IPED: Inheritance Path Based Pedigree Reconstruction Algorithm Using Genotype Data

Dan He[1,⋆], Zhanyong Wang[2], Buhm Han[3,4],
Laxmi Parida[1], and Eleazar Eskin[2]

[1] IBM T.J. Watson Research, Yorktown Heights, NY 10598, USA
[2] Department of Computer Science, University of California Los Angeles, Los Angeles, CA 90095, USA
[3] Division of Genetics, Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA
[4] Program in Medical and Population Genetics, Broad Institute of Harvard and MIT, Cambridge, MA, USA
dhe@us.ibm.com

Abstract. The problem of inference of family trees, or pedigree reconstruction, for a group of individuals is a fundamental problem in genetics. Various methods have been proposed to automate the process of pedigree reconstruction given the genotypes or haplotypes of a set of individuals. Current methods, unfortunately, are very time consuming and inaccurate for complicated pedigrees such as pedigrees with inbreeding. In this work, we propose an efficient algorithm which is able to reconstruct large pedigrees with reasonable accuracy. Our algorithm reconstructs the pedigrees generation by generation backwards in time from the extant generation. We predict the relationships between individuals in the same generation using an inheritance path based approach implemented using an efficient dynamic programming algorithm. Experiments show that our algorithm runs in linear time with respect to the number of reconstructed generations and therefore it can reconstruct pedigrees which have a large number of generations. Indeed it is the first practical method for reconstruction of large pedigrees from genotype data.

1 Introduction

Inferring genetic relationships from genotype data is a fundamental problem in genetics and has a long history [5,9,1,6,10,12]. Pedigree reconstruction is a hard problem and even constructing sibling relationships is known to be NP-hard [7]. In this work, we focus on reconstruction methods using genotype data. Various methods have been proposed for automatically reconstructing pedigrees using genotype data, which can be categorized into two categories. The first category is methods which reconstruct the haplotypes of the unknown ancestors in the pedigree. Thompson [14] proposed a machine learning approach to find the pedigree that maximizes the probability of observing the data. As the method

⋆ Corresponding author.

M. Deng et al. (Eds.): RECOMB 2013, LNBI 7821, pp. 75–87, 2013.

reconstructs both the pedigree graph and the ancestor haplotypes at the same time, it is very time-consuming and can be only applied to small families of size 4-8 people. The second category is methods which reconstruct the pedigree directly without reconstructing ancestor haplotypes. Thatte and Steel [13] proposed a HMM based model to reconstruct arbitrary pedigree graphs. However, their model, in which every individual passes on a trace of their haplotypes to all of their descendants is unrealistic. Kirkpatrick et al. [7] proposed an algorithm to reconstruct pedigrees based on pairwise IBD (identity-by-descent) information without reconstructing the ancestral haplotypes. A generation-by-generation approach is employed and the pedigree is reconstructed backwards in time, one generation at a time. The input of the algorithm is the set of extant individuals with haplotype and IBD information available. At each generation, a compatibility graph is constructed, where the nodes are individuals and the edges indicate the pair of individuals which could be siblings. The edges are defined via a statistical test such that an edge is constructed only when the test score between the pair of individuals is less than a pre-defined threshold. Sibling sets are identified in the compatibility graph using a Max-clique algorithm iteratively to partition the graph into disjoint sets of vertices. The vertices in the same set have edges connecting to all the other vertices of the same set. Both categories of methods encounter difficulties depending on the structure of pedigree. When the individuals are not related through inbreeding, these methods are fast and accurate. However, when inbreeding is present, the reconstruction becomes much more complicated and these methods perform poorly.

In this work, we propose an efficient algorithm, IPED (**I**nheritance **P**ath based **Ped**igree Reconstruction), which enables the reconstruction of very large pedigrees, with and without the presence of inbreeding. Our algorithm follows the approach of [7] and starts from extant individuals and reconstructs the pedigree generation by generation backwards in time. For each generation, we predict the pairwise relationships between the individuals at the current generation and create parents for them according to their relationships. When we evaluate the pairwise relationships for a pair of individuals, we consider the pairwise IBD length for their extant descendants, namely the leaf individuals in the pedigree. We then apply a statistical test on the two individuals to determine if they are siblings or not siblings.

