Tree Pattern Mining for Gene Mapping

Petteri Sevon, Hannu T.T. Toivonen, Vesa Ollikainen

Dept. of Computer Science, University of Helsinki
P.O. Box 26, FIN-00014 University of Helsinki, Finland

Corresponding author:
Petteri Sevon
Tel. +358 9 191 44427
Fax. +358 9 191 44441
petteri.sevon@cs.helsinki.fi

ABSTRACT

We describe TreeDT, a novel gene mapping method. Gene mapping aims at discovering a statistical connection from a particular disease or trait to a narrow region in the genome. In a typical case-control setting of a gene mapping study, data consists of genetic markers typed for a set of disease-associated chromosomes and for a set of control chromosomes. The data from each chromosome constitutes its haplotype; a computer scientist would view haplotypes simply as strings.

TreeDT extracts, essentially in the form of substrings and haplotype trees, information about the historical recombinations in the population. This information is used to locate fragments potentially inherited from a common mutation-carrying founder, and to map the disease gene into the most likely such fragment. The method measures for each chromosomal location the disequilibrium of the tree of marker strings starting from the location, to assess the distribution of disease-associated chromosomes.

In this paper we give a detailed description of TreeDT, we analyze it formally, and we evaluate its performance experimentally on both simulated and real data sets. Experimental results demonstrate that TreeDT has high accuracy on difficult mapping tasks, and comparisons to state of the art methods (TDT, HPM) show that TreeDT is very competitive.

Keywords: Gene mapping, algorithms, permutation tests, haplotype trees.

1. INTRODUCTION

Gene mapping aims at discovering a statistical connection from a given trait or disease to a narrow region in the genome, which can then be further investigated by laboratory methods. Discovery of new disease susceptibility genes can have an immense importance for human health care. The gene and the proteins it produces can be analyzed to understand the disease causing mechanisms and to design new medicines. Further, gene tests on patients can be used to assess individual risks and for preventive and individually tailored medications. Obviously, gene mapping is receiving increasing interest among medical industry.

Genetic markers along chromosomes provide data that can be used to discover associations between patient phenotypes (e.g., diseased vs. healthy) and chromosomal regions (i.e., potential disease gene loci). The growing number of available genetic markers, anticipated to reach hundreds of thousands in the next few years, opens new opportunities but also amplifies the computational complexity of the task.

Human genome sequencing efforts, the first ones now practically complete, read the full human DNA sequence. There are methods for recognizing where there are genes in the sequence — the number of which is currently estimated to be around 30,000. However, we lack methods for deriving the function of a gene from the sequence information. Gene mapping approaches this problem for one disease at a time. It aims at discovering areas in the genome – hopefully small – that have a statistical connection to a given trait, thus narrowing down the area to be analyzed with expensive laboratory methods.

A typical setting for association or linkage disequilibrium based gene mapping is a case-control study of some chromosome of diseased and healthy individuals. Instead of looking at the DNA of the whole chromosome, only certain marker segments distributed along the chromosome are considered. By the analysis of similarities within the disease-associated chromosomes on one hand and the differences between the disease-associated and control chromosomes on the other, one can try to locate likely areas for a gene that predisposes people to the disease analyzed.

We introduce TreeDT, a novel method for gene mapping. It analyses the observed strings of markers by constructing tree-structured patterns that reflect the possible genetic history of a disease susceptibility (DS) gene. The gene is then predicted to be where the strongest genetic contribution is visible in the trees. The contributions of TreeDT are:

(1) A novel approach to gene mapping using tree patterns,

(2) An efficient algorithm for generating and testing tree patterns,

(3) A method for estimating the statistical significance of individual findings as well as the whole process, based on multiple permutations but carried out at the cost of a single permutation.

TreeDT was first introduced in reference [15]. This paper contributes to the topic in the following respects: (1) an algorithm for the permutation procedure is given, and readability of all algorithms is improved and they are also described in more detail, (2) the report on experiments has been extended and, in particular, experiments with a real data set have been added, and (3) algorithms are analyzed formally and claims about their properties are proved.

In the next section we review the gene mapping problem. In the following sections we specify the proposed method and then give algorithms for it. After a brief review of related work, we experimentally evaluate and compare the performance of the method. Finally, we conclude with some directions for future work.

2. PROBLEM BACKGROUND

Let us assume the goal is to locate a disease-susceptibility gene for a given disease. We next briefly review the genetic background; without loss of generality, we restrict the discussion in this paper to one chromosome. (In case one has several chromosomes to analyse, the results for different chromosomes are independent, and Bonferroni correction can be applied to the *p* values obtained by TreeDT for individual chromosomes.)

Marker Data A genetic *marker* is a short polymorphic region in the DNA, denoted here by M1, M2, …. The different variants of DNA that different people have at the marker locus are called *alleles*, denoted in our examples by 1, 2, 3, … . The number of alleles per marker is small: typically less than ten (for so called microsatellite markers) or exactly two (for so called SNPs). The collection of markers used in a particular study constitutes its *marker map*, and the corresponding alleles in a given chromosome constitute its *haplotype* (Figure 1). For the purposes of this paper the input data consists of haplotypes of diseased and control persons – or, in computer science terms, aligned allele strings, classified to positive and negative examples.

Linkage Disequilibrium All the current carriers of a disease-susceptibility mutation have inherited it from some founder individual who introduced it to the population (Figure 2). If there has been

only one such founder, then current carriers are related and segments from the mutation carrying founder chromosome are over-represented among the affected at mutation locus. Relatively young (e.g. 1000 years) population isolates are promising sources of data in this respect: a disease-susceptibility allele may have been introduced by one or two founders only, and the allele may be over-represented in the population. Kainuu region in eastern Finland is an example of such a fruitful area for genetic studies.

If there are conserved regions at the mutation locus, then it can be possible to observe *linkage disequilibrium* (LD), non-random association between nearby markers. One of our goals in this paper is to develop a method that suits well situations where there are a few mutation-carrying founders.

Gene Mapping In diseases with a reasonable genetic contribution, and especially in population isolates where few founders have introduced the mutation, affected individuals are likely to have higher frequencies of founder alleles and haplotype patterns near the DS gene than control individuals. This is the starting point of LD-based mapping methods: where does the set of affected chromosomes show linkage disequilibrium? The problem is far from trivial, however. The recombination history is stochastic; mutation carriers often only have a higher risk of being diseased than non-carriers, and in a case-control study both groups are usually mixes of carriers and non-carriers; finally, there is missing information.

Summary of Background and Problem Statement Genetic markers provide an economical, sparse view of chromosomes. Even sparsely located markers can be very informative: given an ancestor with a mutated gene, the descendants that inherit the gene are also likely to inherit alleles of nearby markers. The exact probability of inheriting any combination of markers depends on the gene location with respect to the markers, the population history or the recombination history, and marker mutations; all of these are unknown.

Our gene mapping framework belongs to the family of association/LD-based mapping methods. It is a case-control setting, where the input consists of haplotypes. Each individual contributes a chromosome pair, so the number of chromosomes is twice the number of individuals. We ignore the fact that chromosomes come in pairs and simply consider the input data as consisting of a set of disease-associated haplotypes (from the cases) and a set of control haplotypes.

The LD-based gene mapping problem is now the following. The input consists of a marker map, and a set of disease-associated haplotypes and a set of control haplotypes on the given map. The task is to predict the location of a disease susceptibility gene on the map.

3. METHOD

3.1 Haplotype Trees

TreeDT is based on estimating and evaluating the genealogy of the observed haplotypes. For any pair of chromosomes in the sample there has been a common origin in the population history, an ancestral chromosome at which their paths have diverged. Due to recombinations different parts of chromosomes have different histories, and at any given location the chromosomes in the sample and their most recent common origins form a genealogical tree for that location. In the genealogical tree for the DS gene location, all the chromosomes in one or more subtrees carry the DS mutation, and we should observe excess of disease-associated haplotypes in these subtrees. Looking at genealogical trees for various locations, the closer the location is to the DS gene the more and larger subtrees are identical to those in the tree at the DS gene location.

