A Branch
Price algorithm for the minimum cost clique cover problem in max-point tolerance graphs

Luciano Porretta, Daniele Catanzaro, Bjarni Halldórsson, Bernard Fortz

Noname manuscript No.
(will be inserted by the editor)

A *Branch&Price* Algorithm for the Minimum Cost Clique Cover Problem in Max-Point Tolerance Graphs

L. Porretta, · D. Catanzaro, · B. V. Halldórsson, · B. Fortz

Abstract A *point-interval* (I_v, p_v) is a pair constituted by an interval I_v of $\mathbb{R}$ and a point $p_v \in I_v$. A graph $G = (V, E)$ is a *Max-Point-Tolerance (MPT)* graph if each vertex $v \in V$ can be mapped to a point-interval in such a way that (u, v) is an edge of G iff $I_u \cap I_v \supseteq \{p_u, p_v\}$. MPT graphs constitute a superclass of interval graphs and naturally arise in genetic analysis as a way to represent specific relationships among DNA fragments extracted from a population of individuals. One of the most important applications of MPT graphs concerns the search for an association between major human diseases and chromosome regions from patients that exhibit loss of heterozygosity events. This task can be formulated as a minimum cost clique cover problem in a MPT graph and gives rise to a $\mathcal{NP}$-hard combinatorial optimization problem known in the literature as the *Parsimonious Loss of Heterozygosity Problem* (PLOHP). In this article, we investigate ways to speed up the best known exact solution algorithm for the PLOHP as well as techniques to enlarge the size of the instances that can be optimally solved. In particular, we present a *Branch&Price* algorithm for the PLOHP and we develop a number of preprocessing techniques and decomposition strategies to dramatically reduce the size of its instances. Computational experiments show that the proposed approach is 10-30x faster than previous approaches described in the literature, and suggest new directions for the development of future exact solution approaches that may prove of fundamental assistance in practice.

Luciano Porretta and Bernard Fortz belong to the Graphs and Mathematical Optimization Unit, Computer Science Department, Université Libre de Bruxelles (ULB), Boulevard du Triomphe, CP 210/01, B-1050, Brussels, Belgium.
Daniele Catanzaro belongs to the Center for Operations Research and Econometrics of the Université Catholique de Louvain (UCL), Voie du Roman Pays 34, L1.03.01, B-1348 Louvain-la-Neuve, Belgium.
Bjarni V. Halldórsson belongs to the Institute of Biomedical and Neural Engineering, School of Science and Engineering, Reykjavík University, Menntavegur 1, 101 Reykjavik, Iceland.

1 Introduction

Interval graphs (namely, intersection graphs of intervals on a line [3, 13, 15]) constitute an important class of graphs in computer science and discrete mathematics both for their significance in practical applications [10] and because classical $\mathcal{NP}$-hard combinatorial optimization problems, such as the maximum clique problem [15, 16], the weighted independent set problem [14, 16] and the coloring problem [16, 18], can be solved in polynomial time in this particular class of graphs [13, 15].

In the last three decades, numerous research efforts have been carried out to extend the properties of interval graphs to more general classes of graphs [2, 9, 10, 17, 18, 19]. In this context, an important generalization is constituted by *tolerance graphs*, first introduced in [19, 20]. Specifically, a graph $G = (V, E)$ is a *tolerance graph* (namely, a *max-tolerance* graph [20]) if every vertex $v \in V$ can be associated with both an interval $I_v \subseteq \mathbb{R}$ and a *tolerance* value $t_v \in \mathbb{R}$ in such a way that (u, v) is an edge of G iff $|I_u \cap I_v| \geq \max\{t_u, t_v\}$. In other words, G is a tolerance graph if each pair of intervals can "tolerate" a non-empty intersection (without forming an edge) as long as the length of this intersection is strictly smaller than the maximum of the corresponding tolerance values. As for interval graphs, solving the maximum clique problem on a tolerance graph can be performed in polynomial time both in the weighted and the unweighted case [22]. In contrast, the recognition problem has been shown to be $\mathcal{NP}$-complete [22] and little is known concerning the complexities of other classical combinatorial optimization problems in this class.

Another generalization relevant to this work is represented by a variant of tolerance graphs, called *Max-Point-Tolerance (MPT)* graphs, recently introduced in [5] and characterized in [6]. Specifically, a graph $G = (V, E)$ is a MPT graph if each vertex $v \in V$ can be mapped to a *point-interval*, i.e., a pair (I_v, p_v) constituted by an interval $I_v = (l_v, r_v)$ of $\mathbb{R}$ and a *point* $p_v \in I_v$ (see Figure 1), in such a way that (u, v) is an edge of G if and only if $I_u \cap I_v = (l_u, r_u) \cap (l_v, r_v) \supseteq \{p_u, p_v\}$. In other words, G is a MPT graph if each pair of intervals can "tolerate" a non-empty intersection (without forming an edge) as long as their distinct points are not contained in this intersection. As an example, Figure 2 shows the MPT graph associated to the point-intervals shown in Figure 1.

As for interval graphs, classical combinatorial optimization problems over MPT graphs can be solved in polynomial time, including the maximum clique problem [5] and the maximum weight independent set problem [6]. In contrast, the coloring problem and the minimum cost clique cover problem [16] have been shown to be $\mathcal{NP}$-hard [6]. The latter problem is particularly relevant to this article due to its remarkable importance in practical applications. Specifically, MPT graphs naturally arise in genome-wide association studies as a way to represent relationships of loss of heterozygosity among DNA fragments extracted from a patient cohort affected by a common genetic disease [5, 21, 21, 28]. In such applications, an interval I represents the maximal boundary on a chromosome region from a patient that may carry a loss of heterozygosity; the point $p \in I$ instead represents a site in the considered region that shows evidence for a loss of heterozygosity [5, 21]. Catanzaro et al [5] showed that searching across the genomes extracted from the patient cohort for massive shared loss of heterozygosity events that can be associated with a given genetic disease is equivalent to solving a particular version of the minimum cost clique cover problem stated as follows:

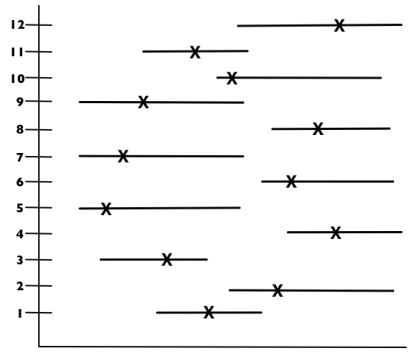

Fig. 1: An example of point-intervals

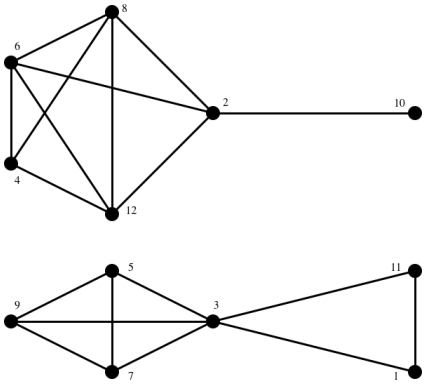

Fig. 2: The MPT graph induced by the point-intervals shown in Figure 1.

