Error Detection in Pedigrees Using Satisfiability-Based Approaches

David Martins

Instituto Superior Técnico and INESC-ID

Abstract. In bioinformatics, a pedigree is a structure that allows the monitoring of individuals and their family relations. With pedigrees, it is possible to track the flow of genetic information from parents to offspring. However, with the high throughput of genotyping techniques, the occurrence of errors in pedigrees is becoming a problem of increasingly greater importance and consistency becomes something that geneticists must deal with.

The pedigree consistency checking problem consists in the problem of verifying if a pedigree (a genealogical tree with associated genotypes) is consistent under the Mendelian laws of inheritance.

This pedigree consistency check is a well-known NP-complete problem and there are two main types of approaches that attempt to tackle it: statistical and combinatorial ones.

The purpose of this work is to try to develop two types of combinatorial approaches based on the modelling of maximum constraint satisfaction problems (CSP) into certain types of maximum satisfiability (Max-SAT) encodings (minimal support encoding and k-AC encoding) and attempt to measure their efficiency against distinct types of pedigree instances.

1 Introduction

One of today's most challenging problems in the field of genetics consists in the correct relation between the genes in the human genome and some biological trait an individual may possess. Example of these traits range from blood type and eye color, to those that can predispose an individual for a particular disease.

One of the methods used to tackle this problem is named linkage analysis [1, 2] which is a well established statistical-based method that has already shown significant results: genes causing major diseases (e.g., Parkinson's disease, obesity and anxiety) have already been discovered using this technique. Linkage analysis works by creating maps that correlate specific genes and/or genetic markers of a determined experimental population. And the more information contained on those maps, that is, the denser these maps become, the higher is the probability that the information may contain errors.

A computational problem that is closely related to linkage analysis is that of consistency checking.

A pedigree describes the family relations amongst a set of individuals, and usually comes associated information on their genotypes, that is, on the pair of alleles at a locus in their genome.

Given a pedigree and information on the genotypes (of some) of the individuals in it, the aim of consistency checking is to determine whether these data are consistent with the classic Mendelian laws of inheritance.

There are a number of approaches that tackle the consistency checking problem, and they range from statistical methods to combinatorial ones. Of these two, combinatorial approaches will be the main focus of this work.

Regarding the combinatorial approaches, there are satisfiability (SAT) approaches [3], constraint satisfaction (CSP) approaches [4] and pseudo-boolean (PBO) approaches [5].

One of the objectives of this work, when concerning the combinatorial approaches, is to determine their efficiency against each other, when trying to tackle this problem, and verify which can be used in a particular situation.

For example, we will experiment some SAT encodings like the direct encoding and the support encoding [6] and their performance will be measured using randomly generated pedigree instances as well as real pedigree instances.

2 Pedigrees

A *pedigree* [7, 8] is a structure used to track the inheritance of genetic traits. Usually, a pedigree has some (even if incomplete) genotype information associated with it.

Technically, a pedigree can also be represented as a graph, where the nodes represent the individuals of a family and the arcs represent the relationships between them.

Figure 1 represents a simple pedigree which consists of three individuals. Individuals 1 and 2 are the parents of individual 3. Squares represent male individuals and circles represent female ones, and therefore individual 1 is the father of 3 and individual 2 is the mother. All individuals in this pedigree have associated genotype `AB`.

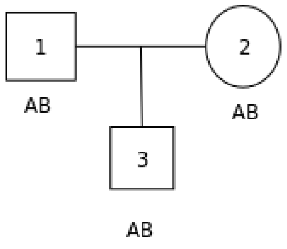

Fig. 1. Example of a simple pedigree with associated genotype.

