An $O(nm)$-time Certifying Algorithm for Recognizing HHD-free Graphs

Stavros D. Nikolopoulos and Leonidas Palios
Department of Computer Science, University of Ioannina
P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, palios}@cs.uoi.gr

Abstract

In this paper, we consider the recognition problem on a class of perfectly orderable graphs, namely, the HHD-free graphs, i.e., graphs that do not contain any induced subgraph isomorphic to a house, a hole, or a domino. We prove properties of the HHD-free graphs which enable us to present an $O(n\,m)$-time and $O(n+m)$-space algorithm for determining whether a given graph G on n vertices and m edges is HHD-free. We also describe how the algorithm can be augmented to provide a certificate (an induced house, hole, or domino) whenever it decides that the input graph is not HHD-free; the certificate computation requires $O(n+m)$ additional time and $O(n)$ space.

Keywords: HHD-free graphs, perfectly orderable graphs, certifying algorithms, recognition.

1 Introduction

A linear order $\prec$ on the vertices of a graph G is *perfect* if the ordered graph $(G, \prec)$ contains no induced P_4 *abcd* with $a \prec b$ and $d \prec c$ (such a P_4 is called an *obstruction*). In the early 1980s, Chvátal [2] defined the class of graphs that admit a perfect order and called them *perfectly orderable* graphs. The interest in perfectly orderable graphs comes from the fact that several problems in graph theory, which are NP-complete in general graphs, have polynomial-time solutions in graphs that admit a perfect order [1, 5]; unfortunately, it is NP-complete to decide whether a graph admits a perfect order [12]. Since the recognition of perfectly orderable graphs is NP-complete, we are interested in characterizing graphs which form polynomially recognizable subclasses of perfectly orderable graphs. Many such classes of graphs, with very interesting structural and algorithmic properties, have been defined so far and shown to admit polynomial-time recognitions (see [1, 5]); note however that not all subclasses of perfectly orderable graphs admit polynomial-time recognition [7].

In this paper, we consider the class of HHD-free graphs: a graph is *HHD-free* if it contains no induced subgraph isomorphic to a house, a hole (i.e., a chordless cycle on ≥ 5 vertices), or a domino (see Figure 1). The HHD-free graphs properly generalize the class of chordal (or triangulated) graphs [5]. In [8], Hoàng and Khouzam proved that the HHD-free graphs admit a perfect order, and thus are perfectly orderable. A superclass of the HHD-free graphs, which also properly generalizes the class of chordal graphs, is the class of HH-free graphs: a graph is *HH-free* if it contains no induced subgraph isomorphic to a house or a hole. Although an HH-free graph is not necessarily perfectly orderable, the complement of any HH-free graph is; this was conjectured by Chvátal and proved by Hayward [6].

Hoàng and Khouzam [8], while studying the class of brittle graphs (a well-known class of perfectly orderable graphs which contains the HHD-free graphs), showed that HHD-free graphs can be recognized

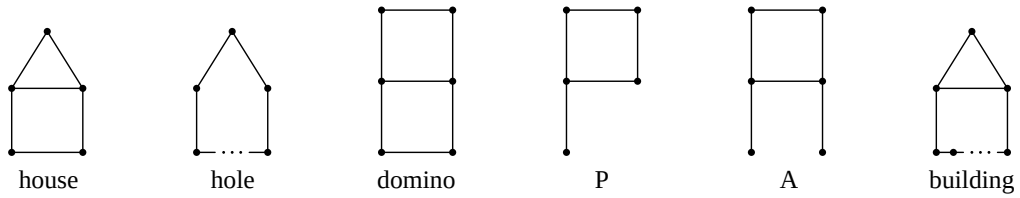

Figure 1: Some simple graphs.

in $O(n^4)$ time, where n denotes the number of vertices of the input graph. An improved result was obtained by Hoàng and Sritharan [9] who presented an $O(n^3)$-time algorithm for recognizing HH-free graphs and showed that HHD-free graphs can be recognized in $O(n^3)$ time as well; for each vertex v of the input graph, their algorithm relies on computing the chordal completion of the (ordered) non-neighbors of v, and checking whether the resulting graph is chordal. A further improvement was achieved by Nikolopoulos and Palios [14]: based on properties characterizing the chordal completion of a graph, they were able to avoid performing the chordal completion step, which is the most time-consuming ingredient of the algorithm in [9], and described algorithms for recognizing HH-free and HHD-free graphs that require $O(n \min\{m\,\alpha(n,n),\, m + n\log n\})$ time and $O(n+m)$-space, where m is the number of edges of the input graph, and $\alpha(\,,\,)$ denotes the very slowly growing functional inverse of Ackerman's function.

