Perfect Phylogeny and Haplotype Assignment

Eran Halperin* Richard M. Karp†

September 29, 2003

Abstract

This paper is concerned with the reconstruction of perfect phylogenies from binary character data with missing values, and related problems of inferring complete haplotypes from haplotypes or genotypes with missing data. In cases where the problems considered are NP-hard we assume a *rich data hypothesis* under which they become tractable. Natural probabilistic models are introduced for the generation of character vectors, haplotypes or genotypes with missing data, and it is shown that these models support the rich data hypothesis. The principal results include:

- A near-linear time algorithm for inferring a perfect phylogeny from binary character data (or haplotype data) with missing values, under the rich data hypothesis;
- A quadratic-time algorithm for inferring a perfect phylogeny from genotype data with missing values with high probability, under certain distributional assumptions;
- Demonstration that the problems of maximum-likelihood inference of complete haplotypes from partial haplotypes or partial genotypes can be cast as minimum-entropy disjoint set cover problems;
- In the case where the haplotypes come from a perfect phylogeny, a representation of the set cover problem as minimum-entropy covering of subtrees of a tree by nodes;
- An exact algorithm for minimum-entropy subtree covering, and demonstration that it runs in polynomial time when the subtrees have small diameter;
- Demonstration that a simple greedy approximation algorithm solves the minimum-entropy subtree covering problem with relative error tending to zero when the number of partial haplotypes per complete haplotype is large;
- An asymptotically consistent method of estimating the frequencies of the complete haplotypes in a perfect phylogeny, under an iid model for the distribution of missing data;
- Computational results on real data demonstrating the effectiveness of a the greedy algorithm for inferring haplotypes from genotypes with missing data, even in the absence of a perfect phylogeny.

*CS department, Princeton University, Princeton, NJ 08544. Email: heran@cs.princeton.edu. Most of this work was done while the author was in UC Berkeley and ICSI, Berkeley, CA. The research was partly supported by NSF ITR Grant CCR-0121555.

†International Computer Science Institute, 1947 Center St., Berkeley, CA 94704. Email:karp@icsi.berkeley.edu.

1 Introduction

A central problem in genetics and medicine is to discover the associations between genetic variations in a human population and phenotypes such as disease. The most common of these variations are Single-Nucleotide Polymorphisms (SNPs), in which two (or occasionally more) nucleotides frequently occur at a given *polymorphic site* within some chromosome. The *genotype* of an individual specifies, for each of a selected set of polymorphic sites, the pair of nucleotides occurring at that site in a homologous pair of chromosomes. The genotype does not resolve the question of how the nucleotides at polymorphic sites are assorted between the two copies of the chromosome. The problem of resolving this ambiguity is called the *phase problem* in genotyping. A successful resolution of the phase problem determines an individual's pair of *haplotypes*, each of which specifies the contents of the polymorphic sites at one of the two copies of the chromosome.

Experimental protocols exist both for genotyping and, at greater expense, for haplotyping. Because genotyping is cheaper, a common approach is to determine the genotypes of individuals experimentally, and then attempt to infer their haplotypes computationally, sometimes with the help of pedigree data. Whichever approach is taken, the resulting genotypes or haplotypes will inevitably contain sites at which the correct nucleotide (or unordered pair of nucleotides in the case of genotyping), cannot be determined with certainty. The problem of resolving such missing data is one of the major themes of this paper.

Given a single genotype containing heterozygous sites and missing data, the problem of resolving it into a pair of completely specified haplotypes involves assigning the two nucleotides at each heterozygous site to one haplotype or the other, as well as filling in the missing data. Typically a single genotype can be resolved in many different ways. However, guidance to the correct resolution can be gleaned by simultaneously considering the genotypes of many individuals and applying two principles which are often valid over genomic regions of limited extent: first, that the same haplotypes tend to occur repeatedly in the genomes of many individuals, and secondly that the evolutionary history of these common haplotypes is very simple. In particular, the evolutionary tree is often a *perfect phylogeny*, in which there is only one mutation event for each polymorphic site.

This leads to the second major theme of the paper: the inference of perfect phylogenies from haplotypes or genotypes with missing data. This problem is isomorphic to a key problem in phylogenetics - the inference of a perfect phylogeny for a set of taxa when the binary character data is incomplete. The problem also occurs in historical linguistics, where the evolution of a set of languages is inferred from incomplete data about two-valued attributes of the languages; see [15] for a striking application of perfect phylogeny to such problems.

The third major theme is the use of a maximum likelihood principle to assign complete haplotypes to genotypes or haplotypes with missing data. This principle can be applied whether or not the set of complete haplotypes is restricted to a perfect phylogeny. We present a probabilistic generative model of haplotype or genotype data. The components of this model are the distribution of the complete haplotypes in a population, the stochastic process of pairing up complete haplotypes to form complete genotypes, and the stochastic process of masking certain components of the haplotypes or genotypes to create missing data. Within this model, we show that the most likely valid resolution of the incomplete haplotypes or genotypes is the one that minimizes the entropy of the resulting distribution of complete haplotypes. We present algorithms for finding or approximating the minimum-entropy assignment, both in the case of a perfect phylogeny and in the unrestricted case, and we present successful computational results on real data.

2 Basic Definitions and Summary of Results

A *complete haplotype* is an element of $\{0,1\}^m$, where each component indicates the nucleotide at a particular polymorphic position in an individual's genome We restrict ourselves to the most common case, in which only two nucleotides occur with non-negligible frequency at a given position; the two nucleotides at each position are encoded as 0 and 1. A *partial haplotype* is an element of $\{0,1,*\}^m$, where $*$ indicates that the nucleotide at a given position is undetermined. A complete haplotype ch is compatible with a partial haplotype ph if ch

and ph take the same value in each position where ph contains 0 or 1. A *complete genotype* is an element of $\{0,1,2\}^m$ corresponding to a pair of complete haplotypes which take the same value as the complete genotype at positions where the complete genotype contains 0 or 1, and take opposite values at positions where the complete genotype contains 2. Finally, a *partial genotype* is an element of $\{0,1,2,*\}^m$, where $*$ indicates that the position in the corresponding complete genotype is undetermined. Thus a pair ch_1, ch_2 of complete haplotypes is compatible with partial genotype pg if ch_1 and ch_2 both contain 0 in a position where pg contains 0, 1 in a position where pg contains 1, and opposite binary values where pg contains 2.

