# Towards a highly scalable hybrid metaheuristic for haplotype inference under parsimony

Stefano Benedettini, Luca Di Gaspero
DIEGM, Università di Udine
via delle Scienze 208, I-33100, Udine, Italy
stefano.benedettini2@studio.unibo.it,
l.digaspero@uniud.it

Andrea Roli
DEIS, Campus of Cesena
*Alma Mater Studiorum* Università di 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 piece of 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*) and to solve it using off-the-shelf combinatorial optimization techniques. In this paper, we present and discuss an approach based on hybridization of two metaheuristics, one being a population based learning algorithm and the other a local search. We test our approach by solving instances from common Haplotype Inference benchmarks. Results show that this approach achieves an improvement on solution quality with respect to the application of a single "pure" algorithm.*

## 1. Introduction

The role of genetic variation and inheritance in human diseases is extremely important, though still largely unknown [12]. To this aim, the assessment of a full Haplotype Map of the human genome has become one of the current high priority tasks of human genomics [11]. A haplotype is one of the two non identical copies 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 information haplotypes convey allows to perform association studies for the genetic variants involved in diseases and the individual responses to therapeutic agents.

The most important variations are the *Single Nucleotide Polymorphisms* (SNPs), which occur when a nucleotide in the DNA sequence is replaced by another one. Technological limitations make it currently impractical to directly collect haplotypes by experimental procedures, but it is possible to collect *genotypes*, i.e., the conflation of a pair of haplotypes. Moreover, instruments can easily identify only whether the individual is *homozygous* (i.e., the alleles are the same) or *heterozygous* (i.e., the alleles are different) at a given site.

Therefore, haplotypes have to be inferred from genotypes in order to reconstruct the detailed information and trace the precise structure of DNA variations in a population. This process is called *Haplotype Inference* (also known as *haplotype phasing*) and the goal is to find a set of haplotype pairs (i.e., a *phasing*) so that all the given genotypes are *resolved*, that is, they can be obtained by combining a pair of haplotypes from the set.

The main methods to tackle the Haplotype Inference are either combinatorial or statistical. Both, however, being of non-experimental nature, need some genetic model that provides criteria for evaluating the solution returned with respect to actual genetic plausibility. In the case of the combinatorial methods, which are the subject of the present work, the most often used criterion is *pure parsimony* [9], which suggests to search for the smallest collection of distinct haplotypes that solves the Haplotype Inference problem. This criterion is consistent with current observations in natural populations for which the actual number of haplotypes is vastly smaller than the total number of possible haplotypes.

The method we present is a hybrid metaheuristic[1] that tackles the Haplotype Inference problem as a constrained optimization problem with an objective function defined by the *pure parsimony* hypothesis. The proposed approach extends and integrates the solution techniques presented in two previous works [1, 5], namely a population-based learning method (ACO) and a local search (tabu search).

Current approaches for solving the problem under the

[1]For an introduction to metaheuristics, see, e.g., [3]

pure parsimony hypothesis are mainly based on exact methods (e.g., [4, 8]). At present, complete approaches are very effective but they seem not to be particularly adequate for very-large size instances. Hence the need for approximate algorithms, such as metaheuristics, that trade completeness for efficiency. Moreover, a motivation for studying and applying approximate algorithms is that the criterion used to evaluate the solutions provide an approximation of the actual solution quality, therefore a proof of optimality is not particularly important.

The remainder of this paper is structured as follows. We formally introduce the problem in Section 2. In Section 3 we describe our integration of the proposed techniques and briefly explain the two algorithms (sections 3.1 and 3.2). Experimental results are discussed in Section 4. Finally, we discuss some possible improvements and future work in Section 5.

## 2. The Haplotype Inference problem

In the Haplotype Inference problem we deal with *genotypes*, i.e., strings of length $m$ that correspond to a chromosome with $m$ sites. Each value in the string belongs to the set $\{0, 1, 2\}$. A position in the genotype is associated with a site of interest on the chromosome (e.g., a SNP) 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 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 $\{0, 1\}$.

