Loops in FASTLINK (revised for version 4.1P)

Alejandro A. Schäffer
formerly Rice University
and
currently National Institutes of Health

This document was originally written to accompany FASTLINK, version 2.0 and beyond. This version is a revision meant to accompany FASTLINK, version 4.1P and beyond. It describes some aspects of pedigree loops in FASTLINK. Due to the code modifications in versions 3.0P and 4.0P, the contents of this document are somewhat intertwined with the contents of `unknown.ps`. I recommend that you continue reading this document before reading `unknown.ps`. The material herein is based on:

- pages 170–171 of the revised edition *Analysis of Human Genetic Linkage* by Ott, or equivalently pages 184–185 of the third edition,
- Section 2.8 of Linkage Analysis Package II: User's Guide To Programs,
- Section 4 of "Avoiding Recomputation in Linkage Analysis" by A. A. Schäffer, S. K. Gupta, K. Shriram, and R. W. Cottingham Jr. (this is paper2.ps that comes with the FASTLINK distribution starting at version 2.0).
- Section 3 of "Faster Linkage Analysis Computations for Pedigrees with Loops or Unused Alleles" by A. A. Schäffer. (This is paper5.ps that comes with the FASTLINK distribution starting at version 3.0P).
- "Automatic Selection of Loop Breakers for Genetic Linkage Analysis" by A. Becker, D. Geiger, and A. A. Schäffer. (This is paper6.ps that comes with the FASTLINK distribution starting at version 4.0P).

It differs from these sources in that the presentation here gives more background and is more colloquial in nature. Thanks to Brian Nichols (U. Iowa) for asking enough questions about loops to motivate me to write this document. Thanks to Sandeep Gupta (Rice University) for helpful comments on my first draft. Thanks to Dylan Cooper for programming assistance with the loop algorithms first put in FASTLINK 3.0P. Thanks to Ann Becker and Dan Geiger for collaborating on the loop breaker selection code put in FASTLINK 4.0P and 4.1P.

Inbreeding Loops vs. Marriage Loops

After reading many papers with pedigree pictures, I have come to the conclusion that geneticists think it is important to distinguish between inbreeding (consanguinity) loops and marriage loops. For the purposes of pedigree traversal algorithms in LINKAGE this distinction is irrelevant. Simply put, a loop represents a cycle in the nuclear family graph defined in the accompanying document *Pedigree Traversal in FASTLINK*. Alternatively, a loop represents an undirected cycle in the undirected version of the marriage graph defined in paper6.ps.

Maxloop

LINKAGE and FASTLINK have a constant called maxloop. In order for the programs to run correctly, maxloop must be at least as large as the number of loops in any of the input pedigrees. There are no significant negative consequences for making maxloop larger than it needs to be when running sequentially, except that the preprocessor program MAKEPED has an output formatting bug on input pedigrees that have 9 or more loops (see the section on Encoding Loops in pedin.dat). The UNKNOWN program had a similar formatting bug, but it was fixed in FASTLINK 3.0P. FASTLINK 2.3P would not run in parallel for some looped pedigrees, especially with multiple loops.

In LINKAGE 5.1 and early versions of FASTLINK, maxloop is set to 2 in the distribution. More recent releases of FASTLINK have set maxloop at 8 or higher. Starting with FASTLINK, version 3.0P, the constant maxloop also appears in unknown.c. Rumor has it that some versions of LINKAGE have been distributed with maxloop set to 1. Some versions of LINKAGE have a bug that if the number of loops in the input pedigree is higher than maxloop, incorrect results are given, rather than an error message. The user is free to increase the value of maxloop.

There is no correctness harm in trying input pedigrees with arbitrarily large number of loops, so long as maxloop is set at least as large as the number of loops. Depending on the data set, the running time may increase exponentially with the number of loops. In FASTLINK 3.0P, the space requirements may also increase substantially; this was not true in earlier versions, but the use of extra space permits substantial time savings.

Counting and Breaking Loops

Through all versions of LINKAGE and FASTLINK 3.0P, the way to break a loop is to "clone" one of the participating individuals a into two people: one person acts as parent and the other acts as child and sibling. FASTLINK 4.0P introduces a more general cloning operation for pedigrees with multiple marriages. If a person participates in k marriages, it is permissible to make up to $k+1$ copies, one child copy and up to k parent copies. The number of copies is constrained by the requirement that the post-cloning pedigree must be connected. *Connected* means that there is a path of relationships (e.g., spouse, child, parent) connecting any two people. The cloning operation is illustrated in paper6.ps.