Given a location in the chromosome – a potential gene locus – the haplotypes to the right of the location can be organized into a tree (Figures 3 and 4), as can haplotypes to the left. In stringology, trees like these are called tries. TreeDT uses the right and left trees as two different estimates of the genealogical tree at the location.

TreeDT builds these two trees between each pair of consecutive markers. It then assesses the subtree clustering of disease-associated haplotypes in these trees. For this task we introduce a novel tree disequilibrium test, intended for predicting DS gene locations. The vicinity of the location for which the test gives the lowest *p* value is the most likely candidate area for the DS gene location. The method also computes the corrected overall *p* value for the best finding. It can be used for predicting whether the chromosome carries a DS gene at all or not.

3.2 Tree Disequilibrium Test

The tree disequilibrium test for a haplotype tree *T* tests the alternative hypothesis *The distribution of the disease-association statuses deviates in some subtrees of T from the overall distribution of statuses* against the null hypothesis *The disease-association statuses are randomly distributed in the*

leaves of T. TreeDT identifies the set *S* of subtrees in which the observed status distribution departs most from the expectation under the null hypothesis. In the next subsection we discuss how to estimate the significance of the departure and how to use it in gene mapping.

For measuring the disequilibrium of a tree, we use a variant of the *Z* test. The test statistic Z_k for a set *S* of *k* deviant subtrees $T_1,\ldots,T_k$ is

$$Z_k(S) = \sum_{i=1}^{k} A_i, \text{ where } A_i = \frac{a_i - n_i p}{\sqrt{n_i p(1-p)}} \text{ if } n_i > 1, \text{ or } A_i = 0 \text{ if } n_i = 1, \quad (1)$$

where a_i is the number of disease-associated haplotypes and n_i the total number of haplotypes in subtree $T_i \in S$, and *p* is the proportion of disease-associated haplotypes in the sample. The score measures the distance of the observed number of disease-associated chromosomes (a_i) from the expectation ($n_i p$) in standard deviations ($\sqrt{n_i p(1-p)}$), under the assumption of binomial distribution with parameters n_i and *p*. Subtrees consisting of a single leaf do not contribute to the test statistic, since it is only possible to extract localization information from two or more haplotypes sharing a region. We use a one-tailed test, since we are interested only in subtrees in which the proportion of disease-associated haplotypes is greater than expected.

For any given tree *T*, we are interested in the maximum value of Z_k over all sets of *k* non-overlapping subtrees of *T*, denoted by $ZMAX_k(T)$:

$$ZMAX_k(T) = \max \{Z_k(S) \mid S \text{ is a set of } k \text{ non-overlapping subtrees of } T\}. \quad (2)$$

One usually considers all different values for *k*, possibly limited by some maximum value k_{max}.

Ideally, we would use the χ^2-statistic from a $2\times(k+1)$ contingency table instead of Z_k. The χ^2-statistic, however, is not easily maximized in the space of all possible subtree sets and is therefore not a very practical choice. Z_k can be efficiently maximized simultaneously for all *k* using a recursive algorithm, as shown in the Algorithms section. Efficiency of the maximization procedure is important, because it is performed many, usually millions of times during the execution of TreeDT. Therefore we use Z_k as a practical approximation of χ^2.

3.3 Significance Tests

After the maximization of Z_k values for all *k* we have a number of $ZMAX_k$ values together representing the disequilibrium of a tree corresponding to a certain location in the chromosome.

Next we want to compress this information into a single value. Since the statistics for different degrees of freedom *k* are not comparable, simple maximization of the $ZMAX_k$ over all possible values of *k* is not justified. Therefore TreeDT estimates the *p* value for each *k* under the null hypothesis of random distribution of disease statuses, and the lowest of these *p* values is used as a measure of disequilibrium of the tree. Because the distribution of $ZMAX_k$ is very complex and dependent on the tree structure, *p* values are estimated by a permutation test.

Next, in order to get a single *p* value representing the disequilibrium at a given location, we need to combine the information from the trees to the left and to the right of the location. As a combined measure we use the product of the lowest *p* values over all *k* from each side. Again, since the measures are not necessarily directly comparable, a new *p* value for the combination is estimated using a permutation test. We are aware that the maximization of the test statistic introduces a selection bias favoring smaller subtrees, but at this point we have not taken any actions to compensate for it.

The output of TreeDT is essentially a list of *p* values for the tested locations. A point prediction for the gene location is obtained by taking the best location; a (potentially fragmented) region of length *l* is obtained by taking best locations until a length of *l* is covered. A direct link between the *p* value and the probability that the gene is indeed close to the location cannot be established. The *p* values are used simply as a method of ranking the locations.

However, a single corrected *p* value for the best finding can be obtained with a third significance test using the lowest local *p* value as the test statistic. The null hypothesis in this case is *At all locations the disease statuses are randomly distributed in the leaves of the trees*. The resulting *p* value can be used to answer the question whether there is a gene in the investigated area in the first place or not: the null hypothesis is rejected, if the resulting *p* value is lower than a predetermined significance level. All the three nested *p* value tests (for each tree and *k*, for each location, for the best location) can be carried out efficiently at the cost of a single test, as shown in the Algorithms section.

4. ALGORITHMS

The *TreeDT* algorithm can be decomposed into two subtasks: construction of the left and right-side haplotype trees for each location and carrying out the permutation procedure. The most time critical

part of the permutation procedure is the computation of the *ZMAX* statistics. These tasks are discussed in detail in the following subsections.

4.1 Constructing Haplotype Trees

The haplotype trees to the left and right from each analyzed location can be efficiently identified using the textbook radix sort algorithm (see eg. [6]). The algorithm produces as intermediate results for each marker a sorted list of the partial haplotypes to the right from the marker. All the right-side trees can be easily derived from these intermediate lists, because the haplotypes belonging to a single node form a continuous block in the sorted list. The left-side trees can be identified similarly by sorting the inverted haplotypes. The computational cost of constructing all the trees is linear both in the number of markers and the number of haplotypes, and it is negligible compared to the cost of the permutation test procedure.

4.2 An Algorithm for Maximizing the Test Statistic

It is essential that the time-complexity of the algorithm for maximizing the *Z* values is as low as possible, because it must be executed for each tree location and permutation in turn. An efficient recursive algorithm, MaximizeZ, which propagates the locally maximized *Z* values upwards in the tree is presented below. The algorithm is based on the recursive definition of *ZMAX*:

1. $ZMAX_1(T) = \max \{Z_1(T)\} \cup \{ZMAX_1(T') \mid T' \text{ is a proper immediate subtree of } T\}$
2. $ZMAX_k(T) = \max \{ \sum_{i \in U} ZMAX_{k_i}(T_i) \mid \sum_{i \in U} k_i = k \text{ and } U \subseteq \{1, \ldots, m\} \text{ and } \{T_1, \ldots, T_m\} \text{ is the set of proper immediate subtrees of } T\}$, where $k > 1$.

The notion *ZMAX* is generalized to apply for forests as well in the obvious way: $ZMAX_k(F)$ is the maximum value of $Z_k(S)$ where *S* is a set of *k* non-overlapping subtrees of forest *F*.

Algorithm *MaximizeZ*(*T*):

Input: Haplotype tree *T*

Output: $ZMAX_k(T)$ for all *k*, $1 \le k \le n$, where *n* is the number of leaves in *T*

Method:

If *T* is a leaf, then set $ZMAX_1(T) := 0$

Otherwise:

1. // For each k: calculate the maximum value $ZMAX_k(T)$ for Z_k over all S that can be obtained by
   // combining subtree sets from each subtree T_i of T.

1.1 $F := \{\}$.

1.2 For each immediate proper subtree T' of T:

1.2.1 Recursively call $MaximizeZ(T')$.

1.2.2 For each k, $1 \le k \le a$, where a is the total number of leaves in forest F:

$ZMAX_k(F \cup \{T'\}) := ZMAX_k(F)$.

1.2.3 For each k, $a < k \le a + b$, where b is the number of leaves in T':

$ZMAX_k(F \cup \{T'\}) := 0$.

1.2.4 For each pair (i,j), $1 \le i \le b$ and $1 \le j \le a$:

If $ZMAX_i(T') + ZMAX_j(F) > ZMAX_{i+j}(F \cup \{T'\})$,

then $ZMAX_{i+j}(F \cup \{T'\}) := ZMAX_i(T') + ZMAX_j(F)$.

