SCALABLE COMPUTATION OF KINSHIP AND IDENTITY COEFFICIENTS ON LARGE PEDIGREES

En Cheng[*], Brendan Elliott, and Z. Meral Ozsoyoglu

Electrical Engineering and Computer Science Department,
Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH, 44106, USA
[*]*Email: exc92@case.edu*

With the rapidly expanding field of medical genetics and genetic counseling, genealogy information is becoming increasingly abundant. An important computation on pedigree data is the calculation of identity coefficients, which provide a complete description of the degree of relatedness of a pair of individuals. The areas of application of identity coefficients are numerous and diverse, from genetic counseling to disease tracking, and thus, the computation of identity coefficients merits special attention. However, the computation of identity coefficients is not done directly, but rather as the final step after computing a set of generalized kinship coefficients. In this paper, we first propose a novel Path-Counting Formula for calculating generalized kinship coefficients, which is motivated by Wright's path-counting method for computing the inbreeding coefficient for an individual. We then present an efficient and scalable scheme for calculating generalized kinship coefficients on large pedigrees using NodeCodes, a special encoding scheme for expediting the evaluation of queries on pedigree graph structures. We also perform experiments for evaluating the efficiency of our method, and compare it with the performance of the traditional recursive algorithm for three individuals. Experimental results demonstrate that the resulting scheme is more scalable and efficient than the traditional recursive methods for computing generalized kinship coefficients.

1. INTRODUCTION

In human genetics, pedigree diagrams are utilized to trace the inheritance of a specific trait, abnormality, or disease, calculate genetic risk ratios, identify individuals at risk, and facilitate genetic counseling. A sample pedigree diagram is shown in Figure 1a. Pedigrees are hierarchical hereditary structures and are typically represented as directed acyclic graphs. More specifically, a pedigree can be defined as "a simplified diagram of a family's genealogy that shows family members' relationships to each other and how a specific trait, abnormality, or disease has been inherited"[7]. Generally speaking, genetic counseling is the process by which patients or relatives, at risk of an inherited trait or disease, are advised of the consequences and nature of the trait or disease, the probability of developing or transmitting it, and the options open to them in management and family planning in order to prevent, avoid or ameliorate it. In order to calculate genetic risk ratios and identify individuals at risk, we need a measure of the degree of relatedness of two or more individuals. It is worthwhile to mention that calculating genetic risk ratios allows mainstream epidemiologists to leverage genetics for the study of diseases. In addition to the study of qualitative diseases, many developments in quantitative genetics also require knowledge of the probability that a pair of relatives have specified genotypes. Calculation of correlations between relatives

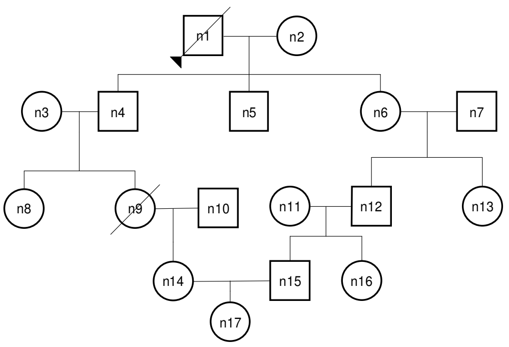


Fig. 1a. Small pedigree diagram

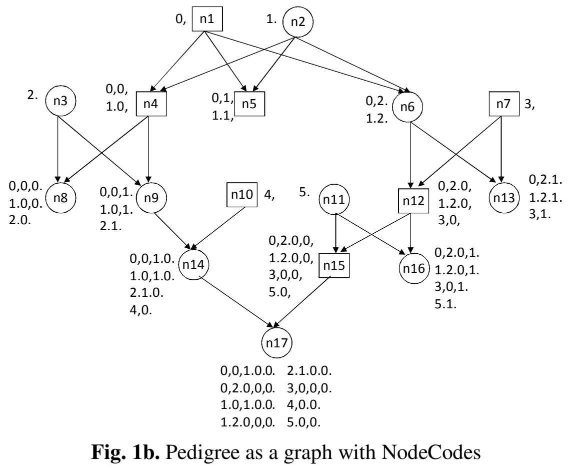


Fig. 1b. Pedigree as a graph with NodeCodes

[*] Corresponding author.

forms the foundation of classical biometrical analyses of quantitative traits such as height, weight, and cholesterol level[10]. In summary, making full use of genealogy information by measuring the degree of relatedness of a pair of individuals is a significant and practical issue in modern genetics.

Note that all measures of relatedness are based on the concept of identical by descent. Two genes are *identical by decent* (IBD) if one is a physical copy of the other or if they are both physical copies of the same ancestral gene. This concept is primarily due to Cotterman[3] and Malecot[14] and has been successfully applied to many problems in population genetics. The simplest measure of relationship between two individuals *a* and *b* is their kinship coefficient Φ_{ab}. The *kinship coefficient* Φ_{ab} is *the probability that a gene selected randomly from* a *and a gene selected randomly from the same autosomal locus of* b *are IBD.* While useful in many applications, the kinship coefficient does not completely summarize the genetic relation between two individuals. For instance, siblings and parent-offspring pairs share a common kinship coefficient of ¼. To better discriminate between different types of "pairs of relatives", *identity coefficients* were introduced by Gillois[6], Harris[8], and Jacquard[11]. Considering four genes of two individuals on a fixed autosomal locus, there are 15 possible IBD relations due to the fact that identity may exist *within* as well as *between* individuals. A notable characteristic of identity coefficients is that they provide a complete description of the probability of identity by descent between single loci of two individuals. Hence, this unique feature of identity coefficients has resulted in their application in a diverse range of fields. This includes the calculation of risk ratios for qualitative disease, the analysis of quantitative traits, genetic counseling in medicine, and wider studies of the genetic structure of populations.