The Parsimonious Loss of heterozygosity Problem (PLOHP) [5]. *Given the MPT graph $G = (V, E)$ induced by a set $\mathcal{I}$ of point-intervals and a cost function f that associates a positive value to every maximal clique C and every* singleton *(clique of cardinality 1) $v \in V$ of G, find a cover $\mathcal{C}^*$ of G into singletons and maximal cliques such that*

$$\mathcal{C}^* = \arg \min_{\mathcal{C} \subseteq \mathcal{C}(G)} \sum_{C \in \mathcal{C}} f(C)$$

where $\mathcal{C}(G)$ is the set of all maximal cliques and singletons in G. To simplify the notation, we define $f(v) = f(\{v\})$, for all $v \in V$.

The $\mathcal{NP}$-hardness of the PLOHP as well as the need to solve practical instances of the problem have justified in recent years the development of enumerative search methods and approximation algorithms such as those described in [5, 21]. In particular, Halldorsson et al [21] proposed both an exact algorithm and a greedy constructive heuristic to solve the problem. The exact algorithm is based on a brute force enumeration of all of the possible clique covers of G and proved to be unpractical even for small instances of the PLOHP [21]. The constructive heuristic instead builds a solution to the PLOHP by finding first the maximal cliques of G and by greedily combining them to obtain a cover of G. As a drawback, the approximate approach may largely overestimate the loss of heterozygosity events across the considered genomes and lead to wrong associations [21].

Catanzaro et al [5] investigated possible ways to overcome the performances of the exact algorithm presented in [21]. In particular, the authors first proved the $\mathcal{NP}$-hardness of the PLOHP and investigated some fundamental properties of MPT graphs. Subsequently, the authors proposed an Integer Linear Programming (ILP) formulation and a number of valid inequalities able to optimally solve instances containing up to 250,000 point-intervals within 5 hours of computing time. In this article, we investigate ways to improve upon the performances of the ILP formulation described in Catanzaro et al [5] s well as techniques to solve instances having size much larger than those solvable via that formulation. To this end, in Section 3 we revisit the ILP formulation described in Catanzaro et al [5]

and we develop a new one based on inverse projection [23] and solvable by column generation approaches and *Branch&Price* techniques. In particular, we show that the pricing oracle reduces to finding a maximum node-weighted cliques in a MPT graph and we develop a polynomial time solution algorithm for this purpose. In Section 2, we introduce a number of preprocessing and decomposition strategies to dramatically reduce the size of a given instance of the problem and to divide a reduced instance into a family of independent subproblems. In Section 4, we measure the performance of the proposed algorithm on a set of artificial and biological instances of the PLOHP and we compare them versus the current state of the art presented in [5, 21]. The results show that proposed algorithm is on average 10-30x faster than the ILP formulation described in [5] and enable the solution of very large practical instances of the PLOHP containing over half million point-intervals within 1 hour computing time.

2 Preprocessing

In this section we develop a number of preprocessing techniques both to reduce the size of an instance of the PLOHP and to decompose an instance into a family of independent subproblems having a smaller size than the original one. We will discuss the computational impact of these techniques in Section 4.

2.1 Removing Vertices

A natural approach to reduce the size of an instance of the PLOHP consists of removing point-intervals inducing isolated vertices in the MPT graph. In fact, any feasible solution to the PLOHP is characterized by identifying as singleton any isolated vertex of the MPT graph. The isolated vertices can be efficiently detected by inspecting the set $\mathcal{I}$ for intervals having no-compatibility between each other. For example, the point-interval $\mathcal{I}_v$ induces an isolated vertex in V if there does not exists in $\mathcal{I}$ a distinct point-interval $\mathcal{I}_u$ such that $(l_u \leq p_v \leq r_u) \wedge (l_v \leq p_u \leq r_v)$. It is easy to see that the removal of the isolated vertices from an instance of the PLOHP can be performed in $O(|\mathcal{I}|(|\mathcal{I}|-1)/2)$.

It is worth noting that the isolated vertices are not the only vertices that can be removed from an instance of the PLOHP. For example, given a point-interval $I_v = [(l_v, r_v), p_v] \in \mathcal{I}$, consider the situation in which there exists another interval $I_u = [(l_u, r_u), p_u] \in \mathcal{I}$ such that $l_u = l_v,\ p_u = p_v,\ r_u = r_v$. In such a case, we say that I_u is a *cloned interval* of I_v. Let Γ_{I_v} be the set of cloned intervals of I_v. Then, it is easy to see that the set of vertices induced by Γ_{I_v} can be removed from V as I_v is representative of the induced equivalence class Γ_{I_v}. The identification of the cloned intervals can be performed in time $O(|\mathcal{I}|(|\mathcal{I}|-1)/2)$ and can be carried out together with the removal of the isolated vertices.

2.2 Decomposing a MPT graph into connected components

The particular nature of point-intervals may give rise to MPT graphs with several connected components. For example, the point-intervals shown in Figure 1 leads to

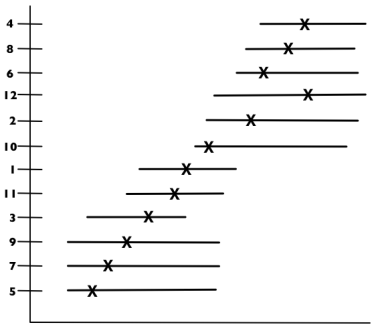

Fig. 3: An example of block decomposition of the point-intervals shown in Figure 1.

to a MPT graph with by two connected components (see Figure 2). The potential existence of connected components in a MPT graph can be exploited both to reduce the time necessary to identify a maximal clique in G as well as to decompose the problem into independent subproblems of smaller size. A possible way to quickly identify connected components in a MPT graph consists of ordering the point-intervals according to a specific precedence rule. In particular, given two distinct point-intervals $I_u, I_v \in \mathcal{I}$, we say that I_v precedes I_u, in symbols $I_v \prec I_u$, if one of the following cases is verified:

1. $l_v < l_u$;
2. if $l_v = l_u$ then $p_v < p_u$;
3. if $l_v = l_u$ and $p_v = p_u$ then $r_v \leq r_u$.

For example, Figure 3 shows the result of the application of the precedence rule on the point-intervals shown in Figure 1. It is easy to realize that, if the point-intervals have been sorted according to the above precedence rule, a connected component arises in G whenever there exist two $I_u = \{(l_u, r_u), p_u\}$ and $I_v = \{(l_v, r_v), p_v\}$ such that $p_v \leq l_u$.

The set of connected components can be identified either by an exact approach such as that described in [8] or by heuristics. As the exact approach is time consuming for very large graphs and not particularly beneficial in terms of identification of connected components, we opted for an heuristic approach able to exploit the particular structure of a MPT graph. In particular, the heuristic provided in Algorithm 1 runs in $O(|\mathcal{I}|)$ and exploits the sufficient condition $p_i < l_{i+1}$ to identify a set of nodes not connected with a given node. The algorithm takes as input the

Algorithm 1 — Connected Components Finder

Input: sorted $\mathcal{I}$
Output: $\mathcal{K}$: set of connected components of G
1: $i = 0$, $K = \emptyset$, $\mathcal{K} = \emptyset$
2: **while** $i < |\mathcal{I}|$ **do**
3: **if** $p_i > l_{i+1}$ **then**
4: $K = K \cup \{I_i\}$
5: **else**
6: $\mathcal{K} = \mathcal{K} \cup \{K\}$, $K = \emptyset$
7: **return** $\mathcal{K}$

set $\mathcal{I}$ sorted according to the above precedence rule. Subsequently, the algorithm visits each point-interval in $\mathcal{I}$ and constructs a connected component B of G by collecting all of the point-intervals I_i such that I_i does not satisfy the condition $p_i \leq l_{i+1}$. The decomposition of G into connected components enables running in parallel several Branch&Price algorithms; this approach may vastly shorten the solution time of the overall solution algorithm for the PLOHP.