We consider the following problems:

P1 Given a set of n partial haplotypes, assign a compatible complete haplotype to each;

P2 Given a set of n partial genotypes, assign a compatible pair of complete haplotypes to each.

These problem statements are ill posed, because they give no reason for choosing one solution over another amongst the vast multiplicity of feasible solutions. We constrain the problem in two ways: first, by requiring that all the complete haplotypes come from a perfect phylogeny; and second, by imposing a maximum likelihood criterion which favors those solutions that are highly concentrated on a small number of complete haplotypes.

2.1 Perfect Phylogeny

A set of complete haplotypes $CH \subset \{0,1\}^m$, $|CH| = n$, can be given as a matrix $A \in \{0,1\}^{n \times m}$. We say that A has a perfect phylogeny if there exists a tree $T = (V, E)$ such that every node in $v \in V$ is labeled by a haplotype $h_v \in \{0,1\}^m$, every edge $e \in E$ is labeled by a column $c_e \in \{1, \ldots, m\}$, and the following conditions are met:

- Every row of A appears as a node label in T.
- No column $\{1, \ldots, m\}$ appears more than once as an edge label in T.
- For an edge $e = (u, v) \in E$, the haplotypes h_u and h_v match in all columns except at column c_e, where they differ. In other words, a site label c_e indicates a mutation, or a change in the value of the nucleotide at site c_e from 0 to 1, or from 1 to 0.

Let c_1, c_2 be columns of A. Let $V(c_1, c_2)$ be the set of values that the pair of columns takes on over all the rows of A, so $V(c_1, c_2) \subset \{(0,0),(0,1),(1,0),(1,1)\}$. It is a well known fact (see e.g. [3, 9]), that a necessary and sufficient condition for A to have a perfect phylogeny is that, for every pair c_1, c_2 of columns of A, $|V(c_1, c_2)| \leq 3$. We call the pairs of values in $V(c_1, c_2)$ the *valid pairs* for columns c_1, c_2.

Call a perfect phylogeny *compatible* with a given set of partial haplotypes if it contains complete haplotypes compatible with all the given partial haplotypes, and *compatible* with a given set of partial genotypes if it contains pairs of complete haplotypes compatible with all the given partial genotypes. Thus we are led to the following problems:

PP1 Construct a perfect phylogeny compatible with a given set of partial haplotypes (or determine that none exists).

PP2 Construct a perfect phylogeny compatible with a given set of partial genotypes (or determine that none exists).

Problem PP1 is NP-hard [13] but is solvable in near-linear time given any one of the haplotypes in the perfect phylogeny [8]. Problem PP2 is proven NP-hard in [12]. In the case where the given haplotype matrix does not contain missing data Gusfield [5] introduced a linear time algorithm.

For each of these problems we present a condition which is satisfied with high probability when the data is generated by a natural probabilistic process, and show that under this condition the perfect phylogeny is unique and can be recovered efficiently. Our recovery algorithm for PP1 runs in near linear time, and our recovery algorithm for PP2 runs in time $O(nm^{\omega-1})$, where $\omega \approx 2.37$ is the best currently known exponent for the matrix multiplication algorithm.

We turn now to P1, the problem of choosing the most probable assignment of compatible complete haplotypes to partial haplotypes. The problem has two main variations, according to whether the compete haplotypes are constrained to come from a given complete phylogeny or whether their choice is unconstrained. Let n be the number of partial haplotypes, n_i the number of partial haplotypes assigned to the ith complete haplotype, and let p_i be $\frac{n_i}{n}$. Under our probabilistic model the most likely assignment is the one that minimizes the entropy $\sum_i -p_i \log p_i$. It is shown in [7] that a simple greedy algorithm produces an assignment whose entropy is within a small additive constant of the minimum entropy. We measure the performance of the greedy algorithm on real data sets, and we find that even though the algorithm is very simple to state and to implement, its performance is comparable and in some cases better than the performance of state of the art phase reconstruction algorithms such as PHASE [14], HAP [2, 6] and HAPLOTYPER [10].

In the case where the complete haplotypes come from a perfect phylogeny they can be viewed as the nodes of a phylogenetic tree T, and the partial haplotypes can naturally be identified with subtrees of T, such that a complete haplotype is compatible with the partial haplotype if and only if the complete haplotype is a node of the corresponding subtree. Similarly, a partial genotype pg can be identified with a pair of disjoint subtrees of T, such that a pair of haplotypes is compatible with pg if and only if one of the haplotypes comes from each of the two corresponding subtrees. Therefore, in the case where the complete haplotypes come from a known perfect phylogeny, Problems P1 and P2 reduce to finding a minimum-entropy assignment of subtrees to nodes within them.

We do not know whether there is a polynomial-time algorithm to find a minimum-entropy assignment of subtrees of a tree T to nodes within them (by contrast, there is a straightforward linear-time algorithm for covering a set of subtrees with a minimum number of nodes). However, we present a dynamic programming algorithm to find the minimum-entropy assignment in the case where all subtrees are of diameter $O(\log \log m)$. Building on this special case, we present approximation algorithms that provide near-optimal solutions with high probability when the partial haplotypes are generated according to our probabilistic model and their number of is sufficiently large. We also provide a method which, given the perfect phylogeny and the set of partial haplotypes generated from a sample of complete haplotypes, gives an asymptotically consistent estimate of the number of occurrences of each complete haplotype in the sample; this information can be used not only to assign partial haplotypes to complete haplotypes, but also to estimate the accuracy of the assignment.

Finally, we present computational results on real data sets, demonstrating the effectiveness of a simple greedy algorithm for resolving genotypes with missing data, even in the absence of a perfect phylogeny.

3 The Probabilistic Models

Throughout the paper, we assume that the haplotypes given to the algorithms as input are randomly generated by different probabilistic generative models. The basic assumption used is that the human population consists of a set of haplotypes with frequencies, and that the haplotypes given to the algorithm are independently drawn from the distribution specified by the frequencies.

