Complete Parsimony Haplotype Inference Problem and Algorithms

Gerold Jäger[1], Sharlee Climer[2], and Weixiong Zhang[3]

[1] Computer Science Institute, University of Halle-Wittenberg
D-06120 Halle (Saale), Germany
jaegerg@informatik.uni-halle.de
[2] School of Medicine, Washington University
St. Louis, Missouri 63110-1093, United States
sharlee@climer.us
[3] Department of Computer Science/Department of Genetics
Washington University
St. Louis, Missouri 63130-4899, United States
weixiong.zhang@wustl.edu

Abstract. Haplotype inference by pure parsimony (HIPP) is a well-known paradigm for haplotype inference. In order to assess the biological significance of this paradigm, we generalize the problem of HIPP to the problem of finding all optimal solutions, which we call complete HIPP. We study intrinsic haplotype features, such as backbone haplotypes and fat genotypes as well as equal columns and decomposability. We explicitly exploit these features in three computational approaches which are based on integer linear programming, depth-first branch-and-bound, and a hybrid algorithm that draws on the diverse strengths of the first two approaches. Our experimental analysis shows that our optimized algorithms are significantly superior to the baseline algorithms, often with orders of magnitude faster running time. Finally, our experiments provide some useful insights to the intrinsic features of this interesting problem.

1 Introduction

In this age of rapid advances in biological and medical fields, a number of compelling computational challenges have arisen and propelled the state-of-the-art. Haplotype inference is one such challenge. A *haplotype* is a set of nucleotides that are in physical proximity on a chromosome strand. Haplotypes do not contain nucleotides that have a common state for all individuals within the given population – only those nucleotides that exhibit variation. Diploid species have pairs of chromosomes and, consequently, pairs of corresponding haplotypes. Current sequencers are capable of producing *genotypes*, which are conflations of haplotype pairs. Thanks to recent advances, genotype data are highly accurate and complete. For example, the International HapMap project has produced genotypes that are 99.7% accurate and 99.3% complete [26]. However, identifying individual haplotypes directly in a laboratory setting is currently infeasible for all but

A. Fiat and P. Sanders (Eds.): ESA 2009, LNCS 5757, pp. 337–348, 2009.

small studies. For this reason, researchers commonly rely on mathematical models and computational algorithms for inferring haplotypes from genotypes. The first widely used algorithm for inferring haplotypes was introduced by Clark in 1990 [4]. This algorithm is based on the assumption that the number of unique haplotypes in a given population is relatively small. Gusfield and Hubbell independently proposed a haplotype inference model using *Pure Parsimony* (HIPP), in which the number of unique haplotypes is absolutely minimized [11]. Given a set of genotypes, HIPP is to find a set of distinct haplotypes with minimum cardinality such that each genotype can be resolved by two haplotypes. HIPP was computationally studied using integer linear programming (IP) by Gusfield [9,10]. Due to its simplicity in formulation and complexity in computation, it has attracted much attention. Many approaches to the problem have been developed, for example those based on IP [2,3,12], depth-first branch-and-bound (DFBnB or BnB for short) [20], and Boolean satisfiability [14,15,16]. It is important to mention that there may exist multiple optimal solutions to a HIPP problem and an algorithm simply returns an arbitrary optimal solution. It is also critical to emphasize that not all optimal solutions in the formulation are *biologically* equal. Furthermore, the true solution, which is the ground truth of the haplotype structure of a given population, may not be a computationally optimal solution, as shown in our previous study [5]. In other words, the mathematical formulation of HIPP captures only some of the biological aspects of haplotype inference (HI). Similar to other computational biology problems, such as RNA folding and multiple sequence alignment, the true solutions to HI are often found within in the set of optimal or even near optimal solutions to the HIPP formulation [5]. Therefore, it is desirable to find all optimal solutions [5], which is the subject of this paper. For convenience, we call the problem of finding all optimal solutions to a HIPP problem *complete HIPP*, or CHIPP. The CHIPP formulation and our new algorithm for CHIPP have helped gaining some initial biological insight into haplotype structures in human population [5]. Note that since it requires to find all optimal solutions, CHIPP is computationally more difficult than HIPP, which is $\mathcal{NP}$-complete [13].