1.2.5 $F := F \cup \{T'\}$.

1.2.6 For each k, $1 \le k \le b$:

If $ZMAX_k(T') > ZMAX_k(F)$, then $ZMAX_k(F) := ZMAX_k(T')$.

2. For each k, $1 \le k \le n$, where n is the total number of leaves in T: $ZMAX_k(T) := ZMAX_k(F)$.

3. Calculate Z_1 for T (Eq. 1). If $Z_1 > ZMAX_1(T)$ then $ZMAX_1(T) := Z_1$.

The time complexity of the algorithm is $O(n^2)$, both on the average and in the worst case, where n is the number of leaves in the tree, i.e. the number of haplotypes in the data set. By setting an upper limit k_{max} for the size of the subtree sets, the time complexity can be reduced to $O(n)$ with a constant coefficient proportional to k_{max}^2, k_{max} being typically small, ≤ 10. The only modification required in the algorithm is an additional condition $\le k_{max}$ for index k and sum $i + j$ in steps 1.2.2 – 1.2.6. In principle there is no need to set an upper limit – the number of chromosomes is the maximum number of subtrees – but whenever LD-mapping is applicable, the majority of mutation carriers is concentrated in only few subtrees of the haplotype trees at the DS gene locus, and using this prior information to restrict the number of subtrees may slightly increase the power of the method, as shown in the Experiments section. In the experiments for this paper we use an upper limit of 3 subtrees.

The space complexity is $O(n)$ if the number of subtrees is not restricted. The average complexity reduces to $O(\log n)$ for the restricted case. Proofs for the correctness and time and space complexities of the algorithm are given in the appendix.

4.3 An Efficient Algorithm for Multiple Nested Permutation Tests

The straightforward algorithm for a three-level nested permutation test using nested loops would have time complexity proportional to q^3, where q is the number of permutations at each level. The test would be intractable already with rather low permutation counts. However, the time complexity can be drastically reduced using the same set of permutations at each level of the test and thus only maximizing the Z values q instead of q^3 times for each location.

Algorithm *NestedPermutationTests*:

Input: Set of tested locations, pair of opposed haplotype trees for each location, number of permutations q

Output: Local p value $p\_local(l)$ for each tested location l, overall corrected p value *p_corrected*

// Level 1

1. For each haplotype tree T:

1.1 Call $MaximizeZ(T)$, which returns $ZMAX_k(T)$ for each number of subtrees k.

1.2 For i in $1 \ldots q$:

1.2.1 Generate a tree $P(T,i)$ from T by permuting the disease-association statuses of the haplotypes. The permutation is some function of i.

1.2.2 Call $MaximizeZ(P(T,i))$, which returns $ZMAX_k(P(T,i))$ for each number of subtrees k.

1.3 For each number of subtrees k:

1.3.1 Calculate a p value $p_k(T)$ by comparing $ZMAX_k(T)$ to $ZMAX_k(P(T,i))$ for all i (see text below for the computation of p values).

1.3.2 For each permutation i: calculate a p value $p_k(P(T,i))$ by comparing $ZMAX_k(P(T,i))$ to all $ZMAX_k(P(T,j))$, $j \neq i$ and $ZMAX_k(T)$.

1.4 $pmin(T) := \min \{p_k(T)\}$ over all k.

1.5 For each permutation i: $pmin(P(T,i)) := \min \{p_k(P(T,i))\}$ over all k.

// Level 2

2. For each tested location l and the pair (T_1,T_2) of opposed haplotype trees located at l:

2.1 Calculate a p value $p\_local(l)$ by comparing product $pmin(T_1)pmin(T_2)$ to $pmin(P(T_1,i))pmin(P(T_2,i))$ for all i.

2.2 For each permutation i: calculate a p value $p\_local(i,l)$ by comparing product $pmin(P(T_1,i))pmin(P(T_2,i))$ to all $pmin(P(T_1,j))pmin(P(T_2,i))$, $j \neq i$, and $pmin(T_1)pmin(T_2)$.

// Level 3

3. $pmin$:= min $\{p\_local(l)\}$ over all l.
4. For each permutation i: $pmin(i)$:= min $\{p\_local(i,l)\}$ over all l.
5. Calculate the overall corrected p value $p\_corrected$ by comparing $pmin$ to $pmin(i)$ for all i.

The p value from a permutation test is determined by comparing the observed value of the test statistic to the distribution obtained from the permutations. p is the proportion of permutations for which the test statistic has at least as extreme value as the observed value:

$$p \approx \frac{\left|\left\{s_i \geq s_0 \middle| 1 \leq i \leq q\right\}\right|}{q},$$

where s_0 is the observed statistic, s_i is the same statistic from the ith permutation, and q is the number of permutations. If the test statistic is being minimized instead of maximized, as is the case with p values as tests statistics, then “less than” should be used instead of “greater than” .

Due to the finite number of permutations, the precision of the p values given by a permutation test may not be sufficient for accurate localization. In some situations even a very large number of permutations does not produce any values for the test statistic as extreme as the observed values, for several consecutive locations. For this reason the p values returned by the first and second level permutation tests are determined slightly unconventionally. In Steps 1.3.1 and 1.3.2, the returned p value is interpolated between the p values corresponding to the next lower and higher values for the test statistic obtained by permutations. If the observed score s_o at level 1 is higher than the highest score s_p obtained from the permutations, the p value is extrapolated using the ratio of the two scores, $p = \frac{s_p}{qs_o} \leq \frac{1}{q}$. At level 2, in Steps 2.1 and 2.2, if the observed score is lower than the lowest obtained from the permutations, a lower boundary value of zero is used to interpolate a value.

Finally, the top-level test returning the overall p value is implemented in the usual conservative manner.

The time complexity of steps 1.3.2 and 2.2 is $O(q \log q)$ using an algorithm which first sorts the values of the test statistic for all the permutations. Step 1.2.2 predominates the time complexity of the algorithm, $O(qn^2s)$, where n^2 is the time complexity of MaximizeZ algorithm, and s is the number of markers (or tested locations) in the chromosome.

4.4 Algorithm *TreeDT*

As a summary, we give an informal description of the TreeDT algorithm. As input it takes a marker map and a set of disease-associated and control haplotypes. The parameters for the algorithm are the number of permutations and optionally the maximum or exact number of subtrees to be tested. Its output consists of local p values for all tested locations and an overall corrected p value, as returned from the NestedPermutationTests algorithm. The set of locations to be tested is the set of intervals between adjacent markers.

The haplotypes are sorted using radix sort algorithm. After each iteration of radix sort, level 1 of the permutation procedure is performed for the right-side haplotype tree implicit in the intermediate sorting result. One only needs to store the smallest p value over all numbers of subtrees for each haplotype tree and each permutation. In a second pass the same is done for the inverted haplotypes and the left-side haplotypes. All there is left after this point is straightforward execution of levels 2 and 3 of the permutation algorithm.

5. RELATED WORK

Most current gene mapping methods based on linkage disequilibrium look just at individual markers or neighboring markers, measure their association to the disease status, and predict the gene locus to be co-located with the strongest association. However, since different mutation carriers share different segments, there is no single marker or pattern that is representative of the shared segments.

In the recent years, several statistical methods have been proposed to detect LD [5][9][11][13][18]. The emphasis has been on fairly involved statistical models of LD around a DS gene. They model whole recombination histories and some are robust to high levels of heterogeneity. On the other hand, the models are based on a number of assumptions about the inheritance model of the disease

and the structure of the population, which may be misleading for the statistical inference. The methods tend to be computationally heavy and therefore better suited for fine mapping than genome scanning.

Haplotype Pattern Mining or HPM [19] is based on analyzing the LD of sets of haplotype patterns, essentially strings with wildcard characters. The method first finds all haplotype patterns that are strongly associated with the disease status, using ideas similar to the discovery of association rules [1][2]. Since the patterns may contain gaps they can account for some missing and erroneous data. In the second step, each marker is ranked by the number of patterns that contain it. Either this score is used as a basis for the prediction or, preferably, a permutation test is used to obtain marker-wise *p* values. HPM has been extended for detecting multiple genes simultaneously [20] and to handle quantitative phenotypes and covariates [14].