Formally, we say that a set of partial haplotypes is *randomly generated* if they are generated in the following way. There exists a (not necessarily known) set of complete haplotypes $h_1, \ldots, h_k \in \{0,1\}^m$, an unknown distribution $p_1, \ldots, p_k$ where $p_1 + p_2 + \ldots + p_k = 1$ and a masking probability p. The haplotypes are picked independently from the distribution one by one, that is, each new haplotype is chosen to be h_i with probability p_i. Once a haplotype h is chosen, the entries of the h are masked by independently replacing the value of each coordinate of h with a * with probability p. The choices to replace those values are independent of each other and are independent of the entries in the string h. We will usually refer to p as a constant smaller than $1/2$, and we assume that the haplotypes $h_1, \ldots, h_k$ are not known unless explicitly stated otherwise.

In some cases we limit ourselves to haplotypes that are randomly generated from a perfect phylogeny. In this case we add the additional constraint that the set of haplotypes $h_1, \ldots, h_k$ is taken from the nodes of a perfect phylogeny tree. We assume that the tree is not given to us unless explicitly stated.

Finally, the genotypes are randomly generated in the following way. We first randomly pick two haplotypes h and h' independently from the distribution $p_1, \ldots, p_k$, h and h' and we get a genotype given by their combination. We then mask the genotypes by masking each of their entries independently with probability p. The assumption that the two haplotypes are drawn independently is called random mating and is sometimes referred to in the literature as the Hardy-Weinberg equilibrium assumption.

Given a set of partial haplotypes $PH = \{ph_1, ph_2, \ldots, ph_n\}$ that are randomly generated by an unknown distribution and a set of complete haplotypes $CH = \{ch_1, \ldots, ch_n\}$ that are compatible with the partial haplotypes, we are interested in estimating the likelihood of these complete haplotypes to be the the ones drawn by the random data generator. Let $n_1, n_2, \ldots, n_l$ where $n_1+n_2+\ldots+n_l = n$ be the frequencies of the complete haplotype of CH. The probability that for each i, ch_i is the actual complete haplotype that was generated is $\Pi_i^l p_i^{n_i}$, where p_i is the unknown probability of the random data generator to generate the i-th complete haplotype (the one which appears n_i times in the data). The probability to observe PH and that CH will be generated by the random data generator is therefore $p^t(1-p)^{nm-t}\Pi_i^l p_i^{n_i}$, where t is the total number of masked entries in PH. Since the expression $p^t(1-p)^{nm-t}$ is fixed, we refer to the likelihood of the assignment as $\Pi_i^l p_i^{n_i}$. The likelihood of the assignment is therefore maximized when for each i, $p_i = n_i/n$. The likelihood of this assignment is $\Pi_i n_i^{\frac{n_i}{m}}$, and its logarithm is $\sum_i n_i \log n_i - n \log n$. In order to maximize this likelihood, we are interested in finding an assignment which maximizes the expression $\sum_i n_i \log n_i$. We call this problem The *Maximum-Concentration Assignment Problem*: Find an assignment of partial haplotypes in PH to complete haplotypes in CH that maximizes the *concentration* $\sum_i n_i \log n_i$. This likelihood model was first suggested in [6], and is consistent with other likelihood models. Note that maximizing the likelihood is equivalent to minimizing the entropy of the distribution $\{p_1, \ldots, p_l\}$, where $p_i = n_i/n$, for the following reason. The entropy is defined as $\sum_i -p_i \log p_i$. Minimizing the entropy is equivalent to maximizing $\sum_i p_i \log p_i = \frac{1}{n}\sum_i n_i \log n_i - \log n$, which in turn is equivalent to maximizing the concentration.

Reference [7] considers the maximum-concentration assignment problem in a general setting where CH and PH are arbitrary finite sets endowed with a compatibility relation $R \subseteq CH \times PH$. It is shown that maximum-concentration assignment problem is NP-hard, but that a simple greedy algorithm yields an approximate solution with additive error $O(m)$. In Section 6 we report on the performance of the greedy algorithm on haplotyping problems where CH is taken to be unconstrained (i,e,, $CH = \{0,1\}^m$. We further show in Section 5 how can one find the maximum concentration assignment when the data is randomly drawn from a known perfect phylogeny tree and the probability p is sufficiently small.

4 Incomplete Perfect Phylogeny Reconstruction

Reconstructing a perfect phylogeny from $A \in \{0,1,*\}^{n\times m}$ is NP-hard in general [13]. In the directed version, where the root of the tree is known, there is a near-linear algorithm for reconstructing a directed perfect phylogeny [8]. Therefore, for the undirected case, the problem can be reduced to finding one haplotype in the tree, since then we can root the tree at this haplotype and use the algorithm of [8].

The rich data hypothesis. Since the problem is NP-hard in general we consider the case where enough explicit information is given on the underlying tree. Rather than requiring a root to be fully specified, we require that the matrix meet a condition we call the *rich data hypothesis*, according to which the valid pairs of all pairs of columns appear somewhere in the matrix. We provide a linear time algorithm for finding one haplotype in perfect phylogeny tree that fits the matrix. In turn, we can use the near-linear time algorithm of Pe'er et al. [8] for complete reconstruction in $\tilde{O}(nm)$ time.

The rich data hypothesis imposes a very specific structure on the data. In particular, it is not hard to see by induction on the size of the tree, that for a matrix of complete haplotypes, if for every two columns in the matrix there are exactly three valid pairs, then the tree is unique. Thus, under the rich data hypothesis, if a tree exists then the tree is unique. On the other hand, the rich data hypothesis is not a necessary condition for

the uniqueness of the phylogeny tree. It is possible to construct examples where a matrix has a unique tree reconstruction, but not all valid pairs appear. Due to space restrictions, we will supply such an example in the full version of the paper.

It is not clear then, what are the necessary and sufficient conditions for the matrix to have a unique perfect phylogeny tree, and we leave this as an open problem.

We claim that the rich data hypothesis holds with high probability when a large number of partial haplotypes are randomly generated by the random data generator described in Section 3. Assume that the haplotypes are randomly generated from a perfect phylogeny tree with a distribution $p_1, \ldots, p_k$ and a masking probability $p \leq 1/2$. Let $x = \min_i p_i$. At each step of the generation of the data, the probability that the complete haplotype h_i will be drawn, and that the specific pair of entries c_1, c_2 will not be masked is $p_i(1-p)^2 \geq x/4$. Therefore, by the Chernoff bound, the probability that this event will never happen is at most $e^{-xn/10}$. By the union bound, the probability that there will be a haplotype h_i and a pair of entries that never appear unmasked in h_i is at most $m^2 k e^{-xn/10}$. Since x, m and k are fixed, for a sufficiently large n, with high probability the rich data hypothesis holds.