In order to develop efficient algorithms for CHIPP, we consider several intrinsic haplotype features in this paper. These include 1) *backbone haplotypes*, which are haplotypes common to all optimal solutions, 2) *decomposability* of a problem into sub-problems, 3) *fat genotypes*, which are genotypes that reduce the solution size by 2, if they are omitted, and 4) *equal columns* [20], which are sites that appear exactly the same for all haplotypes. We then propose and study three classes of algorithms for CHIPP by exploiting these intrinsic haplotype features. The first two classes of algorithms are based on and extend two existing algorithms for HIPP: Gusfield's IP [10] and Wang and Xu's BnB [20]. We extend beyond these existing algorithms by introducing effective optimization techniques that take advantage of the intrinsic haplotype features. The third class of algorithms strategically hybridizes the greatest strengths of the first two.

Note that exploiting these intrinsic haplotype features can also be viewed in the concept of *fixed parameter tractability* (*FPT*). For an overview of FPT

see [7,17]). The motivation of FPT is that even large instances of $\mathcal{NP}$-hard problems can be easy, because they might contain structure that can be exploited. One special idea is *FPT kernelization* which reduces a hard instance by preprocessing to a smaller, equivalent problem kernel [8]. As the intrinsic features *backbone haplotypes*, *decomposability*, *fat genotypes* and *equal columns* lead to a reduction technique, which in a preprocessing step reduces the given CHIPP instance to a smaller one, our approach is also a type of problem kernelization technique.

2 Complete Haplotype Inference by Pure Parsimony

Let G represent a set of genotypes, where $G = [g_1, \ldots, g_n]^T = (g_{ij})_{1 \le i \le n, 1 \le j \le m}$ with $g_{ij} \in \{0, 1, 2\}$, for n individuals with m single nucleotide polymorphism (SNP) sites. Haplotype inference is to find a set of haplotypes, $H = [h_1, \ldots, h_p]^T = (h_{ij})_{1 \le i \le p, 1 \le j \le m}$ with $h_{ij} \in \{0, 1\}$, such that the set of genotypes is *explained*, or *covered*, by the haplotypes. If a site g_{ij} has a value of 0 or 1, it can only be explained by two haplotype sites with values of 0 or 1, respectively. These genotypes are referred to as *homozygous*. A *heterozygous* genotype site has a value of 2, and can only be explained by a haplotype pair with values of 0 and 1. For example, $G = \{g_1 = (1, 2), g_2 = (0, 0), g_3 = (2, 1), g_4 = (2, 2)\}$, will be explained by the haplotype set $H = \{h_1 = (0, 0), h_2 = (1, 0), h_3 = (0, 1), h_4 = (1, 1)\}$. We say that haplotypes h_2 and h_4 *explain* or *cover* genotype g_1 and call h_2 and h_4 an *explaining haplotype pair* for g_1. We also say that both h_2 and h_4 *partly* explain genotype g_1. Note that genotype g_4 can be explained by h_1 and h_4 or by h_2 and h_3. Given a set of genotypes, HIPP is to find a minimum set of unique haplotypes that explains the genotypes. For simplicity and without loss of generality, we assume in this research that all individuals (genotypes) are unique. Notice that a HIPP problem may have multiple optimal solutions [5]. This is evident by a trivial example of one genotype $g_1 = (2, 2)$, which has two optimal solutions, $H_1 = \{(0, 0), (1, 1)\}$ and $H_2 = \{(0, 1), (1, 0)\}$. CHIPP is to find all optimal solutions, i.e. all sets of minimal unique haplotypes covering all genotypes.

In principle, any algorithm for HIPP can lend to an algorithm for CHIPP. We now consider two well-established computational paradigms for HIPP, IP and BnB. In this research, we use these as the baseline algorithms for CHIPP. The pseudo-codes of the algorithms are given as Supporting Information [27].

2.1 CHIPP Algorithm Based on Integer Linear Programming

In 2003, Gusfield [10] developed an exponential-size Integer Linear Program (IP) formulation of HIPP. We now briefly describe this model and discuss how to extend it for solving CHIPP.

