Journal of Bioinformatics and Computational Biology
Vol. 8, Suppl. 1 (2010) 127–146
© The Authors
DOI: 10.1142/S0219720010005142

Imperial College Press
www.icpress.co.uk

EFFICIENT MINING OF HAPLOTYPE PATTERNS FOR LINKAGE DISEQUILIBRIUM MAPPING

LI LIN*, LIMSOON WONG† and TZE-YUN LEONG‡

School of Computing, National University of Singapore
13 Computing Drive, Singapore 117417, Singapore
**hi.linli@gmail.com*
†*wongls@comp.nus.edu.sg*
‡*leongty@comp.nus.edu.sg*

POH SAN LAI

Department of Paediatrics, Yong Loo Lin School of Medicine
National University of Singapore, NUHS Tower 12
1E Kent Ridge Road, Singapore 119228, Singapore
poh_san_lai@nuhs.edu.sg

Received 15 July 2010
Revised 25 August 2010
Accepted 10 September 2010

Effective identification of disease-causing gene locations can have significant impact on patient management decisions that will ultimately increase survival rates and improve the overall quality of health care. Linkage disequilibrium mapping is the process of finding disease gene locations through comparisons of haplotype frequencies between disease chromosomes and normal chromosomes. This work presents a new method for linkage disequilibrium mapping. The main advantage of the proposed algorithm, called LinkageTracker, is its consistency in producing good predictive accuracy under different conditions, including extreme conditions where the occurrence of disease samples with the mutation of interest is very low and there is presence of error or noise. We compared our method with some leading methods in linkage disequilibrium mapping such as HapMiner, Blade, GeneRecon, and Haplotype Pattern Mining (HPM). Experimental results show that for a substantial class of problems, our method has good predictive accuracy while taking reasonably short processing time. Furthermore, LinkageTracker does not require any population ancestry information about the disease and the genealogy of the haplotypes. Therefore, it is useful for linkage disequilibrium mapping when the users do not have such information about their datasets.

Keywords: Linkage disequilibrium mapping; pattern mining; haplotypes.

1. Introduction

Linkage disequilibrium mapping is a process of inferring the disease gene location from observed associations of marker alleles[a] in affected patients and normal controls. The main idea of linkage disequilibrium mapping is to identify chromosomal regions with common molecular marker alleles at a frequency significantly greater than chance. The task of linkage disequilibrium mapping becomes increasingly challenging with the presence of rare variants that are not well tagged by single markers, as well as when the percentage of occurrence of haplotypes with the mutation of interest is very low and with inclusion of errors or noise.

Some existing methods for linkage disequilibrium mapping include Blade,[1] GeneRecon,[2] Haplotype Pattern Mining (HPM),[3,4] and HapMiner.[5]

Liu *et al.* proposed Blade which employs the Markov Chain Monte Carlo method (MCMC) for parameter estimation within a Bayesian framework. The disease haplotypes are grouped into $k+1$ clusters, corresponding to k founder chromosomes in the disease population and a null cluster for all other disease chromosomes. Blade assumes that the disease haplotypes within each cluster are mutually independent given the ancestral haplotype. This alleviates the need for a complex model of the underlying genealogy. However, Blade assumes that all mutations occur in the same location of the disease gene, which means that locus heterogeneity is not incorporated.

To address some of the shortcomings in the algorithm proposed by Liu *et al.*, Mailund *et al.* proposed an algorithm known as GeneRecon. GeneRecon combines the shattered coalescent method by Morris *et al.*[6] and the idea by Liu *et al.* in separating the affected individuals into mutation clusters. The idea of shattered coalescent is to consider genetic heterogeneity at the disease locus by allowing branches of the genealogical tree to be removed. Thus, single leaves correspond to sporadic case chromosomes, and disconnected subtrees correspond to distinct mutations at the disease locus. GeneRecon combines the shattered coalescent method with the idea by Liu *et al.* in separating affected individuals into mutation clusters with a null cluster for individuals affected not due to genetic factors. Although GeneRecon is highly efficient in locating the disease locus on case-control data, its main drawback is that it is computationally intensive and requires several hours or even days for a successful computation on a dataset with a few hundred cases and controls, and with few tens of markers.

Tiovonen *et al.* introduced a linkage disequilibrium mapping algorithm known as Haplotype Pattern Mining (HPM). HPM first uses an association rule mining algorithm (Agrawal & Srikant[7]) to find all patterns that occur in at least a specified percentage (called the support threshold) of the haplotype samples. Then HPM

[a]A molecular marker is an identifiable physical location on the genomic region. An allele is any one of a series of two or more alternate forms of the marker. From the data mining aspect, we could represent markers as attributes, and alleles as attribute values that each attribute could take on.

uses the signed chi-square test on these frequent haplotype patterns to identify those patterns that are significant for discriminating disease association from the control association. Finally, HPM adds up the frequency of each marker's occurrence across these significant haplotype patterns. The marker with the largest frequency is predicted as closest to the disease gene. The main drawback of this algorithm is that it suffers from combinatorial explosion in the number of patterns due to its exhaustive search method. As highly associated patterns are rare in the problem of linkage disequilibrium mapping, the support threshold will need to be set at a very low value to discover those patterns; thus, many useless patterns will also be discovered together with the highly associated patterns.