One of the challenges in our approach is to compute the expected IBD length between a pair of extant individuals efficiently, in the presence of inbreeding. The CIP and COP methods of [7] are efficient for outbreed pedigrees but very inefficient for inbred pedigrees. This is because for the inbreeding case the alleles from an extant individual can be inherited in an exponential number of ways from his or her ancestors with respect to the number of nodes in the pedigree graph. The CIP algorithm applies a random walk from the ancestor to sample these exponential number of ways to estimate the expected IBD length between a pair of extant individuals. In addition, the pedigree needs to be explored multiple times when constructing each generation. Therefore the algorithm is inefficient

even for relatively small number of generations. In our experiments, CIP can not finish for a family of size around 50 individuals with 4 generations.

In order to address this problem, we consider the inheritance paths between the ancestor and the extant individuals, where each inheritance path corresponds to one path in the pedigree from the ancestor to the extant individual. If we know all the inheritance paths from the ancestor to the extant individuals, we can estimate the probability that an allele of the extant individual is inherited from the ancestor. The probability can be further utilized to compute the expected average IBD length between a pair of extant individuals. However, the number of inheritance paths can be exponential. We observed although the number of inheritance paths can be exponential, their lengths are bounded by the height of the pedigree. Therefore we use a hash data structure to hash all the inheritance paths of the same length into a bucket and the number of buckets is bounded by the height of the pedigree and thus is usually small. We save the hash tables for each individual and we develop a dynamic programming algorithm to populate the hash table of the individuals generation by generation. By doing this, we avoid redundant computation of the inheritance paths where the entire pedigree needs to be explored repeatedly and thus the dynamic programming algorithm is very efficient. Also because we avoid the time-consuming sampling step by using the inheritance path, our algorithm IPED is extremely efficient and it does not need to specify whether or not inbreeding is present, which is a big advantage over COP and CIP. Our experiments show that our algorithm is able to reconstruct the pedigree with inbreeding for a family of size 340 individuals with 10 generations in just 14 seconds. To our knowledge, this is the first algorithm that is able to reconstruct such large pedigrees with inbreeding using genotype data.

2 Methods

2.1 Pedigrees

A pedigree graph consists of nodes and edges where nodes are diploid individuals and edges are between parents and children. Circle nodes are females and boxes are males. An example of pedigree graph is shown in Figure 1. Parent nodes are also called *founders*. In the example, individual 13,14,15 are *extant individuals* and their founders are individuals 9, 10 and 11, 12, respectively. *Outbreeding* means an individual mates with another individual from different family. In the example, 3,4 and 6, 7 are both outbreeding cases. *Inbreeding* means an individual mates with another individual from the same family. In the example, 9, 10 is inbreeding case. We can see inbreeding case is usually more complicated as an individual can inherit from his ancestors in multiple ways. For example, 13, 14 can inherit from 1, 2 in two ways but 15 can inherit from 1,2 in only one way.

As we only have extant individuals and we reconstruct their ancestors, the pedigree is reconstructed backwards in time. We use the same notion of generations in [7], namely generations are numbered backwards in time, with larger numbers being older generations. Every individual in the graph is associated with

a generation g. All the extant individuals are associated with g=1 and their direct parents are associated with generation g=2. The *height* of a pedigree is the biggest g. We define an *inheritance path* between a child and his ancestor the same as it is defined in [10], namely as a path between the two corresponding nodes in the pedigree graph. For example, the inheritance path between 1 and 15 consists of nodes 1-6-11-15. There are two inheritance paths between 1 and 13: 1-4-9-13 and 1-6-10-13. Also we assume the inheritance paths are not directed. In this work, we do not consider pedigrees with *half-siblings*, namely we assume an individual only mates with another individual in the same generation.