Consider n genotypes $g_1, \ldots, g_n$, and haplotypes $h_1, \ldots, h_r$ that cover all n genotypes. Let k_i be the number of possible explaining haplotype pairs for genotype g_i for $i = 1, \ldots, n$. It is easy to see that $k_i = \max\{2^{l-1}, 1\}$ if g_i contains l 2's. Let $x = (x_1, \ldots, x_r)$ be a vector defined as

$$x_i = \begin{cases} 1 & \text{if haplotype } h_i \text{ appears in the HIPP solution,} \\ 0 & \text{else} \end{cases}$$

for $i = 1, \ldots, r$. Further, let $\omega = (\omega_{ij})_{1 \le i \le n, 1 \le j \le k_i}$ be a matrix representing haplotype pairs, defined as

$$\omega_{ij} = \begin{cases} 1 & \text{if the } j\text{-th explaining haplotype pair for} \\ & \quad \text{genotype } g_i \text{ is selected in the HIPP solution,} \\ 0 & \text{else} \end{cases}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, k_i$. In addition, let $f_1(i, j)$ and $f_2(i, j)$ be, respectively, the indices of the first and second haplotype of the j-th explaining haplotype pair for the i-th genotype for $i = 1, \ldots, n$ and $j = 1, \ldots, k_i$. Then HIPP can be represented by the following integer linear program (IP):

$$\min \sum_{s=1}^{r} x_i \text{ subject to} \tag{1}$$

$$\begin{cases} \sum_{j=1}^{k_i} \omega_{ij} = 1 & \text{for } i = 1, \ldots, n \\ x_{f_1(i,j)} \ge \omega_{i,j} & \text{for } i = 1, \ldots, n,\ j = 1, \ldots, k_i \\ x_{f_2(i,j)} \ge \omega_{i,j} & \text{for } i = 1, \ldots, n,\ j = 1, \ldots, k_i \\ x_s, \omega_{i,j} \in \{0, 1\} & \text{for } s = 1, \ldots, r,\ i = 1, \ldots, n,\ j = 1, \ldots, k_i \end{cases}$$

In the worst case, this IP needs an exponential number of variables and constraints. In order to solve CHIPP using the IP representation in (1), we implicitly enumerate all optimal solutions to the IP. The key lies in the idea of avoiding the optimal solutions that have been found earlier in the process. In our method, we first solve HIPP using the IP in (1), obtaining one optimal solution. Let p represent the value of the objective function for this solution. In other words, p unique haplotypes are the minimum number of haplotypes that can resolve this set of genotypes. This means that for this solution, there are exactly p entries of the haplotype index vector x having value 1; let $i_1, \ldots, i_p$ be the indices of these haplotypes. In order to avoid the optimal solution that we just found, we introduce to the IP in (1) the following inequality

$$\sum_{s=1}^{p} x_{i_s} \le p - 1. \tag{2}$$

We then solve the newly expanded IP to find the next optimal solution, if any. Notice that (2) can lead to two possibilities. Either the new IP has an objective value larger than p, which means that no new optimal solution exists; or the new IP has an objective value equal to p, which gives rise to another optimal solution. We repeat this process of adding new inequalities in the form of (2) and solving the incrementally expanded and more constrained new IPs to find all optimal solutions.

2.2 CHIPP Algorithm Based on Branch-and-Bound

Wang and Xu [20] introduced an algorithm for HIPP based on BnB. The algorithm starts with a heuristic solution, where for each genotype the explaining haplotype pair is chosen, from which the corresponding haplotypes can partly explain the most genotypes. This provides the initial incumbent (or upper bound) for the BnB search. Now the search implicitly considers all possible explaining haplotype pairs for each genotype, and the best solution found is the optimal solution to be returned. If during the search the node cost is equal to or exceeds the incumbent, the current explaining haplotype pair is discarded and the algorithm continues to the next explaining haplotype pair, if it exists, thus moving on to the next branch of the current search node.

In our implementation of the above BnB algorithm, we first sorted the genotypes in an increasing order of the numbers of 2's that the genotypes have. In other words, we prefer genotypes with less heterozygous sites over ones with more heterozygous sites at nodes near the top of the search tree. For each genotype, we further order the explaining haplotype pairs, in a decreasing order of the number of genotypes that haplotypes can cover. As in Section 2.1, k_i is the number of possible haplotype pairs for genotype g_i for $i = 1, \ldots, n$.

We can directly extend the BnB algorithm for HIPP to an algorithm for CHIPP. In order to find all optimal solutions, pruning is applied only when the node cost strictly exceeds the incumbent. This allows the algorithm to explore a branch that may lead to another optimal solution. Further, if we have found a better solution, all previous "optimal solutions" are discarded.

3 Features of CHIPP and Optimization Techniques

We now consider some important features of CHIPP, and describe how they can be exploited to develop effective methods for solving this challenging problem.