A formal definition of *Pedigree* follows:

Definition 1 ***(Pedigree [7])*** *A pedigree consists of a 4-tuple* $\langle V,F,p,m\rangle$ *where:*

- V *is a finite, non-empty set of members of the pedigree,*
- $F \subseteq V$ *is the set of founders,*
- p*,*m*:* $V \setminus F \longrightarrow V$ *are the paternal and maternal functions, respectively, where*

$$p(V\backslash F) \cap m(V\backslash F) = \varnothing$$

  (that is, no individual in the pedigree can be both mother and father), and
- *the transitive closure of the binary relation obtained by the union of* p *and* m *is irreflexive (which means that a member of the pedigree is never its own ancestor).*

The set $N = V \setminus F$ *is usually referred as the set of non-founders in a pedigree and is always non-empty.*

Although a pedigree is acyclic, loops may still exist, which are classified in two types [8]: *inbreeding* and *marriage* loops. When two individuals who share a common ancestor mate and have an offspring, that creates an inbreeding loop; otherwise, it is a marriage loop. Marriage loops occur naturally in real pedigrees, whenever two siblings mate with the same individual.

The occurrence of loops is one of the main difficulties in pedigree analysis: the pedigree consistency checking problem is NP-complete for pedigrees [7] (considering only marriage loops), and some pedigree analysis problems such as the marginal-probability and the maximum-likelihood problem are NP-hard [8].

Figure 2 depicts an example of a looping pedigree. In this example, it is possible to witness two loops. One is due to inbreeding, and arises because individuals 4 and 5 mate, and have a common ancestor (individual 2). Another is a marriage loop which stems from individual 6 mating individuals 5 and 7, who are brothers. This figure has no genotypes associated with the individuals.

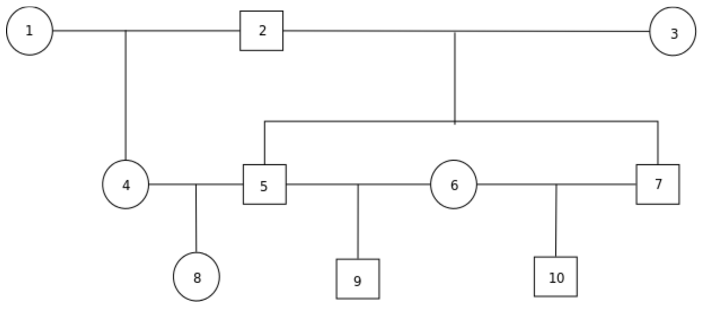

Fig. 2. Example of a looping pedigree.

A definition for *Genotype Information* and for *Consistent Genotype Information* follows:

Definition 2 ***(Genotype Information [7])*** *Let $P = \langle V,F,p,m \rangle$ be a pedigree. A genotype information for P is a partial function $G{:}V \hookrightarrow Two(A)$ that associates a genotype to (some of) the members of the pedigree. The domain, dom(G), of the function is referred to as the set of genotyped members of the pedigree. The genotype information G is complete if dom(G) = V (which means that every member V of the pedigree has an associated genotype).*

Let G and G' be two genotype information for the same pedigree. We say that G' extends G if dom(G) is included in dom(G') (i.e., dom(G)$\subseteq$dom(G')), and $\forall$ $g \in$ dom(G): G(g) = G'(g) (which means that G and G' coincide over dom(G)).

Definition 3 ***(Consistent Genotype Information [7])*** *Let $P = \langle V,F,p,m \rangle$ be a pedigree.*

1. *A complete genotype information G for P is consistent with P if, for each v $\in$ V:*
   (a) *if G(v) = {`A,B`}, then either `A` $\in$ G(p(v)) and `B` $\in$ G(m(v)), or `B` $\in$ G(p(v)) and `A` $\in$ G(m(v)).*
   (b) *if G(v) = {`A,A`}, then `A` $\in$ G(p(v)) $\wedge$ `A` $\in$ G(m(v)).*
2. *An incomplete genotype information for P is consistent with P if it can be extended to a complete and consistent genotype information for P.*

In practice, this means that each individual in the pedigree must inherit exactly one allele from each of its parents.

Figure 3 shows a pedigree with inconsistent and incomplete genotype information. The genotype information is incomplete because individual 4 has no associated genotype. The genotype information for this pedigree is also inconsistent because individual 7 could not have inherited allele `D` from any of its parents.

