The VITESSE algorithm for rapid exact multilocus linkage analysis via genotype set-recoding and fuzzy inheritance

Jeffrey R. O’Connell¹ & Daniel E. Weeks¹,²

As genetic marker maps have improved, multipoint linkage analysis has become a crucial part of all disease mapping studies. Paradoxically, multipoint lod scores become increasingly difficult to compute, particularly as the numbers of markers, marker alleles and untyped people increase. We have solved this problem by using a novel set-recoding scheme to recode each person’s genotype and ‘fuzzy inheritance’ to infer transmission probabilities. Our approach is implemented in a memory-efficient computer program, VITESSE, for extremely rapid computation of exact multipoint likelihoods. VITESSE enables fast and precise multipoint mapping of disease loci with highly polymorphic markers.

Linkage analysis of genetic data has been used successfully to map disease genes involved in a large number of traits. The power of the genetic mapping approach has dramatically improved because of the development of high-resolution genetic maps of highly-polymorphic markers spanning the genome. As the maps have improved, multipoint linkage analysis has become an essential part of any linkage-based disease mapping study because multipoint analyses are much more powerful for detecting linkage and ordering loci than pairwise analyses. In addition, multipoint likelihood computations are crucial for haplotyping programs, simulation methods, and application of regressive approaches to multilocus data. Paradoxically, as the maps have improved, multipoint likelihoods have become increasingly difficult to compute due to memory and time constraints in the existing programs. These programs are based on an algorithm which essentially carries out summations over all possible genotype vectors. This approach has difficulties because any person who is untyped at several very polymorphic markers will have an astronomical number of possible genotypes. This is particularly problematic when the untyped person is a founder (i.e., a person with no parents in the pedigree), since his/her genotypes are not constrained by any parental genotypes. Therefore, existing linkage analysis programs, which were written well before the advent of highly polymorphic markers, fail to run as the number of untyped, markers and marker alleles increase. (Note that these computational problems have been essentially solved for small inbred pedigrees and for nuclear families with grandparents by application of special techniques that take advantage of these limited family structures.)

Computation of multipoint likelihood curves for large general pedigrees containing untyped individuals is so important that investigators have turned to approximation techniques. These techniques include (i) computing overlapping lod score curves with much smaller numbers of loci than desired; (ii) accelerating the computations by discarding partially informative portions of the family data; (iii) constructing ad hoc approximations to the multipoint curves based on pairwise lod score curves; (iv) splitting a larger pedigree into several smaller ones; and (v) using simulation-based approaches to construct an average lod score curve. The first four techniques are unsatisfactory since they fail to extract the maximum amount of information from the data which are often gathered at great cost; the last technique is unsatisfactory because it can be difficult to ascertain if the computationally-intensive algorithms have converged adequately.

Another complementary approach to accelerating multipoint likelihood computations involving highly polymorphic markers is to recode alleles to reduce the number of alleles at each locus without altering the lod score. This approach has arisen because present-day linkage analysis programs have severe memory bottlenecks that are a function of the product of the number of alleles at each locus. Thus, a reduction in the number of alleles may circumvent the memory bottleneck. There have been at least five different approaches to allele-recoding:

(i) All the alleles that do not appear in a pedigree may be lumped into a single allele, whose frequency is the sum of the missing alleles, thus reducing the number of alleles to one more than the number that appear.

(ii) When everyone is typed, the lod score may be
computed correctly using only 4 alleles by recycling allele labels. This works because the lod score is independent of the allele frequencies when every founder's genotype is known. A graph-algorithm for recoding alleles was developed by Braverman; this algorithm alters the likelihood if there are untyped founders. Braverman (personal communication) recently extended his algorithm to enable exact likelihood calculations in the presence of untyped founders. However, this approach usually does not adequately reduce the number of alleles.

(iv) A common practice for simple diseases is to make only disease alleles informative, recycling allele labels as needed. This practice will not work for complex diseases, and may be biased in favor of linkage for ‘simple’ diseases.