3 A Branch&Price algorithm for the PLOHP

In this section we describe an exact solution algorithm for the PLOHP based on *Branch&Price*. Throughout the section we assume, without loss of generality, that the cost function f in the PLOHP is a *size-defined submodular set function*, i.e., it is such that $f(S) = \psi(|S|)$, for any subset $S \subseteq V$ and for some $\psi : [0 \dots |V|] \to \mathbb{R}_0^+$. This characterization of f is commonly assumed in loss of heterozygosity studies [5] and allows to solve the PLOHP in polynomial time when the MPT graph is reducible to an interval-graph [11].

3.1 An integer linear programming formulation for the PLOHP

Consider a set $\mathcal{I}$ of point-intervals and let $G = (V, E)$ be the MPT graph induced by $\mathcal{I}$. Then, the following proposition holds [5]:

Proposition. *A MPT graph contains at most* $|\mathcal{I}|(|\mathcal{I}| - 1)/2$ *maximal cliques.*

As the number of maximal cliques in a MPT graph is polynomially bounded and a polynomial time algorithm to determine all of them exists (see [5]), a polynomial sized formulation for the PLOHP can be obtained as follows. Consider a vertex $v \in V$ and let $\mathcal{C}_v = \{C \in C(G) : v \in C\}$. Let x_v be a decision variable equal to 1 if vertex $v \in V$ forms a singleton in a solution to the problem and 0 otherwise. Similarly, let y_C be a decision variable equal to 1 if the maximal clique $C \in \mathcal{C}(G)$, with $|C| \geq 2$, is selected in a solution to the problem and 0 otherwise. Then, a valid ILP formulation for the PLHOP is:

Catanzaro-Labbé-Halldorsson' ILP Formulation (CLHF).

$$z_{BIP} = min \sum_{\substack{C \in \mathcal{C}(G) \\ |C| \geq 2}} f(C) y_C + \sum_{v \in V} f(v) x_v \tag{1a}$$

$$s.t. \sum_{\substack{C \in \mathcal{C}_v(G) \\ |C| \geq 2}} y_C + x_v \geq 1 \qquad \forall\, v \in V \tag{1b}$$

$$x_v \in \{0, 1\} \qquad \forall\, v \in V \tag{1c}$$

$$y_C \in \{0, 1\} \qquad \forall\, C \in \mathcal{C}(G), |C| \geq 2. \tag{1d}$$

The objective function (1a) minimizes the cost of finding a clique cover of G, and constraints (1b) impose that, for each vertex $v \in V$, the solution contains either a clique or a singleton in $C(G)$ that covers v. Note that, as the overall number of maximal cliques in a MPT graph is polynomially bounded, the CLHF is polynomial-sized. However, when the instances of the PLOHP are very large (e.g.,

they include more than 250,000 point-intervals), the use of the CLHF may become unpractical due to the large number of y variables. A possible approach to overcome such a limitation consists of reformulating the CLHF by inverse projection [23]. To this end, denote $\mathcal{C}^k(G)$ as the subset of $\mathcal{C}(G)$ that includes $k \leq |\mathcal{C}(G)|$ maximal cliques in G and set $\mathcal{C}_v^k(G) = \{C \in \mathcal{C}^k(G) : v \in C\}$. Now, consider the following *Restricted Master Problem* (RMP) associated with the linear programming relaxation of the CLHF:

the CLHF - The Restricted Master Problem (RMP).

$$z_{RMP} = min \sum_{\substack{C \in \mathcal{C}^k(G) \\ |C| \geq 2}} f(C) y_C + \sum_{v \in V} f(v) x_v \tag{2a}$$

$$s.t. \sum_{\substack{C \in \mathcal{C}_v^k(G) \\ |C| \geq 2}} y_C + x_v \geq 1 \qquad \forall\ v \in V \tag{2b}$$

$$x_v \geq 0 \qquad \forall\ v \in V \tag{2c}$$

$$y_C \geq 0 \qquad \forall\ C \in \mathcal{C}^k(G), |C| \geq 2. \tag{2d}$$

Note that in the RMP constraints $x_v \leq 1$, for all $v \in V$, and $y_C \leq 1$, for all $C \in \mathcal{C}^k(G)$, $|C| \geq 2$, have been omitted as they are redundant. The RMP can be solved by standard linear programming techniques. To this end, denote μ_v as the dual variables associated with constraints (2b). The dual of the RMP is characterized by the following dual constraints:

$$\mu_v \leq f(v) \qquad \forall\ v \in V \tag{3a}$$

$$\sum_{v \in V : v \in C} \mu_v \leq f(C) \qquad \forall C \in \mathcal{C}^k(G) : |C| \geq 2. \tag{3b}$$

A variable with negative reduced cost to be added to the RMP corresponds to a dual constraint violated by the current optimal solution. Since all of the x_v variables are present in the RMP, constraints (3a) will never be violated. In contrast, constraints (3b) are violated if

$$\exists\ \hat{C} \in \mathcal{C}(G) : |\hat{C}| \geq 2 \text{ and } \sum_{v \in V : v \in \hat{C}} \mu_v > f(\hat{C}). \tag{4}$$

Checking whether (4) holds in the current optimal solution to the RMP involves solving a maximum node-weighted clique problem in G with weights $\{\mu\}$. This task can be performed e.g., by means of Algorithm 3 that is described in Section 3.2. Algorithm 2 implements the column generation technique by using Algorithm 3 as pricing oracle. Specifically, Algorithm 2 takes G as an input and iterates the following steps: line 2 solves the RMP; line 3 gets the dual values $\{\mu\}$ and associates a weight μ_v to each vertex $v \in V$; line 5 calls the oracle and searches in $\mathcal{C}(G)$ for a maximum node-weighted clique $\hat{C}$ satisfying (4); if such a clique exists, line 7 adds variable $y_{\hat{C}}$ (and its corresponding column) to the RMP and iterates lines 1-8; if, there are no violated constraints (2b), the solution to the RMP is provably optimal and Algorithm 2 returns the optimal value to the RMP. We observed that adding many cliques C having an overall weight larger than $f(C)$ together with the most violated node-weighted clique at each iteration of the pricing oracle may

Algorithm 2 — Column Generation

Input: G
Output: Optimal solution to the RMP
1: **repeat**
2: Solve the RMP
3: Get the dual vector μ associated to (2b)
4: Set weight μ_v on each vertex $v \in V$
5: $\hat{C}$ =PricingOracle(G,μ)
6: **if** $\hat{C} \neq \emptyset$ **then**
7: Add $y_{\hat{C}}$ and its column to the RMP
8: **until** $\hat{C} = \emptyset$
9: return z^*_{RMP}

decrease the running time of Algorithm 2. Section 4 explores, via computational experiments, the problem of deciding how many cliques should be added in a generic instance of the PLOPH.

