Faster Linkage Analysis Computations for Pedigrees with Loops or Unused Alleles

Alejandro A. Schäffer *

National Center for Human Genome Research
National Institutes of Health
Bethesda,MD U.S.A.
and
Department of Computer Science
Rice University
Houston, TX U.S.A.

Keywords: Genetic linkage analysis, algorithms, loops, unknown gentopyes

Address for correspondence: Alejandro A. Schäffer, National Institutes of Health, Building 38A/Room 8N805, 8600 Rockville Pike, Bethesda, MD 20894.
FAX: (301) 480-9241
Telephone: (301) 496-2477 x202

For the Section on: **Methodological Issues**

*schaffer@nchgr.nih.gov

Abstract

There seems to be no limit to the complexity of computations that genetic linkage analysts want to do. Two primary factors that increase the length of computations are pedigree loops and unknown genotypes. I describe the implementation in FASTLINK of some algorithmic improvements to partly address the problems of pedigree loops and unknown genotypes. LINKAGE is by far the most popular software package to do lod score computations on disease pedigrees. FASTLINK is derived from LINKAGE 5.1 and compatible with it. In contrast to LINKAGE, FASTLINK has the virtues that it is faster sequentially, runs well in parallel, is more robust against errors, and includes substantial new documentation. One of the new improvements allows the detection of violation of Mendelian rules of inheritance in input pedigree files with loops. This error-detection capability was not provided in any previous version of LINKAGE or FASTLINK.

1 Introduction

Pedigree loops and unknown genotypes are banes to linkage analysts who hunt disease genes with the popular software package LINKAGE [14, 13, 15]. The slow running times these problems cause become more severe as geneticists increase the number of loci and the number of alleles per locus. In this paper I describe the implementation of several algorithmic improvements to address these problems. Because LINKAGE is much more popular than competing packages, such as MENDEL [12] or PAP [9], it is of practical importance to software users to implement new algorithms in a LINKAGE-compatible manner. I have implemented the improvements in FASTLINK, which is compatible with LINKAGE 5.1, but has the additional benefits that it is faster [4, 24], runs well in parallel [6, 8], is robust against a variety of errors such as computer crashes [24], and includes substantial new documentation(e.g., [22, 21, 23]). The improvements described herein constitute the major code differences between versions 2.3P and 3.0P of FASTLINK.

In this paper I return to the initial goal of the FASTLINK project, implementing faster sequential algorithms for linkage analysis. The improvements described herein differ conceptually from those in [4] in two important ways.

First, the new improvements are targeted at two specific problems (loops and unknown genotypes), while the initial improvements were targeted at virtually all inputs. As a result, the new version of FASTLINK shows measurable improvement only on some inputs, while FASTLINK 1.0 improved on LINKAGE 5.1 for virtually all inputs. However, when the new improvements apply, they can yield a speedup much more substantial than any individual improvement in FASTLINK 1.0.

Second, the new improvements require substantial recoding of seemingly unrelated parts of the approximately 30,000 lines of FASTLINK, while the initial improvements were focused on one complicated section of LINKAGE code. The complexity of the software engineering likely explains why the ideas behind our new improvements are known to a handful of linkage analysis experts, but they were never implemented in LINKAGE.

This paper is organized as follows. The next section describes some background information on linkage analysis, LINKAGE, and FASTLINK. This is followed by two sections on the new algorithms, two sections describing speedup measurements, and finally a short Discussion.

2 Background

In this section I review some background on linkage analysis, the LINKAGE package, and FASTLINK. The basic input to LINKAGE includes an ordered list of loci $L_1, L_2, \ldots, L_{k+1}$ with allele frequencies at each locus and one or more pedigrees with some individuals typed at some loci. The general goal of linkage analysis computations is to evaluate the likelihood of a candidate vector $[\theta_1, \ldots, \theta_k]$ of recombination fractions for the input pedigree(s) and locus order. The component θ_i is the candidate recombination fraction between the loci L_i and L_{i+1}. LINKAGE provides four programs ILINK, LINKMAP, LODSCORE, MLINK to do likelihood evaluations and help select those vectors that give the highest likelihoods. The programs differ largely in how they choose the candidate θ vectors, which is not relevant to this paper.

