EFFICIENT MINING OF HAPLOTYPE PATTERNS FOR DISEASE PREDICTION

A THESIS SUBMITTED BY

LIN LI

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

(FIRST CLASS HONOURS)

UNIVERSITY OF LEICESTER, UK

1999

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2007

Contents

List of Figures

List of Tables

Acknowledgement

I would like to express my gratitude to my supervisor and thesis advisors, A/Prof. Leong Tze Yun, Prof. Wong Limsoon, and Dr. Lai Poh San for their guidance, support, and generosity in sharing their knowledge and wisdom with me. Without their generous help this thesis would not have been possible.

I would also like to thank my external project collaborators A/Prof. Lim Tow Keang and A/Prof. Poh Kim Leng for their generosity in sharing their knowledge and experience with me in medical decision modelling.

My heartfelt thanks to my husband Wong Swee Seong, who is always by my side sharing all my joy and sadness, and been through all the tough times with me. Most of all, thanks for his love and care, and patience with me during my difficult days.

Last but not least, I am eternally grateful to my parents for their love, support, and inspirations that motivate me to reach my goal in achieving academic excellence.

Summary

It was quoted by J. Han [1] that we are at the stage of being data rich but information poor; the profusion in data collection does not correspond with an exponential development in efficient techniques to extract valuable and useful knowledge from data. The filling of such knowledge gap is a challenge faced by all data miners.

This thesis focuses on knowledge extraction from domain specific data known as haplotypes.A major issue in pattern extraction from haplotypes is the ability to identify valuable and useful information for disease pattern prediction which can be applied in prognosis and carrier detection.

This thesis presents a new method known as LinkageTracker for disease gene location inference (or linkage disequilibrium mapping) from haplotypes. This method was compared with some leading methods in linkage disequilibrium mapping such as Haplotype Pattern Mining (HPM) [2, 3], HapMiner [4], Blade [5, 6], and GeneRecon [7]. LinkageTracker provides good predictive accuracies while taking up reasonably short processing times. Furthermore, LinkageTracker does not require any population ancestry information about the disease and the genealogy of the haplotypes. It is a useful tool for linkage disequilibrium mapping when the users do not have much information about their datasets. It represents a promising method for effective linkage disequilibrium mapping.

This thesis also introduces an novel algorithm called ECTracker for extracting useful haplotype patterns for genetic analysis and carrier detection. Experimental studies show that ECTracker is capable of deriving useful patterns when the dataset is very

small. In classification, ECTracker is capable of producing good predictive accuracies that are comparable to some leading machine learning methods. Using biological datasets from wet experiments, ECTracker could efficiently extract patterns that allow for predictive disease classification. Furthermore, it is able to classify samples as unknown if they are almost indistinguishable from the defined classes. This work, in most cases, outperforms the existing methods in classification accuracies for datasets like haplotype patterns for disease class prediction.

Chapter 1

General Introduction

1.1 Introduction

Making medical decisions such as diagnosing the diseases that cause a patient's illness is often a complex task. Much of these complexities arise from the inability to efficiently recognize reliable indicative (predictive) factors associated with the diseases. Fortunately, the profusion in data collection by hospitals and scientific laboratories in recent years has helped in the discovery of many disease associated factors. Embedded within the large collection of data is valuable information that suggests potential factors that are associated with the diseases. Data mining techniques are often used to extract the disease associated factors from the large datasets.

Data mining is the task of discovering previously unknown, valid patterns and relationships in large datasets. Generally, each data mining task differs in the kind of knowledge it extracts and the kind of data representation it uses to convey the discovered knowledge. In this thesis, we examine some of the existing knowledge extraction techniques when applied to haplotypes for disease gene location inference, genetic variations analysis and carrier detection. The main difficulties in pattern extraction for such cases include rarity in the sample haplotypes of interest and noise in the data collected. The main chapters will further elaborate on the addressed problems.

1.2 Motivation and Contribution

This thesis discusses the opportunities and mechanisms to leverage knowledge (or information) extraction performance from biomedical datasets for supporting medical decision making. The extraction of useful information from data, such as factors that promote or increase risk of a disease, helps in medical diagnosis, planning of patient management strategies, and counseling of patients and their family members.

We report the findings observed from literature surveys, propose some efficient algorithms and mechanisms to improve the performance of knowledge extraction, and present the results that we have achieved through experimental studies. Finally, we hope that this thesis will provide useful decision making techniques for the researchers and medical practitioners to improve patient care.

We highlight two main contributions presented in this thesis. First, our research proposal is realized in the domain of disease gene location finding (also known as linkage disequilibrium mapping), where we propose an efficient method for inferring disease gene locations. We compared our algorithm with some leading methods for linkage disequilibrium mapping. Detailed experimental studies and analysis show that our approach is efficient while maintaining good predictive accuracies.

Second, we extend our method to support descriptive analysis and classification of haplotype patterns. Widely used machine learning methods were evaluated with haplotype patterns extracted, for the purpose of both descriptive analysis and classification (or predictive analysis). Experimental studies and comparisons show that

our method is capable of extracting useful patterns to support genetic variation analysis and at the same time producing good predictive accuracies to facilitate carrier detection.

1.3 An Analogy

This section gives a simple analogy to our work before we present the details in the later chapters. The analogy paints a complete picture of the motivation behind the proposed algorithms and what we aim to achieve with them. We illustrate our designs by following a series of tasks performed by a jeweler who deals mainly with diamonds.

Mr. Smith works in Diamond Company based in London. Diamond Company specializes in sales and marketing of diamonds. Each day, diamonds from all over the world arrive at the company where they would sort, value, and sell the diamonds.

In the first example, let's assume a character Lisa who has a blue diamond that she adores very much. One day, Lisa wishes to buy a diamond that has the same characteristics as her favorite blue diamond for her mother as a birthday gift. Lisa approaches Mr. Smith for help. Like other minerals and rocks, diamond crystals contain within themselves a record of their geologic history in terms of their morphology, detailed chemical composition, and etching features. Therefore, diamonds from a particular geographic source will have their very own unique characteristics and with very similar chemical compositions. To help Lisa find another diamond that has the same characteristics as her blue diamond, Mr. Smith needs to first determine the geographic source where Lisa's blue diamond is extracted or mined. There are many diamond mines worldwide. To perform detailed chemical composition analysis on diamonds from all the different diamond mines in the world will take a very long time. Fortunately based on

Mr. Smith's years of working experience, he knows that blue diamonds are mainly found in South Africa mines. With this valuable knowledge, Mr. Smith only needs to analyze chemical compositions of diamonds from the few South Africa mines to quickly identify the geographic source where Lisa's blue diamond was extracted. In our work, we have designed an algorithm that makes use of expert knowledge to efficiently find the disease gene locations to solve the linkage disequilibrium problem. It is similar to what Mr. Smith did to quickly identify the geographic source of Lisa's blue diamond.

Next, a businessman George wants to sell some diamonds to Diamond Company. George presents the diamonds to Mr. Smith. Before buying the diamonds, Mr. Smith needs to ensure that the diamonds are natural diamonds. There are some features that distinguish natural diamonds from synthetic diamonds; these features were discovered by scientists after hundreds of experiments. Firstly, under very intense short-wave ultraviolet lamp, synthetic diamonds will glow very brightly whereas natural diamonds are almost inert under the ultraviolet light. Also, phosphorescence is observed on synthetic diamonds after the ultraviolet lamp is turned off, but not for natural diamonds. Secondly, under a hand lens or optical microscope, planar defects and large metallic inclusions are often found in synthetic diamonds, while natural diamonds have no such properties. Armed with the knowledge of the unique features of natural diamonds, Mr. Smith can easily determine whether the diamonds presented by George are natural or synthetic. In our work, we have designed an algorithm to discover the "unique features" or more specifically the genetic variations of patients affected by a bleeding disorder called hemophilia, and perform predictive inference using the "unique features" discovered. A

similar task to what Mr. Smith did in determining whether George's diamonds are natural based on the knowledge of the unique features of diamonds.

The next section will touch on a set of problems and issues in data mining when applied to biomedical domains. We outline our approaches in addressing these issues. This is followed by a description of biomedical knowledge extraction problems that can be addressed or alleviated using our proposed algorithms.

1.4 Research Problems and Proposed Approaches

We begin by exploring ideas in pattern extraction from biological datasets. Association rules have been studied extensively in the Knowledge Discovery in Databases (KDD) field for pattern extraction, and there exists many efficient algorithms to perform such task. The support and confidence thresholds are usually used to guide the search for interesting patterns. From our literature survey, we observed that most of the pattern mining methods are exhaustive; some practical difficulties arise when the number of items in each record is very large. We explored the use of domain specific expert knowledge to alleviate such technical difficulty (without compromising the quality of patterns mined) in the problem of finding disease gene locations. The process of inferring disease gene locations from observed associations of marker alleles in affected patients and normal controls is known as linkage disequilibrium mapping. The main idea of linkage disequilibrium mapping is to identify chromosomal regions with common molecular marker alleles at a frequency significantly greater than chance. It is based on the assumption that there exists a common founding ancestor carrying the disease alleles,

and is inherited by his descendents together with some other marker alleles that are very close to the disease alleles. The same set of marker alleles is detected many generations later in many unrelated individuals who are clinically affected by the same disease.

Our approach utilizes expert knowledge in genetics to reduce the search space and at the same time maintaining good predictive accuracies. The proposed method mainly focuses on the difficult problems where the occurrence of useful patterns (or pattern of interest) is very low, and consists of errors or noise. We conducted extensive performance studies to evaluate the efficiency of LinkageTracker when compared to some leading methods in linkage disequilibrium mapping including HPM [2, 3], HapMiner [4], Blade [5, 6], and GeneRecon [7].

Next, we explore data mining methods that are capable of performing genetic analysis and carrier detection. Intuitively expressive patterns (or genetic variations) are extracted to provide medical practitioners with insights about the genetic manifestations of patients affected by a disease. The extracted patterns are subsequently used for predictive inference (or classification) to help in carrier detection, which is useful for medical prognosis and decision making processes. We propose ECTracker for performing both pattern extraction and classification, and compare the expressiveness and predictive accuracy of our method with some leading methods in machine learning. The ECTracker algorithm consists of 2 steps: First, it generates combination of haplotype patterns to facilitate the analysis of genetic variations of diseased patients, and second, it performs classification using the haplotype patterns generated in the first step for carrier detection. We compared the performance of ECTracker with some leading machine learning methods including C4.5 [8], Naïve Bayesian Method [9], Artificial Neural

Network [10], Support Vector Machine [11], K-Nearest Neighbor [12], and Bagging [13] (with Naïve Bayesian as base).

1.5 Organization of Thesis

The rest of the thesis is organized as follows. In Chapter 2, we review some of the related work in the literature and also draws out the background knowledge necessary for building the proposed methods. In Chapter 3, we discuss the issues in the domain of disease gene location inference and propose a novel algorithm known as LinkageTracker to efficiently address the issues. In Chapter 4, we present ECTracker for the extraction of genetic variations in patients affected by hemophilia A. The extracted patterns are also used for predictive inference. The efficiency of ECTracker is also assessed using two well studied real datasets namely Cystic Fibrosis [5] and Friedrich Ataxia [14]. Finally, we conclude in Chapter 5 with directions for future research. Some of the proposed designs were published in [15-18].

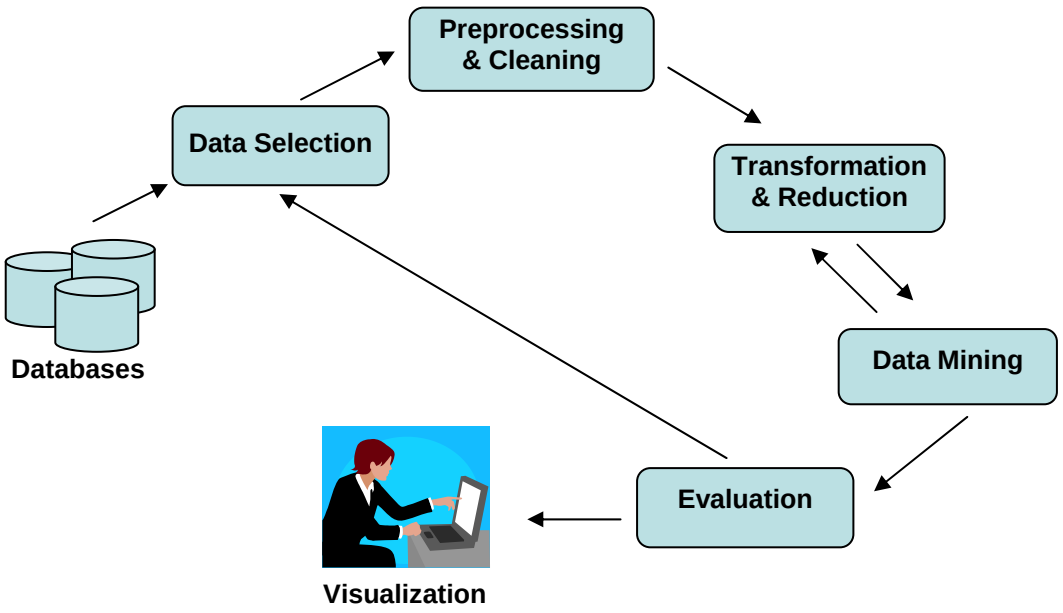
Chapter 2

Related Work

2.1 Background

Data mining has been defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [19] and "the science of extracting useful information from large data sets or databases" [20]. It is the core principle of the knowledge discovery process, which also includes data selection, preprocessing and cleaning, transformation and reduction, evaluation, and visualization. The knowledge discovery process is illustrated in Figure 2.1.

Data mining is not a single technique; it includes any techniques that help in extracting useful information out of data for pattern analysis and prediction of future trends and behaviors, allowing users to make proactive, knowledge-driven decisions. In the context of healthcare and biomedicine, data mining is often viewed as a potential mean to identify various biological, drug discoveries, and patient care knowledge embedded in the extensive data collected. Furthermore, data mining provides results that possibly highlight vaguely understood doctrine and provide useful insights to help in decision making processes. In general, data mining tasks is classified into two broad categories: *descriptive mining* and *predictive mining*. The rest of this chapter covers in greater details the two form of data mining tasks, and presents several leading techniques which are relevant to our work.



1. **Data selection**: Retrieval of relevant data from databases.

2. **Preprocessing & cleaning**: Removal of noise and inconsistent data, detecting and dealing with missing values.

3. **Transformation & reduction**: data sets are reduced to the minimum size possible through sampling or summary statistics. For example, tables of data may be replaced by descriptive statistics such as mean and standard deviation.

4. **Data mining**: Intelligent methods are selected for pattern extraction

5. **Evaluation**: the patterns identified by the data mining methods are interpreted, for instance, determining the clinical relevance of the findings.

6. **Visualization**: knowledge representation techniques such as pie charts and graphs are used to present the mined knowledge to the user

Figure 2.1: Knowledge discovery process

2.2 Descriptive Mining

Descriptive mining automatically extracts new or useful information from large databases and presents the discovered information in intuitively understandable terms for human analysis. Association rule mining is the most well-studied *descriptive mining* method in the Knowledge Discovery in Databases (KDD) field [21-24]. Their primary strength lies in their significant expressive power and their being relatively simple to comprehend, thus making them suitable for incorporation into decision-making processes.

2.2.1 Association Rule Mining

The task of association rule mining was first introduced in 1993 by Agrawal et. al [25]. The idea of association rule mining originates from the analysis of market data whereby the main task is to determine patterns that characterize the shopping behavior of customers from a large database of previous customer transaction records. An association rule has the following format: *X => Y (support, confidence)* to mean item *Y* exists if item *X* is found in the same record. *Support* is the percentage of the database with itemset, *XY*, appearing in the same record and *confidence* is the ratio of item *Y* appearing in records containing item *X*. Frequent itemsets are sets of items with *support* greater than a minimum user-defined *support*. Before association rules can be constructed, the frequencies of the underlying frequent itemsets have to be generated.

Association rule is formally defined as follows. Let $I = \{i_1, i_2, i_3, \ldots, i_m\}$ be a set of attributes called items. Let D be a set of transaction records. Each transaction record t in D consists of a set of items such that $t \subseteq I$. A transaction record t is said to contain an itemset X if and only if all items within X are also contained in t. Each record also contains a unique identifier called *TID*. *Support* of an itemset is the normalized number of occurrences of the itemset within the dataset. An itemset is considered as frequent or large, if the itemset has a *support* that is greater or equal to the user specified *minimum support*. The most common form of association rules is implication rule which is in the form of $X => Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \varnothing$. The support of the rule $X => Y$ is equal to the percentage of transactions in D containing $X \cup Y$. The *confidence* of the rule $X => Y$ equals to the percentage of transactions in D containing X also containing Y, i.e. $|X \cup Y| / |X|$. Depending on the application, the definition of *confidence* can be changed to suit a particular need [26-35]. For example, instead of using *confidence* as the measure of interestingness, *chi-squared* measure, X^2, is also commonly used to measure the correlation in the frequent itemsets. Details of these methods are described in Section 2.2.2.

Once the required minimum *support* and *confidence* are specified, association rule mining task becomes the finding of all association rules that satisfy the minimum requirements. The problem can be further broken down into 2 steps: mining of frequent itemsets and generating association rules [21, 36]. The number of possible combinations of itemsets increases exponentially with $|I|$ and the average transaction record length.

The very first published and efficient frequent itemset mining algorithm is *Apriori* [36]. *Apriori* uses breadth first search (BFS) as the search strategy. At each level, *Apriori*

reduces the search space by using downward closure property of itemset that if an itemset of length *k* is not frequent, none of its superset patterns can be frequent. Candidate frequent itemsets, i.e. itemsets that have the potential to be frequent, C_k where *k* is the length of the itemset, are generated before each data scan. The *supports* of candidate frequent itemsets are counted to verify whether they are frequent or not. Candidate *k* itemsets, C_k, are generated with frequent *k - 1* itemsets. *Apriori* achieves good performance by iterative reduction of candidate itemsets. However, *Apriori* requires *k* data scans to find all frequent *k*-itemsets. In large databases, it is very expensive to scan the data multiple times for very large *k*. Therefore a method that could restrict *k* to a reasonably small value yet without compromising the quality of interesting patterns mined would be very desirable. This motivates our approach to leverage on domain specific expert knowledge to restrict *k* to a small value without compromising the quality of interesting patterns mined. The quality of a pattern is good if the pattern mined could ultimately contribute in the prediction of disease gene location accurately.

Other efforts devoted to improving the efficiency of association rule mining include the mining of frequent closed patterns [37-43], maximal frequent patterns [44-47], and generators[48]. These methods are firstly exhaustive in nature, and secondly, they used *support* and *confidence* to determine the interestingness of a pattern. In the later chapter we will illustrate how LinkageTracker could achieve good predictive accuracies based on expert knowledge without the need for exhaustive search. Also the search for interesting patterns based on support and confidence are not suited to the problem of disease gene location inference. This is because *support* and *confidence* are not able to determine the magnitude of association between a pattern antecedents and consequent.

2.2.2 Mining of association rules based on different scoring methods

Besides finding efficient methods for mining association rules, much effort have also devoted to the finding of interesting rules or patterns. Depending on the application of the patterns mined, the definition of *confidence* can be changed to suit a particular need. Interestingness of a pattern can be measured in terms of underlying structure of the pattern and the data used in the discovery process.

Brin et al. [26, 27] proposed measuring significance of associations via the chi-square test for correlation from classical statistics. This approach requires the consideration of both presence and absence of items as a basis for generating rules. Brin et al. [26, 27] claims that the chi-squared measure is upward closed, i.e. the mining problem is reduced to the search for border correlated and uncorrelated itemsets in the lattice. An itemset is significant if it is supported and minimally correlated, which means that an itemset at level i+1 can be significant only if all its subsets at level i have *support* and none of its subsets at level i are correlated. The finding of correlated rules is equivalent to finding a border in the itemset lattice. In the worst case, when the border is in the middle of the lattice, it is exponential in number of items. In the best case the border is at least quadratic. However, it was later found that chi-squared measure does not posses the upward closure property for exploiting efficient mining of significant rules by DuMouchel et. al [49]. In the later chapter, we will introduce an algorithm known as Haplotype Pattern Mining (HPM) by Toivonen et al. [1, 2], which uses the chi-squared measure to determine interesting patterns for the problem of disease gene location

finding. Detailed comparisons will be made between *HPM* and *LinkageTracker* in that later chapter.