On other related classes of perfectly orderable graphs, Eschen *et al.* [4] recently described recognition algorithms for several of them, among which a recognition algorithm for HHP-free graphs; a graph is HHP-free if it contains no hole, no house, and no "P" as induced subgraphs (see Figure 1). Their algorithm is based on the property that every HHP-free graph is HHDA-free graph (a graph with no induced hole, house, domino, or "A"), and thus a graph G is HHP-free graph if and only if G is HHDA-free and contains no "P" as an induced subgraph. The characterization of HHDA-free graphs due to Olariu [15] (a graph G is HHDA-free if and only if every induced subgraph of G either is chordal or contains a non-trivial module) and the use of modular decomposition [11] allowed Eschen *et al.* to present an $O(n\,m)$-time recognition algorithm for HHP-free graphs.

In this paper, we present a new, faster algorithm for recognizing HHD-free graphs. For each vertex v of a given graph G, our algorithm computes the partition of the non-neighbors of v into sets of vertices based on their common neighbors with v, and following that, the connected components of the subgraphs induced by these partition sets. We show that if G is HHD-free, the graph obtained from G by shrinking each of these connected components into a single vertex is "almost chordal." As a result, we obtain an $O(n\,m)$-time and $O(n+m)$-space algorithm for determining whether a graph on n vertices and m edges is HHD-free. We also describe how the algorithm can be augmented to provide a certificate (an induced house, hole, or domino) whenever it decides that the input graph is not HHD-free; the certificate computation requires $O(n+m)$ additional time and $O(n)$ space.

The paper is structured as follows. In Section 2, we review the terminology and the notation that we use throughout the paper. In Section 3, we establish properties that enable us to efficiently determine whether a given graph is HHD-free, describe the algorithm, and give its analysis and the certificate computation. Section 4 summarizes our results and presents some open problems.

2 Terminology - Notation

We consider finite undirected graphs with no loops or multiple edges. Let G be such a graph; then, $V(G)$ and $E(G)$ denote the set of vertices and of edges of G respectively. The subgraph of G induced by a subset S of G's vertices is denoted by $G[S]$. The vertices adjacent to a vertex x of G form the

neighborhood $N(x)$ of x; the cardinality of $N(x)$ is the *degree* of x. The *closed neighborhood* of x is defined as $N[x] := N(x) \cup \{x\}$. We extend the notion of the neighborhood to sets as follows: for a set $A \subseteq V(G)$, we define $N(A) := \left(\bigcup_{x \in A} N(x)\right) - A$ and $N[A] := N(A) \cup A$.

A *path* in a graph G is a sequence of vertices $v_0 v_1 \cdots v_k$ such that $v_{i-1}v_i \in E(G)$ for $i = 1, 2, \ldots, k$; we say that this is a path from v_0 to v_k and that its *length* is k. A path is called *simple* if none of its vertices occurs more than once; it is called *trivial* if its length is equal to 0. A path (simple path) $v_0 v_1 \cdots v_k$ is called a *cycle* (*simple cycle*) of length $k+1$ if $v_0 v_k \in E(G)$. An edge connecting two non-consecutive vertices in a simple path (cycle) is called a *chord*; then, a simple path (cycle) $v_0 v_1 \cdots v_k$ of a graph G is *chordless* if G contains no chords of the path (cycle), i.e., $v_i v_j \notin E(G)$ for any two non-consecutive vertices v_i, v_j in the path (cycle). The chordless path (chordless cycle, respectively) on n vertices is commonly denoted by P_n (C_n, respectively).

A *connected component* of a graph G is a maximal set $A \subseteq V(G)$ such that the subgraph $G[A]$ is connected, i.e., there exists a path in G connecting any two vertices in A.

3 The Algorithm

In a fashion similar to the algorithms in [9, 14], our algorithm processes each vertex v of the input graph G and checks whether v participates in a hole, is the top vertex of a house or a building (see Figure 1), or is a corner vertex of a domino. Note that all these subgraphs include a path $y_1 uvw y_2$ where y_1, y_2 are non-neighbors of v; it is interesting to observe that the vertices y_1, y_2 have different common neighbors with v. This suggests that it may be a good idea to partition the set of non-neighbors of v based on their common neighbors with v, to shrink each of these partition sets into a single super-vertex, and then to work with the resulting graph. This is reinforced by the following observation.

Observation 3.1 *Let G be a graph, v a vertex of G, and y_1, y_2 be two non-neighbors of v in G such that $|N(y_1) \cap N(v)| = |N(y_2) \cap N(v)|$ and $N(y_1) \cap N(v) \neq N(y_2) \cap N(v)$. If $y_1 y_2 \in E(G)$, then G contains an induced house or C_5.*

Proof: Since $N(y_1) \cap N(v) \neq N(y_2) \cap N(v)$ and $|N(y_1) \cap N(v)| = |N(y_2) \cap N(v)|$, there exist vertices $u, w \in N(v)$ such that $u \in N(y_1) - N(y_2)$ and $w \in N(y_2) - N(y_1)$. But then, the vertices v, u, y_1, y_2, w induce a house or a C_5 depending on whether u, w are adjacent or not. ∎

However, shrinking each of the different partition sets into a single vertex leads to error as the following example indicates: consider the graph G on the left of Figure 2 which contains no hole, house, or domino; the partition of the non-neighbors of v based on the common neighbors with v yields the sets P, Q, R; shrinking these sets into vertices x, y, z, respectively, yields the graph on the right of Figure 2, which contains the hole $vaxyzc$.