A recursive algorithm for the calculation of identity coefficients proposed by Karigl[12] has been known for some time. This method requires that one calculate a set of generalized kinship coefficients, from which one can obtain the identity coefficients via a linear transformation. Although this recursive approach works well for small to moderate-size pedigrees, it can take impractical amounts of time when applied to very large pedigrees, particularly when coefficients are desired for many pairs of individuals. As data collection and storage technology are becoming more readily available at a lower cost, the size and variety of usable pedigree data has been increasing at a high rate. There are already large, heavily used pedigree data collections such as the Utah Population Database[15] with 1.6 million genealogy records. Thus, there is an urgent need for scalable techniques for efficiently calculating identity coefficients on large pedigrees due to both increasing volume of available pedigree data, and increasing use of pedigree data analysis in medical genetics for hereditary diseases.

In this paper, we propose a novel path-counting formula for the calculation of generalized kinship coefficients, motivated by Wright's path-counting formula for the computation of inbreeding coefficients. It has been previously shown that inbreeding coefficient queries can be efficiently evaluated using Wright's path-counting formula in conjunction with the NodeCodes encoding scheme[4]. Thus, once we have defined the path-counting formula, we can utilize NodeCodes and develop an efficient and scalable scheme for calculating the generalized kinship coefficients on very large pedigrees. We also present experimental results evaluating the performance of our strategy for calculating generalized kinship coefficients.

The main contributions of our work are as follows:

I. A novel path-counting formula for the calculation of generalized kinship coefficients.

II. An efficient and scalable scheme for calculating the generalized kinship coefficients and identity coefficients on large pedigrees using NodeCodes.

III. Experimental results demonstrating significant performance gains for calculating the generalized kinship coefficients for three individuals versus the traditional recursive algorithms.

2. RELATED WORK

There are two main approaches for computing kinship coefficients: a path-counting approach and an iterative approach[1]. The path-counting approach requires the detection of common ancestors and the summation of their contributions to the kinship coefficient. The iterative approach does not require the identification of paths through pedigrees. It begins with an initial group of individuals, and proceeds through the pedigree, computing successively the kinship between individuals who are descended from the initial population. The path-counting method has minimal storage requirements, but with some penalty in terms of computing time. The iterative approach is feasible to compute kinship coefficients for many individuals only if the kinship matrix is relatively sparse.

Among previous studies concerning the computation of identity coefficients, Karigl presented a description of identity coefficients and generalized kinship coefficients and proposed a technique that calculates the identity coefficient via a series of recursive calls to first calculate the generalized kinship coefficients and then a linear transformation is applied[12]. The generalized kinship coefficients include kinship coefficients for two, three, four, and two pairs of individuals. The basic problem is that each generalized kinship coefficient requires a separate recursion through the pedigree, which can be very time-consuming if the pedigree is very deep. Thus, the recursive algorithm can be infeasible when applied to very large pedigrees, particularly when coefficients are desired for many pairs of individuals.

Wright's formula[17], for computing the inbreeding coefficient of an individual is a typical example of path-counting formula. Utilizing an encoding scheme called NodeCodes in conjunction with Wright's formula, an efficient method for computing inbreeding coefficient is proposed by Elliott[4]. This paper was motivated by the question that whether we can extend the benefit of utilizing encoding schemes in calculation of the inbreeding coefficient to the computation of generalized kinship coefficients for more than 2 individuals..

3. BACKGROUND

This section describes condensed identity coefficients, generalized kinship coefficients, and path-counting formulas for standard kinship coefficient in more detail.

3.1. Condensed Identity Coefficients

If we consider four genes of two individuals on a fixed autosomal locus, then the 15 possible states can be reduced to 9 condensed identity states if we ignore the distinction between maternally and paternally derived genes. The states range from state 1 in which all four genes are IBD to state 9 in which none of the four genes are IBD. The probabilities associated with each condensed identity state, Δ_1 to Δ_9, are called *condensed identity coefficients*. The 15 states and their respective condensed identity coefficients are shown in Figure 2a.

The condensed identity coefficients can be computed from the generalized kinship coefficients (Φ_{ab} , Φ_{abc} , Φ_{abcd} , and $\Phi_{ab,cd}$) using the linear transformation shown in Figure 2b. Hence, we focus on the computation of generalized kinship coefficients.

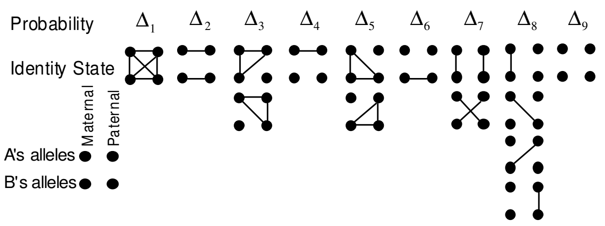


Fig. 2a. The 15 possible identity states for individuals *A* and *B*, grouped by their 9 condensed states. Lines indicate alleles that are IBD.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 2 & 2 & 1 & 1 & 1 \\ 4 & 0 & 2 & 0 & 2 & 0 & 2 & 1 & 0 \\ 8 & 0 & 4 & 0 & 2 & 0 & 2 & 1 & 0 \\ 8 & 0 & 2 & 0 & 4 & 0 & 2 & 1 & 0 \\ 16 & 0 & 4 & 0 & 4 & 0 & 2 & 1 & 0 \\ 4 & 4 & 2 & 2 & 2 & 2 & 1 & 1 & 1 \\ 16 & 0 & 4 & 0 & 4 & 0 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \\ \Delta_4 \\ \Delta_5 \\ \Delta_6 \\ \Delta_7 \\ \Delta_8 \\ \Delta_9 \end{bmatrix} = \begin{bmatrix} 1 \\ 2\Phi_{aa} \\ 2\Phi_{bb} \\ 4\Phi_{ab} \\ 8\Phi_{aab} \\ 8\Phi_{abb} \\ 16\Phi_{aabb} \\ 4\Phi_{aa,bb} \\ 16\Phi_{ab,ab} \end{bmatrix}$$

