Two-level ACO for Haplotype Inference under pure parsimony

Stefano Benedettini[1], Andrea Roli[1], and Luca Di Gaspero[2]

[1] DEIS, Campus of Cesena
Alma Mater Studiorum Università di Bologna
Cesena, Italy
{stefano.benedettini | andrea.roli}@unibo.it
[2] DIEGM, University of Udine
Udine, Italy
l.digaspero@uniud.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 enables 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. At present, the main methods applied to Haplotype Inference are either simple greedy heuristic or exact methods, which are adequate only for moderate size instances.
In this paper, we present an iterative constructive approach to Haplotype Inference based on ACO and we compare it against a state-of-the-art exact method.

1 Introduction

The role of genetic variation and inheritance in human diseases is extremely important, though still largely unknown [1]. 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 [2]. 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 haplotypes information makes it possible to perform association studies for the genetic variants involved in diseases and the individual responses to therapeutic agents. 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 (or explained) 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 could provide criteria for evaluating the solution returned with respect to actual genetic plausibility. In the case of the combinatorial methods, which are the subject of this work, one common criterion is *pure parsimony* [3], i.e., 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, therefore the solutions found to this model are considered as good and informative phasings (see [3] for a discussion on the adequacy of this model).

Current approaches for solving the problem under the pure parsimony hypothesis (HI_{pp}) include simple greedy heuristic [4] and exact methods such as Integer Linear Programming [3, 5–7], Semidefinite Programming [8, 9], SAT models [10, 11] and Pseudo-Boolean Optimization algorithms [12]. At present, complete approaches, i.e., the ones that guarantee to return an optimal solution, such as SAT-based ones, are very effective but they seem not to be particularly adequate 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 criteria used to evaluate the solutions provide an approximation of the actual solution quality, therefore a proof of optimality is not particularly important.

The method we present in this work is a two-level ACO metaheuristic. To the best of our knowledge, besides [13], the only attempt to employ metaheuristic techniques for HI_{pp} is a recently proposed Genetic Algorithm [14] that is, however, not applied on real size instances.

The problem is formally stated in Sect. 2, along with the basic related concepts. In Sect. 3 we describe the two-level ACO we devised and in Sect. 4 we show the results of the experimental analysis in which we first compare the different variants of the two-level ACO, then we assess its performance by comparing it against state-of-the-art exact techniques.

2 The Haplotype Inference problem

In the Haplotype Inference problem we deal with *genotypes*, that is, strings of length m that correspond 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 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.

2.1 Genotype resolution

Given a chromosome, we are interested in finding an unordered[3] 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, and we write $\langle h, k\rangle \triangleright g$ (or $g = h \oplus k$), if the following conditions hold (for $j = 1, \ldots, m$):*

$$g[j] = 0 \Rightarrow \quad h[j] = 0 \wedge k[j] = 0 \tag{1a}$$

$$g[j] = 1 \Rightarrow \quad h[j] = 1 \wedge k[j] = 1 \tag{1b}$$

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

If $\langle h, k\rangle \triangleright g$ we indicate the fact that the haplotype h (respectively, k) contributes in the resolution of the genotype g writing $h \trianglelefteq g$ (resp., $k \trianglelefteq g$). We also say that h is a resolvent *of g. This notation can be extended to set of haplotypes, writing $H = \{h_1, \ldots, h_l\} \trianglelefteq g$, with the meaning that $h_i \trianglelefteq g$ for all $i = 1, \ldots, l$. The operator $\oplus$ is defined accordingly.*

Conditions (1a) and (1b) require that both haplotypes must have the same value in all homozygous sites, while condition (1c) 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 $\langle (0010), (0111)\rangle$.

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 ϕ of n pairs of (not necessarily distinct) haplotypes $\phi = \{\langle h_1, k_1\rangle, \ldots, \langle h_n, k_n\rangle\}$, 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 ϕ, 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 the solution quality.

[3] In the problem there is no distinction between the maternal and paternal haplotypes.

One natural model of the Haplotype Inference problem is the already mentioned *pure parsimony* approach that 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 criterion is APX-hard [6] and therefore NP-hard.

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. In the graph $\mathcal{G} = (G, E)$, the set of vertices coincides with the set of the genotypes. Two 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 (Genotypes 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:*

$$g_1[j] = 0 \Rightarrow g_2[j] \in \{0, 2\} \tag{2a}$$
$$g_1[j] = 1 \Rightarrow g_2[j] \in \{1, 2\} \tag{2b}$$
$$g_2[j] = 2 \Rightarrow g_2[j] \in \{0, 1, 2\} \tag{2c}$$

The same concept can be expressed also between a genotype and a haplotype as in the following definition.