A closer look at the example reveals that the error is due to the fact that the two connected components of the subgraph $G[Q]$ induced by the partition set Q in Figure 2 were shrunk into the same vertex. This suggests that if we intend to apply a shrinking mechanism, we need to treat the connected components of the partition sets as separate entities. In detail, we do the following:

- ▷ consider the partition of the non-neighbors $M(v)$ of vertex v in G based on the common neighbors of the vertices in $M(v)$ with v and let $(S_1, S_2, \ldots, S_\ell)$ be an ordering of the partition sets by non-decreasing number of such common neighbors;
- ▷ for each set S_i, consider the connected components of the subgraph $G[S_i]$;

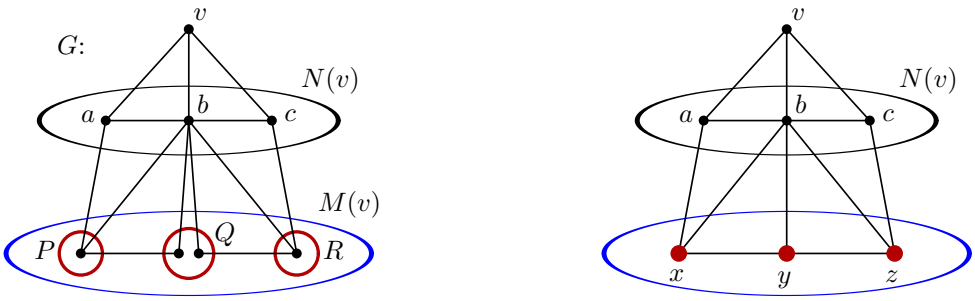

Figure 2: Shrinking the partition sets into vertices may lead to error.

▷ we construct an auxiliary graph G_v by shrinking each of these connected components into a single vertex: namely, for each $i = 1, 2, \ldots, \ell$, let

$$Z_i = \{\, z_{C_1}, z_{C_2}, \ldots, z_{C_{t_i}} \mid C_1, C_2, \ldots, C_{t_i} \text{ are the conn. components of } G[S_i] \,\}; \tag{1}$$

then
$V(G_v) = \{v\} \cup N(v) \cup \left(\bigcup_{i=1}^{\ell} Z_i\right)$
$E(G_v) = \{\, u\,w \mid u, w \in \{v\} \cup N(v) : uw \in E(G) \,\}$
$\quad \cup \{\, u\,z_C \mid u \in N(v), \exists\, x \in \text{conn. component } C \text{ of } G[S_i] : ux \in E(G) \,\}$
$\quad \cup \{\, z_C\, z_{C'} \mid \exists\, x, y \in \text{conn. components } C \text{ and } C' \text{ of } G[S_i] \text{ and } G[S_j], \text{ resp., where } i \neq j :$
$\qquad\qquad xy \in E(G) \,\}.$

Note that when a component C is shrunk into a vertex z_C, then (i) z_C is adjacent to a vertex $u \in N(v)$ iff there exists a vertex $x \in C$ such that $ux \in E(G)$, and (ii) z_C is adjacent to vertex $z_{C'}$ corresponding to a component $C' \neq C$ iff there exist vertices $x \in C$ and $y \in C'$ such that $xy \in E(G)$. The following result has important implications for the graph G_v.

Lemma 3.1 *Let G be an HHD-free graph, v a vertex of G, and A, B, C connected components of the subgraphs $G[S_i], G[S_j], G[S_k]$ induced by three distinct partition sets S_i, S_j, S_k, respectively, where $i < j < k$. Suppose further that there exist non-neighbors x, x', y, z of v in G, where $x, x' \in A$, $y \in B$, $z \in C$, such that $xz \in E(G)$ and $x'y \in E(G)$. Then, $yz \in E(G)$.*

Proof: Suppose for contradiction that $yz \notin E(G)$. Because G is HHD-free and $x'y \in E(G)$, then Observation 3.1 implies that $|N(x') \cap N(v)| \neq |N(y) \cap N(v)|$; in fact since $i < j$, $|N(x') \cap N(v)| < |N(y) \cap N(v)|$, which implies that there exists a vertex $u \in (N(y) - N(x')) \cap N(v)$. Moreover, since $j < k$, we have that $|N(y) \cap N(v)| \leq |N(z) \cap N(v)|$, and because $j \neq k$, there exists a vertex $w \in (N(z) - N(y)) \cap N(v)$. We consider the following cases:

a) $uz \in E(G)$: Then, $x = x'$, otherwise the graph G would contain the hole $uzx \cdots x'y$. But then, if $wx \in E(G)$, the vertices v, w, x, y, u would induce a house or a C_5 in G depending on whether $uw \in E(G)$ or not, whereas if $wx \notin E(G)$, G would contain a house induced by u, y, x, z, w or a domino induced by v, u, y, x, z, w depending on whether $uw \in E(G)$ or not; see the graph on the left in Figure 3.

b) $uz \notin E(G)$: Then, if $wx \in E(G)$, the vertices v, w, x', y, u would induce a house or a C_5 in the graph G depending on whether $uw \in E(G)$, whereas if $wx \notin E(G)$, G would contain the hole $wzx \cdots x'yu$ or $vwzx \cdots x'yu$ depending on whether $uw \in E(G)$ (no matter whether x is equal to x' or not); see the graph on the right in Figure 3.

