EFFICIENT RECURSIVE LINKING ALGORITHM FOR COMPUTING THE LIKELIHOOD OF AN ORDER OF A LARGE NUMBER OF GENETIC MARKERS

S. Tewari
Dept. of Statistics, University of Georgia,
Athens, GA 30605-1952, USA
**Email: statsusant@yahoo.com*

Dr. S. M. Bhandarkar *
Dept. of Computer Science,University of Georgia,
Athens, GA 30605-7404, USA
**Email: suchi@cs.uga.edu*

Dr. J. Arnold
Dept.of Genetics, University of Georgia,
Athens, GA 30605-7223, USA
**Email: arnold@uga.edu*

Assuming no interference, a multi-locus genetic likelihood is implemented based on a mathematical model of the recombination process in meiosis that accounts for events up to double crossovers in the genetic interval for any specified order of genetic markers. The mathematical model is realized with a straightforward algorithm that implements the likelihood computation process. The time complexity of the straightforward algorithm is exponential without bound in the number of genetic markers and implementation of the model for more than 7 genetic markers is not feasible, motivating the need for a novel algorithm. A recursive linking algorithm is proposed that decomposes the pool of genetic markers into segments and renders the model implementable for a large number of genetic markers. The recursive algorithm is shown to reduce the order of time complexity from exponential to linear. The improvement in time complexity is shown theoretically by a worst-case analysis of the algorithm and supported by run time results using data on linkage group-I of the fungal genome *Neurospora crassa*.
Keywords: Crossover, EM algorithm, recursive linking, time complexity, MLE.

1. INTRODUCTION

High density linkage maps are an essential tool for characterizing genes in many systems, fundamental genetic processes, such as genetic exchange between chromosomes, as well as the analysis of traits controlled by more than one gene (*i.e.,*complex traits)[1]. Since genetic maps are most often the critical link between phenotype (what a gene or its product does) and the genetic material, genetic maps can be exploited to address how the genetic material controls a particular trait [2] controlled by one or more genes. Most model systems possess high density linkage maps that can assist in the analysis of complex traits. The bread mold, *Neurospora crassa*[3], which gave us the biochemical function of genes, is no exception[4]. One approach to understanding the genetic basis of a complex trait is to follow its segregation in offspring along with an array of genetic markers. One class of genetic markers frequently used are restriction fragment length polymorphism (RFLP), markers in the DNA itself. These markers in essence allow a triangulation on loci in the DNA affecting the complex trait. Part of this triangulation process involves the construction of a genetic map with many markers. This is a computationally challenging problem [5] and at the heart of understanding complex traits, such as human disease. In this paper we address the problem of genetic map reconstruction from a large number of RFLP markers. We focus on map construction for a model system *N.crassa*[6] where there is a wealth of published information about how markers segregate because the genetic makeup of gametes can be identified. Previous attempts ([7, 8]) at genetic map

*Corresponding author.

construction of a large number of genetic markers are mostly based on pair wise genotypic information, not on likelihood computation. Here we focus on solving the computational challenge posed by our probabilistic modelling of the genetic map.

2. MULTILOCUS GENETIC LIKELIHOOD FOR A SPECIFIED ORDER OF GENETIC MARKERS.

Let S be the sample space of an exchange (crossover) between any two non-sister strands (chromatids) of the tetrad in a single meiosis. Let c denote the probability of exchange of genetic material between any two non sister strands in the tetrad at meiosis.

$$\begin{aligned} S &= \{0,1,2,3,4\} \\ P(i) &= \frac{c}{4}; i = 1,\cdots,4; i \in S \\ P(0) &= 1-c; 0 \in S \end{aligned} \tag{1}$$

The element 0 in S indicates the absence of a crossover event. Elements 1, 2, 3 and 4 indicate that non-sister chromatids $(1,3),(2,3),(2,4)$ and $(4,1)$ took part in exchange respectively.

