Universität Bielefeld

Technische Fakultät
Abteilung Informationstechnik
Forschungsberichte

On Parsimony Haplotyping

Ferdinando Cicalese Martin Milanič

Report 2008-04

Impressum: Herausgeber:
Ellen Baake, Mario Botsch, Robert Giegerich, Ralf Hofestädt,
Franz Kummert, Peter Ladkin, Ralf Möller, Tim Nattkemper,
Helge Ritter, Gerhard Sagerer, Jens Stoye, Ipke Wachsmuth

Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31,
33501 Bielefeld, Germany

ISSN 0946-7831

On Parsimony Haplotyping[*]

FERDINANDO CICALESE AND MARTIN MILANIČ
AG Genominformatik, Faculty of Technology, Bielefeld University, Germany
{nando,mmilanic}@cebitec.uni-bielefeld.de

We present some structural and algorithmic results related to the parsimony haplotyping problem.

1 Preliminaries

A *haplotype* is a 0-1 vector $h \in \{0,1\}^n$. A *genotype* is a vector $g \in \{0,1,2\}^n$. We say that a haplotype h is *consistent with* a genotype g (or equivalently, that g is consistent with h) if $h(i) = g(i)$ whenever $g(i) \in \{0,1\}$. We say that two haplotypes h_1 and h_2 *resolve* (or *parse*) a genotype g if, for every $i \in \{1,\ldots,n\}$, we have $g(i) = \begin{cases} h_1(i), & \text{if } h_1(i) = h_2(i); \\ 2, & \text{otherwise.} \end{cases}$ We denote this relation by $g = h_1 \oplus h_2$.

Fact 1. *For any given pair of haplotypes h_1, h_2 in $\{0,1\}^n$, there is a unique genotype resolved by h_1 and h_2. For every genotype g and for every haplotype h consistent with g, there is a unique haplotype h' such that $g = h \oplus h'$. Such a haplotype h' is called the complement of h w.r.t. g. If $g = h_1 \oplus h_2$ then both h_1 and h_2 are consistent with g.*

A *genotype matrix* $\mathbf{G}$ is a matrix whose rows are genotypes, that is, a matrix in $\{0,1,2\}^{m\times n}$. Similarly is defined a *haplotype matrix* $\mathbf{H}$. With a slight abuse of terminology, we shall identify every genotype matrix with the set of genotypes defining it. We say that a set of haplotypes $\mathbf{H}$ *resolves* a set of genotypes $\mathbf{G}$ if, for every genotype $g \in \mathbf{G}$, there exist $h_1, h_2 \in \mathbf{H}$ such that $g = h_1 \oplus h_2$.

The PARSIMONY HAPLOTYPING problem is the following: Given a set of genotypes $\mathbf{G}$, find a set $\mathbf{H}$ of haplotypes that resolves $\mathbf{G}$ and is of minimum cardinality.

This problem is known to be NP-hard in general, however, to the best of our knowledge, its complexity is still undetermined for the case of $(*,2)$-bounded instances. A genotype matrix $\mathbf{G}$ is (k,l)*-bounded,* if it has at most k 2s per row and at most l 2s per column. It is $(k,*)$*-bounded,* if it has at most k 2s per row, and $(*,l)$*-bounded,* if it has at most l 2s per column.

Two genotypes are said to be *compatible* if there exists a haplotype that is consistent with both. Thus, two genotypes are compatible if and only if there is no position at which one genotype takes value 0 and the other 1. Given a set of genotypes $\mathbf{G} \in \{0,1,2\}^{m\times n}$, its *compatibility graph* is the graph $G = (V(G), E(G))$ where $V(G) = \{g : g \in \mathbf{G}\}$ and $E(G) = \{\{g,g'\} : g \text{ and } g' \text{ are compatible}\}$.

[*]This work was supported by the Sofja Kovalevskaja Award 2004 of the Alexander von Humboldt Foundation and the Bundesministerium für Bildung und Forschung.

We will say that a graph G is a $(*, 2)$-*bounded compatibility graph* if it is isomorphic to the compatibility graph of some $(*, 2)$-bounded set of genotypes.

2 Structure of $(*, 2)$-bounded compatibility graphs

In this section, we study the structure and properties of $(*, 2)$-bounded compatibility graphs.

Let a, b be two vertices in a graph G. Following the terminology of Diestel [1], we say that two a-b paths are *independent* if a and b are their only common vertices.

Lemma 1. *A graph G is a $(*, 2)$-bounded compatibility graph if and only if for every pair $\{x, y\}$ of (distinct) non-adjacent vertices in G, there are at most two independent x-y paths.*

Proof. Let $\mathbf{G}$ be a $(*, 2)$-bounded genotype matrix, and let G be its compatibility graph. Suppose that there is a pair g, g' of non-adjacent vertices in G with three independent g-g' paths P_1, P_2, P_3. Since g, g' are non-adjacent vertices in G, they are incompatible as genotypes, so there is a position j such that (up to symmetry) $g(j) = 0$ and $g'(j) = 1$. It follows that each of the paths P_i must contain an internal vertex g^i such that $g^i(j) = 2$. (Otherwise, there would be two consecutive vertices g^r, g^{r+1} on P such that $g^r(j) = 0$ and $g^{r+1}(j) = 1$, which would contradict the fact that they are adjacent.) Since the paths are independent, the genotypes g^1, g^2, g^3 are all distinct. The fact that all of them have a 2 at position j contradicts the assumption that the instance was $(*, 2)$-bounded.

For the converse direction, suppose that G is a graph such that for every pair $\{x, y\}$ of distinct non-adjacent vertices in G, there are at most two independent x-y paths. We shall construct a $(*, 2)$-bounded genotype matrix $\mathbf{G}$ such that G is the compatibility graph of $\mathbf{G}$.

If G is a complete graph, then $\mathbf{G}$ is the $|V(G)| \times |V(G)|$ matrix given by

$$\mathbf{G}_{i,j} = \begin{cases} 2, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} .$$

Clearly, $\mathbf{G}$ satisfies the desired properties.

In the rest of the proof, we assume that G is not complete. We start with an empty matrix. For each pair $\{x, y\}$ of (distinct) non-adjacent vertices in G, we construct a column C^{xy} whose rows are indexed by the vertices of G, and add C^{xy} to $\mathbf{G}$.

We construct C^{xy} as follows. Since in G there are at most two independent x-y paths, Menger's theorem (see e.g. [1]) implies that there is a set $S \subseteq V(G)$ with $|S| \leq 2$ that separates x from y. Let C_x be the connected component of $G - S$ containing x. Then, for each vertex $v \in V(G)$, we let

$$C^{xy}(v) = \begin{cases} 2, & \text{if } v \in S; \\ 0, & \text{if } v \in C_x; \\ 1, & \text{otherwise.} \end{cases}$$

It follows by construction that $\mathbf{G}$ has at most two 2's per column. Let G' be the compatibility graph of $\mathbf{G}$ such that $V(G') = V(G)$. We need to show that $G = G'$. Suppose that x and y are adjacent in G. Then, to see that x and y are also adjacent in G', it is enough to observe that there is no column j of $\mathbf{G}$ such that

$\{\mathbf{G}_{xj}, \mathbf{G}_{yj}\} = \{0, 1\}$. Conversely, if x and y are non-adjacent in G, then x and y are also non-adjacent in G', since they will disagree in the column C^{xy}. The proof is complete. $\square$

Remark 1. *The statement and the above proof of Lemma 1 generalize naturally to $(*, k)$-bounded compatibility graphs. For every $k \geq 1$, a graph G is a $(*, k)$-bounded compatibility graph if and only if for every pair $\{x, y\}$ of (distinct) non-adjacent vertices in G, there are at most k independent x-y paths.*

Corollary 1. *Every graph of maximum degree k is a $(*, k)$-bounded compatibility graph.*

For a pair $\{a, b\}$ of (distinct) non-adjacent vertices in G, we will refer to an *($\{a, b\}$-)theta* as the union of three independent a-b paths. Therefore, Lemma 1 implies that no $(*, 2)$-bounded compatibility graph contains a theta.