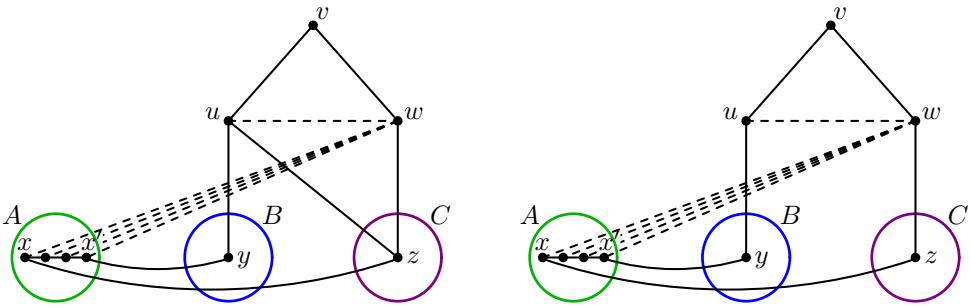

Figure 3: For the proof of Lemma 3.1 (edges that may or may not exist are shown dashed).

In all cases, we concluded that the graph G would contain an induced house, hole, or domino, which contradicts the assumption that G is HHD-free; thus, $yz \in E(G)$. ∎

In terms of the graph G_v, Lemma 3.1 implies the following corollary:

Corollary 3.1 *Let G be an HHD-free graph, v a vertex of G, G_v the auxiliary graph described earlier in terms of G and v, and z_A, z_B, z_C be vertices of G_v such that $z_A \in Z_i$, $z_B \in Z_j$, and $z_C \in Z_k$ where $i < j < k$. If $z_A z_B \in E(G_v)$ and $z_A z_C \in E(G_v)$, then $z_B z_C \in E(G_v)$.*

Lemma 3.1 and Corollary 3.1 prove very useful in the special case in which the graph G is such that for every edge $xy \in E(G)$, where x belongs to a connected component A of a subgraph $G[S_i]$ and y belongs to a connected component B of $G[S_j]$ with $j > i$, no vertex in A is adjacent to any vertex in $S_j - B$. In this case, in the auxiliary graph G_v, for all $1 \leq i < j \leq \ell$, any vertex in Z_i (i.e., corresponding to a connected component of the subgraph $G[S_i]$) is adjacent to at most one vertex in each Z_j (i.e., corresponding to a component of $G[S_j]$). Then, if G is HHD-free, Corollary 3.1 implies that the subgraph $G_v[\bigcup_{t=1}^{\ell} Z_t]$ of G_v induced by the non-neighbors of v is chordal, and hence no chordal completion is needed. In general, however, G may have edges xy and $x'z$, where x, x' belong to a connected component of a subgraph $G[S_i]$ and y, z belong to distinct connected components of $G[S_j]$ with $j > i$; then, we take advantage of the following lemma.

Lemma 3.2 *Let G be an HHD-free graph, A a connected component of the subgraph $G[S_i]$, and B, B' distinct connected components of $G[S_j]$, where $j > i$, such that G contains edges connecting a vertex in A to a vertex in B, and a vertex in A to a vertex in B'. Then:*

(i) In G, each vertex in A that is adjacent to at least one vertex in $B \cup B'$ is adjacent to all the vertices in $B \cup B'$.

(ii) If C is a connected component of the subgraph $G[S_k]$, where $k > j$, such that G contains an edge connecting a vertex in B to a vertex in C. Then, in G, each vertex in C that is adjacent to at least one vertex in $B \cup B'$ is adjacent to all the vertices in $B \cup B'$.

Proof: The facts that the graph G is HHD-free and contains an edge connecting a vertex in A to a vertex in B, that B, B' are connected components of $G[S_j]$, and that $i < j$ imply that

$$\forall\, a \in A,\ b \in B,\ b' \in B', \qquad N(a) \cap N(v) \quad \subset \quad N(b) \cap N(v) \quad = \quad N(b') \cap N(v);$$

otherwise, G would contain an induced house or C_5, a contradiction. Then, there exists a vertex $u \in N(v)$ that is adjacent to all the vertices in $B \cup B'$ and to no vertex in A.