3.2 Finding a maximum node-weighted clique in a MPT graph

A critical step in Algorithm 2 consists of finding a maximum node-weighted clique in G. Although this problem is $\mathcal{NP}$-hard for general graphs, it can be solved in polynomial time when the graph is MPT and the objective function is size-defined submodular [4]. In fact, it is worth noting that, as G is a MPT, any subset of vertices that forms a maximal clique C in G is characterized by having two vertices whose corresponding points, say p_l and p_r, are the furthest to the left and to the right, respectively, in the set of point-intervals induced by C. Each vertex v in the clique is connected to these two vertices and its corresponding point-interval is such that $p_v \in [p_l, p_r] \subseteq [l_v, r_v]$. Hence, a clique can be defined by the leftmost and rightmost vertices, respectively. Then, a possible approach to enumerate the polynomial bounded number of cliques in G consists of picking any pair of adjacent vertices in G (i.e., distinct point-intervals in $\mathcal{I}$, say (I_u, p_u) and (I_v, p_v), such that $I_u \cap I_v = (l_u, r_u) \cap (l_v, r_v) \supseteq \{p_u, p_v\}$) and searching for the largest set of points that fall inside $[p_i = \min\{p_u, p_v\}, p_j = \max\{p_u, p_v\}]$, i.e., the largest set of point-intervals I_k that satisfy condition $(p_i \leq p_k \leq p_j) \wedge (l_k < p_i) \wedge (r_k > p_j)$. Algorithm 3 outlines the pseudo code necessary to perform the overall task. We note that the complexity of Algorithm 3 is $O(|\mathcal{I}|^3)$.

3.3 Initializing the columns of the RMP

In order to initialize the RMP we first search for a subset $\mathcal{C}^k(G) \subset \mathcal{C}(G)$ covering all of the vertices in G and such that any clique in $\mathcal{C}^k(G)$ has cardinality greater than or equal to 2. A possible approach to carry out this task is outlined in Algorithm 4. In particular, the algorithm takes it as input the set $\mathcal{I}$ in the non-decreasing order described in Section 2.2 to rapidly identify the maximal cliques. Subsequently it iterates the following procedure: for each point-interval $I_k \in \mathcal{I}$ (i) find a maximal clique C that cover (the vertex induced by) I_k and (ii) remove from $\mathcal{I}$ the point-intervals (whose induced vertices are) in C. The procedure stops when $\mathcal{I} = \emptyset$. The complexity of Algorithm 4 is $O(|\mathcal{I}|^3)$.

Algorithm 3 — Pricing Oracle

Input: G, μ
Output: $\mathcal{C}^\star$: a maximum node-weighted clique in G
set $C = \emptyset$, $max = 0$, get $\mathcal{I}$ from G
for all $I_i, I_j \in \mathcal{I} : i < j$ **and** adjacent(I_i, I_j) **do**
  set $cost = 0$
  for all $I_k \in \mathcal{I}$ **do**
    if $(p_i \leq p_k \leq p_j) \wedge (l_k < p_i) \wedge (r_k > p_j)$ **then**
      $C = C \cup \{p_k\}$
  for all $v \in V : v \in C$ **do**
    $cost = (cost + \mu_v)$
  if $cost > max$ **then**
    $max = cost$
    $\mathcal{C}^\star = C$
return $\mathcal{C}^\star$

3.4 Branching rules

In order to find the optimal solution to the CLHF, we embody the previously described column generation approach in a *Branch-&-Bound* algorithm, in which the branching is performed on both x and y variables. Specifically, concerning the x variables, the algorithm branches according to the *most fractional variable* rule (see [26]). The same rule also applies to the y variables with the exception that a y variable that has been set to 0 in a previous node of the search tree cannot be set to 1 in one of its descendants. To ensure the application of this rule we perform the following steps: (1) we store all the maximal clique generated in each node; (2) during the column generation process, we filter, from among all of the maximal cliques, the ones associated to the y variables that have been set to 0 in the parent node; (3) we propagate the filtered set to all the descendant nodes.

4 Computational experiments

In this section we analyze the performance of the *Branch&Price* algorithm to solve instances of the parsimonious loss of heterozygosity problem. Our experiments

Algorithm 4 — $\mathcal{C}^k(G)$-Generator

Input: Set of intervals $\mathcal{I}$
Output: $\mathcal{C}^k(G)$
$\mathcal{C} = \emptyset$, $Q = \mathcal{I}$
for all $I_i, I_j \in \mathcal{I} : i < j$ **and** adjacent(I_i, I_j) **and** $Q \neq \emptyset$ **do**
  $C = \emptyset$
  for all $I_k \in \mathcal{I}$ **do**
    if COMPATIBLE(I_i, I_j, I_k) **then**
      $Q = Q \setminus I_k$
      $C = C \cup I_k$
  $\mathcal{C} = \mathcal{C} \cup C$
return $\mathcal{C}$

function COMPATIBLE(I_u, I_v, I_z)
  return $(p_u \leq p_z \leq p_v) \wedge (l_z < p_u) \wedge (r_z > p_v)$

	Nodes	Edges	Cliques
Maximun	66 670	2 060 044 865	52 522
Minimum	35 725	742 125 353	16 915
Average	57 581.35	1 335 575 876.85	36 824.55
Standard Deviation	8 056.94	313 44 967.49	8 011.84

Table 1: Summary of number of nodes, edges and cliques contained in the learning set of instances of the PLOHP used to calibrate the pricing oracle.

were motivated by a number of goals, namely: to compare the performance of the *Branch&Price* algorithm with the ones obtained by the CLHF [5], which currently is the best exact algorithm for the PLOHP; to evaluate the benefits obtained by using the presolving and block decomposition strategies previously described; and finally, to allow the analysis of larger data sets with respect to the ones currently analyzed. As in Catanzaro et al [5] and Halldorsson et al [21], we emphasize that our experiments simply aim to evaluate the computational performance of the exact algorithms. We neither attempt to study the efficiency of the *Branch&Price* algorithm to predict LOH events across the genomes of a population of individuals nor to compare its accuracy versus LOH predictors that do not use the parsimony criterion. This analysis has been performed by [7, 21, 27] and [25] and we refer the interested reader to these articles for further information.

4.1 Implementation

We implemented the CLHF in ANSI C++ by using FICO Xpress 7.6, Optimizer libraries v26.01.04 (64-bit Hyper capacity). This implementation complies with the methodology described in Catanzaro et al [5]. We implemented the *Branch&Price* algorithm in ANSI C++ by using SCIP Optimization Suite 3.1.0 [1] to handle the column generation and the *Branch&Price* routines. Moreover, we used FICO Xpress Optimizer as linear programming solver. The experiments have been performed on an Intel Core i7-4930K CPU, 3.40GHz, equipped with 64 GByte RAM and operating system Ubuntu release 12.10 (kernel Linux 3.5.0-41-generic). During the runtime of the CLHF we activated Xpress automatic cuts, Xpress presolving strategy, and Xpress primal heuristic to generate the first upper bound for the PLOPH. Finally, similar to Halldórsson *et al.* [21] and Catanzaro *et al.* [5], we set the maximum runtime for the Formulations to 5 hours. The executable codes used in the experiments can be downloaded together with the instances at `http://homepages.ulb.ac.be/~lporrett/PLOHP/Code.tar.bz2`.