(v) A founder with one typed offspring can be made homozygous for the allele transmitted to the one typed child (if it can be inferred which allele is transmitted). This local allele recoding, when applicable, can drastically reduce the number of possible genotypes for the founder. This recoding approach is not applicable for multipoint analysis if there is linkage disequilibrium.

All of these allele-recoding approaches are aimed at accelerating the lod score computations by reducing the number of alleles. Due to memory and time constraints of existing programs, multipoint analyses may still be infeasible even after allele recoding.

We have developed a novel algorithm for recoding genotypes that effectively addresses both the memory and computational limitations encountered in the presence of untyped founders and large numbers of marker alleles. Our new algorithm is implemented in a very fast memory-efficient computer program, VITESSE, that computes two-point and multipoint likelihoods for simple pedigrees without loops where all people are descended from one pair of founders. Our algorithms are valid for general pedigrees with loops and VITESSE will handle such pedigrees soon. It is our hope that VITESSE will be a useful tool for fast and precise multipoint mapping of disease loci. VITESSE should permit one to extract much more linkage information from the data via multipoint analyses that were previously either impossible or too time-consuming.

The VITESSE algorithm

The main ideas of VITESSE's new algorithm involve how multilocus genotypes are efficiently stored in memory and how these genotypes are recoded to reduce their numbers. As mentioned above, multilocus likelihood computations involve massive summations over all possible genotype vectors for everyone in the pedigree. In the Elston-Stewart algorithm, these summations are made manageable by viewing the pedigree as consisting of a collection of nuclear families (parents and children) joined together; the overall summation is carried out by successively ‘peeling’ off nuclear families from the periphery of the pedigree (see Methods). VITESSE obtains its performance increase over previous programs by a more memory-efficient approach to representing multilocus genotypes and by dramatically reducing the number of possible genotype vectors by recoding individual's genotypes into sets of transmitted and non-transmitted alleles. Throughout the following discussion, we compare the performance of VITESSE to FASTLINK, currently the fastest linkage program for general pedigrees. FASTLINK is an accelerated version of the LINKAGE programs, which were originally developed by Lathrop and colleagues. FASTLINK comes in a ‘fast’ version which is much more memory-intensive than the ‘slow’ version; we used the fast version of FASTLINK when possible.

Genotype representation

There are two approaches to storing multilocus genotype information. The first approach, used by FASTLINK, is to allocate a matrix with dimension M(M+1)/2 for each person to represent all possible genotypes, and then keep track of which genotypes are valid (that is, consistent with the observed phenotypic information) during the likelihood calculation, where M is the product of the number of alleles. The second approach is to store only single-locus lists, building valid multilocus genotypes as needed from the lists. These two approaches have radically different memory requirements. For example, 5 ten-allele loci would require storage of only 275 single-locus genotypes (55 per locus). In contrast, the same 5 ten-allele loci (M = 100,000) would require storage of 5,000,050,000 multilocus genotypes. By using set-recoding and other algorithmic improvements, we have implemented the second approach without any sacrifice in time performance. Our implementation is based on considering, for each nuclear family in a pedigree, only those pairs of parental genotypes that are consistent with the observed phenotypes. Invalid parental pairs are never considered, in contrast to other programs. To provide a measure of the computational difficulty of each nuclear family encountered, VITESSE prints out the number of valid parental genotype pairs, which is simply the product of the number of pairs at each locus.

Transmitted and non-transmitted sets

In addition to using memory efficiently, VITESSE also accelerates summations over multilocus genotypes via a novel set-recoding scheme. This scheme recodes a person's genotype list at each untyped marker to minimize the size of his/her genotype list. We developed our recoding algorithm by determining, for each untyped person, which alleles have identical roles as recombination indicators. Sets of alleles which have identical roles can then be combined into a single representative allele. Alleles play two different roles in the Elston-Stewart algorithm:

(i) Alleles determine recombination events. Recombination events in a child's haplotype are identified by examining the origin of the child's alleles in an ordered parental genotype. (An 'ordered' genotype indicates the maternal and paternal origin of the alleles by the order of the alleles.) Here it is important to know whether or not the child's allele matches the parental alleles, but the specific label of the allele per se does not matter.