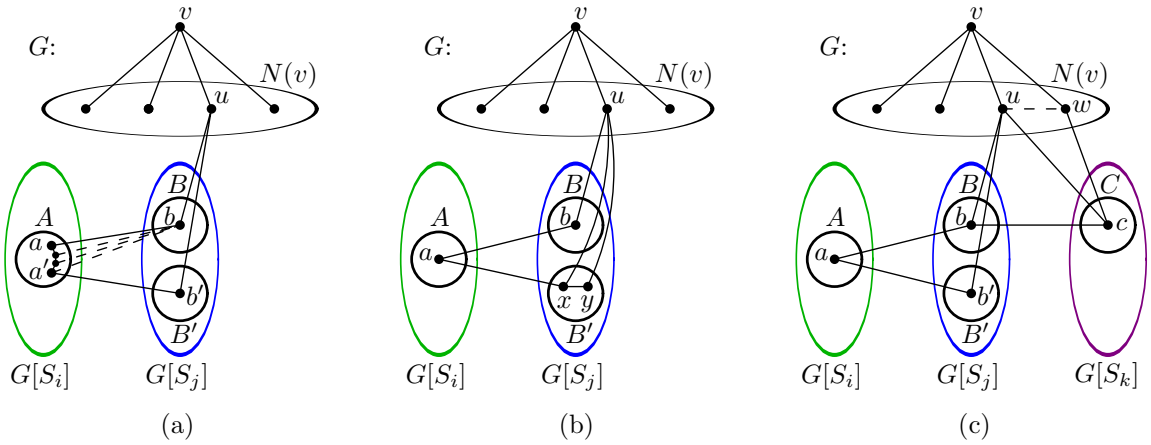

Figure 4: For the proof of Lemma 3.2.

(i) Consider a vertex $a \in A$ that is adjacent to a vertex $b \in B$; we will show that a is adjacent to all the vertices in $B \cup B'$. Because there exists an edge connecting a vertex in A to a vertex in B' and because the subgraph $G[A]$ is connected, G contains a chordless path connecting a to a vertex $b' \in B'$, whose vertices belong to A except for b'; if this path were of length greater than 1, then its vertices along with u and b would induce a house, a hole, or a domino depending on which vertices of the path are adjacent to b (see Figure 4(a)); this is a contradiction. Thus, the path is of length 1, and a is adjacent to b' in G. In fact, a is adjacent to all the vertices in B'; if a were not adjacent to a vertex in B', then because the subgraph $G[B']$ is connected and $ab' \in E(G)$, there exist vertices $x, y \in B'$ such that $\{ax, xy\} \subseteq E(G)$ and $ay \notin E(G)$. Then, the vertices u, a, b, x, y would induce a house in G with y at its top; see Figure 4(b). Similarly, since a is adjacent to a vertex in B', we can show that a is adjacent to all the vertices in B. In summary, a is adjacent to all the vertices in $B \cup B'$.

(ii) Consider a vertex $c \in C$ that is adjacent in G to a vertex $b \in B$. Then, because the graph G is HHD-free and $j < k$, we have that $N(b) \cap N(v) \subset N(c) \cap N(v)$, which implies that the vertex u is adjacent to all the vertices in C. Let a be a vertex in A that is adjacent to at least one vertex in $B \cup B'$; then, in accordance with part (i), a is adjacent to all the vertices in $B \cup B'$. If $ac \in E(G)$, then because a is adjacent to all the vertices in $B \cup B'$, Lemma 3.1 implies that c is adjacent to all the vertices in $B \cup B'$. If $ac \notin E(G)$, then c is adjacent to all the vertices in B', for if there was $b' \in B'$ such that $b'c \notin E(G)$, the vertices a, b, b', c, u would induce a house in G with c at its top; see Figure 4(c). In turn, by the symmetry of B and B', the fact that c is adjacent to a vertex in B' implies that c is also adjacent to all the vertices in B. ∎

It is worth noting that although in an HHD-free graph, each vertex in $A \cup C$ that is adjacent to at least one vertex in $B \cup B'$ is in fact adjacent to each vertex in $B \cup B'$, it is not necessarily true that each vertex in $B \cup B'$ is adjacent to each vertex in $A \cup C$; consider the HHD-free graph shown in Figure 5.

Lemma 3.2, statement (ii) implies that if in an HHD-free graph G there exist edges connecting vertices of a connected component A of a subgraph $G[S_i]$ to vertices in at least 2 connected components $B_1, B_2, \ldots, B_k$ of a subgraph $G[S_j]$ where $j > i$, then the vertices in $B_1 \cup B_2 \cup \ldots \cup B_k$ are adjacent to the exact same neighbors in $\left(\bigcup_{r=j+1}^{\ell} S_r\right)$; additionally, as the components $B_1, B_2, \ldots, B_k$ are all subsets of S_j, the vertices in $B_1 \cup B_2 \cup \ldots \cup B_k$ are adjacent to the same vertices in $N(v)$, and thus they are adjacent in G to the exact same neighbors in $\left(\bigcup_{r=j+1}^{\ell} S_r\right) \cup N(v)$. In terms of the graph G_v, this implies that the vertices $z_{B_1}, z_{B_2}, \ldots, z_{B_k}$ corresponding to the connected components $B_1, B_2, \ldots, B_k$ of $G[S_j]$ are adjacent to the exact same neighbors in $\left(\bigcup_{r=j+1}^{\ell} Z_r\right) \cup N(v)$. Moreover, the transitivity of equality leads to the following corollary.