Let $S_i(= S \times S)$ be the set whose elements denote crossover events covering events up to double crossovers between locus $A_i$and $A_{(i+1)}(i = 1,...,l-1)$, where l=total number of loci being studied. Using equation (1), the probability distribution on S_i is given by:

$$\begin{aligned} P(\{i,j\}) = &\frac{c_i^2}{16} I_{\{i\neq 0; j\neq 0\}} \\ &+ \frac{c_i(1-c_i)}{4}\{I_{\{i=0;j\neq 0\}} + I_{\{i\neq 0;j=0\}}\} \\ &+ (1-c_i)^2 I_{\{i=j=0\}} \end{aligned} \tag{2}$$

where, $\{i,j\} \in S_i$.

Let ϕ_k denote a unique crossover event on S^l as described below:

$$\phi_k = i_1 \times i_2 \times ... \times i_{l-1} \tag{3}$$

where,

$$k = i_1.i_2.i_3...i_{l-1} \ ; \ i_j \in S_i \ ; \ \phi_k \in S^l = \prod_{i=1}^{l-1} S_i$$

Let f_k denote a multi-locus genotype with l loci:

$$f_k = i_1 \times i_2 \times ... \times i_{l-1} \times i_l$$

where,

$$k = i_1.i_2.i_3...i_l \ \ ; \ \ i_j = 0,1; \ \ \forall j = 1,\cdots,l$$

The indices $i_j = 1$ and $i_j = 0$ indicate the paternal and maternal alleles respectively. The progeny are obtained by crossover between homogeneous parents. The observed data set can be represented as:

$$\mathcal{D} = \left\{n_j \ \ ; \ \ \forall j = 1,\cdots,2^l\right\}$$

where, n_j is the observed frequency of f_j.

2.1. Probability distribution on S^l

Let us define the following functions

$$\begin{aligned} f^0(a) &= (a_1,a_2,a_3,a_4)' \\ f^1(a) &= (a_3,a_2,a_1,a_4)' \\ f^2(a) &= (a_1,a_3,a_2,a_4)' \\ f^3(a) &= (a_1,a_4,a_3,a_2)' \\ f^4(a) &= (a_4,a_2,a_3,a_1)' \end{aligned} \tag{4}$$

where,

$$\begin{aligned} a &= (a_1,a_2,a_3,a_4)' \\ a_i &= 0,1 \ \ \forall i \end{aligned}$$

The function $f_{ij}(a) = f_j(f_i(a))$ corresponds to the events in S_i. For a particular crossover ϕ_k we can generate a model tetrad at meiosis using the function f_{ij}. The following matrix R_k of size $4 \times l$ defines the simulated tetrad as follows:

$$R_k = \left(R_0 R_1 \cdots R_{(l-1)}\right) \tag{5}$$

where,

$$R_0 = (1100)' \ \ ; \ \ R_i = f_{jk}(R_{i-1}) \ \forall i = 1,\cdots,l-1$$

and the i^{th} genetic interval S_i observed the crossover event $\{j,k\}$. The conditional distribution of f_i for a given ϕ_k is

$$P(f_i|\phi_k) = \frac{1}{4}\sum_{j=1}^{4} I_{f_i \in R_k(j,.)} \tag{6}$$

where, $R_k(j,.)$is the j^{th} row of R_k.

The marginal density of a single spore f_i is given by

$$\begin{aligned} P(f_i) &= \sum_k P(f_i|\phi_k) \times P_k \\ &= C \times P \end{aligned} \tag{7}$$

where, C is the conditional probability matrix given by :

$$\left.\begin{array}{l} C \ = ((c_{ki})) \\ c_{ki} = P(f_i|\phi_k) \quad \text{(from equation (6))} \end{array}\right\} \tag{8}$$

and P is given by,

$$\begin{aligned} P &= (P_k; \ \ \forall k)' \\ P_k &= P(\phi_k) \\ &= \prod_{j=1}^{l-1} P\left(I_j = i_{k,j}\right) \end{aligned} \tag{9}$$

where, $i_{k,j} \in S_j$ and the probability distribution $P\left(I_j = i_{k,j}\right)$ is as defined in equation(2).