(ii) Alleles determine prior probabilities of observing a particular genotype in the founders. Founders are assumed to represent random people from the population, so, assuming Hardy-Weinberg and linkage equi-
Box 1 provides a precise definition and an example of transmitted and non-transmitted sets.

**Set recoding**

VITESSE dramatically accelerates likelihood computations by minimizing the number of genotypes via the following set recoding scheme: Replace each maternal non-transmitted allele by the set \( \emptyset \), and each paternal non-transmitted allele by the set \( \emptyset \). Replace each transmitted allele \( T \) by the set \( T \) (since transmitted alleles must be distinguishable). Then delete any duplicate genotypes. For example, consider the ordered genotype list \([12, 13, 14, 15]\), with paternal alleles on the left and maternal alleles on the right. If \( P = [1], P = \emptyset, M = [4, 5], \) and \( M = [2, 3] \), then the set-recoded list would be \([1]T, [2, 3], [1]T, [2, 3], [1]T, [4], [1]T[5]\), which contains only 3 pairs. Note that shorter genotype lists lead to significant computational savings. For example, 10,000 multilocus genotypes are required to handle 4 loci with 10 genotypes each, whereas only 6,561 multilocus genotypes are required for 4 loci with 9 genotypes each—a 34% reduction. We apply set-recoding twice, once before genotype elimination and once after, which accelerates genotype elimination by reducing the size of genotype lists. Genotype elimination algorithms \(^{13,15}\) eliminate any genotypes that are incompatible with the observed phenotypes, and have a crucial role in efficient likelihood computations. Note that set-recoding includes both techniques of global allele lumping and 'homogeneous recoding' as special cases (techniques (i) and (v) in the introduction). In set-recoding, any allele not appearing in the pedigree is non-transmitted and is absorbed into each person's non-transmitted set. Also, any untyped person with only untyped descendants becomes homogeneous while still maintaining the likelihood.

**Fuzzy inheritance**

If allele-sets are used, then the procedure for inferring inheritance relations must be redefined (see Methods). Normally, inferences about inheritance are based on allele equality — if a parent has the genotype \( A/B \) and its child has allele \( C \), then \( C \) is inherited from the parent if \( A = C \) or \( B = C \). However, in set-recoding, \( A \) and \( B \) are sets, so instead of allele equality we use set inclusion — \( C \) is inherited from the parent if \( A \subset C \) or \( B \subset C \). We call this 'fuzzy inheritance', as it is similar to fuzzy logic\(^{14}\). For example, consider the pedigree in Fig. 1b: The child has genotype \([1, 2, 3], [4, 5]\) and its mother has genotype \([6], [4, 5]\), so the mother contributed her maternal allele, but the father is homogeneous, as he could have contributed either \( [1] \) or \( [2, 3] \) to the child. Set-recoding of typed loci \( [A/B] = [A][B] \) permits use of fuzzy inheritance throughout.

**Timing tests**

We compared the performance of VITESSE to FASTLINK by computing multipoint likelihoods on the five representative data sets described below. We report the number of pairs of parental genotypes with and without set-recoding. This number provides a measure of the complexity of the problem (ignoring the number of children the parents have) and indicates how much redundant computation is avoided by set-recoding. As
Table 1 Performance of VITESSE and FASTLINK on the example data sets

