

# On Counting the Number of Consistent Genotype Assignments for Pedigrees

Jiří Srba\*

**BRICS\*\***, Department of Computer Science, Aalborg University  
Fredrik Bajersvej 7B, 9220 Aalborg East, Denmark  
`srba@brics.dk`

**Abstract.** Consistency checking of genotype information in pedigrees plays an important role in genetic analysis and for complex pedigrees the computational complexity is critical. We present here a detailed complexity analysis for the problem of counting the number of complete consistent genotype assignments. Our main result is a polynomial time algorithm for counting the number of complete consistent assignments for non-looping pedigrees. We further classify pedigrees according to a number of natural parameters like the number of generations, the number of children per individual and the cardinality of the set of alleles. We show that even if we assume all these parameters as bounded by reasonably small constants, the counting problem becomes computationally hard ( $\#P$ -complete) for looping pedigrees. The border line for counting problems computable in polynomial time (i.e. belonging to the class FP) and  $\#P$ -hard problems is completed by showing that even for general pedigrees with unlimited number of generations and alleles but with at most one child per individual and for pedigrees with at most two generations and two children per individual the counting problem is in FP.

## 1 Introduction

Pedigrees are fundamental structures used in genetics. A pedigree describes family relations among generations of individuals. Genealogists study pedigrees in connection with the genotype information associated to the individuals at a particular locus. A genotype of a given individual is a pair of alleles in its genome (allele is one of the possible forms a gene may have). Due to different reasons, for a given pedigree the genotype information of its individuals can be known only partially. In order to complete the missing genotype information or to filter out erroneous input data, genealogists need to verify that the given partial information is consistent with the classic Mendelian laws of inheritance (see e.g. [5]), which means that every individual in the pedigree has to inherit exactly one

---

\* The author is supported in part by the research center ITI, project No. 1M0021620808

\*\* **B**asic **R**esearch in **C**omputer **S**cience,  
Centre of the Danish National Research Foundation.

allele from each of its parents. This process is called *consistency checking* and as argued in [9], in many real-life cases a manual consistency check is very difficult, time-consuming and sometimes unsuccessful. For an accessible overview of further biological aspects we refer the reader to [1].

To the best of our knowledge only algorithmic issues of consistency and likelihood checking for pedigrees have been studied in the literature so far (see e.g. [6, 10, 12, 2, 1]). In this paper we shall focus on a more general problem of *counting* the total number of complete genotype assignments consistent with the input data. This approach can provide a deeper insight and generalize the algorithms already developed for pure consistency checking. Moreover, knowing the total number of complete genotype assignments consistent with the input data can answer several additional questions. For example the fact that the number of assignments is 1 tells us that the missing information can be uniquely reconstructed from the available data. On the other hand, knowing that there are too many possibilities how to interpret the input data indicates that more genotype sampling is needed in order to reduce uncertainty.

*Our contribution.* We introduce characterization of pedigrees according to a number of natural parameters that describe their shapes. Apart for the standard notion of looping/non-looping pedigrees we further distinguish the number of generations, number of children per individual and the cardinality of the set of alleles. We describe a polynomial time algorithm that counts the number of complete genotype information for a given partial genotype data in non-looping pedigrees. We use this result to show that the counting problems for general pedigrees with at most 2 generations and 2 children per individual and for pedigrees with at most 1 child per individual are also solvable in polynomial deterministic time. We complete the results by demonstrating two parsimonious reductions (i.e. reductions that preserve the number of solutions) from #Bpos-2Sat to the counting problems for pedigrees with (i) 3 generations, 2 children per individual and 2 alleles, and (ii) 2 generations, 3 children per individual and 2 alleles. Together with an obvious containment in #P this proves #P-completeness of the problems.

*Related work.* For the case of pure consistency checking the following results are known. The problem for non-looping pedigrees is decidable in polynomial time using a *genotype elimination* algorithm proposed by Lange and Goradia [7] and further optimized and extended by O’Connell and Weeks [10], and Du and Hoeschele [2]. For general pedigrees there is a recent work by Aceto et al. [1] showing that consistency checking is NP-complete for pedigrees with marriage loops. They prove the result by reduction from 3SAT, however, their reduction is not parsimonious (does not preserve the number of solutions). It also works only for pedigrees with at least 5 generations, 3 children per individual and 3 alleles but under the assumption that there is either a complete or no knowledge about the genotype of single individuals. Another related result is NP-completeness of marginal probability and maximum likelihood by Piccolboni and Gusfield [12].

As discussed in [1], although the problems are closely connected to consistency checking, they cannot be used to imply hardness results for our problem.