Let $\Theta = (c_1, c_2, ..., c_{l-1})'$ denote the unknown parameter vector in the model. The log-likelihood of $\mathbf{f}$ viewed as a function of Θ is given by:

$$\ln(\Theta|D) = \sum_{j=1}^{N} n_j \log\Bigg[\sum_k \Big[\frac{1}{4}\sum_{j=1}^{4} I_{\{f_i \in R_k(j,.)\}} \times \prod_{j=1}^{l-1} P_j\left(I_{k,j} = i_{k,j}\right)\Big]\Bigg] \tag{10}$$

The following two theorems solve equation(10) using a set of recurrence relations obtained via the Expectation-Maximization (EM) algorithm [9]. The proofs are not given in the interest of brevity.

Theorem 1.

The EM-iterative equations are given below.

$$\Theta^{(h+1)} = \left(c_m^{(h+1)} \ \ \forall m = 1, \cdots, l-1\right)'$$

$$\text{where } c_m^{(h+1)} = \left(\frac{2N_{2,m} + N_{1,m}}{2N_m}\right)^{(h)} \tag{11}$$

where

$$N_m = \sum_k n_k^{(h)} = N_{0,m} + N_{1,m} + N_{2,m}$$

$$N_{0,m} = \sum_{k\,\big|\, i_{k,m}=(0,0)} n_k^{(h)}$$

$$N_{1,m} = \sum_{\substack{k\,\big|\, i_{k,m}=(i_1,i_2) \\ i_1=0 \text{ (Strict)OR } i_2=0}} n_k^{(h)}$$

$$N_{2,m} = \sum_{\substack{k\,\big|\, i_{k,m}=(i_1,i_2) \\ i_1\neq 0 \text{ AND } i_2\neq 0}} n_k^{(h)}$$

$$n_k^{(h)} = \sum_j n_j \pi_{k|j}(\Theta^{(h)})$$

$$\pi_{k|j} = P(x_{kj} = 1|f_j) = \frac{\pi_{j|k} \times \pi_k}{p_j}$$

$$p_j = \sum_k \pi_{j|k} \times \pi_k$$

$$\pi_{j|k} = c_{ki} \text{ in equation (8)}$$

$$\pi_k = P_k \text{ in equation (9)}$$

Note that, $i_{k,m}$ denotes an event in S_m for the crossover ϕ_k.

Theorem 2.

Let $\mathbf{f} = (f_1, f_2, f_3, f_4)'$ be the observed frequency vector corresponding to all possible meiotic products for parental genes M and O for two markers. The genotype vector for $\mathbf{f}$ is (MM MO OM OO)'. The maximum likelihood estimator [10] of the exchange probability c under the model represented by Equation(1) is unique and is given as follows:

(1) If $f_1 + f_4 < f_2 + f_3$ then $c_{mle} = 1$
(2) If $f_1 + f_4 \geq f_2 + f_3$ then c_{mle} is given by the unique solution (in the interval $[0, 1]$) of the following equation:

$$f(c) = c^2 - 2c + D = 0 \tag{12}$$

where,

$$D = \frac{2(f_2 + f_3)}{N} \ ; \ N = \sum_{i=1}^{4} f_i$$

This theorem is used to obtain the starting values of c_m for the EM-iterative equations in Theorem 1.