4.1 Finding the Tree

Given the valid pairs one can construct a conjunctive normal form formula from the matrix A as follows. For every entry a_{ij} in A we introduce a boolean variable x_{ij}. We then add the following constraints. If $a_{ij} = 1$ or $a_{ij} = 0$, we add the constraint that $x_{ij} = a_{ij}$. Then, for every two columns c_1 and c_2, let $(x, y) \in \{0,1\}^2$ be such that $(x, y) \notin V(c_1, c_2)$ - by the rich data hypothesis there is exactly one such pair (x, y). For every row r we add the constraint that $(a_{rc_1} \neq x) \vee (a_{rc_2} \neq y)$. One can easily verify that the disjunction of all these constraints can be written as a 2-SAT formula, and therefore, using a linear-time algorithm for 2-SAT, we can find an assignment to the entries of the matrix such that every partial haplotype in the original matrix is compatible with the assignment and that $|V(c_1, c_2)| = 3$ for every pair of columns c_1, c_2, that is, the matrix corresponds to a perfect phylogeny.

4.2 A Linear Time Algorithm

The running time of the algorithm described above is dominated by the time it takes to find the sets of valid pairs for each pair of columns. This procedure takes superlinear time. A naive implementation will take $O(nm^2)$, and a more sophisticated (but still standard) implementation can take $O(nm^{\omega-1})$ where $\omega \approx 2.37$ is the best known matrix multiplication exponent. The details of these algorithms are omitted from this version. Note that in both cases we get a superlinear algorithm. In this section we show how to find the tree in near-linear time.

As described above, by finding one haplotype, we can root the tree in that haplotype and then use the algorithm of [8] to find the whole tree. We present here a linear time algorithm that finds a root.

Let T be the underlying (unique) tree that fits the matrix A. We first need the following lemma, whose proof will be given in the appendix:

Lemma 1. *Let $h \in \{0,1\}^n$ be a complete haplotype located on one of the vertices of T. Then h corresponds to a leaf in T if and only if there is a column c in A such that all the rows r with $A_{rc} = h_c$ are compatible with h.*

By Lemma 1, in order to find a haplotype in T, we can look for a leaf of T. In particular, it is enough to look for a pair (c, x) of a column c and a value $x \in \{0,1\}$ such that for any two rows r, r', if $A_{rc} = x$ and $A_{r'c} = x$, then r and r' are consistent, that is the two rows agree whenever both values are not '*'. In this case we call (c, x) a leaf pair. Once we find a leaf pair, it is a straightforward procedure to find the haplotype h which corresponds to that leaf.

We now turn to the algorithm that finds a leaf pair. We first need some notations and definition. We assume that the columns are numbered $1, \ldots, m$. Let c be a column, and let $x \in \{0,1\}$, we denote by $R(c, x)$ the set of rows r such that $A_{rc} = x$. For two columns c_1, c_2, let $R_x(c_1, c_2) = \{A_{rc_2} \mid r \in R(c_1, x)\}$. That is,

ALGORITHM FIND-LEAF

Input: A matrix A satisfying the rich data hypothesis.
Output: A leaf pair (c, x).

1. Find the two candidates in the first two column $(1, x), (2, y)$. If $x = 1$ swap the values of the entries of the first column (i.e. 0 becomes 1 and 1 becomes 0). If $y = 1$ swap the values of the second column. Eventually, $C = \{(1, 0), (2, 0)\}$.
2. for $k = 3, \ldots, m$ do:
   (a) For every $c \in C$, if $0, 1 \in R_0(c, k)$ we say that c is split. If c is not split, let $v(c)$ be the only element of $R_0(c, k) \cap \{0, 1\}$ (we show in Lemma 2 that this set is not empty).
   (b) If there are no splits and $v(c) = x$ for every $c \in C$, then if $x = 0$ swap the values in the k-th column and add $(k, 0)$ as a candidate.
   (c) If there is exactly one split $c^* \in C$, then by Lemma 5, all the other candidates $c \in C$ have the same value $v(c)$. Let $y = 1 - v(c)$ be the other value. If $y = 1$ swap the values of the k-th column. Remove c^* from C and add k to C.
3. For each candidate $c \in C$ check if $R(c, 0)$ is consistnet. If it is consistnet stop and output this pair.

Figure 1: Algorithm FIND-LEAF.

$R_x(c_1, c_2) \subseteq \{0, 1, *\}$ is the set of values appearing in column c_2 in the sub matrix restricted to the rows of $R(c_1, x)$.

Before describing the algorithm in detail, we give an overview of the flow of the algorithm. The algorithm traverses the matrix from left to right, spending $O(n)$ time for each column. At each point of time we maintain a set of candidates for leaf pairs, $C = \{(c_1, x_1), (c_2, x_2), \ldots, (c_t, x_t)\}$, and we assume that the following invariants are preserved when we reach column k:

1. For each $(c, x) \in C$ we have that $x = 0$.
2. For every $i \neq j$, $i, j \in \{1, \ldots, t\}$, we have that $R(c_i, 0) \cap R(c_j, 0) = \phi$.
3. For every $c < k$ and $x \in \{0, 1\}$, if $(c, x) \notin C$ then (c, x) is not a leaf pair.

The first property is stated only for the convenience of the presentation. We will abuse notations and say that $c \in C$ if $(c, 0) \in C$. The algorithm is given in Figure 1.

The main objective of the algorithm when traversing the k-th column is to verify that the previous candidates are still consistent, and to add $(k, 0)$ or $(k, 1)$ to C unless they are proven to be inconsistent. Clearly, one cannot simply add both of them to C since then the invariants are not going to be preserved.

We now prove the correctness of the algorithm. Throughout the proof we assume that the tree T does exist, and that the rich data hypothesis holds. We first need a lemma which shows that the set $R_0(c, k) \cap \{0, 1\}$ is not empty (this is needed for step 2a in the algorithm). The proof is given in the appendix.

Lemma 2. *For every $c \in C$ there is at least one row $r \in R(c, 0)$ such that $A_{rc} \neq *$.*

The proof of the following is given in the appendix.