4.2 Calibrating the pricing oracle

Determining the number of variables to be added in the RMP at each iteration is crucial for the performances of the proposed *Branch&Price*. To perform this task, we generated a learning set of instances of the PLOHP constituted by 100 random instances. We generated such a set by using the same approach and parameters discussed in [21]. Each random instance has between 35725 and 66670 point-intervals

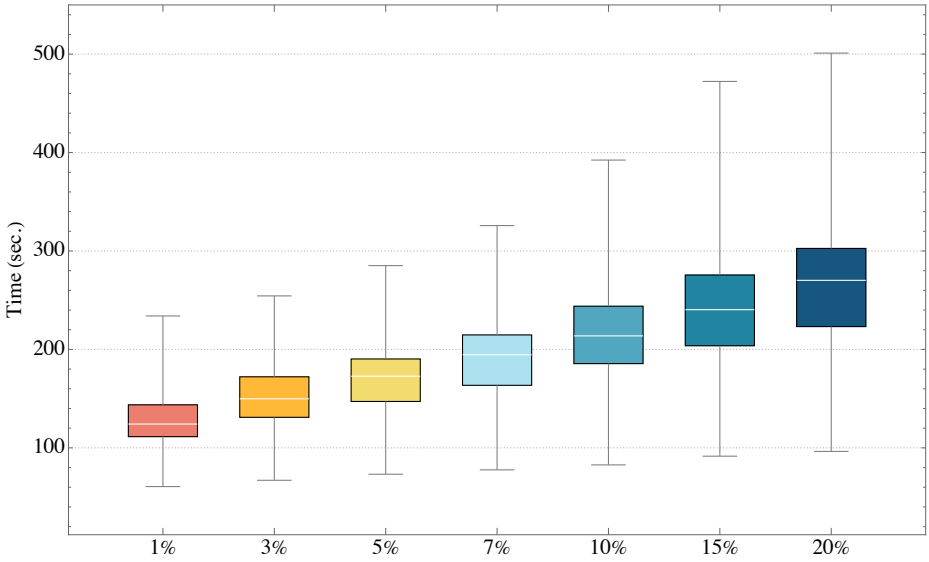

Fig. 4: BoxPlot of the solution time taken by the *Branch&Price* algorithm to exactly solve the learning set of instances of the PLOHP when considering different percentages of violated cliques.

and between 16915 and 52522 cliques on the real segment $[1, 1000]$. Table 1 reports on the main characteristics of this set. We introduced a parametrized number of violated cliques to be added by the pricing oracle at runtime. Such a number is a fraction of the overall number of maximal cliques contained in a random instance and we set it to be equal to one of the following seven different percentages: 1%, 3%, 5%, 7%, 10%, 15% and 20%. Figure 4 shows the box-and-whisker plot histogram of the solution time taken by the *Branch&Price* algorithm to exactly solve the learning set of instances of the PLOHP when considering different percentages of violated cliques. A box shows the range between the 25% and the 75% quantile of the data. The median of the data is indicate by a bar. The whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range from the box. The figure shows that the median of the solution time taken by the *Branch&Price* algorithm to exactly solve the learning set of instances of the PLOHP is minimum when adding in the RMP no more than 1% of violated cliques. To exclude the presence of statistical equivalence between the

	1%	3%	5%	7%	10%	15%	20%
1%	—	1.97784×10^{-18}	1.97784×10^{-18}	1.97784×10^{-18}	1.97796×10^{-18}	1.97796×10^{-18}	1.97796×10^{-18}
3%	1	—	2.95093×10^{-17}	1.97796×10^{-18}	1.97796×10^{-18}	1.97784×10^{-18}	1.97796×10^{-18}
5%	1	1	—	1.79421×10^{-17}	2.03865×10^{-18}	1.97796×10^{-17}	1.97784×10^{-18}
7%	1	1	1	—	6.37311×10^{-18}	1.97784×10^{-18}	1.97796×10^{-18}
10%	1	1	1	1	—	4.83732×10^{-17}	2.16549×10^{-18}
15%	1	1	1	1	1	—	2.4767×10^{-18}
20%	1	1	1	1	1	1	—

Table 2: Wilcoxon signed-rank test on solution time for each pair of percentages

	Reference	Overall point Intervals	Edges	Connected Components	Cloned point Intervals	Singletons
gens-1_α	[1,1 500]	84 435	49 690 310	29	59 666	3
gens-2_α	[1,1 500]	84 436	49 681 938	29	59 659	0
gens-3_α	[1,1 500]	84 427	49 669 238	30	59 669	0
gens-4_α	[1,1 500]	84 440	49 660 650	29	59 656	5
gens-5_α	[1,1 500]	86 467	50 023 009	37	58 052	5
gens-1_β	[1,2 000]	245 205	289 645 236	37	191 817	4
gens-2_β	[1,2 000]	245 198	289 599 050	39	191 832	4
gens-3_β	[1,2 000]	245 177	289 589 450	40	191 907	3
gens-4_β	[1,2 000]	245 197	289 527 061	38	191 904	7
gens-5_β	[1,2 000]	250 539	291 137 895	33	187 199	4
gens-1_γ	[1,3 575]	645 572	1 137 465 967	61	526 689	13
gens-2_γ	[1,3 575]	645 552	1 137 474 258	65	526 688	9
gens-3_γ	[1,3 575]	645 510	1 137 451 358	62	526 657	11
gens-4_γ	[1,3 575]	645 543	1 137 258 659	62	526 703	13
gens-5_γ	[1,3 575]	659 885	1 143 798 666	40	514 953	2

Table 3: Statistics of gens-x_α , gens-x_β , gens-x_γ

solution times when considering different percentages of violated cliques, we run a Wilcoxon signed-rank test with Holmes correction for multiple tests [12]. Table 2 reports on the p-values so obtained. Specifically, the presence of a value smaller then 0.05 in a cell means that the sample in the row is statistically significant smaller than the sample in the column. In contrast, the presence of a value bigger than 0.05 in a cell means that the sample in the row is statistically significant bigger than the sample in the column. In particular, Table 2 shows that adding the 1% of the violated cliques always provide a smaller solution time compared to the others approaches. For this reason, we decided to use this setup in applying the *Branch&Price* algorithm to the remaining experiments.

4.3 Benchmark instances

In order to compare the performance of the *Branch&Price* algorithm with the ones obtained by the CLHF [5], we considered the instances of the PLOHP described in [5] and [21]. These instances, hereafter denoted as "gens-1", "gens-2", "gens-3", "gens-4" and "gens-5", are characterized by having 645572, 645552, 645510, 645543 and 659885 point-intervals on the real segment $[1, 3575]$, respectively, each having minimum length 1 and maximum length 9. Catanzaro et al observed that the ILP formulation presented in [5] proved unable to solve any of these instances within 1 hour of computing time. Hence, the authors also considered smaller instances of the PLOHP obtained from the given ones by extracting from each "gens-x", $x \in \{1, 5\}$, the point-intervals contained in the real segments $\alpha = [1, 1500]$, $\beta = [1, 2000]$ and $\gamma = [1, 3575]$, respectively. The authors obtained this way 15 instances of the PLOHP, that are hereafter grouped into three datasets denoted as "gens-x_α", "gens-x_β" and "gens-x_γ", $x \in \{1, 5\}$, respectively. Table 3 shows the main characteristics of the 15 instances. In particular, the table is composed by three main sections, each one corresponding to the datasets "gens-x_α", "gens-x_β" and "gens-x_γ", $x \in \{1, 5\}$, previously described. The first three columns of Table 3 report on the size of the reference segment, the overall number of point-intervals and edges

Set of Instances	Point-interval Length	Max. number of Point-intervals	Min. number of Point-intervals	Average number of Point-intervals
Sim-1	2	23963	23777	23851 ±52.79
Sim-2	5	24012	23823	23950.5±58.03
Sim-4	7	23615	23454	23553.5±50.67
Sim-3	9	23777	23803	23828 ±42.55
Sim-5	random in [1,9]	29020	28699	28904 ±94.11

Table 4: Parameter sets used to generate random instances of the PLOHP.

contained in a given instance. The last three columns report on the number of Connected components, cloned intervals and singletons contained in a given instance. Table 3 shows that the instances "gens-x$_\alpha$", $x \in \{1, 5\}$ are characterized by a number of connected components ranging from 29 to 37, a number of singletons ranging from 0 to 5 and over 70% of cloned point-intervals. The instances "gens-x$_\beta$", $x \in \{1, 5\}$ are characterized by a number of connected components ranging from 33 to 40, a number of singletons ranging from 3 to 7 and over 70% of cloned point-intervals. Finally, the instances "gens-x$_\gamma$", $x \in \{1, 5\}$ are characterized by a number of connected components ranging from 40 to 61, a number of singletons ranging from 2 to 13 and over 80% of cloned point-intervals.