3. THE STRAIGHTFORWARD ALGORITHM

In the pseudocode below, k denotes a particular crossover ϕ_k as defined in equation(3). The function *getRMatrix* implements equation(5) to create the R_k matrix corresponding to the crossover ϕ_k. The function $kProb$ computes the marginal probability P_k due to crossover ϕ_k as defined in equation(9). The matrix R_k and the probability P_k in turn create matrices C and P progressively during the course of the recursive loop to calculate the marginal probability of each observed distinct genotype as defined in equation(7). These marginal probabilities along with the counts for the distinct genotypes are used to compute the log-likelihood in equation(10). A particular crossover ϕ_k does not enter into the computation of the likelihood so long it does not have positive probability for at least one distinct genotype. The elimination of such crossovers is achieved with the function $kIsWorthy$, which implies that at least some amount of computation cannot be avoided for each crossover. In the recursive algorithm we propose in the paper, this feature is handled more efficiently where a large number of crossovers are eliminated by performing checks on a few. In the pseudocode the vector *sum* computes conditional probabilities across all distinct genotypes. The conditional probability is computed with the help of the function *countMatch* that implements the equation(6), counting the number of strands (out of 4) in R_k that match with the observed genotype. The vector $freq$ has the observed count corresponding to the distinct genotypes(d_g) and *totalObs* is the total sample size and *probFOld* stores the marginal probability of each distinct genotype using equation(7).The array *pCount* implements the EM algorithm via equation(11) by recategorizing the vector *sum* based on the crossover values along the chromosome. Note that the denominator of $\pi_{k|j}$, p_j, the marginal probability due to the j^{th} distinct genotype, is left out from its ($\pi_{k|j}$) computation as that requires going through all the crossovers and is currently being progressively computed by *probFOld*. To compute $n_k^{(h)}$ in equation(11) we need to add up the inverse probabilities ($\pi_{k|j}$) across the distinct genotypes but, as their marginal probabilities are not computed yet it is not possible to do so. We work around this problem by adding another dimension along the number of distinct genotypes to the structure *pCount*. The first dimension of *pCount* is of magnitude 3 to account for $N_{0,m}$,$N_{1,m}$ and $N_{2,m}$ in equation(11) and the second dimension runs along m, accounting for the $(l-1)$ genetic intervals. Once all the crossovers are processed and the marginal probabilities computed, the elements in the third dimension are divided by their corresponding marginal probabilities and then added up across the dimension. This gives us the two dimensional structure *postCount* (not shown in the pseudocode) containing values of $N_{0,m}$,$N_{1,m}$ and $N_{2,m}$ for all the genetic intervals. Then the new value of c_i for each genetic interval is computed using equation(11) and the process iterates until convergence. Despite being a recursive algorithm (crossovers are generated recursively) it suffers from the computational bottleneck to process a huge ($25^{(l-1)}$ for loci l) number of crossovers. This problem is overcome using the proposed recursive linking algorithm. For brevity we show the pseudocode of only the most important part of the straightforward algorithm.

```
{Pseudocode of the Straightforward Algorithm}
loop
  { This is a recursive loop. This dynamically
  generates l-1 FOR loops.}
  r=getRMatrix(k)
  if kIsWorthy()==1 then
    prob=kProb(cProbOld,k);
    for i = 0; i < d_g; i++ do
      sum[i] = countMatch() * prob * freq[i]/4.0 *
      totalObs
      probFOld[i]+ = countMatch()/4.0 * prob
    end for
    for j = 0; j < loci - 1; j++ do
      pCount[CellSpecial(k[j])][j]+ = sum
      { The function CellSpecial() maps
      crossover values from 0 to 24 to the events
      of the set S_i and the addition is a
      componentwise vector addition.}
    end for
  end if
end loop
```

4. THE PROPOSED RECURSIVE LINKING ALGORITHM

Let the entire order of genetic markers be broken into equal segments of width (h), such that all the intervals are covered. So, for l genetic markers the number of segments s is given by the following equation :

$$s = \frac{(l-1)}{h-1} \tag{13}$$

The first segment has an associated array called $kArrayFirst$ that has all the crossovers for the segment. The array $linkInfoFirst$ stores the last row generated by the R matrix for each crossover of the segment. The array $cArrayFirst$ checks for each particular crossover and each observed genotype of the first segment which strands of the simulated (based on the model described in equation(1)) tetrad obtained by R match with the genotype. The matching status forms the last dimension of the array with length 4 and consists of symbols 1 and 0 indicating a match(1) and mismatch(0) respectively. For example, a matching status 1 0 0 1 for the first distinct genotype corresponding to crossover pattern 0 0 2 3 4 in the first segment level indicates that among the 4 tetrads in meiosis generated by the crossover pattern 0 0 2 3 4 in the first segment, the observed genotype in question was found only on the first and the fourth tetrad. When we use this information over a combined segment formed with two segments, only a match at the same tetrad position will ensure a match for the combined segment.Note that R depends on crossover values on all intervals of the segment and its columns are sequentially dependent on each other with a lag one. Each crossover in the first segment branches out to $25^{(h-1)}$ crossovers in the following segment and creates $25^{2(h-1)}$ combined crossovers. This continues till the last segment is accounted for. In order to move along the segments following the model described in equation(1), we need to know the last row (a tetrad pattern at the last locus of the segment) generated by the R matrix of the linked crossover of the previous segment corresponding to each combined crossover of the two segments. The following lemma states that only certain patterns are possible at the end locus of the adjoining segments and we create arrays similar to $kArray$,$linkInfo$ and $cArray$ for all the following segments except the last one and call them $kArrayTemp[]$,$linkInfoTemp[]$ and $cArrayTemp[]$ respectively, where the dimension denotes the segment numbers.