Definition 4 (Compatibility between genotypes and haplotypes). *Let g be a genotype and h a haplotype, g and h are* compatible *if, for all $j = 1, \ldots, m$, the following conditions hold:*

$$g[j] = 0 \Rightarrow h[j] = 0 \tag{3a}$$
$$g[j] = 1 \Rightarrow h[j] = 1 \tag{3b}$$
$$g[j] = 2 \Rightarrow h[j] \in \{0, 1\} \tag{3c}$$

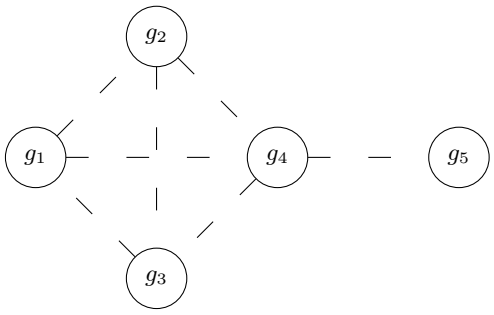
We denote this relation with $h \mapsto g$, and we write $h[j] \mapsto g[j]$ when the conditions hold for the single site j. Moreover with an abuse of notation we indicate with $h \mapsto \{g_1, g_2, \ldots\}$ the set of all genotypes that are compatible with haplotype h.

Notice that the set of genotypes that are compatible with a haplotype can contain only mutually compatible genotypes (i.e., they form a clique in the compatibility graph). As an example of compatibility graph, consider the set of genotypes in Figure 1a, which corresponds to the compatibility graph in Figure 1b.

We also point out that disconnected components of the compatibility graph are necessarily resolved by distinct haplotypes, therefore the optimal set of haplotypes is the union of the optimal sets of each disconnected subgraph. This property is exploited in a specific preprocessing phase of our algorithm.

$g_1 : (2210212)$ $g_2 : (2112110)$
$g_3 : (1212122)$ $g_4 : (1222122)$
$g_5 : (1202201)$

(a) A set of 5 genotypes



(b) The corresponding compatibility graph

Fig. 1: An example of compatibility graph for a set of genotypes.

Another useful property, which is going to be used in our algorithms, is the following:

Proposition 1 (Haplotype complement). *Given a genotype g and a haplotype $h \mapsto g$, 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 Two-level ACO for the Haplotype Inference Problem

The Haplotype Inference problem definition makes constructive procedures very promising. Indeed, a constructive procedure can incrementally build a set H of haplotypes which, taken in pairs, resolve the genotypes. Such a procedure can start from an empty set and add one or two haplotypes at a time, while it scans the set of genotypes G. The objective is to build H as small as possible, i.e., to find a minimal cardinality set of haplotypes that composes the phasing. To this aim, new haplotypes should be added to H only when necessary, i.e., when no pair of haplotypes already in H resolves the current genotype g.

The algorithm we propose is an instance of the Ant Colony Optimization metaheuristic [15] and is composed of two levels: the higher one employs an ACO for finding a good visiting order of genotypes while the lower level, also based on ACO, 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.

In Algorithm 1 we provide the general search scheme that is going to be detailed in the following.

Algorithm 1 ACO-HI

```
1: A: set of ants; G: set of genotypes
2: Preprocessing phase
3: while terminating conditions not met do
4:   for all a ∈ A do
5:     while not all genotypes are resolved do
6:       g ← chooseNode(G)
7:       resolve genotype g
8:       propagate resolvents
9:     end while
10:  end for
11:  pheromoneUpdate()
12: end while
```

3.1 Preprocessing phase

The instances of the Haplotype Inference problem can be reduced by analyzing their structure, while preserving the property that a solution to the reduced instance is a solution to the original one. The first preprocessing step consists in eliminating duplicated genotypes. Furthermore, the analysis of the structure of the compatibility graph enables us to identify independent sub-instances. Indeed, the genotypes belonging to an isolated sub-graph, i.e., a disconnected component, identify a sub-instance that can be solved independently. Therefore, a solution to the original instance can be found by separately solving the sub-instances composing it. A special case of independent instance is represented by isolated nodes, i.e., genotypes that are not compatible with any other genotype. The contribution of such a genotype to the solution of the Haplotype Inference instance is composed by a pair of haplotypes that, by definition of compatibility, cannot be used to resolve any other genotype.