Fig. 2b. Linear transformation to calculate identity coefficients

3.2. Condensed Identity Coefficients

In addition to the kinship coefficients Φ_{ab} for two individuals, there is a set of generalized kinship coefficients for three, four, and two pairs of individuals, which are denoted as Φ_{abc} , Φ_{abcd} , and $\Phi_{ab,cd}$, respectively. Φ_{abc} (or Φ_{abcd}) is the probability that three (or four) randomly chosen genes, one from each individual, are IBD. $\Phi_{ab,cd}$ is the probability that a random gene from *a* is IBD with a random gene from *b* and that a random gene from *c* is IBD with a random gene from *d*.

Recursive equations for generalized kinship coefficients Φ_{abc} , Φ_{abcd} , and $\Phi_{ab,cd}$ are proposed by Karigl[12]. For example, the generalized kinship coefficient for three individuals, Φ_{abc} , is expressed as follows.

$\Phi_{abc} = \frac{1}{2}(\Phi_{fbc} + \Phi_{mbc})$ if *a* is not an ancestor of *b* or *c* (1.1)

$\Phi_{aab} = \frac{1}{2}(\Phi_{ab} + \Phi_{fmb})$ if *a* is not an ancestor of *b* (1.2)

$$\Phi_{aaa} = \frac{1}{4}(1 + 3\Phi_{fm}) \tag{1.3}$$

where *f* and *m* are the father and the mother of *a*, respectively, and $\Phi_{abc} = 0$ if there is no common ancestor of *a*, *b*, and *c*.

3.3. Path-Counting Formula

The approaches for computing the kinship coefficient Φ_{ab} are the iterative approach[12] and the path-counting approach[17]. The recursive formulas for Φ_{ab} used in the iterative approach[12] are:.

$$\Phi_{ab} = \tfrac{1}{2}(\Phi_{fb} + \Phi_{mb}) \text{ if } a \text{ is not an ancestor of } b \quad (1.4)$$

$$\Phi_{aa} = \tfrac{1}{2}(1 + \Phi_{fm}) \quad (1.5)$$

where f and m are the father and the mother of a, respectively, and $\Phi_{ab} = 0$ if there is no common ancestor of a and b. The iterative method exhaustively traverses the ancestors of a and b looking for common ancestors; when it finds them, it also recursively calculates each ancestor's inbreeding.

The path-counting approach is Wright's formula[17]:

$$\Phi_{ab} = \sum_{A}\sum_{P} (\tfrac{1}{2})^{L_{P(A,a)} + L_{P(A,b)} + 1}[1 + INC(A)] \quad (1.6)$$

where A is a common ancestor of a and b, $L_{P(A,\, a)}$ is the length of a path from A to a, $L_{P(A,\, b)}$ is the length of a path from A to b, and $INC(A) = \Phi_{fm}$ is the inbreeding coefficient of A. Paths from a to A to b that do not pass through the same individual more than once are identified and the probability of a gene being IBD is based on the number and length of these paths, modified by the common ancestor's own inbreeding.

4. PATH-COUNTING FORMULAS FOR GENERALIZED KINSHIP COEFFICIENTS

The recursive equations for generalized kinship coefficients were described in section 3.2. To make the computation of identity coefficients feasible for large pedigrees, we propose a set of path-counting formulas for generalized kinship coefficients. In this work, we will focus on showing how to generalize the path-counting formula for calculating the generalized kinship coefficient for three individuals (Φ_{abc}).

4.1. Terminology and Definitions

The following terminology and definitions for *path level* concepts will be utilized in presenting our path-counting formula for Φ_{abc}.

Triple-common ancestor: Given three individuals a, b and c, if A is a common ancestor of the three individuals, then we call A a *triple-common ancestor* of a, b and c.

Double-common ancestor: Given three individuals a, b and c, if D is a common ancestor of two of the three individuals, but it's not the common ancestor of the 3rd individual, then we say that D is a *double-common ancestor* of a, b and c.

P(A,a) denotes the set of all possible paths from A to a, where the paths can only traverse edges in the direction of parent to child such that $P(A,a) \neq \varnothing$ if and only if A is an ancestor of a. P_{Aa} denotes a particular path from A to a, where $P_{Aa} \in P(A,a)$. Let $I(P_{Aa})$ be the set of individuals on P_{Aa}.

Path-Triple denoted as $<P_{Aa}, P_{Ab}, P_{Ac}>$, where $P_{Aa} \in P(A,a)$, $P_{Ab} \in P(A,b)$, $P_{Ac} \in P(A,c)$.

Shared individual(s): The set of *shared individual(s) between two paths* P_{Aa} and P_{Ab}, denoted as $S_2(A, P_{Aa}, P_{Ab}) = I(P_{Aa}) \cap I(P_{Ab}) - \{A\}$, is non-empty if both P_{Aa} and P_{Ab} pass through a common set of individuals (excluding A). Likewise, the set of *shared individual(s) among three paths* P_{Aa}, P_{Ab}, and P_{Ac} is denoted as $S_3(A, P_{Aa}, P_{Ab}, P_{Ac}) = I(P_{Aa}) \cap I(P_{Ab}) \cap I(P_{Ac}) - \{A\}$.