We denote by $\omega(G)$ the size of a maximum clique in a graph G, and by $tw(G)$ its tree-width. In the following lemma, we show that the tree-width of $(*, 2)$-bounded compatibility graphs can only be large due to the presence of a large clique.

Lemma 2. *For every k there is an N such that the tree-width of every $(*, 2)$-bounded compatibility graph G with $\omega(G) \leq k$ is at most N.*

Proof. (Sketch.) For $k \geq 1$, let $\mathcal{G}_k$ denote the set of all $(*, 2)$-bounded compatibility graphs G such that $\omega(G) \leq k$. By a result of Robertson and Seymour [2], graphs of large tree-width must contain a large grid as a minor. Therefore, to show that the tree-width of graphs in $\mathcal{G}_k$ is uniformly bounded, it is enough to show that there is an integer $f(k)$ such that no graph from $\mathcal{G}_k$ contains an $f(k) \times f(k)$ grid as a minor.

We will show that $G \in \mathcal{G}_k$ cannot contain a $3 \times (k + 1)$ grid as a minor. Suppose, by contradiction, that there is a graph $G \in \mathcal{G}_k$ containing a $3 \times (k + 1)$ grid $H = (V(H), E(H))$ as a minor. Then, there exists a collection $\mathcal{C} = \{V_x : x \in V(H)\}$ of pairwise disjoint subsets of $V(G)$ each of which induces a connected graph, and such that there is an edge from a vertex in V_x to a vertex in V_y whenever $\{x, y\} \in E(H)$.

Let $\mathcal{C}' \subseteq \mathcal{C}$ denote the subset of $\mathcal{C}$ consisting of the $2k + 1$ sets V_x corresponding to the middle row of the grid. We enumerate the members of $\mathcal{C}'$ as $V_{x_1}, \ldots, V_{x_{k+1}}$, and for each $i \in \{1, \ldots, k + 1\}$, fix a vertex $v_i \in V_{x_i}$ such that G contains three paths P_x, P_y, P_z connecting v_i to V_x, V_y, V_z respectively where x, y, z are three distinct neighbors of x_i in H, such that v_i is the only common vertex of any two of the paths P_x, P_y, P_z. Since $\omega(G) \leq k$, the set $C = \{v_1, \ldots, v_{k+1}\} \subseteq V(G)$ is not a clique. Thus, there exist two non-adjacent vertices v_i, v_j in C. Using the defining property of the vertices v_i and v_j and the structure of the grid, it is now possible to find to a $\{v_i, v_j\}$-theta in G. We leave to the reader the straightforward verification of this claim.

By Lemma 1, the presence of a theta contradicts the fact that G is a $(*, 2)$-bounded compatibility graph. The proof is complete. $\square$

Remark 2. *The analogue of Lemma 2 does not hold for $(*, 3)$-bounded compatibility graphs. (There exist triangle-free graphs of maximum degree 3 and of arbitrarily large tree-width.)*

In the following lemma, we prove another property of these graphs. A *simplicial vertex* in a graph G is a vertex of G whose neighborhood induces a complete graph. An edge $e \in E(G)$ is a *degree-2 edge* if both of its endpoints have degree two in G.

Lemma 3. *Every $(*, 2)$-bounded compatibility graph contains either a simplicial vertex or a degree-2 edge.*

Proof. We say that a $(*, 2)$-bounded compatibility graph G is *bad* if it has no simplicial vertices and degree-2 edges. Suppose that there exists a bad graph. Let $n \geq 1$ be the smallest number of vertices that a bad graph can have. Among all bad graphs on n vertices, let G be one with the maximum number of edges.

Then:

- G is connected.

  This is clear since every connected component of a bad graph is bad.

- G has no vertices of degree at most 2.

  Clearly, G cannot have a vertex of degree 0 or 1 since such a vertex would be simplicial.

  Suppose that $x \in V(G)$ is a vertex of degree 2, and let a, b denote the two neighbors of x. Then, the vertices a and b are not adjacent (otherwise x would be simplicial). Moreover, since G has no degree-2 edges, we have $d(a) \geq 3$, $d(b) \geq 3$.

  We claim that x is the unique vertex in G that is adjacent only to a and b. Indeed, there is a $y \neq x$ with $N(y) = \{a, b\}$, then it is easy to verify that the graph G' obtained from G by adding to it the edge $\{x, y\}$, is a bad graph on n vertices and with more edges than G, contradicting the choice of G.

  (Clearly, G' cannot have a degree-2 edge. Suppose that v is a simplicial vertex in G'. Then v must be adjacent to both x and y, so we may assume without loss of generality that $v = a$. However, the degree of a is at least 3, so there is a vertex z that is adjacent to a but not to x, contradicting the assumption that a was simplicial. Therefore, G' has no simplicial vertices. Finally, suppose that G' contains a $\{u, v\}$-theta T. Then, the edge $\{x, y\}$ must appear in it, which in turn implies that the vertices a, b, x, y must appear together either as (a, x, y, b) or as (a, y, x, b) on one of the u-v paths in T. But then, replacing such a sequence of with (a, x, b) yields a $\{u, v\}$-theta in G, contrary to the assumption that G is a $(*, 2)$-bounded compatibility graph.)

  Let G' be the graph obtained from G by contracting the edge $\{a, x\}$. (Formally, $V(G') = V(G) \backslash \{x\}$ and $E(G') = E(G) \backslash \{\{x, a\}, \{x, b\}\} \cup \{a, b\}$.) Then, since G' has fewer vertices than G, it cannot be bad. There are three possible reasons for this, and we will show that neither of them can occur. First, note that G' cannot contain a $\{u, v\}$-theta since replacing in such a theta the edge $\{a, b\}$ (if it is there) with the path (a, x, b) would result in a $\{u, v\}$-theta in G. Next, we see that G' cannot have a degree-2 edge (as $d(a) \geq 3$, $d(b) \geq 3$, such an edge would have to be disjoint with $\{a, b\}$ and would thus also be a degree-2 edge in G). Thus, G' must have a simplicial vertex v. Then, v must either coincide with one of a, b, or be adjacent to both a and b. Suppose that $v = a$. (The case $v = b$ is handled similarly.) Since $d(a) \geq 3$, there exist two distinct vertices u, u' in $N(a) \backslash \{b\}$. However, if a is simplicial, then (a, x, b), (a, u, b) and (a, u', b) are three independent $\{a, b\}$ paths in G, a contradiction. Thus, $v \neq a$, and v must be adjacent to both a and b. It follows from the above observation that x is the unique vertex in G that is adjacent only to a and b, that v must be adjacent to a vertex c different from a and b. Again, assuming that v is a simplicial vertex in G', we conclude that (a, x, b), (a, v, b) and (a, c, b) are three independent $\{a, b\}$ paths in G, which is impossible.

