A method and program for estimating graphical models for linkage disequilibrium that scale linearly with the number of loci, and their application to gene drop simulation

Alun Thomas
Department of Biomedical Informatics, University of Utah.
Genetic Epidemiology, 391 Chipeta Way Suite D, Salt Lake City, UT 84108, USA

ABSTRACT

Motivation: Efficient models for genetic linkage disequilibrium are needed to enable appropriate statistical analysis of the dense, genome wide single nucleotide polymorphism assays currently available.

Results: Estimation of graphical models for linkage disequilibrium within a restricted class of decomposable models is shown to be possible using computer time and storage that scale linearly with the number of loci. Programs for estimation and for simulating from these models on a whole genome basis are described and provided.

Availability: Java classes and source code for IntervalLD and GeneDrops are freely available over the internet at http://bioinformatics.med.utah.edu/~alun.

Contact: alun@genepi.med.utah.edu

1 INTRODUCTION

Linkage disequilibrium, or LD, is the non-independence of alleles at proximal genetic when the recombination process along chromosomes has not had enough time to randomize their states. It is a form of allelic association other forms of which can also arise due to selection, relatedness of individuals, and population admixture. Statistical methods that do not take LD into account can be misled into false results. In sparse sets of genetic loci, its effects can be negligible, however, with the dense genetic assays using single nucleotide polymorphisms, or SNPs, currently available, it is critical to properly account for it. Several approaches have been employed for modelling LD, in particular graphical modelling has been used extensively in this context. Verzilli et al. (2006) estimated graphical models for correlated genotypes at proximal loci, while Scherf and Stephens (2006) defined graphical models on variables indicating the cluster of origin of alleles and genotypes. Thomas and Camp (2004), Thomas (2005), Thomas (2007), and Thomas (2009) developed methods for estimating graphical models in which the variables are the alleles themselves, and it is this development that we continue here. The estimation approach presented here is similar to that of Greenspan and Geiger (2004). Their use of the EM algorithm in addressing the missing data problem of unknown phase is similar to the two phase approach described below which may be thought of as a stochastic version of EM where the E-step uses a random imputation instead of a conditional mean. The classes of graphical models used are, however, quite different. Where Greenspan and Geiger (2004) use a Bayesian network to explicitly model the haplotype block structure of the genome, this work models the data in a purely empirical manner using the physical relationships between the loci only to restrict the class of decomposable graphical models that can be considered.

A graphical model for a set of random variables is based on a factorization of their joint distribution as

\[ P(X_1, \ldots, X_n) = \prod_i f_i(T_i) \]

where \( T_i \subset \{X_1, \ldots, X_n\} \), and each \( f_i \) is some non-negative function of a subset \( T_i \) of the variables. From this we define a graph in which each variable is a vertex, and pairs of vertices are connected by edges if they are both contained in the same \( T_i \). This is called the conditional independence graph or Markov graph as it allows the conditional independence relationships between the variables defined by the factorization to be easily read off.

Højsgaard and Thiesson (1995) originally developed methods for estimating graphical models for discrete random variables by maximizing a penalized likelihood function. The penalty, based on the number of parameters, is necessary to avoid fully saturated models. Giudici and Green (1999) developed estimation of graphical models on Gaussian variables as did Dobra et al. (2003) and Jones et al. (2005). Thomas and Camp (2004) adapted the method of Højsgaard and Thiesson (1995) for estimating graphical models for LD from phase known haplotype data, replacing the original deterministic model search with the random methods of Metropolis sampling (Metropolis et al., 1953) and simulated annealing (Kirkpatrick et al., 1982). Thomas (2005) extended this approach to use unphased genotype data using a two stage method. Given an initial model, usually the trivial one that represents linkage equilibrium, and the observed genotypes, complete phase known haplotypes are imputed for the sampled individuals. Then, given these imputed haplotypes, the graphical model is re-estimated. Given a new graphical model, the haplotypes are re-imputed, and so on.