### 2.1. Genotype resolution

Given a chromosome, we are interested in finding an unordered pair of haplotypes that can explain the chromosome according to the following definition:

**Definition 1 (Genotype resolution)** *Given a chromosome $g$, we say that the unordered pair $\langle h, k\rangle$* resolves *$g$ (or, alternatively, that $h$ and/or $k$ are* resolvents *of $g$), and we write $h \oplus k = g$ (or $g = h \oplus k$), if the following conditions hold (for $j = 1, \ldots, m$):*

$$g[j] = 0 \vee g[j] = 1 \Rightarrow \quad h[j] = g[j] \wedge k[j] = g[j] \quad \text{(1a)}$$
$$g[j] = 2 \Rightarrow \quad h[j] \neq h[j] \quad \text{(1b)}$$

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.

After these preliminaries we can state the *Haplotype Inference* problem as follows:

**Definition 2 (Haplotype Inference problem)** *Given a population of $n$ individuals, each of them represented by a genotype string $g_i$ of length $m$ we are interested in finding a set $\phi$ of $n$ pairs of (not necessarily distinct) haplotypes $\phi = \{\langle h_1, k_1\rangle, \ldots, \langle h_n, k_n\rangle\}$, so that $h_i \oplus k_i = g_i, i = 1, \ldots, n$. We call $H$ the set of haplotypes used in the construction of $\phi$, i.e., $H = \{h_1, \ldots, h_n, k_1, \ldots, k_n\}$.*

From the mathematical point of view, there are many possibilities for building the set $H$, since there is an exponential number of possible haplotypes for each genotype. Therefore, a criterion should be added to the model for evaluating solution quality.

The *pure parsimony* approach consists in searching for a solution that minimizes the total number of distinct haplotypes used or, in other words, $|H|$, the cardinality of the set $H$. A trivial upper bound for $|H|$ is $2n$ in the case of all genotypes resolved by a pair of distinct haplotypes. It has been shown that the Haplotype Inference problem under the pure parsimony hypothesis is NP-hard [10].

It is important to stress, at this point, that finding a proven optimal solution is not particularly relevant, because the criterion defining the objective function are an approximation of an (unknown) actual quality function. Therefore, approximate approaches that are able to return solutions of a good quality, even if not optimal, are of notable practical importance.

### 2.2. Compatibility and complementarity

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.[2] In the graph $\mathcal{G} = (G, E)$, the set of vertices coincides with the set of the genotypes; 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 formal definition of this property is as follows.

**Definition 3 (Compatibility)** *Let $g_1$ and $g_2$ be two genotypes, $g_1$ and $g_2$ are* compatible *if, for all $j = 1, \ldots, m$, the following conditions hold:*

$$g_1[j] = 0 \quad \Rightarrow \quad g_2[j] \in \{0, 2\} \quad \text{(2a)}$$
$$g_1[j] = 1 \quad \Rightarrow \quad g_2[j] \in \{1, 2\} \quad \text{(2b)}$$
$$g_2[j] = 2 \quad \Rightarrow \quad g_2[j] \in \{0, 1, 2\} \quad \text{(2c)}$$

[2]In some cases, also a graph representing incompatibilities between genotypes can provide useful information.

*The same concept can be expressed also between a genotype $g$ and a haplotype $h$: we say that they are* compatible *if, for all $j = 1, \dots, m$, the following conditions hold:*

$$g[j] = 0 \vee g[j] = 1 \quad \Rightarrow \quad h[j] = g[j] \qquad \text{(3a)}$$
$$g[j] = 2 \quad \Rightarrow \quad h[j] \in \{0, 1\} \qquad \text{(3b)}$$

Notice that the set of compatible genotypes of a haplotype can contain only mutually compatible genotypes (i.e., they form a clique in the compatibility graph).

Another interesting observation is the following. Due to the resolution definition, when one of the two haplotypes composing the pair, say $h$, has been selected, then the other haplotype can be directly inferred from $h$ and the genotype $g$ by means of to the resolution conditions as expressed by the following property:

**Proposition 1 (Haplotype complement)** *Given a genotype $g$ and a haplotype $h$ compatible with $g$, there exists a unique haplotype $k$ such that $h \oplus k = g$. The haplotype $k$ is called the* complement *of $h$ with respect to $g$ and is denoted with $k = g \ominus h$.*

## 3. A Hybrid Metaheuristic

Here we show the integration between the ant-based algorithm presented in [1] and a variant of the local search proposed in [5]. In this stage of work we decided to keep the structure of the hybrid metaheuristic fairly simple (see section 5): the ant-based algorithm returns a solution which then becomes the initial state of the tabu search.