Li and Jiang proposed an algorithm known as HapMiner for inferring disease gene location. HapMiner is an adaptation of the DBSCAN[8] algorithm, which is a density-based clustering method that is robust to noise. Density-based clustering methods are characterized by a set of parameters that specify the clustering process, input and output, including the density and size of the clusters. The parameter values involved, however, are usually difficult to determine automatically and require direct user input. The first two parameters of HapMiner are ε which specifies the radius of the interested neighborhood, and the density threshold MinPts. The advantages of HapMiner are: Firstly, it is a model-free algorithm which does not rely on any prior information about the genealogy of haplotypes and the inheritance patterns of the diseases. Secondly, the time complexity of HapMiner is very low, which means that it can perform disease gene location inference at a very high speed. HapMiner is shown to outperform algorithms such as HPM and Blade. However, the main disadvantage of HapMiner is that it is very sensitive to its parameter values, many of which the user needs to obtain by trial and error.

To address some of the problems of these existing algorithms, we propose an algorithm known as LinkageTracker; some preliminary results on LinkageTracker were published in Lin *et al.*[9] Comparing to our earlier work in Lin *et al.*, this work compared LinkageTracker with more linkage disequilibrium algorithms including Blade, GeneRecon and HapMiner, and we have also included more experimental results through applying the algorithms on two real datasets. Furthermore, we have introduced a new mechanism for fast convergence of the candidate pattern sets to small number of patterns to improve computational efficiency.

LinkageTracker is model free — it does not require any population ancestry information about the disease and the genealogy of the haplotypes. Furthermore, LinkageTracker does not require the setting of complex parameters prior to the disease gene location inference process. Extensive performance studies show that for a large class of practical problems, the predictive accuracy of LinkageTracker is consistently good under different conditions — from the extremely difficult condition where the samples with the mutation of interest are as low as 10% and with high noise level, to the easier condition where the samples with the mutation of interest are as high as 50%.

2. Methods

There are two main steps in the LinkageTracker algorithm. Step 1 identifies a set of linkage disequilibrium patterns which can effectively discriminate between the abnormal and the normal alleles. Step 2 infers the marker allele that is closest to the disease gene based on the linkage disequilibrium patterns derived in Step 1.

2.1. *Step 1: Discover linkage disequilibrium patterns*

The process of the discovery of linkage disequilibrium patterns begins with searching for potential/candidate patterns using a level-wise neighborhood search method. Then each potential/candidate pattern is scored using a statistical method.

2.1.1. *Level-wise neighborhood search for candidate patterns*

LinkageTracker mines patterns of the form $\langle d_{xi}, d_{xj}, \ldots, d_{xk} \rangle$ where d_{xi} indicates the allele value for marker i of a particular biological sample x. For example, (3, 5, 6, *, *, 4) is a marker pattern of length 4. The symbol "*" represents missing or erroneous marker allele, and will not be considered when testing for the significance of the pattern. Also the symbol "*" is ignored when computing the length of a marker pattern.

The "*" symbol also indicates a gap between two known marker alleles. For instance, the marker patterns (1, *, *, *, 3) has three gaps, and (1, *, 3) has one gap. The user is able to set the maximum number of gaps for the marker patterns. However, we recommend that the maximum allowable gap be set to 6, which gives the highest accuracy when the markers are spaced at 1 cM[b] or less. The basis for this recommendation can be found in our earlier work Lin *et al.*; thus it will only be described briefly here.

To find linkage disequilibrium patterns, one way is to enumerate all the possible marker patterns of length one, two, and three, etc, and then compute the *odds ratio* of each pattern and select those patterns that are significant. However, there are some practical difficulties to this approach: for n markers each with m alleles, there are ${}_n\boldsymbol{C}_k m^k$ marker patterns of length k, which we need to test for significance. Combinatorial explosion occurs as the length of marker patterns increases.

The enumeration of all possible marker patterns is in fact unnecessary. This is because, based on studies by Long and Langley,[10] allelic associations are detectable within a genomic region of 20 cM. Allelic associations beyond 20 cM are weak and are not easily detectable. Depending on the total region size of the markers under study, the detectable genomic region may vary. Let us denote the genomic region that is capable of detecting allelic associations to be β cM. LinkageTracker uses a

[b]cM stands for centimorgan. It is the unit of measurement for genomic distance. In human genome, one centimorgan is approximately equivalent to 1 million base pairs.

heuristic search method by controlling the maximum allowable gap size between two marker alleles. The gap size setting Ψ helps to define the search space of LinkageTracker as well as to ensure its robustness against noise. We propose a scoring method to determine the optimal gap size setting as

$$\Psi = \max_{g} \left(Score\left(g\right) \right) \tag{1}$$

where g is a gap size, and $Score(g)$ can be computed as follows:

$$Score(g) = \frac{\sum_{i=0}^{g} Robustness_i}{\sum_{i=0}^{g} Noise_i} \tag{2}$$

$Robustness_i = p_i * i$ and $Noise_i = q_i$, where p_i is the number of informative patterns formed with exactly i gaps and q_i is the number of confounding patterns formed with exactly i gaps. An informative pattern is formed when a significant marker pattern joins with its neighboring markers that are within the β cM region. A confounding pattern is formed when a significant marker pattern joins with markers that are beyond the β cM region. An example of pattern joining is illustrated below.

LinkageTracker adopts a heuristic level-wise search method which allows only significant marker patterns (or linkage disequilibrium patterns) of length $i-1$ at level i to join with their neighbors (of length 1) whose join satisfies the maximum gap constraint Ψ to form candidate/potential marker patterns of length i, where $1 \leq i \leq n$ and n is the number of markers. We call the procedure of joining linkage disequilibrium patterns at each level to form longer patterns the *neighborhood join*. Note that in *neighborhood join*, only the marker patterns of length $i-1$ need to be significant, the neighbors that they join with need not be significant and may be several markers apart.