The above methods restrict the search for graphical models to those whose conditional independence graphs are decomposable or,
equivalently, chordal or triangulated (Golumbic, 1980). It is this
property that allows the likelihood and degrees of freedom of a
graphical model to be computed. Decomposable graphs, however,
are not easily characterized. The random search methods described
above typically propose changes to an incumbent graph such as
adding or deleting an edge. Before evaluating the penalized likeli-
hood for the proposed graph, it is necessary to first check that it is
decomposable. While Giudici and Green (1999) give efficient
methods for this, in large graphs the probability that a random pro-
posal will be decomposable decreases rapidly, ultimately making the
search procedure very inefficient. Thomas (2009) circumvented this
problem by restricting the search to the easily characterized subclass
of models whose conditional independence graphs are interval gra-
phs, as defined below. This restriction was shown to greatly improve
the search efficiency, without sacrificing power to appropriately
model LD.

In this work we add one further model restriction that enables a
walking window approach to estimation of LD between the alleles
along a chromosome. This is implemented in a program whose
storage and time requirements are linear in the number of loci
considered as we illustrate on examples of over 200000 markers
taken from the HapMap YRI data set (The International HapMap
Consortium, 2005).

While Thomas (2007) showed that graphical models for LD
could, in principle, be used for linkage analysis, full multi point
linkage analysis is not feasible with dense SNP data. However, it
is feasible to use haplotypes generated using LD models in simula-
tion methods to assess significance in association studies involving
unrelated cases and controls. It is also feasible to generate haplo-
types for pedigree founders using the models and then simulate the
descent of the alleles to descendants using the multi locus gene drop
method. These simulations can be used for assessing the statisti-
cal significance of long runs of allele sharing that are used in the
method of mapping by shared genomic segments introduced by of
Thomas et al. (2008) and Leibon et al. (2008). With this in mind, we
have written a program to perform gene drop simulation with linked
markers in LD.

In what follows we briefly review estimating graphical models
from genotype data and describe the role of interval graph in this
context. We then describe the implementation of this approach in
a walking window along the chromosome. We also describe briefly
how the programs implementing these methods can be used. Finally,
we illustrate the linearly scaling performance of the program with
data from HapMap.

2 METHODS

2.1 Estimating graphical models

A graphical model is decomposable if and only if the maximal cliques,
$C_1, \ldots, C_n$, of its conditional independence graph can be ordered so that
the following running intersection property holds:

$$ S_i = C_i \cap \bigcup_{j=i+1}^{n} C_j \subseteq C_k \text{ for some } k > i. $$

The sets $S_i$ are called the separators of the graph, by convention $S_n = \emptyset.$

The joint probability distribution can then be expressed as a function of the
clique and separator marginals:

$$ P(X_1, \ldots, X_n) = \prod_i P(C_i) P(S_i) $$

This then allows the calculation of the maximized log likelihood and degrees
of freedom for the graphical model as

$$ \log(\hat{L}(G)) = \sum_i \log(\hat{L}(C_i)) - \sum_i \log(\hat{L}(S_i)) $$

and

$$ df(G) = \sum_i df(C_i) - \sum_i df(S_i) $$

respectively, from which we can obtain the penalized likelihood or informa-
tion criterion

$$ IC(G) = \log(\hat{L}(G)) - \alpha df(G). $$

In the case of discrete data with no missing values, the clique and separator
marginals are simply contingency tables for which the degrees of freedom
and maximized likelihood are easily calculated.

As noted above, we can optimize $IC(G)$ using random search methods
in which we make small changes to $G$ to obtain $G'$ which then must
be checked for decomposability and subject to the usual Metropolis or
simulated annealing rejection step.

In order to estimate graphical models from genotype data, we must first
impute complete phase known data under an initial model. For this we assume
linkage equilibrium which is equivalent to a graph $G$ with no edges. The
above random search method is then run for some number of iterations and the
resulting graphical model and maximum likelihood parameter estima-
tes are used to obtain new imputations for the complete phase known data.
Details of this process are given by Thomas (2005).