*Full version will appear as a technical report in BRICS research series.*

## 2 Basic Definitions

### 2.1 Pedigrees and Genotype Information

In order to reason about pedigrees and the genotype information that they contain, we need to introduce a formal model. Several formalizations of the notion of pedigree have been presented in the literature on computational genetics (see e.g. [1, 8, 12]). The definition that we provide is equivalent to the ones mentioned above.

**Definition 1 (Pedigree).** A pedigree is a triple  $P = (M, F, \phi)$  where

- $M$  and  $F$  are finite disjoint sets of male, resp. female, individuals,
- $\phi: M \times F \rightarrow 2^{(M \cup F)}$  is a function called family function satisfying:
  1.  $\phi(f) \cap \phi(f') = \emptyset$  for all  $f, f' \in M \times F$  such that  $f \neq f'$ ,
  2. the transitive closure of the parental relation  $\prec \subseteq (M \cup F) \times (M \cup F)$  is irreflexive, where  $\prec$  is defined by  $u \prec v$  iff there is a  $w \in M \cup F$  such that  $u \in \phi(v, w)$  if  $v \in M$ , or  $u \in \phi(w, v)$  if  $v \in F$ .

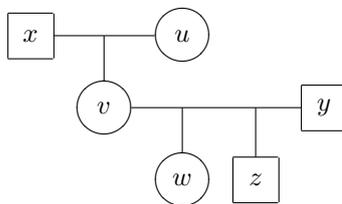
We define a set of families in  $P$  given by  $\phi$  as  $\mathcal{F}(\phi) \stackrel{\text{def}}{=} \{f \in M \times F \mid \phi(f) \neq \emptyset\}$ .

Let us also define  $p(f) \stackrel{\text{def}}{=} \{u, v\}$  for any family  $f = (u, v) \in \mathcal{F}(\phi)$ ; we call  $u$  and  $v$  the parents in the family  $f$ .

Here is an informal explanation of the definition. Given a male  $u \in M$  and a female  $v \in F$ ,  $\phi(u, v)$  is the set of all children they have. Condition 1. says that every child belongs to exactly one family and condition 2. guarantees that no individual can be its own ancestor.

The maximal elements from  $M \cup F$  w.r.t.  $\prec$  are called *founders* of the pedigree. Individuals that are not founders are called *non-founders*. The length of a longest chain (counting the number of nodes) w.r.t.  $\prec$  is called the number of *generations*. The *set of children* of an individual  $u \in M \cup F$  is defined by  $\cup_{f \in \mathcal{F}(\phi), u \in p(f)} \phi(f)$  and the number of *children per individual* is the largest cardinality of this set over all individuals in the pedigree.

*Example 1.* Let  $M \stackrel{\text{def}}{=} \{x, y, z\}$  and  $F \stackrel{\text{def}}{=} \{u, v, w\}$ . We define a pedigree  $P$  by  $\phi(x, u) \stackrel{\text{def}}{=} \{v\}$  and  $\phi(y, v) \stackrel{\text{def}}{=} \{w, z\}$ . In all other cases  $\phi(-, -) \stackrel{\text{def}}{=} \emptyset$ . This is graphically depicted as follows (male individuals are represented by squares, female individuals by circles and families with parental relation by lines).



The founders of the pedigree are the individuals  $x$ ,  $u$  and  $y$ . The pedigree has 3 generations and 2 children per individual.  $\square$

For each individual  $u \in M \cup F$  we define a *community*  $C(u)$  as a collection of all families where  $u$  is a parent, i.e.,  $C(u) \stackrel{\text{def}}{=} \{f \in \mathcal{F}(\phi) \mid u \in p(f)\}$ . *Maximal community size* is the largest cardinality over all communities in the pedigree, i.e.,  $\max_{u \in M \cup F} |C(u)|$ .

*Remark 1.* In any pedigree, the number of children per individual is at least the maximal community size.

We shall now introduce formal definitions of a mating graph and (non-)looping pedigrees. The following definitions are equivalent to the ones in [10] and [1].

**Definition 2 (Mating Graph, Connected Pedigrees).** Let  $P = (M, F, \phi)$  be a pedigree. We define an undirected bipartite graph  $G(P) \stackrel{\text{def}}{=} (M \cup F, \mathcal{F}(\phi), \leftrightarrow)$ , also called the mating graph of  $P$ , by stating that for all  $u \in M \cup F$  and for all  $f \in \mathcal{F}(\phi)$  we have  $\{u, f\} \in \leftrightarrow$  iff  $u \in p(f)$ , or  $u \in \phi(f)$ . We say that a pedigree  $P$  is connected iff  $G(P)$  is a connected graph.

From now on we shall consider only non-empty and connected pedigrees. All results presented in the paper can be extended also to unconnected pedigrees in a straightforward manner.