A marker allele exhibits significant allelic association with the disease gene under two conditions. Firstly, it is significant on its own when tested (i.e. at level 1). Secondly, when joined with other marker alleles that exhibit allelic associations with the disease gene, the joined pattern becomes significant when tested.

The former condition is trivial to detect. The latter condition involves a marker allele which shows significant allelic association with the disease gene when joined with other significant marker alleles, but is insignificant when assessed alone. Let us denote this marker allele *Mx*. This problem can be further divided into two cases.

The first case is that *Mx* is close to a neighbor *Mi* that is significant when tested alone. The term "close" here means that *Mx* will be selected to join with *Mi* directly to form marker patterns for the immediate next level. For example, referring to Fig. 1, the two markers *Mx* and *My* are both not significant at level 1, hence they will be discarded when forming marker patterns for level 2. Now, *Mi* is an immediate neighbor of *My* showing significant allelic association at level 1. Hence, at level 2, *Mi* will join with its neighbors to form marker patterns of length 2. Since *My* is the immediate neighbor of *Mi*, *My* will be selected to form a pattern with

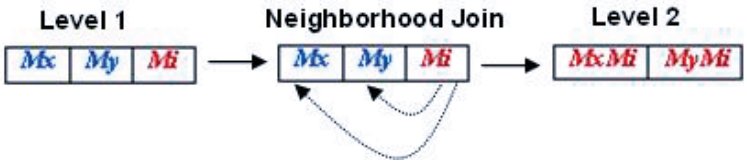
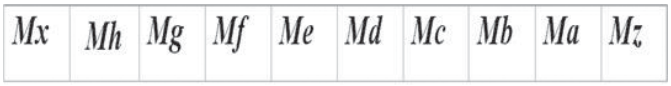


Fig. 1. Illustration of markers before and after joining.



Level	Join Patterns	Significant Patterns
1		(Mz)
2	(Mz,Ma), (Mz,Mb),(Mz,Mc)	(Mz,Mc)
3	(Mz,Mc,Md),(Mz,Mc,Me),(Mz,Mc,Mf)	(Mz,Mc,Mf)
4	(Mz,Mc,Mf,Mg), (Mz,Mc,Mf,Mh),(Mz,Mc,Mf,Mx)	(Mz,Mc,Mf,Mx)

Fig. 2. Illustration of marker positions.

Mi. Although *Mx* is one marker away from *Mi*, *Mx* will also be selected, because LinkageTracker allows joining with markers that are some gaps away. Hence, at level 2, both *My* and *Mx* are included in the marker patterns.

The second case is that *Mx* is very far from a marker allele *Mz* that is significant when tested alone. The term "far" here means that *Mx* is far enough from *Mz* such that *Mx* will not be selected by *Mz* to form marker pattern for the immediate next level. For example, from Fig. 2, *Mx* and *Mz* are eight markers apart. Assuming that the maximum allowable gap size is set to 2, *Mz* will join with *Ma*, *Mb*, and *Mc* to form patterns of length 2. Assuming that (*Mz*, *Mc*) is tested significant, then (*Mz*, *Mc*) will join with *Md*, *Me*, and *Mf* to form patterns of length 3. Assuming that (*Mz*, *Mc*, *Mf*) is tested significant, then (*Mz*, *Mc*, *Mf*) will join with *Mg*, *Mh*, and *Mx* to form patterns of length 4. Hence, *Mx* will ultimately be detected to form marker patterns under the condition that there are sufficient significant "intermediate" allele markers such as *Mc* and *Mf* to facilitate the detection of allelic associative marker alleles that are much further away (i.e. *Mx*). In accordance with the studies by Long and Langley, marker alleles exhibiting allelic associations with the disease gene are quite densely packed within the 20 cM region. Therefore, the chances of LinkageTracker detecting significant marker alleles within the range of 20 cM are relatively high.

2.1.2. *Scoring of candidate patterns*

LinkageTracker uses a statistical method known as *odds ratio* to score each potential/candidate pattern. Odds ratio provides a good measure of the magnitude

of association between a pattern and the binary label L, where $L = \{Diseased, Normal\}$.

The significance of the patterns is determined through comparing the pattern p-values to a threshold value α that is dynamically computed at different search levels. In general, if we have k independent significance tests at the α level, the probability p that we get no significant differences in all these tests is simply the product of the individual probabilities: $(1-\alpha)^k$. In order to guarantee that the overall significance test is still at the α level, Bonferroni Correction[11] is usually applied, that is, through dividing α by k to obtain the significance level for the individual tests. Bonferroni Correction requires the knowledge of the exact value of k which can be difficult to determine. LinkageTracker is an iterative process. At each iteration, the haplotypes may be tested for a different number of times which makes tracking k even more difficult. Furthermore, fast convergence of the candidate pattern set to a small number of patterns as the process iterates is desirable for computational efficiency and to filter out noisy patterns at early stages. As such, we devise a new mechanism to compute the p-value threshold to be used at each iteration or pattern length. We control the number of significant patterns at each iteration by setting the cut-off at the p-value of the t^{th} most significant patterns. The p-value threshold t decreases exponentially by dividing the number of patterns by a factor of 2 raise to the power of iteration $i+1$, as defined below:

$$t = \left\lfloor \frac{number\_of\_patterns\ at\ iteration\ i}{2^{(iteration\ i+1)}} \right\rfloor \tag{3}$$

2.2. *Step 2: Marker inference*

We infer the marker closest to the disease gene by combining the p-values of the highly significant patterns. Fisher's[12] method specifies that one should transform each p-value into $c = -2 * LN(P)$, where $LN(P)$ represents the natural logarithm of the p-value. The resulting n c-values are added together, and their sum, $\sum(c)$, represents a chi-square variable with $2n$ degree of freedom. For example, to find the marker closest to the disease gene, we compute the combined p-value and the frequency for each marker allele. In Fig. 3(a), Marker 2 has allele 4 occurring four times, its combined p-value is $1.4 * 10^{-6}$, for the chi-square distribution of $\sum(c) = 9.4211+10.0719+11.6183+10.8074 = 41.9186$ with 8 degrees of freedom. Figure 3(b) depicts the combined p-value for each of the marker alleles from Fig. 3(a). Marker 2 allele 4 has the lowest combined p-value, and hence we infer that Marker 2 is closest to the disease gene. If more than one marker alleles have the same lowest p-value, the marker with the highest frequency is selected as the marker closest to the disease gene.