- G has no cut vertices.

  Suppose, for the sake of contradiction, that G contains a cut vertex x. Consider a pair C, D of two distinct connected components of $G - \{x\}$, and let c and d denote two neighbors of x in C and D respectively. Then, since x separates the vertices c and d in G, they cannot be adjacent. Let G' be the graph obtained from G by adding to it the edge $\{c, d\}$. Then:

- G' has no degree-2 edges (since its minimum degree is at least 3).
- G' has no simplicial vertices. (Since the degree of c is at least three, there is a vertex adjacent to c but not to d, which shows that c (and similarly d) cannot be simplicial in G'. Similarly x cannot be simplicial since otherwise any neighbor of x other than c and d would have to adjacent to both c and d, contradicting the fact that x separates c and d. Any simplicial vertex in G' other than c, d, x would also be simplicial in G.)
- G' has no thetas. (Otherwise, let T be a $\{u, v\}$-theta in G'. Then, since T is not a theta in G, it must use the edge $\{c, d\}$, which implies (up to symmetry) that $u \in C$ and $v \in D$. However, deleting the edge $\{c, d\}$ as well as the vertex x from T leaves at least one u-v path in G' (and thus in G) avoiding x, contradicting the fact that x separates u and v.)

So, G' is a bad graph on n vertices with more edges than G, contradicting the choice of G.

- G has no cutset consisting of two non-adjacent vertices.

  Suppose that $C = \{a, b\}$ is a cutset in G consisting of two non-adjacent vertices. Then, for every connected component K of $G - \{a, b\}$, there must exist an a-b path P_K whose internal vertices all belong to K. (Otherwise, if K connected to $\{a, b\}$ only through a (b), then a (b) would be a cut vertex in G.)

  Since each of these components contributes an a-b path whose internal vertices all belong to K, and since G has no $\{a, b\}$-theta, we conclude that the graph $G - \{a, b\}$ consists of at most two connected components. On the other hand, since $\{a, b\}$ is a cutset, there must be at least two such components. Let us refer to these two components as K_1 and K_2. Let a_1 denote a neighbor of a in K_1; vertices a_2, b_1, b_2 are defined similarly.

  Let G' be the graph, obtained from G by adding to it the edge $\{a, b\}$.

  Then:

  - G' has no degree-2 edges (since its minimum degree is at least 3).
  - G' has no simplicial vertices. (If a was simplicial in G', then there would be an edge in G connecting a_1 to a_2, which is impossible. Similarly, b cannot be simplicial. Suppose that a vertex $x \in K_1$ is simplicial. Then, since x is of degree at least 3, there is a vertex y in K_1 that is adjacent to x. But now, (a, x, b), (a, y, b) and an a-b path through K_2 are three independent $\{a, b\}$ paths in G, which is impossible.)
  - G' has no thetas. (Otherwise, let T be a $\{u, v\}$-theta in G'. Clearly, G' cannot contain three independent paths linking a vertex of K_1 to a vertex of K_2 (since $\{a, b\}$ is a cutset in G'). Up to symmetry, this implies that $\{u, v\} \cap K_2 = \emptyset$. Since T is not a theta in G, it must use the edge $\{a, b\}$, and therefore we conclude that $V(T) \cap K_2 = \emptyset$. However, replacing the edge $\{a, b\}$ with an a-b path in G through K_2 results in a $\{u, v\}$-theta in G, a contradiction.)

  So, G' is a bad graph on n vertices with more edges than G, contradicting the choice of G.

- G is chordal.

  Suppose not, and let C be an induced cycle in G of at least four vertices. Consider a vertex x on C and its two neighbors a and b on the cycle. Then, by the previous observation, the set $\{a, b\}$ cannot separate x from the remainder of the cycle. Let P be a shortest path from x to $V(C)\setminus\{x, a, b\}$, and let q be the vertex that belongs to both P and C. Then, using the two x-q paths on C and the path P, we can construct in G a $\{x, q\}$-theta, contradicting the assumption that G is a $(*, 2)$-bounded compatibility graph.

Since G is chordal, it must contain a simplicial vertex. This is however a contradiction to the fact that G is bad. Thus, there are no bad graphs, and the proof is complete. □

3 A polynomial reduction for $(*, 2)$-bounded parsimony haplotyping

Lemma 4. *Let $\mathbf{G}$ be a genotype matrix with at most two 2s per column, and let C be a clique of at least three vertices in the compatibility graph of $\mathbf{G}$. Then, there is a unique haplotype h that is consistent with every genotype in C.*

Proof. Let $\mathbf{C}$ be the submatrix of $\mathbf{G}$ defined by the rows in C. Consider the j-th column $\mathbf{C}(j)$ of $\mathbf{C}$. Since the genotypes in C form a clique, all the entries of $\mathbf{C}(j)$ different from 2 must be the same. Furthermore, since $\mathbf{G}$ has at most two 2s per column, the same is true for $\mathbf{C}$. It follows that for every column $\mathbf{C}(j)$ there is a unique element $h(j) \in \{0, 1\}$ that appears in that column (besides the possible 2s). This defines the unique haplotype consistent with all genotypes in C. □

Lemma 5. *Let $\mathbf{G}$ be a genotype matrix with at most two 2s per column. Let $g \in \mathbf{G}$ be an input genotype, and let $g_1, g_2 \in \mathbf{G}$ be two distinct input genotypes different from g that are compatible with g. Then, there is at most one pair $\{h_1, h_2\}$ of haplotypes that resolve g such that h_i is consistent with g_i for $i = 1, 2$.*

Proof. By means of contradiction, suppose that g can be resolved by h_1 and h_2, as well as by h'_1 and h'_2, where h_1, h'_1 are consistent with g_1, and h_2, h'_2 are consistent with g_2. Note that $h_1 \neq h'_1$ (since otherwise $h_2 = h'_2$ and we are done). Thus, there is a column j where h_1 and h'_1 differ. Without loss of generality, we may assume that $h_1(j) = 0$, $h'_1(j) = 1$. Since h_1 and h'_1 are consistent with g_1, we conclude that $g_1(j) = 2$.

Next, we observe that h_1 and h'_1 are both consistent with g (since each of them can be used in a resolution of g). This implies that $g(j) = 2$, which in turn implies $h_2(j) = 1$, $h'_2(j) = 0$. Therefore, since h_2 and h'_2 are consistent with g_2, we get $g_2(j) = 2$. However, we now have $g_1(j) = g_2(j) = g(j) = 2$ which is a contradiction to the assumption that $\mathbf{G}$ is a $(*, 2)$-bounded instance. □

Lemma 6. *Let G be the compatibility graph of a $(*, 2)$-bounded instance $\mathbf{G} \in \{0, 1, 2\}^{m \times n}$ with $|V(G)| = m$. Then, G has $O(m)$ maximal cliques.*

Proof. This is a direct consequence of Lemma 3, since every simplicial vertex is contained in only one maximal clique, and every endpoint of a degree-2 edge is contained in precisely two maximal cliques. By induction, it follows that G has at most $2m$ maximal cliques. □

Lemma 7. *Given a genotype matrix $\mathbf{G} \in \{0, 1\}^{m \times n}$ with at most two 2s per column, we can compute in polynomial time a set $\mathbf{H}'$ of $O(m^3)$ haplotypes such that there is an optimal solution $\mathbf{H}$ to the parsimony haplotyping problem on $\mathbf{G}$ contained in $\mathbf{H}'$.*

Proof. We give a constructive proof. We start with an empty collection $\mathbf{H}'$ of haplotypes, and perform four steps of additions of haplotypes to $\mathbf{H}'$ as follows.

