HAPLO-ASP: Haplotype Inference Using Answer Set Programming

Esra Erdem[1], Ozan Erdem[1], and Ferhan Türe[2]

[1] Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul 34956, Turkey
[2] Department of Computer Science, University of Maryland, College Park, MD 20742, USA

Abstract. Identifying maternal and paternal inheritance is essential to be able to find the set of genes responsible for a particular disease. However, due to technological limitations, we have access to genotype data (genetic makeup of an individual), and determining haplotypes (genetic makeup of the parents) experimentally is a costly and time consuming procedure. With these biological motivations, we study haplotype inference—determining the haplotypes that form a given set of genotypes—using Answer Set Programming; we call our approach HAPLO-ASP. This note summarizes the range of problems that can be handled by HAPLO-ASP, and its applicability and effectiveness on real data in comparison with the other existing approaches.

1 Introduction

Each genotype (the specific genetic makeup of an individual) in a diploid organism has two copies, one from the mother and one from the father. These two copies are called haplotypes, and they combine to form the genotype. The genetic information contained in haplotypes can be used for early diagnosis of diseases, detection of transplant rejection, and creation of evolutionary trees. However, although it is easier to access to the genotype data, due to technological limitations, determining haplotypes experimentally is a costly and time consuming procedure. With these biological motivations, researchers have been studying haplotype inference—determining the haplotypes that form a given set of genotypes—by means of some computational methods.

One haplotype inference problem that has been extensively studied is Haplotype Inference by Pure Parsimony (**HIPP**) [1]. This problem asks for a minimal set of haplotypes that form the given genotypes; the decision version of **HIPP** is NP-hard [1,2]. **HIPP** has been studied with various approaches, such as, HYBRIDIP (based on integer linear programming) [3], HAPAR (based on a branch and bound algorithm) [4], SHIPS (based on a SAT-based algorithm) [5], RPOLY (based on pseudo-boolean optimization methods) [6].

Another haplotype inference problem that has been studied so far is Haplotype Inference from Present-Absent Genotype (**HIPAG**) [7,8]; it is a variation of **HIPP** that takes into account biallelic genotypes only, possibly with missing information, and domain-specific information, like haplotype patterns observed for some specific gene family. **HIPAG** has been studied by [7] with a greedy algorithm (HAPLO-IHP); [9] studies a variation of **HIPAG** for polyallelic and polyploid genotypes but does not take into

E. Erdem, F. Lin, and T. Schaub (Eds.): LPNMR 2009, LNCS 5753, pp. 573–578, 2009.

account domain-specific information, and computes a solution using a SAT-based algorithm (SATLOTYPER).

Recently, we have presented a novel approach to solving both sorts of haplotype inference problems, **HIPP** and **HIPAG**, and their variations, using Answer Set Programming (ASP) [8]; this ASP-based approach to haplotype inference is called HAPLO-ASP. We have extended the applicability of HAPLO-ASP, thanks to its expressive representation language, to solve also variations of **HIPP** and **HIPAG** where we can specify preferences over parts of haplotypes by assigning weights to their sites/alleles in accordance with their importance, and/or where we can consider polyallelic and polyploid genotypes. This note summarizes the range of problems handled by HAPLO-ASP, and its applicability and effectiveness on real data in comparison with the other approaches.

2 Haplotype Inference by Pure Parsimony

Haplotype Inference by Pure Parsimony (**HIPP**) [1] asks for a minimal set of haplotypes that explain the given genotypes. The decision version of **HIPP** (i.e., deciding that a set of k haplotypes that explain the given genotypes exists) is NP-hard [1,2].

A standard definition of the concept of two haplotypes "explaining" a genotype appears in [1]. According to this definition, we view a genotype as a vector of sites, each site having a value 0, 1, or 2; and a haplotype as a vector of sites, each site having a value 0 or 1. The values 0 and 1 (called *alleles*) correspond to complementary bases, like C and G. The sites correspond to single nucleotide polymorphisms (SNPs). A site of a genotype is *ambiguous* if its value is 2; and *resolved* otherwise. Two haplotypes h_1 and h_2 *form (explain)* a genotype g if for every site j the following hold: if $g[j] = 2$ then $h_1[j] = 0$ and $h_2[j] = 1$ or $h_1[j] = 1$ and $h_2[j] = 0$; if $g[j] = 1$ then $h_1[j] = 1$ and $h_2[j] = 1$; and if $g[j] = 0$ then $h_1[j] = 0$ and $h_2[j] = 0$. For instance, the genotype 20110 can be explained by the haplotypes 10110 and 00110.

