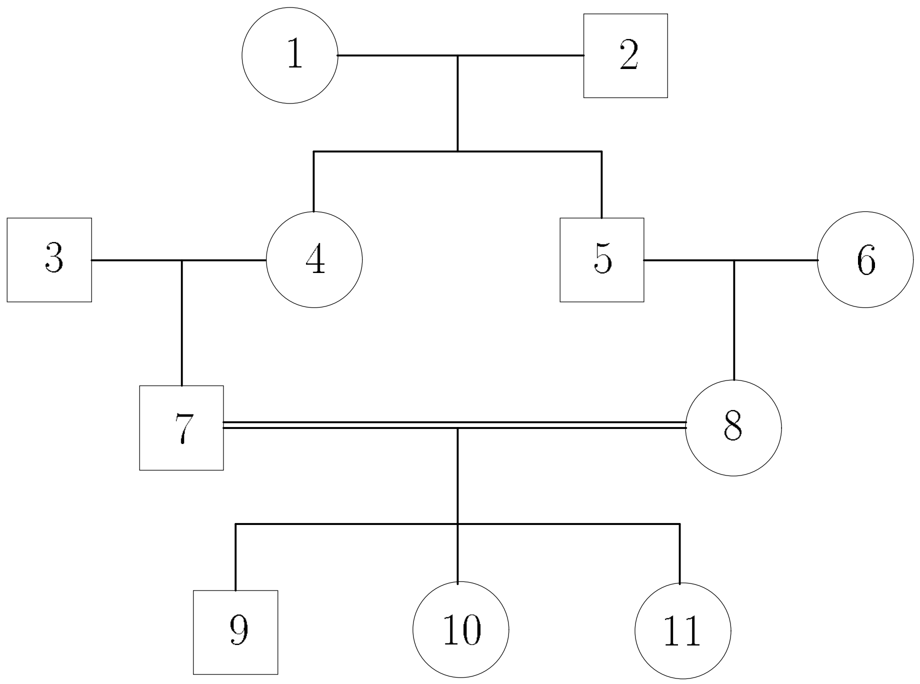

Figure 1: A first-cousin marriage with three offspring

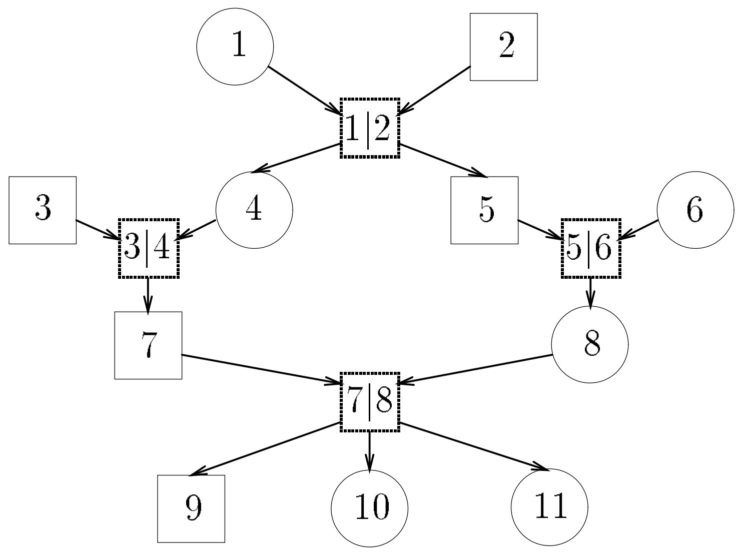

Figure 2: Marriage graph for first cousin marriage pedigree

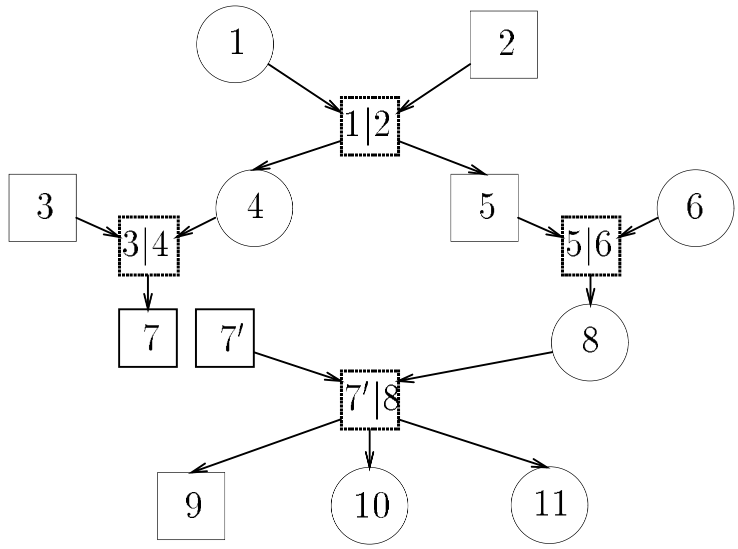

Figure 3: First-cousin marriage with loop broken by cloning

We determine a marriage graph from a pedigree as follows. The vertex set consists of two disjoint sets I and M where I is the set of individuals and M is the set of marriages. The edge set consists of two types of edges; An edge $m \rightarrow i$ from each marriage vertex m to each individual i who is an offspring of m and an edge $j \rightarrow m$ whenever individual j is one of the spouses in marriage m (See Figure 2). We use $x|y$ to denote the marriage vertex of the individuals x and y. The vertical bar in this notation separates the two participants in a marriage vertex as it is used to separate phase in phase-known genotypes.

A *loop* in a pedigree occurs precisely when the marriage graph has a *cycle*, if one ignores the directions on the edges. A *cycle* in an undirected graph is a sequence of vertices $v_1, \ldots v_k$ such there are no duplicate vertices, except $v_1 = v_k$ and each pair of consecutive vertices v_i, v_{i+1} has an edge v_i — v_{i+1} between them. For the loop in the pedigree of Figure 1, the corresponding cycle in Figure 2 is: $1|2, 4, 3|4, 7, 7|8, 8, 5|6, 5, 1|2$, where we ignore the directions on the edges. Cannings, Thompson, and Skolnick [11] distinguished marriage loops and inbreeding loops, but this distinction is neither important for our purposes, nor clear when a pedigree has multiple overlapping loops.

A *path* in a graph is a sequence of distinct vertices $v_1, \ldots v_k$ such that each pair of consecutive vertices v_i, v_{i+1} has an edge v_i — v_{i+1} between them. A *directed path* in a directed graph, is a path in which the edges all point in the forward direction. A graph is *connected* if there is a path between every two vertices. All marriage graphs are connected. A connected, undirected graph with no cycles is called a *tree*.

The essential goal of linkage analysis is to estimate the recombination fraction, θ, based on genotype (and phenotype) information about the pedigree. LINKAGE and other linkage analysis software packages use maximum likelihood methods, so they must compute the pedigree likelihood for different candidate values of θ. To do the computation in LINKAGE, one individual r is chosen as the *root*. Let $P \setminus r$ denote the