Linkage Analysis Based on Multi Terminal Binary Decision Diagrams: ALLEGRO version 2

Daniel F. Gudbjartsson

Decode Genetics, Reykjavik, Iceland

dfg@decode.is

Thorvaldur A. Thorvaldsson

Decode Genetics, Reykjavik, Iceland

Gunnar Gunnarsson[1]

Decode Genetics, Reykjavik, Iceland

and

Anna Ingolfsdottir[2]

Decode Genetics, Reykjavik, Iceland

Received _____; accepted _____

[1]University of California, Santa Barbara, CA, USA

[2]Basic Research in Computer Science, Centre of the Danish National Research Foundation, Department of Computer Science, Aalborg University, Denmark

ABSTRACT

Linkage analysis based on Multi Terminal Binary Decision Diagrams (MTBDDs) is developed. These data structures offer a significant decrease in memory and time requirements for doing exact multipoint linkage analysis. It is shown how unnecessary computations may be avoided when genotypes make some inheritance patterns impossible. Also shown is how convolutions of meioses may be delayed when singlepoint distributions are uninformative for these meioses. The delay of convolution leads to simpler probability distributions that may be stored in more compact MTBDDs. Finally an approximation method with computable, arbitrarily small, error is introduced. This approximation can substantially reduce computation costs at loci where many meioses are uninformative, in particular when no genotype information is available at a locus.

A new version of our linkage software package, ALLEGRO, has been created based on these ideas. The new implementation takes advantage of genotype information to reduce computational burdens and is able to handle large pedigrees with many genotyped individuals, that previous linkage analysis tools have been unable to analyze. For pedigrees, that have fewer genotyped individuals, improvements are still substantial.

Subject headings: linkage analysis

1. Introduction

Most contemporary methods for doing multipoint linkage analysis are based either on the Elston-Stewart algorithm (Elston & Stewart (1971)) or the Lander-Green Hidden Markov model (HMM) (Lander & Green (1987)). The Elston-Stewart algorithm scales exponentially in the number of markers, whereas the algorithms based on the Lander-Green HMM scale exponentially in the number of pedigree members. The VITESSE software package is based on the Elston-Stewart algorithm and can with the most recent advancements handle approximately 6-9 markers in moderately large pedigrees (O'Connell & Weeks (1995); O'Connell (2001)). Currently existing multipoint linkage analysis packages based on the Lander-Green HMM, GENEHUNTER (Kruglyak et al. (1996); Markianos et al. (2001)), ALLEGRO (Gudbjartsson et al. (2000)), and MERLIN (Abecasis et al. (2001)), can handle arbitrarily many markers but are limited to around 25 bits pedigrees. The bit size of a pedigree is calculated with the formula $2n - f$ for GENEHUNTER, where n is the number of non-founders in the pedigree and f is the number of founders in the pedigree. For ALLEGRO and MERLIN the bit size is $2n - f - g$, where g is the number of ungenotyped founder couples in the pedigree.

Binary Decision Diagrams (BDDs) were originally introduced as a compact representation of Boolean functions to reason about digital circuits (Andersen (1997); Bryant (1986)). Later their use has been extended to other disciplines but mainly software verification. BDDs provide a compact symbolic representation of computational problems that often offer a practical solution to difficult problems. The Multi Terminal BDDs (MTBDDs, sometimes also called Algebraic BDDs) are extensions of the original BDDs that allow for encoding of functions from a finite number of Booleans into any set of values (Bahar et al. (1997); Clarke et al. (1993)).

MTBDDs acquire their compactness through two properties. First, if two MTBDDs

represent the same function, they are the same MTBDD, this is referred to as uniqueness. Second, the two children of a node can not be the same MTBDD, this is referred to as non-redundancy. Figure 1 gives an example of how a function can be represented by a binary tree and a MTBDD. An example of uniqueness is that all nodes on level 3 that have children with values 1 and 2 are the same node in the MTBDD. An example of non-redundancy is that the node on level 3 in the binary tree, that has two children with value 0, is removed in the MTBDD.

