Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques

S. de Givry, I. Palhiere, Z. Vitezica and T. Schiex

INRA, Toulouse, France

Abstract. With the arrival of high throughput genotyping techniques, the detection of likely genotyping errors is becoming an increasingly important problem. In this paper we are interested in errors that violate Mendelian laws. The problem of deciding if Mendelian error exists in a pedigree is NP-complete [1]. Existing tools dedicated to this problem may offer different level of services: detect simple inconsistencies using local reasoning, prove inconsistency, detect the source of error, propose an optimal correction for the error. All assume that there is at most one error. In this paper we show that the problem of error detection, of determining the minimum number of error needed to explain the data (with a possible error detection) and error correction can all be modeled using soft constraint networks. Therefore, these problems provides attractive benchmarks for weighted constraint network solvers such as the dedicated WCSP solver `toolbar`.

1 Background

Chromosomes carry the genetic information of an individual. A position that carries some specific information on a chromosome is called a locus (which typically identifies the position of a gene). The specific information contained at a locus is the allele carried at the locus. Except for the sex chromosomes, diploid individuals carry chromosomes in pair and therefore a single locus carries a pair of allele. Each allele originates from one of the parents of the individual considered. This pair of alleles at this locus define the genotype of the individual at this locus. Genotypes are not always completely observable and the indirect observation of a genotype (its expression) is termed the phenotype. A phenotype can be considered as a set of possible genotypes for the individual. These genotypes are said to be compatible with the phenotype.

A pedigree is defined by a set of related individuals together with associated phenotypes for some locus. Every individual is either a founder (no parents in the pedigree) or not. In this latter case, the parents of the individual are identified inside the pedigree. Because a pedigree is built from a combination of complex experimental processes it may involve experimental and human errors. The errors can be classified as parental errors or typing errors. A parental error means that the very structure of the pedigree is incorrect: one parent of one individual is not actually the parent indicated in the pedigree. We assume that parental information is correct. A phenotype error means simply that the phenotype in the pedigree is incompatible with the true (unknown) genotype. Phenotype errors are called Mendelian errors when they make a pedigree inconsistent with Mendelian law of inheritance which states that the pair of alleles of every individual is composed of one paternal and one maternal allele. The fact that at least one Mendelian error exists can be effectively proven by showing that every possible combination of compatible genotypes for all the individual violates this law at least once. Since the number of these combinations grows exponentially with the number of individuals, only tiny pedigree can be checked by enumeration. The problem of checking pedigree consistency is actually shown to be NP-complete in [1]. Other errors are called non Mendelian errors[1].

The detection and correction of errors is crucial before the data can be exploited for the construction of the genetic map of a new organism (genetic mapping) or the localization of genes (loci) related to diseases or other quantitative traits. Because of its NP-completeness, most existing tools only offer a limited polynomial time checking procedure. The only tool we know that really tackles this problem is PedCheck [10, 11], although the program is restricted by a single error assumption. Evaluation of animal pedigrees with thousand of animals, including many loops (a marriage between two individuals which have a parental relation) (resulting in large tree-width) and without assuming the unlikely uniqueness of errors requires further improvements.

In this paper, we introduce soft constraint network models for the problem of checking consistency, the problem of determining the minimum number of errors needed to explain the data and the problem of proposing an optimal correction to an error. These problems offer attractive benchmarks for (weighted) constraint satisfaction. We report preliminary results using the weighted constraint network solver `toolbar`.

2 Modeling the problems

A constraint satisfaction network (X, D, C) [3] is defined by a set of variables $X = \{x_1, \ldots, x_n\}$, a set of matching domains $D = \{d_1, \ldots, d_n\}$ and a set of constraints C. Every variable $x_i \in X$ takes its value in the associated domain d_i. A constraint $c_S \in C$ is defined as a relation on a set of variables $S \subset X$ which defines authorized combination of values for variables in S. Alternatively, a constraint may be seen as the characteristic function of this set of authorized tuples. It maps authorized tuples to the boolean *true* (or `1`) and other tuples to *false* (or `0`). The set S is called the scope of the constraint and $|S|$ is the arity of the constraint. For arities of one, two or three, the constraint is respectively said to be unary, binary or ternary. A constraint may be defined by a predicate that decides if a combination is authorized or not, or simply as a set of combination of values which are authorized. For example, if $d_1 = \{1, 2, 3\}$ and $d_2 = \{2, 3\}$, two possible equivalent definitions for a constraint on x_1 and x_2 would be $x_1 + 1 > x_2$ or $\{(2, 2), (3, 2), (3, 3)\}$.