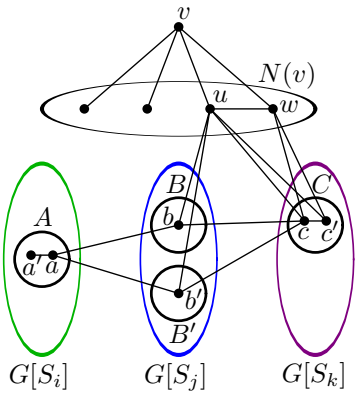

Figure 5: The vertices in $B \cup B'$ are not necessarily adjacent to all the vertices in $A \cup C$.

Corollary 3.2 *Let G be an HHD-free graph, and $A_1, A_2, \ldots, A_h$ be connected components of subgraphs $G[S_r]$, where $r < j$, such that for each $i = 1, 2, \ldots, h$,*

- *the graph G contains edges connecting vertices in at least 2 distinct components of $G[S_j]$ to vertices in A_i, and*
- *there exists a component B of $G[S_j]$ such that at least one vertex in A_i and at least one vertex in $\bigcup_{t=1}^{i-1} N(A_t)$ are adjacent to a vertex in B.*

(in terms of the graph G_v, for each i, we have that $|N_{G_v}(z_{A_i}) \cap Z_j| \geq 2$ and $\Big(N_{G_v}(z_{A_i}) \cap \big(\bigcup_{t=1}^{i-1} N_{G_v}(z_{A_t})\big)\Big) \cap Z_j \neq \emptyset$). Then,

(i) in G, all the vertices in $\bigcup_{i=1}^{h} \big(N(A_i) \cap S_j\big)$ are adjacent to the exact same neighbors in $\big(\bigcup_{r=j+1}^{\ell} S_r\big) \cup N(v)$;

(ii) in G_v, all the vertices in $\bigcup_{i=1}^{h} \big(N_{G_v}(z_{A_i}) \cap Z_j\big)$ are adjacent to the exact same neighbors in $\big(\bigcup_{r=j+1}^{\ell} Z_r\big) \cup N(v)$.

In order to be able to check that case (ii) of Corollary 3.2 holds for the graph G_v, we maintain a partition $\mathcal{P}_{Z_j}$ of the vertices in Z_j that correspond to the connected components of the subgraph $G[S_j]$: initially, each partition set contains a single vertex of Z_j; every time we process a vertex z_{A_i} which is adjacent to vertices $z_{B_1}, z_{B_2}, \ldots, z_{B_t} \in Z_j$, where $t \geq 2$, we union the sets of the partition $\mathcal{P}_{Z_j}$ that contain the vertices $z_{B_1}, z_{B_2}, \ldots, z_{B_t}$. Then, when we process the vertices in Z_j, we check that the vertices in each partition set are adjacent to the exact same neighbors in $\big(\bigcup_{r=j+1}^{\ell} Z_r\big) \cup N(v)$.

Based on these results, we give in Figure 6 Algorithm Recognize-HHD-free, in which, for each vertex v of the given graph G, we construct the auxiliary graph G_v by shrinking each of the connected components of each of the subgraphs $G[S_i]$, $i = 1, 2, \ldots, \ell$, into a single vertex, and then, for each $i = 1, 2, \ldots, \ell$, we process the set Z_i of vertices corresponding to the connected components of $G[S_i]$ together (note that, by construction, in the graph G_v there are no edges between any two vertices $z_C, z_{C'} \in Z_i$). We also mention that Step 1.4 of our algorithm, which is applied on G_v, is an extension of the linear-time algorithm for testing whether an ordering of the vertices of a given graph is a perfect elimination ordering [5, 16].

We note that when we check that all the vertices in a set P_j are adjacent to the same neighbors, we check only the neighbors in $\bigcup_{r=i+1}^{\ell} Z_r$, and not in $\big(\bigcup_{r=i+1}^{\ell} Z_r\big) \cup N(v)$ as suggested by Corollary 3.2, statement (ii); this is so, because all the vertices in P_j correspond to connected components of the subgraph $G[S_i]$ for the same set S_i, and so, by construction, they have the same neighbors in $N(v)$.