First, we add to $\mathbf{H}'$ all the input genotypes that are haplotypes.

Second, we construct the compatibility graph G and compute all the clique haplotypes. This can be done in polynomial time: by Lemma 6, G has $O(m)$ maximal cliques, and they can be generated in polynomial time e.g. by the algorithm of Tsukiyama et al. [4].

We add to $\mathbf{H}'$ all the clique haplotypes h_c and for every such haplotype h_c and every input genotype g containing h_c, we add to $\mathbf{H}'$ the complement of h_c w.r.t. g.

Third, for every input genotype $g \in \mathbf{G}$, we denote by $P(g)$ the set of all pairs $\{g_1, g_2\}$ of input genotypes such that there exists a pair $\{h_1, h_2\}$ of haplotypes that resolve g such that h_i is consistent with g_i for $i = 1, 2$. By Lemma 5, $\{g_1, g_2\} \in P(g)$ implies that there exists a unique pair $p(g, g_1, g_2) = \{h_1, h_2\}$ of haplotypes that resolve g such that h_i is consistent with g_i for $i = 1, 2$. Let $H(g) = \{h \text{ : there exists } \{g_1, g_2\} \in P(g) \text{ such that } h \in p(g, g_1, g_2)\}$. Moreover, let $H = \cup_{g \in \mathbf{G}} H(g)$. The set H can be computed in polynomial time, and $|H| = O(m^3)$.

We add to $\mathbf{H}'$ all the haplotypes in H and also, for every such haplotype $h \in H$ and every input genotype g containing h, we add to $\mathbf{H}'$ the complement of h w.r.t. g.

Next, for each edge $e = \{g_1, g_2\}$ of the compatibility graph, fix one haplotype $h =: h(g_1, g_2)$ consistent with g_1 and g_2 and set $H(e) = \{h(g_1, g_2), h_1, h_2\}$ where h_1 and h_2 are the complements of h w.r.t. g_1 and g_2.

For each $e \in E(G)$, add to $\mathbf{H}'$ the haplotypes in $H(e)$.

Finally, for every input genotype g not resolved with the haplotypes added to $\mathbf{H}'$ so far, add to $\mathbf{H}'$ a pair of haplotypes resolving g.

This completes the description of the construction of the set $\mathbf{H}'$. This set satisfies the following property.

Property 1. For every input genotype g and every haplotype h consistent with it, if $h \in \mathbf{H}'$ then $h' \in \mathbf{H}'$ where h' is the haplotype that together with h resolves g.

It remains to show that there is an optimal solution to the parsimony haplotyping problem on $\mathbf{G}$ contained in $\mathbf{H}'$. To see this, let $\mathbf{H}_{opt}$ be an optimal solution to the parsimony haplotyping problem on $\mathbf{G}$ such that $|\mathbf{H}_{opt} \cap \mathbf{H}'|$ is maximized. Suppose that there is a haplotype h contained in $\mathbf{H}_{opt}$ but not in $\mathbf{H}'$.

We consider three exhaustive cases.

Case 1. h is consistent with three or more input genotypes.

Then, the set of genotypes h consistent with forms a clique of size at least 3 in the compatibility graph. Thus, h is a clique haplotype and belongs to $\mathbf{H}'$ by construction.

Case 2. h is consistent with exactly two input genotypes.

Let g_1 and g_2 be the two input genotypes consistent with h. Let h_1 and h_2 denote the complements of h w.r.t. g_1 and g_2. By Property 1, neither h_1 nor h_2 belong to $\mathbf{H}'$. Therefore, by construction of $\mathbf{H}'$ and Lemma 5, neither of h_1 and h_2 is consistent with an input genotype. (For example, if h_1 was consistent with an input genotype $g' \neq g_1$, then $\{h, h_1\}$ would belong to $P(g_1)$ and thus to H_0.)

So we can replace the three haplotypes $\{h, h_1, h_2\}$ with the three haplotypes $\{h(g_1, g_2), h', h''\}$ where

h' and h'' denote the complements of $h(g_1, g_2)$ w.r.t. g_1 and g_2. This gives a set of haplotypes $\mathbf{H}''$ resolving $\mathbf{G}$ such that $|\mathbf{H}_{opt} \cap \mathbf{H}''| > |\mathbf{H}_{opt} \cap \mathbf{H}'|$, which contradicts the choice of $\mathbf{H}_{opt}$.

Case 3. h is consistent with only one input genotype.

Let $g \in \mathbf{G}$ be the input genotype consistent with h. Let $h' \in \mathbf{H}_{opt}$ be the complement of h w.r.t. g. By Property 1, h' does not belong to $\mathbf{H}'$. Since Cases 1 and 2 are impossible, g is the only input genotype consistent with h'. So we can replace the two haplotypes $\{h, h'\}$ with any two haplotypes from $\mathcal{H}'$ that resolve g. This gives a set of haplotypes $\mathbf{H}''$ resolving $\mathbf{G}$ such that $|\mathbf{H}_{opt} \cap \mathbf{H}''| > |\mathbf{H}_{opt} \cap \mathbf{H}'|$, contrary to the choice of $\mathbf{H}_{opt}$.

This completes the proof. $\square$

3.1 Parsimony haplotyping is polynomial on $(*, 2)$-bounded instances of bounded tree-width

We recall the following theorem from [3].

Theorem 1 ([3]). *There is a polynomial algorithm for parsimony haplotyping on enumerable instances such that the compatibility graph has bounded tree-width.*

In [3], the authors define an *enumerable instance* as an input genotype matrix with a polynomial number of haplotypes that are consistent with any of its genotypes. However, the proof of Theorem 1 remains valid if we relax this condition and only require that, given an input genotype matrix, we can compute in polynomial time a polynomially-sized set of haplotypes that contains an optimal solution. Parsimony haplotyping remains solvable in polynomial time for such instances whenever the compatibility graph has bounded tree-width. Together with Lemma 7, this implies the following result.

Theorem 2. *$PH(*, 2)$ is polynomially solvable on graphs of bounded tree-width.*

4 Other results

Lemma 8 (Inference paths.). *Let $\mathbf{G}$ be a genotype matrix with at most two 2s per column, and let g, g' and g'' be three distinct genotypes from $\mathbf{G}$ such that there exist haplotypes h, h' and h'' such that $g = h \oplus h'$, $g' = h' \oplus h''$, and h'' is consistent with g''. Then, the haplotype h' is uniquely determined by g, g', g''. Moreover, the three haplotypes h, h', h'' can be computed in time $O(n)$ where n is the number of columns of $\mathbf{G}$.*

Proof. Since h' is consistent with both g and g', we have $h'(j) = a$ whenever there is an $a \in \{0, 1\}$ such that $g(j) = a$ or $g'(j) = a$. If $g(j) = g'(j) = 2$, then, since $\mathbf{G}$ has at most two 2s per column, we have $g''(j) = a \in \{0, 1\}$. Then, since h'' is consistent with g'', we must have $h''(j) = a$, which in turn implies $h'(j) = 1 - a$ (since $g'(j) = 2$). Therefore, h' is uniquely determined by g, g', g''. To see that h, h', h'' can be computed in linear time, observe that the haplotypes h and h'' are uniquely determined by (h', g) and (h', g') respectively. $\square$

(Lemma 8 also follows directly from Lemma 5.)