Li et al. [32, 33, 35] proposed the mining of association rules solely based on *confidence* without the *support* threshold. As discuss previously the *confidence* measure is neither downward nor upward closed. The authors overcome this problem by dividing the dataset into two subsets and discover patterns from the two relevant sub-datasets such that the pattern occurs with 100% confidence in one sub-dataset but 0% confidence in the other sub-dataset (known as jumping EPs). From the jumping EPs discovered, they construct association rules. However, this algorithm is very restrictive as it is not able to find patterns that occur with say 85% *confidence* in one sub-dataset and 10% *confidence* in another sub-dataset (as such pattern may be significant when scored with some other statistical method, say Pearson's correlation coefficient). Furthermore, Brin et al. [26] has shown that *confidence* measure may produce counter-intuitive results especially when strong negative correlations are present. For example, the *support* and *confidence* threshold are set to 5% and 50% respectively for a retail transaction dataset, and the association rule margarine $\rightarrow$ butter with *support* 20% and *confidence* 67% will pass the threshold conditions. However, the prior probability of customers purchasing butter is 80%, once customer purchases margarine, the conditional probability of that customer will buy butter reduces by 16.25% (i.e. (0.8-0.67)/0.8 * 100)). Hence the high *confidence* rule margarine $\rightarrow$ butter is misleading.

Tan and Kumar [29] proposed a metric known as *IS* to finding interesting association rules. This work assumes that only positively correlated patterns are of

interest to the data analyst. The interestingness measure of *IS* can be computed as follows:

$$IS = \sqrt{Conf(A \rightarrow B) \times Conf(B \rightarrow A)}$$

The *IS* measure is equivalent to the geometric mean of the confidence rule, however, since the measure of association between rule antecedents and consequence using confidence measure can be misleading as described earlier, therefore this method is not suited to the problem of disease gene finding.

Xiong et al. [30, 31] identified an upper bound for Pearson's correlation coefficient for binary variables and proposed an efficient algorithm known as TAPER to find all item pairs with correlations above the user specified minimum correlation threshold. The Pearson's correlation coefficient ϕ is expressed as shown in the equation below:

$$\phi = \frac{\sup(A,B) - \sup(A)\sup(B)}{\sqrt{\sup(A)\sup(B)(1-\sup(A))(1-\sup(B))}}$$

There are two steps in the TAPER algorithm; the first step is the filtering step where most of the false positive item pairs are pruned off to reduce further processing cost. The second step is the refinement step where the exact correlation is being computed for each surviving items pair from the filtering step. Item pair with correlation higher than the user specified threshold will be returned as output for the user. Although the TAPER algorithm provides good contribution in identifying the upper bound of Pearson correlation coefficient, it only scores the correlation between item pairs rather than itemsets. In the mining of association patterns, most users are interested in the

correlation between sets of items, hence more work need to be done in extending the TAPER algorithm to score the correlation between itemsets.

In a recent work by Li et al. [34], statistical relative risk and odds ratio were proposed to find interesting patterns. The search space was stratified into plateaus of subspaces based on *support* levels of the patterns, such that the space of odds ratio and relative risk can become convex for efficient mining of significant patterns. They proposed two methods for the mining of significant patterns. The first method uses FPclose [50] to find all the closed patterns, and then uses an algorithm known as Gr-growth that they developed to find all the generators [48]. The second method mine closed patterns and generators at the same time using an algorithm known as GC-growth that they proposed. Both algorithms that they proposed used the set-enumeration tree [51, 52] to organize the pattern space. Since the search space needs to be stratified based on *support* levels, the search space will become extremely large when the support threshold is set to a very small value. Furthermore, the finding of all interesting patterns is not essential in the problem of disease gene location finding as expert knowledge can be used to restrict the search space. Also the finding of all interesting patterns exhaustively will introduce noise that will affect the predictive accuracies (refer to chapter 3 for detailed explanation).

Prior to the work by Li et al. [43], we have independently proposed the use of odds ratio in the finding of interesting patterns in [15-17]. The statistical odds ratio has been widely used in the biomedical arena for discriminative studies. We find that odds ratio is very suited to the discovery of patterns with strong magnitude of association to the class labels even when the occurrences of the strongly associated patterns are rare.

Therefore the statistical odds ratio was used in our proposed algorithms to guide the discovery of interesting patterns.

2.3 Prediction Mining

The main *objective* of prediction mining is to assign new data items into one of the few predefined categorical classes [53]. As classification is the most studied data mining and knowledge discovery task [54], there are many classification algorithms. In this section we discuss some of the leading classification algorithms, namely *Artificial* Neural Network (ANN), Support Vector Machine (SVM), Decision Tree (C4.5), and Naïve Bayesian Classifier. These classification algorithms are applied to our haplotype dataset in Chapter 4 and comparisons of predictive accuracies will be performed.

2.3.1 Artificial neural network (ANN)

The main elements of an Artificial Neural Network (ANN) are processing elements or neurons, and weighted interconnections among the neurons. Each neuron performs a very simple computation, such as calculating a weighted sum of its input connections, and computes an output signal that is sent to other neurons. The training (mining) phase of an ANN consists of adjusting the weights of the interconnections, in order to produce the desired output [10, 55]. The adjustment of interconnection weights is usually performed by using some variant of the Hebbian learning rule. The basic idea of

this mechanism is that if two neurons are active simultaneously the weight of their interconnection must be increased.

The basic structure of an ANN is shown in Figure 2.2. In this figure there are layers of nodes, and each node of a given layer is connected to all the nodes of the next layer. This full-connectivity topology is not necessarily the best one, and the definition of the topology of a ANN – number of layers, number of nodes in each layer, connectivity among nodes in different layers, etc – is a difficult task, and it is a major part of the process of using ANN to solve the target problem. Often several different ANN topologies are tried, in order to empirically determine the best topology for the target problem. Each node interconnection is normally assigned a real valued interconnection weight.

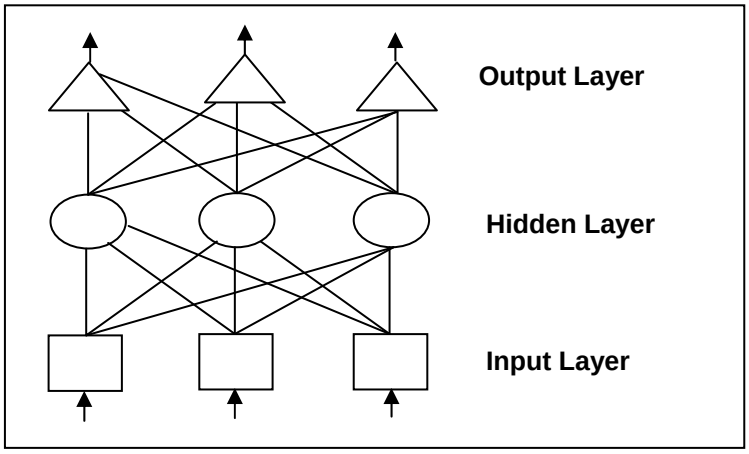


Figure 2.2: Artificial Neural Network

The nodes in the input layer correspond to values of attributes in the database. To classify a new tuple(or input) the values of the tuple's predicting attributes are given to the input layer. Then the network uses these values and the interconnection weights learned during the training phase to compute the activation value of the node(s) in the

output layer. In the case of a two-class problem, the output layer usually has a single node. If the activation value of that node is smaller than a given threshold then the network predicts the first class, otherwise the other class is predicted by the network. In the case of multiple-class problems there can be several nodes in the output layer, one node for each class, so that the node in the output layer with largest activation value represents the class predicted by the network.

2.3.2 Support vector machine (SVM)

Support vector machines are based on the structural risk minimization principle [11, 56] from computational learning theory. The idea of structural risk minimization is to find a hypothesis *h* for which we can guarantee the lowest true error. The true error of *h* is the probability that *h* will make an error on an unseen and randomly selected test example.

SVMs operate by finding a hyper-surface in the space of possible inputs. This hyper-surface will attempt to split the positive examples from the negative examples. The split will be chosen to have the largest distance from the hyper-surface to the nearest of the positive and negative examples [57]. Intuitively, this makes the classification correct for testing data that is near, but not identical to the training data.

SVMs are very universal learners. In their basic form, SVMs learn linear threshold function. Nevertheless, by adding in an appropriate kernel function [58], they can be used to learn polynomial classifiers, radial basic function (RBF) networks, and three-layer sigmoid neural nets.

2.3.3 Decision Tree

A decision tree is a tree-like knowledge-representation structure where every internal (non-leaf) node is labeled with the name of one of the predicting attributes, the branches coming out from an internal node are labeled with values of the attribute in that node, and every leaf node are labeled with a class (i.e. a value of the goal attribute) [59, 60]. A learned tree can also be re-represented as a set of if-then rules to improve human readability. A decision tree classifies a new, unknown-class tuple in a top-down manner. Initially the new tuple is passed to the root node of the tree, which tests which value the tuple has on the attribute labeling that node. Then the tuple is pushed down the tree, following the branch corresponding to the tuple's value for the tested attribute. This process is recursively repeated, until the tuple reaches a leaf node. At this moment the tuple is assigned the class labeling that leaf.

A decision tree is usually built by a top-down, "divide-and-conquer" method. Initially all the tuples being mined are assigned to the root node of the tree. Then the algorithm selects a partitioning attribute and partitions the set of tuples in the root node according to the values of the selected attribute. The goal of this process is to separate the classes, so that tuples of distinct classes tend to be assigned to different partitions. This process is recursively applied to the tuple subsets created by the partitions, producing smaller and smaller data subsets, until a stopping criterion (e.g. a given degree of class separation) is satisfied. The most common decision tree learning algorithms include the ID3 [61, 62] and its successor C4.5 [8]. Decision tree can also be used for descriptive mining as it is very easy to generate a set of rules from a decision tree.

2.3.4 Naïve Bayesian Classifier

A naïve Bayesian classifier [9, 63] is a statistical classifier which computes the probability of a sample belonging to a particular class based on Bayes theorem. Bayes theorem is a mathematical formula used to calculate conditional probabilities – the probability that a hypothesis *H* holds given the observed sample data *D* or posterior probability *P(H|D)*. The posterior probability can be computed from the prior probability *P(H)* together with *P(D)* and *P(D|H)* as follows:

$$P(H|D) = \frac{P(D \mid H)P(H)}{P(D)}$$

A naive Bayes assumes conditional independence among all attributes $A_1,A_2,\ldots,A_n$ given the class variable *C* . It learns from training data the conditional probability $P(A_i|C)$ of each attribute given its class label. Domingos gives a good explanation in [64] why a naive Bayes works surprisingly well despite its strong independence assumption.

2.3.5 Bayesian Belief Network

A Bayesian network (or a belief network) is a probabilistic graphical model that represents a set of variables and their probabilistic dependencies. The term "Bayesian networks" was coined by Pearl in 1985 [65] to emphasize three aspects: (1) the often subjective nature of the input information; (2) the reliance on Bayes's conditioning as the basis for updating information; and (3) the distinction between causal and evidential modes of reasoning, which underscores Thomas Bayes's paper of 1763. A Bayesian

belief network (BBN) is a directed graph, together with an associated set of probability tables. The graph consists of nodes and arcs, the nodes represent variables, which can be discrete or continuous, and the arcs represent causal/influential relationships between variables. If there is an arc from node *A* to another node *B*, *A* is called a *parent* of *B*, and *B* is a *child* of *A*. The set of parent nodes of a node X_i is denoted by parents(X_i). A directed acyclic graph is a Bayesian Belief Network relative to a set of variables if the joint distribution of the node values can be written as the product of the local distributions of each node and its parents:

$$P(X_1,...,X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i))$$

In the simplest case, a Bayesian Belief Network is specified by an expert and is then used to perform inference. In other applications the task of defining the network is too complex for humans. In this case the network structure and the parameters of the local distributions must be learned from data. Learning the structure of a Bayesian network requires a scoring function and a search strategy. A common scoring function is posterior probability of the structure given the training data. The time requirement of an exhaustive search returning back a structure that maximizes the score is super exponential in the number of variables. A local search strategy makes incremental changes aimed at improving the score of the structure. A global search algorithm like Markov chain Monte Carlo can avoid getting trapped in local minima.

Markov chain Monte Carlo (MCMC) method is an algorithm for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its stationary distribution. The state of the chain after a large number of

steps is then used as a sample from the desired distribution. The quality of the sample improves as a function of the number of steps. Many Markov chain Monte Carlo methods move around the equilibrium distribution in relatively small steps, with no tendency for the steps to proceed in the same direction. These methods are easy to implement and analyze, but unfortunately it can take a long time for the walker to explore all of the space. The walker will often double back and cover ground already covered. One of the most commonly used random walk MCMC methods is known as the Metropolis-Hastings algorithm. Metropolis-Hastings method generates a random walk using a proposal density and a method for rejecting proposed moves.

Chapter 3

LinkageTracker – Finding Disease Gene Locations

3.1 Introduction

Medical practitioners are interested in finding the disease linked or exact disease causing gene locations in the human genetic codes, in order to perform genomic analysis select and targeted treatment strategy on their patients. Genomic analysis helps in estimating the probability of occurrence of a particular disease outcome or manifestation in a patient and the extent to which the individual risk can be modified using preemptive strategies. For instance, an individual who was found to have inherited a particular gene mutation from her parents may make her susceptible to a disease like cancer. This individual requires intensive monitoring through regular health screening and skillful counseling on dietary and lifestyle changes to prevent the disease from progressing into the malignant phase. Therefore genomic analysis helps the medical practitioners in the decision making process for managing such patients and their family members, which ultimately increase survival rates and improve the overall quality of health care.

However, before such genomic analysis can be carried out, there is an important task of identifying the exact disease gene locations from the vast amount of genomic data collected. The process of inferring disease gene locations from observed associations of marker alleles in affected patients and normal controls is known as linkage disequilibrium mapping [66-68]. Linkage disequilibrium mapping has been used in the finding of

disease gene locations in many studies [69, 70]. The main idea of linkage disequilibrium mapping is to identify chromosomal regions with common molecular marker alleles[1] at a frequency significantly greater than chance. It is based on the assumption that there exists a common founding ancestor carrying the disease alleles, and is inherited by his descendents together with some other marker alleles that are very close to the disease alleles. The same set of marker alleles is detected many generations later in many unrelated individuals who are clinically affected by the same disease.

3.1.1 Challenges

Finding the exact disease gene location is a non-trivial task. This is because, in reality, the occurrence of such allele patterns is usually very low, and most often consists of errors or noise. Let us illustrate the difficulties in the problem of gene location finding with an example. In the counseling and health management of relatives of breast cancer patients, one of the strategies is to analyze their chromosomes 17 or 13 to determine whether they have inherited the mutated BRCA-1 or BRCA-2 gene. An individual who has inherited the BRCA-1 or BRCA-2 gene will have higher disposition to breast cancer as compared to another individual who has not inherited BRCA-1 or BRCA-2 gene, and therefore requires more regular health screening as compared to normal individuals. However, this genomic analysis is only possible if the exact locations of BRCA-1 and BRCA-2 genes were known.

[1] A molecular marker is an identifiable physical location on the genomic region that either tags a gene or tags a piece of DNA closely associated with the gene. 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.

Now let us assume that we are at the early stage of research for BRCA-1 and BRCA-2 genes, and no one knows the exact locations of the two genes although researchers know that BRCA-1 resides in chromosome 17 and BRCA-2 resides in chromosome 13. To find the exact locations of the two genes, it is required to perform analyses on gene sequences of chromosome 13 and 17 collected from patients affected by breast cancer. However, the hereditary mutations of BRCA-1 and BRCA-2 genes only account for about five to ten percent of all breast cancer patients [71]. This means that, given a set of chromosome 17 or 13 gene sequences collected from breast cancer patients, only at most ten percent of the gene sequences contain the BRCA-1 or BRCA-2 gene mutations. This means that the patterns or gene expressions that we are interested in are very rare within the set of collected data. To further complicate the task of finding disease gene locations, the gene sequences collected also consist of errors or noise due to sample mishandling and contamination.

Due to the complexities in the problem of disease gene location finding, existing data mining methods cannot be directly applied to solve this problem. In the next section we introduce some leading ideas that aim at solving this problem and lay out some observations to distinguish our proposed method. In Section 3.3 we present the LinkageTracker method. In Section 3.4 we report our experimental studies and results. Finally, in Section 3.5 we summarize the mechanisms behind LinkageTracker and its performances and benefits.

3.2 Related Work

There are generally two methods used for detecting disease genes, namely, the direct and the indirect methods. Techniques used in the direct method include allele-specific oligonucleotide hybridization analysis, heteroduplex analysis, Southern blot analysis, multiplex polymerase chain reaction analysis, and direct sequencing. A detailed description of these techniques is beyond the scope of this work but is available in Beaudet et. al [72] and Malcolm et. al [73]. Direct method requires that the gene responsible for the disease be identified and specific mutations within the gene characterized. As a result, direct method is frequently not feasible, and, the indirect method is used.

The indirect methods such as DMLE+ [74, 75], BLADE [5, 6], GeneRecon [7], HPM [2], and HapMiner [4] involve the detection of marker alleles that are very close to or are within the disease gene, such that they are inherited together with the disease gene generation after generation. Such marker alleles are known as haplotypes. Alleles at these markers often display statistical dependency, a phenomenon known as linkage disequilibrium or allelic association [76]. The identification of linkage disequilibrium patterns allows us to infer the disease gene location. Most commonly, linkage disequilibrium mapping involves the comparison of marker allele frequencies between disease chromosomes and control chromosomes.

DMLE+ proposed by Rannala & Reeve [74, 75] uses Markov Chain Monte Carlo method and coalescent model to allow Bayesian estimation of the posterior probability density of the position of a disease mutation relative to a set of markers. A standard-

coalescent model is a retrospective model of population genetics based on the genealogy of gene copies. It uses mathematics for describing the characteristics of the joining of lineages back in time to a common ancestor. This lineage joining is referred to as coalescence. The coalescent model provides the basis for estimation the expected time to coalescence and for establishing the relationships of coalescence times to the population size, age of the most recent common ancestor, and other population genetic parameters [77]. Rannala & Reeve [74, 75] proposed the use of intra-allelic coalescent process in prior-probability modeling. However, the model requires the specification of the age of the mutation, which is unlikely to be known. Furthermore, it is assumed that every sample sequence carries the disease mutation, the concern as to the suitability of this model for mutations with low relative population frequency was raised in [78]. And more importantly, the intra-allelic model assumes that all disease chromosomes descend from the same founding mutation event represented by single genealogy. However, even for Mendelian disorders, sporadic cases of disease are commonly observed and singleton founding-mutations are rare events [79].

Liu et al. proposed an algorithm BLADE which employed the Markov Chain Monte Carlo method (MCMC) for parameter estimations 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 solve some of the shortcomings in the algorithm proposed by Rannala & Reeve [74, 75], Liu et al. [5] and Mailund et al. [7] proposed an algorithm known as GeneRecon. GeneRecon combines the shattered coalescent method by Morris et al. [80] and the idea by Liu et al. [5] in separating the affected individuals into mutation clusters. Affected individuals in the same cluster are assumed to be descendants of a common founder. A null cluster is included for individuals affected due to environmental factors and not genetic factors. An MCMC algorithm of the Metropolis type [81] was used to integrate over unknown population genetic parameters of the shattered coalescence model and sample the marginal posterior probability density for the parameters of interest. Although GeneRecon is highly efficient in locating the disease locus on case/control data, the main drawback is that GeneRecon is very 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. [2] introduced a linkage disequilibrium mapping algorithm known as haplotype pattern mining (HPM). Firstly, HPM uses the association rule mining algorithm [36] to discover a set of highly associated patterns by setting the *Support* threshold to a certain value. Next, HPM uses chi-square test to discriminate disease association from control association. Finally, HPM computes the marker frequency for each of the markers. The frequency for each marker is computed by counting the number of associated patterns consisting of that specific marker. 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 use of exhaustive search method. As it uses association rule mining algorithm to discover highly associated patterns, and such patterns are rare in our problem of linkage disequilibrium mapping. The support threshold will need to be set at a very low value in order to discover those highly associated patterns. Combinatorial explosion occurs where many useless patterns will also be discovered together with the highly associated patterns.