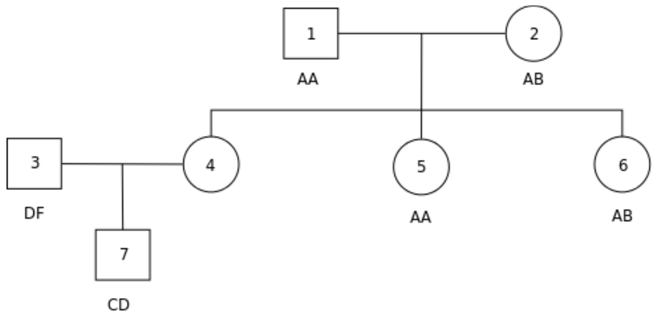

Fig. 3. Example of a pedigree with associated genotype information.

Most of the literature available ([9, 10]) refers the existence of two types of errors: *pedigree errors* and *genotyping errors.*

Pedigree errors occur due to the misidentification of individuals and their relationships [9, 10]. Examples of these include non-paternity, unidentified adoption and sample mix-ups.

As for genotyping errors, these occur mostly due to the misinterpretation of genotypes and data entry error. One can observe a genotyping error when the observed genotype does not correspond to the true underlying genetic information.

For the purposes of this work, only genotyping errors will be considered in all tested pedigrees. This means that it will be assumed that the structure of pedigrees is always correct, that is, there are no pedigree errors in the studied data in the following of this work. This approach has also been followed in [11].

3 Maximum Satisfiability and Pedigree Consistency Checking

In Partial Max-SAT, some clauses are called *non-relaxable* or *hard* clauses and the others are called *relaxable* or *soft* clauses. Solving a Partial Max-SAT instance consists of finding an assignment that satisfies all of the *hard* clauses and the maximum number of *soft* clauses.

When mapping pedigree structures it is first necessary to relate the characteristics found on them to the characteristics of partial Max-SAT instances.

Information on parental relations are mapped in the structure of a pedigree. Therefore that information needs to be mapped into partial Max-SAT. And considering that parental information is assumed to always be correct, there will exist a number of *hard clauses* in partial Max-SAT that describe each of the parental relations. Since those relations involve three individuals, the clauses will be ternary.

Also, a pedigree may come with genotype information on some (maybe all) individuals. For each individual that has a genotype associated to the pedigree, an unary clause is introduced in the partial Max-SAT instance. Since the consistency checking problem is based on the correct assignment of genetic information to each individual present in the pedigree, these clauses will be *soft*: this information may possibly be wrong.

The encodings from CSP to partial Max-SAT that were more extensively explored in this work are the *minimal support encodings* [6, 12] and the *k-AC support encoding* [13].

Both encodings share a number of clauses in common: *at-least-one* clauses, *at-most-one-clauses* and unary clauses. *At-least-one* and the *at-most-one* clauses are created in order to assign one genotype to one individual. These clauses will be *hard* clauses as they cannot be unsatisfied.

Unary clauses are created simply by indicating the SAT variable that takes the allowed domain value for that specific CSP value (individual). The result is a *soft* unary clause for each indicated variable. The pedigree consistency checking problem focuses on finding correct genotype assignments for individuals and the goal is to find the maximum number of correctly assigned individuals. Hence, unary clauses are soft.

For figure 1 the *at-least-one*, *at-most-one* and unary clauses will be:

at-least-one clauses

$$[x1_{AA} \vee x1_{AB} \vee x1_{BB}] \quad [x2_{AA} \vee x2_{AB} \vee x2_{BB}] \quad [x3_{AA} \vee x3_{AB} \vee x3_{BB}]$$

at-most-one clauses

$$\begin{array}{lll}
[\overline{x}1_{AA} \vee \overline{x}1_{AB}] & [\overline{x}1_{AA} \vee \overline{x}1_{BB}] & [\overline{x}1_{AB} \vee \overline{x}1_{BB}] \\
[\overline{x}2_{AA} \vee \overline{x}2_{AB}] & [\overline{x}2_{AA} \vee \overline{x}2_{BB}] & [\overline{x}2_{AB} \vee \overline{x}2_{AB}] \\
[\overline{x}3_{AA} \vee \overline{x}3_{AB}] & [\overline{x}3_{AA} \vee \overline{x}3_{BB}] & [\overline{x}3_{AB} \vee \overline{x}3_{BB}]
\end{array}$$

unary clauses

$$\begin{array}{lll}
(x1_{AB}) & (x2_{AB}) & (x3_{AB})
\end{array}$$

Then, the only difference between the minimal support encoding and the k-AC encoding will be in the way these encodings map ternary clauses (due to the occurrence of paternal relations in a pedigree between father, mother and child).