<table>
<thead>
<tr>
<th>Number of loci</th>
<th>Number of alleles</th>
<th>Product of number of alleles</th>
<th>Timea</th>
<th>Memoryb</th>
<th>Timea</th>
<th>Memoryb</th>
<th>Number of parental pairs with set-recoding</th>
<th>Number of parental pairs without set-recoding</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2-2-2-2-2-2</td>
<td>32</td>
<td>2</td>
<td>slow</td>
<td>3 m 54 s</td>
<td>2</td>
<td>461,370</td>
<td>613,162</td>
<td>1.4</td>
</tr>
<tr>
<td>5</td>
<td>2-2-2-2-2-13</td>
<td>208</td>
<td>3</td>
<td></td>
<td>30 h 18 m</td>
<td>7</td>
<td>3,504,738</td>
<td>118,329,361</td>
<td>102.4</td>
</tr>
<tr>
<td>5</td>
<td>2-2-2-2-2-2</td>
<td>64</td>
<td>4</td>
<td></td>
<td>1 h 2 m</td>
<td>3</td>
<td>3,095,104</td>
<td>3,871,856</td>
<td>1.7</td>
</tr>
<tr>
<td>Example B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2-5-8</td>
<td>80</td>
<td>3</td>
<td>fast</td>
<td>15 m 42 s</td>
<td>3</td>
<td>1,861,659</td>
<td>10,667,145</td>
<td>5.1</td>
</tr>
<tr>
<td>4</td>
<td>2-5-3-5-8</td>
<td>240</td>
<td>7</td>
<td></td>
<td>34 h 24 m</td>
<td>25</td>
<td>62,714,859</td>
<td>357,741,909</td>
<td>3.3</td>
</tr>
<tr>
<td>Example C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2-18-23</td>
<td>828</td>
<td>2</td>
<td>fast</td>
<td>5 h 58 m</td>
<td>100</td>
<td>31,486</td>
<td>160,000,000,000</td>
<td>1022.9</td>
</tr>
<tr>
<td>2</td>
<td>2-17-18-23</td>
<td>14,076</td>
<td>2</td>
<td></td>
<td>-</td>
<td>-</td>
<td>189,528</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12-12-15-17-18-23</td>
<td>15,202,080</td>
<td>7</td>
<td></td>
<td>-</td>
<td>-</td>
<td>3,165,459</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2-6-12-6</td>
<td>864</td>
<td>3</td>
<td>fast</td>
<td>7 h 48 m</td>
<td>22</td>
<td>3,444</td>
<td>28,864,348</td>
<td>7020.0</td>
</tr>
<tr>
<td>5</td>
<td>2-5-6-12-6</td>
<td>4320</td>
<td>4</td>
<td></td>
<td>-</td>
<td>-</td>
<td>37,472</td>
<td>129,143,216</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2-5-2-6-3-3-2</td>
<td>2160</td>
<td>22</td>
<td></td>
<td>-</td>
<td>-</td>
<td>26,914,944</td>
<td>73,671,712</td>
<td></td>
</tr>
<tr>
<td>Example E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2-9-9-9</td>
<td>1458</td>
<td>1</td>
<td>slow</td>
<td>6 m</td>
<td>220</td>
<td>37</td>
<td>1394</td>
<td>360.0</td>
</tr>
<tr>
<td>5</td>
<td>2-9-9-9-9</td>
<td>91,854</td>
<td>1</td>
<td></td>
<td>-</td>
<td>-</td>
<td>129</td>
<td>12,562</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2-9-9-9-9-7-7-6-5-5</td>
<td>96,448,700</td>
<td>62</td>
<td></td>
<td>-</td>
<td>-</td>
<td>4,909</td>
<td>65,600,405</td>
<td></td>
</tr>
</tbody>
</table>

Multipoint likelihoods were computed for six positions of the trait locus in a fixed marker map. A dash indicates that FASTLINK could not be run due to memory (or computational) constraints.

The time is the average of user + system time over two timing tests. Both VITESSE and FASTLINK were compiled using gcc with the -O optimization option. All timings were done on a SUN SPARCstation 514MP with 128 Mb RAM and 370 Mb of swap space. These tests involved calculating a multipoint likelihood for 6 different positions of the trait locus in a fixed marker map. Note that for the FASTLINK runs we made, everyone in a given pedigree was homozygous 1/1 if everyone was untyped.

Not listed because genotype elimination without set-recoding was too time-consuming.