Li and Jiang [4] proposed an algorithm known as HapMiner for the inference of disease gene location. HapMiner is an adaptation of an algorithm known as DBSCAN [82] which is a density based clustering method that is robust to noise. For each marker, HapMiner takes the haplotype segment around the marker and calculate the pair wise haplotype distances according to a distance measure proposed by Li and Jiang [4]. The DBSCAN algorithm is then applied to the distance matrix to identify clusters. A score for each marker will be calculated and the marker with the highest score is taken as the predicted location. 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. The experimental results in Li and Jiang [4] had shown that HapMiner outperformed algorithms such as HMP by Toivonen et al. [2, 3] and BLADE by Liu et al. [5, 6]. However, the main disadvantage of HapMiner is that it is very sensitive to its parameter values. This problem generally applies to the density based clustering method,

where the user needs to guess the optimal parameter values through trial-and-error, which may take a very long time to achieve the best guess.

To address some of the problems of the existing algorithms, we propose an algorithm known as LinkageTracker. LinkageTracker is model free; it does not require any population ancestry information about the disease and the genealogy of the haplotypes. Furthermore, LinkageTracker does no require the setting of complex parameters prior to the disease gene location inference process. *LinkageTracker* identifies the set of linkage disequilibrium patterns using a heuristic level-wise neighbourhood search and score each pattern by computing their *p*-values to ensure high discriminative powers of each pattern. After which, it infers the marker allele that is closest to the disease gene based on the *p*-value scores and allele frequencies of the set of linkage disequilibrium patterns. LinkageTracker is robust as it caters for missing or erroneous data by allowing gaps in between marker patterns. The initial work on *LinkageTracker* was published in [17].

3.3 LinkageTracker

3.3.1 Technical Representation

The general framework of the LinkageTracker is represented as a quintuple $<D, \Omega, L, \Psi, T>$ where

- D is a dataset consisting of M vectors $<x_1,\ldots, x_M>$, where each x_i is a vector $<d_{i1},\ldots, d_{in}>$ that describes the allele values of n genes/markers in a particular biological sample.

- For each position d_{*j}, $\omega_j = \{v_1,\ldots, v_t\}$ denotes the set of all possible expression values that d_{*j} could take on, and Ω is a collection of $\{\omega_1,\ldots, \omega_n\}$.

- A labelling for D is a vector $L = <l_1,\ldots, l_M>$, where the label l_i associated with x_i is either *abnormal* (a biological sequence derived from an individual exhibiting abnormality) or *normal* (a biological sequence belonging to a normal control).

- Ψ is the neighbourhood definition. The neighbourhood determines the maximum allowable gap size within each pattern. The gap setting enables *LinkageTracker* to be tolerant to noise. In the later section, the setting of gap size is described in detail and an optimal gap size is recommended based on expert knowledge on the characteristics of linkage disequilibrium.

- $T \in \Re^+$ is the threshold value for accepting a particular pattern. In statistical terms, T is the level of significance of the test. When the pattern score is less than T, the pattern is considered as significant, and is kept for further processing.

The output P is a set of linkage disequilibrium patterns with high discriminative powers. A pattern $p=<d_{*i}, d_{*j},\ldots,d_{*k}>$ where $p \in P$, such that $i < j < k$. Based on the set of patterns in P, we then infer the marker allele, that is closest to the disease gene. That is, for each marker allele, we combine the p-values of all patterns in P that consist of that marker allele. The method to combine p-values was first introduced by Fisher [83].

3.3.2 Algorithm LinkageTracker

There are two main steps in the LinkageTracker algorithm. Step 1 identifies a set of linkage disequilibrium patterns which is strong in discriminating the abnormal from the normal. Step 2 infers the marker allele that is closest to the disease gene based on the linkage disequilibrium patterns derived in Step 1.

3.3.2.1 Step 1: Discovery of Linkage Disequilibrium Pattern

LinkageTracker uses a statistical method known as odds ratio to score each potential/candidate pattern. After which the significance of the patterns is determined through comparing the pattern p-values to a value α that is dynamically computed at different search levels. Hence, we give the odds ratio scoring method, followed by a description of the computation of the dynamic α value. Finally, we present the level-wise neighbourhood searches for potential/candidate patterns.

Odds Ratio

Odds ratio is a statistical methodology that has been widely used in the biomedical arena to measure the magnitude of association between two categorical variables based on some data collected [84-86]. Odds ratio [87-89] provides a good measure of the magnitude of association between a pattern and the binary label *L*, which is crucial in determining the discriminative power of a pattern.

	Abnormal	**Normal**
not(*1,3*)	$P - \sigma$	$N - \pi$
(*1,3*)	σ	π

Table 3.1. : 2x2 contingency table

Given a pattern *x*, odds ratio computes the ratio of non-association between *x* and the label *L*, to the association between *x* and *L* based on a set of data. For example, given a pattern, say (1,3), we are interested in finding out whether the marker pattern (1,3) is strongly associated with the label *abnormal*. Table 3.1 shows the contingency table for our example. Odds ratio is defined as follows:

$$Odds\ Ratio,\ \theta = \frac{(P-\sigma)\pi}{(N-\pi)\sigma} \qquad (1)$$

The significance of a potential/candidate pattern is determined by computing its p-value. P-value calculates the probability due to chance alone of getting a difference larger than or equal to that actually observed in the data [90, 91]. A small p-value means it is difficult to attribute the observed difference to chance alone, and this can be taken as evidence against the null hypothesis of non-significance. We compare the p-value to α which is known as the level of significance of the test. α is the probability of type 1 error. A type 1 error occurs when the null hypothesis is wrongly rejected (when it should have been accepted). When the p-value is less than α, the difference is statistically significant, hence we can reject the null hypothesis at level α. In this work, the α value is dynamically determined at different search levels, which means that the value α is different for different level. If the *p*-value of a pattern is less than or equals to α, the

pattern is significant and is used for marker inference in the later stage. If the *p*-value is greater than α, the pattern is not significant, and it is discarded.

Computing *P*-Values from *Z-Statistics*

To find the p-value associated with a pattern, we need to first compute the z-statistics as follows:

$$z = \ln \theta \div \sqrt{\frac{1}{\sigma} + \frac{1}{\pi} + \frac{1}{(P-\sigma)} + \frac{1}{(N-\pi)}} \qquad (2)$$

The one-sided p-value is = 1 – Φ(z), where Φ(z) is the distribution function for N(0,1). The value of odds ratio is 0 or ∞ if any of the values in Table 3.1 is 0. In order to overcome this problem [92] and [93] suggested modifying the computation of the odds ratio to:

$$\tilde{\theta} = \frac{(\pi + 0.5)(P - \sigma + 0.5)}{(\sigma + 0.5)(N - \pi + 0.5)} \qquad (3)$$

The addition of 0.5 to each of the values in equation 3 is merely a device to avoid division by zero.

The LinkageTracker Algorithm

LinkageTracker mines patterns of the form $<d_{*i}, d_{*j},\ldots,d_{*k}>$. 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 significance of the pattern. Also the symbol "*" is ignored when computing the length of a marker pattern. Therefore, marker patterns (1,*,*,3), (1,*,3), and (1,3) are all considered as having length of 2.

A gap is a "*" symbol in between two known marker alleles. For instance, the marker patterns (1,*,*,*,3) has three gaps, (1,*,3) has one gap, and (1,3) has no gaps. The maximum number of gaps for this marker pattern (1,*,*,3,*,*,*,*,5) is four, as there are at most four gaps in between any two known marker alleles. The user is able to set the maximum number of gaps for the marker patterns. However, we recommend that a maximum allowable gap to be 6, giving the highest accuracy if the markers are spaced at 1 cM[2] or less. The detail of such a recommendation is given in the later section.

To find linkage disequilibrium patterns, one of the ways is to use the brute force method. That is, we could enumerate all possible marker patterns of length one, two, and three etc, and then compute the *odds ratio* of each of the 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 $\binom{n}{k} m^k$ marker patterns of length k, which we need

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

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, base on studies by Long & Langley [94], allelic associations are detectable within a genomic region of 20cM. Allelic associations beyond 20cM are weak and are not easily detectable. Therefore, enumerating marker patterns whose marker alleles are more than 20cM apart are unlikely to yield significant results. Based on this observation, *LinkageTracker* uses a 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 robustness against noise. For simplicity of illustration, all examples in this work assume that the markers are spaced at 1cM apart.

LinkageTracker is 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 combine 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 is concerned with a marker allele who shows significant allelic association with the disease gene when combine with other significant marker alleles but is insignificant when assessed alone. Let us denote this maker allele as *Mx*. This problem can be further divided into 2 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, two markers say *Mx* and *My* are both not significant at level 1, hence they will be discarded when forming marker patterns for level 2. Now, we have *Mi* which is an immediate neighbor of *My* showing significant allelic association in level 1 (assuming that the markers are ordered as follows: *Mi*, *My* and *Mx*). Hence, in level 2, *Mi* will be made to combine with its neighbors to form marker patterns of length 2. Since *My* is the immediate neighbor of *Mi*, *My* will be selected to form pattern with *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 as described above. Hence, in 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 less than 20 markers away from *Mz*, but is far enough such that *Mx* will not be selected by *Mz* to form marker pattern for the immediate next level. For example, from Figure 3.1, *Mx* and *Mz* is 8 markers apart. Assuming that the maximum allowable gap size is set to 2, *Mz* is made to combine with *Ma, Mb,* and *Mc* to form patterns of length 2. Assuming that (*Mz,Mc*) is tested significant, then (*Mz,Mc*) will combine with *Md, Me,*and *Mf* to form patterns of length 3.

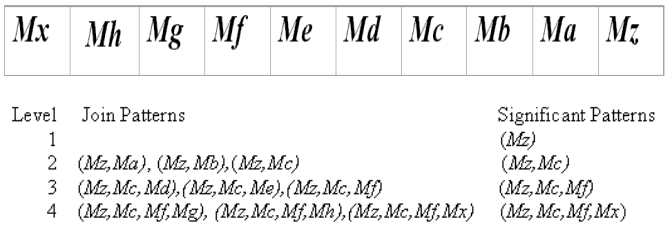


Figure 3.1: Illustration of marker positions

Assuming that (*Mz,Mc,Mf*) is tested significant, then (*Mz,Mc,Mf*) will combine 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*). Nevertheless, as in accordance with the studies by Long & Langley [94], most marker alleles exhibiting allelic associations with the disease gene will occur within a distance of 20cM from the disease gene, which means that marker alleles exhibiting allelic associations with the disease gene are quite densely packed within the 20 makers region. Hence, the chances of *LinkageTracke*r detecting significant marker alleles within the range of 20 markers are relatively high even though *LinkageTracker* is a heuristic method.

Dynamic Computation of α

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: $(1 - \alpha)^k$. For example, with $\alpha = 0.05$ and $k = 10$ we get $p = 0.95^{10} = 0.60$. This means that we now have a 40% chance that one of these 10 tests will

turn out significant, despite each individual test only being at the 5% level. In order to guarantee that the overall significance test is still at the α level, Bonferroni Correction [60] is usually applied; that is through dividing α by k to obtain the significance level for the individual tests.

Bonferroni Correction [60] requires the knowledge of the exact value of k which can be difficult to determine. LinkageTracker is an iterative process. In each iteration, 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 stage. As such we devise a new mechanism to the p-value to be used at each iteration or pattern length.

The idea is to set the p-value at iteration i, to be the significance level of the t most significant patterns at iteration $i-1$, achieving the same effect of raising the significance level as the patterns get longer. We have t defined below:

$$t = \frac{pattern\ set\ size\ at\ iteration\ i}{2^{(iteration\ i+1)}} \qquad (\mathbf{4})$$

For example, at the first iteration i = 0, we have pattern set of length 1 at the α level significance. We set the p-value for the next iteration to be median significance in the pattern set.

3.3.2.2 Step 2: Marker Inference

As mentioned in the earlier section, we infer the marker closest to the disease gene by combining the *p*-values of the highly associated patterns. Now, let us describe how we combine *p*-values from *n* patterns to form a single *p*-value. R.A. Fisher's method [83] 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 combine *p*-value and the frequency for each marker allele. In Figure 3.2(a), Marker 2 has allele 4 occurring four times, its combined *p*-value is $1.4 * 10^{-6}$, which is the chi-square distribution of $\sum(c) = 9.4211 + 10.0719 + 11.6183 + 10.8074 = 41.9186$ with 8 degree of freedom. Figure 3.2(b) depicts the combined *p*-value for each of the marker alleles from Figure 3.2(a). As we can see 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, then 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)

	Freq	∑(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)

Figure 3.2: a) Example of 5 linkage disequilibrium patterns.
b) Combine p-value of each marker allele from (a).

3.3.3 Setting the Optimal Number of Gaps

To accurately find the marker closest to the disease gene, it is important to determine the optimal number of gaps to use. The marker alleles that show significant allelic associations with the disease gene (within 20 markers region according to studies by Long & Langley [94]) should minimize the number of joins with neighbors beyond the 20 markers region. This is because the joining of a significant marker allele with some neighbors that are beyond the 20 markers region will inevitably introduce some false positive marker patterns or noise. Such false positive marker patterns will result in the reduction in accuracy during marker inference. On the other hand, we want to be as robust as possible, that is, to maximize the total possible gaps so as to cater for erroneous marker alleles. Based on these two conditions, we compute the *Score* for each gap setting *g* as follows for patterns of length 2:

$$Score(g) = \frac{\sum_{i=0}^{g} Robustness_i}{\sum_{i=0}^{g} Noise_i} \quad (4)$$

Table 3.2 shows the *Score* values for gap settings range from 0 to 20. Different gap settings will result in different values for *Noise* and *Robustness*. We shall now illustrate how the values of *Noise* and *Robustness* were computed with examples.

Num. of Gaps (*g*)	*Noise*	Num. Of patterns *p* form with *g* gaps	*Robustness* = *p x g*	*Score*(*g*)
0	1	19	0	0
1	2	18	18	6
2	3	17	34	8.67
3	4	16	48	10
4	5	15	60	10.67
5	6	14	70	10.95
6	7	13	78	**11**
7	8	12	84	10.89
8	9	11	88	10.67
9	10	10	90	10.36
10	11	9	90	10
11	12	8	88	9.59
12	13	7	84	9.14
13	14	6	78	8.67
14	15	5	70	8.17
15	16	4	60	7.65
16	17	3	48	7.11
17	18	2	34	6.56
18	19	1	18	6
19	20	0	0	0
20	21	0	0	0

Table 3.2. *Score* values for 0 to 20 gaps

Figure 3.3. The darken circle indicates the disease gene

3.3.3.1 Noise

Noise is defined as the maximum possible number of patterns consisting of markers beyond the 20 markers region. Figure 3.3 shows a disease gene that is very close to marker *M1*, markers *M21 and M22* are in dotted boxes as they are beyond the 20 makers region from the disease gene. Assuming that marker *M2* shows significant association with the disease gene, and we set the maximum allowable gaps to 1, then *M2* can join with its neighbors *M3* and *M4* to form patterns of length 2, i.e. *(M2,M3)* and *(M2,M4)*. Recall that the joining of a significant marker with some neighbors that are beyond the 20 markers region will introduce *Noise*. In this case, if markers *M19* and *M20* are significant, they will join with *M21* and *M22* to form patterns of length 2. We can see from Figure 3.4 that *M19* and *M20* will join with *M21* and *M22* in three ways, as illustrated by the dotted arrows. Hence, the maximum possible number of patterns consisting of markers beyond the 20 markers region (i.e. $\sum_{i=0}^{g} Noise_i$) is 3 when the gap setting g = 1 for this example. The *Noise* values for gap settings from 2 to 20 were computed similarly.

3.3.3.2 Robustness

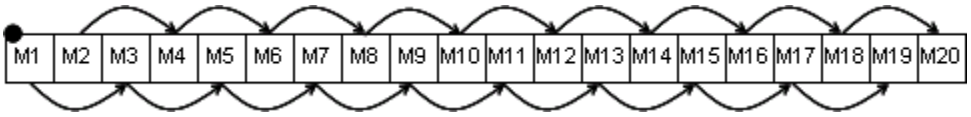


Figure 3.4. Joining of markers when gap setting *g* is 1

Before computing the *Robustness* values, we need to compute the maximum possible number of patterns *p* formed within the 20 markers region when the gap setting is *g*. When the gap setting *g* is set to 1, we can have at most 18 patterns (i.e. $p = 18$) as illustrated by the arrows in Figure 3.4. With the values of *p* for different values of *g*, we define *Robustness* as the maximum number of patterns formed within the 20 markers region weighted by the gap setting *g* itself:

$$Robustness = p \times g. \quad (5)$$

Recall that it is desirable to have wider gaps so as to cater for erroneous marker alleles, hence the value of *Robustness* increases as the value of *g* increases. As we can see from Table 3.2 that the gap setting of 6 has the highest *Score* value, hence we recommend that for a dataset with more than 20 markers to each chromosome (i.e. more than 20 attributes to each record) and each marker is spaced at 1cM apart, the optimal allowable gap setting should be 6.

To verify our above recommendation, we evaluated the performance of LinkageTracker by varying the gap settings from 2 to 10 on 100 realistically simulated datasets generated by Toivonen et al. [2] (details in the next section). The sum-square errors are computed for different gap settings *g* when applied to the 100 datasets. We find

that the gap setting of 6 has the lowest sum-square error, which means that it has the highest accuracy. This is in agreement with our recommendation above.

Evaluation

3.4.1 Time Complexity Analysis

The search space for the enumeration all frequent itemsets is exponential in the record length or the number of attributes to each record. For example, given a dataset with each record having n attributes and each attribute taking on 2 different values, the time complexity will be ${}_nC_k 2^k \approx O(n^k 2^k)$ where k is pattern length, the worst case occurs when $k=0.5(n)$. LinkageTracker uses expert knowledge by Long & Langley [94] and restricted the pattern length k to a value of 20.

3.4.2 Comparison of Performance on Real Datasets

We compared our algorithm LinkageTracker with some leading methods in linkage disequilibrium mapping such as BLADE [5, 6], GeneRecon [7], and HapMiner [4] on two real datasets, and 100 generated datasets. In this section we give the performance of the methods when applied to real datasets. The details on the performances on generated datasets are discussed in the next section.

3.4.2.1 Cystic Fibrosis

Cystic fibrosis is a well known real dataset reported in Kerem et. al [14]. The dataset contains haplotypes on 23 bi-allelic markers around the cystic fibrosis trans-membrane conductance regulator gene on chromosome 7q31.2. The control group has 92 haplotypes and the diseased group has 94. The founder mutation is located between marker 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 it provides us the opportunity to perform rigorous experiments using this dataset. For ease of reference, we divided the cystic fibrosis dataset into three subsets. Let 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.

Experimental Setting 1: Detection Accuracies

In this experiment we assess the algorithms' capability in detecting the disease gene location when only a small portion of the disease haplotypes actually carrying the founder mutation of interest, and others are genetically no different from control population at the locus of interest. Therefore datasets with different percentages of

founder mutation carrying disease haplotypes are generated (at 10%, 20%, 30%, 40% and 50%). For each percentage value we generate 5 different datasets each with 50 disease haplotypes and 50 controls.

For instance, to generate the disease haplotypes with 20% founder mutations, we randomly select 10 founder mutation carrying disease haplotypes from Set-A, and mix 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 select 50 samples to form the control haplotypes. This process is repeated 5 times to generate 5 datasets with 20% founder mutation. The datasets for other percentages of founder mutations are generated similarly.

Avg SSE	10%	20%	30%	40%	50%	Avg SSE over 5 different %
Blade	0.41200	0.42290	0.02427	0.02025	0.00691	0.17727
HapMiner	0.11264	0.02765	0.13234	**0.00380**	0.01647	0.05858
HapMiner (*x + x * 0.001*)	0.32505	0.09121	0.09087	0.04231	0.15701	0.14129
LinkageTracker	**0.01860**	0.02751	0.04065	0.01047	**0.00035**	**0.01952**
GeneRecon	0.03386	**0.016987**	**0.01810**	0.02246	0.01255	0.02079

Table 3.3: Comparison of predictive accuracies based on experimental setting 1

Table 3.3 shows the average sum-squared error of each of the algorithm at various percentages of disease haplotypes carrying the founder mutation. Detailed experimental results for each of the algorithm are given in the last section of this chapter. For the algorithm HapMiner, we assess its predictive accuracies based on the original parameter list provided by the authors [4] (they have used the same dataset in their work), and also based on the slightly modified parameter list. For the slightly modified parameter list, for

each numerical parameter value *x* in the original parameter list, we replace *x* with a new value *y*, such that $y = x + (x * 0.001)$. The performance of HapMiner given the original parameter list is labeled with "HapMiner", and the HapMiner given the modified parameter list is labeled as "HapMiner ($x + x * 0.001$)".