set of all individuals in the pedigree except r. The program computes: $L(P|\theta) = \Sigma_g Prob(\text{r has genotype } g \mid \theta, P \setminus r)$. In words, the likelihood of the pedigree conditional on θ is the sum over all genotypes g, of the conditional probability that r has genotype g conditioned on θ and the information about the other individuals in the pedigree. Elston and Stewart [8] were the first to observe that in loopless pedigrees with one pair of founders this computation can be done efficiently by traversing the pedigree from bottom to top. Ott [12] extended the Elston-Stewart algorithm to all loopless pedigrees. Lange and Elston [9] further extended the Elston-Stewart algorithm to looped pedigrees by defining an operation we call *cloning*. Intuitively cloning means making two copies of an individual and forcing the two copies to have the same genotype. In terms of the marriage graph, an individual vertex i may be cloned if i has an incoming edge (i's parents are in the pedigree) and i has an outgoing edge (i has a child in the pedigree). To clone i, add an individual vertex i' and replace each edge of the form $i \rightarrow m$ with an edge $i' \rightarrow m$. Marriage vertices cannot be cloned. Figure 3, shows the marriage graph in Figure 2, after person 7 has been cloned. Notice that by cloning individual 7, the cycle in the marriage graph (and hence the loop in the pedigree) is broken. For this reason we call the cloned individuals *loop breakers*.

Lange and Elston pointed out that after cloning individuals the following algorithm can be used to compute the likelihood.

Input: *A set of loop breakers* $\{b_1, \ldots, b_t\}$, *a set of loop breakers clones* $\{b'_1, \ldots, b'_t\}$.
a set of possible genotypes of loop breakers $\{G_1, \ldots, G_t\}$.
Output: *The likelihood*
$like \leftarrow 0$
For each vector $[g_1, \ldots, g_t] \in G_1 \times \cdots \times G_t$
1. Compute the likelihood of the loopless pedigree conditional on b_i and b'_i having the same genotype g_i, for $i = 1, \ldots t$.
2. Sum the conditional likelihood into *like*.

return *like*

Let $G = G_1 \times \cdots \times G_t$. Linkage analysis computations can be slow when the Cartesian product set G of possible loop breaker genotypes is large. Let $|G_i|$ and $|G|$ denote the sizes of G_i and G respectively. The *loop breaker selection* (LBS) problem is to find a set of individuals to clone so as to minimize $|G| = \prod_i |G_i|$, or equivalently, to minimize $\log |G| = \Sigma_i \log |G_i|$.