We now briefly describe the two algorithms.

### 3.1. Ant-based Algorithm

The ant-based algorithm is a population method that follows the well-renowned Ant Colony Optimization metaheuristic [7] and specifically adopts the Hypercube framework [2] for the pheromone update function.

ACO-HI (Algorithm 1) is a stochastic constructive procedure that operates on two levels: on the higher level an ACO for finding a good visiting order of genotypes is run while on the lower level, another ACO algorithm searches for the haplotypes to be added to $H$. The two levels are of course coupled, as the order in which genotypes are considered is influenced by the current set of haplotypes in $H$ and, conversely, a generic step in the construction of $H$ depends on the previously resolved genotypes.

Before applying the two-level ACO, the problem instance is preprocessed by a procedure that eliminates replicates among genotypes and identifies disconnected parts in the compatibility graph that can then be treated as independent instances.

**Algorithm 1** ACO-HI

```
1:  Preprocessing phase
2:  while terminating conditions not met do
3:    for all $a \in A$ do
4:      while not all genotypes are resolved do
5:        $g \leftarrow$ chooseNode($G$)
6:        resolve genotype $g$
7:        propagate resolvents
8:      end while
9:    end for
10:   pheromoneUpdate()
11: end while
```

**Lower level: genotype resolution.** An ant $a$ builds a solution by considering in turn each genotype $g \in \mathcal{G}$ (the order is defined in the higher-level) and finding resolvent haplotypes for it. When a new resolvent has to be added to $H$, the values of its heterozygous sites, either 0 or 1, are chosen on the basis of pheromone values[3] and a heuristic strategy.

This strategy (fully detailed in [1]) constrains the possible choices of a value and aims at minimizing the number of haplotypes to be added to the solution, for example by forcing the new resolvent to be compatible with a neighbor of $g$ in the compatibility graph. In those sites unconstrained by the heuristic strategy, we have a pheromone guided binary choice: the value is chosen with a probability proportional to its pheromone value.

**Higher level: genotypes visiting order.** The order in which genotypes are visited has a strong influence on solution quality, therefore the higher level of the algorithm tries to learn a good genotype visiting order. This learning mechanism is primarily guided by pheromone associated to the edges of the compatibility graph. In this way, pairs of consecutive genotypes in the series are learnt. Formally, every edge $(i, j)$ of the compatibility graph is associated to a pheromone value $\tau_{ij}$ and the probability to move from node $i$ to node $j$ is given by:

$$p(i,j) = \begin{cases} \frac{\tau_{ij}}{\sum_{l \in adj(l)} \tau_{il}}, & \text{if } j \in adj(i) \\ \frac{1}{|U|}, & \text{if } j \in U \wedge adj(i) = \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $adj(i)$ is the set of nodes adjacent to $i$ (i.e., the compatible genotypes) still unresolved and $U$ is the set of currently unvisited genotypes not compatible with genotype corresponding to $i$. In such a way, if $i$ has adjacent unresolved nodes, then one among them is chosen according to pheromone values; otherwise, the next genotype in the

[3]Homozygous sites do not represent choice points as they are directly assigned because the haplotype we are constructing must resolve $g$.

sequence is chosen randomly among the remaining unresolved genotypes.

**Pheromone update.** As mentioned before, our algorithm is implemented according to the Hyper-cube framework. The objective function of the problem is the cardinality of $H$, that has to be minimized. Therefore, as a quality function used for the reinforcement, we chose the function $F(H) = 2n - |H|$.

Pheromone is updated in the two levels with the same evaporation parameter and quality function. The only difference is that the solution components of the higher level are edges of the compatibility graph, while in the lower level they are nodes representing values to assign to haplotype sites.