The timing results (Table 1) indicate, VITESSE is much faster than FASTLINK and can handle problems well beyond the memory constraints of FASTLINK. Note that in all of these examples except Example C, conventional allele lumping was not possible because each allele appeared at least once in the data.

Example A. The first example consists of 32 Alzheimer pedigrees containing 451 people in 96 nuclear families. Carrying out linkage analyses on Alzheimer’s pedigrees has been notoriously difficult, due to the late onset of the disease — most pedigrees consist of several genotyped sibships connected via a large number of untyped individuals. This example illustrates two points regarding VITESSE: (i) Even in a situation where set-recoding has no effect (all loci have 2 alleles), VITESSE is faster than FASTLINK to begin with; (ii) FASTLINK is extremely sensitive to the number of marker alleles. Replacing a single 2-allele marker with a 13-allele marker increases FASTLINK’s run time by a factor of 466 and its memory requirements about 3.5 times, while the factors for VITESSE are only 6 and ~1.5, respectively. It is precisely this problem that has caused great difficulties for linkage analysts — for example, when the Alzheimer’s disease locus in the Volga Germans was recently mapped to chromosome 1, the authors stated “multiple linkage analysis was not performed because of the extraordinary computer time necessary with traditional methods.”

Example B. VITESSE is able to handle large numbers of untyped generations with ease, as indicated by this example, where it ran the four-point example ~3.3 times faster with ~3.6 times less memory than the fast version of FASTLINK. These examples come from a recent linkage study which mapped a locus for an autosomal dominant trait involved in conduction system disease and dilated cardiomyopathy to chromosome 1. This example also illustrates the value of multipoint analyses, as the lod score was almost doubled by using multipoint analyses versus two-point analyses: The largest of the two-point lod scores for several chromosome 1 markers was 6.28. However, computation of exact multipoint lod scores became virtually impossible as the number of markers increased, since the pedigree, which contains 132 people in 39 nuclear families over 7 generations (Fig. 2), has the first four generations untyped. Kass et al. then used a simulation technique known as sequential imputation to generate an approximate 7-point multipoint lod score curve, with a maximum multipoint lod score of 13.2. Thus, the multipoint analysis increased the odds in favour of linkage from ~106 to ~1013. Also, the multipoint analyses provided a narrow support interval for the best location of the disease locus.
Example C. Previous linkage programs, such as LINKAGE and FASTLINK, are simply unable to handle many highly polymorphic markers. However, VITSESSE can easily handle large numbers of highly polymorphic markers, as illustrated by this example. These data consist of 91 pedigrees segregating for rheumatoid arthritis containing 482 people in 95 nuclear families typed for 12 highly polymorphic markers, with 8 to 23 alleles; all parents are untyped (ref. 40 and P. Wordsworth, pers. comm.). Note that conventional allele lumping would barely help on this example, as it reduces the 15- and 17-allele marker by only one allele each. Also, the reduction of parental pairs from 160 billion to 31 thousand shows how efficient VITSESSE's set-recoding is in eliminating massive amounts of redundant computations while maintaining the exact likelihood. FASTLINK was able to do this problem because it makes everyone in a given family homozygous 1/1 for any marker for which everyone in the family is untyped. However, this alters the likelihood (but not the lod score).

Example D. VITSESSE's performance is mainly a function of the number of parental genotype pairs in the pedigree, while FASTLINK's performance depends mainly on the product of the number of alleles. This is illustrated by the timing results on this large pedigree segregating for maturity onset diabetes of the young (MODY) with the first generation untyped (Fig. 3). The pedigree contains 155 people in 35 nuclear families over 5 generations. Multipoint likelihood computations on this pedigree are so difficult that this pedigree was used to illustrate the sequential imputation method. On the 4-point computation, VITSESSE is more than seven thousand times faster than the fast version of FASTLINK. VITSESSE achieves this excellent speed-up because set-recoding reduces the number of parental pairs from 29 million to 3,444 pairs. However, for the 7-point problem, where the product of the number of alleles is only 2,160, takes VITSESSE 64 hours because the number of parental pairs after set-recoding is 27 million.