5 Parsimony haplotyping on trees

In this section, we will show that one can solve the parsimony haplotyping problem on trees by dynamic programming. Under certain conditions, this approach leads to polynomial-time solutions.

Let $\mathbf{G} \in \{0,1,2\}^{m \times n}$ be an instance of parsimony haplotyping whose compatibility graph T is a tree. To avoid trivialities, we assume $m \geq 3$. We root the tree T at an arbitrary node r with at least two neighbors. We shall develop a dynamic programming solution to the problem, which will consist in a single bottom-up traversal of the tree.

We first introduce several definitions needed to describe the algorithm. For a node g of T, let $\mathbf{G}(g)$ denote the instance of parsimony haplotyping consisting of the genotypes in $\mathbf{G}$ that belong to the subtree of T rooted at g. Furthermore, let $opt(g)$ denote the optimal value of $\mathbf{G}(g)$, that is, the minimum number of haplotypes needed to resolve all the genotypes in $\mathbf{G}(g)$. For a node g of T different from the root, let $C(g)$ denote the set of all haplotypes consistent both with g and with its parent $P(g)$, and let $D(g)$ denote the set of all haplotypes in $C(g)$ that appear in some optimal solution to the parsimony haplotyping problem on $\mathbf{G}(g)$.

Given a set W of haplotypes that are consistent with a genotype g, we denote by $\overline{W}^{(g)}$ the set of their complements with respect to g, that is, $\overline{W}^{(g)} = \{h : \exists h' \in W \text{ such that } h \oplus h' = g\}$.

The algorithm is based on the following recursive formulas for $opt(g)$ and $D(g)$.

Lemma 9. *If g is a leaf of T, then:*

$$opt(g) = \begin{cases} 1, & \textit{if } g \in \{0,1\}^n; \\ 2, & \textit{otherwise.} \end{cases} \tag{1}$$

$$D(g) = C(g)\,. \tag{2}$$

Let g be an internal node of T, with children $\{g_1, \ldots, g_k\}$ (where $k \geq 1$). Let $W(g) = \cup_{i=1}^{k} D(g_i)$. Then:

$$opt(g) = \sum_{i=1}^{k} opt(g_i) + \begin{cases} 0, & \textit{if } \exists h, h' \in W(g) \textit{ such that } g = h \oplus h'; \\ 2, & \textit{if } W(g) = \emptyset; \\ 1, & \textit{otherwise.} \end{cases} \tag{3}$$

$$D(g) = \begin{cases} \emptyset, & \textit{if } \exists h, h' \in W(g) \textit{ such that } g = h \oplus h'; \\ C(g), & \textit{if } W(g) = \emptyset; \\ C(g) \cap \overline{W(g)}^{(g)}, & \textit{otherwise.} \end{cases} \tag{4}$$

Proof. The equations (1) and (2) are straightforward.

Now, let g be an internal node of T, with children $\{g_1, \ldots, g_k\}$, and let $W(g)$ be defined as in the lemma. We split the remainder of the proof into three cases, as in the above recursions.

Case 1: $\exists h, h' \in W(g)$ such that $g = h \oplus h'$.

First, we show that $opt(g) = \sum_{i=1}^{k} opt(g_i)$. Clearly, $opt(g) \geq \sum_{i=1}^{k} opt(g_i)$, since every set of haplotypes optimally resolving $\mathbf{G}(g)$ must contain disjoint solutions to $\mathbf{G}(g_1), \ldots, \mathbf{G}(g_k)$.

For the converse inequality, let $h, h' \in W(g)$ such that $g = h \oplus h'$. Let $i_1, i_2 \in [k]$ such that $h \in D(g_{i_1})$, $h' \in D(g_{i_2})$. Suppose that $i_1 = i_2$. Then, every haplotype that is consistent with g is also consistent with g_{i_1}. In particular, since the children of g are pairwise inconsistent, this implies $k = 1$. Moreover, g must be the root since otherwise the parent of g could not be inconsistent with g_{i_1}. However, this contradicts our choice of the root as a vertex with at least two neighbors.

Thus, $i_1 \neq i_2$, and combining two sets of haplotypes that optimally resolve $\mathbf{G}(g_{i_1})$ and $\mathbf{G}(g_{i_2})$ and contain h and h' respectively with arbitrary optimal solutions to $\mathbf{G}(g_i)$ for all other g_i's yields a solution to $\mathbf{G}(g)$ of cardinality $\sum_{i=1}^{k} opt(g_i)$.

Next, we show that $D(g) = \emptyset$. The equality $opt(g) = \sum_{i=1}^{k} opt(g_i)$ implies that every set of haplotypes optimally resolving $\mathbf{G}(g)$ must consist of disjoint solutions to $\mathbf{G}(g_1), \ldots, \mathbf{G}(g_k)$. However, every $h \in C(g)$ is consistent with the parent of g, and therefore inconsistent with all genotypes from $\mathbf{G}(g_1), \ldots, \mathbf{G}(g_k)$. Therefore, h cannot belong to any optimal solution to the parsimony haplotyping problem on $\mathbf{G}(g)$. Consequently, $D(g) = \emptyset$.

Case 2: $W(g) = \emptyset$.

First, we show that $opt(g) = \sum_{i=1}^{k} opt(g_i) + 2$.

Suppose that $opt(g) \leq \sum_{i=1}^{k} opt(g_i) + 1$, and let H be a set of haplotypes resolving $\mathbf{G}(g)$ such that $|H| \leq \sum_{i=1}^{k} opt(g_i) + 1$. For $i = 1, \ldots, k$, let H_i be a minimal subset of H that resolves $\mathbf{G}(g_i)$.

Suppose that $H = \cup_{i=1}^{k}(H_i)$. Then, there must be two haplotypes h, h' in $\cup_{i=1}^{k}(H_i)$ that resolve g. Let $i_1, i_2 \in [k]$ such that $h \in H_{i_1}$, $h' \in H_{i_2}$. Then, as in Case 1, we conclude that $i_1 \neq i_2$. Moreover, since h is consistent with both g and g_{i_1}, h belongs to $C(g_{i_1})$. Similarly, h' belongs to $C(g_{i_2})$. Since $W(g) = \emptyset$, we conclude that h cannot belong to $D(g_{i_1})$ and h' cannot belong to $D(g_{i_2})$. In particular, this implies that $|H_{i_1}| \geq opt(g_{i_1}) + 1$ and $|H_{i_2}| \geq opt(g_{i_2}) + 1$. Therefore, $|H| \geq \sum_{i=1^k} |H_i| \geq \sum_{i=1^k} opt(g_i) + 2$, contradicting the assumption.

Therefore, there is a haplotype h such that $H \backslash \left(\cup_{i=1}^{k}(H_i)\right) = \{h\}$. Since the cardinality of H is at most $\sum_{i=1}^{k} opt(g_i) + 1$ and at least $\sum_{i=1}^{k} |H_i| + 1$, we conclude that each of the H_i's is optimal for $\mathbf{G}(g_i)$. Moreover, the fact that H is optimal for $\mathbf{G}(g)$ and the observation that h cannot be used in the resolution of any genotype from $\mathbf{G}(g)$ other than g imply that h must be used in the resolution of g. Let $h' \in H$ be the haplotype that together with h resolves g, and let $j \in [k]$ such that $h' \in H_j$. Since H_j is optimal for $\mathbf{G}(g_j)$ and since $h' \in C(g_j)$, we conclude that $h' \in D(g_j)$. However, this contradicts the assumption of the case.