**Definition 3 ((Non-)Looping Pedigree).** We say that a pedigree  $P$  is looping if there is a loop in the mating graph  $G(P)$ . Otherwise we call  $P$  a non-looping pedigree.

Consistency checking of a pedigree is based on its associated genotype information; intuitively, the pedigree defines the structure of the family relationships that are being modelled, and the genotype information is the data which must be consistent with the structure. Let  $\mathcal{A}$  be a finite and non-empty set of *alleles*. A particular genotype information associated to every individual is represented by an element from  $\mathcal{A}^2$  modulo the least equivalence on  $\mathcal{A}^2$  satisfying  $xy \equiv yx$  (the order of alleles in a genotype does not play any role).

**Definition 4 ((Partial) Genotype Information).** Let  $P = (M, F, \phi)$  be a pedigree. A (partial) genotype information for  $P$  is a function  $\mathcal{G} : M \cup F \rightarrow 2^{\mathcal{A}^2}$  that associates a set of possible genotype data to the individuals in the pedigree, s.t.  $\mathcal{G}(u) \neq \emptyset$  for all  $u \in M \cup F$ .

The intuition is that  $\mathcal{G}(u) = \{AB\}$  means that the genotype of the individual  $u$  is known to be exactly  $AB$ . If e.g.  $\mathcal{G}(u) = \{AB, AC\}$  then we have only a partial information about the individual  $u$  (we know that its genotype can be either  $AB$  or  $AC$ ). If only one allele (let us say  $A$ ) from the pair is known then we model it by  $\mathcal{G}(u) = \{AX \mid X \in \mathcal{A}\}$ . In case that nothing is known about the genotype of  $u$  then  $\mathcal{G}(u) = \mathcal{A}^2$ .

**Definition 5 (Specialization).** Let  $\mathcal{G}$  and  $\mathcal{G}'$  be two partial genotype information. We say that  $\mathcal{G}$  specializes into  $\mathcal{G}'$  iff  $\mathcal{G}'(u) \subseteq \mathcal{G}(u)$  for all  $u \in M \cup F$ .

**Definition 6 (Complete Genotype Information).** A genotype information  $\mathcal{G}$  is called complete if  $|\mathcal{G}(u)| = 1$  for every  $u \in M \cup F$ , i.e., every individual is assigned exactly one genotype.

Verifying the consistency for a specific gene amounts to checking whether the pedigree and the genotype information are consistent according to the Mendelian law of segregation (see e.g. [5]). The law of segregation implicitly defines the following constraint on consistent genotype assignments:

*Each individual inherits exactly one allele from both of its parents.*

Our order of business will now be to formalize this constraint, and what it means that a genotype information is consistent with respect to a pedigree. Given two genotypes  $AB$  and  $CD$ , we define  $zygote(AB, CD) \stackrel{\text{def}}{=} \{AC, AD, BC, BD\}$  as the set of all possible combinations of the given genotypes. Note that due to the commutativity introduced on  $\mathcal{A}^2$  we get that e.g.  $zygote(AB, AB) = \{AA, AB, BB\}$  and  $zygote(AA, AB) = \{AA, AB\}$ .

**Definition 7 (Consistent Genotype Information).**

1. A complete genotype information  $\mathcal{G}$  is consistent if for all  $(u, v) \in \mathcal{F}(\phi)$  such that  $\mathcal{G}(u) = \{AB\}$ ,  $\mathcal{G}(v) = \{CD\}$  for some  $A, B, C, D \in \mathcal{A}$  and for all  $w \in \phi(u, v)$  it is the case that  $\mathcal{G}(w) \subseteq zygote(AB, CD)$ .
2. A partial genotype information  $\mathcal{G}$  is consistent if there is a complete consistent genotype information  $\mathcal{G}'$  such that  $\mathcal{G}$  specializes into  $\mathcal{G}'$ .

Let  $P$  be a pedigree and  $\mathcal{G}$  a genotype information for  $P$ . By  $\#(P, \mathcal{G})$  we denote the number of complete and consistent genotype information into which  $\mathcal{G}$  specializes (or simply the number of solutions). A natural algorithmic problem is that of computing  $\#(P, \mathcal{G})$  and we call it the *counting problem* for a pedigree  $P$  and a genotype information  $\mathcal{G}$ .

## 2.2 Counting Problems and Complexity Classes FP and #P

The complexity classes FP and #P for counting problems play a similar role as the complexity classes P and NP in case of decision problems. Let  $R$  be a polynomially balanced ( $(x, y) \in R$  implies  $|y| \leq |x|^k$  for some constant  $k > 0$ ) and polynomial time decidable binary relation [11]. The counting problem #R is for a given input  $x$  to count how many different  $y$  there are such that  $(x, y) \in R$ . To provide the answer to such a problem is generally considered as a hard computational task. #P is the class of all such problems. Alternatively, #P can be defined as the class of functions that can be computed by counting the number of accepting paths of a polynomial time nondeterministic Turing machine. On the other hand, the complexity class FP is the class of functions that are computable by a deterministic Turing machine in polynomial time (also

called the class of feasible functions, i.e., those solvable by computers). We can easily notice that  $\text{FP} \subseteq \#\text{P}$  and it is widely conjectured that the inclusion is strict.