Two things are gained by this compact storage of probability functions. First, larger probability distribution may potentially be stored. Second, it is easy to check if calculations have already been performed for a sub-diagram, as comparing if two sub-diagrams are equivalent is as simple as checking if they are the same sub-diagram. Thus, comparing two MTBDDs, both representing functions of N booleans, is a constant time operation, while the corresponding comparison for the same functions, represented as binary trees or vectors, has complexity of the order of 2^N.

Some of the patterns captured by MTBDDs is also captured by the gene flow trees of Abecasis et al. (2001), but the MTBDDs are more compact, in addition to comparisons having constant cost.

2. Multipoint linkage calculations with MTBDDs

Our multipoint linkage analysis algorithm is based on the standard Lander-Green HMM (Lander & Green (1987)). Let $G_1, G_2, \ldots, G_M$ be genotype information for M polymorphic markers genotyped for the pedigree being investigated. Let $v_m = v_{m1}v_{m2}\ldots v_{mN}$, with $v_{mi} \in \mathbb{B} = \{0, 1\}$, be the inheritance vector at marker m. Each bit, v_{mi}, in the inheritance corresponds to the status of meiosis i at marker m. The size of the inheritance vector,

N, dominates the computational complexity of most previous multipoint linkage analysis algorithms based on the Lander-Green HMM, and is of the order of $N2^N$.

The end result of the multipoint analysis is the probability distribution over inheritance vectors at each marker m, given all the observed data. Assuming the HMM and linkage equilibrium between markers, this distributions factors into a contribution from markers to the left of m, a single point contribution from marker m itself, and a contribution from markers to the right of m;

$$P\left(v_m|G_1,G_2,\ldots,G_M\right) \propto P\left(v_m|G_1,G_2,\ldots,G_{m-1}\right) P\left(v_m|G_m\right) P\left(v_m|G_{m+1},G_{m+2},\ldots,G_M\right). \tag{1}$$

Calculating the contributions from left and right requires convolving probability distributions and it is the cost of performing these convolutions that dominates the total computational cost. The convolution of a probability distribution p over $\mathbb{B}^N$ by a transition matrix T is the probability distribution

$$\forall v \in \mathbb{B}^N : (T * p)(v) = \sum_{w \in \mathbb{B}^N} T(w,v)p(w). \tag{2}$$

If we let $s_j = P\left(\cdot|G_j\right)$, and T_j be the transition matrix going from marker j to marker $j+1$, then, given the assumptions made, the contribution from markers to the left of marker m may be expressed as alternating convolutions and element wise multiplications of probability distributions:

$$P\left(\cdot|G_1,G_2,\ldots,G_{m-1}\right) = T_{m-1} * ((\ldots(T_2 * ((T_1 * s_1) \cdot s_2))\ldots) \cdot s_{m-1}). \tag{3}$$

Each of the transition matrices, $T_1, T_2, \ldots, T_{M-1}$, has typically been assumed to be functions of single genetic distance θ_m between markers m and $m+1$, or at most two genetic distances in the case of sex-specific recombination rates. However, because of delaying the convolution of uninformative bits (explained below) we shall assume that T_m is a function of N genetic distances, $\theta_{m1}, \theta_{m2}, \ldots, \theta_{mM}$, one for each meiosis.

In the MTBDD representation of probability distributions, each bit in the inheritance vector corresponds to one level in the MTBDD.

2.1. MTBDD based convolution algorithm

An algorithm similar to that of Idury & Elston (1997) may be used to perform multipoint linkage analysis. In order to convolve a probability distribution f over all inheritance vectors, let $f_0 = f$. Subsequently go trough all bits i that need to be convolved and calculate f_i based on f_{i-1} and the recombination fraction θ_i corresponding to the bit.