2.2 Interval graphs

Under a perturbation scheme that simply adds or deletes edges from the
incumbent graph, as the number of vertices increases the probability that
a random proposal $G'$ is decomposable decreases. Thomas (2009) shows
that for the LD problem, the probability decreases approximately as $1/n$ but
that this can be avoided by restricting the conditional independence graphs
to be interval graphs. A graph is an interval graph if and only if the vertices
can be made to correspond to intervals of the real line such that two vertices
are connected if and only if their corresponding intervals overlap. This has
some intuitive appeal for the LD problem because loci are ordered linearly
along a chromosome, and we expect that LD will decay with distance. In
order to reflect this, we assign each locus a point on the line and require its
corresponding interval cover its location. In this application the points are
evenly spaced in chromosomal order, but could be made to reflect the phy-
sical distances between loci. Thomas (2009) also required that in order for
two vertices to be connected, their intervals must overlap by a minimum,
non-zero amount. This allows some flexibility for loci positioned between
two correlated loci, but which appears to be stochastically independent, to
be assigned a small interval and hence avoid a forced connection with one of
the flanking loci.

Interval graphs are a subclass of decomposable graphs (Golumbic, 1980),
and it is easily shown that the additional restriction described above still
define interval graphs. Moreover, the characterization of the graph struc-
ture in terms of intervals of the line make it simple to propose new graphs
in the same class without having to check for decomposability. For exam-
ple, if we propose changing the length of an interval the resulting perturbed
graph is obviously still an interval graph, and hence decomposable. Thomas
(2009) showed that this leads to considerable computational efficiencies, and
that the haplotype frequencies from models with interval graphs do not dif-
fer greatly from those under general decomposable models, nor from those
implied by the models of Scheet and Stephens (2006).

In order to store and manipulate the intervals, Thomas (2009) used a stan-
dard structure called an interval tree (de Berg et al., 2000). This structure
allows addition and deletion of intervals and queries as to which intervals overlap with a given one to be carried out in, at best, \(O(\log(n))\) time. Together with the the required storage manipulations this resulted in super linear time requirements for large data sets of 10,000 loci or more. To overcome this, we now introduce a final model restriction: we require that the interval representing a locus extends no more that some maximum value \(\mu\) to each side of its associated point. This allows the intervals to be stored in a simple array ordered by the position of the required point. To identify connections between the vertices in the corresponding graph, shown below. Note that the interval corresponding to locus 3 is shorter than the minimum required overlap, shown on the left hand side, and hence the vertex in the corresponding graph has no connections. Similarly, the overlap between the intervals for loci 9 and 10 is insufficient to create a link. Note also that the graph shown here and the one in figure 2 are both decomposable which is a consequence of their derivation from the interval representation.

![Interval representation](image1)

**Fig. 1.** The relationship between the interval representation of an interval graph and the graph itself. A box represents the interval assigned to a locus which is constrained to include the locus’s assigned position, shown as a line crossing the box. The whiskers represent the maximum extent allowed for the interval. Boxes that overlap by at least the minimum requirement lead to connections between the vertices in the corresponding graph, shown below. Note that the interval corresponding to locus 3 is shorter than the minimum required overlap, shown on the left hand side, and hence the vertex in the corresponding graph has no connections. Similarly, the overlap between the intervals for loci 9 and 10 is insufficient to create a link. Note also that the graph shown here and the one in figure 2 are both decomposable which is a consequence of their derivation from the interval representation.

![Corresponding graph](image2)

2.3 Graph updates in a window

The first stage of the search method involves searching the space of interval graphs given fixed, imputed haplotypes. In doing this we restrict the vertices being considered to those within a contiguous window of the line. We propose an update to the incumbent graph, \(G\), say, by choosing a vertex in the window and perturbing its corresponding interval by generating new random end points each side of the fixed point. The new distances to the end points are generated independently from the Uniform\((0, \mu)\) distribution. The proposed graph, \(G'\), is then either accepted as the next incumbent, or rejected, based on the value of the information criterion. Thomas (2009) showed that both the likelihood and the degrees of freedom of \(G'\) can be evaluated by considering the values for \(G\) and the small subgraph within the maximum extent of the interval being updated. This is independent of the number of loci and hence very fast.