This shows that $opt(g) \geq \sum_{i=1}^{k} opt(g_i) + 2$. The converse inequality is considerably simpler. Combining sets $H_1, \ldots H_k$ of haplotypes that optimally resolve $\mathbf{G}(g_1), \ldots, \mathbf{G}(g_k)$ together with two additional haplotypes h, h' that resolve g yields a solution to $\mathbf{G}(g)$ of cardinality $\sum_{i=1}^{k} opt(g_i) + 2$. This observation also implies that $D(g) = C(g)$, as any $h \in C(g)$ can be used in a resolution of g.

Case 3: Neither Case 1 nor Case 2.

First, we show that $opt(g) = \sum_{i=1}^{k} opt(g_i) + 1$.

Suppose that $opt(g) \leq \sum_{i=1}^{k} opt(g_i)$, and let H be a set of haplotypes resolving $\mathbf{G}(g)$ such that $|H| \leq \sum_{i=1}^{k} opt(g_i)$. Then, for any $i = 1, \ldots, k$, the subset H_i of H that resolves $\mathbf{G}(g_i)$ must be optimal for $\mathbf{G}(g_i)$. Therefore, g must be resolved by a pair h, h' of haplotypes belonging to $\cup_{i=1}^{k} H_i$. By the optimality of the H_i's, we have $h, h' \in W(g)$, contradicting the fact that we are not in Case 1.

Therefore, $opt(g) \geq \sum_{i=1}^{k} opt(g_i) + 1$. For the converse inequality, let h be an arbitrary haplotype in $W(g)$, and let $j \in [k]$ such that $h \in D(g_j)$. Combining a set of haplotypes containing h that optimally resolves $\mathbf{G}(g_j)$ with arbitrary optimal solutions to $\mathbf{G}(g_i)$ for all other g_i's, together with the haplotype h' such that $h \oplus h' = g$ yields a solution to $\mathbf{G}(g)$ of cardinality $\sum_{i=1}^{k} opt(g_i) + 1$. This shows that $opt(g) = \sum_{i=1}^{k} opt(g_i) + 1$.

We now show $D(g) = C(g) \cap \overline{W(g)}^{(g)}$ by proving both containments.

First, let $h \in D(g)$. Then, $h \in C(g)$, and there is an optimal solution H to $\mathbf{G}(g)$ that contains h. We only need to show that $h \in \overline{W(g)}^{(g)}$, that is, that the haplotype h' satisfying $h \oplus h' = g$ belongs to $W(g)$. Since h belongs to $C(g)$, it cannot be used in the resolution of any genotype from $\mathbf{G}(g)$ other than g; thus, h must be used in the resolution of g, therefore h' belongs to H. For $i \in [k]$, let H_i be a minimal subset of H that resolves $\mathbf{G}(g_i)$. Since $opt(g) = \sum_{i=1}^{k} opt(g_i) + 1$, each of the H_i's is optimal for $\mathbf{G}(g_i)$. Let $j \in [k]$ be the index of the set H_j that contains h'. Since H_j is optimal for $\mathbf{G}(g_j)$, we conclude that h' belongs to $D(g_j)$. Thus, h' belongs to $W(g)$ and this shows that $D(g) \subseteq C(g) \cap \overline{W(g)}^{(g)}$.

To see the converse, let $h \in C(g) \cap \overline{W(g)}^{(g)}$. Therefore, the haplotype h' such that $h \oplus h' = g$ belongs to $W(g)$. Let $j \in [k]$ be the index of the set $D(g_j)$ that contains h'. Combining a set of haplotypes containing h' that optimally resolves $\mathbf{G}(g_j)$ with arbitrary optimal solutions to $\mathbf{G}(g_i)$ for all other g_i's, together with the haplotype h' yields an optimal solution to $\mathbf{G}(g)$. Therefore, h belongs to $D(g)$. □

At each internal node g, the algorithm stores an integer $opt(g) \in \{1, \ldots, m\}$, and a (possibly empty) matrix $\mathbf{D}(g) \in \{0, 1, 2\}^{p(g) \times n}$ whose $p(g)$ rows represent the genotypes in $D(g)$. The algorithm traverses the tree bottom-up, using the recursive relations $(1) - (4)$ from Lemma 9 to compute the values of $opt(g)$ and $D(g)$.

The minimum number of haplotypes needed to resolve all the genotypes in $\mathbf{G}(g)$ is given by $opt(r)$, where r is the root of T. Also, an optimal set of haplotypes can be constructed by a simple backtracking procedure, traversing the tree top-down. We omit the details.

Details of the implementation. Analysis of the running time.

The following observation is easily proved by induction on the height of g.

Claim 1. *Each $D(g)$ can be represented by a (possibly empty) set of pairwise incompatible genotypes.*

Claim 2. *Given genotypes $g_1, g_2, g \in \{0, 1, 2\}^n$ such that every haplotype consistent with g_1 or g_2 is consistent with g, we can determine in time $O(n)$ whether there exist haplotypes h and h', each consistent with either g_1 or g_2, such that $g = h \oplus h'$.*

Proof. One only needs to verify that there is no column j such that $g(j) = 2$ and $g_1(j) = g_2(j) \in \{0, 1\}$. □

At each leaf of the tree, the algorithms spends $O(n)$ time.

Consider an internal node g of the tree, with children $\{g_1, \ldots, g_k\}$. For each $i \in [k]$, let the matrix $\mathbf{D}(g_i)$ be of dimensions $p_i \times n$.

Suppose that all p_i's are zero. Then $W(g) = \emptyset$, and the algorithm sets $opt(g) = \sum_{i=1}^{k} opt(g_i) + 2$ and (unless g is the root), sets $\mathbf{D}(g)$ to be the $1 \times n$ matrix given by

$$\mathbf{D}(g)_{1,j} = \begin{cases} 0, & \text{if at least one of } g(j), g'(j) \text{ is 0;} \\ 1, & \text{if at least one of } g(j), g'(j) \text{ is 1;} \\ 2, & \text{otherwise.} \end{cases}$$

for all $j \in [n]$, where g' is the parent of g.

Suppose that not all the p_i's are zero. Then $W(g) \neq \emptyset$. Let $\mathbf{M}(g)$ be the matrix of dimensions $p \times n$ (where $p = \sum_{i=1}^{k} p_i$) whose set of rows is the union of the sets of rows of the matrices $\mathbf{D}(g_1) \ldots, \mathbf{D}(g_k)$. For each of the $\binom{p}{2}$ pairs $\{g_1, g_2\}$ of rows of $\mathbf{M}(g)$, the algorithm verifies whether there exist haplotypes h and h', each consistent with either g_1 or g_2, such that $g = h \oplus h'$. Using Claim 2, this can be verified in time $O(mp^2)$.

If there exists such a pair of haplotypes h and h', the algorithm sets $opt(g) = \sum_{i=1}^{k} opt(g_i)$ and (unless g is the root) sets $\mathbf{D}(g)$ to be the empty matrix.