The loop breaker selection problem applies to implementations of the Elston-Stewart algorithm in all versions of LINKAGE/FASTLINK. Other algorithms for traversal of looped pedigrees, also called *peeling*, were proposed and implemented by Cannings, Thompson, and Skolnick [11], and by Lange and Boehnke [13]. A related algorithm for peeling a looped graph was considered in a more abstract, non-genetic setting by Lauritzen and Spiegelhalter [14]. That setting is described in the next section. Subsequent papers [15, 16, 17] develop optimization algorithms for finding a good peeling order for the Lauritzen-Spiegelhalter method. These algorithms have not yet been applied to genetic linkage analysis.

Users of LINKAGE/FASTLINK must currently solve instances of the LBS problem manually or with the assistance of a preprocessor program called LOOPS [18] which identifies loops in a given pedigree. LOOPS analyzes a pedigree and reports a set of loops (if any) that it contains. According to [19], pp. 93–96, the suggested usage of LOOPS is:

1. Run LOOPS on the current pedigree.
2. If LOOPS reported at least one loop, choose an individual p in one loop to clone.
3. If p is selected in step 2, clone p and all previously selected loop breakers.

These steps are repeated until LOOPS does not find any more loops. The procedure is not difficult to use to get a valid loop breaker set. Using this procedure guarantees a solution to LBS that is *minimal* in the sense that no loop breakers can be omitted. The procedure does not guarantee a solution to LBS that is minimum in the number of loop breakers. Furthermore, there is no explicit guidance on how to choose the loop and the loop breaker p at step 2, so that $|G|$ is minimized.

3 Background from other fields

The statistics and artificial intelligence literature (e.g., [14, 20]), treat the following problem in which loop breaker selection arises as a subproblem. The input is a directed graph with no directed cycles such that each vertex v corresponds to a random variable x_v and such that

$$P(x_1, \ldots, x_n) = \prod_v P(x_v | X_{A(v)})$$

where $X_{A(v)}$ are the random variables corresponding to the vertices $\{a : a \rightarrow v \text{ is an edge}\}$. The output is the probability $P(Y = y|Z = z)$ for any given disjoint subsets of variables $Y, Z \subset \{x_1, \ldots, x_n\}$. This problem is called the *inference problem*; it includes evaluating the pedigree likelihood in linkage analysis as a special case. The inference problem is more general since the random variables can be discrete, Gaussian, combination thereof, or other types, and there could be any structure to the graph as long as it has no directed cycles. The graph together with the probability distribution is called a *Bayesian network*. The inference problem has application in domains where a probability distribution serves as a model, such as economics, sociology, artificial intelligence, error correcting codes, and other fields.

Kim and Pearl [21] developed a method to solve the inference problem for Bayesian networks without loops and Pearl [22], unaware of [8, 9], extended it to Bayesian networks with loops (see also [6]). Pearl's method includes a multi-copy extension of the Lange-Elston cloning operation. Suppose that individual i to be cloned has outgoing edges $i \rightarrow m_1, i \rightarrow m_2, \ldots, i \rightarrow m_k$. Then it is possible to clone i with k new copies $m_1, \ldots m_k$. Making more than 2 copies does not change the applicability of the Lange-Elston algorithm that iterates over G nor does it change the size of G. The generalized loop breaker selection problem is to find a set of individuals to clone and the number of copies of each so as to minimize the product $\prod_i |G_i|$ or equivalently the sum $\sum_i \log |G_i|$.

Pearl's cloning has a very practical and specific application for marriage graphs. When the individual i to be cloned participates in *multiple marriages* one can make multiple clone copies of i, one per marriage. It is clear that this was not observed by the developers of LINKAGE because the constant 2 is hard-coded (i.e., not represented symbolically) for the number of possible copies of an individual in many places in the software. We have modified FASTLINK to allow multi-copy cloning, and we illustrate in the results section how beneficial it is for speed.