Lemma 3. *Let $c_1, c_2 \in C$. Then for every complete haplotype h of T, if $h_{c_1} = 0$ then $h_{c_2} = 1$.*

Note that in the flow of the algorithm, the size of C never decreases. Therefore, if we show that the third invariant property is preserved throughout the algorithm, then it is enough to check every candidate for consistency, and thus the algorithm finds a leaf pair if one exists.

The first invariant property always holds since the only candidates added to C throughout the algorithm are of the form $(c, 0)$. The second property can be easily verified by observing that whenever the algorithm adds a candidate it is disjoint to all previous candidates.

We now prove that the last invariant property holds by induction. One can verify that it holds for the first two columns. Assume that it holds before we traverse the k-th column. Note that we remove a candidate c from C only if it was split, in which case $R(c, 0)$ is not consistent. Therefore, it is sufficient to show that if we do not add (k, x) to C for some value x then $R(k, x)$ is not consistent. We show this by case analysis using the following three lemmas (the proofs are given in the appendix).

Lemma 4. *If c is not split and $v(c) = x$ for some $x \in \{0, 1\}$ then $R(k, x)$ is not consistent.*

Lemma 5. *If there is one split $c \in C$ then $v(c_1) = v(c_2)$ for every $c_1, c_2 \in C$.*

Lemma 6. *There is at most one split.*

Consider now the k-th iteration. By Lemma 6, we cover all cases in the algorithm. If there are no splits, then by Lemma 4, if there are $c_1, c_2 \in C$ such that $v(c_1) \neq v(c_2)$, then neither $R(k, 0)$ nor $R(k, 1)$ are consistent. If $v(c) = x$ for every $c \in C$, then again, by Lemma 4, $R(k, x)$ is not consistent, and the algorithm adds $(k, 1 - x)$ to C (if $x = 0$ it flips the values in that column). If there is exactly one split, then by Lemma 5 all other candidates have the same value y, and then by Lemma 4 $R(k, y)$ is not consistent, and the algorithm can leave it out of C. Therefore, the third invariant is maintained throughout the algorithm.

We now turn to show that the algorithm runs in linear time.

Theorem 1. *Algorithm* FIND-LEAF *finds a leaf pair in time $O(nm)$.*

Proof. Step 2 of the algorithm can be implemented in $O(n)$ time since the sets $R(c, 0)$ are pairwise disjoint. We run this step m times so the total running time before the last step of the algorithm is $O(nm)$. In the last step we have to check for each candidate whether it is consistent. It is easy to see that checking whether $R(c, 0)$ is consistent can be done in time $O(m \cdot |R(c, 0)|)$. Since the sets $R(c, 0)$ are pairwise disjoint this sums up to at most $O(nm)$. □

5 Maximum Concentration on a Tree

In the case where the complete haplotypes come from a *known* perfect phylogeny the haplotype assignment problem can be recast as a problem of assigning each subtree of a tree to a node within the subtree. Let T be the tree of a perfect phylogeny. We will speak interchangeably of a complete haplotype and the node in T that it labels. It is easy to verify that for any partial haplotype the set of compatible complete haplotypes is the set of nodes of a subtree of T. Thus assigning a partial haplotype to a compatible complete haplotype can be viewed as assigning a subtree to a node within it.

A similar situation holds for the problem of assigning a partial genotype pg to a pair of complete haplotypes. To see this, first observe that the set of single complete haplotypes compatible with pg is the set of nodes of some subtree T', just as in the case of a partial haplotype. Let S be the set of edges of T' corresponding to columns where the pg contains a 2. Then two nodes u and v of T' can be chosen as the pair of complete haplotypes of T' compatible with pg if and only if every edge in S lies on the path of T' between u and v. Except in the special case where S is empty, this uniquely determines two vertex-disjoint subtrees of T', each of which must contain one of the two chosen complete haplotypes; any such choice will work. When S is empty u and v need only be drawn from T', and need not be distinct.

We now turn to the problem of finding a maximum-concentration assignment of subtrees to nodes. As shown in Section 3, finding the maximum concentration assignment is a natural problem arising from the search for a maximum likelihood assignment. Let m denote the number of nodes in the tree T, and n, the number of distinct subtrees to be covered.

Any assignment of subtrees to nodes can be thought of as a disjoint cover of the subtrees, where each set in the cover is the set of subtrees mapping to a particular node. We will sometimes use the terminology of covers rather than assignments.

5.1 Constructing a Maximum-Concentration Cover

Theorem 2. *There is an algorithm for computing a maximum-concentration cover of subtrees of a tree T by nodes which, for trees of fixed maximum degree, runs in polynomial time when all subtrees have diameter $O(\log \log m)$.*

Proof. Let X be a multiset of subtrees of a tree T. Call a set $Y \subseteq X$ *consistent* if, whenever Y contains one copy of a subtree in the multi set X, it contains all copies of the subtree. Note that, in any maximum-concentration cover of any multiset, every multiset in the cover is consistent.

For each vertex v in T let $S(v)$ be the multiset of subtrees from X that contain v. Let T be rooted at an arbitrary node r. In this rooted tree each vertex v has a unique parent $p(v)$ (where by convention the root is its own parent), a set of children $C(v)$ and a set of descendants $D(v)$, where in particular $v \in D(v)$.

For each vertex v and each consistent multiset $A \subseteq S(v) \cap S(p(v))$, let $H(v, A)$ be the maximum concentration of a disjoint cover of $\cup_{u \in D(v)}(S(u) - S(p(v)) \cup A$. In particular, $H(r, X)$ is the maximum concentration of a disjoint cover of X by vertices of T.

The algorithm computes $H(v, A)$ for all v and A by working upward from the leaves of the rooted tree towards the root. Let the children of v be $u_1, u_2, \cdots, u_c$. Define a *legal partition* with respect to (v, A) as a family of disjoint consistent sets $A_0, A_1, \cdots, A_c$ such that, for $i = 1, 2, \cdots$, $A_i \subseteq S(u_i)$ and $\cup_{i=0}^{c} A_i = (S(v) - S(p(v)) \cup A$.

Then $H(v, A) = \min(|A_0| \log |A_0| + \sum_{i=1}^{c} H(u_i, A_i))$, where the minimum is taken over all legal partitions with respect to v, A.