Algorithm Recognize-HHD-free(G)

1. **for** each vertex v of G **do**

   1.1. Compute the neighbors $N(v)$ and non-neighbors $M(v) = V(G) - (N(v) \cup \{v\})$ of v in G; compute the partition $\mathcal{S}_v$ of the set $M(v)$ based on the common neighbors of the vertices in $M(v)$ with v in G, and order the partition sets by non-decreasing number of such common neighbors; let $\mathcal{S}_v = (S_1, S_2, \ldots, S_\ell)$ be the resulting ordering;

   1.2. **for** each edge xy of G, where $x, y \in M(v)$, **do**
   if x, y belong to different partition sets of $\mathcal{S}_v$ **and** $|N(x) \cap N(v)| = |N(y) \cap N(v)|$
   then print("The graph G is not HHD-free"); **exit**;

   1.3. Construct the auxiliary graph G_v by shrinking each connected component C of the subgraphs $G[S_i]$, $i = 1, 2, \ldots, \ell$, into a single vertex z_C;

   1.4. **for** $i \leftarrow 1$ to ℓ **do**
   form a partition $\mathcal{P}_{Z_i}$ of the set Z_i of vertices of G_v that correspond to the connected components of $G[S_i]$, by placing each of these vertices in a separate partition set;
   for each vertex $z_C \in Z_i$ **do**
   associate with z_C an initially empty set $A(z_C)$;
   for $i \leftarrow 1$ to ℓ **do**
   let the partition $\mathcal{P}_{Z_i}$ of Z_i be $\mathcal{P}_{Z_i} = \{P_1, P_2, \ldots, P_t\}$;
   for $j \leftarrow 1$ to t **do**
   let z_C be any vertex contained in the set P_j;
   $X' \leftarrow N_{G_v}(z_C) \cap \left(\bigcup_{r=i+1}^{\ell} Z_r\right)$;
   if P_j is not a singleton set
   then if there exists a vertex in P_j that is not adjacent in G_v to a vertex in X' or is adjacent to a vertex in $\left(\bigcup_{r=i+1}^{\ell} Z_r\right) - X'$
   then print("The graph G is not HHD-free"); **exit**;
   if $X' \neq \emptyset$
   then let Z_k be the minimum-index set such that $X' \cap Z_k \neq \emptyset$;
   $W \leftarrow X' \cap Z_k$;
   union the sets of the partition $\mathcal{P}_{Z_k}$ (of Z_k) that contain the vertices in W;
   $X \leftarrow X' \cup \left(N_{G_v}(z_C) \cap N(v)\right)$;
   choose any $z_B \in W$ and concatenate the set $X - W$ to $A(z_B)$;
   if $\left(\bigcup_{z \in P_j} A(z)\right) - N_{G_v}(z_C) \neq \emptyset$
   then print("The graph G is not HHD-free"); **exit**;

2. **print**("The graph G is HHD-free");

Figure 6: Algorithm Recognize-HHD-free.

Additionally, when this check is successful, we take advantage of the fact that all the vertices in P_j are adjacent to the same neighbors in $\bigcup_{r=i+1}^{\ell} Z_r$, when we union the sets of the partition $\mathcal{P}_{Z_k}$ that contain the vertices in W; we need only do the unioning once for the entire set P_j rather than once for each vertex in the set.

The correctness of the Algorithm Recognize-HHD-free is established in Theorem 3.1 with the help of Lemma 3.3 (*due to space limitation, the proofs have been omitted; however, the proofs are given in the Appendix in order to assist the reviewers*).

Lemma 3.3 *Let G be a graph, v a vertex of G, G_v the auxiliary graph described earlier in terms of G and v, and Z_i $(i = 1, 2, \ldots, \ell)$ the sets of vertices of G_v defined in Eqn. (1). Suppose that there exist vertices $z_A \in Z_i$ and $z_{A'} \in Z_j$ of G_v corresponding to connected components A, A' of subgraphs $G[S_i]$ and $G[S_j]$, respectively, where $i < j$, such that $z_A z_{A'} \in E(G_v)$ and there exists a vertex $x \in \left(\bigcup_{r=j+1}^{\ell} Z_r\right) \cup N(v)$ such that $x \in N_{G_v}(z_A) - N_{G_v}(z_{A'})$. Then if Algorithm Recognize-HHD-free is run on G, it reports that G is not HHD-free and stops.*

Theorem 3.1 *When Algorithm Recognize-HHD-free is run on a graph G, it reports that G is not HHD-free if and only if G is indeed not HHD-free.*

Remark
We note that in the course of Algorithm Recognize-HHD-free we do not check whether the conditions of Lemma 3.2, statement (i), hold, although we know that if they do not hold then the input graph G is not HHD-free. In fact, we do not check the conditions of Lemma 3.2, statement (ii), either; instead, we check the weaker conditions stated in Corollary 3.2. Nevertheless, the conditions that we check suffice to enable us to recognize HHD-free graphs.

3.1 Time and Space Complexity

Let n be the number of vertices and m be the number of edges of the graph G. Since each of the forbidden subgraphs that we are looking for (a house, a hole, or a domino) is connected, we may assume that G is connected, otherwise we work on G's connected components which we can compute in $O(n + m)$ time [3]; thus, $n = O(m)$. Below, we give the time and space complexity of each step of Algorithm Recognize-HHD-free.