Some notation must be defined before giving the updating step. Let x_i be the MTBDD defined by

$$x_i(v) = \begin{cases} 1 & \text{if the } i\text{th bit of } v \text{ is 1,} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

and let $\bar{x}_i$ be the MTBDD corresponding to flipping x_i; $\bar{x}_i(v) = 1 - x_i(v)$. Define the ith cofactor of f, denoted by $\mathrm{Cof}_i(f)$, as the MTBDD that you would get by removing (or ignoring) the ith bit of $x_i f$, the elementwise product of x_i and f. Similarly, define the ith complemented cofactor of f, denoted by $\overline{\mathrm{Cof}}_i(f)$, as the MTBDD that you would get by removing the ith bit of $\bar{x}_i f$. Given these definitions f may be represented as sum of two disjoint components: $f = x_i \mathrm{Cof}_i(f) + \bar{x}_i \overline{\mathrm{Cof}}_i(f)$. The updating step is then

$$f_i = x_i \left\{(1-\theta_i)\mathrm{Cof}_i(f_{i-1}) + \theta_i \overline{\mathrm{Cof}}_i(f_{i-1})\right\} + \bar{x}_i \left\{(1-\theta_i)\overline{\mathrm{Cof}}_i(f_{i-1}) + \theta_i \mathrm{Cof}_i(f_{i-1})\right\}. \tag{5}$$

Here, and in what follows, the constant time comparison cost of MTBDDs is used to cache results of all calculations, so that if a set of calculations have been done once they need not be performed again. The above algorithm performs convolutions efficiently, in terms of taking advantage of cached calculations, and this caching makes a substantial difference to the running time of the implementation of the algorithm.

2.2. Bit reductions

The founder reduction of Kruglyak et al. (1996) and the founder couple reduction of Gudbjartsson et al. (2000) are handled in a straightforward manner. For every reduction, split the unreduced inheritance vector v into the bit that will be reduced, $v_r \in \mathbb{B}$, and the remaining unreduced bits $v_u \in \mathbb{B}^{N-1}$; $v = v_r v_u \in \mathbb{B}^N$. Let π be the transformation on $\mathbb{B}^{N-1}$ that specifies the reduction. Then for every probability distribution p that can arise based on genotype data and the assumptions made satisfies $p(0v_u) = p(1\pi(v_u))$ and $p(1v_u) = p(0\pi(v_u))$. Let p' be the reduced distribution, $p'(v) = p(0v) = p(1\pi(v))$, and define $\pi(p')$ by $(\pi(p'))(v) = p'(\pi(v))$. Then convolving the reduced bit corresponding to π for distribution f over the space of reduced inheritance vectors is a matter of calculating:

$$(1 - \theta_\pi)f + \theta_\pi \pi(f), \tag{6}$$

where θ_π is the recombination fraction for the reduced bit.

For the founder reduction, π simply flips the bits corresponding to the meioses of the founder. For the founder couple reduction, π swaps the bits corresponding to the children of the founding couple, and flips the bits of their grandchildren. The implementation takes advantage of the well known fact that if a founder has exactly two children then the reduced bit may be dropped completely and instead the recombination intensity of the unreduced bit doubled.

The convolution steps may be performed in almost any order. The only constraint is that if two bits are swapped by a founder couple reduction, they must either both be convolved before the founder couple reduced bit is convolved or both be convolved after the founder couple reduced bit is convolved.

2.3. Dropping unnecessary parts of convolved distributions

As demonstrated in equation (3), every convolved probability distribution is subsequently multiplied by the singlepoint distribution at the marker that it is being convolved to. For inheritance vectors, such that the value of the singlepoint distribution is zero, the value of the convolved distribution at said inheritance vector will be multiplied by zero. Therefore there is no need to calculate the value of the convolved distribution for these inheritance vectors. Given which level of the distribution is being convolved, and which levels have already been convolved it is straightforward to calculate which values of the convolved distribution are going to be needed. Let z_i be an MTBDD that is 1 if the corresponding value of f_i will be needed and 0 if it will subsequently be multiplied by zero and will therefor not be needed. Then (5) is replaced by:

$$\begin{aligned} f_i =& z_i \left[x_i \left\{ (1-\theta_i)\mathrm{Cof}_i(f_{i-1}) + \theta_i \overline{\mathrm{Cof}}_i(f_{i-1}) \right\} + \bar{x}_i \left\{ (1-\theta_i)\overline{\mathrm{Cof}}_i(f_{i-1}) + \theta_i \mathrm{Cof}_i(f_{i-1}) \right\} \right] \\ =& x_i \left\{ (1-\theta_i)\mathrm{Cof}_i(z_i)\mathrm{Cof}_i(f_{i-1}) + \theta_i \mathrm{Cof}_i(z_i)\overline{\mathrm{Cof}}_i(f_{i-1}) \right\} + \\ & \bar{x}_i \left\{ (1-\theta_i)\overline{\mathrm{Cof}}_i(z_i)\overline{\mathrm{Cof}}_i(f_{i-1}) + \theta_i \overline{\mathrm{Cof}}_i(z_i)\mathrm{Cof}_i(f_{i-1}) \right\}. \end{aligned} \tag{7}$$

The second formulation illustrates how the unnecessary calculations can be dropped.

The z_i indicator MTBDDs can be calculated through the following recursion algorithm. If N is the last bit to be convolved and s is the singlepoint distribution that the convolved distribution will be multiplied with, then define z_N by:

$$z_N(v) = \begin{cases} 0 & \text{if } s(v) = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{8}$$

For $i < N$ let

$$z_i = \mathrm{Cof}_{i+1}(z_{i+1}) \vee \overline{\mathrm{Cof}}_{i+1}(z_{i+1}), \tag{9}$$

where "$\vee$" is the or operator. For a reduced bit, whose reduction transformation is specified

by π, (9) is replaced by:

$$z_i = z_{i+1} \vee \pi(z_{i+1}). \tag{10}$$

For the first few bits that are being convolved, typically, close to every value is needed. But, for the later bits the amount of calculations saved becomes more substantial. The savings achieved for the last bit convolved are independent of the order in which bits are convolved. Interestingly the order in which bits are convolved may affect the amount of calculations saved for other bits. For example if z_N is equal to x_k, for some bit k, then it will be best to convolve bit k first, because then the number of values that need to be calculated are immediately cut in half. Figuring out the optimal order is a non-trivial task in itself. In the software implementation, an algorithm based on greedily letting convolutions float to the top, in a bubbling or sifting fashion, if the change of order reduces the size of the indicator MTBDDs, is used. The best order found from a couple of starting positions is then used in the actual convolution.

That the singlepoint distribution at a marker has value zero, for an inheritance vector, means that the inheritance vector was inconsistent with the given genotypes. More complete genotype information tends to lead to more inconsistent inheritance vectors, so that this improvement relies on the completeness of genetic information.

2.4. Delaying the convolution of uninformative bits

Equation (7) shows how substantial savings occur when the singlepoint distribution being convolved to is very informative. At the other extreme, when the singlepoint distribution at a marker is very uninformative, savings are also possible. Assume that bit number i in the inheritance vector at marker m is completely uninformative for the

singlepoint distribution at marker m, or more precisely that

$$\forall v_{m1}v_{m2}\ldots v_{mN} \in \mathbb{B}^N : P\left(v_{m1}v_{m2}\ldots, v_{mi}, \ldots, v_{mN}|G_m\right) = P\left(v_{m1}v_{m2}\ldots, \bar{v}_{mi}, \ldots, v_{mN}|G_m\right), \tag{11}$$

where $\bar{v}$ is the complement of the bit. Then convolving bit i may be delayed. If the recombination fraction between $m-1$ and m is θ_{m-1}, θ_{m-1} may be added to the next recombination fraction. So that if the recombination fraction between m and $m+1$ is θ_m, we use recombination fraction $\theta'_m = \theta_m + \theta_{m-1} - 2\theta_m\theta_{m-1}$ instead of θ_m for bit i when the probability distribution is convolved from marker m to marker $m+1$. This assertion is proved in Appendix A.