The dynamic programming algorithm implied by this recursive formula computes a maximum-concentration cover of X in time $O(m2^{dk})$ where m is the number of nodes in T, d is the maximum number of children of any node , and k is the maximum, over all nodes v, of the number of distinct subtrees in $S(v) \cap S(p(v))$. This algorithm will run in polynomial time if $dk = O(\log m)$. This will be the case, for example, when all subtrees are of diameter $O(\log \log m / \log d)$. □

5.2 Approximation Algorithms

Theorem 3. *There is a quadratic-time approximation algorithm that solves the minimum-entropy subtree covering problem with relative error tending to zero as $m \to \infty$, provided that $n \geq m^c$ for some constant $c > 1$.*

Proof. The *greedy algorithm* for assigning subtrees to nodes simply repeats the following step until all subtrees have been assigned: choose a node occurring in the maximum number of subtrees remaining to be assigned, and assign all those subtrees to the node. A general result in [7] implies that the concentration of the resulting *greedy assignment* differs from that of a maximum-concentration assignment by at most $O(m)$. Since the maximum-concentration assignment has concentration at least $n \log(n/m)$, the relative error of the greedy assignment tends to zero when $n \geq m^c$. □

The greedy algorithm described in Theorem 3 does not use the information about the tree. In fact, this algorithm can be applied to a set of partial haplotypes that are not compatible with any perfect phylogeny tree. The following is an alternative approximation algorithm that may outperform the greedy algorithm in practice, and which uses the tree structure. Let X' be the multiset of subtrees within X of diameter at most $a \log \log m$ for some constant a. Using the above dynamic programming algorithm compute a minimum-entropy cover of X' having concentration C'. Ths can be done in polynomial time. Then insert each subtree in $X - X'$ into the largest set in this cover whose covering node contains the subtree.

Theorem 4. *The alternative approximation algorithm has relative error tending to 0 with high probability, under the following assumptions: $n \geq m^c$, for some $c > 1$; the partial haplotypes are generated by our probabilistic model; and $pd < 1$, where p is the masking probability and d is the maximum degree of T.*

Proof. Let C be the concentration of a maximum-concentration cover of X. Then $C \geq n \log(n/m) = \Omega(n \log n)$. Let t be the cardinality of the multiset $X - X'$. If the elements of $X - X'$ are deleted from

the maximum-concentration cover of X then the concentration is reduced by at most $t \log n$. Therefore there is a cover of X' with concentration at least $C - t \log n$. Hence the cover produced by the algorithm has concentration at least $C - t \log m$. Asymptotically this construction will achieve asymptotic relative error tending to 0 provided that $t = o(n)$.

We now now show that, under our assumptions, $t = o(n)$ with high probability. Each subtree has a root vertex (alias complete haplotype) drawn from the vertex set of T and consists of the connected component of that vertex in a random subtree of T in which each edge is present with probability p. The probability that a subtree so constructed has radius greater than or equal to $(a/2) \log \log n$ is bounded above by $(pd)^{(a/2) \log \log n} = O(1/\log n)$. It follows (by a Chernoff bound) that t, the number of subtrees of diameter greater than $a \log \log n$ is $O(\frac{n}{\log n})$. It follows that our construction produces an assignment with concentration $C(1 - O(1/logn))$ whp. □

5.3 Estimating Frequencies of Complete Haplotypes

Assuming our probabilistic model of the generation of partial haplotypes from complete haplotypes (or subtrees of T from nodes of T), we next give a method of estimating the number of subtrees generated from each node. Call the subtree from which a subtree is generated the *root* of that subtree. Let y_i be the number of subtrees containing node i. Let y_{ij} be the number of subtrees containing edge (i, j). Let x_i be the number of subtrees rooted at node i. Let f_{ij} be the number of subtrees containing edge (i, j) such that node i is on the path of T from the root of the subtree to node j. Given the y_i and the y_{ij} we would like to solve for the f_{ij}. Once we know the f_{ij} we can solve for the x_i by the formula $x_i = y_i - \sum_k f_{ki}$ where k ranges over the neighbors of i.

Consider a particular edge (i, j). Let z_{ij} be the number of subtrees containing node i such that i is on the path from the root of the subtree to j in T; such a subtree may or may not contain node j. Let Z_{ji} be defined similarly. We get the following equations: $y_i = z_{ij} + f_{ji}$, $y_j = z_{ji} + f_{ij}$, $y_{ij} = f_{ij} + f_{ji}$

The following additional equation would allow us to solve for f_{ij} and f_{ji}: $\frac{f_{ij} z_{ji}}{f_{ji} z_{ij}} = 1$. This gives the result $f_{ij} = \frac{y_{ij}(y_i - y_{ij})}{y_i + y_j - y_{ij}}$ and $f_{ji} = \frac{y_{ij}(y_j - y_{ij})}{y_i + y_j - y_{ij}}$.

We claim that the additional equation is "asymptotically correct;" i.e., for a given probability distribution over the nodes, the left-hand-side of the equation should approach 1 as the number of subtrees tends to infinity. The equation states that $\frac{f_{ij}}{z_{ij}} = \frac{f_{ji}}{z_{ji}}$. ; i.e., that the frequency with which a subtree contains edge (i, j), given that it contains i and i is on the path in T from the root to j, is equal to the corresponding frequency when i and j are interchanged; but each of these frequencies should approach the masking probability p as the number of subtrees grows. Thus, when the number of subtrees is large, the estimation of the f_{ij} derived from this equation will be accurate with high probability, yielding an accurate estimation of the x_i.

This estimation method yields another way of assigning each subtree to a node: choose the node within the subtree for which the associated estimated value of x_i is largest. The method has the advantage of providing an estimate of the probability that this choice is correct; namely, the estimated value of x_i, divided by the sum of the estimated values for all the nodes in the subtree.

6 Experimental Results

In Section 5.2 we presented a greedy algorithm for the maximum concentration problem that does not use any information about the tree. In haplotype terminology, given a set of partial haplotypes, the algorithm iteratively finds the complete haplotype which is compatible with the maximum possible number of partial haplotypes, removes this set of partial haplotypes and continues in that manner. In [7] a more general results implies that this process results in an additive error of $O(m)$ to the maximum concentration. A similar algorithm can be defined for partial genotypes, and in [7] it is shown that for genotypes the algorithm gives a multiplicative error of 2.