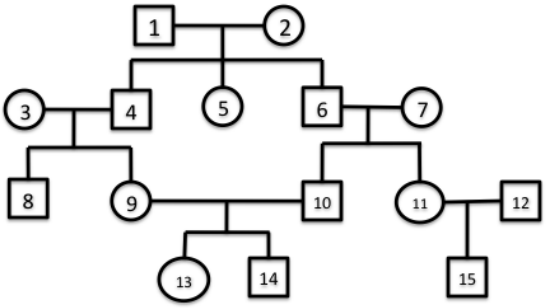

Fig. 1. An example of pedigree graph

2.2 Metrics to Evaluate the Relationship of a Pair of Individuals

As our algorithm reconstructs the pedigree generation by generation, we need to determine the relationship of any pair of individuals at a generation. We consider two different metrics for extant individuals and ancestral individuals, respectively.

To determine the relationship of a pair of extant individuals, we consider the IBD (identity-by-descent) length of the two individuals. In order to be distinguished from IBS (identity-by-state), the IBD region needs to be long enough, for example, of size 1Mb. If we are given the genotypes of the extant individuals, we can compute the IBD regions between a pair of individuals using existing tools such as Beagle [3]. In this work, in our simulation, we assume we are given haplotypes of the extant individuals and we consider identical regions of length greater than 1Mb between the two individuals as their IBD regions. We consider the averaged IBD length instead of total length of IBD to handle the cases where IBD regions are unevenly distributed. For simplicity, we use "IBD length" to denote "averaged IBD length".

Then for a pair of extant individuals i, j, we conduct a statistical test and compute a score $v_{i,j}$ as the following:

$$v_{i,j} = \frac{\left(estimate(IBD_{i,j}) - E(IBD_{i,j})\right)^2}{var(IBD_{i,j})} \tag{1}$$

where $estimate(IBD_{i,j})$ is the estimated IBD length between individuals i and j, $E(IBD_{i,j})$ is the expected IBD length between i and j, $var(IBD_{i,j})$ is the variance of the IBD length between i and j. $estimate(IBD_{i,j})$ can be computed easily given genotypes or haplotypes of individual i and j. As recombination occurs in meioses, it is shown [4] that the length of IBD between i and j follows an exponential distribution $exp(Mr)$, where M is the number of meioses between i and j, r is the recombination rate which is set as 10^{-8}, namely the probability for recombination occurs at any loci is 10^{-8}. Therefore, $E(IBD_{i,j})$ and $var(IBD_{i,j})$ are computed as the following:

$$E(IBD_{i,j}) = \frac{1}{M \times r} \tag{2}$$

$$var(IBD_{i,j}) = \frac{1}{(M \times r)^2} \tag{3}$$

For outbreeding case, $M = 2(g-1)$ where g is the generation. So for extant individuals, as we are constructing the second generation, $g = 2$. For inbreeding case, a random walk algorithm whose complexity is exponential is applied. More details will be given in the next section.

As we need to consider both paternal and maternal alleles, our IBD estimation is chromosome-wise instead of individual-wise. As i, j both have a pair of chromosomes noted as i_1, i_2, j_1, j_2, there are two possible ways to compare them for IBD, namely $[(i_1, j_1), (i_2, j_2)]$ or $[(i_1, j_2), (i_2, j_1)]$. We select the way that maximizes the sum of the averaged IBD length for both chromosomes. Without losing generality, assuming we select $[(i_1, j_1), (i_2, j_2)]$. Then we compute $v_{i,j} = \frac{v_{i_1,j_1} + v_{i_2,j_2}}{2}$, where v_{i_1,j_1} is computed according to Formula 1 by considering the estimated IBD between i_1, j_1. Notice $E(IBD_{i,j})$ and $var(IBD_{i,j})$ don't depend on the chromosomes of i and j.

In the method of Kirkpatrick et al. [7], if the test score $v_{i,j}$ is less than a pre-defined threshold value S, i, j are considered as siblings. However, it is not clear how to determine the value S and the threshold usually varies for individuals of different relatedness. In [7], the threshold is determined empirically by simulating many pedigrees. As we show in our experiments, the performance of the algorithms varies with the threshold.

In our work, we try to avoid using a threshold. As the pair of nodes are either siblings or non-siblings, we can compute the number of meioses between them for each case. For the case that they are siblings, the number of meioses is 2 and we can compute the length of the expected IBD using Formula 3. For non- sibling cases, we don't know exactly how many meioses there are between the pair of nodes. However, we can compute a lower bound for such number: namely the two nodes are first-cousin and the number of meioses is 4, which is the minimum number for a pair of non-sibling nodes. Then we can compute the length of the expected IBD for non-sibling again using Formula 3. We compare the two test scores and determine the pair of nodes are siblings if the test score for sibling case is lower.