2.5. Variable reordering and rounding

The level of compression of a probability distribution provided by a MTBDD depends on the order of variables in the MTBDD. An important feature of MTBDDs, and in particular the implementation of MTBDDs being used in the implementation, is that efficient algorithms exist for improving the order of variables. The sifting algorithm of Rudell (1993) is used to optimize variable ordering.

During the course of the convolution of a distribution, the growth of its representation is monitored and the variables routinely reordered in order to improve compression.

The compression features of the MTBDD data structure depend on equalities existing between values of the functions being represented. The amount of equivalences that exist depend on how exactly values are stored in the MTBDD. The less exact the storage is, the higher level of compression is achieved. By default, real values are stored with accuracy corresponding to four significant digits, which we have found ample for our applications. Our software implementation allows the user to control this level of accuracy.

2.6. Completing uninformative convolutions

The delaying of convolutions makes a final step necessary before probability distributions at any given marker become available. Before the final step the algorithm has calculated l and r, the incomplete left and right probability distributions, and s, the single point probability distribution at the locus. Split the inheritance vector v into bits v_i, for which s is informative, and v_u, for which s is uninformative; $v = v_i v_u$ and $s(v_i v_u) = s'(v_i)$. The remaining convolutions are over the bits comprising v_u. Denote the left recombination fractions over these bits as θ_l and the right recombination fractions over these bits as θ_r. The result of the final step is the probability distribution at the locus given all the available genotypes, as shown in equation (1);

$$P\left(v|G_1, G_2, \ldots, G_M\right) \propto \left[(T_{\theta_l} * l)s(T_{\theta_r} * r)\right](v). \tag{12}$$

Some meioses are not directly interesting themselves. Typically these are meioses occurring in unphenotyped individuals, included in the analysis to provide extra phase information, such as unphenotyped children or siblings of affected individuals. Singling out a bit, v_t, corresponding to one such uninteresting meiosis, split v_u into v_w and v_t; $v = v_i v_w v_t$. Then $P\left(v|G_1, G_2, \ldots, G_M\right)$ is not the distribution of interest, but rather

$$P\left(v_i v_w|G_1, G_2, \ldots, G_M\right) = P\left(v_i v_w 0|G_1, G_2, \ldots, G_M\right) + P\left(v_i v_w 1|G_1, G_2, \ldots, G_M\right). \tag{13}$$

In this case it does not matter what values for the left and right recombination fractions, θ^*_{lt} and θ^*_{rt}, for the uninteresting bit were used, as long as they conserve the total genetic length of the correct recombination fractions θ_{lt} and θ_{rt}. That is to say $\theta^*_{lt} + \theta^*_{rt} - 2\theta^*_{lt}\theta^*_{rt} = \theta_{lt} + \theta_{rt} - 2\theta_{lt}\theta_{rt}$. In particular it is sufficient to set either θ^*_{lt} or θ^*_{rt} to zero. The proof of this result is based on similar arithmetic as was used in the proof of the validity of delaying convolving bits and is omitted.

An obvious benefit of this result is that fewer convolution steps need to be made

which is good in and of itself, but it also leads to a reduction in the complexity of the resulting MTBDD. Another benefit is that, since bits in one of the distributions are not convolved, some of its inheritance vectors may remain impossible (have probability zero), and thus the dropping of unnecessary calculations may lead to savings. Which of the two recombination fractions is made zero depends on which choice will lead to a smaller final indicator MTBDD, thus minimizing the amount of calculations required.

2.7. Approximating the completion of uninformative convolutions

The final convolution of uninformative bits may be prohibitively expensive for large problems. However, it is possible to approximate the completion of uninformative convolutions through an approximation method based on the result in the previous section.