In Bayesian networks each vertex v can be selected as a loop breaker but if v is selected it does not break loops formed by traversing a path $a \rightarrow v \leftarrow b$, where both edges point into v (because once the value of v is known, a and b become dependent and a virtual edge connects them). The theory explaining this constraint is developed in [6] and it is based on a graph-theoretic criterion called d-separation that fully characterizes which vertices are conditionally independent of others [23]. This criterion is used to justify any solution to the inference problem because any solution must use the properties of conditional independence in order to be efficient. For marriage graphs the selection of loop breakers is conceptually easier because marriage vertices cannot be selected and therefore, the $a \rightarrow v \leftarrow b$ condition does not constrain the loop breaker selection. Consequently, in marriage graphs the direction of the edges can be ignored for the purpose of finding loop breakers (but not for the purpose of likelihood computations).

For Bayesian networks, Suermont and Cooper [24] showed that the LBS problem belongs to a class of intractable problems that computer scientists call NP-complete. This means that it is highly unlikely that there is an algorithm to solve the problem optimally whose computation time grows as a polynomial in the number of vertices. A similar result holds for undirected graphs as well. For undirected graphs finding a set of vertices whose removal (or, equivalently, cloning) creates a graph with no cycles is known in combinatorial optimization as the "*feedback vertex set* problem," and it was one of the first problems shown to be NP-complete in the seminal work by Karp [25].

Based on Karp's NP-completeness result, computer scientists sought an algorithm that has a computation time polynomial in the size of the input graph H and adheres to the following property:

> If $\hat{G}$ is the minimum-size set of genotype vectors for loop breakers of graph H and G is set of genotype vectors found by the loop breaker selection algorithm, then one is guaranteed that $\log |G| \leq c \times \log |\hat{G}|$, where $c > 1$ is some error constant.

Such an algorithm is called a *constant approximation algorithm*. The first constant approximation algorithm for loop breaker selection is given in [26] (which appeared in a conference version in early 1994). The error constant is 4, and the algorithm assures this constant only if the sizes of $|G_i|, i = 1 \dots t$ are all equal. A constant approximation algorithm that works also when the sizes of $|G_i|, i = 1 \dots t$ are not necessarily equal is described in [27]. A full analysis of this algorithm, which achieves an error constant of 2, is given in [28]. A similar algorithm is also given in [29]. The cited error constants are determined by a worst-case analysis; in practice these algorithms usually find closer to optimal solutions, especially when there are few loops. We use the simplest greedy algorithm analyzed in [28] as part of our hybrid loop breaking algorithm.

4 Methods

Our loop breaker selection algorithm is implemented by modifying FASTLINK 3.0P [30]. It is implemented almost entirely as a subroutine called from the preprocessor program UNKNOWN, with a few modifications elsewhere, primarily for the case of multiple marriages. This program organization implies that the pedigree is input with a set of loop breakers selected by the user, perhaps with the assistance of the LOOPS [18] program. If our algorithm can find a better set of loop breakers, the intermediate representation of the pedigree file (`pedfile.dat` and `ipedfile.dat`) is modified accordingly. Our method could free the user completely from the burden of selecting a loop breaker set, but this deviates from entrenched patterns of usage for the LINKAGE package. FASTLINK is available by anonymous ftp at `fastlink.nih.gov` in the subdirectory `pub/fastlink`.

We compared FASTLINK, version 3.0P to the new version on some data sets with looped pedigrees. All the runs were done using ILINK. We use a three-locus run (disease plus two markers) in each case, except ALZ, where the data set has only one marker locus. The timing experiments were run on an unloaded Sun SPARCStation 20 computer with 128 Mbytes of RAM. This machine runs the operating system SunOS, version 5.5, which is also known as Solaris, version 2.5, and is an implementation of UNIX. To compile all versions of the programs we used the `gcc` compiler, version 2.7.2 using the `-O` flag for optimization. The times reported in the next section are the sum of the user and system times given by the `time` command. In those cases where the new algorithm finds a better set of loop breakers we also report estimates of the difference in the number of loop breaker vectors (i.e., $|G|$).

5 New Algorithm

Our main contributions are the development of a fast algorithm for the loop breaking selection problem using generalized cloning, and a demonstration, on real data, of a significant improvement in running time of linkage computations. As a subroutine of our algorithm, we developed another LBS algorithm, termed SPANNINGTREE, which is guaranteed to produce an *optimal* loop breaker set whenever the input pedigree has no multiple marriages. In this section we first describe the spanning tree algorithm, then we describe our main LBS algorithm for general pedigrees, LOOPBREAKER, and, finally, we conclude with experimental results.