Marker	1 2 3 4 5 6	*P-V*alue	c = -2 * ln(*P*)
Pattern01	* 4 3 * * *	0.0090	9.4211
Pattern02	2 4 * * 6 1	0.0065	10.0719
Pattern03	2 4 3 5 * *	0.0030	11.6183
Pattern04	* * 3 5 * 1	0.0100	9.2103
Pattern05	2 4 * 5 6 *	0.0045	10.8074

(a)

	Frequency	∑(c)	Combine *P-V*alue
Marker 1 allele 2	3	32.4975	1.3098E-05
Marker 2 allele 4	4	41.9186	1.4027E-06
Marker 3 allele 3	3	30.2497	3.5236E-05
Marker 4 allele 5	3	31.6390	1.9160E-05
Marker 5 allele 6	2	10.0719	0.0392
Marker 6 allele 1	2	19.2822	0.007

(b)

Fig. 3. (a) Example of 5 linkage disequilibrium patterns. (b) Combine p-value of each marker allele from (a).

3. Results on Real Datasets

We compared LinkageTracker with some leading methods in linkage disequilibrium mapping such as Blade, GeneRecon, and Hap-Miner on two real datasets, and 100 generated datasets. The unit used for the distance measure is cM for all the datasets. In this section we present a comparative analysis of the methods based on the real datasets. The experimental results on generated datasets are discussed in the next section.

Due to the slow processing speed of GeneRecon, this algorithm was assessed based on only five datasets for all the three experimental settings in this section. Whereas the rest of the algorithms were assessed based on 50 datasets for each percentage of founder mutation. However, the bias in the comparison of GeneRecon with other algorithms is very low because GeneRecon is very consistent in its predictions on similar datasets with low standard deviations.

3.1. *Cystic fibrosis*

Cystic fibrosis is a well-known real dataset reported in Kerem *et al.*,[13] used in a study based on SNP markers. The total region size for the dataset is 1.7298 cM, The dataset contains haplotypes on 23 bi-allelic markers around the cystic fibrosis trans-membrane conductance regulator gene. The control group has 92 haplotypes and the diseased group has 94. The founder mutation is located between markers 17 and 18, approximately 0.88 cM away from the leftmost marker. Only 67% of the disease haplotypes carry the founder mutation of interest. Furthermore, the disease haplotypes have about 39% of missing observations at certain markers.

In this dataset, we know exactly which are the disease haplotypes carrying the founder mutation of interest, and which are the disease haplotypes without the founder mutation. Therefore the dataset provides us with the opportunity to perform various rigorous experiments. For ease of reference, we divided the cystic fibrosis dataset into three subsets. Set-A consists of disease haplotypes carrying the founder mutation of interest, Set-B consists of disease haplotypes without the founder mutation of interest, and Set-C consists of haplotypes from the normal control group. There are in total 63, 31 and 92 samples in Set-A, Set-B and Set-C respectively.

3.1.1. *Experimental setting 1: Detection accuracies*

In this experiment we assessed the algorithms' capability in detecting the disease gene location when only a small portion of the disease haplotypes actually carries the founder mutation of interest, and others are genetically no different from the control population at the locus of interest. Therefore the datasets with different percentages of founder mutation carrying the disease haplotypes were generated (at 10%, 20%, 30%, 40% and 50%). For each percentage value we generated 50 different datasets each with 50 disease haplotypes and 50 controls.

For instance, to generate the disease haplotypes with 20% founder mutations, we randomly selected 10 founder mutation carrying disease haplotypes from Set-A, and mixed with 40 haplotypes randomly selected from control set Set-C (thus only 20% of the haplotypes actually carry the founder mutation). From the remaining 52 samples from Set-C, we randomly selected 50 samples to form the control haplotypes. This process was repeated 50 times to generate 50 datasets with 20% founder mutation. The datasets for other percentages of founder mutations were generated similarly.

For the algorithm HapMiner, we assessed its predictive accuracies based on the original parameter list provided by the authors (Li & Jiang) (they have used the same dataset in their work), and also based on the parameter list with the first two parameter values modified. The HapMiner algorithm is an adaptation of the density-based clustering method. As mentioned, the density-based clustering method is very sensitive to the values of the parameters, especially those that specify the radius of the neighborhood of interest, ε and the density threshold of the clusters, MinPts; the user also needs to guess the optimal parameter values. We tested HapMiner over a range of parameter values by varying 10% to 500% of the values as recommended in Li & Jiang, the experimental results are shown in Table 1. The resulting SSE fluctuated over a large range, showing the sensitivity of HapMiner to the choice of parameter values and the difficulty of selecting optimal values.

For illustration, we modified the relevant parameter values by multiplying each of the original values of ε and MinPts by 3 and showed the results in the respective tables.

Table 1. Experiment on HapMiner varying ε and MinPts: The average SSEs were obtained from running the first five datasets in Section 3.1.3.

