Checking Pedigree Consistency with PCS*

Panagiotis Manolios, Marc Galceran Oms, and Sergi Oliva Valls

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280 USA
{manolios,mgalceran3,soliva3}@gatech.edu

Abstract. Many important problems in bioinformatics and genetics require analyses that are NP-complete. For example, one of the basic problems facing researchers that analyze pedigrees—data that represents relationships and genetic traits of a set of individuals—is evaluating whether they are consistent with the Mendelian laws of inheritance. This problem is NP-complete and several specialized algorithms have been devised to solve the types of problems occurring in practice efficiently. In this paper, we present *PCS*, a tool based on Boolean Satisfiability (SAT) that is orders of magnitude faster than existing algorithms, and more general. In fact, *PCS* can solve real pedigree checking problems that cannot be solved with any other existing tool.

Key words: Boolean satisfiability, SAT, Pedigree Consistency checking, bioinformatics, genetics, computational biology

1 Introduction

Computational methods have become increasingly important in the fields of biology and genetics. In fact, the computational needs of these fields have led to grand challenge problems in computing, such as solving the protein folding problem, one of the main motivations behind IBM's Blue Gene project. Many of the problems in these domains turn out to be NP-complete, and therefore reducible to Boolean satisfiability (SAT). Given the recent improvements in SAT-solving technology, a natural question is whether SAT-based methods can be used to solve important problems arising in biology and genetics. In this paper, we provide evidence that this is in fact likely.

We focus on the pedigree consistency checking problem, a well studied and important problem. Pedigrees describe genotype information about a collection of related individuals. When we say that a pedigree is consistent, we mean that it is consistent with the laws of Mendelian inheritance. Pedigree checking is important for numerous reasons. For example, it turns out that inconsistent pedigree data can adversely affect linkage analysis, the process by which human genes are linked to traits such as the predisposition to various diseases [10, 1].

* This research was funded in part by NSF grants CCF-0429924, IIS-0417413, and CCF-0438871.

The consistency checking problem for pedigrees is NP-complete [1] and has been tackled in essentially two different ways. The first approach is based on specialized algorithms. This includes algorithms for dealing with the simpler "non-looping" pedigrees, *e.g.*, by K. Lange and T. Goradia [3] and algorithms for loop-breaking, which reduce the problem to the simpler non-looping case.[1] The Pedcheck tool, developed by J. O'Connell and E. Weeks, is the best known example of this approach [8, 9]. Secondly, there is another, very recent approach by de Givry et al. that is based on the use of weighted constraint satisfaction techniques. MendelSoft is a tool implementing this approach [2].

In this paper, we describe *PCS*, a SAT-based tool that leads to orders of magnitude performance improvements over existing tools for checking pedigrees.

2 Pedigree Consistency

A *pedigree* represents family relationships among a set of individuals, as well as genotype information on the individuals. The genotype information consists of a pair of *alleles*, DNA codings appearing in given positions on chromosomes. Alleles are DNA stands that correspond to a gene, the basic unit of heredity. The pedigree is consistent, with the Mendelian laws of inheritance, if every individual inherits exactly one allele from each of its parents.

Existing systems require that all individuals in pedigrees have either two or no parents and, similarly, two or no alleles. In *PCS*, we can also handle pedigrees containing partial information, *e.g.*, individuals with one unknown parent and/or one undefined allele are allowed. These extensions were easy to implement due to the flexibility of our approach, which involves translating the pedigree consistency checking problem to a satisfiability problem.

3 Tool Description

In this section, we give a brief overview of the internals of *PCS* [4] and also describe how *PCS* is used. *PCS* can be downloaded from `http://www.cc.gatech.edu/~manolios/pcs/`. The input to *PCS* is linkage-format data given in the formats describe at `http://linkage.rockefeller.edu/soft/linkage/`. In brief, the information for a member of the pedigree appears on one line as a sequence of integers. These integers indicate the family identifier, the member identifier, the father identifier, the mother identifier, the sex (1 for male and 2 for female), the first allele number, and the second allele number.

The pedigree data is preprocessed to rule out simple errors, *e.g.*, we check that every member's father is a male and every member's mother is a female. We also check that in case the data is declared to be sex-linked, all males are homozygots, which means that all males have only one allele.

[1] Loops in pedigrees arise when there is a loop in the graph of mates, a graph whose nodes are individuals and whose edges encode the mating relationship. For example, marriage loops are formed when one individual mates with two siblings.

The main phase of *PCS* is the translation of the consistency problem into a SAT problem. We cannot describe the details of this translation here, but we note that we make essential use of the BAT tool [6, 5]. BAT implements a decision procedure for the BAT language, a powerful hardware description language. This allows us to express the consistency problem in a high-level language and to leave the details of generating reasonable CNF to BAT.