Crossover & Overlap individual(s): If $s \in S_2(A, P_{Aa}, P_{Ab})$ (e.g. a double-common ancestor), we call s a *crossover individual* with respect to P_{Aa} and P_{Ab} if the two paths pass through different parents of s (i.e. one path passes through the mother and one passes through the father). On the other hand, if P_{Aa} and P_{Ab} pass through *same* parent of s, then we call s an *overlap individual* with respect to P_{Aa} and P_{Ab}.

Overlap Path: If s is an overlap individual with respect to P_{Aa} and P_{Ab}, then both P_{Aa} and P_{Ab} pass through the same parent-child edge (i.e. both mother or both father) and this edge is called an *overlap edge*. If this parent of s, denoted by p, is also an overlap individual on both paths, then there is an overlap edge regarding p as well. These two overlap edges are *consecutive* with respect to P_{Aa} and P_{Ab}. All consecutive edges constitute a path and this path is called an *overlap path*. If p is not an overlap individual, then s is simply a *crossover individual* and there is no overlap path. However, if the overlap path extends all the way to the triple common ancestor A, we instead call it a *root overlap path*. The *length of a path-triple* $<P_{Aa}, P_{Ab}, P_{Ac}>$ is denoted as $L_{<P_{Aa}, P_{Ab}, P_{Ac}>}$. Computing the length of a path-triple is given in the next section.

4.2. Path-Counting Formula for Φ_{abc}

Given a path-triple, we use the logic in Figure 3 to decide if a path-triple is counted toward the kinship value or rejected and the traversal through this diagram determines which *case* the path-triple belongs to. Identifying the case for a path-triple involves processing crossover, overlap, and shared individuals among three paths.

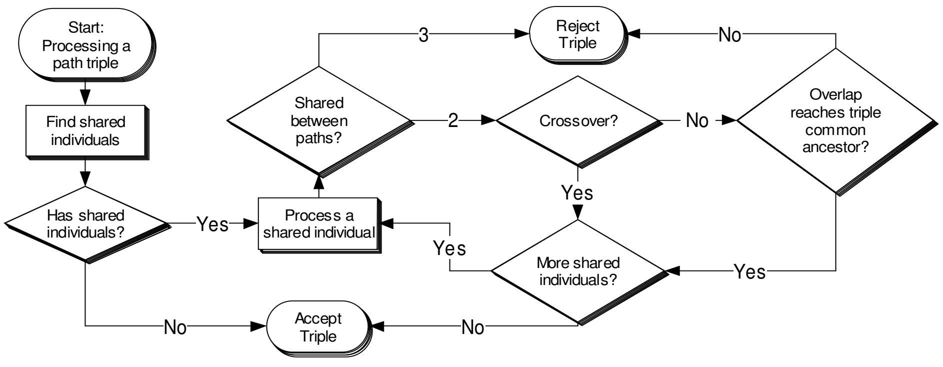
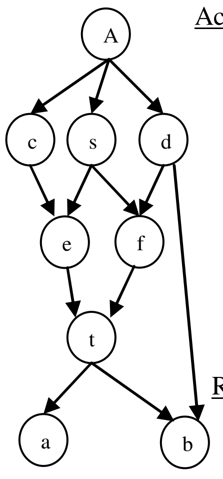


Fig. 3. Processing a path-triple



Accept Cases 1-4

$$Case\,1 :< P_{Aa}, P_{Ab}, P_{Ac} >= \begin{cases} A \to s \to e \to t \to a \\ A \to d \to b \\ A \to c \end{cases}$$

$$Case\,2 :< P_{Aa}, P_{Ab}, P_{Ac} >= \begin{cases} A \to s \to e \to t \to a \\ A \to d \to f \to t \to b \\ A \to c \end{cases}$$

where t is a crossover individual.

$$Case\,3 :< P_{Aa}, P_{Ab}, P_{Ac} >= \begin{cases} A \to s \to e \to t \to a \\ A \to s \to e \to t \to b \\ A \to c \end{cases}$$

where t is an overlap individual and the overlap path is a root overlap path.

$$Case\,4 :< P_{Aa}, P_{Ab}, P_{Ac} >= \begin{cases} A \to s \to e \to t \to a \\ A \to s \to f \to t \to b \\ A \to c \end{cases}$$

where t is a crossover individual; s is an overlap individual and the overlap path is a root overlap path.

Reject Cases 5-6

$$Case\,5 :< P_{Aa}, P_{Ab}, P_{Ac} >= \begin{cases} A \to c \to e \to t \to a \\ A \to c \to e \to t \to b \\ A \to c \end{cases}$$

where c is a shared indiviudal among three paths.

$$Case\,6 :< P_{Aa}, P_{Ab}, P_{Ac} >= \begin{cases} A \to c \to e \to t \to a \\ A \to s \to e \to t \to b \\ A \to c \end{cases}$$

where t is is an overlap individual and the overlap path is not a root overlap path.

Fig. 4. Six cases with respect to a path-triple

$$\Phi_{abc} = \sum_{A} \left(\sum_{\substack{<P_{Aa}, P_{Ab}, P_{Ac}> \in Case\,1\ or \\ <P_{Aa}, P_{Ab}, P_{Ac}> \in Case\,2}} (1/2)^{L_{<P_{Aa}, P_{Ab}, P_{Ac}>}+2} [1 + 3 * INC(A)] + \sum_{\substack{<P_{Aa}, P_{Ab}, P_{Ac}> \in Case\,3\ or \\ <P_{Aa}, P_{Ab}, P_{Ac}> \in Case\,4}} (1/2)^{L_{<P_{Aa}, P_{Ab}, P_{Ac}>}+2} [1 + INC(A)] \right) \quad (1.7)$$

According to Figure 3, we categorize all possible cases regarding a path-triple to 6 cases, and an example for each case is shown in Figure 4. Four of them are *accept cases* (1-4), in which case, they will contribute to the computation of Φ_{abc}. The other two cases are *reject cases* (5-6), and the path-triple does not contribute to the compuation of Φ_{abc}. A detailed description follows.