We also considered a set of 5 artificial datasets of the PLOHP, hereafter denoted as "Sim-1", "Sim-2", "Sim-3", "Sim-4" and "Sim-5", each containing 10 random instances of the problem and mainly differing from one another both by the length and the number of the point-intervals contained in each of the corresponding instances. We generated the instances in each dataset by first fixing the real segment $[1, 6000]$ as a reference and by generating, by means of the Mersenne Twister library [24], a random number of triplets (l_v, p_v, r_v) such that $1 \leq l_v \leq p_v \leq r_v \leq 6000$. Table 4 summarizes the characteristics of the considered datasets. Finally, in order to evaluate the performance of the *Branch&Price* algorithm also on biological instances of the PLOHP, we considered a biological instance provided by deCode Genetics and hereafter called "Bio22". This instance is constituted by 20022 point-intervals on the real segment $[1, 6000]$, each having minimum length 2 and maximum length 6000. The point-intervals in "Bio22" refers to genetic fragments from chromosome 22 extracted from a population of 18360 individuals and consists of 6000 single nucleotide polymorphisms. All of the instances used in our experiments can be downloaded at `http://homepages.ulb.ac.be/~lporrett/PLOHP/Datasets.tar.bz2`.

4.4 Computational performances

Table 5 shows the performances of the CLHF and the *Branch&Price* in solving the instances derived from the 5 real instances described in [21] with and without the presolving strategies. In particular, the table is constituted by three main sections, each one corresponding to the datasets "gens-x$_\alpha$", "gens-x$_\beta$" and "gens-x$_\gamma$", $x \in \{1, 5\}$, previously described. Moreover, the columns are divided in four sections, each of them showing the solution time (expressed in seconds) and the number of cliques needed to optimally solve a given instance of the PLOHP. Specifically,

	CLHF		B&P		CLHF + Preprocessing		B&P + Preprocessing	
	Time (sec.)	Cliques	Time (sec.)	Cliques	Time (sec.)	Cliques (Avg.)	Time (sec.)	Cliques (Avg.)
gens-1_α	1 062.46	63660	316.99	1 845	70.83	2 490.00± 3 684.44	63.06	587.59 ±1 410.13
gens-2_α	1 145.84	63784	302.88	1 859	68.14	2 497.38± 3 686.79	58.91	256.62 ± 563.91
gens-3_α	1 090.85	63303	312.52	1 845	63.42	2 288.30± 2 491.55	56.54	166.97 ± 166.97
gens-4_α	1 035.16	63507	301.02	1 871	67.67	2 421.41± 2 719.01	62.13	551.07 ± 877.04
gens-5_α	1 198.22	69266	378.390	2 852	74.19	2 055.49± 2 118.55	68.93	551.07 ± 193.30
gens-1_β	9 039.54	148 862	2 455.14	3 532	350.39	5 095.03± 9 235.71	314.69	1 267.97±4 578.73
gens-2_β	8 965.42	148 505	2 743.00	3 521	313.93	4 843.21± 7 851.87	267.17	534.77 ±1 338.22
gens-3_β	8 901.90	148 042	2 562.65	3 510	292.18	4 691.43± 7 651.59	258.19	305.28 ± 756.50
gens-4_β	8 925.68	147 249	2 630.90	3 508	328.28	4 909.92± 7 926.41	279.21	797.00 ±2 868.75
gens-5_β	9 488.87	138 627	3 805.31	5 739	500.18	5 033.58± 7 523.26	396.62	906.76 ±2 107.23
gens-1_γ	>5h	—	>5h	—	2 005.03	7 219.23±15 025.36	1 800.91	752.97 ±2 334.55
gens-2_γ	>5h	—	>5h	—	1 901.62	6 939.11±14 928.34	1 708.04	525.54 ±1 390.90
gens-3_γ	>5h	—	>5h	—	1 973.48	7 227.92±15 273.65	1 757.70	604.76 ±1 881.98
gens-4_γ	>5h	—	>5h	—	1 978.90	7 065.68±15 232.85	1 754.85	252.03 ± 579.87
gens-5_γ	>5h	—	>5h	—	3 292.84	9 311.35±11 355.82	2 859.05	437.95 ± 635.70

Table 5: Computational performances comparison of the CLHF and the *Branch&Price* algorithm with and without presolving strategies

the first pair of columns is related to the execution of the CLHF. The second pair of columns is related to the execution of the *Branch&Price*. The third pair of columns is related to the execution of the CLHF using the presolving strategies. Finally, the fourth pair of columns are related to the execution of the *Branch&Price* using the presolving strategies. Table 5 shows the benefits of using the presolving strategies. Specifically, we have observed the removal of at least 70% of cloned intervals (hence x_v variables in the CLHF) in each of the considered instances. Moreover, the detection of the connected components in the considered instances enabled the decomposition of the corresponding induced MPT graphs into at least 29 connected components (i.e., independent subproblems); the combination of the two strategies allowed a speed up of about 18x of the solution times taken by the CLHF. Table 5 also shows that, independently of the presence (or absence) of the presolving strategies, the *Branch&Price* is always faster than the CLHF. This fact alone justifies and shows the importance of using column generation when dealing with very large instances of the PLOHP. In this context, it is worth noting that, although the *Branch&Price* is faster than the CLHF, it also proves unable to solve the instances "gens-x$_\gamma$", $x \in \{1, 5\}$, within the considered time limit. This fact is mainly due to the pricing oracle, which proves particularly time-consuming in gens-x$_\gamma$.

4.5 Computational performances on larger datasets of the PLOHP

Table 6 shows the performance of the CLHF and the *Branch&Price* in solving the "Sim" instances when using the presolving strategies. Both the CLHF and the *Branch&Price* algorithm without presolving strategies proved unable to solve any instance in this set. The table reports on the mean and standard deviation of (i) the solution time (expressed in seconds) necessary to solve a random instance of the PLOHP; (ii) the gap (expressed in percentage), i.e., the difference between the optimal value to a given instance of the PLOHP in a specific dataset and the objective function value of the linear programming relaxation at the root node of the respective search tree, divided by the optimal value; and (iii) the nodes explored in the search tree. In general, Table 6 shows that the CLHF and the *Branch&Price*, combined with presolving strategies, is able to solve each simulated

instance in about half an hour, except for the parameter set "Sim-5" that required on average 26 minutes for the *Branch&Price* while the CLHF exceeds the 64GB of memory available for the experiments. The table also shows that the solution times necessary to solve the artificial instances of the PLOHP are longer than the corresponding ones necessary to solve any other instance considered in this article. This fact is possibly due to the larger number of point-intervals present in the artificial instances which in turn increases the number of cliques in the corresponding MPT graphs.