The LINKAGE package also includes a preprocessor program UNKNOWN, which is conventionally used before all the programs except LODSCORE. The purposes of UNKNOWN are to detect violations of Mendelian rules of inheritance and to infer possible genotypes for all untyped individuals who have children in the pedigree. If the input is error-free, UNKNOWN outputs a file of possible genotypes called speedfile.dat, which is used by the main program to reduce the number of possible genotypes of unknown individuals during the likelihood computation. FASTLINK replaces the four main LINKAGE programs and UNKNOWN; through FASTLINK version 2.3P it was possible to use versions of UNKNOWN distributed with LINKAGE, but much of the new code in version 3.0P is in UNKNOWN. All the other LINKAGE preprocessor programs are used unchanged with FASTLINK.

A significant flaw of UNKNOWN is that for pedigrees with loops, UNKNOWN does no error-checking and no genotype inference. All genotypes are listed as possible in speedfile.dat. The major thrust of our current improvements is to correct these two flaws in UNKNOWN. The looped pedigree flaws in UNKNOWN were well-hidden for many years because UNKNOWN was essentially undocumented. I supplied documentation [23] beginning with FASTLINK, version 2.2.

For loopless pedigrees, the likelihood evaluation proceeds by visiting one nuclear family at a time, *peeling* that family from the rest of the pedigree. The order of the nuclear family peeling is described in [22]. For our purposes what matters is that they occur in such a way that the current nuclear family (except the last one) has exactly one member m who connects that family to the unvisited part of the tree. For each genotype i, we compute:

$$P(m, i \mid V, \theta),$$

which is the probability that person m expresses the phenotype associated with genotype i, conditional on the genotypes of people already visited, V, and the candidate θ. If r is the last person visited, then the pedigree likelihood is $\Sigma_i P(r, i)$.

The peeling approach to computing the likelihood was first suggested by Elston and Stewart [7] for pedigrees that can be peeled from bottom to top, but even for such pedigrees LINKAGE does not necessarily peel in the Elston-Stewart order [22]. Peeling was generalized to arbitrary pedigrees by Lange and Elston [11]. They proposed that a loop can be "broken" by replacing one individual with two copies, where one copy acts as parent and spouse and the other copy acts as child and sibling. The number of loops in a pedigree is the minimum number of individuals who must be *cloned* into two to make the pedigree into a tree. I call the cloned individuals *loop breakers*.

A comprehensive explanation of how loops are handled in LINKAGE and FASTLINK can be found in the combination of Chapter 7 of [25] and [21]. I summarize a few salient points here. Suppose first that there is only one loop and the loop breaker is b. To use the tree traversal algorithm on the adjusted pedigree correctly, one must force the genotype of the two copies of b, g_b to be identical. Therefore, the likelihood of pedigree P is computed as:

$$\Sigma_g L(P \mid g_b = g, \theta)$$

Where g ranges over all genotypes and $L(P \mid g_b = g, \theta)$ is the pedigree likelihood when b is constrained to have genotype g.

The number of pedigree traversals required to compute the likelihood is the number of possible genotypes of b. For this reason, users are encouraged to choose loop breakers (in the preprocessor program MAKEPED) of known genotype, if possible. A previous algorithmic improvement in FASTLINK [24] saves that part of the traversal computation that is independent of the genotype of b, thereby mitigating slightly the cost of multiple traversals. The running time grows exponentially if the pedigree has $j > 1$ loops. In this case, the user must define a vector $[b_1, \ldots, b_j]$ of loop breakers and the likelihood computation must do a traversal for each possible genotype combination of $[g_{b_1}, \ldots, g_{b_j}]$. For example, if there are 3 loops, and the loop breakers have 12, 20, and 25, possible genotypes respectively, then one likelihood computation requires $12 \times 20 \times 25 = 6000$ pedigree traversals.

FASTLINK is available by anonymous ftp to softlib.cs.rice.edu in the subdirectory pub/fastlink.