To determine the relationship of a pair of ancestral individuals, we use a similar strategy as the one in [7]. Assuming individuals k and l are at generation $g > 1$. The sets of all extant descendants of k and l are K and L, respectively. We compute a score $v_{k,l}$ between k and l as

$$\begin{aligned} v_{k,l} &= \frac{1}{|K||L|} \sum_{i \in K} \sum_{j \in L} v_{i,j} \\ &= \frac{1}{|K||L|} \sum_{i \in K} \sum_{j \in L} \frac{\left(estimate(IBD_{i,j}) - E(IBD_{i,j})\right)^2}{var(IBD_{i,j})} \end{aligned} \tag{4}$$

where $|K|$ is the size of K, the number of extant descendants of k, $i \in K$ is an extant individual in K, $v_{i,j}$ is computed via Formula 1. Again, we compute $v_{k,l}$ for both sibling case and first-cousin case and determine k, l are siblings if the score for sibling case is lower. More details will be given in the next section on how to compute $E(IBD_{i,j})$ and $var(IBD_{i,j})$.

2.3 IPED: Inheritance Path Based Pedigree Reconstruction Algorithm

The computation of $E(IBD_{i,j})$ and $var(IBD_{i,j})$ is complicated in that the number of possible meioses between i and j can be exponential with respect to the nodes in the pedigree graph. To estimate the expected length of IBD between a pair of extant individuals, we need to consider all possible options for a pair of alleles to inherit from the shared ancestor, which is also exponential to the number of nodes in the pedigree. A random walk algorithm *CIP* from the founders with sampling is applied in [7]. However, the sampling is still time consuming in an exponential search space. What's more, as the reconstruction is generation-by-generation, from generation 2 to higher generation, the sampling strategy needs to be conducted every time when we move from one generation to the next generation backwards, which obviously involves redundant computation. Therefore, CIP is not efficient for inbreeding case. In our experiments, CIP can not finish for a family of size around 50 individuals with 4 generations.

To address the aforementioned two problems, we proposed a very efficient algorithm IPED (Inheritance Path based Pedigree Reconstruction Algorithm), which is based on the idea that the probability that a pair of alleles from two individuals are inherited from shared ancestor depends on the number of possible *inheritance paths* and their corresponding lengths from the shared ancestor. An example of inheritance path is shown in Figure 1. We can see the length of inheritance path determines the number of meioses between the two individuals and thus determines the probability of a pair of alleles from the two extant individuals inherited from the same ancestor. For example, the number of meioses between 8, 9 is 2 as they are siblings and the distance between them in the pedigree is 2. The number of meioses between 13 and 15 can be either 6 or 4, as there are multiple paths in the pedigree graph between them. In our algorithm, if there are multiple possible

numbers of meioses, we used the averaged value to approximate the IBD length. So for 13 and 15 the average number of meioses is 5.

Therefore, to determine the number of possible distances, or possible meioses between the extant individuals, for any founder in the current generation, we save the number of inheritance paths and the length of these inheritance paths from the founder to all the extant descendant individuals. Notice for inbreeding, there maybe an exponential number of inheritance paths with respect to the number of nodes in the pedigree. However, the length of the inheritance paths is finite, which is bounded by the height of the pedigree. Therefore, what we need to save is just a hash table with (length, number) pairs where the length of the inheritance path is the key and the number of inheritance paths with such length is the value. For example, there are 2 length-2 paths, 5 length-3 paths, 6 length-4 paths, then we just need to save three pairs (2,2), (3,5), (4,6), instead of saving all 9 paths separately. Therefore, we don't need to save exponential number of paths. Instead, we save only a small number of pairs, which is bounded by the height of the pedigree. Notice we need to save such pairs $[i, ((l_{i_1}, n_{i_2}), \ldots, (l_{i_k}, n_{i_k})]$ between the founder and every extant descendant of it, where i is the i-th extant descendant, (l_{i_k}, n_{i_k}) is the k-th (length, number) pair between the founder and the descendant. We call such pairs *Inheritance Path Pair (IPP)*. Given the number of extant individuals is fixed and is usually not a big number, the complexity is bounded by a constant.