In order to show  $\#\text{P}$ -hardness of a counting problem, we often use the notion of *parsimonious reduction*. Let  $\#\text{R}$  and  $\#\text{S}$  be two counting problems. We say that there is a parsimonious reduction from  $\#\text{R}$  to  $\#\text{S}$  if there is a polynomial time transformation  $f$  from the instances of  $\#\text{R}$  to the instances of  $\#\text{S}$  which preserves the number of solutions, i.e., for all  $x$  we have that  $|\{y \mid (x, y) \in R\}| = |\{y \mid (f(x), y) \in S\}|$ .

*Remark 2.* This notion of reduction is little too restrictive so sometimes one defines that  $\#\text{R}$  reduces to  $\#\text{S}$  if there is a polynomial time algorithm for  $\#\text{R}$  given an oracle that solves  $\#\text{S}$ . Nevertheless, all the reductions presented in this paper are parsimonious.

As noted in [11] p. 439: “Even in cases in which the decision problem is polynomial, counting the solutions may be highly nontrivial.” An example of such a problem is e.g. counting the number of perfect matchings in a bipartite graph. This is a  $\#\text{P}$ -complete problem, while the decision variant of the problem is in  $\text{P}$ . On the other hand, showing that a counting problem is in  $\text{FP}$  immediately gives a polynomial time algorithm for the corresponding decision problem. Hence proving that the counting problem for a certain subclass of pedigrees (e.g. non-looping pedigrees) is in  $\text{FP}$  provides a stronger claim than only showing that the decision problem of consistency checking is in  $\text{P}$ .

### 3 Pedigrees with the Counting Problem in $\text{FP}$

In this section we demonstrate that for an arbitrary non-looping pedigree  $P$  and a given genotype information  $\mathcal{G}$ , the number  $\#(P, \mathcal{G})$  can be computed in polynomial time on a deterministic Turing machine. Hence we generalize the result by Lange and Goradia [7] where they showed that the decision version of the problem is solvable in polynomial time using a genotype elimination algorithm. In our approach, we provide a different solution which exploits dynamic programming and enables us to count (and list if necessary) the total number of complete and consistent genotype information. We also show how to count in polynomial time the number of consistent pedigree assignments for pedigrees with 2 generations and maximal community size 2, and for pedigrees with at most one child per individual.

Let us consider a pedigree  $P = (M, F, \phi)$  with a partial genotype information  $\mathcal{G}$  over the alleles from  $\mathcal{A}$ . When counting  $\#(P, \mathcal{G})$  we will use the techniques of dynamic programming and store the intermediate results in the following table.

$$T : (M \cup F) \times \mathcal{A}^2 \rightarrow \mathbb{N}$$

We shall often denote a table element  $T(u, XY)$  where  $u \in M \cup F$  and  $XY \in \mathcal{A}^2$  by  $T^u(XY)$ . The main idea of the algorithm is that we shall process all families

(and the corresponding individuals) of the pedigree in a particular order such that the number assigned to the table position  $T^u(XY)$  stands for the number of solutions in a subpedigree that was already processed and is connected to  $u$ , all under the assumption that the genotype information of  $u$  is fixed to  $XY$ .

The procedure **initialize** in Figure 1 initializes the table  $T$  for all individuals from  $M \cup F$ .

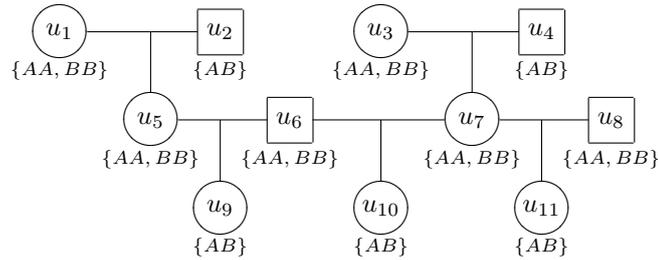
Let  $(x, y) \in \mathcal{F}(\phi)$ ,  $u \in M \cup F$  s.t.  $u \leftrightarrow (x, y)$  and  $XY \in \mathcal{G}(u)$ . The function **update** in Figure 1 returns the number of solutions in the already processed subpedigree connected (in the mating graph) to the individual  $u$ , under the assumption that the genotype information for  $u$  is fixed to  $XY$  and that the table  $T^v$  is fully computed for all the individuals  $v$  connected to the family  $(x, y)$  except for  $u$ .