![Window of variables corresponding to changed intervals](image3)

**Fig. 2.** The vertices possibly affected by updates to the graph in a fixed window. The intervals for the vertices in the window, the black circles in the solid box, can be changed by generating new end points. Edges between these and the grey vertices in the dotted box may change as a result, and these need to be considered in the new calculation of maximized likelihood and degrees of freedom. No part of the graph outside the dotted box can change until the window moves.

Although we limit the intervals updated to a fixed window, the effects can extend to either side of the window. This is illustrated in figure 2. In the example shown, a window of 5 variables is being updated, however, as figure 1 shows, the maximum length of an interval is 3.4 units, where a unit is the distance between two consecutive markers. Hence, a vertex may be connected to the any of the three previous or next vertices. The effects of changes to the window of 5 vertices may, therefore, extend to an enclosing window of 11. However, outside of this, no edges can change. In the implementation described below, the walking window covers 100 vertices at a time and an interval can extend be up to 10 units long.

2.4 Phase imputations in a window

The second stage of the search is to update the imputed haplotypes given the current model and the observed genotypes. Again we do this in a restricted contiguous window of loci. The current graphical model is applied to both the paternal and maternal haplotypes for each observed individual. The values at each locus determine the observed genotypes. Thus, we obtain a compound graphical model connecting haplotypes to genotypes as shown in figure 3. A random imputation for the haplotypes can now be made using the usual forward-backward graphical modelling methods as described by Thomas (2009). This requires determining from the graph an ordering of the variables to be updated. The forward step then proceeds through this list calculating the conditional distribution of the state of each given the states of those that appear later in the list. The conditional independences implied by the graph mean that this conditional distribution typically depends on only a small subset of the remaining variables so that this is usually a quick computation. The backward step then moves through the same list in reverse order using the conditional distributions previously calculated to simulate a state for each variable given the states of those already determined.

Note that in making these updates, we change only values in the current window, however, these may depend on the values of alleles at loci bordering the window, as shown in figure 3.

While the graphical models for the haplotypes, figure 2, are guaranteed by the interval graph representation to be decomposable, the compound graphical models, figure 3, are typically not. Consequently, to find the ordering of the vertices needed to make the forward-backward steps described above, we first need to find a triangulation of the graph within the window, and hence find a decomposition. This step is super linear in the time required, and this is why we implement the walking window approach. Initial testing showed that the computational time required to make this update grows approximately as \(\omega \log(w)\) where \(\omega\) is the length of the window.
Variables that may need to be conditioned on

Fig. 3. The graphical model shown in figure 2 is applied to the paternal and maternal haplotypes of each individual in the sample in parallel. These affect the observed genotypes shown as white squares. The states of the black vertices are updated conditional on the genotypes and the current graphical mode. This may require conditioning on states of vertices outside the window, as shown in grey.

3 IMPLEMENTATION

3.1 Model estimation

The above methods have been implemented in a program called IntervalLD that is available as part of the author’s packages of Java programs for graphical modelling and genetic analysis. The program takes input in the same standard file formats as the much used LINKAGE programs (http://linkage.rockefeller.edu). Two files are input: one specifying the nature of the genetic loci being analyzed, the second giving a list of individuals and their genotypes. The format allows for specifying pedigree relationships between the individuals, but also, by listing the parents as zero, allows for samples of unrelated individuals as required by these methods. IntervalLD will treat all individuals as unrelated even if relationships are specified.

The program is run using the following call

```
java IntervalLD in.par in.ped [w] [p] [g]
```

where

- `in.par` and `in.ped` are the LINKAGE format input files described above,
- `w` is the width of the window of loci to be considered. The default is 100.
- `p` is the number of phase updates to make in each window. The default is 5.
- `g` is the number of sweeps of graph updates to try between each phase update. The default is again 5.

In the above and what follows, arguments in square brackets – [] – are optional.

Having read in the data and set up the appropriate data structures, the program makes an initial imputation of phase based on the assumption of linkage equilibrium.