We consider the following decision version of **HIPP**:

HIPP-DEC Given a set G of n genotypes each with m sites, and a positive integer k, decide whether there is a set H of at most k unique haplotypes such that each genotype in G is explained by two haplotypes in H.

Erdem and Türe have presented in [8] an ASP program that describes **HIPP-DEC**. An instance of **HIPP** can be solved with that ASP program, by trying various values for k (the number of unique haplotypes explaining the given genotypes). HAPLO-ASP computes an approximate lower bound l for k and an upper bound u for k, using ASP, and tries to find the optimal value for k by a binary search between l and u. The computation of such a lower bound and upper-bound is discussed in [8].

3 Haplotype Inference from Present-Absent Genotype

Haplotype Inference from Present-Absent Genotype (**HIPAG**) is another haplotype inference problem, which asks for the minimal set of haplotypes "compatible" with the given genotypes. Both haplotypes and genotypes can be viewed as vectors, as in **HIPP**.

In this problem, each site of a haplotype takes one of the two values 0 or 1 specifying the presence/absence of a particular gene. Sites of genotypes are biallelic; the value of each site is a pair of numbers from $\{0, 1, ?\}$. For instance, $(1, ?)(1, ?)(1, 1)$ is a genotype.

For a genotype g of the form $(g_{11}, g_{12})...(g_{m1}, g_{m2})$, let us denote by g^1 the vector $g_{11}g_{21}...g_{m1}$ and by g^2 the vector $g_{12}g_{22}...g_{m2}$. We say that two alleles i and j are *compatible* if they are identical or if one of them is $?$. Two haplotypes h_1 and h_2 are *compatible* with a genotype g, all with m sites, if, for every site j, $h_1[j]$ is compatible with one of the two alleles, $g^1[j]$ or $g^2[j]$, and $h_2[j]$ is compatible with the other. For instance, the haplotypes 011 and 111 are compatible with $(1, 0)(1, 1)(1, 1)$ and $(1, ?)(1, ?)(1, 1)$ but not with $(1, ?)(1, 0)(1, 1)$. Note that, we can discard the sites with missing information while computing a solution for **HIPAG**.

We consider the following decision version of **HIPAG**:

HIPAG-DEC Given a set G of n genotypes each with m biallelic sites, and a positive integer k, decide that there is a set H of at most k unique haplotypes such that each genotype in G is compatible with two haplotypes in H.

We have shown that **HIPAG-DEC** is also NP-complete.

[8] discusses how to describe **HIPAG-DEC** as an ASP program. Then an instance of **HIPAG** can be solved with that ASP program, by trying various values for k, as in **HIPP** instances.

4 Other Variations of Haplotype Inference

HAPLO-ASP can solve also variations of **HIPP** and **HIPAG** where we can specify domain-specific information (like haplotype patterns observed for some gene family), preferences over parts of haplotypes by assigning weights to their sites/alleles in accordance with their importance in detecting the cause of a disease, and/or where we can consider polyallelic and polyploid genotypes. In the following, we will briefly discuss these problems.

Observed Haplotype Patterns. In haplotype studies of some gene families, sometimes some patterns may be observed. For instance, [7] derived three patterns of haplotypes for the family of KIR genes of Caucasian population, from the observations in KIR haplotype studies like [10]. These patterns may help generating more accurate haplotypes, if included in the computation of haplotypes. Such domain-specific information can be described by an ASP program, as described in [8], and, during computation of haplotypes, it can be given to the answer set solver as an input in a separate program file.

Weighted Haplotype Inference. In some populations, some sites/alleles may have a more significant role in identifying, for instance, the cause of a disease, and thus have a larger weight. With this motivation, we have studied modified versions of **HIPP** and **HIPAG**: *Weighted Haplotype Inference by Pure Parsimony* (**WHIPP**) and *Weighted Haplotype Inference from Present-Absent Genotype* (**WHIPAG**). We have studied the decision versions of **WHIPP** and **WHIPAG** and presented them in ASP. We have been testing HAPLO-ASP on some real data.