The inheritance path pairs can be used to compute the possible distances, or the average number of meioses of a pair of extant individuals. Assuming a pair of founders G and K with inheritance path pairs $[i, ((l_{g_1}, n_{g_1}), \ldots, (l_{g_h}, n_{g_h})]$ and $[j, ((l_{k_1}, n_{k_1}), \ldots, (l_{k_f}, n_{k_f}))]$. The average number of meioses between individual i, j can be computed with Algorithm 1, where t is a test option. For sibling case, $t = 1$ and for first-cousin case, $t = 2$. Once the number of meioses is computed, it can be applied to Formula 3 directly to compute the statistic test score.

Algorithm 1. Calculate the average number of meioses between i, j

Input: t (test option), $[i, ((l_{g_1}, n_{g_1}), \ldots, (l_{g_h}, n_{g_h})]$ and $[j, ((l_{k_1}, n_{k_1}), \ldots, (l_{k_f}, n_{k_f}))]$
Output: The average number of meioses between i, j

```
Length ← 0
Num ← 0
for a = 1 to h do
  for b = 1 to f do
    Num ← Num + n_{g_a} × n_{k_b}
    Length ← Length + (l_{g_a} + t + l_{k_b} + t) × (n_{g_a} × n_{k_b})
  end for
end for
number of meioses ← Length / Num
```

Notice some of the inheritance paths may be shared by two extant individuals for inbreeding case. For example, in Figure 1, the inheritance paths between 1 and 15 1-6-11-15 and between 1 and 13 1-6-10-13 share one edge 1-11. Thus the

number of meioses is 4 instead of 6. Using the above algorithm, we will have 6 as the number of meioses. However, as we want to avoid saving the exponential paths explicitly, we just assume the paths do not overlap. Therefore, IPED is not optimal. Instead, it is an approximation algorithm. Another approximation our method is employing is that we approximate the mean and variance of the IBD length by using the average number of meioses (Algorithm 1). We also assume that if there are multiple paths between two individuals, it is not possible for the individuals to be IBD through one path at a locus and IBD through another path at the next locus. Such case should be rare in practice because multiple recombination events should simultaneously occur in the pedigree at one locus. Despite of these approximations, our experiments show that IPED achieves good reconstruction accuracy.

Once we save such pairs for each founder at one generation, when we reconstruct the next generation (the parents of the current generation) backwards, we need to compute such pairs between all the possible founders in the next generation and all the extant individuals. A naive algorithm is to compute the IPPs between every founder and every extant individual on each generation. However, this requires significant redundant computations since all the nodes of lower generation will be explored multiple times when computing the inheritance paths. We developed a dynamic programming algorithm where the IPPs of the current generation can be used to compute the IPPs of the next generation.

The dynamic programming algorithm starts the reconstruction from generation 2 as generation 1 consists of all the known extant individuals. Then at generation 2, assuming we have a founder G_2^i (without losing generality, assuming he is father) and his k children in generation 1 as $G_1^{i_1}, G_1^{i_2}, \ldots, G_1^{i_k}$. Then for every paternal allele of each child, obviously we have 1 possible length 1 inheritance path from the founder. Therefore, we save $[G_1^{i_j}, (1, 1)]$ for G_2^i for $1 \leq j \leq k$. Now let's assume we are at generation T, and we are reconstructing generation $T+1$. Again, assuming we have a founder G_{T+1}^i as father and his k children in generation T as $G_T^{i_1}, G_T^{i_2}, \ldots, G_T^{i_k}$. We then obtain the IPPs for G_{T+1}^i by merging the IPPs for $G_T^{i_1}, G_T^{i_2}, \ldots, G_T^{i_k}$. The recursion is shown as below:

$$IPP(G_{T+1}^i) = \sum_{j=1}^{k} IPP(G_T^{i_j}) + \mathbf{1}$$

where $IPP(G_{T+1}^i)$ is the set of IPPs for node G_{T+1}^i. Assuming for $G_T^{i_j}$, we have IPPs
$[G_1^t, ((L_{j_1}, N_{j_1}), \ldots, (L_{j_m}, N_{j_m}))]$, $IPP(G_T^{i_j}) + \mathbf{1}$ is to update these pairs as $[G_1^t, (L_{j_1}+1, N_{j_1}), \ldots, (L_{j_m}+1, N_{j_m})]$. $IPP(G_T^a)+IPP(G_T^b)$ is to merge two sets of IPPs. When we merge two pairs (L_a, N_a) and (L_b, N_b), if $L_a = L_b$, we obtain a merged pair (L_a, N_a+N_b). Otherwise we keep the two pairs. Therefore, after the merge, we obtain $[G_1^t, ((L_1, N_1), \ldots, (L_m, N_m))]$ for each extant individual G_1^t who is the descendant of G_{T+1}^i, where $L_1, \ldots, L_m$ are all unique and $m \leq T+1$. The summation $(\sum)$ is similarly defined as the repeated merging operation over multiple sets of IPPs.

An example of the dynamic programming algorithm is shown in Figure 2. As we can see in the example, when we merge the IPPs, we increase the length of the paths by 1 and add the number for the paths of the same length. The complexity of this dynamic programming algorithm is $O(E \times k \times H)$ where E is the number of extant individuals, k is the number of direct children for each founder, H is the height of the pedigree. Therefore it is linear time with respect to the height of the pedigree.

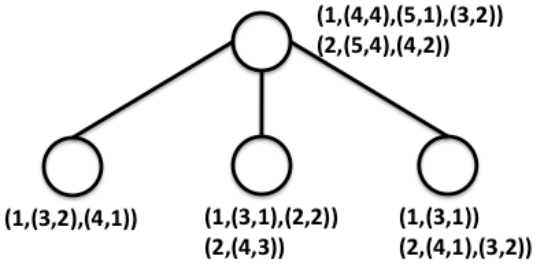

Fig. 2. An example of the dynamic programming algorithm

Once we compute the inheritance path pairs for each founder, we can calculate the number of meioses of any pair of extant individuals using Algorithm 1 and further compute the test score according to Formula 4.

2.4 Creating Parents

Once we determined the relationships of all the individuals of the current generation, we need to create parents for them. In order to guarantee that we create the same parents for all the individuals that are siblings, we create a graph for all the individuals at the current generation. Every individual is a node and there is an edge between a pair of nodes if they are determined as siblings according to the test. We call the graph *Sibling Graph*. Then we apply a Max-Clique algorithm [2] on the sibling graph for the current generation. We select the maximum clique where all the individuals in the clique are siblings to each other. We then create parents for them, and remove them from the sibling graph. We then select the next maximum clique from the remaining sibling graph and we repeat the procedure until all nodes are selected and all parents are created.

2.5 Performance Evaluation

Once we reconstructed the pedigree, we need to evaluate the accuracy of the reconstruction. We can not simply compare the reconstructed pedigree with the true pedigree directly due to graph polymorphism [8]. Therefore we consider the following metric:

$$accuracy(R, O) = \frac{\sum_{i \in E, j \in E} F(R_{i,j}, O_{i,j})}{|E|^2}$$

$$F(R_{i,j}, O_{i,j}) = \begin{cases} 1 & \text{if } R_{i,j} = O_{i,j} \\ 0 & \text{otherwise} \end{cases}$$

where R is the reconstructed pedigree, O is the original pedigree, E is the set of extant individuals, $|E|$ is the number of extant individuals, $R_{i,j}$ is the distance of individual i and j in pedigree R and $R_{i,j} = \infty$ if i, j are not connected in the pedigree graph. Notice if there are multiple paths between i and j in R, we select the shortest path. Therefore in this metric, we only compare the distance of extant individuals. If the distance between a pair of extant individuals in two pedigrees are the same (or two individuals are not connected in both pedigrees as the pedigrees are not high enough), we consider the reconstruction correct for this pair.