Then, one round of random sampling is made as follows. In each window, `g` sweeps are made of the loci in order, randomly perturbing the corresponding interval by proposing new end points, calculating the likelihood and degrees of freedom of the new graph, and either accepting or rejecting the change based on Metropolis acceptance probabilities. After `g` sweeps are made, a new phase imputation is made in the window. This new imputation is randomly chosen given the updated graphical model and the observed genotypes. This is repeated `p` times, with `g` Metropolis sweeps between each imputation. The window is then advanced along the chromosome by one half window length and the above process is repeated until the end is reached.

Following the round of random sampling a round of random uphill optimization is made. This follows the same format as the random sampling, but the Metropolis sampling of the graph is replaced by an uphill search: the proposed graph is accepted only if it is as good as or better than the current. Also, instead of making random phase imputations, imputations are made by choosing the most probable haplotypes. As with the random version, this is done using standard graphical modelling forward-backward methods.

The multinomial parameters of the resulting graphical model are then output to a file. The file format is straightforward and human readable, although it is intended primarily for input into other programs. It is a list of the conditional distributions of the state of the alleles at each locus given the values at the loci that the graphical model defines as relevant.

3.2 Gene drop simulation under LD

Gene drop is a method for simulating the genotypes of related individuals in a pedigree. Alleles are allocated to founders at random, and these are then dropped down the pedigree mimicking Mendelian inheritance until the genotypes of all individuals are allocated. The single locus version is described by MacCluer et al. (1986). The multi locus version is similar, the difference being that the probabilities of inheritances at successive loci depend on the recombination fraction between them. An implementation of multi locus gene drop is given in the MERLIN program (Abecasis et al., 2001).

The implementation given here differs only in that the founder alleles are allocated by simulating haplotypes from a graphical model as estimated from control data using IntervalLD. The program is called as follows:

```
java GeneDrops in.par in.ped n pfx [ldmod] [-a]
```

where

- `in.par` and `in.ped` are again LINKAGE format input files. In this case the pedigree structure specified in `in.ped` is used for simulations.
- `n` is the number of simulations to perform.
- `pfx` is a prefix used in naming the output files. For example if `n` is 10 and `pfx` is `gdout` then the output files will be named `gdout.01`, `gdout.02`, ... `gdout.10`. Each of these files will be a LINKAGE pedigree differing from `in.ped` only in that the genotypes given by them are simulations rather than the actual observations.
- `ldmod` is the name of the file containing the graphical model for LD. Note that this can be omitted in which case standard gene drop simulations are made under the assumption of linkage equilibrium using the alleles frequencies given in `in.par`.
- `-a` is an optional argument that determines what to output. By default genotypes are only output to match those observed, as specified in the `in.ped` file: that is, if a genotype is unspecified in the input file it will be unspecified in the output file also. If the `-a` option is given, however, complete genotypes are output for all individuals.

4 RESULTS

To illustrate the linear scaling of IntervalLD and GeneDrops we ran both programs on SNP data for chromosome 1 downloaded from HapMap. We used the YRI data which are the genotypes of 30 parent-offspring trios of Yoruba people from Ibadan, Nigeria. For model estimation we used only data on the 60 unrelated parents. This first required rewriting the downloaded files to make LINKAGE format input files. We then selected subsets of different numbers of loci, the smallest being 100 the largest being all 223110
loci available for chromosome 1. All the results described here were obtained using the author’s lap top computer running Java 1.5.0_02 under Linux. The machine has two 2.8 GHz processors and 4 Gbytes of random access memory.

Figure 4 shows the times taken in seconds and the storage required in Mbytes for model estimation using IntervalLD with the default parameters described above. Also given are the times needed to make 100 gene drop simulations for a small three generation pedigree. The pedigree consisted of a sibship of 10 children, their parents and grandparents. The linear scaling for all these measures is clear.

Figure 5 shows how the resources required for LD model estimation change with the size of the sample. These results were obtained from artificial data sets obtained by duplicating and reduplicating the 60 individuals of the YRI data described above. In each case 10000 loci were modelled. This again shows clear linear scaling.