	Average SSE
ε and MinPts * 0.1	0.000408
ε and MinPts * 0.5	0.000408
ε and MinPts * 2	0.000283
ε and MinPts * 3	0.04376
ε and MinPts * 4	0.5622
ε and MinPts * 5	0.5622

Table 2. Predictive accuracy based on experimental setting 1.

Average SSE	10%	20%	30%	40%	50%	Standard deviation of SSE over five different %	Average SSE over five different %
Blade	0.2808	0.3395	0.1850	0.0678	0.0455	0.1287	0.1837
HapMiner	0.1391	0.0760	0.0437	**0.0127**	**0.0058**	0.0544	0.0555
HapMiner (*modified*)	0.2170	0.2013	0.2081	0.1770	0.1393	0.0313	0.1886
LinkageTracker	0.1056	0.0447	0.0184	0.0177	0.0130	0.0388	0.0399
GeneRecon (assessed on five datasets)	**0.0339**	**0.0170**	**0.0181**	0.0225	0.0125	**0.0081**	**0.0208**

Table 2 shows the average sum-squared error (SSE) of each algorithm at various percentages of disease haplotypes carrying the founder mutation. The SSE is the sum of squared differences between the predicted marker location and the actual disease gene location across the simulations.

Generally, the predictive accuracies of the algorithms improve as the percentage of disease haplotypes carrying the founder mutation increases. However, LinkageTracker and GeneRecon showed consistent predictive accuracies at different percentage values.

At 10%, 20% and 30% of disease haplotypes carrying the founder mutation, GeneRecon had the lowest SSE followed by LinkageTracker. At 40%, LinkageTracker came in second after HapMiner, and was in third placing at 50%.

Next, we looked at the average execution time of the algorithms (refer to Table 3). HapMiner was the fastest algorithm, given the original parameter list

Table 3. Run time for experimental setting 1.

	Average time over five different %	Average time with LinkageTracker as base unit
Blade	1 m 1.06 s	6.76
HapMiner	**2.01 s**	**0.22**
LinkageTracker	9.03 s	1
GeneRecon	102 m 24.16 s	680.35

HapMiner took about two seconds to execute. The execution time for HapMiner with modified parameter values is not shown as the execution time is similar to the one with original parameter list. Blade took over a minute to execute on the average, GeneRecon took over one hour and LinkageTracker took about nine seconds on average to execute.

In terms of predictive accuracy, GeneRecon is comparable with LinkageTracker. However, the execution time of GeneRecon is much longer than LinkageTracker. If we consider only algorithms that take less than five minutes to execute, LinkageTracker had the lowest SSE at 10%, 20% and 30%. Furthermore, LinkageTracker also had the lowest average SSE over the five different percentage values. A major design objective of LinkageTracker is to have an algorithm with good consistency in finding disease gene location even when the occurrence of disease haplotypes carrying the founder mutation is very small. The experimental results in Table 2 show that our objective for LinkageTracker is met. The results indicated low standard deviation and good predictive accuracy for LinkageTracker, with SSE below 0.05 for four out of five percentages of disease haplotypes carrying the founder mutation.

3.1.2. *Experimental setting 2*: *Noisy data*

Next, we assessed the algorithms' performance with noises in the data. We assessed the algorithms' capability in detecting the disease gene location when only a small portion of the disease haplotypes actually carry the founder mutation of interest, while others are disease haplotypes without the founder mutation of interest. The disease haplotypes without the founder mutation are confounding factors that could influence the predictive accuracy of an algorithm.

Similar to experimental setting 1, datasets with different percentages of founder mutation carrying disease haplotypes were generated (at 10%, 20%, 30%, 40% and 50%). However, the data generation procedure is more elaborate. A dataset is generated as given in Table 4. For example, there are two main steps for generating datasets for the 10% mutation test. First we generated the disease set by

Table 4. Data generation for experimental setting 2.

Mutation level	Data type	Set-A	Set-B	Set-C	Total
10%	Disease set	5/63	All 31	14/92	50
	Control set	—	—	50/(92–14)	50
20%	Disease set	10/63	All 31	9/92	50
	Control set	—	—	50/(92–9)	50
30%	Disease set	15/63	All 31	4/92	50
	Control set	—	—	50/(92–4)	50
40%	Disease set	20/63	30/31	—	50
	Control set	—	—	50/92	50
50%	Disease set	25/63	25/31	—	50
	Control set	—	—	50/92	50

Table 5. Predictive accuracy based on experimental setting 2.

Average SSE	10%	20%	30%	40%	50%	Standard deviation of SSE over five different %	Average SSE over five different %
Blade	0.2810	0.2528	0.1332	0.0412	0.0503	0.1115	0.1517
HapMiner	0.0467	0.0038	**0.0024**	**0.0012**	**0.0015**	0.0199	0.0111
HapMiner (*modified*)	0.2864	0.3012	0.1299	0.1081	0.0562	0.1106	0.1764
LinkageTracker	**0.0047**	**0.0034**	0.0052	0.0056	0.0037	**0.0009**	**0.0045**
GeneRecon (assessed on five datasets)	0.0249	0.0132	0.0171	0.0231	0.0252	0.0053	0.0207

randomly selecting 5 out of 63 samples from Set-A, all 31 samples from Set-B, and randomly selecting 14 out of 92 samples from Set-C. Next we generated the control set, by randomly selecting 50 samples out of the remaining 78 samples from Set-C (as 14 samples have already been taken out for the disease set). There are 50 samples in both the disease and control sets. The data generation was repeatedly performed for 50 test datasets at the same mutation level.