Otherwise, the algorithm sets $opt(g) = \sum_{i=1}^{k} opt(g_i) + 1$ and (unless g is the root) computes $\mathbf{D}(g)$ as follows. $\mathbf{D}(g)$ is initialized to be the empty matrix. For each row g' of $\mathbf{M}(g)$, let g'' be the complement of g' with respect to g, that is,

$$g''(j) = \begin{cases} 0, & \text{if either } g(j) = 0\text{, or } g(j) = 2 \text{ and } g'(j) = 1; \\ 1, & \text{if either } g(j) = 1\text{, or } g(j) = 2 \text{ and } g'(j) = 0; \\ 2, & \text{otherwise.} \end{cases}$$

We have to verify whether g'' is compatible with $C(g)$, and if yes, to compute the set of haplotypes consistent with both. If there is a column j such that $C(g)(j) = 0$ and $g''(j) = 1$ or vice-versa, then g'' is incompatible with $C(g)$, and we do nothing. Otherwise, we define a new row $\tilde{g}$ by

$$\tilde{g}(j) = \begin{cases} 0, & \text{if at least one of } C(g)(j), g''(j) \text{ is 0;} \\ 1, & \text{if at least one of } C(g)(j), g''(j) \text{ is 1;} \\ 2, & \text{otherwise.} \end{cases} \tag{5}$$

We add $\tilde{g}$ to $\mathbf{D}(g)$.

Overall, the amount of time the algorithm spends at node g is at most $O(n \cdot p(g)^2)$. Therefore, the total time complexity of the algorithm is $O(n \sum_{g \in T} (p(g))^2)$.

It is easy to see that if g is an internal node at height h, then $p(g) \leq \Delta^h$, where Δ is the maximum degree of a node in T. This shows that the time complexity of the algorithm is at most $O(n\Delta^{2h})$, where h is the height of the rooted tree T. This could result in an exponential worst-case time and space complexity.

5.1 Parsimony haplotyping is polynomial for trees of small diameter

The upper bound $O(n\Delta^{2h})$ on the running time of the algorithm shows that the algorithm runs in polynomial time for trees of diameter at most $O\left(\frac{\log m}{\log \Delta}\right)$. Also, note that for every node g with children $\{g_1, \ldots, g_k\}$ such that all but one of the $p(g_i)$'s are zero, we have $p(g) \leq \max_{1 \leq i \leq k} p(g_i)$. In particular, this implies that the possibly exponential growth of p cannot be due to the nodes with only one children. Thus, we have the following conclusion.

Theorem 3. *Parsimony haplotyping is solvable in polynomial time on trees of diameter $O\left(\frac{\log m}{\log \Delta}\right)$ (where m and Δ denote the number of nodes and maximum vertex degree respectively), as well as on subdivisions of such trees.*

Corollary 2. *Parsimony haplotyping is solvable in time $O(mn)$ on paths.*

5.2 Parsimony haplotyping on $(*, 2)$-bounded tree instances: an alternative proof

In view of Theorem 2, parsimony haplotyping is polynomial on $(*, 2)$-bounded tree instances. In this subsection we provide an alternative and direct proof of this fact, based on Lemma 9. In particular, we will show that the recursive formulas from Lemma 9 lead to a polynomial time algorithm on arbitrary trees that arise from $(*, 2)$-bounded instances.

Let $\mathbf{G}$ be a $(*, 2)$-bounded instance of parsimony haplotyping, whose compatibility graph T is a tree. It is sufficient to show that for all internal nodes g different from the root, the values of $p(g)$ are bounded above by a polynomial in m. In fact, we will show that for every internal node g different from the root, $p(g)$ is bounded above by the number of children of g.

We start with an auxiliary lemma.

Lemma 10. *Let g be an internal node of T different from the root that is of "Type 3" (that is, $W(g) \neq \emptyset$ and $D(g) = C(g) \cap \overline{W(g)}^{(g)}$). Then, each row of $\mathbf{D}(g)$ is a haplotype.*

Proof. Let $\tilde{g}$ be a row of $\mathbf{D}(g)$, and suppose that there is a column j such that $\tilde{g}(j) = 2$. Then, $\tilde{g}$ is obtained via (5) from $C(g)$ and g'', the complement of g' with respect to g, where g' is a genotype from (say) $D(g_{i_1})$ for a child g_{i_1} of g.

Since $\tilde{g}(j) = 2$, we have by (5) that $C(g)(j) = g''(j) = 2$. Denoting by $P(g)$ the parent of g, this implies that $g(j) = (P(g))(j) = g_{i_1}(j) = 2$. This is a contradiction to the assumption that $\mathbf{G}$ is a $(*, 2)$-bounded instance. □

Lemma 11. *Let g be an internal node of T, different from the root, with children $\{g_1, \ldots, g_k\}$. Then, $p(g) \leq k$.*

Proof. Let us partition the set of children of g into two sets, as follows: $C_1 := \{g_i : 1 \leq i \leq k, p(g_i) \leq 1\}$ and $C_2 := \{g_i : 1 \leq i \leq k, p(g_i) > 1\}$.

Consider an arbitrary $g_i \in C_2$. Clearly, g_i is of "Type 3" (that is, $W(g) \neq \emptyset$ and $D(g) = C(g) \cap \overline{W(g)}^{(g)}$), for otherwise $p(g_i)$ would be at most 1. Therefore, by Lemma 10, each row of $\mathbf{D}(g_i)$ is a haplotype.

We now show that there is at most one row h of $\mathbf{D}(g_i)$ such that the complement of h with respect to g is consistent with $C(g)$. Suppose not, and let h_1, h_2 be two rows of $\mathbf{D}(g_i)$ such that both h_1' and h_2' (where $h_1 \oplus h_1' = h_2 \oplus h_2' = g$) are consistent with $C(g)$. Let j be a column such that $h_1(j) \neq h_2(j)$. Then, since both h_1 and h_2 are consistent with g, we have $g(j) = 2$, and similarly, $g_i(j) = 2$. Next, $h_1(j) \neq h_2(j)$ implies that $h_1'(j) \neq h_2'(j)$. Let $P(g)$ denote the parent of g. Since by assumption both h_1' and h_2 are consistent with $C(g)$ (and hence with $P(g)$, we have $(P(g))(j) = 2$. However, this is impossible since $\mathbf{G}$ is a $(*, 2)$-bounded instance.

It follows that each $g_i \in C_2$ contributes at most one row to $D(g)$. Therefore, $p(g) \leq \sum_{g_i \in C_1} p(g_i) + \sum_{g_i \in C_2} 1 \leq |C_1| + |C_2| = k$. $\square$

Theorem 4. *$PH(*, 2)$ is polynomially solvable on trees.*

References

[1] R. Diestel. *Graph Theory. Third Edition.* Springer-Verlag, Heidelberg, 2005.

[2] N. Robertson and P.D. Seymour. Graph minors. V. excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.

[3] R. Sharan, B.V. Halldórsson, and S. Istrail. Islands of tractability for parsimony haplotyping. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(3):303–311, 2006.

[4] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.*, 6:505–517, 1977.

