# Stochastic local search for large-scale instances of the Haplotype Inference Problem by Parsimony

Luca Di Gaspero[1] and Andrea Roli[2]

[1] DIEGM, University of Udine, via delle Scienze 208, I-33100, Udine, Italy
l.digaspero@uniud.it
[2] DEIS, University of Bologna, via Venezia 52, I-47023 Cesena, Italy
andrea.roli@unibo.it

**Abstract.** Haplotype Inference is a challenging problem in bioinformatics that consists in inferring the basic genetic constitution of diploid organisms on the basis of their genotype. This information allows researchers to perform association studies for the genetic variants involved in diseases and the individual responses to therapeutic agents.
A notable approach to the problem is to encode it as a combinatorial problem (under certain hypotheses, such as the *pure parsimony* criterion) and to solve it using off-the-shelf combinatorial optimization techniques. The main methods applied to Haplotype Inference are either simple greedy heuristic or exact methods that, at present, are adequate only for moderate size instances.
In this paper, we present an approach based on the combination of local search metaheuristics and a reduction procedure based on an analysis of the problem structure. Results on a set of Haplotype Inference benchmarks show that this approach achieves a good trade-off between solution quality and execution time.

## 1 Introduction

A fundamental tool of analysis to investigate the genetic variations in a population is based on *haplotype* data. A haplotype is a copy of a chromosome of a diploid organism (i.e., an organism that has two copies of each chromosome, one inherited from the father and one from the mother).

The collection of haplotypes from the genetic material is not an easy task: in fact, due to technological limitations it is currently infeasible to directly collect haplotypes in an experimental way, but rather it is possible to collect *genotypes*, i.e., the conflation of a pair of haplotypes. Therefore, haplotypes have to be inferred from genotypes in order to reconstruct the detailed information and trace the precise structure of human populations. This process is called *Haplotype Inference* and the goal is to find a set of haplotype pairs so that all the genotypes are *resolved*.

Current approaches for solving the problem include simple greedy heuristics [1] and exact methods such as Integer Linear Programming [2,3], Semidefinite Programming [4,5], SAT models [6] and Pseudo-Boolean Optimization algorithms [7]. These approaches, however, at present seem not to be particularly

adequate for very-large size instances. Conversely, we believe that metaheuristic (and hybrid) approaches could provide better scalability than exact algorithms. To the best of our knowledge, the only attempt to employ metaheuristic techniques for the problem is a recently proposed Genetic Algorithm [8]. However, the cited paper does not report results on real size instances.

In this work we present a metaheuristic approach to tackle the Haplotype Inference problem by pure parsimony. We introduce the problem in Section 2 and we sketch an analysis of the problem structure. The outcome of the analysis is a reduction procedure that can be combined with the metaheuristic approach developed in Section 3 in order to improve the performance of local search. Experimental results concerning a comparison of our technique against the state-of-the-art for Haplotype Inference by parsimony are discusses in Section 4.

## 2 The Haplotype Inference problem

In the Haplotype Inference problem we deal with *genotypes*, that is, strings of length $m$ that corresponds to a chromosome with $m$ sites. Each value in the string belongs to the alphabet $\{0, 1, 2\}$. A position in the genotype is associated with a site of interest on the chromosome (called a SNP: single nucleotide polymorphism) and it has value 0 (wild type) or 1 (mutant) if the corresponding chromosome site is a homozygous site (i.e., it has that state on both copies) or the value 2 if the chromosome site is heterozygous. A *haplotype* is a string of length $m$ that corresponds to only one copy of the chromosome (in diploid organisms) and whose positions can assume the symbols 0 or 1 according to the following rules:

$$g[j] = 0 \Rightarrow h[j] = 0 \wedge k[j] = 0 \tag{1}$$
$$g[j] = 1 \Rightarrow h[j] = 1 \wedge k[j] = 1 \tag{2}$$
$$g[j] = 2 \Rightarrow (h[j] = 0 \wedge k[j] = 1) \vee (h[j] = 1 \wedge k[j] = 0) \tag{3}$$

We say that $h$ is a *resolvent* of $g$, and we write $h \trianglelefteq g$, if there exists a companion haplotype $k$ such that $\langle h, k\rangle \triangleright g$. This notation can be extend to sets of haplotypes, and we write $H = \{h_1, \ldots, h_l\} \trianglelefteq g$, meaning that $h_i \trianglelefteq g$ for all $i = 1, \ldots, l$, or to sets of genotypes, in this case we write $h \trianglelefteq A$ if $h \trianglelefteq g$ for all $g \in A$.