Finally, the function **count** in Figure 1 computes the number  $\#(P, \mathcal{G})$  where the notion of a mediator for  $f \in \mathcal{F}(\phi)$  and  $Z \subseteq \mathcal{F}(\phi)$  is defined as follows: an individual  $u \in M \cup F$  is a *mediator* for  $f$  w.r.t.  $Z$  iff  $u \leftrightarrow f$  and there is some  $f' \in Z$  such that  $f \neq f'$  and  $u \leftrightarrow f'$ . In other words a mediator is an individual that connects two different families in the mating graph. The function **count** first initializes the table  $T$  to its initial values and creates a set  $Z$ , which represents the set of families to be processed. It then removes the families from  $Z$  one by one in a particular order which ensures that the table  $T^u$  for a mediator  $u$  can be easily computed. Finally, when  $Z$  contains only one family, the final number of solutions is computed and returned.

**Theorem 1.** *The counting problem for non-looping pedigrees is in FP.*

*Proof. (Sketch)* It is easy to see that the algorithm runs in polynomial time. We have to argue that for a given pedigree  $P$  and a genotype information  $\mathcal{G}$  the function **count**( $P, \mathcal{G}$ ) returns the number  $\#(P, \mathcal{G})$ . The requirement that  $P$  is non-looping (and connected) ensures that we can always select a family  $f \in Z$  with exactly one mediator  $u$  with respect to  $Z$ . Moreover, whenever a value is assigned to  $T^u(XY)$  the following assertion holds: “ $T^u(XY)$  is the number of solutions in the subpedigree generated by  $u$  and the families in  $\mathcal{F}(\phi) \setminus Z$  (together with their children) that are connected to  $u$  in the mating graph”.  $\square$

*Example 2.* We shall demonstrate the algorithm for counting the number of solutions on the following pedigree.



The pedigree consists of 11 individuals  $u_1, \dots, u_{11}$  and 5 families  $(u_2, u_1)$ ,  $(u_4, u_3)$ ,  $(u_6, u_5)$ ,  $(u_6, u_7)$  and  $(u_8, u_7)$ . The genotype information for each individual is depicted in the picture (e.g.  $\mathcal{G}(u_5) = \{AA, BB\}$  and  $\mathcal{G}(u_2) = \{AB\}$ ).

```

initialize =
for all  $v \in M \cup F$  do
  for all  $XY \in \mathcal{A}^2$  do
     $T^v(XY) := \begin{cases} 1 & \text{if } XY \in \mathcal{G}(v) \\ 0 & \text{otherwise} \end{cases}$ 
  end for
end for

update $((x, y), u, XY)$ :int =
if  $u = x$  then
  ***  $u$  is a male parent in the family ***
  let  $\{w_1, \dots, w_\ell\} = \phi(x, y)$  be all children in the family  $(x, y)$ 
  return  $\sum_{\substack{AB \in \mathcal{G}(y) \\ X_1Y_1, \dots, X_\ell Y_\ell \in \text{zygote}(XY, AB)}} T^y(AB) \cdot T^{w_1}(X_1Y_1) \cdot \dots \cdot T^{w_\ell}(X_\ell Y_\ell)$ 
else if  $u = y$  then
  ***  $u$  is a female parent in the family ***
  let  $\{w_1, \dots, w_\ell\} = \phi(x, y)$  be all children in the family  $(x, y)$ 
  return  $\sum_{\substack{AB \in \mathcal{G}(x) \\ X_1Y_1, \dots, X_\ell Y_\ell \in \text{zygote}(AB, XY)}} T^x(AB) \cdot T^{w_1}(X_1Y_1) \cdot \dots \cdot T^{w_\ell}(X_\ell Y_\ell)$ 
else
  ***  $u \in \phi(x, y)$  is a child in the family ***
  let  $\{w_1, \dots, w_\ell\} = \phi(x, y) \setminus \{u\}$  be all the children except for  $u$ 
  return  $\sum_{\substack{AB \in \mathcal{G}(x) \text{ and } CD \in \mathcal{G}(y) \\ \text{s.t. } XY \in \text{zygote}(AB, CD) \\ X_1Y_1, \dots, X_\ell Y_\ell \in \text{zygote}(AB, CD)}} T^x(AB) \cdot T^y(CD) \cdot T^{w_1}(X_1Y_1) \cdot \dots \cdot T^{w_\ell}(X_\ell Y_\ell)$ 
end if

count $(P = (M, F, \phi), \mathcal{G} : M \cup F \rightarrow 2^{\mathcal{A}^2})$ :int =
 $Z := \mathcal{F}(\phi)$ 
initialize
while  $|Z| > 1$  do
  select  $f \in Z$  s.t.  $f$  has exactly one mediator  $u$  w.r.t.  $Z$ 
  for all  $XY \in \mathcal{G}(u)$  do
     $T^u(XY) := T^u(XY) \cdot \text{update}(f, u, XY)$ 
  end for
   $Z := Z \setminus \{f\}$ 
end while
let  $f = (x, y)$  where  $\{f\} = Z$ ;
let  $\{w_1, \dots, w_\ell\} = \phi(x, y)$  be all children in the family  $(x, y)$ 
return  $\sum_{\substack{AB \in \mathcal{G}(x) \text{ and } CD \in \mathcal{G}(y) \\ X_1Y_1, \dots, X_\ell Y_\ell \in \text{zygote}(AB, CD)}} T^x(AB) \cdot T^y(CD) \cdot T^{w_1}(X_1Y_1) \cdot \dots \cdot T^{w_\ell}(X_\ell Y_\ell)$ 