Case 1: $S_3(A,a,b,c) = \varnothing$ and no shared individual between any two of the three paths**.**

Case 2: only crossover(s) exist between any two of the three paths**.**

Case 3: only overlap(s) exist between any two of the three paths, but the overlap path is a root overlap path**.**

Case 4: both crossover(s) and overlap(s) exist between any two of the three paths, but the overlap path is a root overlap path.

Case 5: $S_3(A,a,b,c) \neq \varnothing$.

Case 6: overlap exists between any two of the three paths, but the overlap path is not a root overlap path**.**

Now, we can formally introduce a path-counting formula for Φ_{abc} (1.7) where A is a triple-common ancestor of a, b and c, and $INC(A)$ is the inbreeding coefficient of A.

Intuitively, case 1 and case 2 are simple triple-common ancestor paths to A (as in eq. 1.3), case 3 and case 4 are paths going through a double-common ancestor D which reduce to the kinship between A and D plus the distance to D (as in eq. 1.5), while case 5 and case 6 are the equivalents to traditional overlap for calculating Φ_{ab} by the path counting formula.

To utilize the equation (1.7) for computing Φ_{abc}, we need a method to calculate the length of a path-triple $L_{<P_{Aa}, P_{Ab}, P_{Ac}>}$. Let $L_{P_{Aa}}$ denote the total number of parent-child edges in P_{Aa}. Then $L_{<P_{Aa}, P_{Ab}, P_{Ac}>}$ is computed as follows.

$$L_{<P_{Aa}, P_{Ab}, P_{Ac}>} = \begin{cases} L_{P_{Aa}} + L_{P_{Ab}} + L_{P_{Ac}} & \textit{for case } 1 \,\&\, 2 \\ L_{P_{Aa}} + L_{P_{Ab}} + L_{P_{Ac}} - L_{P_{As}} & \textit{for case } 3 \,\&\, 4 \end{cases} \quad (1.8)$$

where s is an overlap individual and the overlap path is a root overlap path.

The path-counting formulas for Φ_{abcd} and $\Phi_{ab,cd}$ can be formulated using the approach given above for Φ_{abc}. For the rest of this paper, we focus on the computation of the generalized kinship coefficient for three individuals. The generalized kinship coefficients can be then directly utilized for the computation of identity coefficients.

5. CALCULATING Φ_{abc} USING NODECODES

In this section, we present an efficient and scalable NodeCodes-based scheme for our path-counting formula, motivated by the effectiveness of NodeCodes in conjunction with Wright's formula for inbreeding coefficient[4].

5.1. NodeCodes

NodeCodes is a graph encoding scheme originally proposed for encoding single source directed graphs[2,16], which was later adapted to encode pedigree data[5]. Pedigree data is represented by a directed acyclic graph, where the nodes represent individuals and directed edges represent parent-child relationships. Using NodeCodes, each node is assigned labels which are sequences of integers and delimiters. The integers represent the sibling order, and the delimiters denote the generations as well as indicating the gender of the node. We use ".", ",", and ";" to denote female, male or unknown respectively.

First the *progenitors* (nodes with in-degree 0) are labeled (we may consider adding a virtual root r and making all progenitors children of r). For each node u in the graph, the set of NodeCodes of u, denoted NC(u), are assigned using a depth-first-search traversal starting from the source node as follows:

- If u is the virtual root node r, then NC(u) contains only one element, the empty string.
- Let u be a node with NC(u), and v_0, v_1, … v_k be u's children in sibling order, then for each x in NC(u), a code $xi*$ is added to NC(v_i), where $0 \le i \le k$, and * indicates the gender of the individual represented by node v_i.

An example of NodeCodes is shown in Figure 1b using the pedigree from Figure 1a converted to a graph of parent-child edges.

5.2. Calculating Φ_{ab} and $INC(A)$

According to our path-counting formula (1.7), the calculation of Φ_{abc} requires the computation of $INC(A)$ as a final step. In our work, we utilize the efficient NodeCodes-based method described by Elliott[4] to compute $INC(A) = \Phi_{fm}$. Note that, inbreeding coefficient of an individual is actually the kinship coefficient for the individuals' parents. As a result, the method for computing inbreeding coefficient described by Elliott[4] can be utilized to calculate Φ_{ab} in general.