Nakaya et al investigate the effect of multiple separate markers, each one thought to correspond to one gene, on quantitative phenotypes [12]. Their work is a generalization of the LOD score to multiple loci, and it does not handle haplotype patterns.

An alternative approach for LD-based mapping is linkage analysis. The idea is to analyze family trees, and to find out which markers tend to be inherited to offspring in conjunction with the disease. Linkage analysis does not rely on common founders, so in that respect it is more widely applicable than LD-based methods. The downside is that estimates are rough (due to the smaller effective number of meioses sampled), and that collecting information from larger families is more difficult and expensive.

Transmission/disequilibrium tests (TDT) [16] are an established way of testing association and linkage in a sample where linkage disequilibrium exists between the mutation locus and nearby marker loci. TDT detects deviations between observed and expected counts for each allele, or, in its multipoint variant, haplotype of several alleles, transmitted from heterozygous parents to affected offspring. We performed TDT analysis using GENEHUNTER2 software package [7] for our experiments in the next section.

Single permutation tests have been used in mapping studies before [4][8][10], but we are not aware of any multiple permutation tests similar to the ones presented in this paper.

6. EXPERIMENTS

We compare TreeDT empirically to TDT, to multipoint TDT (m-TDT) using haplotypes of up to four markers, and to HPM, our recent proposal based on pattern discovery. We evaluate the methods on difficult data collections carefully simulated to resemble a realistic population isolate.

6.1 Data Sets

Simulated Data We designed several different test settings, with variation in the fraction (A) of mutation carriers in the disease-associated chromosomes, in the number of founders who introduced the mutation to the population, in the amount of missing information, and in the sample size. For statistical analyses, we created 100 independent artificial data sets in each test setting. Great care was taken to generate realistic data by a simulation procedure that included four steps: pedigree generation, simulation of inheritance, diagnosing, and sampling.

The population pedigree was set to grow exponentially from 100 to 100,000 individuals in a period of 20 generations. In each generation, the individuals were first randomly coupled. Then, for each child, a couple was randomly selected for the parents. The total number N of children in each generation was fixed, thus the number of children for each couple followed the binomial distribution with parameters N and 1 / (number of couples).

The inheritance of chromosomes within the population pedigree was simulated by first allocating a continuous chromosomal segment of 100 cM[1] to each founder individual in generation 0. Next, the entire pedigree was traversed top-down, and, in each inheritance event, gametes were created by simulating meioses under the assumption that the number of chiasmata in the pair of homologous chromosomes was drawn from the Poisson distribution with parameter one (corresponding to the genetic length of 100 cM), and their locations selected randomly. A related approach was originally presented in [17].

The location of the mutation was selected randomly for each of the 100 data sets produced, but the sources (founder chromosomes) of the mutation were selected each time specifically to make the

[1] Morgan is a unit of genetic length. 1 cM is the distance at which crossover is expected to occur once in 100 meioses, on average about 10^6 base pairs. Human chromosomes are about 50–300 cM.

frequency of the mutation in the final generation as close to 2% as possible. In the baseline setting there is one mutation-carrying founder chromosome.

The risk for becoming affected was set in such a way that the expected proportion of mutation-carrying chromosomes among the affected in the final generation was equal to a parameter *A*. For a baseline test setting we selected a challenging disease model where only a small proportion (*A*=10%) of the disease-associated chromosomes carries the disease-predisposing mutation, a complication that often is encountered in the analysis of common diseases.

Next, 100 random samples of affected individuals (100 individuals in each sample in the baseline setting) and their parents were selected. Since the number of mutation-carriers was fixed only at the level of generation – rather than a single sample – considerable variance remains in the number of mutation-carriers between samples, which makes localization more challenging than for data sets presented in [19]. For all members in the sample, allelic data were created using a map of 101 equidistantly spaced markers, each having 5 alleles, one of which had a frequency of 0.4 whereas the the four other alleles had equal frequencies of 0.15.

In real-life the haplotypes of an individual cannot be read directly. For each marker only its *genotype*, the pair of alleles without knowledge about their parental origins, is observed. To construct the haplotypes in a realistic way we also need to utilize the genotype information from the parents. By looking at the genotypes of the offspring-parents-trio we can infer which alleles have been transmitted to the offspring. The haplotypes constructed from the alleles transmitted to each affected individual in a sample were labeled disease-associated, whereas the haplotypes constructed from the non-transmitted alleles of the parents can be conveniently used as controls. However, in a rare case of three similar heterozygote genotypes at a marker (both parents and the offspring), determining the parental origin of the inherited alleles is impossible. We chose to take this into account by setting the alleles unknown in these cases. Thus, 3.7% of alleles are missing, making the mapping task more difficult, but also more realistic.

In one of the experiments, we randomly removed 5% of the alleles in the genotype data, adding further complexity to the determination of parental origin. Consequently, about 15% of the allele information in the resulting haplotypes is missing.

The final output of this simulation procedure is a collection of 100 data sets (with a mutation proportion of *A* among the affected chromosomes), consisting of 200 disease-associated and 200 control chromosomes in the baseline setting, each containing the alleles of 101 adjacent markers.

Real Type 1 Diabetes Data In order to test and demonstrate the real-life performance of TreeDT, we analysed a Type 1 Diabetes dataset [3]. We have used exactly the same dataset in the article introducing HPM [19]. The study subjects were genotyped for 25 markers covering 14Mb including the HLA-region. There is a known major susceptibility locus located in the center of the marker map. The data consists of 385 affected sib-pairs and their parents, out of which 100 randomly chosen families were used. One child was randomly selected from each family. The non-transmitted parental haplotypes were used as controls, and the transmitted as cases.

6.2 Analysis of TreeDT

First we assess the prediction accuracy of TreeDT with different values of *A*, the proportion of disease-associated chromosomes that actually carry the mutation (Figure 5A). The results are reported as curves that show the percentage of 100 data sets where the gene is within the predicted region, as a function of the length of the predicted region. Or, in other words, the *x* coordinate tells the cost a geneticist is willing to pay, in terms of the length of the region to be further analyzed, and the *y* coordinate gives the probability that the gene is within the region. For *A*=20% or 15% the accuracy is very good, and with lower values of *A* the accuracy decreases until with *A*=5% only in 20-30% of data sets can the gene be localized within a reasonable accuracy of 10–20 cM. We remind the reader that the test settings have been designed to be challenging, and to test the limits of the approach.

Next we evaluate the effect of the only parameter of TreeDT, the number of deviant subtrees that are searched for in each tree. An upper limit of 6 subtrees, used in the previous test, is evaluated against fixed amounts of 1, 2 or 3 subtrees, with a varying number of founders that introduced the mutation (Figure 5B). As we increase the number of founders, evidence about the gene location becomes more fragmented, and accordingly the performance degrades. While the differences between different numbers of subtrees are not large, it is interesting to note that for each number of founders, the same number of subtrees gives marginally the best result. The upper limit of 6 subtrees gives consistently competitive results, so we continue using it in the following experiments.

As expected, the localization power increases with the sample size, although increasing the sample sizes from 800 to 1,200 chromosomes does not improve the results significantly any more (Figure 5C). We also evaluated the effect of the maximum number of subtrees in the baseline setting, where

there was a single mutation-carrying founder. Decreasing the maximum number does increase the power, as shown in Figure 5D, because the number of tests on the lowest level decreases with it. However, tests with different numbers of subtrees are highly correlated, and the differences in power are rather small. Furthermore, the number of mutation-carrying founders for a real dataset is usually unknown, so it might not be wise to set too low a limit.

Gene mapping studies like the ones imitated in the above tests assume, based on some other analyses, that a disease susceptibility gene is indeed present in the analyzed area. TreeDT has the important advantage over plain gene localization methods that it can also be used to predict whether the analyzed region contains a disease susceptibility gene at all or not. The overall *p* value TreeDT produces indicates the corrected significance of the best single finding, and by setting an upper limit for its value TreeDT can be used to classify data sets to ones that do or do not contain a mutation. In order to verify the correctness of the permutation procedure, we generated 100 data sets where the disease-association statuses were randomly chosen for each individual, that is, there is no genetic contribution from the simulated chromosome. For these data sets, TreeDT should produce overall *p* values as well as local *p* values from the uniform distribution in [0,1]. Figure 6A shows the cumulative distribution of the observed overall *p* values on these data sets; for only 100 data points the deviation from the diagonal is within expectations. The local *p* values follow the uniform distribution very convincingly (Figure 6B).