3.2 Lower level: genotype resolution

As depicted in Algorithm 1, an ant a builds a solution by considering in turn each genotype $g \in G$ (the order is defined in the higher-level, see Sect. 3.3) and finding resolvent haplotypes for it. The basic heuristic for this phase consists in trying to resolve g with haplotypes already in H and add new haplotypes only if necessary. When a new resolvent has to be added to H, the values of its heterozigous sites are chosen on the basis of pheromone values.[4] For each (heterozygous) site j on genotype i, we have two pheromone components, $\tau^0_{i,j}$ and $\tau^1_{i,j}$, corresponding to values 0 and 1, respectively. The value assigned to the haplotype site is chosen with probability $p_{i,j}(v) = \tau^v_{i,j}/(\tau^0_{i,j} + \tau^1_{i,j})$, with $v \in \{0, 1\}$.

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

Excluding the case in which H already contains a pair of haplotypes resolving g, the are three different cases to be considered for the resolution of a genotype g in this step of the algorithm: (i) no resolving candidates in H, (ii) one candidate, (iii) more than one candidate. In the following, we detail the procedure defined for these cases:

Case (i): A haplotype h is built by a pheromone guided construction procedure, as previously described. Then, $k = g \ominus h$, the complement of h, is built and both are added to H. Then, these two new haplotypes are *propagated* along the compatibility graph in order to update the list of resolving candidates for the genotypes.

Case (ii): When one resolving candidate is already available, its complement w.r.t. g is built and this step completed as in the previous case.

Case (iii): When there are two or more candidates that can resolve g, but no pair of them can resolve it, we have to choose one among these haplotypes. We implement this operation by iteratively considering each site and applying the following procedure: if, among the candidates, the homologous sites have different values (i.e., at least in a pair there are both values 0 and 1) one of the two is chosen probabilistically (using pheromone values) and all the candidates with a different value are discarded. The procedure ends when only one candidate is left and the final steps of the previous cases are performed again.

The algorithm, named *ACO-HI*, can be further improved by slightly modifying the procedure implemented for cases *(i)*. In fact, since the new haplotype added to H must resolve the current genotype g, a heuristic bias toward the construction of a haplotype that also resolves another genotype compatible with g can be beneficial. Thus, the genotype g' that has to be visited after g is determined by the higher level and a haplotype is probabilistically constructed (as in the original procedure) that not only resolves g, but also g'. Therefore, the number of sites to be assigned on the basis of the pheromone values is restricted to the set of sites which are ambiguous in both the genotypes g and g'. In this way, haplotype construction is still guided by pheromone only, but a simple kind of heuristic criterion is introduced to avoid building a new haplotype compatible only with genotype g. A similar procedure is also applied, with slight modifications, also in case *(iii)*. We will refer to the improved version of *ACO-HI* as *ACO-HI*$^+$.

3.3 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. It would be possible to learn larger building blocks, such as triplets, but we decided to limit the case to pairs because of efficiency reasons. Formally, every edge (i, j) of the compatibility graph is associated to a pheromone value τ_{ij} and the probability to move from node i to node j is given by:

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

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 sequence is chosen randomly among the remaining unresolved genotypes.

3.4 Pheromone update

Our algorithm is implemented according to the Hyper-cube framework [16]. 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 higer level are edges of the compatibility graph, while in the lower level they are nodes representing values to assign to haplotype sites.

4 Experimental analysis

With the aim of understanding the contribution of each algorithmic component, we compared *ACO-HI* and its improved version, *ACO-HI*$^+$, against two versions of the algorithm with the learning mechanism disabled: *ACO-HI-random* that chooses the sequence of genotypes to be visited randomly and *ACO-HI*$^+$*(no learning)* that is a version of *ACO-HI*$^+$ equipped only with heuristic haplotype construction (ties are broken randomly) and random sequence of visited genotypes.

The sets of instances chosen for the experimental analysis are the common benchmark used in previous works and they are composed of two parts. The first one, composed of the sets Harrower uniform and Harrower hapmap, is the benchmark used in [7]. 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 features of the instance sets are summarized in Table 1.

We present a comparison of the different versions of ACO in Figure 4, in which statistics on solution quality are plotted. The boxplots represent statistics

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

Benchmark set	**Number of instances**	**Number of genotypes**	**Number 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: Cumulative statistics on the running time of algorithm *ACO-HI*$^+$. The total running time (in seconds) for solving all the instances of each set is considered.