[1] Non Mendelian errors may be identified only in a probabilistic way using several locus simultaneously and a probabilistic model of recombination and errors [4].

The central problem of constraint networks is to give a value to each variable in such a way that no constraint is violated (only authorized combinations are used). Such a variable assignment is called a solution of the network. The problem of finding such a solution is called the CONSTRAINT SATISFACTION PROBLEM (CSP). Proving the existence of a solution for an arbitrary network is an NP-complete problem.

Often, tuples are not just completely authorized or forbidden but authorized and forbidden to some extent or at some cost. Several ways to model such problems have been proposed among which the most famous are the semi-ring and the valued constraint networks frameworks [2]. In this paper we consider weighted constraint networks (WCN) where constraints map tuples to non negative integers. For every constraint $c_S \in C$, and every variable assignment A, $c_S(A[S]) \in N$ represents the cost of the constraint for the given assignment where $A[S]$ is the projection of A on the constraint scope S. The aim is then to find an assignment A of all variables such that the sum of all tuple costs $\sum_{c_S \in C} c_S(A[S])$ is minimum. This is called the Weighted Constraint Satisfaction Problem (WCSP), obviously NP-hard. Several recent algorithms for tackling this problem, all based on the maintenance of local consistency properties [13] have been recently proposed [9, 7].

2.1 Genotyped pedigree and constraint networks

Consider a pedigree defined by a set I of individuals. For a given individual $i \in I$, we note $pa(i)$ the set of parents of i. Either $pa(i) \neq \varnothing$ (non founder) or $pa(i) = \varnothing$ (founder). At the locus considered, the set of possible alleles is denoted by $A = 1, ..., m$. Therefore, each individual carries a genotype defined as an unordered pair of alleles (one allele from each parent, both alleles can be identical). The set of all possible genotypes is denoted by G and has cardinality $\frac{m(m+1)}{2}$. For a given genotype $g \in G$, the two corresponding alleles are denoted by g^l and g^r and the genotype is also denoted as $g^l|g^r$. By convention, $g^l \leq g^r$ in order to break symmetries between equivalent genotypes (e.g. $1|2$ and $2|1$). The experimental data is made of phenotypes. For each individual in the set of observed individuals $I' \subset I$, its observed phenotype restricts the set of possible genotypes to those which are compatible with the observed phenotype. This set is denoted by $G(i)$ (very often $G(i)$ is a singleton, observation is complete).

A corresponding constraint network encoding this information uses one variable per individual *i.e.* $X = I$. The domain of every variable $i \in X$ is simply defined as the set of all possible genotypes G. If an individual i has an observed phenotype, a unary constraint that involves the variable i and authorizes the genotypes in $G(i)$ is added to the network. Finally, to encode Mendelian law, and for every non founder individual $i \in X$, a single ternary constraint involving i and the two parents of i, $pa(i) = \{j, k\}$ is added. This constraint only authorizes triples (g_i, g_j, g_k) of genotypes that verify Mendelian inheritance *i.e.* such that one allele of g_i appears in g_j and the other appears in g_k. Equivalently:

$$(g_i^l \in g_j \wedge g_i^r \in g_k) \vee (g_i^l \in g_k \wedge g_i^r \in g_j)$$

For a pedigree with n individuals among which there are f founders, with m possible alleles, we obtain a final CN with n variables, a maximum domain size of $\frac{m(m+1)}{2}$ and $n - f$ ternary constraints. Existing problems may have more than 10,000 individuals with several alleles (the typical number of alleles varies from two to a dozen).

A small example is given in Fig. 1. There are $n = 12$ individuals and $m = 3$ distinct alleles. Each box corresponds to a male individual, and each ellipse to a female. The arcs describe parental relations. For instance, individuals 1 and 2 have three children 3,4, and 5. The founders are individuals 1,2,6, and 7 ($f = 4$). The possible genotypes are $G = \{1|1, 1|2, 1|3, 2|2, 2|3, 3|3\}$. There are 7 individuals (1,3,6,7,10,11, and 12) with an observed phenotype (a single genotype). The corresponding CSP has 12 variables, with maximum domain size of 6, and 8 ternary constraints. This problem is inconsistent.

A solution of this constraint network defines a genotype for each individual that respects Mendelian law (ternary constraints) and experimental data (domains) and the consistency of this constraint network is therefore obviously equivalent to the consistency of the original pedigree. As such, pedigree consistency checking offers a direct problem for constraint networks. In practice, solving this problem is not enough (i) if the problem is consistent, one should simplify the problem for further probabilistic processing by removing all values (genotypes) which do not participate in any solution, this specific problem is known as "genotype elimination", (ii) if the problem is inconsistent, errors have to be located and corrected.