3 Experimental Results

We use the simulator from [7] to simulate the pedigrees. Instead of genotype data, we simulate haplotypes directly. The haplotypes of the individuals are generated according to the Wright-Fisher Model [11] with monogamy. The model takes parameters for a fixed populations size, a Poisson number of offspring and a number of generations (or the height of pedigree). We consider identical regions of length greater than 1Mb as IBD regions. We only compare our algorithm IPED with COP and CIP as the pedigree size in our simulation is relatively big and can not be handled by other algorithms. All the experiments are done on a 2.4GHz Intel Dual Core machine with 4G memory.

3.1 Outbreeding Simulation

We first test the outbreeding case. In the Wright-Fisher simulation, we fix the average number of children of each founder as 3, the individual of each generation is 20 and we vary the height of the pedigree. Notice according to the Wright-Fisher model, the number of individuals simulated each generation may not be 20. We compare the accuracy of COP and IPED. We randomly simulate 10 pedigrees for each parameter setting and show the averaged accuracy in Table 1. We can see that generally the accuracy drops as the generation and family size increase. IPED achieves slightly better results for outbreeding cases compared to COP. Also IPED is very fast, comparable to COP. For all different generations, IPED finishes in less than one second.

Next we show that COP algorithm is affected by the score threshold. As the empirically determined threshold is 0.7 in the work of [7], we vary the score threshold as 0.7 and 0.9. We show the results in Table 2. As we can see, the performance of COP varies with different thresholds. Our algorithm IPED, on the contrary has the advantage of not relying on any threshold.

Table 1. Outbreeding Accuracy for IPED and COP. Average number of children of each founder is 3. The number of individuals for each generation is 20. We vary the height of the pedigree.

Height	Family Size	IPED	COP
g = 3	52	0.966	0.955
g = 4	84	0.782	0.751
g = 5	144	0.831	0.836
g = 6	266	0.78	0.79
g = 7	384	0.706	0.655
g = 8	860	0.617	0.64

Table 2. Outbreeding Accuracy for COP with different test score thresholds. Average number of children of each family is 3. The number of individuals for each generation is 20. We vary the height of the pedigree.

Height	COP (0.7)	COP (0.9)
g = 4	0.905	0.89
g = 5	0.77	0.816
g = 6	0.874	0.895
g = 7	0.684	0.605

3.2 Inbreeding Simulation

Next we test the inbreeding case. As the CIP algorithm is very inefficient for inbreeding case, even for small pedigree it takes a long time and most often just simply crashes, we only compare our algorithm with CIP for pedigrees of height 3, with family size 40. IPED achieves an average accuracy of 0.91 while CIP achieves an average accuracy of 0.902 on 10 randomly simulated pedigrees.

Then we compare our algorithm with COP, which is aimed for outbreeding case, as it is able to finish fast on the simulated data sets. When COP is applied to a pedigree with inbreeding, it simply assumes there is only outbreeding in the pedigree.

We first fix the average number of children as 3, the individual of each generation is 20 and we vary the height of the pedigree. We show the averaged accuracy of IPED and COP in Table 3. We can see that for all generations, IPED achieves better results consistently. The accuracy generally drops for both methods. When the generation number is small, such as 3 and 4, the performances of IPED and COP are similar. However, as the pedigree gets bigger and more complicated, our algorithm significantly outperforms COP, which is reasonable as COP doesn't consider inbreeding. The algorithm CIP does consider inbreeding but it is not able to handle pedigrees of this size. IPED, on the contrary, is able to finish in just a few seconds for all parameter settings.

Next we show the performance of both algorithms for different family sizes. We vary the number of individuals of each generation as 20, 40 and 60. We set the generation number as 6. We show the averaged results from 10 random

Table 3. Inbreeding Accuracy of IPED and COP for different pedigree heights. Average number of children of each family is 3. The number of each generation is 20. We vary the height of the pedigree.

Height	Family Size	IPED	COP	improvement
g = 3	50	0.93	0.924	0.6%
g = 4	62	0.722	0.715	0.9%
g = 5	74	0.689	0.605	13.9%
g = 6	88	0.65	0.446	45.7%
g = 7	94	0.599	0.335	78.8%
g = 8	110	0.533	0.297	79.5%

Table 4. Inbreeding Accuracy of IPED and COP for different population size. Average number of children of each family is 3. We vary the number of individual for each generation used in the Wright-Fisher model as 20, 40, 60.