Table 7 reports on the minimum, maximum and average number of connected components, intervals, singletons, edges and cloned nodes in the considered simulated instances. Table 8 reports on the minimum, maximum and average time (expressed in seconds) to filter both the singletons and the cloned intervals as well as to identify the connected components. Table 9 shows the performance of the *Branch&Price* in solving "Bio22" instance using the presolving strategies. The table reports on (i) the solution time (expressed in seconds) necessary to solve the biological instance of the PLOHP; (ii) the gap (expressed in percentage); (iii) the nodes explored in the search tree; and (iv) the overall number of connected components, intervals, cloned intervals and singletons, respectively.

Table 9 shows that the proposed *Branch&Price*, combined with presolving strategies, is able to solve "Bio22" in 664.40 seconds. This instance is characterized by having a number of point-intervals smaller than those relative to the instances considered in Table 3 as well as by a smaller number of cloned ones. Filtering the cloned point-intervals in "Bio22" reduces the number of x_v variables by the 20% as opposed to 70% in Table 5. However, the relative higher number of connected components in "Bio22" allows to decompose the problem into a bigger family of small independent subproblems; this fact justifies the remarkable speed up in terms of solution time.

5 Conclusion

The *Parsimonious Loss of heterozygosity Problem* (PLOHP) is a $\mathcal{NP}$-hard combinatorial optimization problem consisting of solving the minimum cost clique cover problem on a MPT graph. The optimal solutions to the PLOHP have a remarkable importance in practice, as they enable the association of major human diseases with chromosome regions from patients that exhibit loss of heterozygosity events. As genome-wide association studies usually involve the analysis of massive amount

	CLHF			**Branch&Price**		
Instance	Time (sec.)	Gap (%)	Nodes	Time (sec.)	Gap (%)	Nodes
Sim-1	1 217.96±41.14	0.017±0.18	1.012±0.20	577.77±18.24	0.003±0.13	1.003±0.12
Sim-2	1 200.47±39.02	0.009±0.17	1.010±0.18	552.44±40.46	0.004±0.12	1.004±0.11
Sim-3	1 200.24±48.61	0.007±0.13	1.010±0.19	575.27±32.15	0.001±0.05	1.001±0.06
Sim-4	1 162.67±94.31	0.007±0.14	1.012±0.26	509.01±84.75	0.003±0.09	1.003±0.08
Sim-5	**OOM±OOM**	**OOM±OOM**	**OOM±OOM**	969.56±424.29	0.002±0.09	1.001±0.06

Table 6: Computational performances comparison of the CLHF+Preprocessing and the *Branch&Price* algorithm + Preprocessing on artificial instances of the PLOHP.

	Sim1			**Sim2**		
	Min	*Max*	*Average*	*Min*	*Max*	*Average*
# Conn. Comp.	293	311	298	291	307	295
# Intervals	20 562	20 763	20 689	205 270	20 693	20 587
# Singletons	18	32	24	16	30	26
# Edges	2 824 349	2 922 094	2 881 541	281 133	2 889 021	2 854 612
# Cloned Inter.	3 218	3 243	3 237	3 223	3 255	3 241
	Sim3			**Sim4**		
	Min	*Max*	*Average*	*Min*	*Max*	*Average*
# Conn. Comp.	288	303	298	279	305	293
# Intervals	20 539	20 670	20 567	20 187	20 359	20 312
# Singletons	16	30	23	18	31	26
# Edges	2 810 257	2 862 659	2 833 031	2 728 854	2 767 026	2 744 176
# Cloned Inter.	3 222	3 254	3 242	3 221	3 241	3 233
	Sim5					
	Min	*Max*	*Average*			
# Conn. Comp.	259	291	28			
# Intervals	25 328	25 649	25 525			
# Singletons	5	16	11			
# Edges	4 688 361	4 810 792	4 760 66			
# Cloned Inter.	3 358	3 391	3 371			

Table 7: Statistics of simulated instances $Sim1$, $Sim2$, $Sim3$, $Sim4$, $Sim5$.

	Sim1			**Sim2**		
	Min	*Max*	*Average*	*Min*	*Max*	*Average*
Singleton	0.43	0.52	0.44	0.43	0.52	0.43
Cloned Inter.	0.42	0.43	0.42	0.42	0.50	0.42
Conn. Comp.	$6.2{\cdot}10^{-5}$	$1.3{\cdot}10^{-5}$	$6.2{\cdot}10^{-5}$	$6.1{\cdot}10^{-5}$	$1.7{\cdot}10^{-5}$	$6.2{\cdot}10^{-5}$
	Sim3			**Sim4**		
	Min	*Max*	*Average*	*Min*	*Max*	*Average*
Singleton	0.43	0.52	0.43	0.41	0.42	0.41
Cloned Inter.	0.42	0.50	0.42	0.40	0.41	0.42
Conn. Comp.	$6.1{\cdot}10^{-5}$	$6.2{\cdot}10^{-5}$	$6.2{\cdot}10^{-5}$	$6.0{\cdot}10^{-5}$	$1.3{\cdot}10^{-5}$	$6.1{\cdot}10^{-5}$
	Sim5					
	Min	*Max*	*Average*			
Singleton	0.73	0.78	0.74			
Cloned Inter.	0.65	0.66	0.65			
Conn. Comp.	$7.7{\cdot}10^{-5}$	$7.9{\cdot}10^{-5}$	$7.8{\cdot}10^{-5}$			

Table 8: The minimum, the maximum and the average time (expressed in seconds) both to filter the singletons and the cloned intervals, and to identify the connected components in $Sim1$, $Sim2$, $Sim3$, $Sim4$, $Sim5$.

of data, practical instances of the PLOHP may have very large sizes. This fact precludes the use of exact solution approaches to the PLOHP such as those described in [5, 21]. In this article, we have investigated ways (i) to overcome this limit, (ii) to speed up the solution time of the best known exact solution algorithm for the

PLOHP, and (iii) to enlarge the size of the instances that can be optimally solved within the same time limit. We developed a number of preprocessing strategies able to both dramatically reduce the size of the instances analyzed and decompose the reduced instances into a collection of small independent subproblems. These strategies allowed to speed up of about 8 times the solution times of the best known exact solution algorithm for the PLOHP as well as enabled the resolution, within the same time limit, of instances previously too large to be tackled. Moreover, we combined such preprocessing strategies with a *Branch&Price* algorithm based on column generation techniques able to exploit the combinatorial properties of the MPT graphs. The overall performances of the proposed algorithm proved to be up to 30 times faster than previous approaches described in the literature; such performances could be further enhanced, by parallelizing the way in which the *Branch&Price* algorithm handle the collection of small independent subproblems provided by the preprocessing strategies. This result helps to illustrate the power of ILP approaches to tackle problems arising from genome-wide association studies and characterized by complex sets of constraints. Hopefully, future research efforts will provide new insights on the combinatorics of MPT graphs and will enable the analysis of even larger genomic fragments.