Marriage graphs have the property that every path alternates between a vertex from I, representing an individual, and a vertex from M, representing a marriage. Whenever every individual is married at most once, each vertex i in I has at most two neighboring vertices m_1 and m_2 representing the marriage that brought i about and the one marriage that i may participate in. We can now apply the following three transformations. First, every path of the form $m_1 \rightarrow i \rightarrow m_2$ is replaced with an edge $m_1 — m_2$. Second, every individual i that is not married is removed from the marriage graph along with the edge $m_1 \rightarrow i$. Third, every founder i is removed from the marriage graph along with the edge $i \rightarrow m_2$. The resulting undirected graph is denoted by H'. The vertex set of H' is M and every edge represents an individual. We now set the weight $w(e)$ of each edge e in H' to be $\log |G_i|$—the logarithm of the number of genotypes of the corresponding individual.

Solving the LBS problem is equivalent to removing a set of edges whose sum of weights is minimum from the undirected graph H' such that we remain with a tree. This is an instance of the well studied maximum spanning tree problem in combinatorial optimization appearing in virtually every text book on graph algorithms (e.g., [31]). One simple algorithmic solution is as follows. Start with an empty graph T and repeat the following three steps: Select an edge e in H' with maximum weight $w(e)$. If e does not create a cycle in T, add it to T.

Remove e from H'. This algorithm was shown by Kruskal [32] to produce a tree T such that the sum of the edge weights in T is maximum. Consequently, the sum of weights of the edges left out is minimum. The *edges* left out in building T from the derived graph H' correspond to the *individuals* selected to be loop breakers in the marriage graph H.

The conversion of the marriage graph H to a graph H' is needed only for the purpose of demonstrating via Kruskal's famous result that the joint genotype vector size of the loop breakers set found by this algorithm is indeed minimum. Our algorithm can be easily described in terms of the marriage graph H itself without the transformation to the graph H', using the notation $H[V]$. The notation assumes that V is a subset of the vertices of H. The graph $H[V]$ is the undirected graph whose vertex set is V and whose edge set consists of all edges in H that connect two vertices in V. Our algorithm is as follows.

ALGORITHM SpanningTree

Input: *A marriage graph H with a marriage vertex set M,*
an individual vertex set I, and a genotype vector size $|G_i|$ for each $i \in I$.
Output: *A loop breaker set F of minimum joint genotype vector size.*

$F \leftarrow \emptyset$
While $I \neq \emptyset$ **do**
1. Pick a vertex $i \in I$ for which $|G_i|$ is maximum
2. **If** $H[M \cup \{i\}]$ is not a tree **then** $M \leftarrow M \cup \{i\}$ **else** $F \leftarrow F \cup \{i\}$
3. $I \leftarrow I \setminus \{i\}$

return F

When a pedigree contains multiple marriages the transformation from H to H' is no longer justified and a different approach should be preferred. The idea is that one individual that participates in multiple marriages can serve as a loop breaker to several loops and the question is how to select loop breakers so that the joint genotype vector size of the loop breaker set is as small as possible. There are two, possibly conflicting, selection criteria: We should select an individual that participates in as many marriages as possible that form loops, and we should select individuals that have a small genotype vector size $|G_i|$. The algorithm computes for each individual a cost $f(i)$ that depends on both criteria and greedily chooses an individual i with the smallest cost as a loop breaker. It then removes the individual i from the marriage graph. If the resulting graph has no multiple marriages anymore, then SpanningTree resolves the remaining problem optimally.

The function f uses a heuristic estimate of the number of loops in which i participates by first removing from the marriage graph vertices that do not participate in any loop and then checking the number of vertices that are neighbors of i in the remaining graph. This number is denoted by $d(i)$— the degree of i. The function f is then defined by $f(i) = \log(|G_i|)/d(v_i)$. Low values of $f(i)$ indicate that individual i breaks each loop at the cost of adding only a small number of genotypes to the loop breakers set. The algorithm is as follows.

ALGORITHM LoopBreaker

Input: *A marriage graph H with a marriage vertex set M, an individual vertex set I, and a genotype vector size $|G_i|$ for each $i \in I$.*
Output: *A loop breaker set F having low joint genotype vector size.*

$F \leftarrow \emptyset$
Repeatedly remove from H any vertex in $I \cup M$ that has only one neighbor
If no individual is multiply married **then return** $F \cup \text{SpanningTree}(H)$

While the graph H is not empty **do**
1. Pick an individual i for which $\log(|G_i|)/d(i)$ is minimum in H
2. $F \leftarrow F \cup \{i\}$
3. $I \leftarrow I \setminus \{i\}$
4. Repeatedly remove from H any vertex in $I \cup M$ that has only one neighbor