Lemma 4.1. *Under the model described by equation(1) at any particular locus only one of the tetrad patterns 1 1 0 0, 0 1 1 0, 1 0 1 0, 1 0 0 1, 0 1 0 1 and 0 0 1 1 could occur.*

Consider a combined crossover for all the segments. To compute equation(6) i.e, to count the matches for the entire crossover we have to examine the matching status of all the segments and update them. To be considered a match for the whole segment at a particular position (out of 4 possible positions) one must have a match for all the segments at that position. Hence when the matching status of two segments are updated the resulting matching status is 1 if and only if both the segments have 1 at that position and 0 otherwise. This updated matching status is termed a *spore* in this paper. The distinction between a spore and matching status is that while matching status is the original status of the segment the spore is the matching status obtained after updating the matching status of all the previous segments. The following lemma restricts the number of possible spore patterns.

Lemma 4.2.

Under model described by equation(1), for any crossover on any observed genotype the spore patterns 0 1 1 1, 1 0 1 1, 1 1 0 1, 1 1 1 0 and 1 1 1 1 are not possible.

Analogous to the function *kIsWorthy* in the straightforward algorithm we implement the concept of counting active crossovers for each segment. We call a crossover of a particular segment active if it has positive probability for at least one distinct genotype. Note that active crossovers will be different across segments as an active crossover depends both on the observed genotypes(that varies across segments) and the tetrad pattern used in the generation of the R matrix.

It is important to emphasize the huge computational gain achieved by elimination of the crossovers on a segment wise basis in the proposed algorithm compared to the straight forward algorithm which

eliminates crossovers one at a time. A single elimination of a crossover in the first segment has the effect of 25^{l-2} eliminations of crossovers in the straightforward algorithm. The effect is 25^{l-3} for the 2nd segment and so on.

For the last segment we create an array called *tempSumIndex*. Corresponding to all possible spores (the dimension is restricted by Lemma 4.2) the matching status of the last segment is updated and then it is summed (along its dimension) to compute equation(8) for the last segment except that the matching status is computed for all possible tetrad patterns in Lemma 4.1. Note that Lemma 4.1 and Lemma 4.2 restrict the array size and ensure storage economy. Next we implement equation(7) for the last segment for all possible spores denoted by the array *tempSum*. To update θ using equation(11) we use another array called *tempPCount* which computes $N_{0,m}$,$N_{1,m}$ and $N_{2,m}$ via index manipulation as shown in the pseudocode.

{***Pseudocode for the recursive structure***}
tempSum[6][d_g][11]
tempPCount[6][d_g][11][3][loci-1]
loop
  $\{i_1, i_2, i_3, ; 0 <= i_4 <$ activeKLast[i_1]$\}$
  tempDouble=kProb(lastCProb,kArrayLast[i_1][i_4])
  condProb=tempSumIndex[i_1][i_2][i_3][i_4]/4.0
  tempSum[i_1][i_2][i_3]+ = condProb$\times$tempDouble
  for $j =$ loci $-$ h; $j <$ loci $-$ 1; $j++$ **do**
    tempPCount[i_1][i_2][i_3][CellSpecial(kArrayLast[i_1][i_4][$j-$loci + h])][index]+ = condProb $\times$ tempDouble;
  end for
end loop