We measured the performance of the greedy algorithm in practice, both for genotype phase reconstruction and for haplotype missing data completion. We used a branch and bound procedure to find the complete haplotype which is compatible with the maximal number of partial haplotypes. Our results show that the greedy algorithm, which is very simple to state and to implement, performs reasonably well, and for parts of the data it is even better then previous phase reconstruction algorithms such as PHASE [14], HAPLOTYPER [10]and HAP [2, 6]. We would like to emphasize that one of the great advantages of the greedy algorithm is its simplicity compared to the algorithms mentioned above.

The data sets. We applied our algorithm to two haplotype data sets by [1, 11] and population D of [4]. The first data set is a 500 kilobase region of chromosome 5q31 containing 103 SNPs from the studies of [1] and [11]. In this study, genotypes for the 103 SNPS are collected from 129 mother, father, child trios from a European-derived population in an attempt to identify a genetic risk factor for Crohn's disease. A significant portion of the genotype data (about 10%) is missing with an average of 10 SNPs per individual's genotype missing. This data set was partitioned in [1, 11] into eleven blocks of high correlation. Since this set consists of trios, we can infer each individual's haplotypes in all positions except for the positions where all three individuals are heterozygous or missing. We use populations D from the [4] data which has pedigree information. The data consists of genotypes of SNPs from 62 regions. Population D consists of 90 individuals from 30 trios from Yoruba.

Completing missing haplotypes. We measured the performance of the greedy algorithm on haplotype data with missing data. We first took the data two data sets and considered the haplotypes of the parents inferred by the trios. These haplotypes contain a certain amount of missing data that is a result of the missing data in the original data and positions where the mother, father and the child had heterozygous or missing data. We added random missing data by masking each position independently with probability p for some value p. We then partitioned the data into blocks and ran the greedy algorithm on each of these blocks. The block size for the data taken from [4] was fixed as size 10, and the blocks size for the data taken from [1, 11] was determined by the blocks given in [1]. We then compared the resulting haplotypes of the greedy algorithm to the original haplotypes. We consider each masked position and we observed if it was correctly reconstructed or not. We found that the error rate in the reconstruction is only a few percents in both data sets, even when the missing data consists of about 25% of the data. The results are given in Figure 2 in the appendix.

Phasing genotype data We used the greedy algorithm to phase genotype data. We used the trios in order to infer the haplotypes of the parents, and we measured the performance of the algorithm on the set of genotypes of the parents. We compared our results to the results given by three other phasing algorithms, namely HAP [2, 6], PHASE [14] and HAPLOTYPER [10]. The results of the comparison are given in Figure 3 in the appendix.

We observe that although the greedy algorithm is much simpler to state then the other algorithms, the results achieved by the greedy algorithm are competitive with the other results. In fact, for the the data taken from [1], the performance of the greedy algorithm is superior to the performance of all the other algorithms. For the Gabriel [4] data, the greedy algorithm is inferior than the other algorithms, but it still gives reasonable results.

7 Future Work

Following our discussion, the following problems are left open and it would be interesting to settle them. First, it is still open whether there is a polynomial-time algorithm for constructing a perfect phylogeny compatible with a given set of partial haplotypes, under the assumption that exactly one such tree exists? What if the given set is a set of partial genotypes instead of partial haplotypes. Another problem that stands open is whether there a polynomial-time algorithm for computing a minimum-entropy cover of subtrees by nodes? Finally, does the greedy algorithm for assigning complete haplotypes to genotypes with missing data approximate the minimum-entropy assignment with additive error bounded above by a constant (such a result holds for haplotypes with missing data)?

References

[1] MJ Daly, JD Rioux, SF Schaffner, TJ Hudson, and ES Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29(2):229–32, Oct 2001.

[2] Eleazar Eskin, Eran Halperin, and Richard M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2003.

[3] C. Johnson G. Estabrook and F. McMorris. An idealized concept of the true cladistic character. *Math. Bioscience*, 29, 1976.

[4] GB. Gabriel, SF. Schaffner, H. Nguyen, JM. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, SN. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, ES. Lander, MJ. Daly, and D. Altshuler. The structure of haplotype blocks in the human genome. *Science*, 296:2225–2229, 2002.

[5] D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.

[6] E Halperin and E Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. Technical report, Hebrew University Computer Science, 2003.

[7] E. Halperin and R. M. Karp. On the greedy set cover algorithm. In preperation, 2003.

[8] R. Shamir I. Pe'er and R. Sharan. ncomplete directed perfect phylogeny. In *Proceedings of CPM*, pages 143–153, 2000.

[9] C. Meacham. Theoretical and computational considerations of the compatibility of qualitative taxonomic characters. *NATO ASI series on Numerical Taxonomy*, G1, 1983.

[10] Niu, Qin, Xu, and Liu. In silico haplotype determination of a vast set of single nucleotide polymorphisms. Technical report, Department of Statistics, Harvard University, 2001.

[11] JD Rioux, MJ Daly, MS Silverberg, K Lindblad, H Steinhart, Z Cohen, T Delmonte, K Kocher, K Miller, S Guschwan, EJ Kulbokas, S O'Leary, E Winchester, K Dewar, T Green, V Stone, C Chow, A Cohen, D Langelier, G Lapointe, Gaudet D, J Faith, N Branco, SB Bull, RS McLeod, AM Griffiths, A Bitton, GR Greenberg, ES Lander, KA Siminovitch, and TJ Hudson. Genetic variation in the 5q31 cytokine gene cluster confers susceptibility to crohn disease. *Nature Genetics*, 29(2):223–8, Oct 2001.

[12] R. Sharan. Personal communication, 2003.

[13] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

[14] M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68:978–989, 2001.

[15] D. Ringe T. Warnow and A. Taylor. Reconstructing the evolutionary history of natural languages. In *Proc. ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 314–322, 1996.

Data Set	Total missing	Added missing	Error rate
Daly et al.	26%	10%	2.8%
Gabriel et al.	10.5%	0.5%	8.1%
Gabriel et al.	15%	5%	7.4%
Gabriel et al.	20%	10%	7.8%

Figure 2: The performance of the greedy algorithm under the different data sets and the different missing data ratio. The first column specifies the data set on which the experiment was done. The second column specifies the total missing data given to the algorithm - this missing data contains the added missing data and the missing data given by the original data and since some positions are unresolvable by the trios. The third column correspond to the ratio of added missing data (that is - it specifies the value of p), and the fourth column specifies the error rate of the algorithm, that is the number of incorrectly reconstructed masked positions divided by the total number of masked positions.