3 Better Algorithms for Loops

This section describes the algorithmic improvements for looped pedigrees that are new to FASTLINK 3.0P. The improvements address the cases of loop breakers with known genotypes and loop breakers with unknown genotypes. The improvements work well together on pedigrees with loop breakers of each kind, as is illustrated under Results. The improvements for the unknown genotype case require significant changes to both UNKNOWN and the main programs.

I consider first the unknown genotype case. Here the improvements within UNKNOWN can be summarized as follows:

For each locus L
    For each vector V_L of genotypes assigned to the loop breakers:
        Run the algorithm in UNKNOWN to infer
        possible genotypes for other individuals
        Store the possible genotypes for each individual
        indexed by loop-breaker vector
        If for some individual and locus there are
        no possible genotypes,
            then V_L is not a possible loop-breaker vector
    If all loop-breaker vectors are impossible,
        then (and only then) there is a Mendelian error in the pedigree.

Despite this simple description, the implementation required well over 1000 lines of new code for a variety of reasons. We structured the code so one is able to turn on and off the loop improvements for easy testing, debugging, and timing. However, this increased the code size substantially. I highlight some of the other difficult implementation issues here.

First, the list of possible genotypes for each loop-breaker vector and locus must be communicated to the main program. We solved this by using a file, loopfile.dat, similar in spirit to speedfile.dat. UNKNOWN writes out loopfile.dat and the main program reads it in. Large amounts of code were written to give the data a reasonable format that might be comprehensible to the debugging human eye as well as to to the computer. Details of the loopfile.dat format are given in README.loopfile, which will be distributed with FASTLINK 3.0P.

Second, the problem of encoding and decoding genotypes required a surprising amount of code. The problem arises due to implementation choices made long ago in LINKAGE that we found too entrenched to change. Consider a 3-locus phase known joint genotype:

$$A_1 \mid A_2$$

$$B_1 \mid B_2$$

$$C_1 \mid C_2$$

UNKNOWN works one locus at a time, so its code converts the pairs (A_1, A_2), (B_2, B_2), and (C_1, C_2) into single locus genotypes. The main programs do all loci at once, so they first translate the columns (A_1, B_1, C_1) and (A_2, B_2, C_2) into joint haplotypes, H_1 and H_2 and then translate that pair into a joint genotype. In the main program a loop-breaker vector is presented as a vector of joint genotypes encoded as haplotype pairs. To look up the loop-breaker vector one must decode the haplotype pairs into single-locus alleles and then re-encode them into a vector of single locus genotypes.

Third, we are storing the possible genotypes for each individual and each loop-breaker vector. Because, UNKNOWN works one locus at a time this amount of information is not too large. The information for one locus is written out before the next locus is started. However, the number of loop-breaker vectors can get arbitrarily large as the number of loops grows, straining space usage. Therefore, the user can limit how many loop-breaker vectors are considered, which implicitly limits how many loops are considered. Substantial code is required to safely distinguish those additional loops that are not considered when the number of loop-breaker vectors gets too large.

My improvement for the case of loop breakers of known genotype corrects a small, but serious algorithmic inefficiency in the LINKAGE implementation. To illustrate the inefficiency, suppose that there are 3 loop breakers and 50 joint genotypes. Then the routine `likelihood` used to iterate over all $50 \times 50 \times 50 = 125,000$ vectors in the order $[1,1,1] \ldots [50,1,1] \ldots [1,2,1] \ldots [50,50,1]$, $[1,1,2], \ldots [50,50,2]$. The old code did a pedigree traversal only for those vectors that are consistent with the known genotypes. However, it did a separate check for each vector $[i,j,k]$ to see if it is possible. This is inefficient because if k is not a possible genotype for the 3rd loop breaker, all 2500 vectors that end in k can be ruled out at once, rather than checking each vector separately.

In the case where the loop breakers have known genotypes even at a single locus, most of the genotype vectors are ruled out. Run-time profiling of the MD1 example (described in Results) showed that more than 95% of the time was spent ruling out impossible loop-breaker vectors. Ruling out a loop-breaker vector can be done in under a millisecond with just a few lines of code. However, when there are multiple loops, the number of vectors being considered was growing exponentially in the number of loops.