The minimal support encoding [6] is considered a variant of the support encoding [14]. The basis of this encoding starts with the observation that the support encoding contains unnecessary clauses. For a binary constraint C_k with scope $\{X, Y\}$, it is enough to add the support for either the values of X or the values of Y; it is not necessary to add a clause in each *direction*. This signifies that minimal support encoding only considers clauses of the type $father \wedge mother \rightarrow$ *child*.

The ternary clauses in minimal support encoding for figure 1 are:

minimal support clauses

$$\begin{array}{lll}
[x1_{AA} \wedge x2_{AA} \rightarrow x3_{AA}] & [x1_{AA} \wedge x2_{AB} \rightarrow x3_{AA} \vee x3_{AB}] & [x1_{AA} \wedge x2_{BB} \rightarrow x3_{AB}] \\
[x1_{AB} \wedge x2_{AA} \rightarrow x3_{AA} \vee x3_{AB}] & [x1_{AB} \wedge x2_{AB} \rightarrow x3_{AA} \vee x3_{AB} \vee x3_{BB}] & [x1_{AB} \wedge x2_{BB} \rightarrow x3_{AA} \vee x3_{BB}] \\
[x1_{BB} \wedge x2_{AA} \rightarrow x3_{AB}] & [x1_{BB} \wedge x2_{AB} \rightarrow x3_{AB} \vee x3_{BB}] & [x1_{BB} \wedge x2_{BB} \rightarrow x3_{BB}]
\end{array}$$

The k-AC encoding differs from the minimal support encoding due to the fact that k-AC maintains a form of arc consistency through unit propagation whereas minimal support does not [13]. In the k-AC encoding all directions of a clause are explicited. In the pedigree case, not only clauses of the type $father \wedge mother \rightarrow$ *child* are present, but also clauses of the type $father \wedge child \rightarrow mother$ and $mother \wedge child \rightarrow father$.

k-AC encoding was also used because, since parental relations will originate ternary clauses, the support encoding could not be used since they are only valid for binary restraints [13].

The ternary clauses in k-AC encoding for figure 1 are:

k-AC clauses

$$\begin{array}{lll}
[x1_{AA} \wedge x2_{AA} \rightarrow x3_{AA}] & [x1_{AA} \wedge x2_{AB} \rightarrow x3_{AA} \vee x3_{AB}] & [x1_{AA} \wedge x2_{BB} \rightarrow x3_{AB}] \\
[x1_{AB} \wedge x2_{AA} \rightarrow x3_{AA} \vee x3_{AB}] & [x1_{AB} \wedge x2_{AB} \rightarrow x3_{AA} \vee x3_{AB} \vee x3_{BB}] & [x1_{AB} \wedge x2_{BB} \rightarrow x3_{AB} \vee x3_{BB}] \\
[x1_{BB} \wedge x2_{AA} \rightarrow x3_{AB}] & [x1_{BB} \wedge x2_{AB} \rightarrow x3_{AB} \vee x3_{BB}] & [x1_{BB} \wedge x2_{BB} \rightarrow x3_{BB}] \\
 & & \\
[x1_{AA} \wedge x3_{AA} \rightarrow x2_{AA} \vee x2_{AB}] & [x1_{AA} \wedge x3_{AB} \rightarrow x2_{AB} \vee x2_{BB}] & [x1_{AA} \wedge x3_{BB} \rightarrow False] \\
[x1_{AB} \wedge x3_{AA} \rightarrow x2_{AA} \vee x2_{AB}] & [x1_{AB} \wedge x3_{AB} \rightarrow x2_{AA} \vee x2_{AB} \vee x2_{BB}] & [x1_{AB} \wedge x3_{BB} \rightarrow x2_{AB} \vee x2_{BB}] \\
[x1_{BB} \wedge x3_{AA} \rightarrow False] & [x1_{BB} \wedge x3_{AB} \rightarrow x2_{AA} \vee x2_{AB}] & [x1_{BB} \wedge x3_{BB} \rightarrow x2_{AB} \vee x2_{BB}] \\
 & & \\
[x2_{AA} \wedge x3_{AA} \rightarrow x1_{AA} \vee x1_{AB}] & [x2_{AA} \wedge x3_{AB} \rightarrow x1_{AB} \vee x1_{BB}] & [x2_{AA} \wedge x3_{BB} \rightarrow False] \\
[x2_{AB} \wedge x3_{AA} \rightarrow x1_{AA} \vee x1_{AB}] & [x2_{AB} \wedge x3_{AB} \rightarrow x1_{AA} \vee x1_{AB} \vee x1_{BB}] & [x2_{AB} \wedge x3_{BB} \rightarrow x1_{AB} \vee x1_{BB}] \\
[x2_{BB} \wedge x3_{AA} \rightarrow False] & [x2_{BB} \wedge x3_{AB} \rightarrow x1_{AA} \vee x1_{AB}] & [x2_{BB} \wedge x3_{BB} \rightarrow x1_{AB} \vee x1_{BB}]
\end{array}$$

By observing the two previous set of clauses, we notice that there are three times more k-AC clauses than there are minimal support clauses when encoding a ternary parental relation.

4 Tests on the studied encodings

For the comparative tests between minimal support encoding and k-AC encoding, one of the types of pedigree instances used were random instances. These instances are divided into four collections (A, B, C and D), and each collection is composed of subclasses grouped by number of individuals (except for collection D in which the subclasses are grouped by number of founding males.)

The tests are listed bellow:

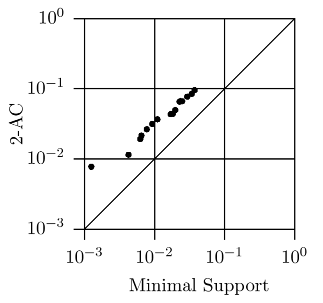

Fig. 4. CPU times for minimal support encoding vs 2-AC encoding in collection A of random instances.

Each dot in a plot represents the average running time of a sub collection in both the minimal support and the k-AC encoding, and as we can observe, instances encoded with the minimal support take, in general, a lesser amount of time when compared to the running time of the k-AC encoding. This may be because, since the number of clauses in k-AC are three times more than

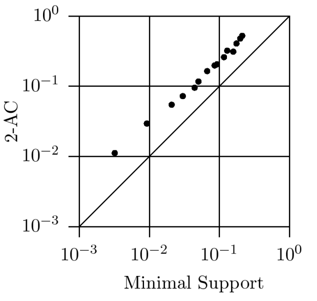

Fig. 5. CPU times for minimal support encoding vs 2-AC encoding in collection B of random instances.

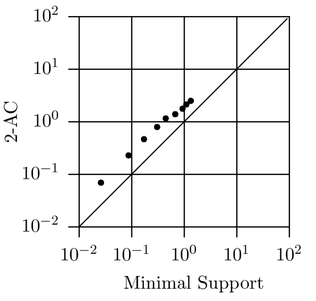

Fig. 6. CPU times for minimal support encoding vs 2-AC encoding in collection C of random instances.

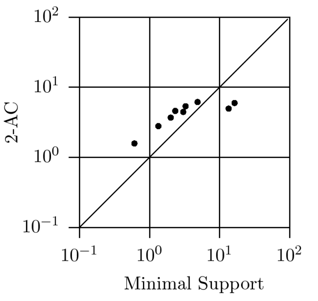

Fig. 7. CPU times for minimal support encoding vs 2-AC encoding in collection D of random instances.

the clauses in minimal support. However, we can observe, in figure 7 that there were two collection of instances that took lesser time in k-AC than in minimal support. This can signify that for larger instances, the k-AC encoding can run instances in faster time since unit propagation can be using without breaking consistency.