Conditions (1) and (2) require that both haplotypes must have the same value in all homozygous sites, while condition (3) states that in heterozygous sites the haplotypes must have different values.

Observe that, according to the definition, for a single genotype string the haplotype values at a given site are predetermined in the case of homozygous sites, whereas there is a freedom to choose between two possibilities at heterozygous places. This means that for a genotype string with $l$ heterozygous sites there are $2^{l-1}$ possible pairs of haplotypes that resolve it.

As an example, consider the genotype $g = (0212)$, then the possible pairs of haplotypes that resolve it are $\langle(0110), (0011)\rangle$ and $\langle(0010), (0111)\rangle$.

The *Haplotype Inference* problem under the *pure parsimony* hypothesis is the problem of finding a set $R$ of $n$ pairs of (not necessarily distinct) haplotypes $R = \{\langle h_1, k_1\rangle, \ldots, \langle h_n, k_n\rangle\}$, so that $\langle h_i, k_i\rangle \triangleright g_i, i = 1, \ldots, n$. We call $H$ the set of haplotypes used in the construction of $R$, i.e., $H = \{h_1, \ldots, h_n, k_1, \ldots, k_n\}$ and our goal is to minimize the cardinality of $H$. It has been shown that this problem is APX-hard [9] and therefore NP-hard.

It is possible to define a graph that expresses the compatibility between genotypes, so as to avoid unnecessary checks in the determination of the resolvents. Let us build the graph $\mathcal{G} = (G, E)$, in which the set of vertices coincides with the set of the genotypes; in the graph, a pair of genotypes $g_1, g_2$ are connected by an edge if they are *compatible*, i.e., one or more common haplotypes can resolve both of them. The same concept can be expressed also between a genotype and a haplotype.

On the basis of the compatibility graph it is possible to devise a reduction procedure whose goal is to try to decrease the number of distinct haplotypes while satisfying the resolution constraint. The intuition behind the procedure is that a possible way of reducing the haplotype number is to resolve a genotype by a haplotype that is compatible, but not currently resolving it. A step of the reduction procedure is described by the following proposition.

**Proposition 1 (Haplotype local reduction).** *Given $n$ genotypes $G = \{g_1, \ldots, g_n\}$ and the resolvent set $R = \{\langle h_1, k_1\rangle, \ldots, \langle h_n, k_n\rangle\}$, so that $\langle h_i, k_i\rangle \triangleright g_i$. Suppose there exist two genotypes $g, g' \in G$ such that $g \triangleleft \langle h, k\rangle$, $g' \triangleleft \langle h', k'\rangle$, $h$ is compatible also with $g'$ and $h \neq h'$, $h \neq k'$, $h' \trianglelefteq A$, $k' \trianglelefteq B$.*

*The replacement of $\langle h', k'\rangle$ with $\langle h, g' \ominus h\rangle$[3] in the resolution of $g'$ is a correct resolution that employs a number of distinct haplotypes according to the following criteria:*

- *if $|A| = 1$ and $|B| = 1$, the new resolution uses* at most *one less distinct haplotype;*
- *if $|A| > 1$ and $|B| = 1$ (or symmetrically, $|A| = 1$ and $|B| > 1$), the new resolution uses* at most *the same number of distinct haplotypes;*
- *in the remaining case the new resolution uses* at most *one more distinct haplotype.*

*Proof.* The proof of the proposition is straightforward. The resolution is obviously correct because $h$ is compatible with $g'$ and $g' \ominus h$ is the complement of $h$ with respect to $g'$.

Concerning the validity of the conditions on the cardinality, let us proceed by cases and first consider the situation in which $g' \ominus h$ does not resolve any other genotype but $g'$.

If $|A| = |B| = 1$, then $h'$ and $k'$ are not shared with other genotype resolutions so they will not appear in the set $H$ after the replacement, therefore since in the new resolution $h$ is shared between $g$ and $g'$ the cardinality of $H$ is decreased by one.

[3] With $g' \ominus h$ we denote the complementary haplotype of $h$ w.r.t. $g$. It is straightforward to prove that such a haplotype exists and is unique.