To fix this problem, the new code precomputes a list of possible genotypes for each loop breaker. Then it iterates over vectors composed only of possible genotypes for each loop breaker. This is a product that is still exponential in the number of loops, but the terms in the product are much smaller. Specifically if the genotype of a loop breaker is known, it now contributes 2^{h-1} where h is the number of heterozygous loci, rather than the number of genotypes. In the example run done with MD1 this would replace the term 171 (number of genotypes) with a term of at most 2 for each loop breaker with known genotype. In that case 3 of the 4 loop breakers have known genotype.

4 Amalgamating Unused Alleles

Allele amalgamation is a technique that reduces the number of possible haplotypes for each individual in a pedigree. Since the running time and space used in LINKAGE and FASTLINK grow roughly as the fourth power of the number of haplotypes, reducing the number of alleles at any locus is useful. Allele amalgamation is feasible when there are k possible alleles at a locus, but only $j < k - 1$ alleles appear in known genotypes of individuals in the pedigree. When this occurs, the $k - j$ unused alleles can be combined into one allele, which I call the *catch allele*. The frequency of the catch allele is specified as the sum of the frequencies of the original unused alleles. For example, suppose we specify 5 alleles, with frequencies: 0.4, 0.25, 0.15, 0.12, 0.08. If only the first 3 alleles are used, then the last two are combined into a catch allele of frequency $0.2 = 0.12 + 0.08$.

Allele amalgamation was already implemented in MENDEL [12], but was not documented, except in a tiny description of the parameter where one can turn it on or off [5]. Allele amalgamation is particularly helpful for pedigrees with unknown genotypes because it reduces significantly the number of possible genotypes that the untyped individuals can have. I mention allele amalgamation here for 3 reasons:

1. To bring the method to the attention of more linkage analysts and explain some subtle implementation issues.

2. To document that amalgamation is now implemented in FASTLINK, which will help many users.

3. To make the assessment of the speedups provided by the looped pedigree improvements more fair.

Allele amalgamation is relatively straightforward to do, by hand editing the input files, if there is only one pedigree. The need for automation arises when data sets have multiple pedigrees. Then what typically happens is that among the entire data set all k alleles occur, but within each specific pedigree some alleles may be unused. Since the likelihood is computed one pedigree at a time, it is highly desirable to use the minimum number of required alleles and haplotypes for each pedigree. The likelihood evaluation routines in LINKAGE and FASTLINK assume that the alleles are numbered consecutively starting at 1 for easy table lookups. Therefore implementing allele amalgamation requires renumbering the alleles that are present starting at 1.

I added some routines that are used before any likelihood evaluations are done to assess the prospects for allele amalgamation. These routines determine for each [pedigree, locus] pair which alleles are present and compute the mapping between old allele numbers and frequencies and new allele numbers and frequencies. The alleles associated with each individual can be renumbered just once before any likelihoods are evaluated because each individual appears in only one pedigree. However, the tables that store the correspondences between allele numbers and haplotype numbers, and between haplotype numbers and genotype numbers, may need to be recomputed between each likelihood evaluation of different pedigrees. The tables need to get recomputed if and only the number of alleles present at any locus changes, so the new code predetermines between which pairs of consecutive pedigrees the tables stay the same and between which pairs the tables must be recomputed. The pedigrees continue to be analyzed in the same order they are input, to simplify achieving output compatibility with LINKAGE.

A further substantial complication is that the allele amalgamation must be integrated with the loop improvements. I found that the implementation of the loop improvements was greatly simplified if I required the allele numbers in loopfile.dat to be the post-amalgamation allele numbers because this makes it unnecessary to have extra genotype to haplotype to allele conversion routines. As a consequence, UNKNOWN uses almost the same allele amalgamation routines as the main FASTLINK programs. Doing allele amalgamation in UNKNOWN has the further benefits of speeding it up and reducing the size of loopfile.dat. A special constant is prepended to speedfile.dat specifying whether allele amalgamation is "on" or "off", to ensure that the main program will proceed if and only if it is in the same mode as UNKNOWN.