If no individual is multiply married **then** return $F \cup \text{SpanningTree}(H)$
return F

The loop breaker sets found by LoopBreaker are not guaranteed to be optimal but in the data sets we have examined, the results seem optimal, due to the small number of loops in human pedigrees,

There are some subtle technical changes we introduced to solve two implementation problems. One problem is that LINKAGE/FASTLINK requires the input pedigree to correspond to a marriage graph that is a tree (cf. the discussion on the MD data set below). When a pedigree has a loop breaker with k marriages, it may be the case that making $k+1$ copies disconnects the marriage graph. Therefore, we start with 2 copies and keep adding copies, unless the marriage graph becomes disconnected.

A second problem in FASTLINK is to estimate the sizes of possible genotype sets $|G_i|$. For this purpose we employ the user's initial selection for loop breakers and carry out FASTLINK's built-in genotype inference algorithm in order to rule out impossible genotypes. This method does not rule out all impossible genotypes, however, it provides sufficiently-accurate relative estimates of $|G_i|$ for different candidate loop breakers.

6 Results

We illustrate the performance of the new version of FASTLINK using the following data sets, each containing one looped pedigree. We also tested with a few other looped pedigrees where the new software did not change the user's loop breaker selection. The running time of LoopBreaker is too negligible to be effectively measured.

- BAD: data on a portion of the Old Order Amish pedigree 110 (OOA 110), with bipolar affective disorder (BAD) from the laboratory of David R. Cox and Richard M. Myers at the University of California at San Francisco [33]. This pedigree has 1 loop. The assumed mode of inheritance is dominant with reduced penetrance.

- ALZ: Data on an Amish family with Alzheimer's disease from Margaret Pericak-Vance, Jonathan Haines, and Marcy Speer [34]. The pedigree was sent to us with 4 loop

Data Set	Old N	New N	Old Time	New Time	Speedup
ALZ	15552	8256	67m	18m	3.7
BAD	8	3	154s	61s	2.5
JP	126	6	329s	16s	21
MD	3072	32	1297m	28m	46
RP01	1296	364	247m	41m	6.0

Table 1: Comparison of FASTLINK 3.0P and new code

breakers, but with our new code we discovered that this disconnected the pedigree. Only 3 loop breakers are needed. The results in the table use 3 of the original loop breakers to start. The assumed mode of inheritance is dominant.

- JP: data on a small family some of whose offspring exhibit autosomal recessive juvenile parkinsonism from Shoji Tsuji [35]. This is a small pedigree (number 547 in the cited paper) with one loop created by a marriage of second cousins once removed. We use the pedigree as sent to us by Dr. Tsuji.

- MD: data on a large family exhibiting two clinically distinct forms of muscular dystrophy from Ken Morgan, Tracey Weiler, Cheryl Greenberg, and Klaus Wrogemann [36]. The pedigree was presented to us with 7 loop breakers. There are 2 multiply married individuals. The mode of inheritance is recessive.

- RP01: data on a large family, UCLA-RP01, with autosomal dominant retinitis pigmentosa (RP1) from the laboratory of Stephen P. Daiger. This pedigree has 2 loops and 2 multiply married individuals. As shown in [37], this pedigree had to be split into 3 pieces because computation on the whole family together was prohibitively long. Here we leave the loops in.

More detailed descriptions of the data sets can be found in the papers cited for each one.

The results are shown in Table 1. We report a number N, which is the number of loop breaker genotype vectors for which the likelihood is not provably 0, by any of the preliminary checks in FASTLINK. The number N is not a perfect predictor of running time because the preliminary checks for 0 likelihood are not exhaustive; the pedigree traversal time varies with the loop breaker genotypes, and the average pedigree traversal time changes when the loop breaker set changes.

The improvements on BAD and JP illustrate that even on 1-loop pedigrees, linkage experts will not always choose the optimal loop breaker. The improvements on MD and RP01 illustrate the case of multiple marriages in different ways. The original MD used 7 loop breakers; by using multi-copy cloning, we can reduce the number to 5. Both the original RP01 and the new RP01 use 2 loop breakers, but multi-copy cloning allows us to break a loop in a different place where $|G_i|$ is smaller.

The improvements in ALZ are important even beyond the running time change. The original ALZ pedigree was presented to us with 4 loop breakers. Using our algorithm we discovered that the associated marriage graph is disconnected, and only 3 loop breakers should be used. Using 4 loop breakers LINKAGE/FASTLINK gives plausible, but wrong results.