Discussion

Efficient computational algorithms are crucial to successful linkage analysis, and will become even more important as genetics attempts to map disease-susceptibility genes involved in complex traits. However, as the numbers and polymorphism of the markers have increased, current linkage analysis programs have reached their limits, making it difficult or impossible to generate exact multipoint lod score curves for many disease studies. This is unfortunate, since multipoint analyses are much more efficient for detecting linkage than two-point analyses. Also, multipoint analyses can also provide support intervals for the best location of the trait locus — these intervals have a critical role in the search for candidate loci and in designing physical mapping strategies. Our algorithmic developments, as implemented in VITSESSE, compute likelihoods exactly and quickly on large complex pedigrees by using novel set-recoding techniques to reduce the number of genotypes considered. VITSESSE is able to handle up to 10 highly polymorphic markers on pedigrees with many untyped individuals, and should greatly improve the linkage analyses that can be carried out in a reasonable amount of time.

VITSESSE, via its automated set-recoding algorithms, will greatly simplify and accelerate the often arduous task of carrying out multipoint likelihood analyses. For example, when Nygaard et al. mapped the dopa-responsive dystonia (DRD) gene to chromosome 14 two years ago with a maximum two-point lod score of 4.38, the multipoint analyses were very difficult.
since all the markers in the region of interest were highly polymorphic, and one, D14S77, had 20 alleles. They were forced to recode the alleles, losing allele frequency information, and so the multipoint lod score curves were no longer exact. Even after allele recoding, D14S77 still had 9 alleles, and so only 4-point analyses were possible with LINKAGE. They then combined several overlapping 4-point curves analyses, which yielded a flat multipoint lod score curve with a maximum around 6.3 and two intervals equally likely to contain the disease gene. Now with VITESSE we have easily computed exact 6-point curves, increasing the maximum multipoint lod score to 7.2 and indicating that one of the intervals is 10 times more likely than the other to contain the disease gene. (The DRD locus has been cloned\cite{13}, but not yet mapped relative to our genetic markers). Thus, not only does VITESSE provide faster computations, but also leads to better mapping.

We are currently implementing our algorithms to handle general pedigrees with loops in VITESSE. We expect excellent results for loops because set-recording reduces the number of genotypes of an untyped loop-breaker, which in turn reduces the number of times the pedigree must be traversed. We expect VITESSE will have a strong impact on these types of pedigrees, where currently even two-point analyses can be so prohibitive that loops must be ignored, thus losing valuable information. We are also interested in incorporating VITESSE as a likelihood engine into several other genetic analysis programs, such as the simulation program SLINK\cite{14} and the analysis program REGRESS\cite{15,16}.

To summarize the philosophy behind VITESSE, there are two important questions regarding the usefulness of a computer program: (i) Do we have enough memory to run it? and (ii) Do we have enough time to run it? The former is limited by physical constraints, and the latter only by our patience. Thus, these questions are sequential: if the answer to the first is no, then the second question is meaningless. VITESSE has, for the most part, eliminated the memory bottleneck. We believe this is the beginning of a new generation of space- and time-efficient algorithms for computing multipoint likelihoods.

The VITESSE program may be obtained by anonymous ftp to watson.hgen.pitt.edu.

**Methods**

**Computation of exact multipoint likelihoods.** To extract the maximum amount of information from the linkage data, it is crucial to be able to compute the exact multipoint likelihoods, rather than approximate them. We provide a proof that VITESSE computes the exact multipoint likelihood. To do this, it is first necessary to briefly review the Elston-Stewart algorithm\cite{17} for computing the likelihood of a pedigree.