Table 5 shows the average sum-squared error in the predictions of each algorithm at various percentages of disease haplotypes carrying the founder mutation. LinkageTracker had the lowest SSE at 10% and 20% whereas HapMiner had the lowest SSE at 30%, 40% and 50%. At 20% to 50%, the predictive accuracies of LinkageTracker and HapMiner were comparable since the differences in SSE were less than 0.005. LinkageTracker showed good consistency in its prediction with the lowest average SSE and standard deviation over the five different percentages of the founder mutation.

Next we examined the average execution time of the algorithms (refer to Table 6). HapMiner is the fastest algorithm, followed by LinkageTracker. HapMiner took about 2.3 seconds to execute, and LinkageTracker took about 15 seconds to execute. GeneRecon is the slowest compared to all the algorithms.

3.1.3. *Experimental setting 3*

In this experiment, we assessed the algorithms' performance when applied to the cystic fibrosis dataset without any modification to the ratios of the original disease

Table 6. Run time for experimental setting 2.

	Average time over five different %	Average time with LinkageTracker as base unit
Blade	1 m 4.02 s	4.1652
HapMiner	2.33 s	0.1516
LinkageTracker	15.37 s	1
GeneRecon	103 m 47.10 s	405.1461

haplotypes. Fifty datasets were generated for this experimental setting. The steps for generating the 50 datasets are: First, samples from Set-A and Set-B were combined to form a new set, Set-X, which consists of 94 disease samples. Next, 50 samples were randomly picked from Set-X to form the disease set. Lastly, 50 samples were randomly picked from the 92 control samples (i.e. Set-C) to form the control set. The last two steps were repeated 50 times to form 50 datasets.

Table 7 shows the average sum-squared error of each of the algorithm for the 50 datasets. LinkageTracker came in second, behind HapMiner which had the lowest average SSE for experimental setting 3. The standard deviation of HapMiner is also the lowest followed by LinkageTracker. The average execution time of HapMiner is the shortest followed by LinkageTracker. GeneRecon is the slowest algorithm, requiring more than one hour to execute one dataset on average.

3.2. *Friedreich ataxia*

Friedreich ataxia is an autosomal recessive degenerative disease that involves the central and peripheral nervous system and the heart. The friedreich ataxia dataset was first reported by Liu *et al.* for linkage disequilibrium mapping. This dataset contains 58 disease haplotypes and 69 control haplotypes with 12 microsatellite markers. The gene is located between the fifth and sixth markers, approximately 9.8125 cM away from the leftmost marker. The total region size in this study is 15 cM.

The experiments performed using the friedreich ataxia dataset is similar to the experimental setting 3 in the previous section. The procedure of the data generation is as follows: First, pick 50 samples randomly from the 58 disease samples. Next, pick 50 samples randomly from the 69 control samples. The procedure was performed 50 times to form 50 datasets.

Table 8 shows the average sum-squared error of each of the algorithm for the 50 friedreich ataxia datasets. HapMiner had the lowest SSE, Blade was second and LinkageTracker was third in predictive accuracy. However, the predictive accuracies of the three algorithms are comparable since the differences in SSEs were at most 0.05. No results were produced by GeneRecon for the friedreich ataxia dataset because GeneRecon accepts only binary valued attributes, whereas markers in the friedreich ataxia dataset are microsatellite markers each with more than 10 possible

Table 7. Predictive accuracy and run time for experimental setting 3.

	Standard deviation of SSE	Average SSE	Average time (s)
Blade	0.2045	0.0615	75.1905
HapMiner	**0.0041**	**0.0007**	**2.0336**
HapMiner (*modified*)	0.1715	0.0977	2.0336
LinkageTracker	0.0146	0.0085	9.7635
GeneRecon (assessed on five datasets)	0.025	0.0266	6046.967

Table 8. Predictive accuracy and run time for friedreich ataxia dataset.

	Standard deviation	Average SSE	Average time (s)
Blade	0.0640	0.0572	258.4912
HapMiner	**0.0142**	**0.0264**	2.0138
HapMiner (*modified*)	42.4966	61.3714	**1.2095**
LinkageTracker	0.0326	0.0770	4.6310
GeneRecon	–	–	–

alleles. HapMiner had the shortest execution time for the friedreich ataxia dataset followed by LinkageTracker.

4. Results on Generated Datasets

We compared the performance of LinkageTracker with HapMiner (given the original parameter list) on 100 generated datasets. HapMiner with the original parameter list has shown to be efficient based on the results from real datasets in the previous section. Furthermore, HapMiner also made use of the same 100 generated datasets in the original paper by Li & Jiang. The datasets used in this experiment were generated by Toivonen *et al.* Unfortunately, the program HPM by Toivonen *et al.* is not available to us. Nevertheless, we report the results of HPM in their original paper[3] and compare the performance with LinkageTracker and HapMiner.

There are in total 100 datasets, each consisting of 400 biological sequences where 200 sequences are labeled "abnormal" and the rest of the 200 sequences are labeled "normal". Each biological sequence consists of 101 microsatellite markers, and the total region size is 101 cM. The datasets were generated such that each dataset has a different disease gene location. The main task is to predict the marker that is nearest to the disease gene for each dataset.