The generated CNF file can then be given to any standard SAT solver. If a satisfying assignment is found, then the problem is consistent. If the formula is unsatisfiable, then there are incompatibilities in the pedigree. In this case, it is absolutely necessary to determine the problem and to communicate it to the user. We do this by extracting an *unsatisfiable core*, an unsatisfiable subset of the clauses that will become satisfiable if any of its clauses are removed. In the worst case, this includes the set of all clauses, but in practice this is highly unlikely. By extracting an unsatisfiable core, we can determine which set of members have inconsistent genomic information and why. Instead of reporting one such error at a time, *PCS* iteratively generates unsatisfiable cores and removes the genotype information of the individuals involved until we reach a fixed point. When the fixed point is reached, we have a satisfiable problem and *PCS* generates a report outlining all of the inconsistencies found.

4 Results

We compare the performance of our approach with Pedcheck, the most widely used program for Pedigree Checking [8, 9] and with MendelSoft [2], a tool that is based on a new approach involving weighted constraint satisfaction techniques.

We use the zchaff SAT-solver [7] for satisfiability testing and for extracting unsatisfiable cores [11]. All experiments were run on an Intel 3.06GHz Xeon machine, with 512 KB of L2 cache running on a GNU/Linux OS with kernel version 2.6.9.

We used both randomly generated benchmarks and actual pedigree problems from various domains. *PCS* detected the same errors as Pedcheck and MendelSoft, but it was two to three orders of magnitude faster than Pedcheck and one to two orders of magnitude faster than MendelSoft.

One of the most complicated examples we used consisted of actual pedigree data from sheep. This data was obtained from a repository provided by the authors of the MendelSoft system [2]. After some preprocessing, this data set consists of 8,920 members. The data could not be handled directly by neither MendelSoft nor PedCheck. Therefore, it was partitioned into four smaller problems, entitled sheep4r_4_0, ..., sheep4r_4_3. PedCheck took over 10 hours to solve the subproblems and MendelSoft tool over an hour. *PCS* was able to solve all four problems in under 20 seconds. This includes the total time required by BAT, the SAT solver, and the unsatisfiable core generator. In addition, *PCS* can deal with the whole pedigree, sheep4r directly, without having to partition the problem into subproblems (which was done by removing parent child relationships, something that can mask inconsistencies). *PCS* was able to correctly

solve this problem, which was not solvable by any other existing tool, in under a minute.

5 Conclusions

We introduced *PCS*, a SAT-based tool for checking the consistency of pedigrees. *PCS* is orders of magnitude faster than the most efficient existing algorithms. It is also more general and it is capable of easily solving real pedigree checking problems that cannot be solved with existing tools. Our work benefited greatly from the use of BAT's high-level language and the BAT decision procedure. The high-level language allowed us to think and operate at a much higher level than CNF without sacrificing efficiency, as the BAT decision procedure was able to quickly generate compact CNF optimized for current SAT solvers. Encouraged by our results, we believe that SAT-based methods should be applied to other hard problems in computational biology.

References

[1] L. Aceto, J. A. Hansen, A. Ingólfsdóttir, J. Johnsen, and J. Knudsen. The complexity of checking consistency of pedigree information and related problems. In *Proceedings of the Eighth Italian Conference on Theoretical Computer Science (ICTCS'03)*, pages 174–187, 2003.

[2] S. de Givry, I. Palhiere, Z. Vitezica, and T. Schiex. Mendelian error detection in complex pedigree using weighted constraint satisfaction techniques. In ICLP-05 workshop on Constraint Based Methods for Bioinformatics, Sitges, Spain, 2005.

[3] K. Lange and T. Goradia. An algorithm for automatic genotype elimination. *American Journal of Human Genetics*, 40(3):250–256, 1987.

[4] P. Manolios, M. G. Oms, and S. O. Valls. PCS: Pedigree Checking with SAT. 2007. Available from `http://www.cc.gatech.edu/~manolios/pcs/`.

[5] P. Manolios, S. K. Srinivasan, and D. Vroon. Automatic memory reductions for RTL-level verification. In *ACM-IEEE International Conference on Computer Aided Design (ICCAD 2006)*, November 2006.

[6] P. Manolios, S. K. Srinivasan, and D. Vroon. BAT: The Bit-level Analysis Tool. 2006. Available from `http://www.cc.gatech.edu/~manolios/bat/`.

[7] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conference (DAC'01)*, pages 530–535, 2001.

[8] J. R. O'Connell and D. E. Weeks. Pedcheck: A program for identification of genotype incompatibilities in linkage analysis. *American Journal of Human Genetics*, 63(1):259–266, 1998.

[9] J. R. O'Connell and D. E. Weeks. An optimal algorithm for automatic genotype elimination. *American Journal of Human Genetics*, 65(6):1733–1740, 1999.

[10] E. Sobel, J. C. Papp, and K. Lange. Detection and integration of genotyping errors in statistical genetics. *American Journal of Human Genetics*, 70:496–508, 2002.

[11] L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Proceedings of the Design and Test in Europe Conference*, pages 10880–10885, March 2003.