Smaller thresholds for the overall *p* value result in less false positives, but also in less true positives. Figure 6C shows the experimental relationship, in the form of a ROC curve, between power (ratio true positives/all positives) and overall *p* (ratio false positives/all negatives). For higher values of *A* the classification accuracy is extremely good. However, for *A*=5% the classification accuracy is no better than random guessing, although the localization accuracy for an existing gene is still adequate in 20–30% of the cases (Figure 5A).

6.3 Comparison to other methods

TreeDT, HPM, and m-TDT have practically identical performance in localizing the DS gene in the baseline setting (Figure 7A). The parameters we used for HPM were χ^2 threshold 7, maximum pattern length 7 and one gap of at most one marker. TDT is clearly inferior compared to the other methods. Tests with other values of *A* or other sample sizes give similar results (not shown).

In a test setting with three founders who introduced the mutation to the population, differences between the three best methods start to appear (Figure 7B). TreeDT has an edge over HPM, which in turn has an edge over m-TDT. TDT barely beats random guessing.

Finally, we compare the methods with a large amount of missing data (Figure 7C). Expectedly, HPM is most robust with respect to missing data since it allows gaps in its haplotype patterns. Surprisingly, TreeDT is not much weaker than HPM, although no actions have been taken in it to account for missing or erroneous data. Using a simple method for imputing values for the missing alleles improved the results of TreeDT to be on par with HPM (results not shown). Performance of m-TDT degrades much more clearly.

On the real Type 1 diabetes data, TreeDT pinpoints the known DS locus very convincingly using 10,000 permutations (Figure 7D). HPM with the same number of permutations is able find the locus as well, but local *p* values given by TreeDT are much smaller, and the extrapolation mechanism further highlights the predicted location. None of the permutations gave at least as small lowest local p value as the lowest for the observed data. The overall corrected *p* value is thus $<10^{-5}$.

Method to method comparisons of prediction (not shown) indicate that errors are mostly caused by random effects in population history – since different methods tend to make mistakes in the same data sets – rather than by systematic differences between the methods. However, those cases where one method succeeds and another fails will give useful input for further improvements of the methods.

The execution time of TreeDT for a simulated data set with 400 haplotypes is about one minute using 1,000 permutations on a 1400 MHz Pentium 4. The respective time for HPM with permutations is over 4 minutes.

7. DISCUSSION

We have introduced TreeDT, a novel method for gene mapping. It is based on detecting linkage disequilibrium in the haplotype trees to the right and left of the disease susceptibility gene location. We have shown how tree disequilibrium can be efficiently evaluated between every pair of consecutive markers, and be subsequently tested for statistical significance using multiple, nested permutation tests with the cost of one. Empirical evaluation on a realistic, simulated data shows that the method is competitive with other recent data mining based methods, and clearly outperforms more traditional methods.

Nowadays the focus of gene mapping methodology is on complex diseases, where there are several genes and possibly environmental factors contributing to susceptibility. With complex diseases the associations of individual genes are diluted by the effects of the other factors. Therefore methods looking for individual susceptibility genes must be able to detect rather weak genetic effects. Towards this end, in the simulated data sets used in the experiments only a small fraction (5–20%) of chromosomes carried the mutated allele, creating a mapping challenge similar to that of mapping individual genes for a complex disease. Models including gene–gene and gene–environment interactions may capture the underlying mechanisms of the disease better than methods looking for single gene association. However, typical sample sizes may not warrant the additional complexity of such models.

One of the problems with real haplotype data is that there are missing alleles due to problems in allele calling and ambiguities in determination of the haplotypes. The algorithm as described in this paper regards a missing allele as just another allele symbol, which may result in premature or delayed branching of shared haplotypes in the tree.

Even if there is no missing or erroneous data, the haplotype trees TreeDT uses as estimates may differ from the true genealogical trees in two ways: 1) The order of nodes may differ from that in the true genealogical tree, e.g., in Figure 4, 34--- might actually be a more recent node than 3411-. However, because the expected length of the shared region of two chromosomes decreases monotonically as the time from their divergence increases, it is easy to see that the order given by subsumption is the most likely one. 2) Because haplotypes may also share a substring by chance, the internal nodes of the estimate may represent a combination of nodes in the true genealogical tree. As a result, the mutated subtree in the genealogical tree may be split to a few subtrees in the estimated tree. Or, on the other hand, it may be merged with other subtrees, diluting the observable disequilibrium in the combined subtree. We believe that in most cases the structure of the estimated tree is close enough to that of the genealogical tree to allow for testing the disequilibrium with a reasonable power. This view is supported by the results from our experiments.

Our experiments show that TreeDT is effective in extreme conditions typical for current mapping problems: with lots of noise (only 10–20% of affected chromosomes carry the mutation, lots of missing data) and with small sample sizes (200 affected and 200 control chromosomes). However, the highest potential of the method lies in the data intensive tasks of future – such as genome

scanning with larger samples and larger number of markers – due to its low computational complexity.

In comparison to state of the art methods, TreeDT is most competitive. In terms of gene localization accuracy, it gave best results in the case of multiple founders and demonstrated good robustness with respect to missing data. Unlike the compared methods, TreeDT can be used to predict whether a gene is present at all or not. Finally, in comparison to its closest competitor, HPM, TreeDT has much smaller computational cost. An additional advantage of TreeDT is that it has only one optional input parameter, the (maximum) number of deviant subtrees, whereas for HPM one has to set several parameters.

As with any association or LD–based methods, some limitations inevitably apply to TreeDT as well. 1) A dense enough marker map is needed to be able to observe LD between the disease locus and the nearby markers. Sufficient density depends on the expected length of haplotype sharing in the population under study, but typically at least one marker per cM is used. Simulation experiments imitating the target population, such as those reported in this article, are most useful in determining marker density and sample size needed to acquire desired power to detect susceptibility genes. 2) As a pure case-control association method, TreeDT is vulnerable to the potential false positives caused by population substructure. The problem can be avoided by carefully matching the controls with the cases, e.g. by using the non-transmitted chromosomes of the parents of the cases as control chromosomes. 3) Case–control studies are ineffective if the disease-predisposing mutation is carried in many infrequent haplotypes. Isolated founder populations are invaluable for case–control studies, as recent genetic bottlenecks – periods of slow growth of a small population – have reduced the number of different haplotypes carrying the mutation in current population.

Our future work will address several issues. One is more complex haplotype data: robustness towards missing information, errors, and marker mutations is important with noisy, real-life data sets. A whole set of issues concerns improving tests and models for the tree disequilibrium. Now we combine the left and right trees at a locus without considering how the haplotype strings actually extend over the locus; obviously we miss some information. Another way of improving the model performance is to average the disequilibrium test over all different tree structures. The test statistic itself will be improved to better account for the genetic processes that produce the data.

8. ACKNOWLEDGEMENTS

Thanks to Päivi Onkamo and Juha Kere for support and co-operation during this research. This research has been funded by Helsinki Graduate School in Computer Science and Engineering (HeCSE) and Graduate School in Computational Biology, Bioinformatics, and Biometry (ComBi).

REFERENCES

[1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: P. Buneman and S. Jajodia, (Ed.), Proc. ACM SIGMOD Conference on Management of Data, ACM, Washington, DC, 1993, pp. 207-216.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.Verkamo, Fast discovery of association rules, in: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, (Ed.), Advances in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, 1996, pp. 307-328.

[3] S. Bain, J. Todd, J. Barnett, The British Diabetic Association – Warren repository, Autoimmunity, 7 (1990) 83-85.

[4] G. Churchill, R. Doerge, Empirical threshold values for quantitative trait mapping, Genetics, 138 (1994) 963-971.

[5] B. Devlin, N. Risch, K. Roeder, Disequilibrium mapping: composite likelihood for pairwise disequilibrium, Genomics, 36 (1996) 1-16.