LINKAGE/FASTLINK had a longstanding flaw that some pedigrees with disconnected marriage graphs would be tolerated without a crash or 0 likelihood; this flaw is now fixed automatically by using LoopBreaker.

7 Discussion

In this paper, we addressed the loop breaker selection (LBS) problem in genetic linkage analysis and implemented practical solutions in FASTLINK. We presented an optimal algorithm for pedigrees without multiple marriages and an approximation algorithm for general pedigrees. The latter algorithm uses a generalized cloning operation that allows more than 2 copies of a multiply married individual. We illustrated that the new algorithm speeds up the computations on real pedigrees.

Users of FASTLINK will no longer have to think carefully about where to break their pedigree loops. For the sake of backwards compatibility to previous versions of LINKAGE/FASTLINK and syntactic compatibility with other programs, we require the user to choose a loop breaker set. Choosing a valid loop breaker set is not difficiult to do using the LOOPS program, as described in Section 2. What is difficult for users to do, and what we have automated is to choose a loop breaker set that leads to a fast running time of the linkage analysis. We fixed the longstanding flaw that LINKAGE/FASTLINK might give plausible, incorrect results when the user specified too many loop breakers.

Our implementation is specific to FASTLINK, but the algorithms we described may be applicable to other software packages. Most notably, VITESSE [38] is currently the best linkage analysis package for large simple pedigrees, but it does not currently accept complex pedigrees. It seems possible to extend VITESSE to complex pedigrees using the approach of Lange and Elston [9], in which case our algorithms for the LBS problem would apply.

Our work follows in the spirit of some other papers such as [13, 7] that defined mathematically precise optimization problems based on existing linkage analysis software, presented new algorithms for those problems, and demonstrated the practicality of these algorithms with software implementations.

Acknowledgments

The research of Dan Geiger is supported by the fund for promotion of research at the Technion. Thanks to David R. Cox, Stephen Daiger, Cheryl Greenberg, Ken Morgan, Richard M. Myers, Marcy Speer, Shoji Tsuji, Tracey Weiler, and Klaus Wrogemann for providing the data sets used in this paper. The collection of the ALZ data set was supported by a grant from NIH. The collection of the MD data set was supported by grants from the Medical Research Council of Canada, the Muscular Dystrophy Association of Canada, and the Children's Hospital of Winnipeg Research Foundation. The development of the RP01 data set was supported by grants from the National Retinitis Pigmentosa Foundation and the George Gund Foundation.

Thanks to Jurg Ott for answering some questions about the LOOPS program. Thanks to Emil Ginzburg for stimulating discussions. Thanks to Doron Bustan, Yael Zbar, and other

students of the AI lab course of Winter 1997 at the Technion computer science department who helped us test various alternative algorithms for the LBS problem.

References

[1] Lathrop GM, Lalouel JM, Julier C, Ott J: Strategies for multilocus linkage analysis in humans. Proc Nat Acad Sci 1984; 81:3443–3446.

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

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

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

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

[6] Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, California, 1988.

[7] Kong A: Efficient methods for computing linkage likelihoods of recessive diseases in inbred pedigrees Genet Epidem 1991; 8:81–103.

[8] Elston RC, Stewart J: A general model for the analysis of pedigree data. Hum Hered 1971; 21:523–542.

[9] Lange K, Elston RC: Extensions to pedigree analysis. I. Likelihood calculation for simple and complex pedigrees. Hum Hered 1975; 25:95–105.

[10] Bennett RL, Steinhaus KA, Uhrich SB, O'Sullivan CK, Resta RG, Lochner-Doyle D., Markel DS, Vincent V, Hamanishi J: Recommendations for Standardized Human Pedigree Nomenclature. Am J Hum Genet 1995; 56:745–752.

[11] Cannings C, Thompson EA, Skolnick MH: Probability functions on complex pedigrees. Adv Appl Prob 1978; 10:26–61.

[12] Ott J: Estimation of the recombination fraction in human pedigrees: Efficient computation of the likelihood for human linkage studies. Am J Hum Genet 1974; 26:588–597.

[13] Lange K, Boehnke M: Extensions to pedigree analysis. V. Optimal calculation of mendelian likelihoods. Hum Hered 1983; 33:291–301.

[14] Lauritzen SL, Spiegelhalter DJ: Local computations with probabilities on graphical structures and their application to Expert Systems (with discussion). J Royal Stat Soc B 1988; 50:157–224.

[15] Reed B: Finding approximate separators and computing the treewidth quickly. Proc. 24th Ann. ACM Symp. Theory Comp., 1992, 221–228.