3.1 Backbones

The *backbone* of a combinatorial optimization problem refers to the set of variables that have common values across all optimal solutions for the problem. Backbones are intrinsic features of combinatorial optimization problems, and have been used to characterize many difficult optimization problems [19,21,23], such as the traveling salesman problem and maximum satisfiability, and have been exploited in algorithms for solving these well-studied problems [6,22,24].

For our purpose, the backbone is a set of haplotypes that appear in every optimal solution of HIPP. We call these haplotypes *backbone haplotypes*. One important consequence of using backbone haplotypes is that those genotypes that can be explained by two backbone haplotypes can be omitted in a haplotype inference procedure, as these genotypes can be explained by any solution that the procedure will return. This holds for HIPP and for CHIPP. We call such genotypes *backbone genotypes*. Therefore, solving CHIPP can be accelerated, if we can identify all backbone haplotypes or backbone genotypes.

A special case of backbone haplotypes is when some genotypes have zero or one SNP site, which can be easily identified. A genotype with no 2's gives rise to one backbone haplotype, which is the same as the genotype. A genotype with only one 2 leads to two backbone haplotypes, which are equal to the genotype except for the site with the 2, where one backbone haplotype has entry 0 and the other has entry 1. We call such backbone haplotypes *trivial backbone haplotypes*. In their BnB program, Wang and Xu [20] implicitly considered trivial backbone haplotypes.

A more difficult problem is to identify all backbone haplotypes. At first sight, the problem seems to be as difficult as finding all optimal solutions, but it turns out that all backbone haplotypes can be identified without finding all optimal solutions. Our idea for the problem is based on the fact that when a backbone haplotype is omitted, no optimal solution to a HIPP or CHIPP problem can be found. Therefore, we first find an optimal solution, and then iteratively remove, one and one at a time, the haplotypes in the solution, and repeatedly solve each of the resulting problems to determine if a new optimal solution can be found. If this is the case, the considered haplotype is no backbone haplotype, otherwise it is a backbone haplotype. In the worst case, the complexity of our algorithm is p times the complexity of finding an optimal solution, where p is the cardinality of the set of an optimal solution haplotypes.

3.2 Equal Column Technique

One important technique used in Wang and Xu's BnB program for HIPP is the so called *equal column technique*. The idea of this technique is as follows. Assume we have found an optimal solution for the genotype matrix $G_1 = [g^1, \ldots, g^k]$, where for $l = 1, \ldots, k$, g^l is the l-th column of G_1, and we want to find an optimal solution for the genotype matrix $G_2 = [g^1, \ldots, g^k, g^{k+1}]$, where $g^{k+1} = g^l$ for some $l \in \{1, \ldots, k\}$. We can simply obtain an optimal solution to G_2 by copying the l-th column of all haplotypes in the optimal solution to the $(k+1)$-th column, because the $(k+1)$-th column of all genotypes can be explained in the same way as the l-th column of all genotypes. It is worthwhile to point out that this technique is also applicable in the same spirit to Gusfield's IP algorithm for HIPP.

However, the equal column technique cannot be directly used to find all optimal solutions, since some optimal solutions might be lost by this technique. This can be seen from the simple example of one genotype $g_1 = (2, 2)$. It has two optimal solutions, $H_1 = \{(0, 0), (1, 1)\}$ and $H_2 = \{(0, 1), (1, 0)\}$. However, the equal column technique will only result in the first optimal solution, because only in this case the first and the second column of the optimal solution are equal. This example also shows that the equal column technique cannot be used to find all backbone haplotypes, as $g_0 = (2)$ has two backbone haplotypes $(0), (1)$ and $g_1 = (2, 2)$ has no backbone haplotype. Therefore, special care must be taken when extending the equal column technique to finding all optimal solutions. Again, let $G_1 = [g^1, \ldots, g^k]$ be a genotype matrix with solution value p and $G_2 = [g^1, \ldots, g^k, g^{k+1}]$ another genotype matrix, where $g^{k+1} = g^l$ for