5.3. Calculating Φ_{abc}

The basic idea of the path counting formula for Φ_{abc} is to identify the common ancestors of a, b and c and sum their contributions to Φ_{abc}. Note that, the NodeCodes of an individual i effectively capture all ancestors that pass genes to i. Thus, given the NodeCodes of three individuals a, b, and c, we can identify all triple-common ancestors of a, b, and c via longest common prefix matching and each NodeCode from a, b, and c containing the shared prefix represents a path to the shared individual. We process each triple-common ancestor at path-level to form path-triples by taking the cross products of the sets of prefix-matched NodeCodes from a, b, and c to obtain all path-triples to be processed for that common ancestor. For each path-triple, we identify crossover, overlap, and shared individuals among three paths, and then utilize the logic described in Figure 3 to decide the triple's case and thus how it should contribute to the sum according to equation (1.7). This process is repeated for each such shared NodeCode prefix which is a *Longest Common Prefix* (LCP) for matching (which will be defined shortly) to obtain the final sum as the value for Φ_{abc}. The general outline for calculating Φ_{abc} using NodeCodes is presented in algorithm *Generalized-Kinship-Coefficient-* Φ_{abc}.

```
Algorithm Generalized-Kinship-Coefficient-$\Phi_{abc}$
 Input: NodeCodes NC(a), NC(b), and NC(c)
 Output: $\Phi_{abc}$
 1. Initialize $\Phi_{abc} = 0$.
 2. Identify a set of triple-common ancestors of a, b and c.
 3. For each common ancestor A
     a. Find a set of <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>.
     b. For each <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>
         - Process-Path-Triple (<$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>).
         - If <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 1 or Case 2 ,
              then $\text{var} = (\frac{1}{2})^{L_{<P_{Aa},P_{Ab},P_{Ac}>}+2}[1+3*INC(A)]$.
         - If <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 3 or Case 4,
              then $\text{var} = (\frac{1}{2})^{L_{<P_{Aa},P_{Ab},P_{Ac}>}+2}[1+INC(A)]$.
         - Otherwise, $\text{var} = 0$.
         - $\Phi_{abc} = \Phi_{abc} + \text{var}$.
 4. Return $\Phi_{abc}$.
```

```
Algorithm Process-Path-Triple
Input: <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>
Output: the case that <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$> fits in
1. Initialize crossover=false, overlap=false.
2. Identify a set of shared individuals between any two of
   the three paths, and among all three paths.
3. If no shared indiviudal,
     then return <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 1.
4. For each shared individual $s_i$
   -If $s_i$ is shared among all three paths,
       then return <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 5.
   -If $s_i$ is a crossover individual, then crossover=true.
   -Else, check if the overlap path is a root overlap path.
       - If it is a root overlap path, then overlap=true.
       - Otherwise, return <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 6.
5. If crossover=true && overlap=false,
     then return <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$> ∈ Case 2.
6. If crossover=false && overlap=true,
     then return <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 3.
7. If crossover=true && overlap=true,
     then return <$P_{Aa}$, $P_{Ab}$, $P_{Ac}$>∈ Case 4.
```

In this algorithm, step 2 and step 3.a are based on finding the LCP for matching and then find the unique set of shared individuals by treating the prefixes as NodeCode and retrieving individual identifiers by the NodeCodes to eliminate duplicates. Step 3.b calls the algorithm *Process-Path-Triple*, which implements the logic presented in Figure 3, to return path-triple's case. In this procedure, we identify crossover, overlap individuals, and root overlap paths, which are the critial steps for processing a path-triple. We will explain them in detail.

Longest Common Prefix (LCP) for matching: Let X, Y, and Z be (sub)sets of the NodeCodes for *a*, *b*, and *c*. Then *p* is the longest common prefix for matching X, Y, and Z, if there is no *p'* where *p* is a prefix of *p'*, and *p'* is a common prefix of **all** x_i in X, **all** y_i in Y, and **all** z_i in Z.

Identifying triple-common ancestors: We use the notation *p*=LCP(X,Y,Z) to denote that *p* is the LCP for matching sets X, Y, and Z. Given NodeCodes NC(*a*), NC(*b*), and NC(*c*), identifying triple-common ancestors requires matching NC(*a*), NC(*b*), and NC(*c*) having the longest common prefix for matching sets.

Identifying path-triples: Let *A* be a triple-common ancestor of *a*, *b*, and *c*, p_i, $1 \le i \le k$, be the NodeCodes of *A* such that p_i=LCP(X_{pi},Y_{pi},Z_{pi}) for some nonempty subsets X_{pi}, Y_{pi}, and Z_{pi} of NC(*a*), NC(*b*), and NC(*c*), respectively. Let *p* be any one of such p_i's. Then, the set of path-triple from *A* to *a*, *b*, and *c* can be represented as PT(*A*, *p*)={(*x*,*y*,*z*)| *p*=LCP(X_p, Y_p ,Z_p) and $x \in X_p$, $y \in Y_p$, and $z \in Z_p$ }.

Identifying crossover and overlap individuals: If *s* is a shared individual between two paths P_{Aa} and P_{Ab}, then there must be a NodeCode $n_{Aa} \in NC(s)$ that is proper prefix of P_{Aa} and a NodeCode $n_{Ab} \in NC(s)$ that is proper prefix of P_{Ab}. We call *s* a crossover individual with respect to P_{Aa} and P_{Ab} if n_{Aa} and n_{Ab} pass through different parents of *s* (i.e. one code passes through the mother and one passes through the father, identified by gender delimiters). However, if n_{Aa} and n_{Ab} pass through *same* parent of *s*, then *s* is an overlap individual with respect to P_{Aa} and P_{Ab}.

Identifying the root overlap path: If *s* is an overlap individual with respect to P_{Aa} and P_{Ab}, then there must be a NodeCode $n_{Aa} \in NC(s)$ that is proper prefix of P_{Aa} and a NodeCode $n_{Ab} \in NC(s)$ that is proper prefix of P_{Ab}. We identify an overlap path with respect to *s* as a root overlap path if n_{Aa} is equal to n_{Ab}; otherwise, it is not a root overlap path.

5.4. Computing Φ_{aab} and Φ_{aaa}

When calculating the condensed identity coefficients, we also need to directly calculate Φ_{aab} and Φ_{aaa}. However, these cases can be transformed and reduced to Φ_{abc} and Φ_{ab}, respectively, which can directly be computed according to (1.7) and Wright's formula (1.6).

For Φ_{aab}, assume *a* has two virtual children *x* and *y*, and we first compute Φ_{xyb}. According to the recursive formula (1.1), we get $\Phi_{xyb} = (1/2)^2 * \Phi_{aab}$, which can be rewritten as $\Phi_{aab} = 4.0 * \Phi_{xyb}$. To evaluate this using NodeCodes, we can artificially construct the NodeCodes for *x* and *y* based on the NodeCodes for *a.* With $NC(x)$ and $NC(y)$, we can apply the formula (1.7) to compute Φ_{xyb}.