Through FASTLINK 3.0P, the number of loops and the number of people cloned were equated. The introduction of generalized cloning means that the number of people cloned is now less than or equal to the number of loops. In terms of the nuclear family graph the number of loops is the minimum number of edges whose removal from the nuclear family graph makes that graph into a tree (i.e. a connected graph with no cycles). When two of these edges are incident to two distinct nuclear families with a common parent (who married multiple times), then that common parent can be cloned into > 2 copies.

We call the individuals who are cloned, *loop breakers*. In FASTLINK 4.0P, it is still the case that the pedigree file input to unknown must have at most 2 copies per loop breaker. Within unknown, the choice of loop breakers may be adjusted, and it may be the case that the pedigree presented to the main program (ilink, linkmap, mlink) has > 2 copies of some loop breakers. In FASTLINK 4.1P, unknown can now accept pedigrees in which there are > 2 copies of some loop breakers. This is important for some non-human pedigrees, in which it is not possible to break all loops using at most 2 copies per loop breaker.

The selection of loop breakers can have a drastic impact on running time. Through FASTLINK 3.0P it was essential that the user be extremely clever in choosing the loop breakers. In FASTLINK 4.0P it was only necessary to choose a valid set of loop breakers. FASTLINK 4.1P provides the ability to choose a good set of loop breakers automatically, as follows.

1. Put the pedigree file in `pedfile.dat`.
2. Put the locus file in `datafile.dat`.
3. Call `unknown -l`.

A new pedigree file with loop breakers chosen will be printed to `lpedfile.dat`. The new pedigree file can then be used as a standard post-makeped pedigree file in input for more linkage analysis. A very important benefit is that it is no longer necessary to break loops in the `makped` preprocessor program. The user can now simply tell makeped that all pedigrees *have no loops*, even though that may be false, and then break the loops using `unknown -l`. In particular, the iterative, and interactive method of breaking loops using makeped and a program called LOOPS [Xie X, Ott J: Finding all loops in a pedigree. Am J Hum Genet 1992; 51:A205], as advocated on pages 93–96 of Handbook of Human Genetic Linkage Analysis by Terwilliger and Ott, is now *obsolete*.

The problem of selecting loop breakers cleverly is the topic of paper6.ps. This problem has deep connections to work in combinatorial optimization and probabilistic inference. FASTLINK 4.0P implements a solution to mathematical definition of the loop breaker selection problem. The problem is to minimize a mathematical quantity that should be well-correlated with computation time. The solution in FASTLINK 4.0P is provably optimal for pedigrees without multiple marriages, and shows especially good improvement (but is not necessarily optimal) on pedigrees with multiple marriages. FASTLINK 4.1 introduces a new method of loop breaker selection for looped pedigrees with multiple marriages. This method was developed by Reuven Bar-Yehuda, Ann Becker, and Dan Geiger, and is explained in paper7.ps.

Encoding Loops in pedin.dat

I have learned the hard way that the way loops are encoded in pedin.dat is not prominently explained in the LINKAGE documentation. Here is my attempt at a comprehensive explanation.

The presence or absence of loops is encoded in column 9 of pedin.dat. The entries in column 9 range from 0 to (1 + number of loops). For each loop the user must designate one individual to be the loop breaker. The user should then make two copies of the row of data for the loop breaker. All the genotype data in those two rows is the same, but the first columns are different. In one instance the loop breaker has no parents, but has children, and in the other the loop breaker has children but no parents.

The two copies of the loop breaker for the i^{th} loop must have the number i+1 in column 9 unless one of them is also the proband in which case one copy has a 1 and the other has i+1 in column 9. I.e., the two copies of the loop breaker for the first loop have a 2, the two copies of a loop breaker for the second loop have a 3, etc. In each pedigree the user may designate one non-loop breaker to be the "proband" by giving that person a 1 in column 9. Any other individuals who are not the proband and not loop breakers get a 0 in column 9. The setup of pedin.dat, including the designation of loop breakers, can be done semi-automatically with the MAKEPED program as described by Terwilliger and Ott. Beware of old versions of MAKEPED containing a bug that puts a 2 in column 9 for *every* copy of *every* loop breaker.