some $l \in \{1, \ldots, k\}$. The solution value of G_2 is p as well, and each optimal solution for G_2 can be written as an optimal solution for G_1 plus an additional last column. Therefore, given an optimal solution H_1 to G_1, we have to find all optimal solutions H_2 to G_2 which are equal to H_1 in the first k columns. Using the equal column technique for HIPP, we can obtain one optimal solution of G_2, if we use the l-th column of H_1 as $(k+1)$-th column of H_2. However, there may exist additional optimal solutions of G_2. Each such optimal solution has p haplotypes and trivially, each entry of the last column can be 0 or 1. Thus there are a total of 2^p possible optimal solutions for G_2 from which at least one must be in fact optimal. Let H_0 be such a possible optimal solution. Then H_0 is an optimal solution, if and only if all genotypes can be explained by the haplotypes of H_0. So we can make use of this characteristic. Furthermore, we can reduce the number of all possible optimal solutions to be tested, which is 2^p, by the following idea. Each haplotype of an optimal solution must be used to partly explain at least one genotype. We test this characteristic for all $2p$ haplotypes which can possibly appear in an optimal solution. If this characteristic is not fulfilled for a particular haplotype, we do not have to consider this haplotype and all possible optimal solutions which contain this haplotype. The extended equal column technique for CHIPP may still require up to 2^p steps for each equal column in the worst case. Nevertheless, as we will observe in Section 4, it is rather efficient in practice.

Furthermore, columns of a genotype matrix which contain only 0's (1's) can also be omitted for solving HIPP and CHIPP because of the following reason. If a column g^l of the genotype matrix contains only 0 or 1, then the l-th column of the haplotype matrix of each optimal solution also contains only 0's or 1's, respectively.

3.3 Decomposability

Two genotypes g_1 and g_2 may not share any common explaining haplotypes; in this case we say that g_1 does not overlap with g_2. This happens when g_1 has an entry 0 while g_2 has an entry 1, or vice versa, at one site. The concept of non-overlapping of two genotypes can be generalized to two sets of genotypes $E = \{e_1, \ldots, e_s\}$ and $F = \{f_1, \ldots, f_t\}$. If each genotype $e \in E$ does not overlap with any genotype $f \in F$, E does not overlap with F as well.

A HIPP problem instance can be decomposed, if it contains non-overlapping sets of genotypes, where a sub-problem contains one of these sets. It is evident that it is sufficient to solve each of the sub-problems in order to solve the original problem; a solution of the original problem is a union of the solutions to the sub-problems.

This method can be simply extended to CHIPP. Here a problem instance is decomposable if it contains $l > 1$ non-overlapping sets of genotypes, each of which forms a sub-problem. Assume that each sub-problem has q_i solutions for $i = 1, \ldots, l$. The original problem will have $\prod_{i=1}^{l} q_i$ solutions, corresponding to all combinations of the solutions to the sub-problems. A decomposable HIPP or CHIPP problem can be solved with a significantly reduced complexity due to

the potential exponential growth in computation time as a function of the size of the instance.

3.4 Omitting Explaining Haplotype Pairs

RTIP: In order to improve his IP approach, Gusfield [10] used a reduction formulation, called RTIP. RTIP removes from the problem all explaining haplotype pairs in which the two corresponding haplotypes partly explain only one genotype. The genotypes that can only be explained by such haplotype pairs can be explained in the HIPP solution in an arbitrary way. Wang and Xu [20] used a very similar idea in their BnB program. In order to extend this idea to CHIPP, we first introduce the notion of *fat genotypes*.

Fat genotypes: Let G be a set of input genotypes with solution value p, and g a genotype in G such that removing g from G results in a solution to the new instance with solution value $p-2$. We call such a genotype *fat genotype*.

A special case of a fat genotype is a genotype that does not overlap with any other genotype (see Section 3.3) and has at least one entry 2. For example, consider $g_1 = (0,2,0), g_2 = (2,1,0), g_3 = (1,2,1)$, where g_3 does not overlap with g_1 and g_2. Then for $G = \{(g_1, g_2)\}$ the (only) optimal solution is $\{(0,0,0),(0,1,0),(1,1,0)\}$ with solution value 3, whereas for $G' = (g_1, g_2, g_3)$ the (only) optimal solution is $\{(0,0,0),\ (0,1,0),(1,1,0),(1,0,1),\ (1,1,1)\}$ with solution value 5. Thus g_3 is a fat genotype. Note that there are cases of fat genotypes which do overlap with other genotypes. For example, let $g_1 = (1,2,2), g_2 = (2,1,1), g_3 = (2,0,1)$, where g_3 overlaps with g_1. Then for $G = (g_1, g_2)$ the (only) optimal solution is $\{(1,0,0),(0,1,1),\ (1,1,1)\}$ with solution value 3, whereas for $G' = (g_1, g_2, g_3)$ one optimal solution is $\{(1,0,0),(0,1,1),\ (1,1,1),\ (0,0,1),\ (1,0,1)\}$ with solution value 5. Thus g_3 is a fat genotype. It is easy to see that fat genotypes are the only genotypes for which omitting a corresponding explaining haplotype pair by RTIP could cause some optimal solutions to be lost. The fatness of all genotypes can be easily tested.