5 Methods

I compared FASTLINK version 2.3P to FASTLINK 3.0P on some data sets where the changes are likely to make the newer version run faster. Version 2.3P is the most recent release of FASTLINK from August 1995. The timing experiments were run on a lightly loaded Sun SPARCStation 5 computer with 64 Mbytes of RAM. This machine runs the operating system SunOS, version 5.4, which is also known as Solaris, version 2.4, and is an implementation of UNIX. To compile all versions of the programs we used the `gcc` compiler, version 2.7.0 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.

6 Results

I present timing for FASTLINK 3.0P with several input data sets. To keep things simple and relevant to most users, I show user + system time when running on 1 processor. The times would be much smaller if the runs had been done in parallel on multiple processors, which has been feasible beginning with FASTLINK, version 2.3P The timings come from the following data sets.

- RP01: data on a large family, UCLA-RP01, with autosomal dominant retinitis pigmentosa (RP1) from the laboratory of Dr. Stephen P. Daiger at the University of Texas Health Science Center at Houston. This family has 2 loops. As shown in [3], this pedigree had to be split into 3 pieces because computation on the whole family together was prohibitively long. Here I leave the loops in.
- BAD1: data on a portion of the Old Order Amish pedigree 110 (OOA 110), with bipolar affective disorder (BAD) from the laboratory of Drs. David R. Cox

and Richard M. Myers at the University of California at San Francisco [16]. This pedigree has 1 loop. The assumed mode of inheritance is dominant.

- BAD2: Data from chromosome 18 on 22 pedigrees with with bipolar affective disorder (BAD), supplied by Dr. Lynn Goldin [2]. One large pedigree in the set has a loop. The data set has two different disease models, and I happened to use the one that was dominant.

- CLP: Data on 12 families with autosomal dominant nonsyndromic cleft lip and palate (CLP) from the laboratory of Dr. Jacqueline T. Hecht at the University of Texas Health Science Center at Houston [10].

- MD1: Data on one Brazilian family with one form of autosomal recessive muscular dystrophy from Dr. Jacques Beckmann [19]. This pedigree has 4 loops.

- MD2: Data on one Tunisian family with a different form of autosomal recessive muscular dystrophy from Dr. Marcy Speer [1]. This pedigree has 6 loops, but Dr. Speer reports that the linkage analysis computations were deemed infeasible with all 6 loops, so only 3 were left in originally.

- ALZ: Data on an Amish family with Alzheimer's disease from Drs. Margaret Pericak-Vance, Jonathan Haines, and Marcy Speer [20]. This pedigree has 4 loops. The assumed mode of inheritance is dominant.

- WS: Data on an Italian family with Werner syndrome from Dr. Ellen Wijsman [17, 26]. This pedigree has 3 loops. The data set is particularly interesting for methodological studies because it includes descriptions for two marker loci, but no marker genotypes are available, only affection status is available. The loop improvements help here because by choosing genotypes for the 3 loop breakers, one can infer genotypes for others. The mode of inheritance is dominant.

More detailed descriptions of the data sets can generally be found in the papers cited for each one. All the runs, except for BAD2, were done with ILINK; the BAD2 runs were done with MLINK to ensure that each run would include the same number of likelihood evaluations for all runs. The choice of allele product represents either all the markers that are available or a run of modest size for FASTLINK 2.3P. Several of the data sets used reduced penetrance and multiple liability classes for the disease locus; the data sets BAD1 and BAD2 also assumed a non-zero phenocopy rate.