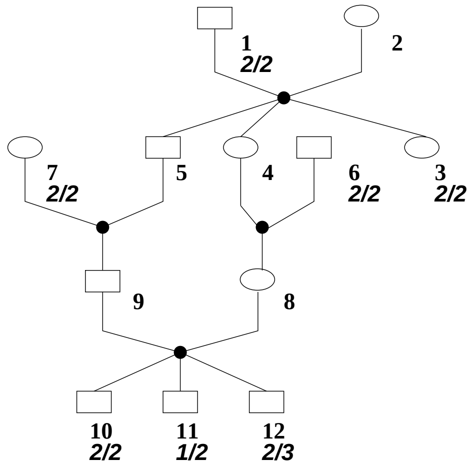

Fig. 1. Pedigree example taken from [11] with 12 individuals.

3 Error detection

From an inconsistent pedigree, the first problem is to identify errors. In our knowledge, this problem is not perfectly addressed by any existing programs which either identify a not necessarily minimum cardinality set of individuals that only restore a form of local consistency or make the assumption that a single error occurs. The first approach may detect too many errors that moreover may not suffice to remove global inconsistency and the second approach is limited to small datasets (even high quality automated genotyping may generate several errors on large datasets).

A typing error for individual $i \in X$ means that the domain of variable i has been wrongly reduced: the true (unknown) value of i has been removed. To model the possibility of such errors, genotypes in G which are incompatible with the observed phenotype $G(i)$ should not be completely forbidden. Instead, a soft constraint forbids them with a cost of 1 (since using such a value represents one typing error). We thereby obtain a weighted constraint network with the same variables as before, the same hard ternary constraints for Mendelian laws and soft unary constraints for modeling genotyping information. Domains are all equal to G.

If we consider an assignment of all variables as indicating the real genotype of all individuals, it is clear that the sum of all the costs induced by all unary constraints on this assignment precisely gives the number of errors made during typing. Finding an assignment with a minimum number of errors follows the traditional parsimony principle (or Ockham's razor) and is consistent with a low probability of independent errors (quite reasonable here). One solution of the corresponding WCSP with a minimum cost therefore defines a possible diagnostic (variable assigned with a value forbidden by $G(i)$ represent one error).These networks have the same size as the previous networks with the difference that all variables now have the maximum domain size $|G|$. The fundamental difference lies in the shift from satisfaction to optimization. The fact that only *unary* soft constraints arise here is not a simplification in itself w.r.t. the general WCSP since every n-ary weighted constraint network can be simply translated in an equivalent dual network with only unary soft constraints and hard binary constraints [8].

In the previous example of Fig. 1, the problem still has 12 variables, with domain size of 6. It has 8 hard ternary constraints and 7 soft unary constraints. The minimum number of typing errors is one. There are 66 optimal solutions of cost one, which can occur in any of the typed individuals except individual 10. One optimal solution is $\{(1, 2|2), (2, 1|2), (3, 2|2), (4, 1|2), (5, 2|2), (6, 2|2), (7, 2|2), (8, 1|2), (9, 2|2), (10, 2|2), (11, 1|2), (12, 2|2)\}$ such that the *erroneous* typing 2|3 of individual 12 has been changed to 2|2.

3.1 Error correction

When errors are detected, one would like to optimally correct them. The simple parsimony criterion is usually not sufficient to distinguish alternative values.

More information needs to be taken into account. Errors and Mendelian inheritance being typically stochastic processes, a probabilistic model is attractive. A Bayesian network is a network of variables related by conditional probability tables (CPT) forming a directed acyclic graph. It allows to concisely describe a probability distribution on stochastic variables. To model errors, a usual approach is to distinguish the observation O and the truth T. A CPT $P(O|T)$ relates the two variables and models the probability of error.

Following this, we consider the following model for error correction: we first have a set of n variables T_i each representing the true (unknown) genotype of individual i. The domain is G. For every observed phenotype, an extra observed variable O_i is introduced. It is related to the corresponding true genotype by the CPT $P_i^e(O_i|T_i)$. In our case, we assume that there is a constant α probability of error: the probability of observing the true genotype is $1-\alpha$ and the remaining probability mass is equally distributed among remaining values.

For the individual i and its parents $pa(i)$, a CPT $P_i^m(T_i|pa(i))$ representing Mendelian inheritance connects T_i and its corresponding parents variables. Each parent having two alleles, there are four possible combinations for the children and all combinations are equiprobable (probability $\frac{1}{4}$). However, since all parental alleles are not always different, a single children genotype will have a probability of $\frac{1}{4}$ times the number of combination of the parental alleles that produce this genotype.