Acknowledgements Part of this work has been developed when L. Porretta and D. Catanzaro were visiting both Reykjavik University and DeCODE Genetics. L. Porretta acknowledges support from the Belgian National Fund for Scientific Research (FRS-FNRS) who funded his short-term research visit. D. Catanzaro acknowledges support from the D. Catanzaro acknowledges support from the FRS-FNRS via the grant "Crédit de Recherche" ref. S/25-MCF/OL J.0026.17, the Université Catholique de Louvain via the "Fonds Spéciaux de Recherche" (FSR) 2017-2019, the Fondation Louvain via the grant "Le numérique au service de l'humain", and the computational resources provided by the "Consortium des Équipements de Calcul Intensif" (CÉCI), funded by the FRS-FNRS via the grant ref. 2.5020.11. The authors thank the anonymous referees, whose insightful suggestions helped to improve the manuscript.

References

1. Achterberg T (2009) Scip: solving constraint integer programs. Mathematical Programming Computation 1(1):1–41
2. Asinowski A, Cohen E, Golumbic MC, Limouzy V, Lipshteyn M, Stern M (2012) Vertex intersection graphs of paths on a grid. Journal of Graph Algorithms and Applications 16:129–150
3. Booth KS, Lueker GS (1976) Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Sciences 13(3):335–379

Instance	Time (sec.)	Gap (%)	Nodes	Connected Components	Point Intervals	Cloned point Intervals	Singletons
Bio22	661.40	0	1	321	19968	3151	54

Table 9: Computational performances of the Branch&Price on biological instance of the PLOHP.

4. Catanzaro D, Labbé M (2009) The pure parsimony haplotyping problem: overview and computational advances. International Transactions in Operational Research 16(5):561–584, URL http://www.ingentaconnect.com/content/bpl/itor/2009/00000016/00000005/art00002
5. Catanzaro D, Labbe M, Halldorsson BV (2013) An integer programming formulation of the parsimonious loss of heterozygosity problem. IEEE/ACM Transactions on Computational Biology and Bioinformatics 10(6):1391–1402
6. Catanzaro D, Chaplick S, Felsner S, Halldórsson BV, Halldórsson MM, Hixon T, Stacho J (2017) Max point-tolerance graphs. Discrete Applied Mathematics 216(1):84–97
7. Conrad DF, Andrews TD, Carter NP, Hurles ME, Pritchard JK (2006) A high-resolution survey of deletion polymorphism in the human genome. Nature Genetics 38(1):75–81
8. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to Algorithms. The MIT Press, Cambridge MA
9. Corneil DG, Kamula PA (1987) Extensions of permutation and interval graphs. In: Proceedings of 18th Southeastern Conference on Combinatorics, Graph Theory and Computing, pp 267–275
10. Diestel R (2010) Graph Theory. Springer Berlin Heidelberg
11. Dion G, Jost V, Queyranne M (2007) Clique partitioning of interval graphs with submodular costs on the cliques. RAIRO Operations Research 41:275–287
12. Fay MP, Proschan MA (2010) Wilcoxon-mann-whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. Statistics Surveys 4:1–39
13. Fishburn P (1985) Interval orders and interval graphs: A study of partially ordered sets. John Wiley & Sons, New York
14. Frank A (1976) Some polynomial algorithms for certain graphs and hypergraphs. In: Proceedings of the 5th British Combinatorial Conf. (Aberdeen 1975), Congressus Numerantium XV, pp 211–226
15. Fulkerson DR, Gross OA (1965) Incidence matrices and interval graphs. Pacific Journal of Mathematics 15:835–855
16. Garey MR, Johnson DS (2003) Computers and Intractability: A guide to the theory of NP-Completeness. Freeman, New York
17. Gavril F (1978) A recognition algorithm for the intersection graphs of paths in trees. Discrete Mathematics 23:211–227
18. Golumbic MC (2004) Algorithmic graph theory and perfect graphs. Elsevier, North-Holland
19. Golumbic MC, Monma CL (1982) A generalization of interval graphs with tolerances. Congressus Numerantium 35:321–331, proceedings of the 13th Southeastern Conference on Combinatorics, Graph Theory and Computing
20. Golumbic MC, Trenk A (2004) Tolerance Graphs, Cambridge Studies in Advanced Mathematics, vol 89. Cambridge University Press
21. Halldorsson BV, Aguiar D, Tarpine R, Istrail S (2011) The Clark phaseable sample size problem: Long-range phasing and loss of heterozygosity in GWASlark phaseable sample size problem: long-range phasing and loss of heterozygosity in GWAS. Journal of Computational Biology 18(3):323–333
22. Kaufmann M, Kratochvil J, Lehmann K, Subramanian A (2006) Max-tolerance graphs as intersection graphs: Cliques, cycles and recognition. In: Proceedings of 17th annual ACM-SIAM Symposium On Discrete Algorithms

SODA 06, SIAM, pp 832–841
23. Martin RK (1999) Large Scale Linear and Integer Optimization: A Unified Approach. Springer US
24. Matsumoto M, Nishimura T (1998) Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Transactions on Modeling and Computer Simulation 8(1):3–30
25. McCarroll SA, Kuruvilla FG, Korn JM, Cawley S, Nemesh J, Wysoker A, Shapero MH, de Bakker PIW, Maller JB, Kirby A, Elliott AL, Parkin M, Hubbell E, Webster T, Mei R, Veitch J, Collins PJ, Handsaker R, Lincoln S, Nizzari M, Blume J, Jones KW, Rava R, Daly MJ, Gabriel SB, Altshuler D (2008) Integrated detection and population-genetic analysis of snps and copy number variation. Nature Genetics 40(10):1166–1174
26. Nemhauser GL, Wolsey LA (1989) Optimization. In: Nemhauser GL, Kan AHGR, Todd MJ (eds) Handbooks in Operations Research and Management Science, Elsevier, North-Holland
27. Speed T, Huang H, Corona E, Raphael B, Eskin E (2007) Identification of Deletion Polymorphisms from Haplotypes, vol 4453, Springer Berlin Heidelberg, pp 354–365
28. Stefansson H, Rujescu D, Cichon S, Pietiläinen OPH, Ingason A, Steinberg S, Fossdal R, Sigurdsson E, Sigmundsson T, Buizer-Voskamp JE, Hansen T, Jakobsen KD, Muglia P, Francks C, Matthews PM, Gylfason A, Halldorsson BV, Gudbjartsson D, Thorgeirsson TE, Sigurdsson A, Jonasdottir A, Jonasdottir A, Bjornsson A, Mattiasdottir S, Blondal T, Haraldsson M, Magnusdottir BB, Giegling I, Moeller HJ, Hartmann A, Shianna KV, Ge D, Need AC, Crombie C, Fraser G, Walker N, Lonnqvist J, Suvisaari J, Tuulio-Henriksson A, Paunio T, Toulopoulou T, Bramon E, Forti MD, Murray R, Ruggeri M, Vassos E, Tosato S, Walshe M, Li T, Vasilescu C, Moehleisen TW, Wang AG, Ullum H, Djurovic S, Melle I, Olesen J, Kiemeney LA, Franke B, Sabatti C, Freimer NB, Gulcher JR, Thorsteinsdottir U, Kong A, Andreassen OA, Ophoff RA, Georgi A, Rietschel M, Werge T, Petursson H, Goldstein DB, Nöthen MM, Peltonen L, Collier DA, Clair DS, Stefansson K (2008) Large recurrent microdeletions associated with schizophrenia. Nature 455:232–236