5 Discussion

Pedigree consistency checking is a NP-complete problem and therefore, no efficient solution to solve the problem in polimonial time has been found.

The main focus of this work was to attempt to use certain SAT encodings and model the pedigree consistency checking problem into a SAT one, more specifically, into a partial Max-SAT problem. In partial Max-SAT, there are hard clauses and soft clauses, and the objective is to satisfy all the hard clauses and the maximum number of soft ones. Since in the used pedigrees all of the paternal relations were considered correct and the associated genotype information could be considered incorrect, partial Max-SAT instances could be a good approach for this problem by transforming paternal relations into hard clauses and transforming genetic information into soft clauses.

Two main partial Max-SAT encodings were explored: minimum support encoding and k-AC encoding; being that the only difference between them is in the way that parental (ternary) relations will be mapped in the encoding. For the minimal support encoding, only a single *direction* of the relation need be mapped, as in the k-AC encoding all three directions require being encoded. In practice, this will generate three times more clauses in k-AC than in minimal support, but is what allows using unit propagation when solving and still maintain consistency.

In theory, the possibility of using unit propagation when running Max-SAT instances in a SAT solver should significantly decrease running times of tested instances. However this was not the case: instances encoded by means of the minimal support encoding took less time to run when compared to the k-AC encoding. Another reason for minimal support encoding being faster maybe due to the fact that, with relatively small instances, the number of clauses will be of bigger influence than the possibility of unit propagation. Hence, minimal support encoding, which as three times less clauses than k-AC, usually takes less time.

References

1. Göring, H.H.H., Terwilliger, J.D.: Linkage Analysis in the Presence of Errors I: Complex-Valued Recombination Fractions and Complex Phenotypes. American Journal of Human Genetics **66** (2000) 1095–1106
2. Ott, J.: Computer-simulation Methods in Human Linkage Analysis. Proceedings of the National Academy of Sciences **86** (1989) 4175–4178

3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Second Edition. McGraw-Hill Book Company (2001)
4. Russel, S., Norvig, P.: Artificial Intelligence A Modern Approach, Second Edition. Prentice Hall (2003)
5. Boros, E., Hammer, P.L.: Pseudo-Boolean Optimization. Discrete Applied Mathematics **1–3** (2002) 155–255
6. Argelich, J., Cabiscol, A., Lynce, I., Manyà, F.: Encoding Max-CSP into Partial Max-SAT. In: ISMVL '08: Proceedings of the 38th International Symposium on Multiple Valued Logic (ismvl 2008), Washington, DC, USA, IEEE Computer Society (2008) 106–111
7. Aceto, L., Hansen, J.A., Ingólfsdóttir, A., Johnsen, J., Knudsen, J.: The Complexity of Checking Consistency of Pedigree Information and Related Problems. Journal of Computer Science and Technology **19**(1) (2004) 42–59
8. Piccolboni, A., Gusfield, D.: On the Complexity of Fundamental Computational Problems in Pedigree Analysis. Journal of Computational Biology **10**(5) (2003) 763–773
9. Ehm, M.G., Kimmel, M., Cottingham, R.W.: Error Detection for Genetic Data Using Likelihood Methods. American Journal of Human Genetics **58**(1) (1996) 225–234
10. Broman, K.: Cleaning Genotype Data. In: Genetic Epidemiology, Wiley (1999)
11. Sanchez, M., Givry, S., Schiex, T.: Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. Constraints Journal **13**(1-2) (2008) 130–154
12. Walsh, T.: SAT v CSP. In: CP '02: Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming, London, UK, Springer-Verlag (2000) 441–456
13. Bessiere, C., Hebrard, E., Walsh, T.: Local Consistencies in SAT. In: Proc. SAT-2003, Springer (2003) 299–314
14. Kasif, S.: On the Parallel Complexity of Discrete Relaxation in Constraint Satisfaction Networks. Artificial Intelligence **45** (1990) 275–286