Finally, prior probabilities $P^f(i)$ for each genotype must be given for every founder i. These probabilities are obtained by directly estimating the frequency of every allele in the genotyped population. For a genotype, its probability is obtained by multiplying each allele frequency by the number of way this genotype can be built (a genotype $a|b$ can be obtained in two ways by selecting a a from a father and a b from the mother or the converse. If both alleles are equal there is only one way to achieve the genotype). The probability of a complete assignment $P(O,T)$ (all true and observed values) is then defined as the product of the three collections of probabilities (P^e, P^m and P^f). Note that equivalently, its log-probability is equal to the sum of the logarithms of all these probabilities.

The evidence given by the observed phenotypes $G(i)$ is taken into account by reducing the domains of the O_i variables to $G(i)$. One should then look for an assignment of the variables $T_i, i \in I'$ which has a maximum a posteriori probability (MAP). The MAP probability of such an assignment is defined as the sum of the probabilities of all complete assignments extending it and maximizing it defines an NP^{PP}-complete problem [12], for which no exact methods exist that can tackle large problems. PedCheck tries to solve this problem using the extra assumption of a unique already identified error. This is not applicable in large datasets either. Another very strong assumption (known as the Viterbi assumption) considers that the distribution is entirely concentrated in its maximum and reduces MAP to the so-called Maximum Probability Explanation problem (MPE) which simply aims at finding a complete assignment of maximum probability. Using logarithms as mentioned above, this problem directly reduces to a WCSP problem where each CPT is transformed in an additive cost function. This allows to solve MPE using the dedicated algorithms introduced in `toolbar`.

Taken the pedigree example given in Fig. 1, we computed the maximum likelihood $P(O,T|e)$ for all possible minimal error corrections using `toolbar` with the MPE formulation. Note that only one correction is needed here to suppress all Mendelian errors. For a given error correction $T_j = g$, the given evidence e corresponds to reducing the domains of all the O_i *and* T_i variables to $G(i)$, except for T_j which is assigned to g. We used equifrequent allele frequencies $P^f(i)$ for the founders and a typing error probability $\alpha = 5\%$. The results are given in Table 1. The ratio $\frac{P(O,T|e)^{best}}{P(O,T|e)}$ is the ratio between the best found maximum likelihood for any correction and the maximum likelihood for a given correction. The results shown that either individual 11 or 12 is the most likely source of the error. Optimal corrections with respect to the Viterbi assumption are either $T_{11} = 2|2$, or $T_{11} = 2|3$, or $T_{12} = 1|2$, or $T_{12} = 2|2$.

Individual	Correction	$P(O,T\|e)$	$\frac{P(O,T\|e)^{best}}{P(O,T\|e)}$
1	1|2	$3.462e-10$	128.0
1	2|3	$3.462e-10$	128.0
3	1|2	$2.77e-09$	16.0
3	2|3	$2.77e-09$	16.0
6	1|1	$5.539e-09$	8.0
6	1|2	$2.77e-09$	16.0
6	1|3	$2.77e-09$	16.0
6	2|3	$2.77e-09$	16.0
6	3|3	$5.539e-09$	8.0
7	1|1	$5.539e-09$	8.0
7	1|2	$2.77e-09$	16.0
7	1|3	$2.77e-09$	16.0
7	2|3	$2.77e-09$	16.0
7	3|3	$5.539e-09$	8.0
11	2|2	$4.431e-08$	1.0
11	2|3	$4.431e-08$	1.0
11	3|3	$5.539e-09$	8.0
12	1|1	$5.539e-09$	8.0
12	1|2	$4.431e-08$	1.0
12	2|2	$4.431e-08$	1.0

Table 1. Likelihood ratios for possible error corrections for the problem given in Fig.1.

4 Conclusion

Preliminary experiments on the applicability of WCSP techniques to these problems have been made. The first real data sets are human pedigree data presented in [10, 11]. For all these problems, `toolbar` solves the consistency, error minimization and correction (MPE) problems very rapidly.