Generally, we expect that the predictive accuracies of an algorithm to improve as the percentage of disease haplotypes carrying the founder mutation increases. The algorithm BLADE shows, in general, such characteristics as shown in Table 1 that the sum-squared error (SSE) decreases as the percentage of disease haplotypes carrying the founder mutation increases. However, the rest of the algorithms do not show such characteristics. HapMiner fluctuates inconsistently at various percentage values, whereas LinkageTracker and GeneRecon show consistent predictive accuracies at different percentage values.

At both ends of the range, 10% and 50% of disease haplotypes carrying the founder mutation, LinkageTracker has the lowest SSE followed by GeneRecon (at 10%) or Blade (at 50%). At 20% and 40%, LinkageTracker coming in second, and is in third placing at 30%. Our objective when designing LinkageTracker is to have an algorithm for finding disease gene location even when the occurrence of disease haplotypes carrying the founder mutation is very small. The experimental results in Table 1 show that our objective for LinkageTracker is met. Furthermore, LinkageTracker continues to show good predictive accuracies as the percentage of disease haplotypes carrying the founder mutation increases, with SSE below 0.05 for the entire range. LinkageTracker also has the lowest average SSE over the five different percentage values.

Next we look at the average execution time of the algorithms (refer to Table 3.4). HapMiner is the fastest algorithm, given the original parameter list HapMiner takes about 3 seconds to execute, whereas given the slightly modified parameter list HapMiner takes about 5 seconds to execute. BLADE and LinkageTracker take over a minute to execute on the average, and GeneRecon takes over 2 hours.

Avg Time (seconds)	**10% (s)**	**20% (s)**	**30% (s)**	**40% (s)**	**50% (s)**	**Avg time over 5 different %**	**Avg time with Linkage Tracker as base unit**
Blade	75.50	72.67	63.37	71.98	73.82	1m 11.47s	0.74
HapMiner	**2.51**	**2.60**	**2.58**	**2.62**	**2.57**	**2.57s**	**0.03**
HapMiner (*x + x * 0.001*)	4.56	4.31	4.88	4.33	4.32	4.48172s	0.05
LinkageTracker	27.51	116.30	96.89	120.71	126.92	1m 36.66s	1
GeneRecon	10806.19	10318.56	10333.55	10593.06	10309.79	2hrs 54m 32.23s	108.33

Table 3.4: Comparison of run time based on experimental setting 1

In terms of predictive accuracies, GeneRecon is comparable with LinkageTracker. However, the execution time of GeneRecon is orders of magnitude longer than LinkageTracker. LinkageTracker is not the fastest algorithm. Some possible reasons may be that LinkageTracker uses the simple level wise search strategy for interesting patterns. Also our current implementation of LinkageTracker is not optimized. It is programmed in Java using complex data structures for fast prototyping, whereas the rest of the algorithms are coded in C/C++.

Experimental Setting 2: Noisy data

Next we assess the algorithms' performance when there are noises in the data. We are interested to know 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 that could influence the predictive accuracy of an algorithm. As 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. As mentioned earlier, there are three subsets for the cystic fibrosis dataset:

Set-A - 63 disease samples with the known founder mutation at the specific site.

Set-B - 31 disease samples without the founder mutation.

Set-C - 92 non disease control samples.

A dataset is generated as given in Table 3.5. For example, there are two main steps for generating datasets for the 10% mutation test. First we generate the disease set by 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 generate 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 is repeatedly performed for 5 test datasets at the same mutation level.

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 3.5: Data generation for experimental setting 2

Table 3.6 shows the average sum-squared error in predictions of each of the algorithm at various percentages of disease haplotypes carrying the founder mutation. LinkageTracker has the lowest average SSE, followed by GeneRecon. HapMiner would have performed well on this dataset if not for extremely poor performance at the 10% mutation set.

Avg SSE	10%	20%	30%	40%	50%	Avg SSE over 5 different %
Blade	0.12414	0.13140	0.18466	0.10704	0.13875	0.13720
HapMiner	0.42124	**0.00010**	**0.00010**	**0.00010**	**0.00010**	0.08433
HapMiner (*x + x * 0.001*)	0.63847	0.62987	0.55138	0.66370	0.55138	0.60696
LinkageTracker	**0.00742**	0.01580	0.01004	0.00232	0.00619	**0.00835**
GeneRecon	0.02467	0.01305	0.01078	0.02759	0.02283	0.01979

Table 3.6: Comparison of predictive accuracy based on experimental setting 2

Next we look at the average execution time of the algorithms (refer to Table 3.7). HapMiner is the fastest algorithm. Given the original parameter list, HapMiner takes about 1.5 seconds to execute, whereas given the slightly modified parameter list HapMiner takes about 6 seconds to execute. Although in terms of predictive accuracies, GeneRecon is comparable with LinkageTracker, the execution time of GeneRecon is orders of magnitude longer than LinkageTracker.

Avg Time (seconds)	10% (s)	20% (s)	30% (s)	40% (s)	50% (s)	Avg time over 5 different % (s)
Blade	47.31	44.03	49.37	50.15	48.41	47.85
HapMiner	**1.40**	**1.57**	**1.57**	**1.57**	**1.56**	**1.53**
HapMiner (x + x * 0.001)	6.37	5.84	5.83	5.92	6.17	6.03
LinkageTracker	204.55	136.38	172.78	141.16	111.61	153.29
GeneRecon	4867.00	4943.17	4923.15	4813.95	4845.85	4878.63

Table 3.7: Comparison of running time based on experimental setting 2

Experimental Setting 3

In this experiment, we assess the algorithms' performance when applied to the cystic fibrosis dataset without any modification to the ratios of the original disease haplotypes. Five datasets are generated for this experimental setting. The steps for generating the five datasets are: Firstly, samples from Set-A and Set–B are combined to form a new set, Set-X, which consists of 94 disease samples. Next, randomly pick 50 samples from Set-X to form the disease set. Lastly, pick 50 samples randomly from the 92 control samples (i.e. Set-C) to form the control set. The last 2 steps are repeated five times to form 5 datasets.

	Blade	HapMiner	HapMiner (*x + x * 0.01*)	GeneRecon	LinkageTracker
Avg SSE	0.01564	**0.00588**	0.58522	0.01466	0.00811
Avg Time (Seconds)	58.02020	**1.98600**	6.50040	4775.65660	125.53400

Table 3.8: Comparison of predictive accuracy and running time of the algorithms based on experimental setting 3

Table 3.8 shows the average sum-squared error of each of the algorithm for each of the 5 datasets. Detailed experimental results for each of the algorithm are given at the end of the chapter. LinkageTracker comes in second, marginally behind HapMiner which has the lowest SSE for experimental setting 3.

3.4.2.2 Friedreich Ataxia

Friedreich ataxia is an autosomal recessive degenerative disease that involves the central and peripheral nervous system and the heart. The data came from the Acadian population of Louisiana (Sirugo et al 1992). Campuzano et al (1996) identified the gene responsible for friedreich ataxia and discovered that the disease is caused by trinucleotide repeat expansion. The friedreich ataxia dataset was first reported by Liu et. al. [5] for linkage disequilibrium mapping. The friedreich ataxia dataset contains 54 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.

	Blade	HapMiner	HapMiner (*x + x * 0.01*)	GeneRecon	LinkageTracker
Avg SSE	10.367	**0.060**	0.416	-	0.135
Avg Time (Seconds)	742.515	**3.194**	3.801	-	108.192

Table 3.9: Comparison of predictive accuracy and running time of the algorithms when applied to the friedreich ataxia dataset

The experiments performed here using the friedreich ataxia dataset is similar to the experimental setting 3 in the previous section. The procedure of the data generation is as such: Firstly, pick 50 samples randomly from the 54 disease samples of the friedreich ataxia dataset. Next, pick 50 samples randomly from the 69 control samples of the friedreich ataxia dataset. The procedure is performed five times to form 5 datasets.

Table 3.9 shows the average sum-squared error of each of the algorithm for the 5 friedreich ataxia datasets. LinkageTracker is second to HapMiner in predictive accuracy. 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 alleles. Detailed experimental results for each of the algorithm can be found in the last section of this report.

3.4.2.3 Observations from the experiments on real datasets

From the experiments on the two real datasets, we see that in general, LinkageTracker and HapMiner have the best predictive accuracy, with HapMiner being the fastest algorithm. In instances where HapMiner is the better of the two, LinkageTracker follows closely behind HapMiner to give comparable predictions. It is

noted that the predictive accuracies of HapMiner with slightly modified parameter list are generally not as good when compared to all the other algorithms. This shows that HapMiner's performance is extremely sensitive to its parameter setting and robustness of the algorithm is a concern.

Based on the experimental results, HapMiner will be the best algorithm to use if the user knows exactly what values to set for each of its parameters. However, some parameters such as density threshold and radius may require many rounds of trial-and-error to achieve the optimal value. On the other hand, LinkageTracker produces good predictive accuracies and does not require the setting of complex parameters. Therefore, LinkageTracker will be a useful tool for linkage disequilibrium mapping when users do not have much information about their datasets.

3.4.3 Comparison of Performance on Generated Datasets

In this section we compare our algorithm LinkageTracker with HapMiner (given the original parameter list) on 100 generated datasets. The reason being HapMiner with original parameter list has shown to be efficient based on the results from real datasets in the previous section. Furthermore, HapMiner also made used of the same 100 generated datasets in their original papers [2]. The datasets used in this experiment were generated by Toivonen et al. [2]. Unfortunately the program HPM by Toivonen et al. [2] is not available to us. Nevertheless, we report the results of HPM in their original paper [2] and compare the predictive accuracies with LinkageTracker and HapMiner.

The datasets are downloadable from the following URL:

http://www.genome.helsinki.fi/eng/research/projects/DM/index-ajhg.html.

The simulated datasets correspond with the realistic isolated founder populations which grow from 300 to about 100,000 individuals over a period of 500 years. The simulation of isolated population is suited to linkage disequilibrium studies as recommended by Wright et al. [95].

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 labeled "*normal*". Each biological sequence consists of 101 markers. The datasets are 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.

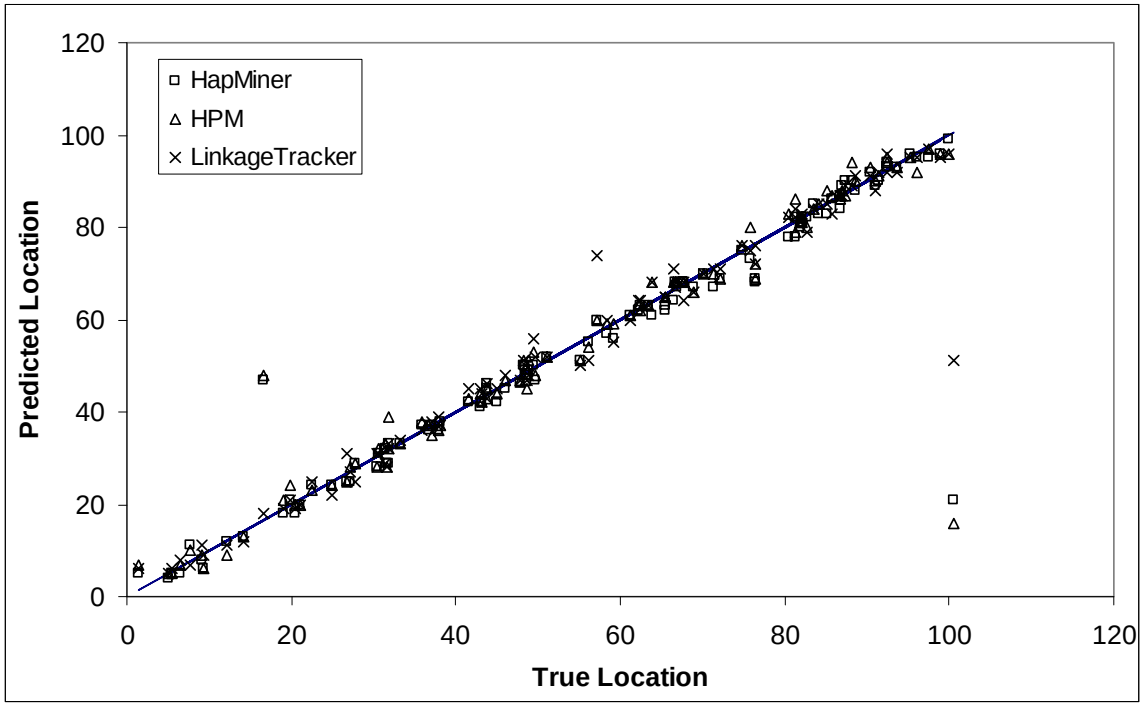


Figure 3.5: Comparison of prediction accuracy among HapMiner, *HPM* and *LinkageTracker*

Figure 3.5 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. Table 3.10 shows the predictive accuracy of HapMiner, LinkageTracker, and HPM over the 100 generated datasets. Among the three algorithms, LinkageTracker has the lowest SSE for the 100 datasets. It is observed that all the three algorithms did not perform well on the second dataset (refer to Table 3.10, row number 2), hence we exclude the second dataset in the performance assessment. LinkageTracker continues to be the algorithm with the lowest SSE, even after the exclusion of the second dataset for performance assessment.

Dataset	Exact Location	HapMiner	SSE (HapMiner)	HPM	SSE (HPM)	Linkage Tracker	SSE (Linkage Tracker)
1	86.9832	89	4.06748224	88	1.0338822	91	16.13468
2	100.497	21	6319.773009	16	7139.743	51	2449.953
3	85.1152	83	4.47407104	88	8.322071	85	0.013271
4	88.1118	90	3.56529924	94	34.670899	94	34.6709
5	27.1749	25	4.73019001	28	0.68079	27	0.03059
6	71.3791	67	19.17651681	70	1.9019168	71	0.143717
7	91.4263	90	2.03433169	91	0.1817317	92	0.329132
8	97.4294	95	5.90198436	97	0.1843844	97	0.184384
9	46.0612	45	1.12614544	47	0.8813454	48	3.758945
10	85.6649	86	0.11229201	87	1.782492	83	7.101692
11	56.1308	55	1.27870864	54	4.5403086	53	9.801909
12	95.2145	96	0.61701025	95	0.0460103	95	0.04601
13	96.0643	95	1.13273449	92	16.518534	95	1.132734
14	6.5231	5	2.31983361	7	0.2274336	6	0.273634
15	37.0228	37	0.00051984	35	4.0917198	36	1.04612
16	74.7825	75	0.04730625	76	1.4823063	76	1.482306
17	31.6615	29	7.08358225	28	13.406582	28	13.40658

18	88.4862	88	0.23639044	90	2.2915904	91	6.31919
19	86.8215	84	7.96086225	86	0.6748623	87	0.031862
20	65.406	63	5.788836	65	0.164836	65	0.164836
21	81.2496	78	10.55990016	79	5.0607002	82	0.5631
22	81.3287	82	0.45064369	86	21.821044	84	7.135844
23	63.4151	63	0.17230801	63	0.172308	59	19.49311
24	68.8194	67	3.31021636	66	7.9490164	66	7.949016
25	49.43	50	0.3249	53	12.7449	56	43.1649
26	92.4113	93	0.34656769	95	6.7013677	96	12.87877
27	7.6075	11	11.50905625	10	5.7240563	8	0.154056
28	82.7023	82	0.49322529	80	7.3024253	80	7.302425
29	67.6077	68	0.15389929	68	0.1538993	67	0.369299
30	31.8872	33	1.23832384	39	50.591924	32	0.012724
31	26.7347	25	3.00918409	25	3.0091841	31	18.19278
32	5.0485	4	1.09935225	5	0.0023522	5	0.002352
33	43.1726	42	1.37499076	42	1.3749908	44	0.684591
34	84.0212	83	1.04284944	85	0.9580494	85	0.958049
35	30.6477	31	0.12411529	32	1.8287153	31	0.124115
36	61.2179	61	0.04748041	61	0.0474804	60	1.48328
37	25.0116	24	1.02333456	24	1.0233346	27	3.953735
38	82.1955	81	1.42922025	81	1.4292202	83	0.64722
39	49.7319	47	7.46327761	48	2.9994776	52	5.144278
40	65.3964	62	11.53553296	65	0.157133	65	0.157133
41	86.7881	87	0.04490161	87	0.0449016	87	0.044902
42	48.5025	49	0.24750625	47	2.2575062	42	42.28251
43	62.4334	63	0.32103556	62	0.1878356	63	0.321036
44	16.6554	47	920.7947492	48	982.48395	18	1.807949
45	48.1984	50	3.24576256	51	7.8489626	51	7.848963
46	5.4983	5	0.24830289	5	0.2483029	5	0.248303
47	1.3383	5	13.40804689	7	32.054847	6	21.73145
48	80.4148	78	5.83125904	83	6.683259	80	0.172059
49	37.9742	36	3.89746564	36	3.8974656	39	1.052266
50	48.5517	48	0.30437289	45	12.614573	52	11.89077
51	98.8413	96	8.07298569	96	8.0729857	95	14.75559
52	87.4368	90	6.56999424	87	0.1907942	89	2.443594
53	33.1849	33	0.03418801	33	0.034188	34	0.664388
54	43.7423	45	1.58180929	45	1.5818093	46	5.097209
55	66.9502	67	0.00248004	68	1.10208	69	4.20168
56	41.5095	42	0.24059025	43	2.2215902	47	30.14559
57	19.8586	21	1.30279396	24	17.151194	21	1.302794
58	9.1709	8	1.37100681	9	0.0292068	9	0.029207
59	12.1537	12	0.02362369	9	9.9458237	11	1.331024
60	38.0134	38	0.00017956	37	1.0269796	37	1.02698
61	27.8384	29	1.34931456	29	1.3493146	25	8.056515
62	92.5326	94	2.15326276	94	2.1532628	92	0.283663
63	47.8187	46	3.30766969	47	0.6702697	47	0.67027
64	31.7271	29	7.43707441	32	0.0744744	32	0.074474
65	57.2332	60	7.65518224	60	7.6551822	74	281.1256

66	82.0091	82	8.281E-05	82	8.281E-05	82	8.28E-05
67	90.4501	92	2.40219001	93	6.50199	93	6.50199
68	67.7722	68	0.05189284	68	0.0518928	68	0.051893
69	55.1578	51	17.28730084	51	17.287301	53	4.656101
70	48.9422	49	0.00334084	51	4.2345408	52	9.350141
71	72.2161	69	10.34329921	69	10.343299	71	1.478899
72	9.3478	6	11.20776484	6	11.207765	9	0.120965
73	58.4323	57	2.05148329	59	0.3222833	55	11.78068
74	43.0613	41	4.24895769	44	0.8811577	44	0.881158
75	83.4535	85	2.39166225	84	0.2986622	84	0.298662
76	36.603	36	0.363609	37	0.157609	37	0.157609
77	62.1854	62	0.03437316	63	0.6635732	61	1.405173
78	35.95	37	1.1025	38	4.2025	39	9.3025
79	19.0096	18	1.01929216	21	3.9616922	21	3.961692
80	43.6985	46	5.29690225	43	0.4879023	43	0.487902
81	91.0723	89	4.29442729	90	1.1498273	91	0.005227
82	59.0882	56	9.53697924	59	0.0077792	55	16.71338
83	20.4244	18	5.87771536	20	0.1801154	19	2.028915
84	21.1371	20	1.29299641	20	1.2929964	22	0.744596
85	22.4228	24	2.48755984	23	0.3331598	23	0.33316
86	76.4812	69	55.96835344	72	20.081153	72	20.08115
87	75.7599	73	7.61704801	80	17.978448	76	0.057648
88	51.1806	52	0.67141636	52	0.6714164	52	0.671416
89	31.3206	32	0.46158436	33	2.8203844	36	21.89678
90	44.9818	42	8.89113124	44	0.9639312	45	0.000331
91	14.1838	13	1.40138244	13	1.4013824	14	0.033782
92	76.3524	68	69.76258576	69	54.057786	73	11.23859
93	70.1111	70	0.01234321	70	0.0123432	68	4.456743
94	93.5851	93	0.34234201	93	0.342342	94	0.172142
95	100.021	99	1.042441	96	16.168441	96	16.16844
96	66.6209	68	1.90191681	68	1.9019168	55	135.0453
97	66.4395	64	5.95116025	68	2.4351603	71	20.79816
98	30.4381	28	5.94433161	28	5.9443316	27	11.82053
99	63.9333	61	8.60424889	68	16.538049	65	1.137849
100	81.824	80	3.326976	81	0.678976	83	1.382976
Avg SSE			76.90774632		86.710212		**34.30228**
Avg SSE exclude dataset 2			13.84850125		15.467457		**9.901764**

Table 3.10: Comparison of predictive accuracies over 100 datasets

Discussion

We have introduced a new method for linkage disequilibrium mapping known as *LinkageTracker*. We compared LinkageTracker with some leading methods in linkage disequilibrium mapping. Experimental results show that *LinkageTracker* is highly accurate in both simulation-generated and real genetic datasets when compared to other methods. However, LinkageTracker is not superlative since HapMiner is faster in processing when compared to LinkageTracker. The predictive accuracies of HapMiner is very sensitive to its parameter values, and hence may not be the most efficient method to use when the user do not have sufficient knowledge to set the parameters. GeneRecon shows good predictive accuracies that are comparable to LinkageTracker. However, GeneRecon is very slow in processing that it requires hours to run a dataset with 23 markers and 100 samples. Furthermore, GeneRecon is not able to work on microsatellite makers with more than two alleles. The overall performance of LinkageTracker is promising as it provides good predictive accuracies while taking a reasonably short processing time, and also it is 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 use additional information such as genealogy of the haplotypes to improve performance when the additional information is available.