The arrays *tempSum* and *tempPCount* taken together are termed a recursive structure for the last segment. This structure has the property that $25^{(h-1)}$ crossovers have already been processed in a form so that equation(11) can be implemented and it can handle any spore generated from the previous segment. Note that when a crossover from the previous (allowing for all possible spore patterns of its previous segment) segment is processed the recursive structure identifies and uses an appropriate spore from the last segment and thus processes simultaneously $25^{(h-1)}$ combined crossovers of these two segments. After all the crossovers from the previous segment are processed the proposed algorithm generates a recursive structure that is exactly the same as that of the last segment without adding to the storage requirement. The recursive structure for the last but one segment now has $25^{2(h-1)}$ crossovers processed within it with provision for all possible spores from the previous segment. This very feature shows how we geometrically increase the information base (in terms of crossovers) of the "table look up " procedure and avoid traversing all the crossovers one at a time. One of the reasons this procedure works is because if we look into the combined crossovers of two segments we see that crossover values for the left segment change only for every $25^{(h-1)}$ combined crossovers. This lets us delay the probability value updates when linking the segments. The process is best understood by looking at the pseudocode below and noticing how the arrays *tempSum* and *tempPCount* are updated in the recursive linking process.

Recursive Linking :
temp1Sum=tempSum
temp1PCount=tempPCount
int startPos=0
int endPos=loci-h
for $i = 0$; $i <$ segments-2 ; $i++$ **do**
  startPos=endPos-h+2
  firstCProb $\rightarrow$ *from* startPos *to* endPos in cProbOld
  spores[11][4] $\rightarrow$ generate spores
  for $j_0 = 0$; $j_0 < 6$; j_0++ **do**
    for $j_1 = 0$; $j_1 <$ dG ; j_1++ **do**
      for $j_2 = 0$; $j_2 < 11$; j_2++ **do**
        for $j_k = 0$; $j_k <$ activeKTemp[i][j_0] ; j_k++ **do**
          $k =$ kArrayTemp[i][j_0][j_k]
          prob=kProb(firstCProb,k);
          int j_{01} =linkInfoTemp[i][j_0][j_k]
          int spike=SporeMatch(UpdateScore(cArrayTemp[i][j_0][j_k][j_1],spores[j_2]))
          temp2Sum[j_0][j_1][j_2]+ =temp1Sum[j_{01}][j_1][spike]$\times$prob
          for m =startPos-1 ; $m <$ endPos ; $m++$ **do**
            temp2PCount [j_0][j_1][j_2][CellSpecial(k[m $-$ startPos + 1])][m]+ =tempSum[j_{01}][j_1][spike]$\times$prob
          end for

```
            for n = 0 ; n < 3; n++ do
              for index=endPos ; index < loci-1 ;
              index++ do
                temp2PCount[j0][j1][j2][n][index]+
                =temp1PCount[j01][j1][spike][n]
                [index]× prob
              end for
            end for
          end for
        end for
      end for
    end for
    temp1Sum=temp2Sum
    temp1PCount=temp2PCount
    endPos=startPos-1
  end for
```

Linking with the first segment is the last step of the likelihood computation process. In this phase we do not have any previous matching status vectors and instead of *tempPCount* we have the structure *pCount* as in the straightforward algorithm. Note that the length of the first segment must be adjusted to account for both even and odd number of genetic markers. That entails a trivial modification of the algorithm, hence we do not mention the details. In the interest of clarity of description of the algorithm we have assumed all the segments to be of equal length and hence both even and odd number of markers can not be implemented without first changing the length of at least one segment; preferably that of the first one.

5. TIME COMPLEXITY COMPARISON OF THE ALGORITHMS

In the analysis of time complexity of the algorithms [11] the running variable is l, the number of genetic markers. The core function of the algorithm is to process a large number of crossovers in real time. The algorithm would be required even if one wanted to compute just the likelihood for the initial probabilities and not use the subsequent EM iterations. Hence in both the straightforward and the proposed algorithm we provide run time complexity analysis for the main computationally intensive phase, namely processing all the crossovers for a single iteration.

The loop in the straightforward algorithm runs for $25^{(l-1)}$ iterations. In each iteration the computation time of matrix R is $O(l)$. This is so because R has l columns and the computation time for each column is fixed. Since the observed genotypes are not known in advance we do a worst-case analysis for the computation of vector *sum*. The worst case occurs when there is a match between an observed genotype and any of the 4 columns of the matrix R resulting in run time complexity of order $O(l)$. The vector *sum* has d_g elements which does not vary with l and hence the total execution time for computing vector *sum* is $O(l)$. The vector addition involved in computing *pCount* entails d_g elements and thus accounts for run time complexity of $O(l)$. Hence the total run time complexity of the straightforward algorithm is

$$R_{st} = O\left(l25^{(l-1)}\right) \tag{14}$$

In the recursive linking algorithm the run time for each of the s segments is $O((h-1)25^{(h-1)}))$. So the total run time for the recursive algorithm is

$$R_{rl} = sO\left((h-1)25^{(h-1)}\right)$$

which is minimized for $h = 3$ for any odd number of loci l. Hence for $h = 3$ the run time complexity for the recursive linking algorithm is of order $O(l)$ using equation(13).