For example, two lines that start:

1 3 0 0 7 0 0 1 2 [Genotype data here]
1 4 1 2 0 5 5 1 2 [same genotype data here]

would encode a loop.

The 1 in column 1 indicates pedigree number 1. The 3 and 4 in column 2 indicate the two numbers assigned to the two copies of the loop breaker. This person is a male due to the 1 in column 8 and is a loop breaker due to the 2 in column 9. Individual number 3 acts as a parent; the 7 in column 5 indicates that individual 7 is the first child of the loop breaker. Individual number 4 acts as a child and sibling. The 1 in column 3 indicates that 1 is the father of the loop breaker. The 2 in column 4 indicates that 2 is the mother of the loop breaker. The 5 in columns 6 and 7 indicates that individual number 5 is the next paternal sibling and next maternal sibling of the loop breaker. The numbers assigned to the two copies of the loop breaker in column 2 need not be consecutive. The two lines for a loop breaker do not have to occur consecutively in pedin.dat. Since FASTLINK 4.0P uses multi-copy cloning it is possible to use more than two lines with the same number in column 9. In such a case, only one copy should be a child (i.e., have non-zero entries in columns 3 and 4).

Here is a tabular summary of what the columns encode:

Column 1: Pedigree number
Column 2: Individual number
Column 3: Number of father
Column 4: Number of mother
Column 5: First child
Column 6: Next sibling with same father
Column 7: Next sibling with same mother
Column 8: Male (encoded as 1) or female (encoded as 2)
Column 9: Neither proband nor loop breaker (0), proband (1), or loop breaker for loop i $(i+1)$

It is highly preferable to choose loop breakers whose genotypes are known. This is done automatically in FASTLINK 4.0P and beyond. This was not as apparent as it should have been in LINKAGE 5.1 or earlier versions of FASTLINK because there was an algorithmic inefficiency in handling loop breakers of known genotype. I have corrected this inefficiency in FASTLINK 3.0P and the correction is described in paper5.ps. Thanks to Jacques Beckmann for sending me a data set that brought the problem to my attention.

As a result of the changes in FASTLINK 3.0P, having more loop breakers of known genotype does not increase the running time or space usage too badly. What matters now is the number, u, of loop breakers of unknown genotype. It was always true that the running time increased exponentially as u increased. In FASTLINK 3.0P, the space usage also increases exponentially with u in return for substantial time savings. One can trade time for space by lowering the constant max_vectors_considered in unknown.c. When space is a problem, the code will output a diagnostic suggesting the next lower value of max_vectors_considered that would lead to a substantial space savings.

Loop Ordering

One of the things that the loop encoding system used to imply is that the user gets to choose the numbering of the loops. For example, suppose there are three loops. Then there will be three loop breaker pairs of individuals (in pedin.dat). The user can choose which pair gets 2 in column 9, which pair gets 3 in column 9, and which pair gets 4 in column 9. The programs should give the same answer no matter which loop ordering is chosen.

However, in FASTLINK 2.0, the first loop, whose loop breakers get a 2 in column 9, is treated specially. What this means for the user is that the choice of which loop gets encoded as 2 can have a drastic effect on running time, but not correctness of the results.

FASTLINK 4.0P automatically reorders the loop breakers according to a heuristic that should give the best running time in most cases. The heuristic is to put the loop breaker with the most possible genotypes first (number 2 in column 9), and then sort the rest from fewest possible genotypes to most possible.

Error Detection for Looped Pedigrees

There are three different aspects of error detection in looped pedigrees: non-Mendelian inheritance, too few loop breakers, too many loop breakers.

The preprocessor program UNKNOWN (See unknown.ps) is supposed to detect violations of Mendelian rules in input pedigrees. Until a recent posting on bionet.molbio.gene-linkage by Lucien Bachner, it was a well-kept secret that UNKNOWN did no genotype inference or error detection for pedigrees with loops. I have corrected this deficiency in FASTLINK 3.0P. When a looped pedigree contains an instance of non-Mendelian inheritance, UNKNOWN will simply report the locus where there is a problem. If one then removes the loops by zeroing out all entries > 1 in column 9 of pedin.dat, one can rerun UNKNOWN with a loopless pedigree. With loopless pedigrees, UNKNOWN gives more information about the vicinity of the non-Mendelian inheritance.