Our task is to find all optimal solutions that might be lost by RTIP. For this purpose, assume the explaining haplotype pair (h_1, h_2) is omitted by RTIP, but nevertheless is contained in at least one optimal solution. Furthermore, let H_0 be an optimal solution found by RTIP. By the definition of RTIP, h_1 and h_2 partly explain only one genotype g. The only possibility to recover a lost optimal solution which includes h_1 and h_2 is to replace an explaining haplotype pair for g, $(h_3, h_4) \in H_0$, by (h_1, h_2). Let H_0^{new} be H_0 after the replacement. Then H_0^{new} is a new optimal solution, if and only if all genotypes can be explained by haplotype pairs in H_0^{new}. This idea leads to an algorithm which finds all optimal solutions after using RTIP.

If we need to save the (possibly expensive) identification of all fat genotypes, we can try all possible replacements not only for the *fat* genotypes, but for all genotypes. As mentioned, for all non-fat genotypes no further optimal solutions will be found. On the other hand, the time saved by omitting the identification of all fat genotypes can be lost by many more tests for possible new optimal solutions.

Further case. Wang and Xu [20] considered a case of two genotypes g_1 and g_2, where g_1 only has explaining haplotype pairs (h_1, h_2) and (h_4, h_5), and g_2 only has explaining haplotype pairs (h_2, h_3) and (h_5, h_6). In the considered case, all haplotypes $h_1, h_2, h_3, h_4, h_5, h_6$ do not partly explain any other genotype. For this case when solving HIPP, we can use (h_1, h_2) as explaining haplotype for genotype g_1 and (h_2, h_3) as explaining haplotype for genotype g_2, and omit the pairs (h_4, h_5) and (h_5, h_6).

When extending this case to CHIPP, we only have to replace in each optimal solution the haplotypes h_1, h_2, h_3 by the haplotypes h_4, h_5, h_6 to find a new optimal solution.

4 Experimental Analysis

We have implemented in C++ all the different combinations of HIPP and CHIPP algorithms. All our experiments were carried out on a PC with an Athlon 1900MP dual CPU and 2GB shared memory, while our programs ran on a single processor of the machine. As IP solver we used Cplex [25]. An individual experiment was terminated after 6 hours of CPU time. The test data come from two types of human biological data: *genotype* data from the International HapMap Project [26], and *known haplotype* pairs [1,18]. Overall, we use 73 test instances: 66 random instances and 7 known instances. Details about these data and the experimental results can be found in the Supporting Information [27] and in [5].

In Section 2 we introduced two baseline CHIPP algorithms, one based on IP (CHIPP-IP) and the other based on BnB (CHIPP-BnB). Furthermore we consider a hybrid algorithm, in which the heuristic for computing the initial upper bound for a BnB algorithm is replaced by a HIPP algorithm (CHIPP-HY). In other words, the initial upper bound for CHIPP-HY is the cost of the optimal solution, which can significantly reduce the search space to be explored and computation time needed. The optimization techniques that we discussed in Section 3 can be combined in different ways with these three algorithms. These possible combinations give rise to many different algorithms. These optimization techniques or algorithmic components include: equal column technique (**E**) or not; trivial backbones (**T**), all backbones (**A**) or no backbones; RTIP with computation of fat genotypes (**F**), RTIP with no computation of fat genotypes (**R**) or no RTIP. In summary, we have a total of $54 = 3 \times 2 \times 3 \times 3$ algorithms for CHIPP to analyze, where each algorithm is written in the following way: **W-XYZ**, where **W** = **IP** or **W** = **BnB** or **W** = **HY** (hybrid); **X** = **E** or **X** = $\emptyset$; **Y** = **T** or **Y** = **A** or **Y** = $\emptyset$; **Z** = **F** or **Z** = **R** or **Z** = $\emptyset$. For example, **IP-EAF** is used to indicate an algorithm based on IP using the equal column technique, all backbones and RTIP with fat genotypes. Note that for all algorithms with the option **A**, i.e. algorithms that exploit all backbones, a HIPP algorithm is needed. For these algorithms that are based on BnB, the heuristic for the initial upper bound can be replaced by a HIPP algorithm. In this regard, the BnB algorithm and the HY algorithm using option **A** are in fact the same. Finally note that with the purpose to simplify the experiments, we decided to use the technique of