### 3.2. Local Search

Local search is a search process that iteratively modifies the current candidate solution by applying move operators trying to follow trajectories in the search space leading to good solutions.

The design process of local search metaheuristics involves the definition of the local search model, which includes the specification of the *search space*, the *cost function*, and the *neighborhood relation*, and the choice of a search strategy. In the following we detail our design and implementation choices .

**Search space.** In our approach, the search space is composed of the pairs of haplotypes $\langle h, k\rangle$ that resolves genotype $g$, for all $g \in G$. Therefore in this representation all the genotypes are completely resolved at each state by construction. Thus, the search space is the collection of sets $\phi$ defined as in the problem statement.

**Cost function.** The cost function is a measure of solution quality including components related both to the solution evaluation criteria of choice and to heuristic information that could guide search toward good solutions.

The component related to the evaluation criterion is an objective function defined as the cardinality $|H|$ of the set of haplotypes employed in the resolution.

$$f_1^{\text{par}}(\phi) = |H| \tag{5}$$

Moreover, we also include a heuristic measure related to the potential quality of the solution. In this respect, we counted the number of incompatible sites (to be minimized) between each genotype/haplotype pair:

$$f_2 = \sum_{h\in H}\sum_{g\in G}\sum_{j=1}^{m} \chi(h[j] \text{ is not compatible with } g[j]) \tag{6}$$

In the formula, $\chi$ denotes the truth indicator function, whose value is 1 when the proposition in parentheses is true and 0 otherwise.

The cost function $F$ is the weighted sum of the two components $F = \alpha_1 f_1^{\text{par}} + \alpha_2 f_2$, in which the weights $\alpha_1$ and $\alpha_2$ specify the trade-offs between the different components. In our experimentation we chose the values $\alpha_1 = \alpha_2 = 1$.

**Neighborhood relation and search strategy.** Differently from [5], where two of the authors defined the local search neighborhood on the basis of the unitary Hamming distance between haplotypes, the move we present in this work explicitly takes into account the compatibility graph.

The neighborhood of a solution is defined by considering all the pairs of compatible genotypes and enumerating all their possible resolving haplotypes. In the worst case, the complexity of this operation is $O(n^2 m)$. However, in practice the complexity is inferior, since the number of resolving haplotypes for a given pair of compatible genotypes is determined by the number of compatible sites (usually much less than $m$) and the number of compatible genotypes pairs is usually less than $n^2$. This move considerably improves the solver performance with respect to our previous work.

As for the search strategy, we designed a stochastic local search technique, based on the *tabu search* metaheuristic template. The algorithm is defined in Figure 2. The algorithm starts with a phasing supplied by the ant algorithm and possibly further reduced by means of a reduction procedure [5] which tries to remove from $H$ haplotypes that are not necessary to resolve some genotype.

After this preprocessing phase, the solver explores the search space by iteratively modifying pairs of resolving haplotypes trying to reduce $F$. Then, the iterative process is repeated as long as a termination criterion is met: in our implementation, we allotted a maximum number of non-improving iterations. Tabu search explores all the neighbors of the incumbent solution $s$ that can be reached by applying moves that are not in the tabu list and chooses the best neighbor (lines 5–7). A move is tabu if it or its inverse have been applied in the last $tl$ iterations.

The main strength of the local search we designed is to be found in the move operator, that exploits the instance structure (namely the compatibility graph) in order to consider the most promising neighbors of a solution. Unfortunately, this move is not always applicable on instances that show a particular structure of the compatibility graph; on these instances, in which the algorithm fails prematurely, the initial solution is returned.