Chapter 4

ECTracker – Haplotype Analysis and Classification

Introduction

This chapter explores data mining methods that are capable of performing genetic analysis and carrier detection. Intuitively expressive patterns (or genetic variations) are extracted to provide insights about the genetic manifestations of patients affected by a disease. The extracted patterns are subsequently used for predictive inference (or classification) to help in carrier detection. In this chapter, we propose a new method known as ECTracker for pattern extraction and classification, and applied our algorithm on three real biological datasets. The first biological dataset consists of haplotypes of patients affected by hemophilia A from Singapore, and a set of matching unaffected control individuals [16]. The second and third datasets are Cystic Fibrosis and Friedreich Ataxia that are also used in the previous chapter for the finding of disease gene location. The performance of ECTracker in terms of expressiveness of patterns and predictive accuracies are compared to some leading methods in machine learning including C4.5, Naïve Bayesian Method, Artificial Neural Network, Support Vector Machine, K-Nearest Neighbor, Bagging (with Naïve Bayesian as base).

ECTracker

There are mainly two steps in the ECTracker. The first step finds all interesting patterns and the second step performs classification using those interesting patterns found in the first step. The basic idea of the ECTracker algorithm is to first derive all high precedence patterns for analysis, then, subsequently use the same high precedence patterns as a classifier.

4.2.1 Step 1 – Finding of Interesting Patterns

In the first step of ECTracker, we derive two sets of high precedence patterns; the first set pertaining to the disease samples and the second set pertaining to the normal/control samples. The algorithm for the finding of interesting patterns for this step is the same as the algorithm in Section 3.3.2.1 for the finding of linkage disequilibrium patterns. In other words, a level-wise neighborhood search method is used to find all significant patterns and the search is guided by the statistical odds ratio scores.

4.2.2 Step 2 – Predictive Inference or Classification

This section presents the algorithm for predictive inference using the patterns derived from the previous step. Before presenting the algorithm, let us define the order of precedence of the derived patterns. This is used in selecting patterns for our classifier.

Definition: Given two patterns, r_i and r_j, $r_i >> r_j$ (also called r_i precedes r_j or r_i has a higher precedence than r_j) if

1. The p-value of r_i is less than that of r_j, the smaller the p-value of a pattern the greater the statistical significance of that pattern.

2. Both patterns having the same p-values and $r_i \subset r_j$, the pattern length of r_i is shorter than that of r_j. The pattern with shorter pattern length that can correctly classify an unseen case is preferred.

3. Both patterns having the same p-values and $r_i \not\subset r_j$, but r_i is generated earlier than r_j.

Let R^d be the set of patterns pertaining to the disease samples and R^c be the set of patterns pertaining to the normal/control samples derived in step 1. The basic idea of the algorithm is to choose a set of high precedence patterns in R^d and a set of high precedence patterns in R^c as our classifier.

Let $R = R^d \cup R^c$ and D be the training data used to derive R, our classifier is of the following format: $<r_1, r_2, \ldots, r_n>$, $<v_1, v_2, \ldots, v_m>$, <*default_class*>, where $r_i \in R^d$, $r_a >> r_b$ if $b > a$, $v_i \in R^c$, $v_a >> v_b$ if $b > a$. The *default_class* is the chosen class for an unseen

case when no patterns in the classifier could classify the unseen case. The *default_class* can be specified by the user. However, if the user decides to let our classifier to select the *default*_class, then the majority class in the data *D* will be chosen as the *default_class*.

We shall now describe the algorithm for building our classifier. It consists of five steps:

Step 1: Generate patterns R^d and R^c with a given p-value *pv*.

Step 2: Sort the generated patterns in R^d and R^c according to the relation ">>". This is to ensure that we choose the highest precedence patterns for our classifier.

Step 3: For each pattern *r* in sorted R^d and for each pattern *v* in the sorted R^c, if there exists another pattern *r'* (or v') such that the p-values of both *r* and *r'* (or *v* and *v'*) are the same, and $r' \subset r$ (or $v' \subset v$), then remove *r* (or *v*) from sorted R^d (or R^c). This ensures that we choose the pattern with the shortest pattern length for each p-value. The top pattern set $\mathcal{R}$ for classification is formed with the remaining sorted sequence.

Step 4: Perform classification on the training data *D* using pattern classifier $\mathcal{R}$ and compute the true positive rate of the prediction.

Step 5: If the true positive rate is less than the user defined minimum true positive rate, then repeat Step 1 thru Step 4 using a different p-value *pv* to generate R^d and R^c.

We now describe the classification phase of Step 4 in greater details. In classifying an unseen case, the first top pattern/rule that matches the case perfectly will classify it. Given an unseen case *a* and a rule $r \in R^d$, *r* is said to be the perfect match of

case *a* if and only if *a* = *r*, this means that if *r* is a proper subset of *a* and the length of *r* is less than the length of *a*, then it is *not* a perfect match. In the case, when an unseen case *a* matches a top rule from R^d perfectly and also matches a top rule from R^c perfectly, then *a* will be classified as belonging to class R^d. If there is no pattern that perfectly matches the case, a scoring method will be used for each of the classes, the class with the highest score classifies the case. However, if the scoring method produces the same score for each of the available classes, then the unseen case will take on the default class. Figure 4-1 shows the pseudo code for scoring the classes given an unknown case pattern that does not match perfectly to any of the top patterns/rules.

1. Given an unknown case pattern $A = (a_{1, \ldots, a_k})$
2. Given a set of top rules $R^x \in \mathcal{R}$ that classify class C_x, $R^x = \{r_1,\ldots,r_n\}$
3. Given a set of top rules $R^y \in \mathcal{R}$ that classify class C_y, $R^y = \{r'_1,\ldots,r'_m\}$
4. For each item $a_i \in A$ do
5. For each rule $r_j \in R^x$ do
6. $$s_i = \sum_{j=1}^{n} \frac{\max|r|}{|r_j|} w_j \quad where \quad \begin{cases} w_j = 1 & if\ a_i \in r_j \\ w_j = 0 & otherwise \end{cases}$$
   where max|*r*| is the maximum length of rules in R^x
7. End
8. $$Score\left(A^{C_x}\right) = \sum_{i=1}^{k} s_i^2$$
9. For each rule $r'_l \in R^y$ do
   $$s'_i = \sum_{l=1}^{m} \frac{\max|r'|}{|r'_l|} w_l \quad where \quad \begin{cases} w_l = 1 & if\ a_i \in r'_l \\ w_l = 0 & otherwise \end{cases}$$
   where max|*r'*| is the maximum length of rules in R^y
10. End
11. $$Score\left(A^{C_y}\right) = \sum_{i=1}^{k} s'^2_i$$
12. End

Figure 4.1: Pseudo code for computing *score* of each class

The Hemophilia Dataset

Hemophilia A is an X-linked recessive bleeding disorder that results from deficiency and/or abnormality of coagulation factor VIII (FVIII) [ref]. The FVIII gene spans 186 kb of DNA and resides on 0.1% of the X chromosome (band Xq28). A set of five common PCR-based polymorphisms located on chromosome Xq28 which tags the hemophilia A disease gene were collected and analyzed from 47 patients and 47 matched normal controls. The five polymorphisms collected are two microsatellite repeats in introns 13 and 22, and three RFLPs namely *Bcl*I-intron 18, *Hind*III-intron 19, and *Xba*I-intron 22, the exact location of the markers are shown in Figure 4.2.

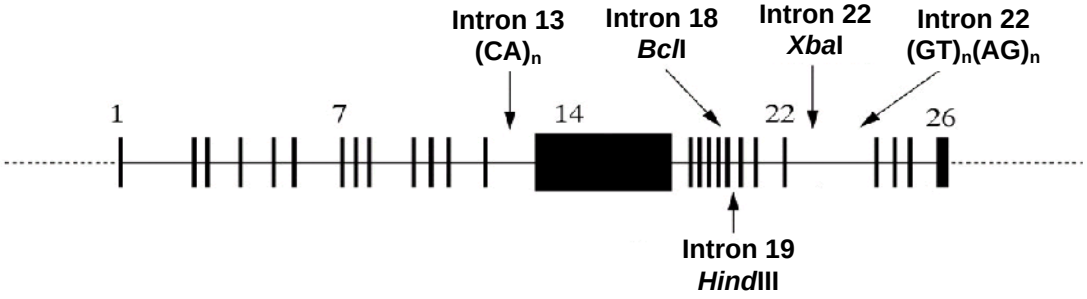


Figure 4.2: Factor VIII Gene

In the next section, we describe the allelic frequencies of Factor VIII gene observed in our local population and the allelic frequencies reported by the authoritative resource website [96] for hemophilia A disease. The reporting of the allelic frequencies of our local population is useful for other medical practitioners not located in Singapore to decide whether they could make use of our discovery of the genetic variations for prognosis and counseling of their patients.

4.3.1 Allelic Frequencies

RFLPs	Allele Frequencies (This Study)		Allele Frequencies (Reported by [35])	
	(-)	(+)	(-)	(+)
	1	2	1	2
BclI	**0.22**	**0.78**	**0.29**	**0.71**
HindIII	**0.78**	**0.22**	**0.75**	**0.25**
XbaI	**0.56**	**0.44**	**0.41**	**0.59**

Table 4.1: Allelic Frequencies of RFLPs

The allelic frequencies observed in this study and those reported by *Hemophilia A Mutation, Structure, Test and Resources Site* [96] are tabulated in Tables 4.1, 4.2, and 4.3. Our results for *Bcl*I, *Hind*III, and *Intron-13(CA)n* are significantly similar to those reported in [96] with $\chi 2 < 3.841$ (at 1 degree of freedom, and p-value>0.05) for *Bcl*I and *Hind*III, and $\chi 2 < 12.59$ (at 6 degree of freedom, p-value>0.05) for *Intron-13(CA)n*, they are all within 95% confidence interval. However, the frequencies for *Xba*I and *Intron22(GT)n(AG)n* are significantly different from those reported by [96] with $\chi 2 > 3.841$ (at 1 degree of freedom, and p-value < 0.05) for *Xba*I and $\chi 2 > 12.59$ (at 6 degree of freedom, p-value<0.05) for *Intron22(GT)n(AG)n*.

Intron 13 (CA)n Repeats	Allele Frequencies						
	24	23	22	21	20	19	15
	1	2	3	4	5	6	10
This Study	0.01	0.10	0.06	0.26	0.52	0.04	0.01
Reported by [35]	0.013	0.05	0.11	0.29	0.45	0.07	0

Table 4.2: Allelic Frequencies of Intron 13 $(CA)_n$ Repeats

Intron 22 (GT)n/(AG)n Repeats	Allele Frequencies						
	31	30	29	28	27	26	25
	1	2	3	4	5	6	7
This Study	0.01	0.01	0.04	0.03	0.09	0.63	0.19
Reported by [35]	0	0	0	0.013	0	0.667	0.307

Table 4.3: Allelic Frequencies of Intron 22 $(GT)_n/(AG)_n$ Repeats

It is observed that samples with *Bcl*I-intron 18 allele 1 are always associated with *Hind*III-intron 19 allele 2 with $\chi 2$ p-value < 0.001. The observation is expected as there is reported linkage disequilibrium between *Bcl*I and *Hind*III alleles from literature such as Ahrens et al. [97] and EL-Maarri et al. [98]. The *Hind*III marker is thus excluded since *Bcl*I and *Hind*III are in linkage disequilibrium, we could easily predict the value of the other attribute base on the value of one attribute, and hence 4 markers are sufficient in the analysis.

Furthermore, it is found that 70% of the samples have exactly the same allele values in all the markers in both patient and normal controls. This means that the 5 markers/attributes in the dataset are insufficient for separating 70% of the samples. After removing those samples whose disease and normal haplotypes cannot be distinguished, there are 28 samples remaining – 18 samples belonging to the disease phenotype and 10 samples belonging to the normal/control phenotype. Tables 4.4 and 4.5 show the frequencies of the disease and normal/control haplotypes respectively.

For descriptive analysis, we report on the expressive and interesting patterns extracted from the remaining 30% of the dataset, detailed description is given in Section 4.4.

For classification or predictive analysis, we divide out experiment into two parts. In the first part we assess the accuracies of the five classifiers based on the full hemophilia dataset. This part of the experiment further elaborated in Section 4.5.1. In the second part of the experiment, we concentrate our study on the 30% of the dataset where those samples whose disease and normal haplotypes cannot be distinguished are removed. The details are presented in Section 4.5.2.

Marker	**Disease Haplotypes**											**Total**
Intron-13 (CA)n	**3**	**4**	**4**	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**10**	
BclI	**1**	**2**	**2**	**2**	**2**	**1**	**2**	**2**	**2**	**1**	**2**	
XbaI	**1**	**1**	**1**	**2**	**2**	**1**	**2**	**1**	**2**	**1**	**1**	
Intron-22 (GT)n/(AG)n	**3**	**1**	**3**	**3**	**5**	**7**	**2**	**4**	**5**	**6**	**6**	
No. of Probands	**1**	**1**	**2**	**1**	**1**	**6**	**1**	**2**	**1**	**1**	**1**	**18**

Table 4.4: Haplotype Frequencies of Probands with Disease Phenotype

Markers	Normal/Control Haplotypes									Total
Intron-13 (CA)n	1	2	2	3	3	4	4	5	6	
BclI	1	1	2	1	1	2	2	2	2	
XbaI	1	1	2	1	1	1	1	1	2	
Intron-22 (GT)n/(AG)n	7	5	6	5	7	4	5	7	6	
No. of Probands	1	1	1	1	2	1	1	1	1	10

Table 4.5: Haplotype Frequencies of Probands with Normal/Control Phenotype

4.4 Descriptive Analysis – Interesting Pattern Extraction

In order to facilitate haplotype or genetic variations analysis, it is required that the data mining method be capable of generating a set of patterns or haplotypes (or genetic variations) such that the patterns are highly associated with the disease phenotype. Haplotype analysis is very useful in providing rapid information for genetic counseling. Among the popular machine learning methods mentioned earlier in the introduction section, only C4.5 is capable of producing descriptive patterns for haplotype analysis, so we compare ECTracker with it for descriptive analysis.

4.4.1 Expressive patterns derived by C4.5

C4.5 deduced that haplotype patterns (or genetic variations) of 4–*–*–*, 5–*–*–*, or 10–*–*–* (Intron13(CA)n–*Bcl*I–*Xba*I–Intron22(GT)n(AG)n) are highly associated with the disease phenotype. This derivation is not very useful as we can see from Table 4.5 that there are 3 probands with normal/control phenotype having intron-13 (CA)n allele values 4 and 5. Furthermore allele value 10 in intron-13 (CA)n only occurs once in the proband with disease phenotype (from Table 4.4). Hence it is not able to give a generalize conclusion based only on allele value 10 of intron-13 (CA)n.

The possible reason for such deduction of C4.5 may be due to the problem that the dataset is very small, and as a result the selection for partitioning attribute becomes bias for those attributes with more attribute values. Hence attributes with more attribute

values will be assigned higher information gain as compared to attributes with less attribute values.

4.4.2 Expressive patterns derived by ECTracker

As described in the earlier section of this chapter, the smaller the p-value of a pattern the higher the statistical significance of that pattern. Among the set of patterns derived by ECTracker, we select those patterns with the smallest p-value (i.e. most significant). There can be several patterns with the same lowest p-value, and we call these patterns the most significant patterns.

The longest most significant pattern associated with the disease phenotype derived by ECTracker is 4–1–1–7 (Intron13(CA)n–*Bcl*I–*Xba*I–Intron22(GT)n(AG)n). This is an interesting observation as the haplotype occurs in 33.3% of the disease phenotype and 0% of the normal/control phenotype with $\chi 2 > 3.841$, which means that such observation occurs significantly greater than by chance. From Table 4.4, the haplotype occurs in 6 probands with disease phenotype as compare to other haplotypes which occur in no more than 2 probands. The shortest most significant patterns derived by ECTracker are 4–*–*–7 or 4–1–*–*. This means that two markers alone are sufficient to define the disease haplotype. However, the longest most significant pattern provides a useful insight for the medical practitioners or scientists who seek to better understand the genetic variations of the disease.

This experiment shows that ECTracker is capable of deriving useful patterns even when the dataset is very small. As we could see that C4.5 is not able to handle such small dataset very well.

4.5 Predictive Analysis – Classification of the Hemophilia A Dataset

There are a total of 94 records in the hemophilia dataset, 47 records belonging to the class patient and 47 records belonging to the class normal. The classification methods that we use include C4.5, Naïve Bayesian Classifier, Neural Network, Support Vector Machine, K-Nearest Neighbor, Bagging (with Naïve Bayesian) and ECTracker. Except for ECTracker, all the other six classification algorithms are applied from WEKA. WEKA is an open source data mining and machine learning software [99].

4.5.1 Classification Based on Full Hemophilia Dataset

All the classifiers are evaluated using 5-fold cross-validation method. Table 4.6 below shows the performance of various classifiers including their respective precision and recall. Precision is defined as the proportion of instances that are correctly classified among all the instances that are predicted to be belonging to a particular class. Recall (which is the same as True Positive Rate) is the proportion of instances that are correctly classified among the all instances that are actually belonging to a class.

One of the possible settings of C4.5 in WEKA is the minimum number of instance per leave. The default value of this setting is 2. When we perform classification using the default settings for C4.5, 61.7% of the instances are classified correctly.

However, when we change the setting of the minimum number of instance per leave to 1, the accuracy improved by about 2%, i.e. from 61.7% improve to 63.8%.

Naïve Bayesian Classifier is the simplest classifier in WEKA as it does not require setting of any parameters. The predictive accuracy of Naïve Bayesian classifier is not as good as C4.5.

For Artificial Neural Network, some of the parameter settings in WEKA include the number of hidden layers, the number of epochs to train through, and the amount the weights are updated. We vary the number of hidden layers from 0 to 22 (i.e. number of attribute values + classes), and found that 2 hidden layers (i.e. number of classes) produces a classifier with the highest accuracy. When we vary the number of epochs to train from 500 to 1000, there is no change in the accuracy of the classifier. Next we vary the amount of the weights to update from 0.1 to 0.9, and we find that values from 0.4 to 0.6 produce the best accuracy of 54.26%.

Some of the parameter settings for Support Vector Machine include *filterType* which determines how the data will be transformed, *exponent* which determines the degree of the polynomial kernel, RBF kernel, and *gamma* which is a parameter setting for RBF kernel. There are three choices to the parameter *filterType*. The first choice is not to transform the data, second choice is to normalize the data, and third choice is to standardize the data. All three choices produce the same accuracy of 61.7%. Next we vary the degree of polynomial kernel from 1 to 4. The best result they produce is 61.7%. After that, we used RBF kernel varying the gamma from 0.01 to 0.1. The best result is 63.8%.