```

**Fig. 1.** Algorithm for computing the number  $\#(P, \mathcal{G})$

During the call of the function **count** we first for all individuals initialize the table  $T$  to either 0 and 1 according to the given genotype information. Next we start eliminating the families in the pedigree. Let us assume that the first selected family in the while-loop is  $(u_2, u_1)$  and the mediator  $u$  is equal to  $u_5$ . As  $u_5$  is a child in the family  $(u_2, u_1)$  we compute  $T^{u_5}(AA) = 1 \cdot 1 = 1$  and  $T^{u_5}(BB) = 1 \cdot 1 = 1$  according to the third case in the function **update**. The family  $(u_2, u_1)$  is then removed from  $Z$ . Next assume that the second selected family is  $(u_6, u_5)$  with a mediator  $u_6$  and we compute  $T^{u_6}(AA) = T^{u_6}(BB) = 1 \cdot 1 = 1$  by using the first case in the function **update**. The family  $(u_6, u_5)$  is removed from  $Z$ . Assume that the third selected family is  $(u_4, u_3)$  where  $u_7$  is the mediator and we compute  $T^{u_7}(AA) = T^{u_7}(BB) = 1 \cdot 1 = 1$ . The family  $(u_4, u_3)$  is removed from  $Z$ . Now we can select e.g. the family  $(u_8, u_7)$  with the mediator  $u_7$  and compute  $T^{u_7}(AA) = T^{u_7}(BB) = 1 \cdot 1 = 1$  while removing  $(u_8, u_7)$  from  $Z$ . Finally, only the family  $(u_6, u_7)$  remains in  $Z$  and we return the final value

$$T^{u_6}(AA) \cdot T^{u_7}(BB) \cdot T^{u_{10}}(AB) + T^{u_6}(BB) \cdot T^{u_7}(AA) \cdot T^{u_{10}}(AB) = 1 + 1 = 2.$$

Indeed, there are exactly two possibilities for the assignment of a genotype to  $u_1$  and this uniquely determines the assignments in the rest of the pedigree. In particular, one can see that  $u_3$  has to be assigned exactly the same genotype as  $u_1$  in order to preserve consistency.  $\square$

**Theorem 2.** *If a pedigree  $P = (M, F, \phi)$  has at most one child per individual then it is non-looping.*

**Corollary 1.** *The counting problem for pedigrees with at most one child per individual is in FP.*

Let us now consider pedigrees with 2 generations only and the maximal community size at most 2. We can demonstrate that the counting problem for this subclass is also in FP by applying the observations about “loop-breakers” from [4].

**Theorem 3.** *The counting problem for pedigrees with 2 generations and maximal community size 2 (and an arbitrary number of alleles) is in FP.*

**Corollary 2.** *The counting problem for pedigrees with 2 generations and at most two children per individual (and an arbitrary number of alleles) is in FP.*

*Proof.* Directly from Theorem 3 and Remark 1.  $\square$

## 4 Pedigrees with #P-Complete Counting Problem

In this section we shall argue that the counting problems in all other pedigrees except for those considered in Section 3 are computationally hard. We will demonstrate #P-hardness (with respect to parsimonious reduction) for pedigrees with 3 generations, 2 children per individual and 2 alleles, and for pedigrees with

2 generations, 3 children per individual and 2 alleles. Even for general pedigrees one can easily see that the problems are in  $\#P$ , which together with the hardness results implies  $\#P$ -completeness. This completes the full picture of the computational complexity of counting problems for pedigrees.