Haplotype Inference with Polyallelic and Polyploid Genotypes. In the haplotype inference problems above, every genotype is explained by two haplotypes (since the individuals are *diploid*) and every genotype has at most two kinds of alleles (e.g., in **HIPAG**, the genes are *biallelic*). However, there are many species, like some varieties of the potato (e.g., Solanum tuberosum), with polyploid and polyallelic genes in nature, where every genotype is explained by more than two haplotypes, and each genotype consists of more than two kinds of alleles. *Haplotype Inference from Polyallelic and Polyploid Genotype* **(HIPPG)** extends **HIPAG** to such polyallelic and polyploid genotypes. We have studied the decision version of **HIPPG** and presented it in ASP. We have tested HAPLO-ASP on some real data for cultivated potato genes (described in [11]), and obtained promising results as in [9].

5 Experimental Results for HIPP and HIPAG

We have implemented a haplotype inference system, also called HAPLO-ASP, based on the ASP-based approach above; it is a PERL script including system calls to answer set solvers. HAPLO-ASP is available at `http://people.sabanciuniv.edu/~esraerdem/haplo-asp.html`. We have performed two groups of experiments using this system [8].

In these experiments, the executable for SHIPS is obtained from their authors. RPOLY (executables) and HAPLO-IHP (source files) are available at their web pages. We use the versions of these systems available on January 28, 2008. In the experiments, as their search engines, RPOLY uses MINISAT+ (Version 1.0), SHIPS uses MINISAT (Version 2.0), and HAPLO-ASP uses CMODELS (Version 3.74) with LPARSE (Version 1.0.17) and MINISAT (Version 2.0). In our experiments, we have used a workstation with 1.5GHz Xeon processor and 4x512MB RAM, running Red Hat Linux(Version 4.3).

Experimenting with **HIPP** *Problems.* In these experiments, we have compared HAPLO-ASP with the other state-of-the-art haplotype inference systems, RPOLY [6] (based on pseudo-boolean optimization methods) and SHIPS [5] (based on a SAT-based algorithm); these systems can solve **HIPP** problems only. We have excluded from our experiments the systems based on integer linear programming (ILP), such as HYBRIDIP [3], and HAPAR [4] (based on a branch and bound algorithm) since RPOLY and SHIPS perform much better than these systems [5,6].

We have experimented with 334 instances of **HIPP**, used also in the experiments of [3,5,6]: 40 instances generated using MS of [12] (20 *uniform*, 20 *nonuniform*), 294 real instances (24 *hapmap*, 90 *abcd*, 90 *ace*, 90 *ibd*). (These datasets are explained in detail in the cited articles above.) All problem instances are simplified by eliminating duplicates (of genotypes and haplotypes) as described in [5,6] before our experiments. For each haplotype inference system we have assigned 1000 sec.s of CPU time to solve each problem. Table 1 shows the number of problem instances solved by each system. According to this table, HAPLO-ASP solves the most number of problems (332 out of 334 problems). RPOLY aborts for only one problem that HAPLO-ASP solves, but in most of the problems for which it computes a solution in 1000 sec.s it is faster than HAPLO-ASP by a magnitude of up to 300.

Table 1. Number of problems solved, with a timeout of 1000 sec.s for each problem

Group of problems	# of problems	# of problems solved		
		SHIPs	HAPLO-ASP	RPOLY
abcd	90	90	90	90
ace	90	90	90	90
hapmap	24	24	23	23
ibd	90	78	89	88
unif	20	20	20	20
nonunif	20	20	20	20

Both SHIPs and HAPLO-ASP use MINISAT as their search engine. Usually the propositional theory prepared for MINISAT by SHIPs is smaller than the one prepared by HAPLO-ASP.

Experimenting with **HIPAG** *Problems.* We have compared HAPLO-ASP with the haplotype inference system HAPLO-IHP [7] with respect to the computation time and the accuracy of generated haplotypes. We have considered the accuracy measure of [8] to check how much the inferred haplotypes match the original ones. HAPLO-IHP is based on statistical methods and it can compute approximate solutions to instances of **HIPAG** only. The other haplotype inference systems can not solve **HIPAG**.