For K-Nearest Neighbor method, we vary the number of neighbors from 1 to 50, used different distance weighting function, and used normalized and non-normalized settings. The best result that we could obtain for the K-Nearest Neighbor method is 64.9%.

For Bagging with Naïve Bayesian method as the base classifier, we vary the size of each bag from 10% to 100%, and vary the number of iterations from 10 to 1000. The best result obtained is about 62.3%. We also tried using KNN as the base classifier since it produces good results as a single classifier. However, the results are similar to that of using Naïve Bayesian method as the base classifier.

For ECTracker, we vary the odds ratio p-values from 0.05 to 0.35 in training our classifier. It was found that a p-value of less than or equal to 0.2, which is within the 80% confidence interval produces the best classification results. As discussed previously in Section 4.3.1, 70% of the samples have exactly the same allele values in all the markers in both patient and normal controls. Hence when we set the p-value to a higher confidence interval such as within 95% confidence interval, no significant odds ratio patterns is found. The classification accuracy for ECTracker is 65.96%. As we can see from Table 4.6, ECTracker has the highest accuracy compared to all the other classifiers.

	Instances Correctly predicted	Instances Incorrectly Predicted	Accuracy	Precision for Class *Patient*	Recall for Class *Patient*	Precision for Class *Normal*	Recall for Class *Normal*
C4.5	60	34	63.83%	0.641	0.702	0.654	0.582
Naïve Bayesian Network	54	40	57.45%	0.585	0.524	0.576	0.627
Artificial Neural Network	51	43	54.26%	0.564	0.573	0.470	0.509
Support Vector Machine	60	34	63.83%	0.642	0.682	0.649	0.662
KNN	61	33	64.89%	0.659	0.640	0.650	0.665
Bagging	59	35	62.28%	0.659	0.636	0.625	0.627
ECTracker	62	32	**65.96%**	0.669	0.724	0.680	0.604

Table 4.6: Analysis of classifiers based on full hemophilia dataset

4.5.2 Classification Based on the Pruned Hemophilia Dataset

As described in the earlier section of this chapter, the dataset is insufficient to separate 70% of the samples. Hence in this section we concentrate our study on the remaining 30% of the samples. There are 28 samples remaining after removing those indistinguishable samples – 18 samples belonging to the disease phenotype and 10 samples belonging to the normal/control phenotype.

Similar to our previous analysis on the full hemophilia dataset, the classifiers used include C4.5, Naïve Bayesian Classifier, Artificial Neural Network, Support Vector Machine, K-Nearest Neighbor, Bagging, and ECTracker. However, we used the leave-one-out evaluation method rather than the 5-fold cross-validation method in this analysis. This is because we now have a smaller dataset and the leave-one-out evaluation method allows more data to be used for training the classifiers.

Table 4.7 shows the performance of various classifiers. We vary the parameter settings for the classifiers in a similar way as we did when we classified the full hemophilia dataset. Only the best results of the classifiers are shown in Table 4.7.

	Instances Correctly Predicted	**Instances Incorrectly Predicted**	**Accuracy**	**Precision of Class *Patient***	**Recall for Class *Patient***	**Precision for Class *Normal***	**Recall for Class *Normal***
C4.5	20	8	71.43%	0.708	0.944	0.75	0.3
Naïve Bayesian Network	18	10	64.29%	0.70	0.778	0.5	0.4
Artificial Neural Network	22	6	78.57%	0.833	0.833	0.7	0.7
Support Vector Machine	20	8	71.43%	0.75	0.833	0.625	0.5
KNN	23	5	82.14%	0.81	0.944	0.857	0.6
Bagging	18	10	64.29%	0.682	0.833	0.5	0.3
ECTracker	24	4	**85.71%**	0.818	1.0	1.0	0.6

Table 4.7: Analysis of classifiers based on pruned hemophilia dataset

From Table 4.7, we find that all classifiers show improvement in performance after we remove the indistinguishable samples. The classifier that gives the highest accuracy is ECTracke*r* with 85.71% predictive accuracy, and is followed by K-Nearest Neighbor with 82.14% predictive accuracy.

Next, we build the same set of classifiers listed in Table 4.7 with the pruned hemophilia dataset. The 70% of the inseparable samples that were being pruned earlier is used as the test dataset for the classifiers. Table 4.8 shows the performance of the various classifiers. ECTracker has the highest predictive accuracy.

	Instances Correctly Predicted	**Instances Incorrectly Predicted**	**Accuracy**	**Precision of Class *Patient***	**Recall for Class *Patient***	**Precision for Class *Normal***	**Recall for Class *Normal***
C4.5	29	37	43.94%	0.426	0.793	0.5	0.162
Naïve Bayesian Network	29	37	43.94%	0.426	0.793	0.5	0.162
Artificial Neural Network	29	37	43.94%	0.423	0.759	0.5	0.189
Support Vector Machine	29	37	43.94%	0.426	0.793	0.5	0.162
KNN	29	37	43.94%	0.435	0.931	0.5	0.054
Bagging	29	37	43.94%	0.426	0.793	0.5	0.162
ECTracker	36	30	**54.55%**	0.606	0.55	0.485	0.54

Table 4.8: Classification models built using pruned hemophilia dataset and tested on the 70% inseparable data

Since there are substantial amount of inseparable instances (66 instances in total) in the hemophilia dataset, we further modify the ECTracker algorithm to classify new unseen cases to a third class call *Unknown* if the cases are almost indistinguishable from those in the *Patient* and *Normal* classes. We modify Step 4 of the ECTracker algorithm (refer to Section 4.2.2). Given a new unseen case *u*, let $S_{patient}$ be the score of *u* for class *Patient* and S_{normal} be the score of *u* for class *Normal*. The original ECTracker will assign *u* as belonging to class *Patient* if $S_{patient} > S_{normal}$ and *u* to class *Normal* otherwise. The new modified ECTracker will assign the new unseen case *u* as belonging to class *Patient* if $S_{patient} > x*S_{normal}$ and assign *u* to class *Normal* if $x*S_{patient} < S_{normal}$, otherwise *u* will be assigned to class *Unknown.*

Table 4.9 shows the predictions of the modified ECTracker on the pruned hemophilia dataset and on the inseparable hemophilia dataset. As we can see that when x=1.5 it maximizes the number of correct prediction on the pruned dataset and also maximizes the prediction of the inseparable instances to the *Unknown* class.

	Instances from the pruned hemophilia dataset (28 instances in total)			**Inseparable instances from the hemophilia dataset (66 instances in total)**
x	**Instances correctly predicted**	**Instances incorrectly predicted**	**Instances predicted as unknown**	**Instances predicted as Unknown**
1.2	18	7	3	22
1.25	18	5	5	22
1.3	18	5	5	22
1.35	18	4	6	25
1.4	18	4	6	25
1.45	17	3	8	51
1.5	**17**	**2**	**9**	**53**
1.55	16	2	10	53
1.6	16	2	10	53
1.65	16	2	10	53

Table 4.9: Predictive accuracy of modified ECTracker

4.5.3 Classification Based on Cystic Fibrosis and Friedreich Ataxia Dataset

Finally we compared the predictive accuracies of the various machine learning methods (i.e. C4.5, Naïve Bayesian Method, Artificial Neural Network, Support Vector Machine, K-Nearest Neighbor, Bagging (with Naïve Bayesian as base) and ECTracker) when applied to the Cystic Fibrosis and Friedreich Ataxia datasets. Tables 4.10 and 4.11 show the predictive accuracies. ECTracker has the highest predictive accuracy for Cystic Fibrosis dataset, whereas Support Vector Machine has the highest predictive accuracy for Friedreich Ataxia dataset.

	Instances Correctly Predicted	**Instances Incorrectly Predicted**	**Accuracy**	**Precision of Class *Patient***	**Recall for Class *Patient***	**Precision for Class *Normal***	**Recall for Class *Normal***
C4.5	124	56	68.89%	0.65	0.711	0.725	0.667
Naïve Bayesian Network	132	48	73.33%	0.720	0.778	0.825	0.689
Artificial Neural Network	127	53	70.56%	0.631	0.778	0.824	0.633
Support Vector Machine	123	57	68.33%	0.615	0.722	0.782	0.644
KNN	123	57	68.33%	0.716	0.744	0.759	0.622
Bagging	131	49	72.78%	0.700	0.789	0.821	0.667
ECTracker	145	35	**80.56%**	0.799	0.856	0.833	0.756

Table 4.10: Classification accuracies when applied to Cystic Fibrosis dataset

	Instances Correctly Predicted	Instances Incorrectly Predicted	Accuracy	Precision of Class *Patient*	Recall for Class *Patient*	Precision for Class *Normal*	Recall for Class *Normal*
C4.5	87	33	72.50%	0.737	0.691	0.763	0.754
Naïve Bayesian Network	86	34	71.67%	0.658	0.818	0.809	0.631
Artificial Neural Network	74	46	61.67%	0.589	0.581	0.645	0.646
Support Vector Machine	88	32	**73.33%**	0.714	0.727	0.763	0.738
KNN	74	46	61.67%	0.577	0.618	0.659	0.615
Bagging	82	38	68.33%	0.629	0.800	0.795	0.585
ECTracker	75	45	62.5%	0.629	0.800	0.808	0.509

Table 4.11: Classification models built using Friedreich Ataxia dataset

4.6 Discussion

In this work, we re-examined the issues of descriptive and predictive analyses using our proposed method called ECTracker. In descriptive analysis, ECTracker is capable of extracting comprehensible and useful patterns from the hemophilia A dataset to facilitate haplotype analysis by medical practitioners. On the other hand, the patterns derived by C4.5 used only intron-13 (CA)n for prediction and this derivation is not very useful as described in Section 4.4.1. The main reason for the poor performance of C4.5 is

that the pruned hemophilia A dataset is very small. From this experimental result we show that ECTracker is capable of extracting useful patterns even when the dataset is very small.

In classification of hemophilia A dataset, ECTracker performed slightly better than the rest of the other classifiers (i.e. C4.5, Naïve Bayesian Method, Artificial Neural Network, Support Vector Machine, K-Nearest Neighbor, Bagging (with Naïve Bayesian as base)) on both un-pruned and pruned hemophilia A datasets, as shown in Tables 4.6 and 4.7. 70% of the inseparable data (or records) from the original hemophilia A dataset was removed to form the pruned dataset. Experiment was performed where the classifiers were built using the pruned dataset and tested using the 70% of the inseparable data. Table 4.8 shows the predictive accuracies when the classifiers are applied to the 70% inseparable data. *ECTracker* outperformed the rest of the classifiers by about 10% higher in predictive accuracy.

In classification of Cystic Fibrosis dataset, ECTracker outperformed the rest of the classifiers with about 10% higher in predictive accuracy as shown in Table 4.10. Furthermore the precision and recall values of ECTracker for both patient and normal class are also the highest among the classifiers.

In classification of Friedreich Ataxia dataset, Support Vector Machine has the highest predictive accuracy with about 10% higher in predictive accuracy compared to the rest of the algorithms (refer to table 4.11). However, the recall value of patient class for ECTracker and the precision value of normal class for ECTracker are both higher than Support Vector Machine. This means that ECTracker has lower false negative rate and higher false positive rate than Support Vector Machine on the Friedreich Ataxia dataset.

Some possible reasons for this are that there are 14 attribute values for each attribute in the Friedreich Ataxia dataset. During the training phase only 56 normal class data and 47 patient class data were used. There were insufficient data to learn for each attribute value. Furthermore, the Friedreich Ataxia dataset is originally used for linkage disequilibrium mapping. As such, only the patient class data exhibits statistical dependencies among attributes that are close to the disease gene, whereas dependencies among attributes in the normal class data are very much random. However, the F-measure for Support Vector Machine is 0.3602 and the F-measure for ECTracker is 0.3521, suggesting that the overall performance difference on this dataset between the two methods is very small.

In this work, we explored methods that are capable of extracting understandable and useful patterns, and also capable of performing inference on the patterns to make prediction. Through our experimental studies, we show that our proposed method ECTracker is capable of extracting useful patterns, and at the same time producing good predictive accuracies in classification that are comparable to the leading machine learning methods.

Chapter 5

Conclusion

5.1 Discussion

This thesis focuses on the knowledge extraction from haplotypes. First, the problem of pattern extraction for linkage disequilibrium mapping was examined. The major challenge is on how to maximize the haplotype information extraction in the association mapping of complex diseases in case–control studies under extreme conditions; in such conditions the occurrence of samples with the mutation of interest is very low, and consists of errors or noise. We proposed a new method called LinkageTracker to address the problem. Extensive performance studies show that the predictive accuracies of LinkageTracker are consistently good under different conditions; from the extremely difficult condition where the samples with the mutation of interest is as low as 10% and with high noise level, to the easier condition where the samples with the mutation of interest is as high as 50%. Experimental results in Section 3.4.2 elucidated the variation in predictive accuracies under different conditions for the various algorithms; LinkageTracker has low variations and with good predictive accuracies under all the different conditions. LinkageTracker and HapMiner have the best predictive accuracies in general. However, the variances in the sum-squared error of the predictions for HapMiner are higher than LinkageTracker for all the experiments. This means that LinkageTracker is more consistent in its predictions as compared to HapMiner when

applied to datasets with different conditions. Furthermore, the predictive accuracies of HapMiner with slightly modified parameter list are generally not as good when compared to all the other algorithms, which means that HapMiner's performance is extremely sensitive to its parameter setting. However, HapMiner is the fastest algorithm among all the algorithms. GeneRecon is comparable with LinkageTracker in terms of predictive accuracies; however, the execution time of GeneRecon is orders of magnitude longer than LinkageTracker. Furthermore, GeneRecon only works on bi-allelic markers. The overall performance of LinkageTracker is promising as it provides consistently good predictive accuracies while taking reasonably short processing times, and also it is easy to use since it does not require the setting of complex parameters.

Next, we examined methodologies capable of extracting useful and easily comprehensible patterns, and subsequently making use of the patterns extracted for classification. We proposed an algorithm called ECTracker to perform the tasks on haplotypes, to extract previously unknown, potentially useful and easily comprehensible haplotype patterns or genetic variations to provide insights about the genetic manifestations of diseases. The extracted patterns are subsequently used for classification to help in carrier detection. Extensive experiments were performed in comparing ECTracker with machine learning methods such as C4.5, Naïve Bayesian Method, Artificial Neural Network, Support Vector Machine, K-Nearest Neighbor, and Bagging (with Naïve Bayesian as base). Three real biological datasets were used in our experiments –namely the Hemophilia dataset, Cystic Fibrosis dataset and Friedreich Ataxia dataset. When comparing the expressiveness of patterns extracted with C4.5, we showed that ECTracker is capable of deriving more useful patterns when the dataset is

very small. In classification, ECTracker showed good performance in the Cystic Fibrosis dataset with the highest predictive accuracy, precision and recall compared to all the other methods. In instances where ECTracker is not the algorithm with the highest predictive accuracy, ECTracker exhibits comparability to the algorithm with the highest predictive accuracy with very small difference in the F-measures between the two algorithms. Furthermore, ECTracker has an extra feature whereby it allows samples to be classified as *unknown* if the samples are almost indistinguishable from the defined classes.

5.2 Future Research Directions

For LinkageTracker, we have restricted the interestingness of patterns to be guided by statistical odds ratio, generalizing to other types of scoring methods is certainly a possible extension. LinkageTracker is easy to use as it does not require any population ancestry information about the disease and the genealogy of the haplotypes as input. On the other hand, the main weakness of LinkageTracker is that it is not able to make use of the extra information (such as population ancestry information about the disease and the genealogy of the haplotypes) to improve performance even when the extra information is available. Hence, the next possible task will be to study how LinkageTracker can be improved to accept extra information for the prediction process.

For ECTracker, the predictive accuracy in the classification of Friedreich Ataxia dataset is not as good as some of the machine learning methods such as Support Vector Machine. However, it is observed that the recall for the class patient and the precision for

the class normal for ECTracker are both higher than Support Vector machine. This means that ECTracker is able to predict the patient class very well but not the normal class. And as in the discussion section of chapter 4, we mentioned that the possible reasons for such observations are that there are 14 attribute values to each attribute in the Friedreich Ataxia dataset, and there are insufficient data to learn for each attribute value. Also, Friedreich Ataxia dataset is originally used for linkage disequilibrium mapping, which means that only the patient class data exhibits statistical dependencies among attributes that are close to the disease gene, whereas dependencies among attributes in the normal class data are random. To improve the classification accuracies for datasets where only alleles within the disease chromosomes exhibit allelic associations that are higher than random chance and the allelic association within the normal chromosomes are equivalent to random chance, the finding of high precedence patterns pertaining only to the disease set may be worth exploring.

Bibliography

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed.: Morgan Kaufmann, 2006.

[2] H. Toivonen, P. Onkamo, K. Vasko, V. Ollikainen, P. Sevon, H. Mannila, M. Herr, and J. Kere, "Data Mining Applied to Linkage Disequilibrium Mapping," *American Journal of Human Genetics*, pp. 133-145, 2000.

[3] H. Toivonen, P. Onkamo, K. Vasko, V. Ollikainen, P. Sevon, H. Mannila, and J. Kere, "Gene Mapping by Haplotype Pattern Mining," in *Proceedings of IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE)*, 2001, pp. 99-108.

[4] J. Li and T. Jiang, "Haplotype-based linkage disequilibrium mapping via direct data mining," *Bioinformatics*, vol. 21, pp. 4384-4393, 2005.

[5] J. Liu, C. Sabatti, J. Teng, B. Keats, and N. Risch, "Bayesian Analysis of Haplotypes for Linkage Disequilibrium Mapping " *Genome Research*, vol. 11, pp. 1716-1724, 2001.

[6] X. Lu, T. Niu, and J. Liu, "Haplotype information and linkage disequilibrium mapping for single nucleotide polymorphisms," *Genome Research*, vol. 13, pp. 2112-2117, 2003.

[7] T. Mailund, M. H. Schierup, C. N. S. Pedersen, J. N. Madsen, J. Hein, and L. Schauser, "GeneRecon - A coalescent based tool for fine-scale association mapping," *Bioinformatics*, vol. 22, pp. 2317–2318, 2006.

[8] J. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann, 1993.

[9] P. Langley, W. Iba, and K. Thompson, "An Analysis of Bayesian Classifiers," in *International Conference on Artificial Intelligence*, 1992, pp. 223-228.

[10] L. Fu, *Neural Networks in Computer Intelligence*: McGraw-Hill, 1994.

[11] V. N. Vapnik, *The Nature of Statistical Learning*. New York: Springer, 1995.

[12] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37-66, 1991.

[13] L. Breiman, "Bagging predictors," *Machine Learning,* vol. 24, pp. 123-140, 1996.

[14] B. S. Kerem, J. M. Rommens, J. A. Buchanan, D. Markiewicz, T. K. Cox, and A. Chakravarti, "Identification of the cystic fibrosis gene: genetic analysis," *Science,* vol. 245, pp. 1073-1080, 1989.

[15] L. Lin, L. Wong, T. Y. Leong, and P. S. Lai, "Mining of Correlated Rules in Genome Sequences," in *Proceedings of the AMIA Conference,* San Antonio, Texas, 2002.

[16] L. Lin, L. Wong, T. Y. Leong, and P. S. Lai, "Mining of Disease Associated Haplotype Patterns for Hemophilia A," in *Asia-Pacific Conference on Human Genetics (HUGO)* Biopolis , Singapore, 2004.

[17] L. Lin, L. Wong, T. Y. Leong, and P. S. Lai, "LinkageTracker: A Discriminative Pattern Tracking Approach to Linkage Disequilibrium Mapping," in *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)*, Beijing - China, 2005, pp. 30-42.

[18] L. Lin, L. Wong, T. Y. Leong, and L. P.S., "ECTracker – An Efficient Algorithm for Haplotype Analysis and Classification," in *Proceedings of the 12th World Congress on Health (Medical) Informatics – Building Sustainable Health Systems*, 2007, pp. 1270-1274

[19] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus, "Knowledge Discovery in Databases: An Overview," *AI Magazine,* pp. 213-228, 1992.

[20] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*: MIT Press, 2001.

[21] R. Agrawal, H. Mannila, R. Srikant, H. Tiovonen, and A. I. Verkamo, "Fast Discovery of Association Rules," *Advances in Knowledge Discovery and Data Mining,* pp. 307-328, 1996.

[22] D. W. Cheung, V. T. Ng, and Y. Fu, "Efficient Mining of Association Rules in Distributed Databases," in *IEEE Transaction on Knowledge and Data Engineering,* 1996, pp.911-922.