6. RUN TIME RESULTS

We ran several jobs on a DELL PC (Model DM051 Pentinum(R) 4 CPU 3.40GHz and 4GB of RAM) for different number of genetic markers in their natural order of precedence on a data set from the linkage group-I of *Neurospora crassa*[4] for both the algorithms. It was verified that both the algorithms provide the same likelihood for the same order of markers as they essentially solve the same problem but in different ways. The run time corresponds to the average time it takes for a single EM iteration before convergence on multiple starts. The resulting speedup is clearly evident from the table below.

Table 1. Run time (in seconds) Comparison of Straightforward and Recursive Algorithm

Loci(l)	Straightforward Algorithm	Recursive Linking Algorithm
5	2.49	0.073
7	1336.45	0.298
9	$>$ 43200.0	0.66
21	∞	5.61
41	∞	8.93
61	∞	13.03

7. CONCLUSIONS

Assuming no interference, a multi-locus genetic likelihood was implemented based on a mathematical model of the recombination process in meiosis that accounted for events up to double crossovers in the genetic interval for any specified order of genetic markers. The mathematical model was realized with a straight forward algorithm that implemented the likelihood computation process. The time complexity of the straightforward algorithm was exponential without bound in the number of genetic markers and implementation of the model for more than 7 genetic markers turned out to be not feasible, motivating the need for a novel algorithm. The proposed recursive linking algorithm decomposed the pool of genetic markers into segments and rendered the model implementable for a large number of genetic markers. The recursive algorithm has been shown to reduce the order of time complexity from exponential to linear. The improvement in time complexity has been shown theoretically by a worst-case analysis of the algorithm and supported by run time results using data on linkage group-I of the fungal genome *Neurospora crassa*.

ACKNOWLEDGEMENTS

The research was supported in part by a grant from the U.S. Dept. of Agriculture under the NRI Competitive Grants Program (Award No: GEO-2002-03590) to Drs. Bhandarkar and Arnold. We thank the College of Agricultural and Environmental Sciences, University of Georgia for their support.

References

1. Lander ES, Schork NJ. Genetic dissection of complex traits. *Science* 1994; **265**: 2037–2048.
2. Doerge RW, Zeng ZB, Weir BS. Statistical issues in the search for genes affecting quantitative traits in experimental populations. *Statistical Science* 1997; **12**: 195–219.
3. Raju NB. Meiosis and ascospore genesis in *Neurospora*. *Eur.J.Cell.Biol.* 1980; **23**: 208–223.
4. Nelson MA, Crawford ME, Natvig DO. Restriction polymorphism maps of Neurospora crassa: 1998 update. *http://www.fgsc.net/fgn45/45rflp.html* 1998; :
5. Lander ES, Green P. Construction of multi-locus genetic linkage maps in humans. *Proc.Natl.Acad.Sci.USA* 1987; **84**: 2363–2367.
6. Barratt RW, Newmeyer D, Perkins DD, Garnjobst L. Map construction in *Neurospora* crassa. *Advances in Genetics* 1954; **6**: 1–93.
7. Mester D, Romin Y, Minkov D, Nevo E, Korol A. Constructing large-scale genetic maps using an evolutionary strategy algorithm. *Genetics* 2003; **165**: 2269–2282.
8. Cuticchia AJ, Arnold J, Timberlake WE. The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. *Genetics* 1992; **132**: 591–601.
9. Dempster A, Laird N, Rubin D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society ,Series B* 1977; **39:1**: 1–38.
10. Rao CR.Linear Statistical Inference and Its Application, 2nd ed. Wiley-Interscience. 2002.
11. Thomas HC, Charles EL, Ronald LR, Clifford S.Introduction to Algorithms, 2nd ed. The MIT Press. 2001.