97-04	Rose: Generating Sequence Families (Jens Stoye, Dirk Evers, Folker Meyer)
97-05	Fuzzy Quantifiers for Processing Natural Language Queries in Content-Based Multimedia Retrieval Systems (Ingo Glöckner, Alois Knoll)
97-06	DFS – An Axiomatic Approach to Fuzzy Quantification (Ingo Glöckner)
98-01	Kognitive Aspekte bei der Realisierung eines robusten Robotersystems für Konstruktionsaufgaben (Alois Knoll, Bernd Hildebrandt)
98-02	A Declarative Approach to the Development of Dynamic Programming Algorithms, applied to RNA Folding (Robert Giegerich)
98-03	Reducing the Space Requirement of Suffix Trees (Stefan Kurtz)
99-01	Entscheidungskalküle (Axel Saalbach, Christian Lange, Sascha Wendt, Mathias Katzer, Guillaume Dubois, Michael Höhl, Oliver Kuhn, Sven Wachsmuth, Gerhard Sagerer)
99-02	Transforming Conditional Rewrite Systems with Extra Variables into Unconditional Systems (Enno Ohlebusch)
99-03	A Framework for Evaluating Approaches to Fuzzy Quantification (Ingo Glöckner)
99-04	Towards Evaluation of Docking Hypotheses using elastic Matching (Steffen Neumann, Stefan Posch, Gerhard Sagerer)
99-05	A Systematic Approach to Dynamic Programming in Bioinformatics. Part 1 and 2: Sequence Comparison and RNA Folding (Robert Giegerich)
99-06	Autonomie für situierte Robotersysteme – Stand und Entwicklungslinien (Alois Knoll)
2000-01	Advances in DFS Theory (Ingo Glöckner)
2000-02	A Broad Class of DFS Models (Ingo Glöckner)

2000-03	An Axiomatic Theory of Fuzzy Quantifiers in Natural Languages (Ingo Glöckner)
2000-04	Affix Trees (Jens Stoye)
2000-05	Computergestützte Auswertung von Spektren organischer Verbindungen (Annika Büscher, Michaela Hohenner, Sascha Wendt, Markus Wiesecke, Frank Zöllner, Arne Wegener, Frank Bettenworth, Thorsten Twellmann, Jan Kleinlützum, Mathias Katzer, Sven Wachsmuth, Gerhard Sagerer)
2000-06	The Syntax and Semantics of a Language for Describing Complex Patterns in Biological Sequences (Dirk Strothmann, Stefan Kurtz, Stefan Gräf, Gerhard Steger)
2000-07	Systematic Dynamic Programming in Bioinformatics (ISMB 2000 Tutorial Notes) (Dirk J. Evers, Robert Giegerich)
2000-08	Difficulties when Aligning Structure Based RNAs with the Standard Edit Distance Method (Christian Büschking)
2001-01	Standard Models of Fuzzy Quantification (Ingo Glöckner)
2001-02	Causal System Analysis (Peter B. Ladkin)
2001-03	A Rotamer Library for Protein-Protein Docking Using Energy Calculations and Statistics (Kerstin Koch, Frank Zöllner, Gerhard Sagerer)
2001-04	Eine asynchrone Implementierung eines Microprozessors auf einem FPGA (Marco Balke, Thomas Dettbarn, Robert Homann, Sebastian Jaenicke, Tim Köhler, Henning Mersch, Holger Weiss)
2001-05	Hierarchical Termination Revisited (Enno Ohlebusch)
2002-01	Persistent Objects with O2DBI (Jörn Clausen)
2002-02	Simulation von Phasenübergängen in Proteinmonoschichten (Johanna Alichniewicz, Gabriele Holzschneider, Morris Michael, Ulf Schiller, Jan Stallkamp)
2002-03	Lecture Notes on Algebraic Dynamic Programming 2002 (Robert Giegerich)

2002-04	Side chain flexibility for 1:n protein-protein docking (Kerstin Koch, Steffen Neumann, Frank Zöllner, Gerhard Sagerer)
2002-05	ElMaR: A Protein Docking System using Flexibility Information (Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)
2002-06	Calculating Residue Flexibility Information from Statistics and Energy based Prediction (Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)
2002-07	Fundamentals of Fuzzy Quantification: Plausible Models, Constructive Principles, and Efficient Implementation (Ingo Glöckner)
2002-08	Branching of Fuzzy Quantifiers and Multiple Variable Binding: An Extension of DFS Theory (Ingo Glöckner)
2003-01	On the Similarity of Sets of Permutations and its Applications to Genome Comparison (Anne Bergeron, Jens Stoye)
2003-02	SNP and mutation discovery using base-specific cleavage and MALDI-TOF mass spectrometry (Sebastian Böcker)
2003-03	From RNA Folding to Thermodynamic Matching, including Pseudoknots (Robert Giegerich, Jens Reeder)
2003-04	Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt (Sebastian Böcker)
2003-05	Systematic Investigation of Jumping Alignments (Constantin Bannert)
2003-06	Suffix Tree Construction and Storage with Limited Main Memory (Klaus-Bernd Schürmann, Jens Stoye)
2003-07	Sequencing from compomers in thepresence of false negative peaks (Sebastian Böcker)
2003-08	Genalyzer: An Interactive Visualisation Tool for Large-Scale Sequence Matching – Biological Applications and User Manual (Jomuna V. Choudhuri, Chris Schleiermacher)

2004-01 Sequencing From Compomers is NP-hard
(Sebastian Böcker)

2004-02 The Money Changing Problem revisited: Computing the Frobenius number in time
$O(k\,a_1)$
(Sebastian Böcker, Zsuzsanna Lipták)

2004-03 Accelerating the Evaluation of Profile HMMs by Pruning Techniques
(Thomas Plötz, Gernot A. Fink)

2004-04 Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection
(Ferdinando Cicalese, Peter Damaschke, Ugo Vaccaro)

2004-05 Compressed Representation of Sequences and Full-Text Indexes
(Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, Gonzalo Navarro)

2005-01 Overlaps Help: Improved Bounds for Group Testing with Interval Queries
(Ferdinando Cicalese, Peter Damaschke, Libertad Tansini, Sören Werth)

2005-02 Two batch Fault-tolerant search with error cost constraints: An application to learning
(Ferdinando Cicalese)

2005-03 Searching for the Shortest Common Supersequence
(Sergio A. de Carvalho Jr., Sven Rahmann)

2005-04 Counting Suffix Arrays and Strings
(Klaus-Bernd Schürmann, Jens Stoye)

2005-05 Alignment of Tandem Repeats with Excision, Duplication, Substitution and Indels (EDSI)
(Michael Sammeth, Jens Stoye)

2005-06 Statistics of Cleavage Fragments in Random Weighted Strings
(Hans-Michael Kaltenbach, Henner Sudek, Sebastian Böcker, Sven Rahmann)

2006-01 Decomposing metabolomic isotope patterns
(Sebastian Böcker, Zsuzsanna Lipták, Anton Pervukhin)

2006-02 On Common Intervals with Errors
(Cedric Chauve, Yoan Diekmann, Steffen Heber, Julia Mixtacki, Sven Rahmann, Jens Stoye)

2007-01 Identifying metabolites with integer decomposition techniques, using only their mass spectrometric isotope patterns
(Sebastian Böcker, Matthias C. Letzel, Zsuzsanna Lipták, Anton Pervukhin)

2007-02	2-Stage Fault Tolerant Interval Group Testing (Ferdinando Cicalese, José Augusto Amgarten Quitzau)
2008-01	Online Abelian Pattern Matching (Tahir Ejaz, Sven Rahmann, Jens Stoye)
2008-02	A Space Efficient Representation for Sparse de Bruijn Subgraphs (José Augusto Amgarten Quitzau, Jens Stoye)
2008-03	Computing with Priced Information: game trees and the value dependent cost model. (Ferdinando Cicalese, Martin Milanič)