## 4. Experimental results

The algorithms have been developed in C++, exploiting the EASYLOCAL++ framework for the local search com-

**Algorithm 2** TS-HI$^+$

```
1:  $s \leftarrow$ FetchInitialSolution()
2:  $s \leftarrow$ Reduce($s$)
3:  $s_b \leftarrow s$
4:  **while** termination conditions not met **do**
5:      $\mathcal{N}_a(s) \leftarrow \{s' \in \mathcal{N}(s) \mid s'$ does not violate the tabu condition$\}$
6:      $s' \leftarrow \text{argmin}\{F(s'') \mid s'' \in \mathcal{N}_a(s)\}$
7:      $s \leftarrow s'$        {i.e., $s'$ replaces $s$}
8:      **if** $F(s) < F(s_b)$ **then**
9:          $s_b \leftarrow s$
10:     **end if**
11: **end while**
12: Return best solution found $s_b$
```

| label | type | pop. size | max iter | max idle |
|---|---|---|---|---|
| ACO-HI | ant-based | 75 | 400 | 200 |
| TS-HI$^+$ | tabu search | $\|tl\| \in$ [10, 20] | | 600 |
| Hybrid$_1$ = ACO-HI$\triangleright$TS-HI$^+$ | hybrid | 75 | 400 | 200 |
| Hybrid$_2$ = ACO-HI$\triangleright$TS-HI$^+$ | hybrid | 50 | 200 | 100 |

Table 1: Parameters of the solvers (the evaporation factor of ACO-HI is set to 0.1).

ponent [6]. The software was compiled with the GNU g++ compiler v. 4.1.3 and tested on a Intel Pentium 4 3.0GHz machine running Ubuntu 7.10 (kernel 2.6.22).

We compared the pure ACO-HI and TS-HI$^+$ solvers with two hybridizations, as presented in Table 1. The two hybrid solvers differ only with respect to the computational resources needed, Hybrid$_2$ being more frugal. For the TS-HI$^+$ solver we employ as initial solution the solution returned by the ant algorithm after a single iteration. The algorithms were tuned according to the parameter setting reported in the table.

The sets of instances employed in the experimental analysis are Harrower uniform and Harrower hapmap – taken from [4] – and Marchini SU1, Marchini SU2, Marchini SU3 and Marchini SU-100kb – downloadable from `http://www.stats.ox.ac.uk/~marchini/phaseoff.html`. The major characteristics of benchmark instances are shown in Table 2.

| Benchmark set | N. of instances | N. of genotypes | N. of sites |
|---|---|---|---|
| Harrower uniform | 200 | 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: Main features of the benchmarks.

The results of the experiments are presented in Figure 1. The boxplots collects the data of 10 independent runs of the algorithms on all the instances of each set. The solution value plotted is the average cardinality of haplotype set summed up over all the instances of each benchmark. Table 3 summarizes the statistics (averages and standard deviations) on the cumulative running time for solving all the instances of each set.

Experimental results show that even a coarse integration of the two basic algorithms improves solution quality. Moreover, the results of Hybrid$_2$ solver demonstrate that reducing the strength of the ant algorithm can substantially decrease running times, at the expense of a modest loss in the solution quality. This can be explained by the fact that population algorithms are good at finding promising areas in the search space, rapidly improving the solution in the early stages of the search, but they are not able to explore promising areas intensifying around good solutions.Therefore, the winning strategy would be stopping earlier the computational expensive ant algorithm, find a good initial state for the local search, and then let the fast tabu search improve the solution.

Harrower hapmap and Marchini SU-100kb instance sets exhibit an unusual behavior: in those instances the ant algorithm is dominated even by the pure local search. This behavior is due to the particular structure of the instances – we conjecture that is due to the sparseness of their compatibility graph. Therefore, we believe that a hybrid approach could be particularly useful in general, since each component compensates for each other weaknesses.

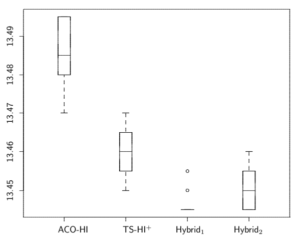
## 5. Conclusions and future works

We have presented a hybrid metaheuristic approach to tackle the Haplotype Inference problem that combines an ACO-based algorithm and a local search. The hybrid algorithm improves its basic components and it can be tuned to balance the trade-off between execution time and accuracy.

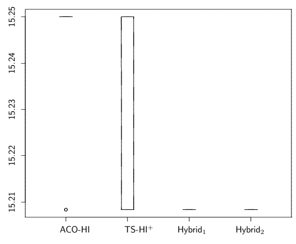
In the future we will study a deeper integration of the two algorithms. In particular, we will boost the construction procedure of the ant algorithm by applying local search on each solution. This way we aim to further improve solution quality and/or decrease the computational resources.