Table 1 shows sample speedups that users would perceive. The only general conclusions one should draw are that all the improvements help, and the amount of speedup varies widely from data set to data set. The RP01, BAD1, MD1, ALZ, and WS data sets have no opportunity for allele amalgamation, so they show the

Data Set	No. of Alleles	2.3P Time	3.0P Time	Speedup
RP01	$2 \times 5 \times 3$	703m	186m	3.8
BAD1	$2 \times 4 \times 4$	397s	275s	1.4
BAD2	$2 \times 9 \times 10$	1507m	43m	35
CLP	$2 \times 4 \times 4 \times 4$	40m	21m	1.9
MD1	2×9	578m	24m	24
MD2(3)	$2 \times 4 \times 3$	4206s	86s	49
MD2(6)	$2 \times 4 \times 3$	∞	122m	∞
ALZ	2×4	1670m	884m	1.9
WS	$2 \times 2 \times 2$	223m	89m	2.5

Table 1: Comparison of FASTLINK 2.3P and 3.0P

Loops	Allele	Time
Off	Off	612m
On	Off	266m
Off	On	29m
On	On	6m

Table 2: Comparison of Improvements on Looped Pedigree from BAD2

effect of the loop speedups. The CLP data set has no loops, so it shows the effect of allele amalgamation on a data set where it applies only to some of the pedigrees. The MD2 data set is shown with 3 loops and all 6 loops.

In the MD2 data set, it is possible to reduce the number of alleles at the second locus from 4 to 3. I did this renumbering by hand editing the file, and the run took 857s. One might be tempted to conclude that the speedup due to allele amalgamation is $4206/857 = 4.9$ and the speedup due to the loop improvements is $857/86 = 10.0$. However, this would be erroneous, since the improvements interact.

The BAD2 data set also allows one to measure the effect of each improvement separately. I extracted the one looped pedigree it contains and tried the same run as in Table 1 on that pedigree alone. I tried the run with the loop improvements on/off and the allele improvements on/off and the results are shown in Table 2.

In this case the effect of the allele amalgamation is clearly greater, and this is because it is possible to reduce the allele product from 180 all the way down to 60. Nevertheless, it is worth noting that the speedup on the looped pedigree by itself is roughly 100, while the speed up on the entire data set is 35. Put another way, without the new improvements, the looped pedigree consumes roughly 40% of the total time, while with the improvements, it consumes roughly 14% of the total time.

7 Discussion

I described the implementation of several algorithmic improvements in the linkage analysis package FASTLINK 3.0P. These improvements speed up FASTLINK computations on data sets with looped pedigrees and data sets with unused alleles. The running time measurements above show that the new algorithms can make a qualitative difference in which linkage analysis computations can be carried out within a reasonable amount of time.

The amalgamation of unused alleles was implemented in MENDEL some years ago. Somehow this feature of MENDEL did not get the publicity it deserved. The ideas behind the loop improvements were also known to experts, but getting the implementation details correct proved to be quite complicated.

O'Connell and Weeks [18] have developed and recently a new linkage analysis package called VITESSE, which is compatible with LINKAGE and FASTLINK and replaces the MLINK and LINKMAP programs. VITESSE is significantly faster than FASTLINK on some data sets, especially for multilocus analysis. At this time, VITESSE does not replace ILINK, and does not handle complex pedigrees. Therefore, the improvements described herein will be useful to linkage analysts who want to use ILINK or have data sets with looped pedigrees.

From the user's pragmatic point of view the most important change in FASTLINK 3.0P improves functionality rather than running time. The preprocessor program UNKNOWN now detects violations of Mendelian rules of inheritance in looped pedigrees. All previously distributed versions of LINKAGE and FASTLINK did not detect such errors. This omission caused some users to waste large amounts of computing time only to get a pedigree likelihood of 0 because the input genotypes were inconsistent.

Acknowledgments

I should like to thank many people for there assistance in this research: Dylan Cooper and Christopher Hoelscher for programming assistance; Christopher Hyams for assisting many FASTLINK users, so that I could write more new code; hundreds of FASTLINK users around the world for having the courage to try new software and the patience to wait for improvements; Lucien Bachner, Lynn Goldin, Kimmo Kallio, Cathryn Lewis, Jim Tomlin, and Gerard Tromp for beta testing FASTLINK 3.0P; Lucien Bachner for a posting to bionet.molbio.gene-linkage raising the problem of detecting Mendelian errors in looped pedigrees; Jacques Beckmann, Susan Blanton, David Cox, Stephen Daiger, Jonathan Haines, Jacqueline Hecht, Lynn Goldin, Richard Myers, Margaret Pericak-Vance, Marcy Speer, and Ellen Wijsman for challenge data sets. Development of the RP01 data set was supported by grants from the National Retinitis Pigmentosa Foundation and the George Gund Foundation. Development of the BAD1 data set was supported by a grants from NIH.