Conversely, if one of the sets $|A|$ or $|B|$ consists of more than a genotype and the other set of just one genotype, there is no guarantee of obtaining an improvement from the replacement. Indeed, since one of the two haplotypes is already shared with another genotype there is just a replacement of the shared haplotype with another one in the set $H$.

Finally, when $|A| > 1$ and $|B| > 1$ both $h'$ and $k'$ are shared with other genotypes therefore the replacement introduces the new haplotype $g' \ominus h$ in the set $H$.

Moving to the situation in which $g' \ominus h$ resolves also other genotypes, the same considerations apply; additionally, given that $g' \ominus h$ is already present in $H$, the number of distinct haplotypes employed in the resolution is decreased by one. For this reason the estimation of the changes of $|H|$ is conservative. □

Even though in principle the reduction procedure can be employed with any selective solution method (such as Local Search or Genetic Algorithms), in this paper we decided to focus on a tabu search algorithm which seemed to be very promising.

## 3 Local Search techniques for Haplotype Inference

As the search space for this problem we adopt a *complete* representation of the genotype resolution. That is, we consider, for each genotype $g$, the pair of haplotypes $\langle h, k \rangle$ that resolves it. In this representation all the genotypes are fully resolved at each state by construction. The search space is therefore the collection of sets $R$ defined as in the problem statement. The complete representation has the advantage of allowing to design anytime algorithms, since the search can be interrupted any moment and return a feasible solution, i.e., a set (not necessarily minimal) of haplotypes that resolve the given genotypes.

For the cost function, we identify different components related either to optimality or to heuristic measures. A natural component is the objective function of the original problem, that is the cardinality $|H|$ of the set of haplotypes employed in the resolution. Moreover, we also include some heuristic related to the potential quality of the solution, namely the number of incompatible sites between each genotype/haplotype pair. The cost function $F$ is then the weighted sum of the two components.

We designed a family of local search strategies, namely *Best improvement*, *Stochastic first improvement*, *Simulated annealing*, and *Tabu search*. The techniques are instances of the general strategies described in [10]. All of them start with a set of haplotypes of cardinality $2n$, where $n$ is the number of genotypes, and they explore the search space by iteratively modifying pairs of resolving haplotypes trying to reduce the number of distinct ones. Best improvement and Stochastic first improvement traverse the search space by moving from a state to a neighboring one with a lower cost function value, by choosing the best and first neighbor respectively. Simulated annealing moves also to worse states than the current one, on the basis of a probabilistic choice function. Finally, Tabu search

behaves in principle like Best improvement but restricts the neighborhood by forbidding recently performed moves.

Local search moves are defined upon a Hamming neighborhood function. A good trade-off between exploration and execution time is the 1-Hamming distance neighborhood w.r.t. each haplotype in the current solution. This kind of move can be thought as a *flip*, performed at a given position in a pair of haplotypes resolving a given genotype. The complete exploration of such a neighborhood has a time complexity bounded from above by $O(nk)$, where $k$ is the number of haplotypes and $n$ the number of sites per haplotype. In practice, the time complexity can be further reduced by restricting the number of neighbors to heterozygous sites and haplotypes resolving non isolated genotypes.

## 4 Experimental results

We developed a set of local search solvers (Tabu Search, Hill Climbing and Simulated Annealing) using EasyLocal++ [11], a framework for the development of local search algorithms. The algorithms have been implemented in C++ and compiled with *gcc 3.2.2* and run on a Intel Xeon CPU 2.80GHz machine with SUSE Linux 2.4.21-278-smp. Each algorithm was run on every instance one time and we allotted 300 seconds for each execution of the algorithms. Since Tabu search (TS) showed superior performance over the other local search algorithms, we only discuss results of this technique.

Our Tabu search implementation considers as tabu all the moves that insist on a pair of haplotypes that recently changed. The tabu list scheme adopted is a *dynamic* one, that is for each move performed we consider it as prohibited for a number of iterations that randomly varies between two values $k_{min}$ and $k_{max}$. The values of these parameters were chosen according to the results of an exploratory analysis based on the $F$-Race method [12], and were set to $k_{min} = 10$, $k_{max} = 20$. These settings have shown to be quite robust across the variety of instances tested. Moreover, since the algorithm that incorporates the initial graph reduction sharply outperforms the one without graph reduction, we report only the results of the former one.