[6] D. Knuth, The art of computer programming, volume III – sorting and searching, Addison-Wesley, Reading, MA, 1975.

[7] L. Kruglyak, M. Daly, M. Reeve-Daly, E. Lander. Parametric and nonparametric linkage analysis: a unified multipoint approach, Am J Hum Genet, 58 (1996) 1347-1363.

[8] T. Laitinen, P. Kauppi, J. Ignatius, T. Ruotsalainen, M. Daly, H. Kääriäinen, L. Kruglyak et al, Genetic control of serum IgE levels and asthma: linkage disequilibrium studies in an isolated population, Hum Mol Genet, 6 (1997) 2069-2076.

[9] L. Lazzeroni, Linkage disequilibrium and gene mapping: an empirical least-squares approach, Am J Hum Genet, 62 (1998) 159-170.

[10] A. Long, C. Langley, The power of association studies to detect the contribution of candidate genetic loci to variation in complex traits, Genome Res, 9 (1999) 720-731.

[11] M. McPeek, A. Strahs, Assessment of linkage disequilibrium by the decay of haplotype sharing, with application to fine-scale genetic mapping, Am J Hum Genet, 65 (1999) 858-875.

[12] A. Nakaya, H. Hishigaki, S. Morishita, Mining the quantitative trait loci associated with oral glucose tolerance in the Oletf rat, in: Proc. of Pacific Symposium on Biocomputing, 2000, pp. 367-379.

[13] S. Service, D. Temple Lang, N. Freimer, L. Sandkuijl, Linkage-disequilibrium mapping of disease genes by reconstruction of ancestral haplotypes in founder populations, Am J Hum Genet, 64 (1999) 1728-1738.

[14] P. Sevon, V. Ollikainen, P. Onkamo, H. Toivonen, H. Mannila, J. Kere, Mining associations between genetic markers, phenotypes and covariates, Genetic Analysis Workshop 12, Genetic Epidemiology, 21 (Suppl. 1) (2001) S588-S593.

[15] P. Sevon, H. Toivonen, V. Ollikainen, TreeDT: gene mapping by tree disequilibrium test, in: F. Provost and R. Srikant (Ed.), Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, CA, 2001, pp. 365-370.

[16] R. Spielman, R. McGinnis, W. Ewens, Transmission test for linkage disequilibrium: the insulin gene region and insulin-dependent diabetes mellitus (IDDM), Am J Hum Genet, 52 (1993) 506-516.

[17] J. Terwilliger, M. Speer, J. Ott, Chromosome-based method for rapid computer simulation in human genetic linkage analysis, Genetic Epidemiology, 10 (1993) 217-224.

[18] J. Terwilliger, A powerful likelihood method for the analysis of linkage disequilibrium between trait loci and one ore more polymorphic marker loci, Am J Hum Genet, 56 (1995) 777-787.

[19] H. Toivonen, P. Onkamo, K. Vasko, V. Ollikainen, P. Sevon, H. Mannila, M. Herr, J. Kere, Data mining applied to linkage disequilibrium mapping, Am J Hum Genet, 67 (2000) 133-145.

[20] H. Toivonen, P. Onkamo, K. Vasko, V. Ollikainen, P. Sevon, H. Mannila, J. Kere, Gene mapping by haplotype pattern mining, in: Proc. IEEE Int'l Symposium on Bio-Informatics and Biomedical Engineering, Arlington, VA, 2000, pp. 99-108.

APPENDIX

A.1 Correctness of *Maximize_Z* algorithm

Lemma 1. The values produced in step 1 for vector $ZMAX(F)$ for any set F of immediate subtrees of some tree T are correct, given that vector $ZMAX(T')$ is correct for each $T' \in F$.

Proof by induction. If $F = \{\}$ the theorem holds trivially as the dimensionality of vector $ZMAX(F)$ is zero. Otherwise, the algorithm has already produced vector $ZMAX(F')$ for a forest F', $F = F' \cup \{T'\}$. Let us assume that theorem 1 holds for F'. The collection of sets of k non-overlapping subtrees of F is the union of those of F' and T' and those sets that can be obtained by combining subtrees from both. Thus, $ZMAX_k(F)$ is $\max\{ZMAX_k(F'), ZMAX_k(T'), \max\{ZMAX_i(F') + ZMAX_j(T') \mid i + j = k \text{ and } 1 \le i, j\}\}$. It is easy to see that steps 1.2.2 – 1.2.6 produce correct value for vector $ZMAX(F)$.

Lemma 2. For any tree T step 1 of the algorithm produces a value for $ZMAX_k(F)$ for all $k \le n$, where F is the set of immediate subtrees of T and n is the total number of leaves in forest F. Proof omitted as trivial.

Theorem 3. The algorithm produces correct values of $ZMAX_k(T)$ for any tree T.

Proof by induction. If T is a leaf, theorem 3 holds trivially. If T is not a leaf then let us assume that theorem 3 holds for each immediate subtree of T. Theorems 1 and 2 state that vector $ZMAX(F)$, where F is the set of immediate subtrees of T, is correctly produced by the algorithm. The space of sets of k non-overlapping subtrees of tree T is the same as that of forest F, added with $\{T\}$ if $k = 1$. Therefore $ZMAX_k(T) = ZMAX_k(F)$ for all $k > 1$ (step 2), and $ZMAX_1(T) = \max\{ZMAX_1(F), Z_1(\{T\})\}$ (steps 2 and 3), which proves the theorem.

A.2 Time complexity of *Maximize_Z* algorithm

Let $t(T)$ be the time required for processing tree T. It can be defined recursively:

$$t(T) = \sum_{i=1}^{m} t(T_i) + A\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} n_i n_j + B\sum_{i=1}^{m} n_i + C_1 \text{, if } T \text{ is not a leaf, and}$$

$$t(T) = C_2 \text{, if } T \text{ is a leaf,}$$

where $T_1, \ldots, T_m$ are the immediate subtrees of T, $m \geq 2$, $n_i \geq 1$ is the number of leaves in subtree i, and $n = \sum n_i$, and A, B, C_1 and C_2 are implementation dependent constants.

Lemma 4. $\sum_i x_i^2 \leq \left(\sum_i x_i \right)^2$, if $x_i \geq 0$ for all i. Proof omitted as trivial.

Corollary 5. Utilizing lemma 4 and the facts that $m \geq 2$ and $n_i \geq 1$ for all i, we infer

$$\sum_{i=1}^{m} n_i^2 = 2n - m + \sum_{i=1}^{m} (n_i - 1)^2 \leq 2n - m + (n-m)^2 = n^2 - (2n-m)(m-1) \leq n^2 - n.$$

Theorem 6. For any tree T holds $t(T) \leq \left(\frac{A}{2} + B \right) n^2 + C(2n-1)$, where $C = \max\{C_1, C_2\}$.

Proof by induction. If T is a leaf, the theorem holds trivially. Otherwise, let us assume that the theorem holds for each immediate subtree T_i of T. Then,

$$t(T) \leq \sum_{i=1}^{m} \left(\left(\frac{A}{2} + B \right) n_i^2 + C(2n_i - 1) \right) + A \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_i n_j + Bn + C_1.$$

Regrouping the terms and replacing C_1 by C yields

$$t(T) \leq \frac{A}{2} \left(\sum_{i=1}^{m} n_i^2 + 2 \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_i n_j \right) + B \left(\sum_{i=1}^{m} n_i^2 + n \right) + C \left(\sum_{i=1}^{m} (2n_i - 1) + 1 \right).$$

By reduction and applying corollary 5 to the middle group we get

$$t(T) \leq \frac{A}{2} n^2 + Bn^2 + C(2n - m + 1).$$

Finally, because $m \geq 2$, we get the desired form

$$t(T) \leq \left(\frac{A}{2} + B \right) n^2 + C(2n-1),$$

which proves the theorem.

Theorem 7. For any tree T holds $t(T) \geq \frac{D}{2} n^2$, where $D = \min\{A, 2C_2\}$.