Benchmark set	**Min.**	**1st Q.le**	**Median**	**Mean**	**3rd Q.le**	**Max.**
Harrower uniform	54.00	55.50	65.00	64.30	72.75	75.00
Harrower hapmap	13.00	21.00	27.00	30.40	34.75	59.00
Marchini SU1	1634	1743	1808	1797	1861	1948
Marchini SU2	466.0	516.2	538.0	533.9	546.8	584.0
Marchini SU3	1401	1452	1488	1487	1541	1549
Marchini SU-100kb	148.0	155.0	169.5	165.8	174.8	182.0

over 10 independent runs of the algorithms on all the instances of each set. The solution value considered for the statistics is the sum of solutions returned on all the instances of each benchmark.[5] Algorithms are stopped after 300 iterations of the main construction loop; this value of maximum iterations is set in such a way that the algorithm reaches stagnation. Pheromone evaporation has been set to 0.1, according to a *brute force* analysis over a representative sample of the instances. The algorithms have been implemented in C++ and run on a 1GHz Intel Core Duo with 2GB of RAM and Linux Ubuntu 7.10 (kernel 2.6.22).

We can observe that, except for the set Marchini SU-100kb, *ACO-HI*$^+$ is superior to the other variants and the synergy between its constructive heuristic and learning mechanism based on pheromone (both for higher and lower levels) is quite effective. We conjecture that the good performance of *ACO-random* on the set Marchini SU-100kb is caused by the structure of the instances; indeed, most of these instances have a very sparse compatibility graph and this characteristic makes the high level learning component much less effective, maybe even misleading for the search.

For lack of space, we omit detailed data on running times and we just report the cumulative statistics on the running time of algorithm *ACO-HI*$^+$ in Table 2, in which the running time to the best solution, in seconds, is considered.

[5] Detailed results are omitted because of lack of space and are available from the authors upon request.

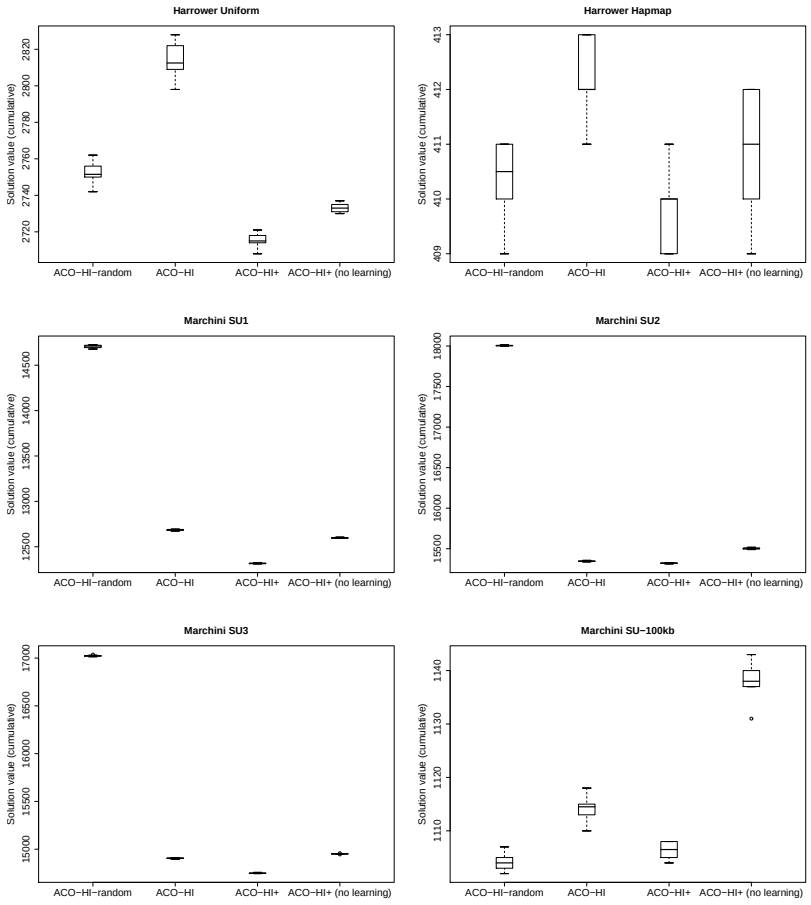


Fig. 2: Comparison of the main ACO variants w.r.t. solution quality

5 Comparison against the state of the art

Algorithm *ACO-HI*$^+$ has a very good performance, both in terms of quality and time. In this section we compare its performance against the state-of-the-art exact solver, in order to assess its effectiveness. To the best of our knowledge, the best complete solver for the Haplotype Inference problem is *rpoly* [12]. We run the solver 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 hapmap, Marchini SU1 and Marchini SU2 were completely solved. From Marchini SU3 only 89 over 100 instances were solved and from Marchini SU-100kb were solved 23 over 29 instances. Overall, most of the instances could be solved with a runtime higher than 12 hours. In Table 3

Table 3: Solution quality of *ACO-HI*$^+$ and local search [13] w.r.t. optimal solution values.