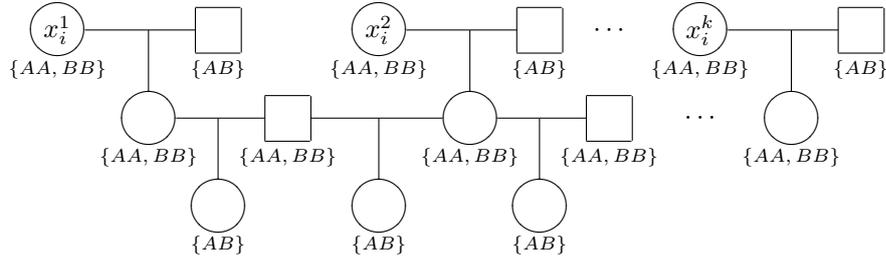
Let us consider the  $\#Bpos\text{-}2Sat$  counting problem [3]. We are given two disjoint sets of variables  $\{x_1, \dots, x_n\}$  and  $\{y_1, \dots, y_m\}$  and a formula  $C_1 \wedge C_2 \wedge \dots \wedge C_k$  where for every  $\ell$ ,  $1 \leq \ell \leq k$ , the clause  $C_\ell$  is of the form  $x_i \vee y_j$  such that  $1 \leq i \leq n$  and  $1 \leq j \leq m$ . Counting the number of satisfying truth assignments of such a formula is a  $\#P$ -complete problem [3]. Note that the corresponding decision problem is trivial as any  $\#Bpos\text{-}2Sat$  formula is satisfiable.

We shall reduce  $\#Bpos\text{-}2Sat$  to the counting problem for pedigrees of particular shapes such that the reduction preserves the number of solutions (i.e. it is a parsimonious reduction). In our reductions we shall use only two alleles  $\mathcal{A} \stackrel{\text{def}}{=} \{A, B\}$  such that  $AA$  represents *true* and  $BB$  represents *false*.

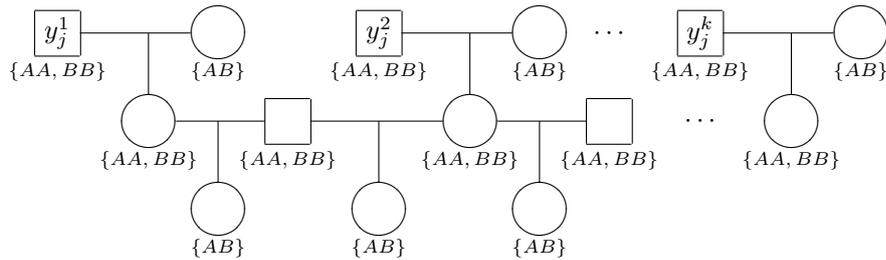
**Theorem 4.** *The counting problem for pedigrees with 3 generations, 2 children per individual and 2 alleles is  $\#P$ -complete.*

*Proof. (Sketch)* Containment in  $\#P$  is easy. We shall argue for  $\#P$ -hardness of the problem. Let  $C_1 \wedge C_2 \wedge \dots \wedge C_k$  be a given instance of  $\#Bpos\text{-}2Sat$ . We shall construct a pedigree  $P$  as follows.

First, for every variable  $x_i$ ,  $1 \leq i \leq n$ , we create  $k$  of its copies by constructing the following pedigree part.

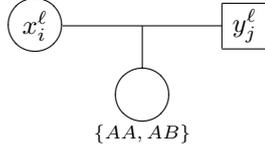


Next we create a similar structure for every variable  $y_j$ ,  $1 \leq j \leq m$ .



As argued in Example 2, we can now see that the pedigrees for  $x_i$  (resp.  $y_j$ ) have exactly two solutions so that  $x_i^1, \dots, x_i^k$  (resp.  $y_j^1, \dots, y_j^k$ ) can simultaneously take the assignment  $AA$  or  $BB$ , hence representing the truth value true or false.

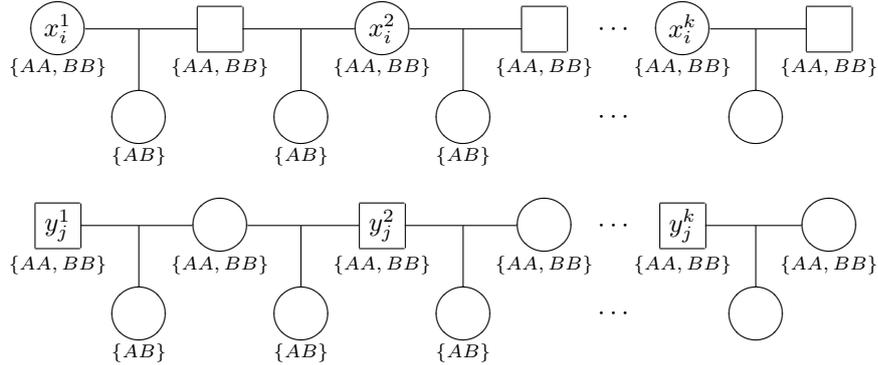
Now for every  $\ell$ ,  $1 \leq \ell \leq k$ , such that  $C_\ell \equiv x_i \vee y_j$  we will add the following pedigree part where the individuals  $x_i^\ell$  and  $y_j^\ell$  are identified with the corresponding nodes in the pedigrees above.