Proof by induction. If T is a leaf, the theorem holds trivially. Otherwise, let us assume that the theorem holds for each immediate subtree T_i of T. Then,

$$t(T) \geq \sum_i \frac{D}{2} n_i^2 + A \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_i n_j + Bn + C_1 \geq \sum_i \frac{D}{2} n_i^2 + D \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} n_i n_j = \frac{D}{2} n^2,$$

which proves the theorem.

A.3 Space complexity of *Maximize_Z* algorithm

At any node locally, three vectors of dimensionality n or $k_{\max}$ are needed; one for the result of the most recent recursive call for immediate subtree T', one for the forest F of already processed immediate subtrees, and one for the forest $F \cup \{T'\}$.

Memory for the locally stored vectors need not be allocated until after the first recursive call. Let $T_1 \supset \ldots \supset T_d$ be those trees in the recursion stack, for which the local memory has been allocated, and let $n_1 > \ldots > n_d$ be the number of leaves in the trees. The space needed for local vectors in the root nodes of all these trees is $O(\sum_{i=1}^{d} n_i)$. Always processing the largest subtree first guarantees that $n_{i+1} \leq \frac{n_i}{2}$ holds for all i, $1 \leq i < d$, and thus $\sum_{i=1}^{d} n_i \leq 2n_1$.

The depth of the tree is $O(\log n)$ on the average, and $O(n)$ in the worst case. If the number of subtrees is limited, then the space needed for a node is constant, and the space complexity of the algorithm is in the same class with the depth of the tree.

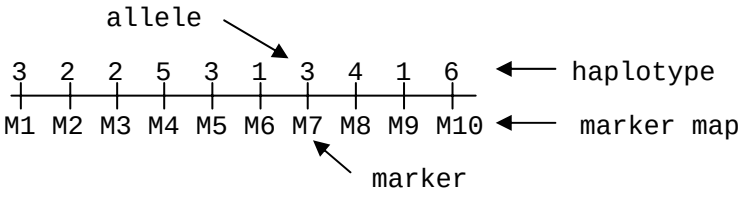

Fig. 1. A marker map of ten markers and a sample haplotype.

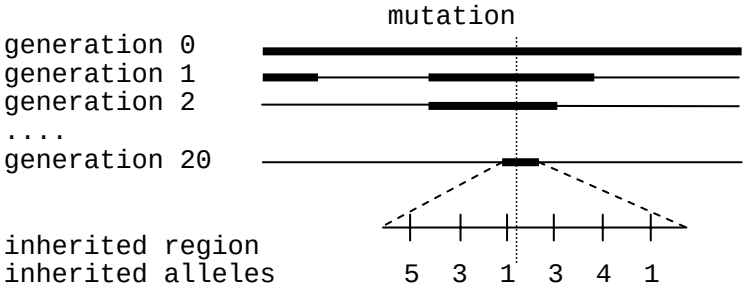

Fig. 2. A carrier of the mutation in generation 20 has inherited alleles from the ancestral chromosome in generation 0 around the gene locus.

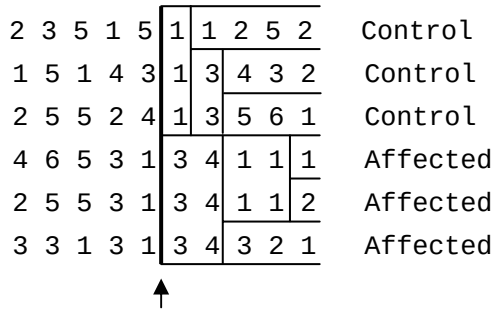

Fig. 3. String-sorted set of haplotypes to the right from the location pointed by the arrow.

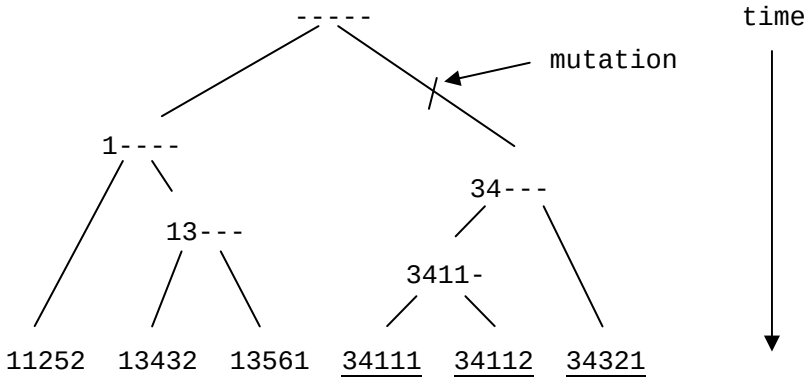

Fig. 4. The haplotype tree, and also a possible genealogical tree, for the haplotypes and the pointed location in Figure 3.

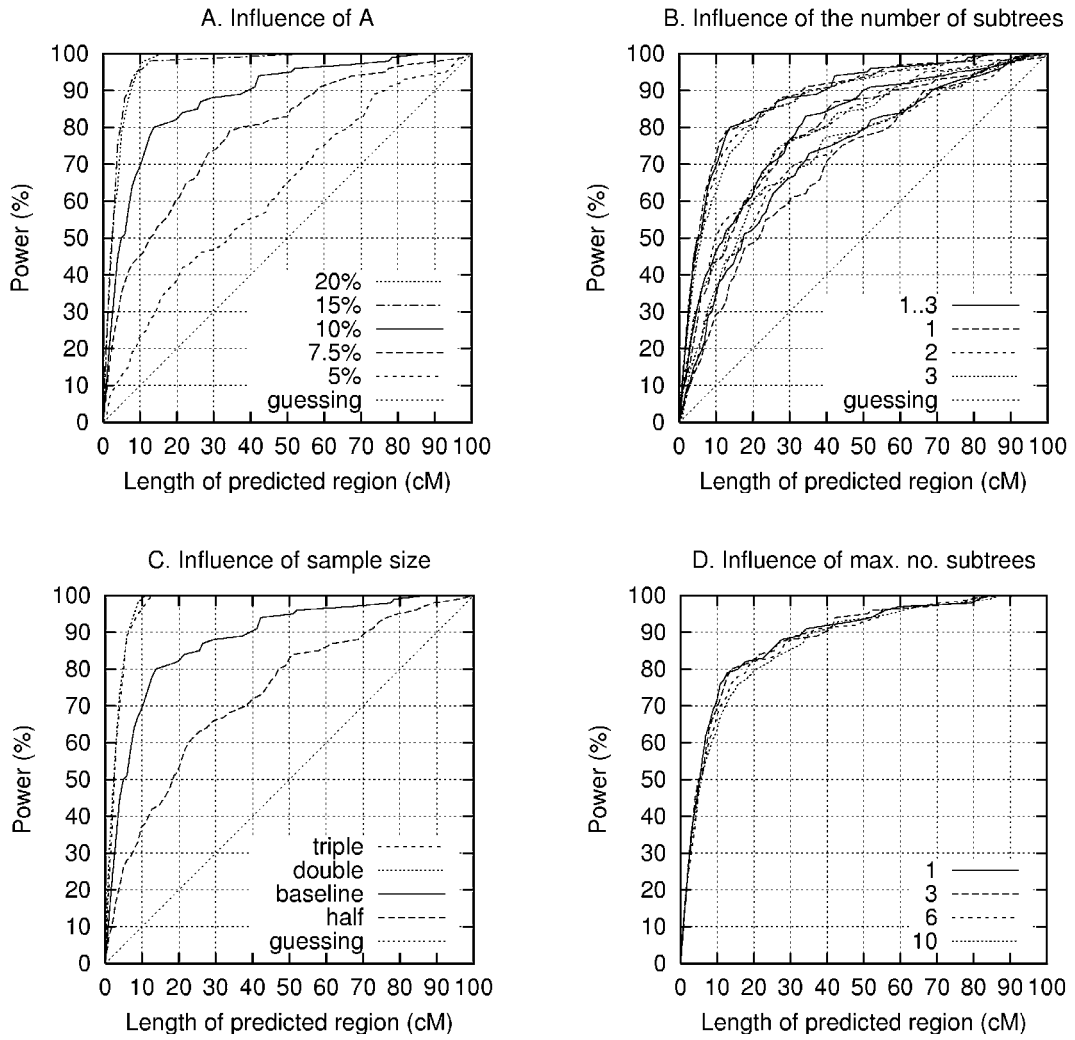

Fig. 5. Analysis of the performance of TreeDT. A: Gene localization power with different values of *A*, the proportion of disease-associated chromosomes that actually carry the mutation. B: Gene localization power with different numbers of subtrees (parameter of the method, given in the legend) and different numbers of founders (population parameter; 1 for the highest set of curves, 2 for the curves in the middle, and 3 for the lowest set of curves). C: Gene localization power with different sample sizes. D: Gene localization power with different maximum numbers of subtrees.