The benchmark instances are composed of two parts. The first one, composed of the sets Harrower uniform, Harrower non-uniform and Harrower hapmap, is the benchmark used in [3]. The second part of the instances, namely Marchini SU1, Marchini SU2, Marchini SU3 and Marchini SU-100kb, were taken from the website `http://www.stats.ox.ac.uk/~marchini/phaseoff.html`.

The main characteristics of the instance sets are summarized in Table 1.

In order to estimate the quality of solutions produced by TS, we need to compute the optimal solution of the benchmark instances. We tackled the instances with *rpoly* [7], a state-of-the-art exact solver for the Haplotype Inference. The solver is run on the same benchmark instances and on the same machine. We allotted *rpoly* 24 hours of computation for each instance. The instances of the set Harrower uniform, Harrower non-uniform, Harrower hapmap, Marchini SU1 and Marchini SU2 were completely solved. From Marchini SU3 and Mar-

Table 1: A summary of the main characteristics of the benchmarks.

| **Benchmark set** | **N. of instances** | **N. of genotypes** | **N. of sites** |
|---|---|---|---|
| Harrower uniform | 200 | 10÷100 | 30÷50 |
| Harrower non-uniform | 90 | 10÷100 | 30÷50 |
| Harrower hapmap | 24 | 5÷68 | 30÷75 |
| Marchini SU1 | 100 | 90 | 179 |
| Marchini SU2 | 100 | 90 | 171 |
| Marchini SU3 | 100 | 90 | 187 |
| Marchini SU-100kb | 29 | 90 | 18 |

Table 2: Fraction of instances solved by *rpoly* from each benchmark.

| **Benchmark set** | Fraction of solved instances | **Benchmark set** | Fraction of solved instances |
|---|---|---|---|
| Harrower uniform | 200/200 | Marchini SU1 | 100/100 |
| Harrower non-uniform | 90/90 | Marchini SU2 | 100/100 |
| Harrower hapmap | 24/24 | Marchini SU3 | 89/100 |
| | | Marchini SU-100kb | 23/29 |

chini SU-100kb only a portion of the instances were solved. Overall, most of the instances could be solved with a runtime higher than 12 hours per instance. A summary of the fraction of solved instances is reported in Table 2.

The plots in Figure 1 report the comparison between the TS and *rpoly*; a point $(x, y)$ in the plot represents the number of haplotypes in the best solution returned by TS and *rpoly*, respectively. A point below the line means that the solution returned by the algorithm corresponding to the $y$-axis is better than the one returned by the algorithm associated to the $x$-axis.

Notice that the solution quality achieved by TS approximates the optimal one returned by *rpoly* on some benchmarks, namely Harrower sets and Marchini SU-100kb, whilst the performance on Marchini SU2 is considerably inferior. The performance on benchmarks Marchini SU1 and Marchini SU3 is inferior, but it has to be taken into account that TS returned a feasible solution to all the instances of the sets, whilst *rpoly* solved only a fraction of the instances of Marchini SU3. We also observe that our approach scales very smoothly.

These results enlighten the complementarity of the two approaches: the algorithm that also returns the proof of optimality is definitely preferable over the incomplete one when the execution time allotted can be large, while we can resort to the approximate algorithm to have a feasible and (hopefully) near-optimal solution in very short time.

## References

1. A. G. Clark, Inference of haplotypes from PCR-amplified samples of diploid populations, Molecular Biology and Evolution 7 (1990) 111–122.