		Sum of solution values (Perc. error)	
Benchmark set	***rpoly***	***ACO-HI*$^+$**	***HI-Tabu search [13]***
Harrower uniform	2689	2694 (0.186)	3252 (21.0)
Harrower hapmap	321	321 (0.0)	343 (6.854)
Marchini SU1	2453	2483 (1.223)	3456 (40.89)
Marchini SU2	14794	15102 (2.081)	17735 (19.88)
Marchini SU3	2113	2121 (0.379)	2333 (10.41)
Marchini SU-100kb	661	667 (0.009)	755 (14.22)

we provide the comparison between the solutions returned by *ACO-HI*$^+$ and the optimal one provided by *rpoly* (when available) for each benchmark. The solution values have been summed up over the instances composing the set (for *ACO-HI*$^+$ we considered the best among the 10 runs) and also the error w.r.t. the sum of optimal solutions is reported. We can observe that *ACO-HI*$^+$ achieves a very good performance in terms of solution quality, as the error w.r.t. the optimum is rather small. Furthermore, *ACO-HI*$^+$ also compares quite favourably with the state-of-the-art local search for HI_{pp} [13] in terms of overall solution quality.[6]

6 Conclusions and future work

We have presented an adaptive constructive approach for the Haplotype Inference problem under the pure parsimony criterion, which relies on a two-level ACO procedure for determining first the genotype to be resolved and then choosing the haplotypes to resolve it.

The experimental evaluation of the algorithm has shown that the algorithm is very effective for solving medium- to large-scale instances of the problem on common benchmarks and that its running time scales well with the dimension of the instances.

In future developments we plan to enhance the behavior of the ACO metaheuristic by testing hybrid approaches. A possible research direction we intend to pursue is to couple the ACO with a Local Search procedure [13] with the aim improving the solution found by each ant. Other possibilities include the replacement of the ACO for genotype ordering with other metaheuristics, such as evolutionary algorithms or again local search.

Acknowledgments

We thank Inês Lynce and Ana Sofia Graça for kindly providing us their instances and solvers, and we also thank Ian M. Harrower for sending us his datasets.

[6] We omit the complete direct comparison of the two techniques because of limited space and we refer the reader to [13].

References

1. The International HapMap Consortium: A haplotype map of the human genome. Nature **437** (2005)
2. The International HapMap Consortium: The international HapMap project. Nature **426** (2003) 789–796
3. Gusfield, D.: 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., Berlin-Heidelberg, Germany, Springer-Verlag (2003) 144–155
4. Clark, A.G.: Inference of haplotypes from PCR-amplified samples of diploid populations. Molecular Biology and Evolution **7** (1990) 111–122
5. Halldórsson, B.V., Bafna, V., Edwards, N., Lippert, R., Yooseph, S., Istrail, S.: A survey of computational methods for determining haplotypes. In: Computational Methods for SNPs and Haplotype Inference. Volume 2983 of Lecture Notes in Computer Science., Springer (2002) 26–47
6. Lancia, G., Pinotti, M.C., Rizzi, R.: Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. INFORMS Journal on Computing **16**(4) (2004) 348–359
7. 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) (2006) 141–154
8. Kalpakis, K., Namjoshi, P.: Haplotype phasing using semidefinite programming. In: BIBE, IEEE Computer Society (2005) 145–152
9. Huang, Y.T., Chao, K.M., Chen, T.: An approximation algorithm for haplotype inference by maximum parsimony. In: Proceedings of the 2005 ACM Symposium on Applied Computing (SAC 2005), ACM (2005) 146–150
10. Lynce, I., Marques-Silva, J.: SAT in bioinformatics: Making the case with haplotype inference. In: SAT. Volume 4121 of Lecture Notes in Computer Science., Berlin-Heidelberg, Germany, Springer-Verlag (2006) 136–141
11. Lynce, I., Marques-Silva, J.: 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, Menlo Park, CA, USA, AAAI Press (2006)
12. Graça, A., Marques-Silva, J., Lynce, I., Oliveira, A.L.: Efficient haplotype inference with pseudo-boolean optimization. In: AB. (2007) 125–139
13. Di Gaspero, L., Roli, A.: 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.
14. Wang, R.S., Zhang, X.S., Sheng, L.: Haplotype inference by pure parsimony via genetic algorithm. In: Operations Research and Its Applications: the Fifth International Symposium (ISORA'05), Tibet, China, August 8–13. Volume 5 of Lecture Notes in Operations Research. Beijing World Publishing Corporation, Beijing, People Republic of China (2005) 308–318
15. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA, USA (2004)
16. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. Transactions on Systems, Man, and Cybernetics – Part B **34**(2) (2004)