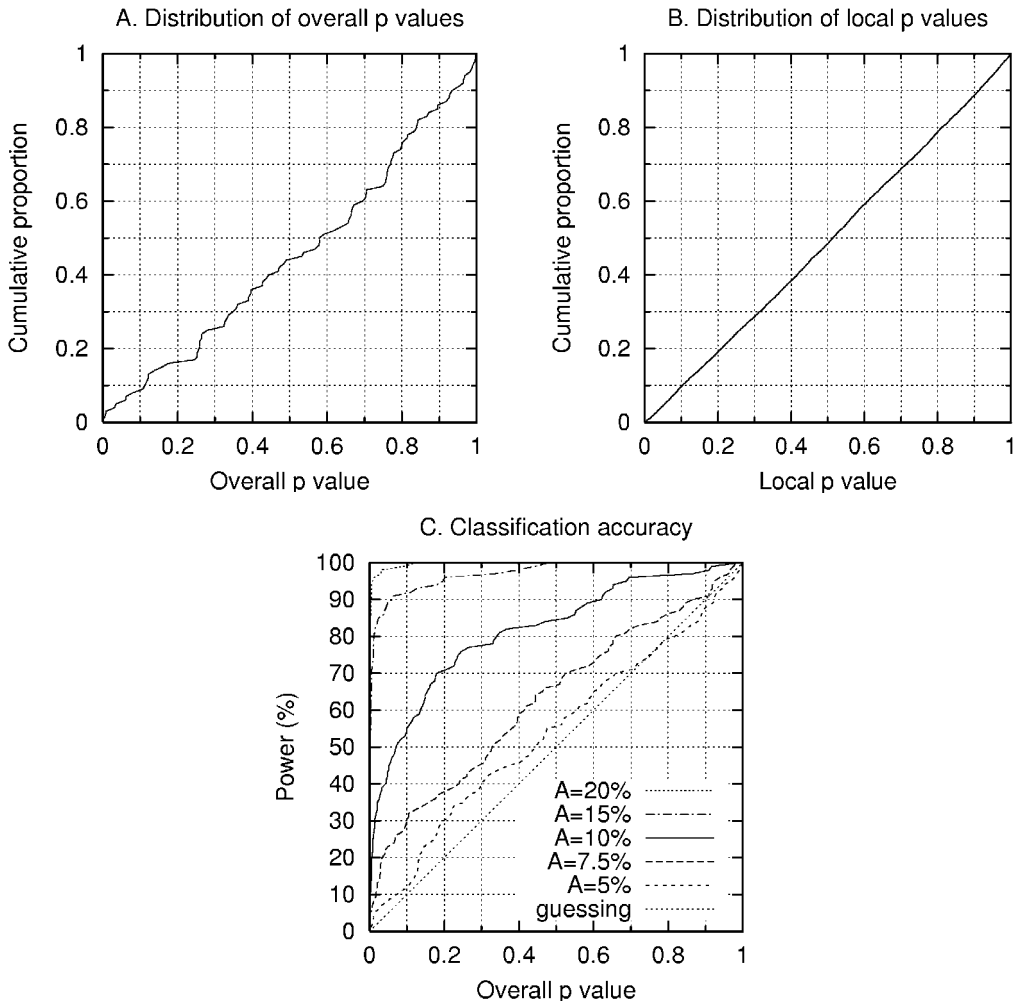

Fig. 6. A: The cumulative distribution of overall *p* values on 100 data sets, in which there were no DS genes. B: The cumulative distribution of local *p* values on the same data (pooled over all the 102 tested locations per data set). C: Classification accuracy for the existence of a disease susceptibility gene.

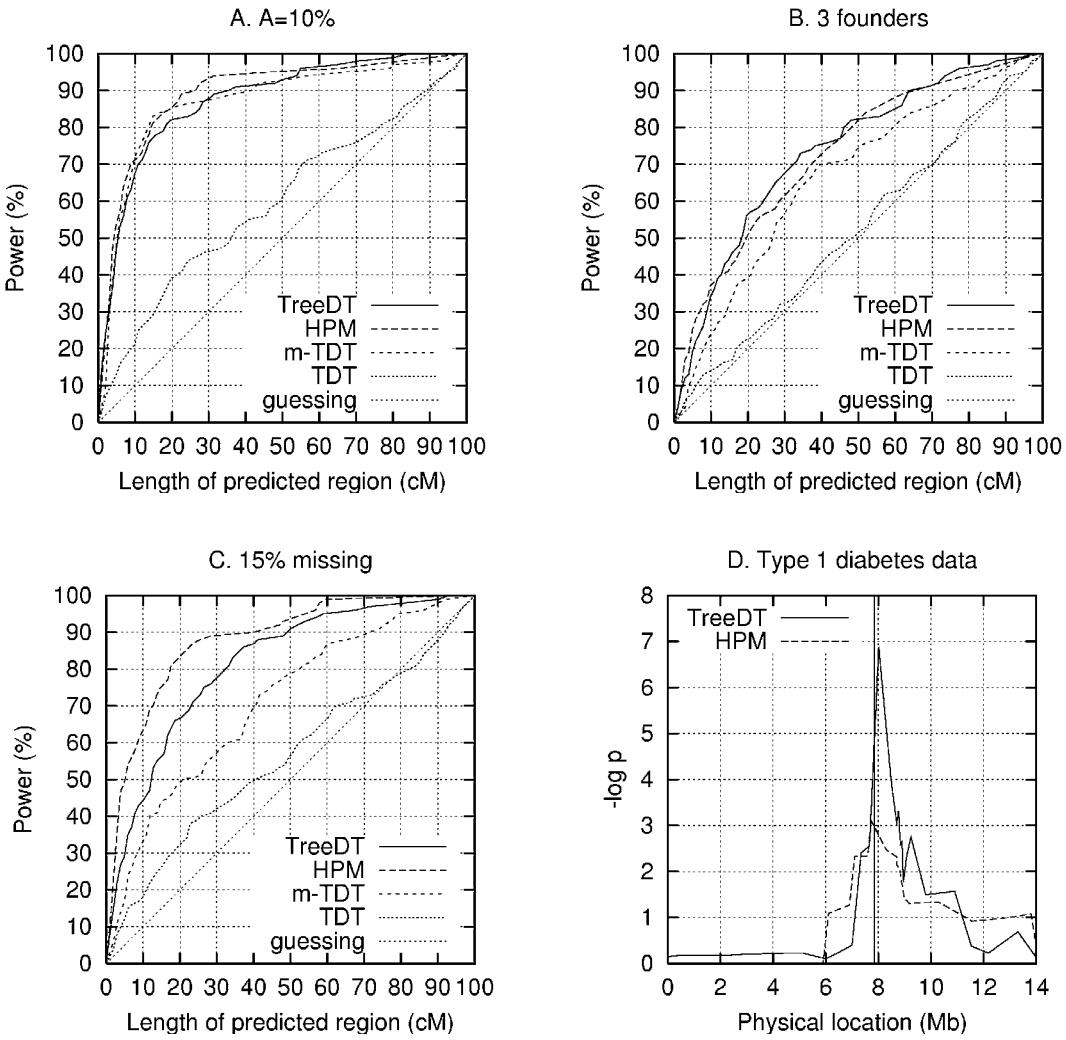

Fig. 7. Comparison of the gene localization performance of TreeDT, HPM, multipoint TDT (m-TDT), and TDT. A: The baseline test setting. B: The baseline setting with three founders. C: The baseline setting with 15% missing data. D: Comparison of TreeDT and HPM on Type 1 diabetes data. The known DS locus is denoted with a vertical line.