[23] H. Mannila, H. Tiovonen, and I. Verkamo, "Efficient Algorithms for Discovering Association Rules," in *AAAI Workshop on Knowledge Discovery in Databases*, 1994, pp. 181-192.

[24] A. Sarasere, E. Omiecinsky, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," in *International Conference on Very Large Databases (VLDB)*, 1995, pp. 432-444.

[25] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," in *ACM SIGMOD International Conference on Managament of Data*, Washington, 1993, pp. 207-216.

[26] S. Brin, R. Motwani, and C. Silverstein, "Beyond Market Baskets: Generalizing Association Rules to Correlations," in *ACM SIGMOD International Conference on Managament of Data*, 1997, pp. 265-276.

[27] C. Silverstein, S. Brin, and R. Motwani, "Beyond Market Baskets: Generalizing Association Rules to Dependence Rules," *Data Mining & Knowledge Discovery*, pp. 39-68, 1998.

[28] C. C. Aggarwal and P. S. Yu, "A New Framework for Itemset Generation," in *Proceedings of the PODS Conference*, 1998.

[29] P. Tan and V. Kumar, "Interestingness Measures for Association Patterns: A Perspective," Research Report 00-036 Computer Science and Engineering, University of Minnesota, Twin Cities Research Report 00-036, 2000.

[30] H. Xiong, S. Shekhar, P. Tan, and V. Kumar, "Exploiting a Support-Based Upper Bound of Pearson's Correlation Coefficient for Efficiently Identifying Strongly Correlated Pairs," in *Proceedings of the Conference on Knowledge Discovery & Data Mining (KDD)*, 2004, pp. 334-343.

[31] H. Xiong, S. Shekhar, P. Tan, and V. Kumar, "Taper: An Efficient Two-Step Approach for All-Pairs Correlation Query in Transaction Databases," Research Report 03-020, Computer Science and Engineering, University of Minnesota, Twin Cities 2003.

[32] G. Dong and J. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," in *Proceedings of the Conference on Knowledge Discovery & Data Mining (KDD)*, 1999, pp. 43-52.

[33] G. Dong, J. Li, and X. Zhang, "Discovering Jumping Emerging Patterns and Experiments on Real Datasets," in *Proceedings of the International Database Conference on Heterogeneous and Internet Databases (IDC99)*, Hong Kong, 1999, pp. 15-17.

[34] H. Li, J. Li, L. Wong, M. Feng, and Y. P. Tan, "Relative Risk and Odds Ratio: A Data Mining Perspective," in *Proceedings of the PODS Conference*, Baltimore, Maryland, 2005, pp. 368-377.

[35] J. Li, X. Zhang, G. Dong, K. Ramamohanarao, and Q. Sun, "Efficient Mining of High Confidence Association Rules without Support Thresholds," in *Proceedings of the PKDD Conference*, 1999, pp. 406-411.

[36] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Association Rules," in *Proccedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, 1994, pp. 487-499.

[37] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules," in *Proceedings of the 7th ICDT Conference*, 1999, pp. 398-416.

[38] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient Mining of Association Rules Using Closed Itemset Lattices," *Information Systems*, pp. 25-46, 1999.

[39] M. J. Zaki, "Mining Non-Redundant Association Rules," *Data Mining and Knowledge Discovery*, pp. 223-248, 2004.

[40] M. J. Zaki and C. J. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining," in *Proceedings of the SIAM International Conference on Data Mining*, 2002, pp. 457-473.

[41] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," in *ACM SIGMOD International Conference on Management of Data*, 2000, pp. 1-12.

[42] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," *Data Mining & Knowledge Discovery*, pp. 53-87, 2004.

[43] B. Ganter and K. Reuter, "Finding All Closed Sets: A General Approach," in *ORDER*, 1991, pp. 283-290.

[44] R. J. Bayardo, "Efficiently Mining Long Patterns from Databases," in *ACM SIGMOD International Conference on Management of Data*, 1998, pp. 85-93.

[45] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad, "Depth First Generation of Long Patterns," in *Proceedings of the ACM SIGKDD Conference*, 2000, pp. 108-118.

[46] D. Burdick, M. Calimlim, and J. Gehrke, "MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases," in *Proceedings of ICDE*, 2001, pp. 443-452.

[47] G. Yang, "The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns," in *Proceedings of the Conference on Knowledge Discovery & Data Mining (KDD)*, 2004, pp. 344-353.

[48] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal, "Mining Frequent Patterns with Counting Inference," *ACM SIGKDD Explorations Newsletter*, pp. 66 - 75, 2000.

[49] W. DuMouchel and D. Pregibon, "Empirical Bayes Screening for Multi-Item Associations," in *Proceedings of the ACM SIGKDD Conference*, 2001, pp. 67-76.

[50] G. Grahne and J. Zhu, "Efficiently Using Prefix-Trees in Mining Frequent Itemsets," in *FIMI'03 Workshop on Frequent Itemset Mining Implementations*, 2003.

[51] R. Rymon, "Search Through Systematic Set Enumeration," in *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 1992, pp. 81-93.

[52] F. Pan, G. Cong, A. K. H. Tung, J. Yang, and M. Zaki, "CARPENTER: Finding Closed Patterns in Long Biological Datasets," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003, pp. 673-642.

[53] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn: Classification and Prediction Methods from Statistical, Neural Nets, Machine Learning, and Expert Systems*, San Francisco: Morgan Kaufman, 1991.

[54] D. Michie, D. Spiegelhalter, and C. Taylor., *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood, 1994.

[55] R. Rojas, *Neural Networks: A Systematic Introduction*: Springer-Verlag, 1996.

[56] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, 1998.

[57] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*: Cambridge University Press, 2000.

[58] S. Schölkopf, C. J. C. Burges, and A. J. Smola, *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, 1999.

[59] P. H. Winston, *Artificial Intelligence*, 3 ed.: Addison-Wesley, 1992.

[60] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.

[61] J. Quinlan, "Induction of Decision Trees," *Machine Learning*, pp. 81-106, 1986.

[62] J. R. Quinlan, "Simplifying Decision Trees," *International Journal of Man-Machine Studies*, pp. 221-234, 1987.

[63] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*: John Wiley & Sons, 1973.

[64] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss," *Machine Learning*, 1997.

[65] J. Pearl, "Bayesian networks: A model of self-activated memory for evidential reasoning," in *Proceedings of the 7th Conference of the Cognitive Science Society*, University of California, Irvine, 1985, pp. 329-334.

[66] N. E. Morton, "Linkage Disequilibrium Maps and Association Mapping," *Journal of Clinical Investigation*, pp. 1425-1430, 2005.

[67] N. Maniatis, A. Collins, C. F. Xu, L. C. McCarthy, D. R. Hewett, W. Tapper, S. Ennis, X. Ke, and N. E. Morton, "The First Linkage Disequilibrium (LD) Maps: Delineation of Hot and Cold Blocks by Diplotype Analysis," *Proceedings of the National Academy of Sciences of the United States of America*, pp. 2228-2233, 2002.

[68] A. Collins and N. E. Morton, "Mapping a Disease Locus by Allelic Association," *Proceedings of the National Academy of Sciences of the United States of America*, pp. 1741-1745, 1998.

[69] J. Hastbacka, A. de-la-Chapelle, I. Kaitila, P. Sistonen, A. Weaver, and E. Lander, "Linkage Disequilibrium Mapping in Isolated Founder Populations: Diastrophic Dysplasia in Finland," *Nature Genetics*, pp. 204-211, 1992.

[70] L. Ozelius, P. Kramer, D. de-Leon, N. Risch, S. Bressman, and D. Schuback, "Strong Allelic Association Between the Torsion Dystonia Gene (DYT1) and Loci on Chromosome 9q34 in Ashkenazi Jews," *American Journal of Human Genetics*, pp. 619-628, 1992.

[71] NCI, "National Cancer Institute. Cancer Facts http://cis.nci.nih.gov/fact/3_62.htm," 2002.

[72] A. Beaudet, C. Scriver, W. Sly, and D. Valle, "Genetics, Biochemistry, and Molecular Basis of Variant Human Phenotypes," in *The Metabolic and Molecular Basis of Inherited Disease*, 7th ed, C. R. Scriver, A. L. Beaudet, and W. S. Sly, Eds. New York: McGraw-Hill Inc, 1995, pp. 2351-2369.

[73] S. Malcolm, "Molecular Methodology," in *Emery and Rimoin's Principles and Practice of Medical Genetics*, 3rd ed, D. L. Rimoin, J. M. Connor, and R. E. Pyeritz, Eds. New York: Churchill Livingstone, 1997, pp. 67-86.

[74] B. Rannala and J. Reeve, "High-Resolution Multipoint Linkage-Disequilibrium Mapping in the Context of a Human Genome Sequence," *American Journal of Human Genetics*, pp. 159-178, 2001.

[75] J. Reeve and B. Rannala, "DMLE+: Bayesian Linkage Disequilibrium Gene Mapping," *Bioinformatics*, pp. 894-895, 2002.

[76] D. Goldstein and M. Weale, "Population Genomics: Linkage Disequilibrium Holds the Key," *Current Biology*, pp. R576-R579, 2001.

[77] M. Nordborg, "Coalescent theory," *Handbook of Statistical Genetics*, pp. 179–212 2001.

[78] C. Wiuf and P. Donnelly, "Conditional genealogies and the age of a neutral mutant," *Theoretical Population Biology*, vol. 56, pp. 183-201, 1999.

[79] E. Pennisi, "A closer look at SNPs suggests difficulties," *Science*, vol. 281, p. 17871789, 1998.

[80] A. P. Morris, J. C. Whittaker, and D. J. Balding, "Fine-scale mapping of disease loci via shattered coalescent modeling of genealogies," *The American Journal of Human Genetics*, vol. 70, pp. 686–707, 2002.

[81] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, pp. 1087-1092 1953.

[82] M. Ester, H. P. Kriegel, H. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial datasets with noise," in *Proceedings of Knowledge Discovery in Data (KDD)*, 1996, pp. 226–231.

[83] R. Fisher, *Statistical Methods for Research Workers*, 14th ed. New York: Hafner/MacMillan, 1970.

[84] E. M. John, G. G. Schwartz, J. Koo, D. Van Den Berg, and S. A. Ingles, "Sun Exposure, Vitamin D Receptor Gene Polymorphisms, and Risk of Advanced Prostate Cancer," *Cancer Research,* pp. 5470-5479, 2005.

[85] N. Le Moual, V. Siroux, I. Pin, F. Kaufmann, and S. M. Kennedy, "Asthma Severity and Exposure to Occupational Asthmagens," *American Journal of Respiratory Critical Care Medicine,* 2005.

[86] D. Deheinzelin, M. T. Lourenco, C. L. Costa, and R. N. Younes, "The Level of Nicotine Dependence is an Independent Risk Factor for Cancer: A Case Control Study," *Clinics*, vol. 60, pp. 221-226, 2005.

[87] S. K. Kachigan, *Statistical Analysis*: Radius Press, 1986.

[88] J. A. Rice, *Mathematical Statistics and Data Analysis*: Wadsworth, 1993.

[89] A. Tamhane and D. Dunlop, *Statistics and Data Analysis: From Elementary to Intermediate*: Prentice Hall, 2000.

[90] L. Ott, *An Introduction to Statistical Methods and Data Analysis*. Massachussetts: PWS-Kent Publishing Company, 1988.

[91] S. J. Theodorou, D. J. Theodorou, and Y. Kakitsubata, "Statistical Analysis in Clinical Studies: An Introduction to Fundamentals for Physicians," *Internet Medical Journal,* 2004.

[92] J. Haldane, "The estimation and significance of the logarithm of a ratio of frequencies," *Annals of Human Genetics,* vol. 20, pp. 309-311, 1956.

[93] A. Agresti, *Categorical Data Analysis*: Wiley, 2002.

[94] A. Long and C. Langley, "The Power of Association Studies to Detect the Contribution of Candidate Genetic Loci to Variation in Complex Traits," *Genome Research,* vol. 9, pp. 720-731, 1999.

[95] A. Wright, A. Carothers, and M. Pirastu, "Population Choice in Mapping Genes for Complex Diseases," *Nature Genetics,* vol. 23, pp. 397-404, 1999.

[96] HAMSTeRS, *Haemophilia A Mutation, Structure, Test and Resource Site*.

[97] P. Ahrens, T. A. Kruse, M. Schwartz, P. B. Rasmussen, and N. Din, "A New HindIII Restriction Fragment Length Polymorphism in the Hemophilia A Locus," *Human Genetics,* vol. 76, pp. 127-128, 1987.

[98] O. EL-Maarri, K. Kavakli, and H. Caglayan, "Intron 22 Inversions in the Turkish Haemophilia A Patients: Prevalence and Haplotype Analysis," *Haemophilia,* vol. 5, pp. 169-173, 1999.

[99] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*: Morgan Kaufmann, 2000.

Appendix A

Detail Experimental Results

A.1 Cystic Fibrosis from Section 3.4.2.1

Experimental Setting 1

BLADE

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	1.5426	85.072	-0.6626	0.43904
Set 2	0.88	1.2259	73.19	-0.3459	0.11965
Set 3	0.88	1.3835	75.75	-0.5035	0.25351
Set 4	0.88	0.1258	70.835	0.7542	0.56882
Set 5	0.88	1.704	72.64	-0.824	0.67898
			75.4974		0.412

Table A.1.1: BLADE in Exp Setting 1 with 10% Founder Mutation

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.1053	77.965	0.7747	0.60016
Set 2	0.88	0.1616	78.48	0.7184	0.5161
Set 3	0.88	0.1886	73.74	0.6914	0.47803
Set 4	0.88	0.7102	56.28	0.1698	0.02883
Set 5	0.88	0.179	76.89	0.701	0.4914
			72.671		0.42291

Table A.1.2: BLADE in Exp Setting 1 with 20% Founder Mutation

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.6919	60.729	0.1881	0.03538
Set 2	0.88	0.6828	61.845	0.1972	0.03889
Set 3	0.88	0.7726	69.103	0.1074	0.01153
Set 4	0.88	0.7503	63.724	0.1297	0.01682
Set 5	0.88	0.7431	61.44	0.1369	0.01874
			63.3682		0.02427

Table A.1.3: BLADE in Exp Setting 1 with 30% Founder Mutation

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7209	67.314	0.1591	0.02531
Set 2	0.88	1.1344	84.11	-0.2544	0.06472
Set 3	0.88	0.8151	70.97	0.0649	0.00421
Set 4	0.88	0.8329	66.217	0.0471	0.00222
Set 5	0.88	0.8107	71.27	0.0693	0.0048
			71.9762		0.02025

Table A.1.4: BLADE in Exp Setting 1 with 40% Founder Mutation

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8251	75.904	0.0549	0.00301
Set 2	0.88	0.7959	76.33	0.0841	0.00707
Set 3	0.88	0.9292	73.824	-0.0492	0.00242
Set 4	0.88	0.9487	72.46	-0.0687	0.00472
Set 5	0.88	0.7484	70.587	0.1316	0.01732
			73.821		0.00691

Table A.1.5: BLADE in Exp Setting 1 with 50% Founder Mutation

HapMiner

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8598	2.501	0.0202	0.000408
Set 2	0.88	1.6298	2.483	-0.7498	0.5622
Set 3	0.88	0.8598	2.526	0.0202	0.000408
Set 4	0.88	0.8698	2.504	0.0102	0.000104
Set 5	0.88	0.8698	2.516	0.0102	0.000104
			2.506		0.1126448

Table A.1.6: HapMiner in Exp Setting 1 with 10% Founder Mutation

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8598	2.62	0.0202	0.000408
Set 2	0.88	0.5948	2.584	0.2852	0.081339
Set 3	0.88	0.6848	2.602	0.1952	0.038103
Set 4	0.88	0.7448	2.601	0.1352	0.018279
Set 5	0.88	0.8898	2.582	-0.0098	9.604E-05
			2.5978		0.027645

Table A.1.7: HapMiner in Exp Setting 1 with 20% Founder Mutation

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	2.593	0.0102	0.000104
Set 2	0.88	0.6148	2.57	0.2652	0.070331
Set 3	0.88	0.7098	2.562	0.1702	0.028968
Set 4	0.88	1.6298	2.576	-0.7498	0.5622
Set 5	0.88	0.8698	2.575	0.0102	0.000104
			2.5752		0.1323414

Table A.1.8: HapMiner in Exp Setting 1 with 30% Founder Mutation

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	2.574	0.0102	0.000104
Set 2	0.88	0.8698	2.556	0.0102	0.000104
Set 3	0.88	0.8698	2.577	0.0102	0.000104
Set 4	0.88	0.8598	2.55	0.0202	0.000408
Set 5	0.88	0.7448	2.853	0.1352	0.018279
			2.622		0.0037998

Table A.1.9: HapMiner in Exp Setting 1 with 40% Founder Mutation

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	2.59	0.0102	0.000104
Set 2	0.88	0.8598	2.559	0.0202	0.000408
Set 3	0.88	0.8598	2.565	0.0202	0.000408
Set 4	0.88	0.8698	2.569	0.0102	0.000104
Set 5	0.88	0.5948	2.578	0.2852	0.081339
			2.5722		0.0164726

Table A.1.10: HapMiner in Exp Setting 1 with 50% Founder Mutation

HapMiner (*x + x * 0.001*)

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0	4.071	0.88	0.7744
Set 2	0.88	1.6298	5.589	-0.7498	0.56220004
Set 3	0.88	0.5698	4.993	0.3102	0.09622404
Set 4	0.88	0.5698	4.062	0.3102	0.09622404
Set 5	0.88	0.5698	4.098	0.3102	0.09622404
			4.5626		0.32505443

Table A.1.11: HapMiner(*x + x * 0.001*) in Exp Setting 1 with 10% Founder Mutation

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.5698	4.322	0.3102	0.09622404
Set 2	0.88	0.5348	4.29	0.3452	0.11916304
Set 3	0.88	0.7448	4.313	0.1352	0.01827904
Set 4	0.88	0.5698	4.314	0.3102	0.09622404
Set 5	0.88	0.5248	4.328	0.3552	0.12616704
			4.3134		0.09121144

Table A.1.12: HapMiner(*x + x * 0.001*) in Exp Setting 1 with 20% Founder Mutation

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.6198	4.487	0.2602	0.06770404
Set 2	0.88	0.5698	4.676	0.3102	0.09622404
Set 3	0.88	0.6848	5.569	0.1952	0.03810304
Set 4	0.88	0.5248	5.375	0.3552	0.12616704
Set 5	0.88	0.5248	4.31	0.3552	0.12616704
			4.8834		0.09087304

Table A.1.13: HapMiner(*x + x * 0.001*) in Exp Setting 1 with 30% Founder Mutation

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8598	4.406	0.0202	0.00040804
Set 2	0.88	0.5698	4.296	0.3102	0.09622404
Set 3	0.88	0.5698	4.291	0.3102	0.09622404
Set 4	0.88	0.7448	4.302	0.1352	0.01827904
Set 5	0.88	0.8598	4.351	0.0202	0.00040804
			4.3292		0.04230864

Table A.1.14: HapMiner(*x + x * 0.001*) in Exp Setting 1 with 40% Founder Mutation

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8598	4.392	0.0202	0.00040804
Set 2	0.88	0.7798	4.293	0.1002	0.01004004
Set 3	0.88	0.8898	4.324	-0.0098	9.604E-05
Set 4	0.88	0.8898	4.305	-0.0098	9.604E-05
Set 5	0.88	0	4.286	0.88	0.7744
			4.32		0.15700803

Table A.1.15: HapMiner(*x + x * 0.001*) in Exp Setting 1 with 50% Founder Mutation

LinkageTracker

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.9048	4.644	-0.0248	0.000615
Set 2	0.88	0.7798	28.583	0.1002	0.01004
Set 3	0.88	0.9048	12.588	-0.0248	0.000615
Set 4	0.88	0.5948	62.78	0.2852	0.081339
Set 5	0.88	0.8598	28.94	0.0202	0.000408
			27.507		0.018603

Table A.1.16: LinkageTracker in Exp Setting 1 with 10% Founder Mutation

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.6148	74.406	0.2652	0.070331
Set 2	0.88	0.8898	146.993	-0.0098	9.6E-05
Set 3	0.88	0.6548	132.737	0.2252	0.050715
Set 4	0.88	0.9598	130.817	-0.0798	0.006368
Set 5	0.88	0.7798	96.526	0.1002	0.01004
			116.2958		0.02751