decomposability (Section 3.3) and the further case of Section 3.4 in all versions except the baseline versions. As the CHIPP-HY algorithm contains a HIPP algorithm and finding a single optimal solution is required by the identification of all backbone haplotypes and by the identification of fat genotypes, an efficient HIPP algorithm is an important component of a CHIPP algorithm. Experiments (not described here due to space limit) show that for both HIPP versions, all of the three features equal column technique, RTIP, and trivial backbones lead to a larger efficiency of the HIPP algorithm, where the most important feature is the equal column technique. This is not surprising, as each of these features can substantially reduce the problem sizes. The equal column technique can help remove some sites that carry redundant information, and the techniques of trivial backbones and RTIP can be used to omit some highly constrained explaining haplotype pairs from the core computation of haplotype inference. Furthermore the mentioned experiments show that the HIPP algorithm based on IP is more efficient than that based on BnB. As a consequence of this comparison, we used this HIPP-IP algorithm as a sub-routine in all CHIPP algorithms, where a HIPP algorithm is needed, except one special case. The exception is the algorithm for finding all backbone haplotypes, where we have to omit the equal column technique. As mentioned in Section 3.2, this technique cannot be used for finding all backbone haplotypes. To reduce the overall computation for testing all 54 algorithms and all 73 instances (the 66 random and the 7 known instances), we first tested 8 small- to medium-difficulty typical instances in the first-stage analysis. We found that except for a few cases, most top performers use the all backbone technique without RTIP, or with RTIP with the computation of fat genotypes, i.e. the algorithms with options **A**, **AF**, **EA**, **EAF**. As mentioned, these four algorithms are the same for BnB and HY. Therefore, we have identified eight top contenders for the champion for solving CHIPP. In order to find the overall champion, we further tested these top contenders on all 73 instances. The results clearly show that the optimized algorithms are superior to the baseline algorithms. On many difficult problem instances, the former run orders of magnitude faster than the latter. The results also show that for the baseline algorithms, IP outperforms BnB, which suggests that replacing the heuristic for computing the upper bound by a HIPP algorithm is important. However, the result comparing the optimized IP-based and HY-based algorithms is mixed. On some instances the optimized HY algorithms were able to solve CHIPP, while the IP algorithms failed within the given 6 hours of running time. On the other hand, there are other instances for which the IP algorithms were faster. One IP-based CHIPP algorithm, algorithm **EA**, which uses the techniques of *all backbones* and *equal columns*, is the champion for 19 of the 73 instances tested. This algorithm is also the overall champion. In addition, among the hybrid versions, the algorithm **EA** is also the best one.

5 Summary

In summary, we made three major contributions in this paper. First, we introduced the problem CHIPP to expand the capability of haplotype inference by

pure parsimony for finding all optimal solutions. In [5] extensive experiments showed that CHIPP problem instances can have a large number of optimal solutions and the first optimal solution returned by an algorithm may not necessarily be the true solution. Our results on seven known problem instances also showed that the true solutions to four of these problems are indeed among the optimal solutions. All these results support to find all optimal solutions.

Second, we studied many intrinsic haplotype features, some of which were studied in earlier research on HIPP, particularly by Gusfield [10] and Wang and Xu [20]. However, strategies to exploit these features cannot be directly applied to CHIPP and we formulated methods to recapture solutions lost by these strategies. Furthermore, we introduced the concepts of backbone haplotypes, decomposability and fat genotypes, and formulated and discussed in more detail the concept of equal columns. All these concepts can be viewed as FPT kernelization techniques.

Third, we systematically studied three approaches to CHIPP, one based on integer linear programming [10], another based on depth-first branch-and-bound [20] and one integrating these two. In our algorithms, we explicitly exploited the intrinsic haplotype features that we studied. In our experiments, we analyzed the possible interactions of different problem features and optimization techniques. These studies revealed the best algorithms under these two general problem solving paradigms, as well as the best hybrid algorithms that combines the favorable features of integer linear programming and depth-first branch-and-bound.

Acknowledgement

The research was supported in part by an Olin Fellowship to S.C., an Alzheimers Association grant and two NSF grants (IIS-0535257 and DBI-0743797) to W.Z.