The real challenge lies in a real 129,516 individual pedigree of sheeps (*sheep4n*), among which 2,483 are typed with 4 alleles and 20,266 are founders. By removing uninformative individuals, it can be reduced to a still challenging 8,990 individual pedigree (*sheep4nr*) among which 2,481 are typed and 1,185 are founders. As an animal pedigree, it contains many loops and the min-fill heuristics gives a 226 tree-width bound. From the reduced pedigree *sheep4nr*, we removed some simple nuclear-family typing errors by removing parental and typing data of the corresponding erroneous children, resulting in a 8,920 individual pedigree (*sheep4r*) among which 2,446 are typed and 1,186 are founders. We solved the error detection problem (Section 3) using a decomposition approach. We used the hypergraph partitioning software *shmetis*[2] [6] using multilevel recursive bisection with imbalance factor equal to 5% to partition the *sheep4r* hypergraph into four parts. Among the 8,662 hyperedges (each hyperedge connects one child to its parents), 1,676 have been cut, resulting into four independent subproblems described in Table 2. This table gives the minimum number of typing errors (*Optimum*), and if non zero and equal to one, the list of individual identifiers such that removing one typing will suppress all Mendelian errors. The experiments were performed on a 2.4 GHz Xeon computer with 8 GB. The Table reports CPU time needed by `toolbar` and `PedCheck` [10, 11] (*level 3*) to find the list of erroneous individuals. We used default parameters for `toolbar` except a pre-projection of ternary constraints into binary constraints, a singleton consistency preprocessing [5], and an initial upper bound equal to two. Recall that `PedCheck` is restricted by a single error assumption.

Instance	N. of individuals	Optimum	Errors	`toolbar` CPU time	`PedCheck` CPU time
sheep4r_4_0	1,944	0	$\emptyset$	33.3 sec	27.1 sec
sheep4r_4_1	2,037	0	$\emptyset$	35.7 sec	28.2 sec
sheep4r_4_2	2,254	1	$\{126033\}$	124.2 sec	3 hours 51 min
sheep4r_4_3	2,685	1	$\{119506, 128090, 128091, 128092\}$	295.0 sec	7 hours 25 min

Table 2. Error detection for a large pedigree *sheep4r* decomposed into four independent weighted CSPs.

By removing two typings in *sheep4r* (individuals 126033 for *sheep4r_4_2* and 119506 for *sheep4r_4_3*), we found that the modified pedigree has no Mendelian errors.

These problems have been made available as benchmarks on the `toolbar` web site (carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP) for classical, weighted CSP and Bayesian net algorithmicists.

[2] `http://www-users.cs.umn.edu/~karypis/metis/hmetis/index.html`.

References

[1] Aceto, L., Hansen, J. A., Ingólfsdóttir, A., Johnsen, J., and Knudsen, J. The complexity of checking consistency of pedigree information and related problems. *Journal of Computer Science Technology 19*, 1 (2004), 42–59.

[2] Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., and Verfaillie, G. Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison. *Constraints 4* (1999), 199–240.

[3] Dechter, R. *Constraint Processing.* Morgan Kaufmann Publishers, 2003.

[4] Ehm, M. G., Cottingham Jr., R. W., and Kimmel, M. Error Detection in Genetic Linkage Data Using Likelihood Based Methods. *American Journal of Human Genetics 58*, 1 (1996), 225-234.

[5] S. de Givry. Singleton consistency and dominance for weighted csp. In *Proc. of 6th International CP-2004 Workshop on Preferences and Soft Constraints*, page 15p., Toronto, Canada, 2004.

[6] Karypis, G., Aggarwal, R., Kumar, V., and Shekhar S. Multilevel Hypergraph Partitioning: Applications in VLSI Domain. In *Proc. of the Design and Automation Conference* (1997).

[7] Larrosa, J., de Givry, S., Heras, F., and Zytnicki, M. Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proc. of the* 19^{th} *IJCAI* (Aug. 2005).

[8] Larrosa, J., and Dechter, R. On the dual representation of non-binary semiring-based CSPs. CP'2000 Workshop on Soft Constraints, Oct. 2000.

[9] Larrosa, J., and Schiex, T. In the quest of the best form of local consistency for weighted CSP. In *Proc. of the* 18^{th} *IJCAI* (Aug. 2003), pp. 239–244.

[10] O'Connell, J. R., and Weeks, D. E. PedCheck: a program for identification of genotype incompatibilities in linkage analysis. *American Journal of Human Genetics 63*, 1 (1998), 259–66.

[11] O'Connell, J. R., and Weeks, D. E. An optimal algorithm for automatic genotype elimination. *American Journal of Human Genetics 65*, 6 (1999), 1733–40.

[12] Park, J. D., and Darwiche, A. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research 21* (2004), 101–133.

[13] Schiex, T. Arc consistency for soft constraints. In *Principles and Practice of Constraint Programming - CP 2000*, vol. 1894 of *LNCS*, pp. 411–424.