LINKAGE and early versions of FASTLINK contained a flaw that they would allow the input of looped pedigrees in which the selection of loop breakers would be insufficient to break all the loops. In most cases including all pedigrees without multiple marriages, UNKNOWN would go into an infinite loop. This infinite loop was detected with a diagnostic starting in FASTLINK 2.3P.

Ken Morgan pointed out that for pedigrees with multiple marriages it was possible to have an insufficient set of loop breakers and avoid the infinite loop. Therefore, in FASTLINK 3.0P I added a diagnostic that I believe catches all unbroken pedigree loops.

LINKAGE and all versions of FASTLINK, through 3.0P contained a flaw in that they accepted some pedigrees with too many loop breakers, causing the pedigree to be disconnected. The results (i.e., likelihoods and lod scores) for disconnected pedigrees would often, but not always, be sufficiently implausible as to indicate a serious, unexplained problem in the input. This problem is corrected implicitly, without a diagnostic, in FASTLINK 4.0P.

Loop Algorithm

The basic algorithm for handling loops in LINKAGE is described in Ott's book, pages 170–171. We describe it here using more algorithmic language. As described in the accompanying document on *Pedigree Traversal in FASTLINK*, the basic goal is to compute for each individual x and each genotype g the conditional probability $P(x, g)$ that x has genotype g. This probability is conditioned on the parts of the pedigree traversed so far and the candidate θ.

The probability that x has genotype g may depend on the genotype of the loop breaker. Because the loop breaker is treated as two different people, we must force those two people to have the same genotype during a given traversal. This means that we must re-traverse the pedigree for each possible genotype of the loop breaker and sum the probabilities for each traversal. For example, suppose the loop breaker may have any of three genotypes $\{1, 2, 3\}$. Then for $i = 1, 2, 3$ we compute the conditional probability $P_i(x, g)$ that x has genotype g where we condition on the additional assumption that the loop breaker has genotype i. Then $P(x, g)$ would be computed as $(P_1(x, g) * B(1)) + (P_2(x, g) * (B_2)) + (P_3(x, g) * B_3)$, where $B(i)$ is the probability that the loop breaker has genotype i.

Since each possible genotype of the loop breaker requires another complete traversal of the loop, it is highly desirable to choose loop breakers that have the fewest number of possible genotypes. When there are multiple loops, the algorithm must do one traversal for each vector of possible genotypes of all the loop breakers. I am calling these *loop-breaker vectors*. For example, if the first loop breaker has 3 possible genotypes and the second loop breaker has 4 possible genotypes, there are $3 \times 4 = 12$ loop-breaker vectors and 12 traversals of the pedigree will be done for each likelihood evaluation. This explains why it may be impractical to handle pedigrees with more than 2 loops, which in turn explains why LINKAGE was distributed with maxloop set to 2. The speed of FASTLINK has emboldened some users to try more complicated pedigrees, so we have boosted the default setting to 3.

In FASTLINK 2.0, I introduced the following optimization. Those parts of the pedigree whose conditional genotype probabilities are independent of the genotype of the loop breaker for the first loop (labeled as 2 in column 9 of pedin.dat) are only traversed once. In LINKAGE these parts are traversed once for each possible genotype of the loop breaker.

In FASTLINK 3.0P, I introduced two optimizations. First in the case of loop breakers of known genotypes, the new code iterates much more efficiently over the possible loop breaker vectors. Second, UNKNOWN now does genotype inference for looped pedigrees. As a consequence, it figures out which vectors of loop breaker genotypes are not consistent with the rest of the pedigree. It can also sharply reduce the set of possible genotypes for each individual, for each loop breaker vector.

FASTLINK 4.0P includes a sophisticated algorithm to try to minimize the number of loop breaker vectors. The algorithm was devised by Ann Becker and Dan Geiger of the Technion, and integrated by me into FASTLINK.

FASTLINK 4.1P includes a slightly better implementation of the ideas introduced in 3.0P. It also includes changes to UNKNOWN to select loop breakers from scratch.

The genotype inference information is printed out in the file `loopfile.dat`, which is read in and used by the main program (ILINK, MLINK, or LINKMAP). The exact syntax of `loopfile.dat` is described in `README.loopfile`. Some of the changes in FASTLINK 4.1P can make loopfile.dat a lot smaller than it used to be for pedigrees with multiple loops.