As in the previous section, split the inheritance vector v into v_i and v_u, where s is informative for the bits in v_i, but uninformative for the bits in v_u. Denote the final result of the convolution by p^*, then the result of the previous section shows how $p_i^*(v_i) = \sum_{v_u} p^*(v_i v_u)$ can be calculated efficiently. Given $p_i^*(v_i)$ for all v_i and $\epsilon > 0$, it is possible to pick a threshold τ_ϵ such that

$$\sum_{p_i^*(v_i) > \tau_\epsilon} p_i^*(v_i) > 1 - \epsilon \tag{14}$$

and thus, by simply ignoring the $v = v_i v_u$ having $p_i^*(v_i) \leq \tau_\epsilon$, approximating $p^*(v)$ within arbitrary precision.

When a locus has few informative bits, or indeed, is completely uninformative then complexity may be reduced by selecting a few bits to convolved, and then treating them as if they had been informative, truncating the space of inheritance vectors with non-trivial probability mass and iterating.

3. Parametric peeling and score calculations

Parametric peeling is performed based on the Elston-Stewart algorithm. Only difference is that intermediate and final results are MTBDDs instead of real numbers.

The score calculations are essentially the same as the ones described in Gudbjartsson et al. (2000). In addition to the score functions described there, the wpc score function of Commenges & Beurton-Aimar (1999), a general pairwise scoring function, and the parent specific scoring functions (useful when studying imprinted disease) described in Karason et al. (2003), are all implemented.

4. Viterbi algorithm based analysis

Two important types of analysis are based on the Viterbi algorithm (Viterbi (1967)); finding the most probable haplotype assignments given the genotypes (Kruglyak et al. (1996)) and counting the minimum number of forced recombinations given the genotypes which is very useful when performing genetic mapping (Kong et al. (2002)).

Counting the number of forced recombinations is achieved by replacing the singlepoint distributions at each marker with an indicator function over inheritance vectors, which is zero if the inheritance vector is incompatible with the observed genotypes at the marker, and one if it is consistent. Also, all recombination distances must be set as one fixed number, between zero and one half. Given these artificial single point distributions and recombination fractions, the standard haplotyping version of the Viterbi algorithm can be used to count the forced number of recombinations.

The classical implementation of the Viterbi algorithm involves storing all intermediary most probable paths up until every given marker (Rabiner (1989)). This approach saves time in the backtracking phase of the algorithm, but at the cost of a substantial amount of

memory. At the larger bit sizes the amount of memory required becomes prohibitive. In our implementation we have chosen to trade speed for memory usage, in order to be able to tackle these larger problems, and redo some calculations in the backtracking phase.

Implementing the Viterbi algorithm based on MTBDDs is very similar to the implementation of the multipoint linkage analysis algorithm for MTBDDs; essentially summations are changed to maximizations. The savings obtained when inheritance vectors are incompatible with genotype information carry over, as do the bit skipping savings. Interestingly, the completion step is substantially less expensive, as the approximation procedure described above for the linkage analysis completion step translates into an exact procedure for the Viterbi algorithm.

A. Proof of bit-skipping

Let l_{m-1} be the probability distribution over inheritance vectors at marker $m-1$, given all genotype data to the left of, and at, $m-1$. Let s_m be the singlepoint distribution over inheritance vectors at marker m. Assume that s_m is completely uninformative about the last bit in the inheritance vector (in the sense of equation 11); if we split inheritance vectors v, into the first $N-1$ bit and the last bit, $v = v_f v_N$, then $\forall v_1 : s_m(v_f 0) = s_m(v_f 1)$. Assuming there are no bit reductions, the transition matrices T_{m-1} and T_m also factor:

$$\begin{aligned} T_{m-1}(v,w) &= T_{m-1}(v_f v_N, w_f w_N) &= T_{f,m-1}(v_f, w_f) T_{\theta_{m-1}}(v_N, w_N), \\ T_m(v,w) &= T_m(v_f v_N, w_f w_N) &= T_{f,m}(v_f, w_f) T_{\theta_m}(v_N, w_N), \end{aligned}$$

where $T_\theta(v,w)$ is $1-\theta$ if $v = w$ and θ if $v \neq w$. We show that

$$T_m * ((T_{m-1} * l_{m-1}) \cdot s_m) = T'_m * ((T'_{m-1} * l_{m-1}) \cdot s_m), \tag{A1}$$

where $T'_{m-1} = T_{f,m-1} T_0$, $T'_m = T_{f,m} T_{\theta'}$, and $\theta' = \theta_{m-1} + \theta_m - 2\theta_{m-1}\theta_m$.