We can now easily verify that at least one of  $x_i^\ell$  and  $y_j^\ell$  has to be set to true (i.e. takes the value  $AA$ ) in order to achieve a consistent assignment. It is a routine exercise to check that the number of satisfying truth assignments of the formula is the same as the number of complete genotype information for the constructed pedigree. The construction ensures that the pedigree has 3 generations, at most 2 children per individual and uses only 2 alleles.  $\square$

**Theorem 5.** *The counting problem for pedigrees with 2 generations, 3 children per individual and 2 alleles is #P-complete.*

*Proof. (Sketch)* As in the previous proof, the reduction goes from #Bpos-2Sat. We modify the way in which we generate the truth assignments. We shall use only 2 generation pedigrees at the expense of 3 children per individual.



It is easy to see that  $x_i^1, \dots, x_i^k$  (resp.  $y_j^1, \dots, y_j^k$ ) can simultaneously take the assignment  $AA$  or  $BB$  and after adding the pedigree parts for all clauses  $C_\ell$  as in the construction above, the reduction preserves the number of solutions. The final pedigree has 2 generations and at most 3 children per individual. As before, the containment in #P is easy.  $\square$

## 5 Conclusion

We have studied counting problems for genotype assignments in pedigrees and found a delicate borderline between tractable and intractable instances of the problem. The following table summarizes the main results achieved in the paper.

type	# generations	# children	# alleles	complexity
non-looping	$\infty$	$\infty$	$\infty$	in FP (Thm. 1)
looping	$\infty$	$\infty$	1	in FP (trivial)
looping	1	$\infty$	$\infty$	in FP (trivial)
looping	$\infty$	1	$\infty$	in FP (Cor. 1)
looping	2	2	$\infty$	in FP (Cor. 2)
looping	3	2	2	#P-complete (Thm. 4)
looping	2	3	2	#P-complete (Thm. 5)

The table provides a complete characterization of the computational complexity of counting problems with respect to the selected parameters. Moreover, the hardness results use only marriage loops [12] and contain no inbreeding loops.

In [1] it is shown that consistency checking for general pedigrees with 2 alleles only is in P, provided that for each individual we either know precisely his/her genotype information or we know nothing at all, i.e., for every individual  $u$  we have either  $|\mathcal{G}(u)| = 1$  or  $\mathcal{G}(u) = \mathcal{A}^2$ . It would be interesting to see whether the counting problem is in FP under this restriction. The future research will also focus on investigating possible ways of tackling the counting problem for pedigrees with loops (here some ideas from [10] seem to be applicable) and on the complete complexity characterization of the consistency checking problems.

## References

1. L. Aceto, J.A. Hansen, A. Ingólfssdóttir, J. Johnsen, and J. Knudsen. The complexity of checking consistency of pedigree information and related problems. *J. of Computer Science and Technology*, 19(1):42–59, 2004.
2. F.X. Dua and I. Hoeschele. A note on algorithms for genotype and allele elimination in complex pedigrees with incomplete genotype data. *Genetics*, 156:2051–2062, 2000.
3. M.O. Ball and J.S. Provan. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. on Comp.*, 12(4):777–788, 1983.
4. K. Lange and R.C. Elston. Extensions to pedigree analysis i. likelihood calculations for simple and complex pedigrees. *Human Heredity*, 25(2):95–105, 1975.
5. William S. Klug and Michael R. Cummings. *Concepts of Genetics*. Prentice Hall, 5th edition, 1997.
6. K. Lange and T. Goradia. An algorithm for automatic genotype elimination. *American J. of Human Genetics*, 40:250–256, 1987.
7. Kenneth Lange and Tushar Madhu Goradia. An algorithm for automatic genotype elimination. *American J. of Human Genetics*, 40:250–256, 1987.
8. Jing Li and Tao Jiang. Efficient rule-based haplotyping algorithms for pedigree data. In *Proceedings of RECOMB’03*, pages 197–206. ACM, 2003.
9. J.R. O’Connell and D.E. Weeks. Pedcheck: A program for identification of genotype incompatibilities in linkage analysis. *Am. J. of Human Genetics*, 63:259–266, 1998.
10. J.R. O’Connell and D.E. Weeks. An optimal algorithm for automatic genotype elimination. *American J. of Human Genetics*, 65:1733–1740, 1999.
11. Ch.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
12. A. Piccolboni and D. Gusfield. On the complexity of fundamental computational problems in pedigree analysis. *J. of Computational Biology*, 10(5):763–773, 2003.