[16] Becker A., Geiger D: A sufficiently fast algorithm for finding close to optimal junction trees. Proc. 12th Conf. Uncertainty in Artif. Intel., 1996, 81–88.

[17] Shoikhet K., Geiger D: A practical algorithm for finding optimal triangulations. Proc. AAAI 97, 1997, to appear.

[18] Xie X, Ott J: Finding all loops in a pedigree. Am J Hum Genet 1992; 51:A205.

[19] Terwilliger JD, Ott J. Handbook of Human Genetic Linkage. The Johns Hopkins University Press, Baltimore and London, 1994.

[20] Pearl J. Fusion, propagation and structuring in belief networks. Artificial Intelligence 1986; 29:241–288.

[21] Kim H, Pearl J: A computational model for combined causal and diagnostic reasoning in inference systems. Proc. 8th Inter. Joint Conf. Art. Intel., 1983, 190–193.

[22] Pearl, J: A constraint-propagation approach to probabilistic reasoning. Proc. Workshop on Uncertainty and Probability in Artificial Intelligence, Los Angeles, CA (1985) 31–42; also in L.N. Kanal and J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, North-Holland, Amsterdam, 1986, 357–370.

[23] Geiger D, Verma TS, Pearl J: Identifying independence in Bayesian networks. Networks 1990; 20:507–534.

[24] Suermont HJ, Cooper GF: Probabilistic inference in multiply connected belief networks using loop cutsets. Int J Approx Reasoning 1990; 4:283–306.

[25] Karp, RM: Reducibility among combinatorial problems. In R.E.Miller and J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, 85–103.

[26] Bar-Yehuda R, Geiger D, Naor J, Roth RM: Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and bayesian inference SIAM J. Computing, in press.

[27] Becker A. and Geiger D., The loop cutset problem, Proc. of the Tenth conference on Uncertainty in Artificial Intelligence, 1994, 60–68.

[28] Becker A, Geiger D: Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem Artificial Intelligence 1996; 83:167–188.

[29] Bafna V., Berman P., and Fujito T., Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs, Proc. Sixth Annual International Symposium on Algorithms and Computation (ISAAC 95) . Lecture Notes in Computer Science, volume 1004. , Springer-Verlag, Berlin, 1995, 142–151.

[30] Schäffer AA. Faster linkage analysis computations for pedigrees with loops or unused alleles. Hum Hered 1996; 46:226–235.

[31] Even S: Graph Algorithms, Computer Science Press, Rockville, Maryland, 1979.

[32] Kruskal JB: On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Amer Math Soc 1956; 7:48–50.

[33] Law A, Richard III CW, Cottingham Jr . RW, Lathrop GM, Cox DR, Myers RM: Genetic linkage analysis of bipolar affective disorder in an Old Order Amish pedigree. Hum Genet 1992; 88:562–568.

[34] Pericak-Vance MA, Johnson CC, Rimmler JB, Saunders AM, Robinson LC, Hondt EG, Jackson CE, Haines JL: Alzheimer dementia and APOE-4 in an Amish population. Ann Neur 1996; 39:700–704.

[35] Matsumine H, Saito M, Shimoda-Matsubayashi S, Tanaka H, Ishikawa A, Nakagawa-Hattori Y, Yokochi M, Kobayashi T, Igarashi S, Takano H, Sanpei K, Koike R, Mori H, Kondo T, Mizutani Y, Schäffer AA, Yamamura Y, Nakamura S, Kuzuhara S, Tsuji S, Mizuno Y: Localization of a gene for an autosomal recessive form of Juvenile Parkinsonism to chromosome 6q25.2-27. Am J Hum Genet 1997; 60:588–596.

[36] Weiler T, Greenberg CR, Nylen E, Halliday W, Morgan K, Eggerston D, Wrogemann K: Limb-Girdle muscular dystrophy and Miyoshi myopathy in an aboriginal Canadian kindred map to *LGMD2B* and segregate with the same haplotype. Am J Hum Genet 1996; 59:872–878.

[37] Blanton SH, Heckenlively JR, Cottingham AW, Friedman J, Sadler LA, Wagner M, Friedman LH, Daiger SP: Linkage mapping of autosomal dominant retinitis pigmentosa (RP1) to the pericentric region of human chromosome 8. Genomics 1991; 11:857–869.

[38] O'Connell JR, Weeks DE: The VITESSE algorithm for rapid exact multilocus linkage analysis via genotype set-recoding and fuzzy inheritance. Nature Genet 1995; 11:402–408.