Straightforward calculations using (2), the uninformativeness of s_m on the last bit, the above factorizations of transition matrices, and reordering of sums gives:

$$\begin{aligned} &(T_m * ((T_{m-1} * l_{m-1}) \cdot s_m))(w_f w_N) = \\ &\sum_{v_f v_N} T_{m,f}(v_f, w_f) T_{\theta_m}(v_N, w_N) ((T_{m-1} * l_{m-1})(v_f v_n) s_m(v_f v_N) = \\ &\sum_{v_f v_N} T_{m,f}(v_f, w_f) T_{\theta_m}(v_N, w_N) s_m(v_f v_N) \sum_{u_f u_N} T_{m-1,f}(u_f, v_f) T_{\theta_{m-1}}(u_N, v_N) l_{m-1}(u_f u_N) = \\ &\sum_{v_f, u_f u_N} T_{m,f}(v_f, w_f) s_m(v_f 0) T_{m-1,f}(u_f, v_f) l_{m-1}(u_f u_N) \sum_{v_N} T_{\theta_m}(v_N, w_N) T_{\theta_{m-1}}(u_N, v_N). \end{aligned}$$

But the inner sum is simply equal to $T_{\theta'}(u_N, w_N)$:

$$\begin{aligned} u_N = w_N : &\quad \textstyle\sum_{v_N} T_{\theta_m}(v_N, w_N) T_{\theta_{m-1}}(u_N, v_N) = (1-\theta_m)(1-\theta_{m-1}) + \theta_m \theta_{m-1} = 1 - \theta' \\ u_N \neq w_N : &\quad \textstyle\sum_{v_N} T_{\theta_m}(v_N, w_N) T_{\theta_{m-1}}(u_N, v_N) = (1-\theta_m)\theta_{m-1} + \theta_m(1-\theta_{m-1}) = \theta'. \end{aligned}$$

Substituting the inner sum with $T_{\theta'}(u_N, w_N)$ and going back through (2) readily yields (A1).

The above proof ignores bit reductions (the founder reduction and the founder couple reductions) because having shown that skipping a bit is valid when there are no reductions, it must be remain valid when there are reductions. The only caveat is for bits that are swapped in the founder couple reductions. For these the pair of bits that are swapped must both be uninformative for the skip to be valid. Note that, in particular, skipping is valid for the reduced bits.

REFERENCES

Abecasis, G. R., Cherny, S. S., Cookson, W. O., & Cardon, L. R. 2001, Nature Genetics, 30, 97

Andersen, H. R. 1997, An Introduction to Binary Decision Diagrams (Lecture notes, Technical University of Denmark)

Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., & Somenzi, F. 1997, Formal Methods in Systems Design, 10, 171

Bryant, R. E. 1986, IEEE Transactions on Computers, 8, 677

Clarke, E., Fujita, M., McGeer, P., Yang, J., & Zhao, X. 1993, International Workshop on Logic Synthesis, XX, XX

Commenges, D. & Beurton-Aimar, M. 1999, Genetical Epidemiology, 17 Suppl 1, S515

Elston, R. C. & Stewart, J. 1971, Human Heredity, 21, 523

Gudbjartsson, D. F., Jonasson, K., Frigge, M. L., & Kong, A. 2000, Nature Genetics, 25, 12

Idury, R. M. & Elston, R. C. 1997, Human Heredity, 47, 197

Karason, A., Gudjonsson, J. E., Upmanyu, R., Antonsdottir, A. A., Hauksson, V. B., Runasdottir, E. H., Jonsson, H. H., Gudbjartsson, D. F., Frigge, M. L., Kong, A., Stefansson, K., Valdimarsson, H., & Gulcher, J. R. 2003, Am J Hum Genet., 72, 125