For Φ_{aaa}, we evaluate it by substituting equation (1.3). Again, finding the inbreeding of *a* is done using the NodeCodes-based method proposed by Elliott[4]. Thus, we can now fully compute the generalized kinship coefficient for two or three individuals.

6. EXPERIMENTS

In this section, we show the efficiency of our path-counting method using NodeCodes for Φ_{abc} by making comparisons with the performance of a recursive method proposed by Karigl[12]. We examine the performance of Φ_{abc} using data from the Cleveland Clinic's Familial Polyposis Registry[9] and synthetic pedigrees[4]. Results for Φ_{ab} are equivalent to finding the inbreeding coefficient as in Elliott's work[4], where experiments showed speed improvements of 3-9 times.

6.1. Experimental Setup

The Cleveland Clinic's Familial (CCF) Polyposis Registry[9] database contains pedigrees of 750 families and 11,350 patient histories recorded in the past twenty-five years at CCF. We performed experiments on this dataset using 654 pedigrees containing 8,345 individuals, with the largest pedigree consisting of 118 individuals spanning 8 generations. In order to test scalability of our method, we used twelve synthetic pedigrees[4] ranged from 77 individuals spanning 3 generations for the smallest to 195,197 individuals spanning 19 generations for the largest. The data is stored in a SQLServer database.

We compared the execution time required to calculate Φ_{abc} by the recursive method described by Karigl [12] and the path-counting method using NodeCodes. We analyzed the effects of pedigree size (# individuals), the depth of individuals in the pedigree (the longest path between the individual and a progenitor), and the kinship coefficient value.

6.2. Experimental Results

In the first experiment, 500 random triples were selected from each of our 12 synthetic pedigrees. For each triple, the query was run on cold cache starting with no memoization data to show how the cost of calculating kinship increases with pedigree size for the recursive algorithm and the path-counting method using NodeCodes. We refer to the recursive method as *KinshipIter* and we refer to the path-counting method using NodeCodes as *KinshipNC*.

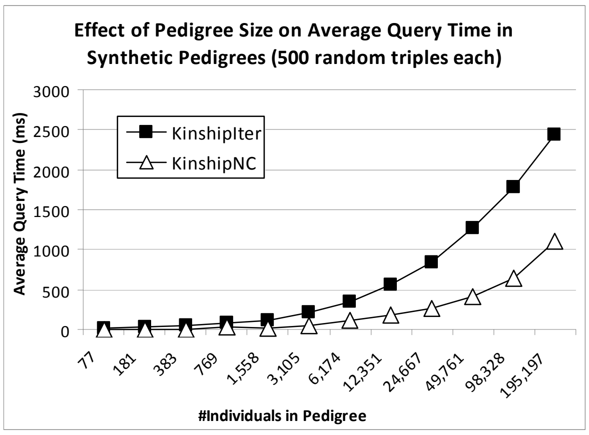


Fig. 5. Effect of pedigree size on average query time in synthetic pedigrees

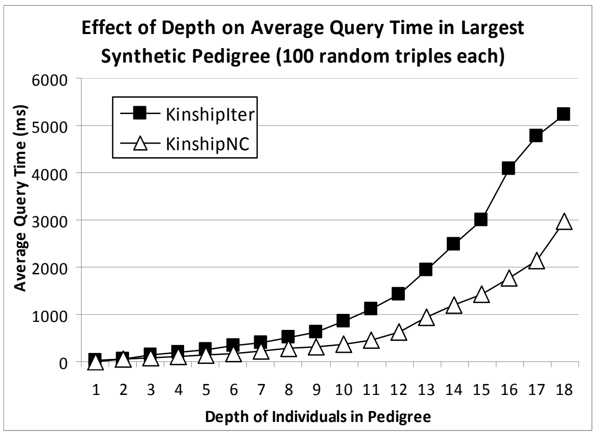


Fig. 6. Effect of depth on average query time in largest synthetic pedigree

Figure 5 shows the average time per query for each pedigree. As can be seen, the average time per query grew increasingly larger for *KinshipIter* method compared to *KinshipNC* as the pedigree size increased, from a comparable amount of time on the small pedigrees (<800 individuals) to 2.2-3.1 times slower per query than *KinshipNC* on the larger pedigrees (>1200 individuals).

In our next experiment, we examined the effect of the depth of the individual in the pedigree (number of steps in the longest NodeCode) on the query time. For each depth, we generated 100 random triples from the largest synthetic pedigree. Figure 6 shows how the average time per query grows as the individual's depth increases. We can see that *KinshipNC* scales better than *KinshipIter*, 1.7-2.3 times faster than *KinshipIter* for large pedigrees. The reason for this is that *KinshipNC* can skip intermediate generations and can jump straight to the common ancestors.

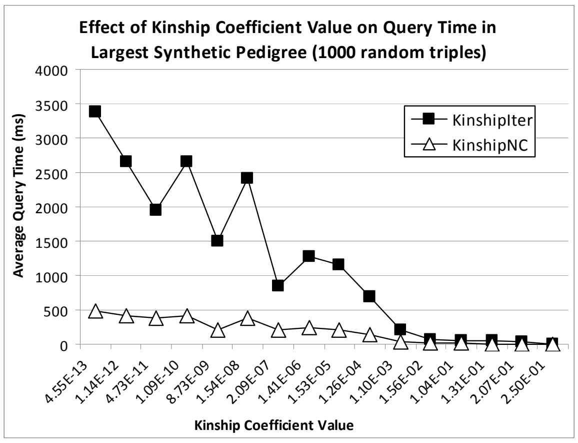


Fig. 7. Effect of kinship coefficient value on average query time in largest synthetic pedigree