	HAPLOTYPER	PHASE	HAP	GREEDY
Gabriel et al.	–	4.4 %	3.7 %	7.3 %
Daly et al.	4%	1.65 %	1.27 %	0.82 %

Figure 3: The results for the genotype phasing algorithm. Each column corresponds to a different algorithm and each row corresponds to a different data set. We did not run HAPLOTYPER on population D of the data from Gabiel et al. [4]. Evidently, on the Daly et al. data set, the greedy algorithm outperform the other algorithms. On the other hand, on the Gabriel data set the greedy algorithm does not perform as well, although its error rate is comparable to the other algorithms.

A Genotype Phase Resolution

As described in Section 3, we are interested in finding the phase of the genotypes, given that the genotypes are randomly generated from an unknown distribution. We next show how this can be done if the sample size is large enough, that is, when n is large enough.

Theorem 5. *For a set of randomly generated partial genotypes from a distribution $p_1, \ldots, p_k$, such that $p_i \geq 480\frac{\log n}{n}$, and such that the masking probability $p < \frac{1}{2}$ we can reconstruct the tree with probability at least $1 - \frac{1}{n^4}$.*

Proof. We claim that for each pair of columns, (c_1, c_2), the set of valid pairs is of size exactly three with high probability. In this case, we can reconstruct the tree using a 2-SAT solver in a similar way to the algorithm given in Section 4.1.

Let $V(c_1, c_2) = \{(x, y), (\bar{x}, y), (\bar{x}, \bar{y})\}$ for some $x, y \in \{0, 1\}$. For $(a, b) \in V(c_1, c_2)$, let P_{ab} be the probability that a haplotype with a pair (a, b) is drawn from by the random data generator. By the conditions of the theorem, $P_{ab} \geq (1-p)^2 480\frac{\log n}{n} \geq \frac{120 \log n}{n}$.

Assume first that the pair with the maximum probability is (x, y). Then, $P_{xy} \geq (1-p)^2 \frac{1}{3} \geq \frac{1}{12}$. Thus, the probability to see (x, y) as a homozygous pair is at least $\frac{1}{144}$ and the probability to see $(2, y)$ and $(2, 2)$ is at least $10\frac{\log n}{n}$. By applying the Chernoff bound, one can verify that the probability not to observe one of those pairs in the data is at most $e^{-4 \log n} = \frac{1}{n^4}$. Similarily, if the pair with the maximum probability is $(\bar{x}, y)$ then it is easy to verify that the probability not to observe one of the following pairs: $(\bar{x}, y), (\bar{x}, 2), (2, y)$ is at most $\frac{1}{n^4}$.

In the latter case, we simply get all the valid pairs by observing the data. In the first case, if we do not get all valid pairs, we infer that the pairs $(2, 2)$ came from (x, y) and $(\bar{x}, \bar{y})$ ((x, y) is the only homozygous pair), and thus get all valid pairs. □

B Proof of some lemmas

Lemma 1. *Let $h \in \{0,1\}^n$ be a complete haplotype located on one of the vertices of T. Then h corresponds to a leaf in T if and only if there is a column c in A such that all the rows r with $A_{rc} = h_c$ are compatible with h.*

Proof. If h corresponds to a leaf in T then consider the edge that is incident with h. This edge corresponds to a column c in A. Since h is a leaf, every haplotype $h' \neq h$ of T satisfies that $h'_c \neq h_c$. Consider a row r such that $A_{rc} = h_c$. Then for every haplotype $h' \neq h$ of T, h' is not compatible with r since $h'_c \neq A_{rc}$.

In order to show the other direction, we assume for contradiction that there is a complete haplotype h which is not a leaf of T such that all rows r with $A_{rc} = h_c$ satisfy that they are compatible with h. Let e_c be the edge of T corresponding to the column c. Let S_1, S_2 be the two connected components of $T \setminus e_c$. Assume without loss of generality that $h \in S_1$. If there is $h' \neq h$ such that $h' \in S_1$, let c_2 be such that $h_{c_2} \neq h'_{c_2}$. By the rich data hypothesis and the contradiction assumption, there is a row r such that $A_{rc} = h'_c, A_{rc_2} = h'_{c_2}$. Then, $A_{rc} = h_c$, but $A_{rc_2} \neq h_{c_2}$ which is a contradiction. □

Lemma 2. *(Lemma 2) For every $c \in C$ there is at least one row $r \in R(c,0)$ such that $A_{rc} \neq *$.*

Proof. Assume the contrary. Then $V(c,k)$ could only contain $(1,0)$ and $(1,1)$, which contradicts the rich data hypothesis. □

Lemma 3. *Let $c_1, c_2 \in C$. Then for every complete haplotype h of T, if $h_{c_1} = 0$ then $h_{c_2} = 1$.*

Proof. By the second invariant property, $R(c_1,0) \cap R(c_2,0) = \phi$. Therefore, by the rich data hypothesis, $V(c_1,c_2) = \{(1,0),(0,1),(1,1)\}$. The lemma then follows. □

Lemma 4. *If c is not split and $v(c) = x$ for some $x \in \{0,1\}$ then $R(k,x)$ is not consistent.*

Proof. If c is not split, then by the rich data hypothesis, $V(c,k) = \{(0,x),(1,0),(1,1)\}$. Therefore, no matter what is the value of x is, two of the valid pairs are $(0,x)$ and $(1,x)$, and thus $R(k,x)$ is not consistent. □

Lemma 5. *If there is one split $c \in C$ then $v(c_1) = v(c_2)$ for every $c_1, c_2 \in C$.*

Proof. Assume for contradiction that $v(c_1) \neq v(c_2)$. Then, by Lemma 3 (applied to c, c_1 and then to c, c_2) the valid pairs of (c,k) contain $(1,0)$ and $(1,1)$. On the other hand, since c is split, the valid pairs of (c,k) also contain $(0,0)$ and $(0,1)$ which is a contradiction to the rich data hypothesis. □

Lemma 6. *There is at most one split.*

Proof. Assume for contradiction that $c_1, c_2 \in C$ are both split pairs. Then, by Lemma 3 (applied to c_1, c_2), $|V(c_1,k)| = 4$. This is a contradiction to the existence of T. □