Kong, A., Gudbjartsson, D. F., Sainz, J., Jonsdottir, G. M., Gudjonsson, S. A., Richardsson, B., Sigurdardottir, S., Barnard, J., Hallbeck, B., Masson, G., Shlien, A., Palsson, S. T., Frigge, M. L., Thorgeirsson, T. E., Gulcher, J. R., & Stefansson, K. 2002, Nat Genet., 31, 241

Kruglyak, L., Daly, M. J., Reeve-Daly, M. P., & Lander, E. S. 1996, American Journal of Human Genetics, 58, 1347

Lander, E. S. & Green, P. 1987, Proc Natl Acad Sci U S A., 84, 2363

Markianos, K., Daly, M. J., & Kruglyak, L. 2001, Am J Hum Genet., 68, 963

O'Connell, J. R. 2001, Human Heredity, 51, 226

O'Connell, J. R. & Weeks, D. E. 1995, Nat Genet., 11, 402

Rabiner, L. R. 1989, Proc IEEE, 77, 257

Rudell, R. 1993, in ICCAD '93: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design (IEEE Computer Society Press), 42–47

Viterbi, A. J. 1967, IEEE Trans Informat Theory, IT-13, 260

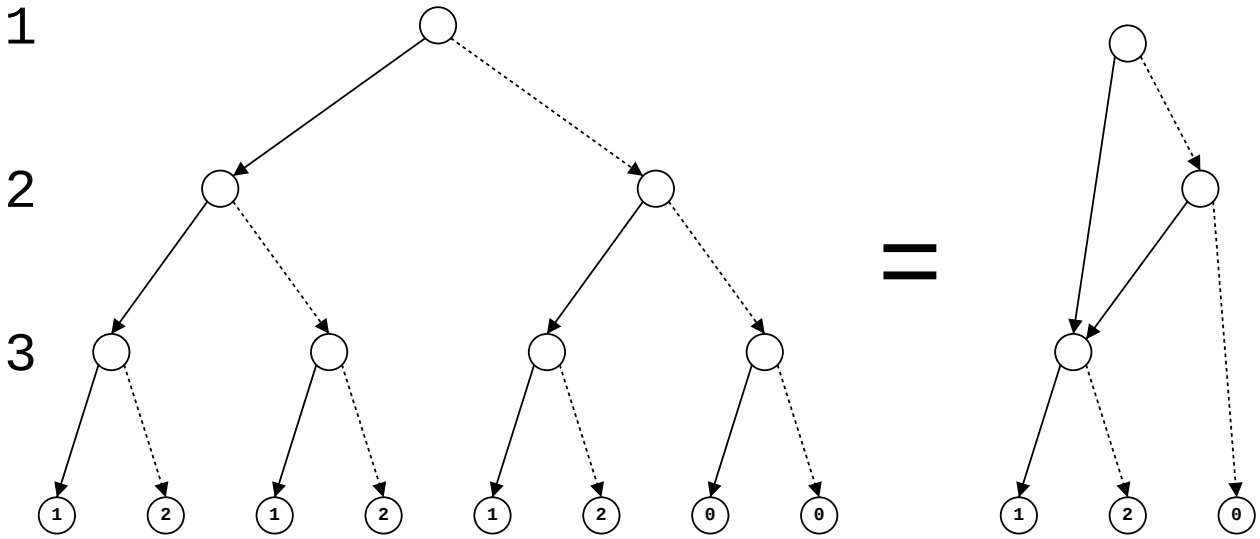

Fig. 1.— A binary tree and its corresponding MTBDD. Both represent a real valued function over $\mathbb{B}^3$. Each of the three levels corresponds to one bit. If the solid edge is traversed, the value of the underlying bit is 0, and if the dotted edge is traversed the value of the bit is 1. The function can also be represented by the vector [1, 2, 1, 2, 1, 2, 0, 0].