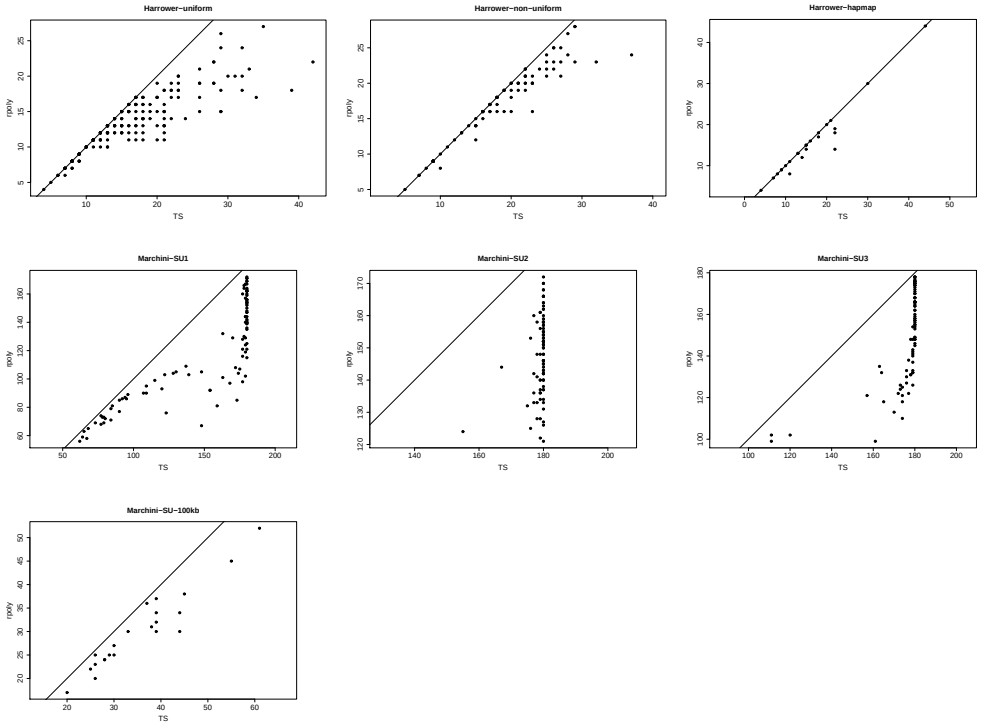


Fig. 1: Comparison between TS and rpoly in terms of number of haplotypes.


2. D. Gusfield, Haplotype inference by pure parsimony., in: R. A. Baeza-Yates, E. Chávez, M. Crochemore (Eds.), Combinatorial Pattern Matching (CPM 2003), Proceedings of the 14th Annual Symposium, Vol. 2676 of Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg, Germany, 2003, pp. 144–155.
3. D. G. Brown, I. M. Harrower, Integer programming approaches to haplotype inference by pure parsimony., IEEE/ACM Transactions on Computational Biology and Bioinformatics 3 (2) (2006) 141–154.
4. K. Kalpakis, P. Namjoshi, Haplotype phasing using semidefinite programming., in: BIBE, IEEE Computer Society, 2005, pp. 145–152.
5. Y.-T. Huang, K.-M. Chao, T. Chen, An approximation algorithm for haplotype inference by maximum parsimony., in: H. Haddad, L. M. Liebrock, A. Omicini, R. L. Wainwright (Eds.), Proceedings of the 2005 ACM Symposium on Applied Computing (SAC 2005), ACM, 2005, pp. 146–150.
6. I. Lynce, J. Marques-Silva, Efficient haplotype inference with boolean satisfiability., in: Proceedings of the 21st National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, AAAI Press, Menlo Park, CA, USA, 2006.
7. A. Graça, J. Marques-Silva, I. Lynce, A. L. Oliveira, Efficient haplotype inference with pseudo-boolean optimization, in: H. Anai, K. Horimoto, T. Kutsia (Eds.),

Algebraic Biology, Second International Conference, AB 2007, Castle of Hagenberg, Austria, July 2-4, 2007, Proceedings, Vol. 4545 of Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg, Germany, 2007, pp. 125–139.
8. R.-S. Wang, X.-S. Zhang, L. Sheng, Haplotype inference by pure parsimony via genetic algorithm, in: X.-S. Zhang, D.-G. Liu, L.-Y. Wu (Eds.), Operations Research and Its Applications: the Fifth International Symposium (ISORA'05), Tibet, China, August 8–13, Vol. 5 of Lecture Notes in Operations Research, Beijing World Publishing Corporation, Beijing, People Republic of China, 2005, pp. 308–318.
9. G. Lancia, M. C. Pinotti, R. Rizzi, Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms., INFORMS Journal on Computing 16 (4) (2004) 348–359.
10. C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys 35 (3) (2003) 268–308.
11. L. Di Gaspero, A. Schaerf, EasyLocal++: An object-oriented framework for flexible design of local search algorithms, Software—Practice and Experience 33 (8) (2003) 733–765.
12. M. Birattari, T. Stützle, L. Paquete, K. Varrentrapp, A racing algorithm for configuring metaheuristics, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), Morgan Kaufmann Publishers, New York (NY), USA, 2002, pp. 11–18.