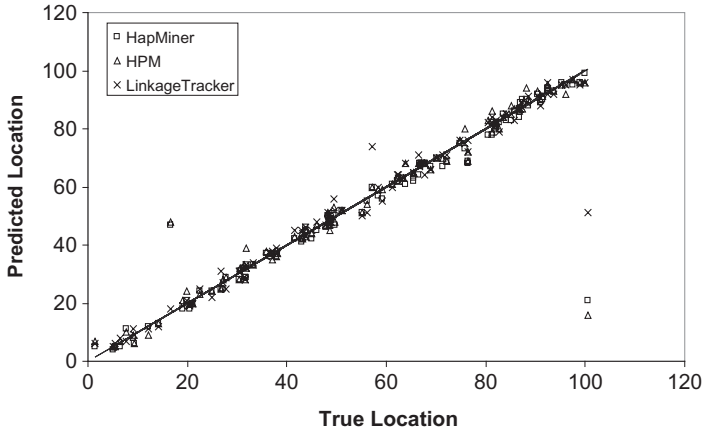


Fig. 4. Comparison of prediction accuracy among HapMiner, HPM and LinkageTracker.

Table 9. Predictive accuracies on generated datasets.

	Average SSE over 100 datasets	Average SSE over 99 datasets (after removing the dataset causing the outlier)
HPM	80.71	15.47
HapMiner	76.91	13.85
LinkageTracker	**34.30**	**9.90**

Figure 4 shows the performance of HapMiner, HPM and LinkageTracker when applied to the 100 generated datasets. The points on the graph depict the predicted disease gene location by each of the algorithms. The straight line depicts that the predicted location is the same as the actual location, therefore the closer the points to the straight line, the more accurate is the prediction. Among the three algorithms, LinkageTracker had the lowest average SSE for the 100 datasets (refer to Table 9). All the three algorithms did not perform well on one of the dataset (refer to the outliers below the straight line in Fig. 4); hence, we excluded that dataset in the performance assessment. LinkageTracker continued to be the algorithm with the lowest SSE, even after the exclusion of the outlier causing dataset from performance assessment as shown in Table 9.

5. Experiment with Large Datasets

HapMiner is generally the fastest algorithm; we compared the processing time of LinkageTracker with HapMiner in larger datasets with different percentages of disease haplotypes carrying the founder mutation of interest. The datasets were generated using the Cystic Fibrosis dataset. For each sample size of 200, 400, 600, 800, and 1000, 300 datasets were generated varying the percentage of disease haplotypes carrying the founder mutation of interest from 10% to 30%. Hence, for each sample size (i.e. 200, 400, 600, 800, 1000) we performed 300 simulations altogether — 100 simulations for each percentage. The processing speed and predictive accuracy of LinkageTracker and HapMiner are as shown in Table 10.

It could be observed that the average processing speed of LinkageTracker is faster than HapMiner in some of the cases. This is because the computational speed of LinkageTracker is highly dependent on the number of candidate patterns considered. When the percentage of haplotypes carrying the founder mutation is low, there is less number of candidate patterns to consider, and hence the processing speed of LinkageTracker is faster. It could also be observed that the predictive accuracy of LinkageTracker is better than HapMiner when the percentage of disease haplotypes carrying the founder mutation is low. Through this experiment, the strength and weakness of the two algorithms become apparent; LinkageTracker performs well in terms of speed and predictive accuracy when the percentage of haplotypes carrying

Table 10. Experiment with large datasets.

		Dataset size 200		Dataset size 400		Dataset size 600		Dataset size 800		Dataset size 1000	
		Linkage Tracker	Hap Miner	Linkage Tracker	Hap Miner	Linkage Tracker	Hap Miner	Linkage Tracker	Hap Miner	Linkage Tracker	Hap Miner
10%	Average SSE	0.1169	0.1436	0.0436	0.0660	0.0461	0.0692	0.0279	0.0227	0.0389	0.0444
	Average time (s)	721.06	6.2305	21.90	21.52	29.27	45.91	44.82	79.70	61.62	122.58
20%	Average SSE	0.0509	0.0525	0.0143	0.0194	0.0122	0.0092	0.0075	0.0001	0.0060	0.0003
	Average time (s)	17.41	12.40	29.15	21.36	38.72	45.68	47.96	79.12	55.20	121.53
30%	Average SSE	0.0145	0.0053	0.0082	0.0021	0.0049	0.0011	0.0032	0.0001	0.0035	0.0001
	Average time (s)	17.88	6.18	27.25	21.32	32.92	45.60	65.1852	79.02	653.75	121.09

the founder mutation is low, whereas HapMiner performs well when the percentage of founder mutation carrying disease haplotypes is high. However, such good performance from HapMiner can only be achieved with correct parameter settings.

6. Discussion

We have introduced a new method for linkage disequilibrium mapping known as LinkageTracker. We have compared LinkageTracker with some leading methods in linkage disequilibrium mapping. Extensive performance studies show that the predictive accuracy of LinkageTracker is consistently good under different conditions from the extremely difficult condition where the samples with the mutation of interest are as low as 10% and with high noise level, to the easier condition where the samples with the mutation of interest are as high as 50%. LinkageTracker and HapMiner have the best predictive accuracy in general. However, the predictive accuracy of HapMiner with the modified parameters was generally not as good when compared to all the other algorithms, which means that HapMiner's performance is very sensitive to its parameter setting. GeneRecon is comparable with LinkageTracker in terms of predictive accuracy; however, the execution time of GeneRecon is much longer than LinkageTracker. Furthermore, GeneRecon only works on bi-allelic markers.

The overall performance of LinkageTracker is promising as it provides consistently good predictive accuracy while taking reasonably short processing times, and it is also easy to use since it does not require the setting of complex parameters. The main weakness of LinkageTracker is that it is not able to make use of additional information such as genealogy of the haplotypes to improve performance, even when the additional information is available. Furthermore, due to the scoring method used by LinkageTracker to find significant patterns, LinkageTracker will not perform well in mixed populations where the haplotype frequency difference between disease and normal samples are due to sampling from different subpopulations. These problems will be addressed in our future work.