Development of the CLP data set was supported by grants from NIH and Shriners Hospital. Development of the MD2 data set was supported by a grant from NIH and grants from the Muscular Dystrophy Association. Development of the ALZ data set was supported by a grant from NIH. Collection of the WS pedigree was supported by grant from NIH. The part of this research done at Rice University was partially supported by grants from NIH and NSF and a contract from IBM.

References

[1] Ben Othmane K, Ben Hamida M, Pericak-Vance MA, Ben Hamida C, Blel S, Carter SC, Bowcock AM, Petrukhin K, Gilliam TC, Roses AD, Hentati F, Vance JM: Linkage of Tunisian autosomal recessive duchenne-like muscular dystrophy to the pericentromeric region of chromosome 13q. Nature Genet 1992; 2:315–317.

[2] Berrettini WB, Ferraro TN, Weeks D, Goldin LR, Detera-Wadleigh S, Nurnberger Jr. JI, Gershon, ES: Pericentric chromosome 18 DNA markers and manic-depressive illness: evidence for a susceptibility gene. Proc Nat Acad Sci 1994; 91:5918–5921.

[3] 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.

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

[5] Diehl SR, Lange KL, MacLean, C: Personal Communication, 1995.

[6] Dwarkadas S, Schäffer AA, Cottingham Jr. RW, Cox AL, Keleher P, Zwaenepoel W: Parallelization of general linkage analysis problems. Hum Hered 1994; 44:127–141.

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

[8] Gupta SK, Schäffer AA, Cox AL, Dwarkadas S, Zwaenepoel W: Integrating parallelization strategies for linkage analysis. Computers and Biomedical Res 1995; 28:116–139.

[9] Hasstedt SJ: *PAP - Pedigree Analysis Package, Revision 3.* Department of Human Genetics, University of Utah, Salt Lake City, Utah, 1989.

[10] Hecht JT, Wang Y, Connor B, Blanton SH, Daiger SP: Non-syndromic cleft lip and palate: No evidence of linkage to HLA or factor 13a. Am J Hum Genet 1993; 52:1230–1233.

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

[12] Lange K, Weeks D, Boehnke M: Programs for pedigree analysis: MENDEL, FISHER, and dGene. Genet Epidem 1988; 5:471–473.

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

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

[15] 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.

[16] 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.

[17] Nakura J, Wijsman EM, Miki T, Kamino K, Yu CE, Oshima J, Fukuchi KI, Weber JL, Piussan C, Melaragno MI, Epstein CJ, Scappaticci S, Fraccaro M, Matsumura T, Murano S, Yoshida S, Fujiwara Y, Saida T, Ogihara T, Martin GM, Schellenberg GD: Homozygosity mapping of the Werner Syndrome locus(WRN). Genomics 1994; 23:600–608.

[18] 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.

[19] Passos-Bueno MR, Richard I, Vainzof M, Fougerousse F, Weissenbach J, Broux O, Cohen D, Akiyama J, Marie SKN, Carvalho AA, Guilherme L, Kalil J, Tsanaclis AM, Zatz M, Beckmann JS: Evidence of genetic heterogeneity in the autosomal recessive adult forms of limb-girdle muscular dystrophy following linkage analysis with 15q probes in Brazilian families. J Med Genet 1993; 30:385–387.

[20] 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. Submitted, 1995.

[21] Schäffer AA: Loops in FASTLINK. One file in the FASTLINK documentation, 1993.

[22] Schäffer AA. Pedigree traversal in FASTLINK. One file in the FASTLINK documentation, 1993.

[23] Schäffer AA. The mystery of (the) Unknown. One file in the FASTLINK documentation, 1994.

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

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

[26] Thompson EA: Monte Carlo estimation of multilocus autozygosity probabilities. In *Proc. of the 1994 Interface Conference*, pp. 498–506. Interface Foundation of North America, 1994.