## References

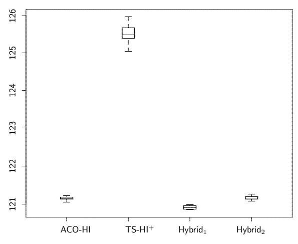
[1] S. Benedettini, A. Roli, and L. Di Gaspero. Two-level ACO for haplotype inference under pure parsimony. In *Ant*



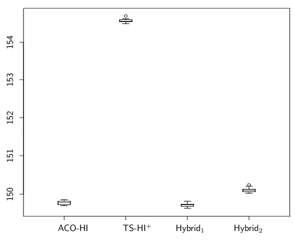
(a) Harrower uniform



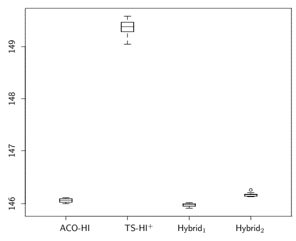
(b) Harrower hapmap



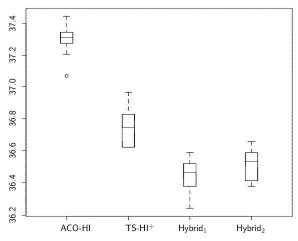
(c) Marchini SU1



(d) Marchini SU2



(e) Marchini SU3



(f) Marchini SU-100kb

Figure 1: Quality of the solutions found by the algorithms.

| Instance set | ACO-HI | TS-HI$^+$ | Hybrid$_1$ | Hybrid$_2$ |
|---|---|---|---|---|
| Harrower uniform | 1002.7 (8.81) | 215.9 (4.51) | 1218.3 (14.51) | 573.9 (7.77) |
| Harrower hapmap | 80.7 (4.03) | 44.5 (2.95) | 124.6 (5.06) | 75.3 (5.76) |
| Marchini SU1 | 1197.9 (8.08) | 337.7 (10.06) | 1572.6 (14.18) | 807 (10.67) |
| Marchini SU2 | 699.4 (8.98) | 12.6 (1.84) | 750.1 (16.34) | 287.6 (5.758) |
| Marchini SU3 | 970.5 (20.93) | 234.3 (15.78) | 1247.4 (24.80) | 607.6 (7.17) |
| Marchini SU-100kb | 611.5 (29.16) | 329.5 (32.06) | 948.4 (41.63) | 536 (18.02) |

Table 3: Running times of the algorithms (in seconds of CPU time).

*Colony Optimization and Swarm Intelligence, 6th International Workshop, ANTS 2008*, Lecture Notes in Computer Science. Springer–Verlag, 2008. To appear.

[2] C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *Transactions on Systems, Man, and Cybernetics – Part B*, 34(2), 2004.

[3] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, September 2003.

[4] D. G. Brown and I. M. Harrower. Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):141–154, 2006.

[5] L. Di Gaspero and A. Roli. Stochastic local search for large-scale instances of the haplotype inference problem by pure parsimony. *Journal of Algorithms in Logic, Informatics and Cognition*, 2008. doi:10.1016/j.jalgor.2008.02.004.

[6] L. Di Gaspero and A. Schaerf. EASYLOCAL++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice and Experience*, 33(8):733–765, 2003.

[7] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA, 2004.

[8] A. Graça, J. Marques-Silva, I. Lynce, and A. L. Oliveira. Efficient haplotype inference with pseudo-boolean optimization. In *AB*, pages 125–139, 2007.

[9] D. Gusfield. Haplotype inference by pure parsimony. In *Combinatorial Pattern Matching (CPM 2003), Proceedings of the 14th Annual Symposium*, volume 2676 of *Lecture Notes in Computer Science*, pages 144–155, Berlin-Heidelberg, Germany, 2003. Springer-Verlag.

[10] G. Lancia, M. C. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, 16(4):348–359, 2004.

[11] The International HapMap Consortium. The international HapMap project. *Nature*, 426:789–796, 2003.

[12] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437, 2005.