Acknowledgments

This research was supported in part by Singapore Agency for Science, Technology and Research grants SERC/072/101/0016 and BM/00/07, and an Academic Research Funding grant R-252-000-111-112/303 from the Ministry of Education in Singapore. We would like to thank the anonymous reviewers for their constructive and helpful suggestions.

References

1. Liu J, Sabatti C, Teng J, Keats B, Risch N, Bayesian analysis of haplotypes for linkage disequilibrium mapping, *Genome Res* **11**:1716–1724, 2001.
2. Mailund T, Schierup MH, Pedersen CNS, Madsen JN, Hein J, Schauser L, GeneRecon — A coalescent based tool for fine-scale association mapping, *Bioinformatics* **22**:2317–2318, 2006.

3. Toivonen H, Onkamo P, Vasko K, Ollikainen V, Sevon P, Mannila H, Herr M, Kere J, Data mining applied to linkage disequilibrium mapping, *Am J Hum Genet* **67**:133–145, 2000.
4. Toivonen H, Onkamo P, Vasko K, Ollikainen V, Sevon P, Mannila H, Kere J, Gene mapping by haplotype pattern mining, *Proc IEEE Int Symp Bioinformatics Bioeng* 99–108, 2001.
5. Li J, Jiang T, Haplotype-based linkage disequilibrium mapping via direct data mining, *Bioinformatics* **21**:4384–4393, 2005.
6. Morris AP, Whittaker JC, Balding DJ, Fine-scale mapping of disease loci via shattered coalescent modeling of genealogies, *Am J Hum Genet* **70**:686–707, 2002.
7. Agrawal R, Srikant R, Fast algorithm for mining association rules, *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, 1994.
8. Ester M, Kriegel HP, Sander H, Xu X, A density-based algorithm for discovering clusters in large spatial datasets with noise, *KDD* 226–231, 1996.
9. Lin L, Wong L, Leong TY, Lai PS, LinkageTracker: A discriminative pattern tracking approach to linkage disequilibrium mapping, *DASFAA*, Beijing, China, 30–42, 2005.
10. Long A, Langley C, The power of association studies to detect the contribution of candidate genetic loci to variation in complex traits, *Genome Res* **9**:720–731, 1999.
11. Weisstein, Eric W, "Bonferroni correction" from MathWorld.
12. Fisher RA, *Statistical Methods for Research Workers*, 14th ed. Hafner/MacMillan, New York, 1970.
13. Kerem BS, Rommens JM, Buchanan JA, Markiewicz D, Cox TK, Chakravarti A, Identification of the cystic fibrosis gene: Genetic analysis, *Science* **245**:1073–1080, 1989.

Li Lin received her Ph.D. degree in Computer Science from the National University of Singapore. She is a post-doctoral fellow at the Institute of High Performance Computing. Her research interests include the application of data mining, machine learning, and probabilistic reasoning methods in the biomedical domain. In the area of data mining and machine learning, she worked on discriminative analysis and pattern mining methods that could efficiently be applied to disease gene location inference and disease carrier detection. In the area of probabilistic reasoning, she focused on design and development of computer-based decision systems that combine physician's subjective judgment, information learned from statistical models and related literatures, to help physicians in disease diagnosis.

Limsoon Wong is a Professor of Computer Science and Professor of Pathology at the National University of Singapore. He is currently working mostly on knowledge discovery technologies and is especially interested in their application to biomedicine. Limsoon has written about 150 research papers, a few of which are among the best cited of their respective fields. He serves on the editorial boards of *Information Systems* (Elsevier), *Journal of Bioinformatics and Computational Biology* (ICP), *Bioinformatics* (OUP), and *Drug Discovery Today* (Elsevier). He received his B.Sc. (Eng) in 1988 from Imperial College London, UK, and his Ph.D. in 1994 from University of Pennsylvania, USA.

Tze-Yun Leong received her B.S., M.S., and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology. She is an Associate Professor of Computer Science at the National University of Singapore. Her research interests include decision-theoretic artificial intelligence, temporal and uncertainty reasoning, and biomedical informatics. Her research focuses on representation, analytic, and learning techniques that model problems from multiple perspectives, integrate information from heterogeneous sources, and adapt solutions to environmental changes. She also has consulting and business experiences in decision support and biomedical computing technologies. She is an associate editor of *Methods of Information in Medicine* and the *Artificial Intelligence in Medicine Journal*; she is also serving on the editorial board of the *International Journal of Biomedical Informatics*.

Poh San Lai is Associate Professor at Department of Pediatrics, National University of Singapore and heads the Human Molecular Genetics Lab. She has been involved in the development of genetic tests for molecular analysis, prenatal diagnosis and genetic counseling of pediatric genetic diseases in Singapore. She set up the laboratory for molecular studies in single gene disorders such as Duchenne muscular dystrophy, retinoblastoma, hemophilia, spinal muscular atrophy, etc. in 1990. Her other major research interest is in molecular therapeutics for gene repair. She is currently the President of the Biomedical Research and Experimental Therapeutics Society of Singapore, Board Member of the Asia-Pacific Society of Human Genetics and member of the Singapore National Medical Research Council Scientific Review Committee. She sits on the Singapore National Youth Award (SYA) Science & Technology Advisory Committee. She is also Reviews Editor of *Annals of*

Human Genetics, member of editorial board of *Human Genomics* and co-Chair, Policy Review Board of the Pan-Asian SNP Initiative studying genetic diversity among Asian populations and ethnicities. She serves as a member of the Institutional Biosafety Committee and Faculty Safety Committee at the National University of Singapore, and as Adjunct Senior Research Fellow with the Defence Medical Research Institute, Defence Science and Technology Agency of Singapore.