**The Elston-Stewart algorithm.** Given a pedigree containing \(n\) members, let \(x_i\) have observed phenotype \(x_i x_i\) and \(G_i\) be a vector of observed genotypes of individuals and \(G_i\) be the set of observed genotypes compatible with person \(i\)'s phenotype. (Genotypes may be unordered or ordered: an ordered genotype indicates the maternal and paternal origin of each allele.) The Elston-Stewart algorithm computes the likelihood via the sum of products:

\[
P(I|x_i G_i) = \sum_{i} \sum_{P(x_i | G_i)} \prod_{j \neq i} P_{P(x_i | G_i)} \prod_{j \neq i} \prod_{k \neq i} \prod_{j \neq k} \prod_{j \neq k} \prod_{k \neq l} \prod_{j \neq k \neq l} P_{P(x_i | G_i)}
\]

(Eq. 1), where \(P_{P(x_i | G_i)}\) is an offspring-parent triple and \(P_{P(x_i | G_i)}\) in the penetrance, which is the probability of individual \(i\) having phenotype \(x_i\) given genotype \(G_i\). \(P_{P(x_i | G_i)}\) is the prior probability of founder \(k\)'s genotype based on population gene frequencies; and \(P_{P(x_i | G_i)}\) is the transmission probability of the child's genotype \(G_i\) given paternal genotype \(G_p\) and maternal genotype \(G_m\). If we used ordered genotypes, then the term \(P_{P(x_i | G_i)}\) may be factored as \(P_{P(x_i | G_i)} = P_h(G_p) P_h(G_m)\), where \(P_h(G_p)\) is the transmission probability of the child's maternal haplotype and \(P_h(G_m)\) is the transmission probability of the child's paternal haplotype. Optimal implementation of the Elston and Stewart algorithm\cite{17} for simple pedigrees involves clipping off or peeling one nuclear family at a time\cite{13,16}. This is equivalent to changing Eq. 1 from a global summation over all the products to a nested summation by moving summation signs into the product as far as possible. A nuclear family is clipped by choosing one person in the family to update conditional on the probabilities of the other members.

**Maintaining the likelihood.** Set-recoding groups together non-transmitted alleles, replacing large summations in Eq. 1 by smaller summations over the grouped terms. The likelihood will not be altered if these summations are the same. First, we show that we can easily redefine the prior probability so that using fuzzy inheritance once poses no problems. Second, we prove that using transmission probabilities generated by fuzzy inheritance does not alter the likelihood. However, it is first necessary to define some terminology: We use the term "genotype" to mean multilocus genotype. When we refer to only one marker at a time, we assume that the remaining loci are fixed and identical for all genotypes.

To preserve the likelihood when using fuzzy inheritance it is necessary to handle the prior probabilities (which involve the allele frequencies) in a special manner. Recall that prior probabilities are only assigned to founders. For founders, we define the frequency of an allele set as the sum of the frequencies of each allele in the set, which works because untyped loci in founders have symmetric genotype lists if an untyped founder has the set-recorded genotype \(AB\), then his/her unrecorded genotype list must contain all genotype pairs from the Cartesian product of \(A\) and \(B\). For example, a set-recorded genotype \([1,2][1,2]\) corresponds to the unrecorded genotype list: \(11, 12, 21, 22\). Thus, the sum of the prior frequencies \(F_1 = F(1) + F(1) F(2) F(2) F(1) + F(1) F(1) F(2) F(2) F(1)\) equals \(F(1) F(1) F(2) F(1) F(1) F(2)\), the prior probability of the set-recorded genotype (where \(F(1) = \text{allele frequency of } a\)).

We now show that using transmission probabilities generated by fuzzy inheritance does not alter the likelihood when summing over set-recoded genotypes. When we peel down to a person \(P\), we compute for each genotype a phenotype conditioned on \(P\)'s ascendants, which we call the descendant conditional probability \(\Psi\). Similarly, when we peel up to a person \(P\) we compute for each genotype a phenotype conditioned on \(P\)'s descendants, which we call the descendant conditional probability \(\Psi\). If several genotypes (that differ only by non-transmitted allele) have the same descendant probabilities, then during the likelihood computation, we need only compute one of them. Set-recoding efficiently determines which subsets of genotypes have identical descendant conditional probabilities. Note that our proof below does not require the assumption of no interference and so VITESSE could easily handle different models of interference as well as no interference.