Thomas (2009) showed that the haplotype frequencies implied by graphical models for LD were similar to those modelled by the fastPHASE program of Scheet and Stephens (2006). To further investigate this we tested the program’s ability to impute missing genotypes from the haplotype model. For each of the first 600 loci, we deleted the genotype of one individual, so that 10 genotypes per individual were deleted in all. We then ran IntervalLD on this data and output the final genotypes imputed by the program for the missing values. The imputed values matched the actual values in 88.5% of cases. As a comparison, we also applied fastPHASE to the problem using its default parameter settings. This was correct 93.5% of the time.

5 DISCUSSION

Current human SNP genotyping assays obtain genotypes on around one million loci for the individuals assayed. As the chromosomes segregate independently, modelling of LD and simulation can be performed separately for each chromosome. The largest number of loci that need to be considered jointly, therefore, are those on chromosome one, the largest chromosome. This number is under 100000 and well within the range considered here. The methods and programs described can therefore be applied to estimation and simulation problems on a genome wide scale. Model estimation of the data set of 223110 loci took just over 11 hours, from which we estimate that an analysis for a million loci would take around 50 hours. The limiting factor here is the storage space needed. Just over 1.8 Gbytes were required for the complete set of 223110 loci.

While the recent increases in the number of loci assayed is dramatic, the computational resources also, clearly, depend on the sample size. More generally we would expect the requirements to scale as $O(nmk)$ where

- $n$ is the number of loci,
- $m$ is the number of individuals assayed
• $k$ is the average complexity of the graphical models considered.

The complexity of a graphical model on binary variables can be measured by $\sum_i 2^{c_i}$ where $c_i$ is the number of variables in the $i$th clique, and the sum is over all cliques. To isolate the effects of increasing sample size and obtain figure 5, the $\alpha$ parameter of equation (1) was manipulated so that the model complexities remained similar. In reality, as the sample size increases weaker interactions may become detectable and the complexity of the graphical models may increase. Hence, in some circumstances super linear scaling may occur as the sample size grows. While this can be fixed by increasing $\alpha$ and enforcing parsimony, care should be taken not to oversimplify the models found. For larger samples, therefore, it might be necessary to break up the data further, perhaps handling the arms of the larger chromosomes separately.

Some experimentation with the parameters as used in the above comparison with the fastPHASE program, showed that results were not particularly sensitive to the choice of the $\alpha$ value for the maximum extent of the intervals. A maximum interval of 5 units each side of the locus allows the alleles at the locus to depend on up to 10 loci on each side. However, dependence on more than 5 was rarely seen. On the other hand the choice of $\alpha$ in equation (1) had a greater effect. The original setting was $\frac{1}{2} \log(h)$ where $h$ is the number of chromosomes in the sample which is in accordance with the Bayesian information criterion of Schwarz (1978). However, better performance was seen with a far lower value that allowed for larger clique sizes. The default value is now set at $\alpha = \frac{1}{17} \log(h)$. We note that while the performance of IntervalLD on the imputation test gave good results, correctly imputing 88.5% of missing genotypes, this was not quite as good as fastPHASE which correctly imputed 93.5%.

Gene drop simulation has many possible applications, however, this work was primarily motivated by its use in assessing the statistical significance of allele sharing in relatives in identity by descent gene mapping strategies. Such methods have been described by Thomas et al. (2008) and Leibon et al. (2008). These authors recognize that LD is likely to increase the lengths of random runs, or streaks, of allele sharing, potentially leading to false positive results. The methods and programs described here can be directly applied to this problem.

**ACKNOWLEDGMENT**

This work was supported by grants NIH R01 GM81417, and DOD W81XWH-07-01-0483 to Alun Thomas. I would like to thank three anonymous referees for their constructive criticisms and suggestions.

**REFERENCES**


Thomas, A. (2009). Estimation of graphical models whose conditional independence graphs are interval graphs and its application to modeling linkage disequilibrium. *Computational Statistics and Data Analysis*. Published on line.