Table A.1.17: LinkageTracker in Exp Setting 1 with 20% Founder Mutation

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.6548	161.485	0.2252	0.050715
Set 2	0.88	0.6548	96.494	0.2252	0.050715
Set 3	0.88	0.8598	73.874	0.0202	0.000408
Set 4	0.88	0.6548	70.475	0.2252	0.050715
Set 5	0.88	0.6548	82.098	0.2252	0.050715
			96.8852		0.040654

Table A.1.18: LinkageTracker in Exp Setting 1 with 30% Founder Mutation

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.6548	152.645	0.2252	0.050715
Set 2	0.88	0.8598	72.889	0.0202	0.000408
Set 3	0.88	0.8598	142.121	0.0202	0.000408
Set 4	0.88	0.8598	140.809	0.0202	0.000408
Set 5	0.88	0.8598	95.098	0.0202	0.000408
			120.7124		0.010469

Table A.1.19: LinkageTracker in Exp Setting 1 with 40% Founder Mutation

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	133.009	0.0102	0.000104
Set 2	0.88	0.8598	136.689	0.0202	0.000408
Set 3	0.88	0.8598	117.406	0.0202	0.000408
Set 4	0.88	0.8598	142.181	0.0202	0.000408
Set 5	0.88	0.8598	105.29	0.0202	0.000408
			126.915		0.000347

Table A.1.20: LinkageTracker in Exp Setting 1 with 50% Founder Mutation

GeneRecon

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.69161	10680.225	0.18839	0.035490792
Set 2	0.88	0.699861	10265.17	0.180139	0.032450059
Set 3	0.88	0.680032	10498.042	0.199968	0.039987201
Set 4	0.88	0.673054	11995.49	0.206946	0.042826647
Set 5	0.88	0.743741	10592.007	0.136259	0.018566515
			10806.1868		0.033864243

Table A.1.21: GeneRecon in Exp Setting 1 with 10% Founder Mutation

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.763706	10605.693	0.116294	0.013524294
Set 2	0.88	0.73932	10079.124	0.14068	0.019790862
Set 3	0.88	0.695696	10061.951	0.184304	0.033967964
Set 4	0.88	0.768839	10556.584	0.111161	0.012356768
Set 5	0.88	0.807234	10289.456	0.072766	0.005294891
			10318.5616		0.016986956

Table A.1.22: GeneRecon in Exp Setting 1 with 20% Founder Mutation

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.733864	10333.018	0.146136	0.02135573
Set 2	0.88	0.750975	10386.504	0.129025	0.016647451
Set 3	0.88	0.723467	10185.117	0.156533	0.02450258
Set 4	0.88	0.776673	10582.594	0.103327	0.010676469
Set 5	0.88	0.748439	10180.494	0.131561	0.017308297
			10333.5454		0.018098105

Table A.1.23: GeneRecon in Exp Setting 1 with 30% Founder Mutation

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.689306	10411.834	0.190694	0.036364202
Set 2	0.88	0.739475	11330.605	0.140525	0.019747276
Set 3	0.88	0.79596	10527.768	0.08404	0.007062722
Set 4	0.88	0.747506	10326.67	0.132494	0.01755466
Set 5	0.88	0.702263	10368.42	0.177737	0.031590441
			10593.0594		0.02246386

Table A.1.24: GeneRecon in Exp Setting 1 with 40% Founder Mutation

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.730001	10596.035	0.149999	0.0224997
Set 2	0.88	0.766118	10277.057	0.113882	0.01296911
Set 3	0.88	0.858114	10044.474	0.021886	0.000478997
Set 4	0.88	0.788015	10356.582	0.091985	0.00846124
Set 5	0.88	0.744648	10274.784	0.135352	0.018320164
			10309.7864		0.012545842

Table A.1.25: GeneRecon in Exp Setting 1 with 50% Founder Mutation

Experimental Setting 2

BLADE

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7414	48.511	0.1386	0.01920996
Set 2	0.88	0.7522	43.284	0.1278	0.01633284
Set 3	0.88	0.6498	42.893	0.2302	0.05299204
Set 4	0.88	0.2272	54.299	0.6528	0.42614784
Set 5	0.88	1.2056	47.557	-0.3256	0.10601536
			47.3088		0.124139608

Table A.1.26: BLADE in Exp Setting 2 with 10% Founder Mutation & Noise

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7253	41.604	0.1547	0.02393209
Set 2	0.88	0.8172	42.648	0.0628	0.00394384
Set 3	0.88	0.1044	53.643	0.7756	0.60155536
Set 4	0.88	0.7778	37.663	0.1022	0.01044484
Set 5	0.88	0.7491	44.571	0.1309	0.01713481
			44.0258		0.131402188

Table A.1.27: BLADE in Exp Setting 2 with 20% Founder Mutation & Noise

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	1.4354	51.981	-0.5554	0.30846916
Set 2	0.88	0.7377	46.973	0.1423	0.02024929
Set 3	0.88	0.131	53.028	0.749	0.561001
Set 4	0.88	0.7695	48.281	0.1105	0.01221025
Set 5	0.88	0.7339	46.563	0.1461	0.02134521
			49.3652		0.184654982

Table A.1.28: BLADE in Exp Setting 2 with 30% Founder Mutation & Noise

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7691	50.63	0.1109	0.01229881
Set 2	0.88	0.7421	50.054	0.1379	0.01901641
Set 3	0.88	0.7499	48.927	0.1301	0.01692601
Set 4	0.88	0.7473	50.428	0.1327	0.01760929
Set 5	0.88	0.1949	50.698	0.6851	0.46936201
			50.1474		0.107042506

Table A.1.29: BLADE in Exp Setting 2 with 40% Founder Mutation & Noise

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7665	55.283	0.1135	0.01288225
Set 2	0.88	0.7417	53.274	0.1383	0.01912689
Set 3	0.88	0.106	49.736	0.774	0.599076
Set 4	0.88	0.7009	43.057	0.1791	0.03207681
Set 5	0.88	0.7051	40.678	0.1749	0.03059001
			48.4056		0.138750392

Table A.1.30: BLADE in Exp Setting 2 with 50% Founder Mutation & Noise

HapMiner

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0	1.158	0.88	0.7744
Set 2	0.88	0.8698	1.158	0.0102	0.00010404
Set 3	0.88	1.6298	1.559	-0.7498	0.56220004
Set 4	0.88	0.6848	1.553	0.1952	0.03810304
Set 5	0.88	0.0248	1.571	0.8552	0.73136704
			1.3998		0.421234832

Table A.1.31: HapMiner in Exp Setting 2 with 10% Founder Mutation & Noise

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	1.552	0.0102	0.00010404
Set 2	0.88	0.8698	1.589	0.0102	0.00010404
Set 3	0.88	0.8698	1.568	0.0102	0.00010404
Set 4	0.88	0.8698	1.593	0.0102	0.00010404
Set 5	0.88	0.8698	1.545	0.0102	0.00010404
			1.5694		0.00010404

Table A.1.32: HapMiner in Exp Setting 2 with 20% Founder Mutation & Noise

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	1.556	0.0102	0.00010404
Set 2	0.88	0.8698	1.56	0.0102	0.00010404
Set 3	0.88	0.8698	1.593	0.0102	0.00010404
Set 4	0.88	0.8698	1.551	0.0102	0.00010404
Set 5	0.88	0.8698	1.591	0.0102	0.00010404
			1.5702		0.00010404

Table A.1.33: HapMiner in Exp Setting 2 with 30% Founder Mutation & Noise

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	1.544	0.0102	0.00010404
Set 2	0.88	0.8698	1.544	0.0102	0.00010404
Set 3	0.88	0.8698	1.569	0.0102	0.00010404
Set 4	0.88	0.8698	1.597	0.0102	0.00010404
Set 5	0.88	0.8698	1.588	0.0102	0.00010404
			1.5684		0.00010404

Table A.1.34: HapMiner in Exp Setting 2 with 40% Founder Mutation & Noise

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8698	1.56	0.0102	0.00010404
Set 2	0.88	0.8698	1.561	0.0102	0.00010404
Set 3	0.88	0.8698	1.559	0.0102	0.00010404
Set 4	0.88	0.8698	1.563	0.0102	0.00010404
Set 5	0.88	0.8698	1.577	0.0102	0.00010404
			1.564		0.00010404

Table A.1.35: HapMiner in Exp Setting 2 with 50% Founder Mutation & Noise

HapMiner (*x + x * 0.001*)

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0	5.818	0.88	0.7744
Set 2	0.88	0.0248	6.719	0.8552	0.73136704
Set 3	0.88	1.6298	5.852	-0.7498	0.56220004
Set 4	0.88	1.6298	6.902	-0.7498	0.56220004
Set 5	0.88	1.6298	6.573	-0.7498	0.56220004
			6.3728		0.63847343

Table A.1.36: HapMiner(*x + x * 0.001*) in Exp Setting 2 with 10% Founder Mutation & Noise

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	1.6298	5.737	-0.7498	0.56220004
Set 2	0.88	0.0248	5.856	0.8552	0.73136704
Set 3	0.88	1.6298	5.935	-0.7498	0.56220004
Set 4	0.88	0.0248	5.876	0.8552	0.73136704
Set 5	0.88	1.6298	5.82	-0.7498	0.56220004
			5.8448		0.62986684

Table A.1.37: HapMiner(*x + x * 0.001*) in Exp Setting 2 with 20% Founder Mutation & Noise

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	1.6298	5.828	-0.7498	0.56220004
Set 2	0.88	0.9048	5.838	-0.0248	0.00061504
Set 3	0.88	0.0248	5.814	0.8552	0.73136704
Set 4	0.88	0.0248	5.822	0.8552	0.73136704
Set 5	0.88	0.0248	5.859	0.8552	0.73136704
			5.8322		0.55138324

Table A.1.38: HapMiner(*x + x * 0.001*) in Exp Setting 2 with 30% Founder Mutation & Noise

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	1.6298	5.715	-0.7498	0.56220004
Set 2	0.88	0.0248	5.842	0.8552	0.73136704
Set 3	0.88	1.6298	5.762	-0.7498	0.56220004
Set 4	0.88	0.0248	6.41	0.8552	0.73136704
Set 5	0.88	0.0248	5.871	0.8552	0.73136704
			5.92		0.66370024

Table A.1.39: HapMiner(*x + x * 0.001*) in Exp Setting 2 with 40% Founder Mutation & Noise

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.9048	6.25	-0.0248	0.00061504
Set 2	0.88	0.0248	6.381	0.8552	0.73136704
Set 3	0.88	0.0248	5.864	0.8552	0.73136704
Set 4	0.88	0.0248	5.817	0.8552	0.73136704
Set 5	0.88	1.6298	6.559	-0.7498	0.56220004
			6.1742		0.55138324

Table A.1.40: HapMiner(*x + x * 0.001*) in Exp Setting 2 with 50% Founder Mutation & Noise

LinkageTracker

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.9598	470.825	-0.0798	0.006368
Set 2	0.88	0.7798	147.081	0.1002	0.01004
Set 3	0.88	0.9048	151.777	-0.0248	0.000615
Set 4	0.88	0.7798	115.838	0.1002	0.01004
Set 5	0.88	0.7798	137.222	0.1002	0.01004
			204.5486		0.007421

Table A.1.41: LinkageTracker in Exp Setting 2 with 10% Founder Mutation & Noise

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7798	90.644	0.1002	0.01004
Set 2	0.88	0.6198	194.256	0.2602	0.067704
Set 3	0.88	0.8598	142.993	0.0202	0.000408
Set 4	0.88	0.8598	169.285	0.0202	0.000408
Set 5	0.88	0.8598	84.706	0.0202	0.000408
			136.3768		0.015794

Table A.1.42: LinkageTracker in Exp Setting 2 with 20% Founder Mutation & Noise

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7798	157.833	0.1002	0.01004
Set 2	0.88	0.7798	229.491	0.1002	0.01004
Set 3	0.88	0.7798	125.324	0.1002	0.01004
Set 4	0.88	0.7798	177.79	0.1002	0.01004
Set 5	0.88	0.7798	173.46	0.1002	0.01004
			172.7796		0.01004

Table A.1.43: LinkageTracker in Exp Setting 2 with 30% Founder Mutation & Noise

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.7798	151.721	0.1002	0.01004
Set 2	0.88	0.8698	149.133	0.0102	0.000104
Set 3	0.88	0.9048	133.529	-0.0248	0.000615
Set 4	0.88	0.8598	148.473	0.0202	0.000408
Set 5	0.88	0.8598	122.949	0.0202	0.000408
			141.161		0.002315

Table A.1.44: LinkageTracker in Exp Setting 2 with 40% Founder Mutation & Noise

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.8598	129.973	0.0202	0.000408
Set 2	0.88	0.7798	111.198	0.1002	0.01004
Set 3	0.88	0.8598	113.558	0.0202	0.000408
Set 4	0.88	0.7798	95.551	0.1002	0.01004
Set 5	0.88	0.7798	107.757	0.1002	0.01004
			111.6074		0.006187

Table A.1.45: LinkageTracker in Exp Setting 2 with 50% Founder Mutation & Noise

GeneRecon

At 10%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.705008	4956.224	0.174992	0.0306222
Set 2	0.88	0.73948	4823.515	0.14052	0.01974587
Set 3	0.88	0.69765	4919.0305	0.18235	0.033251523
Set 4	0.88	0.751951	4560.585	0.128049	0.016396546
Set 5	0.88	0.727308	5075.69	0.152692	0.023314847
			4867.0089		0.024666197

Table A.1.46: GeneRecon in Exp Setting 2 with 10% Founder Mutation & Noise

At 20%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.736583	4864.913	0.143417	0.020568436
Set 2	0.88	0.775892	4872.767	0.104108	0.010838476
Set 3	0.88	0.774682	4819.892	0.105318	0.011091881
Set 4	0.88	0.73542	4928.143	0.14458	0.020903376
Set 5	0.88	0.836756	5230.18	0.043244	0.001870044
			4943.179		0.013054443

Table A.1.47: GeneRecon in Exp Setting 2 with 20% Founder Mutation & Noise

At 30%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.787769	4888.033	0.092231	0.008506557
Set 2	0.88	0.736737	4936.847	0.143263	0.020524287
Set 3	0.88	0.907675	5112.825	-0.02768	0.000765906
Set 4	0.88	0.811358	4763.55	0.068642	0.004711724
Set 5	0.88	0.740718	4914.5	0.139282	0.019399476
			4923.151		0.01078159

Table A.1.48: GeneRecon in Exp Setting 2 with 30% Founder Mutation & Noise

At 40%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.716819	4755.65	0.163181	0.026628039
Set 2	0.88	0.73512	4750.287	0.14488	0.020990214
Set 3	0.88	0.755525	4783.789	0.124475	0.015494026
Set 4	0.88	0.659016	4823.139	0.220984	0.048833928
Set 5	0.88	0.718682	4956.897	0.161318	0.026023497
			4813.9524		0.027593941

Table A.1.49: GeneRecon in Exp Setting 2 with 40% Founder Mutation & Noise

At 50%	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	0.88	0.698576	4941.474	0.181424	0.032914668
Set 2	0.88	0.756794	4863.058	0.123206	0.015179718
Set 3	0.88	0.758039	4944.97	0.121961	0.014874486
Set 4	0.88	0.722735	4821.097	0.157265	0.02473228
Set 5	0.88	0.717372	4658.66	0.162628	0.026447866
			4845.8518		0.022829804

Table A.1.50: GeneRecon in Exp Setting 2 with 50% Founder Mutation & Noise

Experimental Setting 3

Blade					
	Actual Location	Predicted Location	Error	SSE	Time (seconds)
Set 1	0.88	0.7468	0.1332	0.017742	59.509
Set 2	0.88	0.7544	0.1256	0.015775	56.732
Set 3	0.88	0.7443	0.1357	0.018414	59.866
Set 4	0.88	0.7832	0.0968	0.00937	56.323
Set 5	0.88	0.75	0.13	0.0169	57.671
Avg				0.01564	58.0202

Table A.1.51: Blade in Experimental Setting 3

HapMiner					
	Actual Location	Predicted Location	Error	SSE	Time (seconds)
Set 1	0.88	0.8698	0.0102	0.000104	3.396
Set 2	0.88	0.7098	0.1702	0.028968	1.545
Set 3	0.88	0.8698	0.0102	0.000104	1.555
Set 4	0.88	0.8698	0.0102	0.000104	1.862
Set 5	0.88	0.8698	0.0102	0.000104	1.572
Avg				0.005877	1.986

Table A.1.52: HapMiner in Experimental Setting 3

HapMiner (*x + x * 0.001*)					
	Actual Location	Predicted Location	Error	SSE	Time (s)
Set 1	0.88	0.9048	-0.0248	0.00061504	5.737
Set 2	0.88	0.0248	0.8552	0.73136704	6.349
Set 3	0.88	0.0248	0.8552	0.73136704	6.204
Set 4	0.88	0.0248	0.8552	0.73136704	6.203
Set 5	0.88	0.0248	0.8552	0.73136704	8.009
Avg				0.58521664	6.5004

Table A.1.53: HapMiner (*x + x*0.001*) in Experimental Setting 3

LinkageTracker					
	Actual Location	Predicted Location	Error	SSE	Time (seconds)
Set 1	0.88	0.7798	0.1002	0.01004	83.501
Set 2	0.88	0.7798	0.1002	0.01004	86.718
Set 3	0.88	0.7798	0.1002	0.01004	94.778
Set 4	0.88	0.7798	0.1002	0.01004	135.789
Set 5	0.88	0.8598	0.0202	0.000408	226.884
Avg				0.008114	125.534

Table A.1.54: LinkageTracker in Experimental Setting 3

GeneRecon					
	Actual Location	Predicted Location	Error	SSE	Time (seconds)
Set 1	0.88	0.821562	0.058438	0.003415	4788.231
Set 2	0.88	0.765088	0.114912	0.013205	4821.059
Set 3	0.88	0.80258	0.07742	0.005994	4780.1575
Set 4	0.88	0.778928	0.101072	0.010216	4766.607
Set 5	0.88	0.678859	0.201141	0.040458	4722.229
				0.014657	4775.6567

Table A.1.55: GeneRecon in Experimental Setting 3

A.2 Friedreich Ataxia from Section 3.4.2.2

Blade					
	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	9.8125	13.6597	742.909	-3.8472	14.80094784
Set 2	9.8125	7.8637	743.01	1.9488	3.79782144
Set 3	9.8125	4.2873	742.261	5.5252	30.52783504
Set 4	9.8125	9.0035	742.985	0.809	0.654481
Set 5	9.8125	8.3787	741.41	1.4338	2.05578244
Avg			742.515		10.36737355

Table A.2.1: Blade applied to Friedreich Ataxia Dataset

HapMiner					
	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	9.8125	9.75	3.258	0.0625	0.00390625
Set 2	9.8125	9.5	3.164	0.3125	0.09765625
Set 3	9.8125	9.5	3.226	0.3125	0.09765625
Set 4	9.8125	9.75	3.154	0.0625	0.00390625
Set 5	9.8125	9.5	3.168	0.3125	0.09765625
Avg			3.194		0.06015625

Table A.2.2: HapMiner applied to Friedreich Ataxia Dataset

HapMiner (*x + x * 0.001*)					
	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	9.8125	10.5	3.693	0.6875	0.47265625
Set 2	9.8125	10.25	3.676	0.4375	0.19140625
Set 3	9.8125	10.5	3.759	0.6875	0.47265625
Set 4	9.8125	10.5	3.7	0.6875	0.47265625
Set 5	9.8125	10.5	4.177	0.6875	0.47265625
Avg			3.801		0.41640625

Table A.2.3: HapMiner (*x + x*0.001*) applied to Friedreich Ataxia Dataset

LinkageTracker					
	Actual Location	Predicted Location	Time (s)	Error	SSE
Set 1	9.8125	10.25	118.024	0.4375	0.191406
Set 2	9.8125	10.125	114.006	0.3125	0.097656
Set 3	9.8125	10.25	97.97	0.4375	0.191406
Set 4	9.8125	9.5	99.11	-0.3125	0.097656
Set 5	9.8125	9.5	111.851	-0.3125	0.097656
			108.1922		0.135156

Table A.2.4: LinkageTracker applied to Friedreich Ataxia Dataset