**Proposition:** If a person has genotypes \(G1\) and \(G2\) which differ only at one locus by non-transmitted alleles, then \(G1\) and \(G2\) have the same descendant conditional probability, i.e., \(\Psi(G1) = \Psi(G2)\).

**Proof by induction:** From our previous discussion it suffices to prove the proposition for a single line of descent (Note that a person's non-transmitted alleles are either passed on to or not at all). We may assume there is only one child in the line of descent because each child's contribution is conditionally independent. Without loss of generality, assume that there are only two non-transmitted alleles and one transmitted allele, so \(G1 = NG1\) and \(G2 = NG2\). The base case for our induction is a descendant of P who does not pass on \(N1\) and \(N2\). In the induction step, we assume the proposition is true for each child of one of \(P\)'s descendants, say \(P\), and then prove it remains true for \(P\) after peeling up to \(P\).
We make the following assumptions: P, P1, P2, D1, and D2 are untyped, the people connecting P to P2 are untyped, N1 and N2 are non-transmitted in paternal phase, C is an arbitrary allele in maternal phase, and D does not pass on N1 and N2. Note that M1, M2, and P are not necessarily founders. 

Base case: Consider D, who is the first descendent of P who does not pass on N1 and N2. Since D only passes on A, both genotypes have the same descendant conditional probabilities:

\[ \Psi(N1A) = \Psi(N2A) \]

Induction step: Assume the proposition is true for D. We want to show that after we peel up to P1, we have \( \Psi(N1B) = \Psi(N2B) \). There are two cases to consider:

1. B is neither N1 nor N2:
   \[ \Psi(N1B) = \sum \text{Trans}(N1A | N1B, \beta_3 | N1) \Psi(N1A) \]
   \[ \Psi(N2B) = \sum \text{Trans}(N2A | N2B, \beta_3 | N1) \Psi(N2A) \]
   And so \( \Psi(N1B) = \Psi(N2B) \).

2. B is say, N1. Then the two genotypes of P1 are N1N1 and N1N2:
   \[ \Psi(N1B) = \sum \text{Trans}(N1A | N1N1, \beta_1 | N1) \Psi(N1A) \]
   \[ \Psi(N2B) = \sum \text{Trans}(N2A | N1N2, \beta_1 | N1) \Psi(N2A) \]

\[ \Psi(N2B) = \sum \text{Trans}(N1A | N1N2, \beta_1 | N1) \Psi(N1A) \]

Thus, since under the induction hypothesis \( \Psi(N1A) = \Psi(N2A) \), we have \( \Psi(N1B) = \Psi(N2B) \). This concludes the induction step. Thus, for person P, G1 and G2 are set-rectrode to the single genotype \( G_2 = (N1, N2)(C) \) and \( \Psi(G_2) = \Psi(G_1) = \Psi(G_2) \).

Using a similar argument, it can be shown that the descendant conditional probability \( \Omega \) satisfies the relation:

\[ \Psi(G_2) = \Psi(G_1) \]

Thus, peeling down P to the likelihood after set-rectrode in:

\[ \Psi(G_2) = \Psi(G_1) \]

which is the likelihood before set-rectrode. Thus, set-rectrode combined with fuzzy inheritance preserves the likelihood.

Acknowledgments

We thank P. Persoak-Vance, L. Jorde, N. Cox, P. Wordsworth, A. Kong, and P. Knutsen for letting us use their pedigree data; and M. Lathrop, J. Tervellinger, T.C. Mattise, A. Shriver-Moore, T.C. Young, and three anonymous reviewers for their comments. The pedigrees drawings were generated with Pedigree/Draw. This work was supported in part by funds from NIH grant HG00932, Association Francaise Contre Les Myopathies (AFM), the University of Pittsburgh, the Wellcome Trust Centre for Human Genetics at the University of Oxford, and the W.M. Keck Center for Advanced Training in Computational Biology at the University of Pittsburgh, Carnegie Mellon University, and the Pittsburgh Supercomputing Center.

Received 26 May; accepted 10 October 1995.