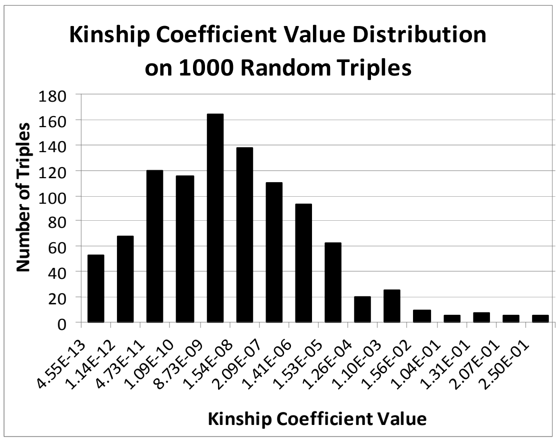


Fig. 8. Kinship coefficient value distribution for triples in Fig. 7

Next, we generated 1,000 random triples from the largest synthetic pedigree and investigated the effect of the kinship coefficient value on query time. For the kinship coefficient value, we expected that individuals with larger kinship coefficients would be more inbred and have more NodeCodes, causing *KinshipNC* to suffer slightly. Figure 7 shows the average query time for each distinct kinship coefficient value, and we can see that for most values of the kinship coefficient, *KinshipNC* outperformed *KinshipIter* by 4.9-7.3 times. As expected, for a few of the highest kinship coefficient values, we see slightly less improvement with respect to the performance of *KinshipNC* over that of *KinshipIter* (only 2.5 times faster). Figure 8 shows the distribution of the kinship coefficient value for the triples used in Figure 7. It clearly shows that the low-range values account for most of the triples, and for those values, *KinshipNC* outperforms *KinshipIter*.

Finally, we compared the results from our experiment on all the real pedigrees, which are all relatively small in comparison; the results are shown in Table 1. We randomly picked 43,862 triples on the real pedigrees. According to the ratio in the table, we can tell the *KinshipNC* is around 8.90 times faster than *KinshipIter*.

Table 1. Performance results on real data

	KinshipIter	KinshipNC	Ratio
Average Time Elapsed (ms)	29.17	3.28	8.90
Average SQL Queries Run	26.10	3.15	8.30

7. CONCLUSION

We have proposed a path-counting formula (PCF) for generalized kinship coefficient by generalizing Wright's path-counting method for three individuals. Based on our PCF, we presented an efficient and scalable method using NodeCodes for the computation of generalized kinship coefficient. We also implemented and tested our method using both real and synthetic data of various sizes to test scalability. Experimental results show that the use of NodeCodes for PCF achieves 2.2-8.9 times faster performance for computing generalized kinship coefficient on pedigree data, especially for real pedigrees as well as synthetic pedigrees of sizes between 800 and 200,000. Our future work includes (i) generalizing PCF for remaining generalized kinship coefficients, (ii) developing a scalable method for calculating identity coefficients utilizing the PCF and an encoding of paths such as NodeCodes.

Acknowledgement

We would like to thank Elena Manilich, Dr. James Church, and the Cleveland Clinic's Familial Polyposis Registry[9] for kindly allowing us to use their data for this study. This work is partially supported by the US National Science Foundation grants DBI-0218061, ITR-0312200 and CNS-0551603.

References

1. Boyce AJ. Computation of inbreeding and kinship coefficients on extended pedigrees, *Journal of Heredity* 1983; **74**:400-404.
2. Bozkaya T, Balkir N, Lee T. Efficient Evaluation of Path Algebra Expressions. *CWRU Tech. Report*, 1997.
3. Cotterman CW. A calculus for statistico-genetics. Unpublished Ph.D thesis, Ohio State University, Columbus, Ohio. Reprinted in Ballonoff, P. (Ed.). *Genetics and Social Structure*, Dowden, Hutchinson & Ross, Stroudsburg, P.A., 1974.
4. Elliott B, Akgul SF, Mayes S, Ozsoyoglu ZM. Efficient Evaluation of Inbreeding Queries on Pedigree Data. In *Proceedings of SSDBM* 2007; **9**: 3-12.
5. Elliott B, Akgul SF, Ozsoyoglu ZM, Manilich E. A Framework for Querying Pedigree Data. In *Proceedings of SSDBM* 2006; **18**:71-80.
6. Gillois M. La relation d'identité en génétique. *Ann Inst Henri Poincare B* **2** :1-94
7. Glosary of Genetic Terms, National Human Genome Research Institute http://www.genome.gov/glossary.cfm?key=pedigree
8. Harris DL. Genotypic covariances between inbred relatives. *Genetics* **50**: 1319-1348.
9. http://www.clevelandclinic.org/registries/
10. Jacquard A. The Genetic Structure of Populations. Springer-Verlag, New York, 1974.
11. Jacquard A. Logique du calcul des coefficients d'identite entre deux individuals. *Population* (Paris), 1966 ; **21**: 751-776.
12. Karigl G. A recursive algorithm for the calculation of identity coefficients. *Ann Hum Genet* 1981; 45:299–305.
13. Lange K. Mathematical and Statistical Methods for Genetic Analysis. Springer-Verlag, NY. 2002.
14. Malecot G. Les mathématique de l'hérédité, Masson, Pairs. Translated edition: The Mathematics of Heredity, Freeman, San Francisco, 1969.
15. Pedigree and Population Resource: Utah Population Database. http://www.hci.utah.edu/groups/ppr/
16. Sheng L, Ozsoyoglu ZM, Ozsoyoglu G. A Graph Query Language and Its Query Processing. In *Proceedings of ICDE Conference*, 1999.
17. Wright S. Coefficients of Inbreeding and Relationship. *The American Naturalist*, Vol. 56, No. 645, 1922.