Number of Individual	Family Size	IPED	COP
S = 20	88	0.65	0.446
S = 40	156	0.66	0.55
S = 60	300	0.631	0.572

simulations in Table 4. We can see for all family sizes, our method achieves better accuracies, and the accuracies remain similar to each other, indicating the performance of our method is very stable w.r.t the size of the pedigree. Again, IPED is very fast and finishes in a few seconds for all datasets.

Finally we simulate a set of deep pedigrees and show the accuracy and running time of our algorithm in Table 5. As we can see, although the accuracy of IPED is relatively low, it is still a few times better than that of COP, the only existing algorithm that is able to handle such large pedigrees. In addition, IPED is faster than COP.

Table 5. Inbreeding Accuracy of IPED and COP for different family size. Average number of children of each family is 3.

Family Size	Generation	IPED	COP	IPED running time (.s)	COP running time (.s)
260	10	0.365	0.125	7	13
340	10	0.227	0.08	14	193

4 Conclusions

We proposed a very efficient algorithm IPED for pedigree reconstruction using genotype data. Our method is based on the idea of inheritance path where the time-consuming sampling can be avoided. A dynamic programming algorithm is developed to avoid redundant computation during the generation-by-generation

reconstruction process. We show our method is much more efficient than the state-of-the-art methods especially when inbreeding is involved in the pedigree. To our knowledge it is the first algorithm that is able to reconstruct pedigrees with inbreeding containing hundreds of individuals with tens of generations. Our algorithm still does not consider all possible complicated cases in pedigrees, such as half-siblings. Also it reconstructs pedigree only from the extant individuals. When the genotype of the internal individuals are known, it is helpful to use all such information. We would like to address these problems in our future work.

Acknowledgement. The authors would like to thank Bonnie Kirkpatrick for her help on the pedigree simulation.

References

1. Abecasis, G.R., Cherny, S.S., Cookson, W.O., Cardon, L.R.: Merlin-rapid analysis of dense genetic maps using sparse gene flow trees. Nature Genetics 30(1), 97–101 (2002)
2. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM 16(9), 575–577 (1973)
3. Browning, B.L., Browning, S.R.: A fast, powerful method for detecting identity by descent. The American Journal of Human Genetics 88(2), 173–182 (2011)
4. Donnelly, K.P.: The probability that related individuals share some section of genome identical by descent. Theoretical Population Biology 23(1), 34–63 (1983)
5. Elston, R.C., Stewart, J.: A general model for the genetic analysis of pedigree data. Human Heredity 21(6), 523–542 (1971)
6. Fishelson, M., Dovgolevsky, N., Geiger, D.: Maximum likelihood haplotyping for general pedigrees. Human Heredity 59(1), 41–60 (2005)
7. Kirkpatrick, B., Li, S., Karp, R., Halperin, E.: Pedigree reconstruction using identity by descent. Journal of Computational Biology 18(3), 1181–1193 (2011)
8. Kirkpatrick, B., Reshef, Y., Finucane, H., Jiang, H., Zhu, B., Karp, R.M.: Comparing pedigree graphs. Arxiv preprint arXiv:1009.0909 (2010)
9. Lander, E.S., Green, P.: Construction of multilocus genetic linkage maps in humans. Proceedings of the National Academy of Sciences 84(8), 2363 (1987)
10. Li, X., Yin, X., Li, J.: Efficient identification of identical-by-descent status in pedigrees with many untyped individuals. Bioinformatics 26(12), i191–i198 (2010)
11. Press, W.H.: Wright-fisher models, approximations, and minimum increments of evolution (2011)
12. Sobel, E., Lange, K.: Descent graphs in pedigree analysis: applications to haplotyping, location scores, and marker-sharing statistics. American Journal of Human Genetics 58(6), 1323 (1996)
13. Thatte, B.D., Steel, M.: Reconstructing pedigrees: A stochastic perspective. Journal of Theoretical Biology 251(3), 440–449 (2008)
14. Thompson, E.A.: Pedigree analysis in human genetics. Johns Hopkins University Press, Baltimore (1986)