We have experimented with one of the data sets generated by Yoo et. al for 17 KIR genes of Caucasian population. This data set contains 200 genotypes with 14 biallelic sites. Yoo et al. derived three patterns of haplotypes for this family of genes from the observations in KIR haplotype studies like [10]. As in our experiments with **HIPP** problems, we have first simplified this data set by eliminating the duplicates, and modified the haplotype patterns accordingly. After eliminating the two genotypes that do not match any patterns, the simplified data set contains 28 genotypes with 11 sites. Without the given haplotype patterns, no solution can be found in 30 minutes by HAPLO-IHP; whereas HAPLO-ASP finds 11 haplotypes in 57.08 CPU sec.s, with the accuracy rate 0.702 (by performing a binary search between 1 and 56.) With the given haplotype patterns, HAPLO-IHP finds 19 haplotypes compatible with the given genotypes in 9.1 CPU sec.s, with the accuracy rate 0.732; whereas HAPLO-ASP finds 11 haplotypes in 640 CPU sec.s, with the accuracy rate 0.768. Here 640 sec.s include 1.4 sec.s to infer the set of haplotypes (for $k = 11$), and 631 sec.s to verify its minimality (for $k = 10$). HAPLO-ASP computes an exact solution to **HIPAG**, whereas HAPLO-IHP computes an approximation with a greedy algorithm; this explains the difference between the computation times as well the higher accuracy rate of HAPLO-ASP's solution.

6 Conclusion

We have briefly described HAPLO-ASP, an ASP-based approach to solving various haplotype inference problems, such as **HIPP** and **HIPAG**, possibly by taking into account the given domain-specific information. We have also discussed the range of problems that can be handled by HAPLO-ASP: some of these problems (**WHIPP**

and **WHIPAG**) allow us to specify preferences over parts of haplotypes by assigning weights to their sites/alleles in accordance with their importance; and some problems (**HIPPG**) allow us to handle polyallelic and polyploid genotypes. HAPLO-ASP is the first haplotype inference approach/system that can solve all these variations of haplotype inference. For some of these problems (**HIPP**, **HIPAG**, **HIPPG**), we have illustrated the applicability and effectiveness of HAPLO-ASP on some real data, in comparison with the other existing approaches.

References

1. Gusfield, D.: Haplotype inference by pure parsimony. In: Baeza-Yates, R., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 144–155. Springer, Heidelberg (2003)
2. Lancia, G., Pinotti, M.C., Rizzi, R.: Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. INFORMS Journal on Computing 16(4), 348–359 (2004)
3. Brown, D., Harrower, I.: Integer programming approaches to haplotype inference by pure parsimony. IEEE/ACM Transactions on Bioinformatics and Computational Biology 3, 348–359 (2006)
4. Wang, L., Xu, Y.: Haplotype inference by maximum parsimony. Bioinformatics 19(14), 1773–1780 (2003)
5. Lynce, I., Marques-Silva, J.: Efficient haplotype inference with boolean satisfiability. In: AAAI (2006)
6. Graça, A., Marques-Silva, J.P., Lynce, I., Oliveira, A.: Efficient haplotype inference with pseudo-boolean optimization. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) AB 2007. LNCS, vol. 4545, pp. 125–139. Springer, Heidelberg (2007)
7. Yoo, Y.J., Tang, J., Kaslow, R.A., Zhang, K.: Haplotype inference for present absent genotype data using previously identified haplotypes and haplotype patterns. Bioinformatics 23(18), 2399–2406 (2007)
8. Erdem, E., Ture, F.: Efficient haplotype inference with answer set programming. In: Proc. of AAAI (2008)
9. Neigenfind, J., Gyetvai, G., Basekow, R., Diehl, S., Achenbach, U., Gebhardt, C., Selbig, J., Kersten, B.: Haplotype inference from unphased snp data in heterozygous polyploids based on sat. BMC Genomics 9(1), 356 (2008)
10. Hsu, K.C., Chida, S., Geraghty, D.E., Dupont, B.: The killer cell immunoglobulin-like receptor (kir) genomic region: gene-order, haplotypes and allelic polymorphism. Immunological Reviews 190(1), 40–52 (2002)
11. Pajerowska-Mukhtar, K., Stich, B., Achenbach, U., Ballvora, A., Lübeck, J., Strahwald, J., Tacke, E., Hofferbert, H.R., Ilarionova, E., Bellin, D., Walkemeier, B., Basekow, R., Kersten, B., Gebhardt, C.: Single nucleotide polymorphisms in the allene oxide synthase 2 gene are associated with field resistance to late blight in populations of tetraploid potato cultivars. Genetics 181, 1115–1127 (2009)
12. Hudson, R.: Generating samples under a wrightfisher neutral model of genetic variation. Bioinformatics 18, 337–338 (2002)