References

1. Andrés, A.M., Clark, A.G., Boerwinkle, E., Sing, C.F., Hixson, J.E.: Assessing the accuracy of statistical haplotype inference with sequence data of known phase. Genet. Epi. 31, 659–671 (2007)
2. Bertolazzi, P., Godi, A., Labbé, M., Tininini, L.: Solving haplotyping inference parsimony problem using a new basic polynomial formulation. Comput. Math. Appl. 55(5), 900–911 (2008)
3. Brown, D.G., Harrower, I.M.: Integer Programming Approaches to Haplotype Inference by Pure Parsimony. IEEE/ACM Transactions on Computational Biology and Bioinformatics 3(2), 141–154 (2006)
4. Clark, A.G.: Inference of Haplotypes from PCR-Amplified Samples of Diploid Populations. Molecular Biology and Evolution 7, 111–122 (1990)
5. Climer, S., Jäger, G., Templeton, A.R., Zhang, W.: How Frugal is Mother Nature with Haplotypes? Bioinformatics 25(1), 68–74 (2009)
6. Climer, S., Zhang, W.: Searching for Backbones and Fat: A Limit-Crossing Approach with Applications. In: Proc. 18th National Conference on Artificial Intelligence (AAAI), pp. 707–712 (2002)

7. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
8. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. SIGACT News 38(1), 31–45 (2007)
9. Gusfield, D.: Inference of Haplotypes from Samples of Diploid Populations: Complexity and Algorithms. J. Computational Biology 8(3), 305–313 (2001)
10. 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)
11. Gusfield, D., Orzack, S.H.: Haplotype Inference. In: Handbook on Bioinformatics (2005)
12. Halldórsson, B.V., Bafna, V., Edwards, N., Lippert, R., Yooseph, S., Istrail, S.: A survey of computational methods for determining haplotypes. In: Istrail, S., Waterman, M.S., Clark, A. (eds.) DIMACS/RECOMB Satellite Workshop 2002. LNCS (LNBI), vol. 2983, pp. 26–47. Springer, Heidelberg (2004)
13. Lancia, G., Pinotti, C.M., Rizzi, R.: Haplotype Populations by Pure Parsimony: Complexity of Exact and Approximation Algorithms. INFORMS J. Computing 16(4), 348–359 (2004)
14. Lynce, I., Marques-Silva, J.: Efficient Haplotype Inference with Boolean Satisfiability. In: Proc. 21st National Conference on Artificial Intelligence (AAAI), pp. 104–109 (2006)
15. Lynce, I., Marques-Silva, J.: SAT in Bioinformatics: Making the Case with Haplotype Inference. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 136–141. Springer, Heidelberg (2006)
16. Lynce, I., Marques-Silva, J., Prestwich, S.: Boosting Haplotype Inference with Local Search. Constraints 13(1-2), 155–179 (2008)
17. Niedermeier, R.: Invitation to Fixed-Parameter Tractability. Oxford University Press, Oxford (2006)
18. Orzack, S.H., Gusfield, D., Olson, J., Nesbitt, S., Subrahmanyan, L., Stanton Jr., V.P.: Analysis and Exploration of the Use of Rule-Based Algorithms and Consensus Methods for the Inferral of Haplotypes. Genetics 165, 915–928 (2003)
19. Slaney, J., Walsh, T.: Backbones in Optimization and Approximation. In: Proc. 17th Intern. Joint Conf. on Artificial Intelligence (IJCAI 2001), pp. 254–259 (2001)
20. Wang, L., Xu, Y.: Haplotype Inference by Maximum Parsimony. Bioinformatics 19(14), 1773–1780 (2003)
21. Zhang, W.: Phase transitions and backbones of 3-SAT and maximum 3-SAT. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 153–167. Springer, Heidelberg (2001)
22. Zhang, W.: Configuration Landscape Analysis and Backbone Guided Local Search: Part I: Satisfiability and Maximum Satisfiability. Artificial Intelligence 158(1), 1–26 (2004)
23. Zhang, W.: Phase Transitions and Backbones of the Asymmetric Traveling Salesman Problem. J. Artificial Intelligence Research 20, 471–497 (2004)
24. Zhang, W., Looks, M.: A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones. In: Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI), pp. 343–350 (2005)
25. Homepage of Cplex, `http://www.ilog.com/products/optimization/archive.cfm`
26. The International HapMap Consortium: A Haplotype Map of the Human Genome. Nature 437, 1299–1320 (2005)
27. Supporting Information to this paper, `http://www.cse.wustl.edu/~zhang/publications/supplemental/ChippSup.pdf`