For a vertex v, its neighbors and non-neighbors in the graph G can be stored in $O(n)$-size arrays for constant-time access; this takes $O(n)$ time. The partition $\mathcal{S}_v$ can be computed in $O(m + n\,deg(v))$ time time and $O(n)$ space, where $deg(v)$ denotes the degree of v in G; see [13][1]. After having computed for a vertex of each of the partition sets of $\mathcal{S}_v$, the number of its common neighbors with v, which can be done in $O(n + m)$ time, we can form the ordered sequence $(S_1, S_2, \ldots, S_\ell)$ in $O(\ell + deg(v)) = O(n)$ time and $O(n)$ space using bucket sorting. Thus, Step 1.1 takes $O(m + n\,deg(v))$ time and $O(n)$ space in total.

Step 1.2 takes $O(m)$ time assuming that each non-neighbor of vertex v stores the index of the set of the partition $\mathcal{S}_v$ to which it belongs; storing this information on each such vertex can be done in $O(n)$ time by traversing the sets $S_1, S_2, \ldots, S_\ell$. Thus, Step 1.2 takes $O(n + m)$ time and $O(n)$ space.

Adjacency-list representations of the subgraphs $G[S_i]$, $i = 1, 2, \ldots, \ell$, can be obtained in $O(n + m)$ time and space by appropriate partitioning of a copy of an adjacency-list representation of the graph G and removal of unneeded records; then, computing the connected components of all these subgraphs takes a total of $O(n + m)$ time and space, from which the graph G_v can be constructed in $O(n + m)$ additional time and space. Thus, Step 1.3 takes a total of $O(n + m)$ time and space.

Crucial for Step 1.4 is the construction and processing of the partitions $\mathcal{P}_{Z_i}$, $i = 1, 2, \ldots, \ell$. These are maintained by means of an auxiliary (multi-)graph H_v: members of the same partition set belong to the same connected component of H_v. The graph H_v has one vertex for each connected component of each subgraph $G[S_i]$, $i = 1, 2, \ldots, \ell$; hence, with a slight abuse of notation, we can write that $V(H_v) = \bigcup_{i=1}^{\ell} Z_i$. Initially, the graph H_v has no edges.

[1] An algorithm to construct a partition of a set L_2 in terms of adjacency to elements of a set L_1 is given in Section 3.2 of [13] with a stated time complexity of $O(m + n\,|L_2|)$; yet, it can be easily seen that the algorithm has a time complexity of $O(m + |L_1| \cdot |L_2|)$, which in our case gives $O(m + n\,deg(v))$ since $|L_1| = deg(v)$ and $|L_2| = O(n)$.

- ▷ In order to compute the partition $\mathcal{P}_{Z_i}$ of the set Z_i, we compute the connected components of the subgraph $H_v[Z_i]$; graph traversing algorithms, such as, depth-first search and breadth-first search, can be used on $H_v[Z_i]$ to yield the connected components in time linear in the number of vertices and edges of $H_v[Z_i]$.

- ▷ In order to union the sets of a partition $\mathcal{P}_{Z_k}$ that contain the vertices in a set W, we pick a vertex, say, $z \in W$, and add edges in H_v connecting z to all the other vertices in W; this takes $O(|W|)$ time and space.

The above description implies that forming the initial partitions $\mathcal{P}_{Z_i}$, $i = 1, 2, \ldots, \ell$, where each vertex in Z_i is placed in a separate partition set corresponds to constructing the graph H_v without any edges; this can be done in $O(n)$ time and space. Initializing the sets $A(\ )$ for all the vertices in $\bigcup_{i=1}^{\ell} Z_i$ also takes $O(n)$ time and space. Now, for each $i = 1, 2, \ldots, \ell$, computing the partition $\mathcal{P}_{Z_i}$ takes time linear in the number of vertices and edges of $H_v[Z_i]$. Let us consider the processing of a set P_j of the partition $\mathcal{P}_{Z_i}$. Computing the set X' takes $O(deg_{G_v}(z_C))$ time and space, where $deg_{G_v}(z_C)$ denotes the degree of vertex z_C in the graph G_v. Checking if P_j is a singleton set takes $O(1)$ time, and checking if all the vertices in P_j are adjacent to exactly the vertices in X' among the vertices in $\bigcup_{i=i+1}^{\ell} Z_i$ takes $O(\sum_{z \in P_j} deg_{G_v}(z))$ time. Next, checking whether X' is non-empty takes $O(1)$ time while doing all the processing if $X' \neq \emptyset$ takes $O(deg_{G_v}(z_C))$ time and space; note that $|W| \leq |X'| \leq deg_{G_v}(z_c)$ and recall that unioning the sets of the partition $\mathcal{P}_{Z_k}$ that contain the vertices in W involves adding $|W| - 1$ edges in H_v. Finally, checking whether $\big(\bigcup_{z \in P_j} A(z)\big) - N_{G_v}(z_C) \neq \emptyset$ takes $O\big(\bigcup_{z \in P_j} |A(z)|\big)$ time. In summary, processing the set P_j takes $O\big(\sum_{z \in P_j} \big(deg_{G_v}(z) + |A(z)|\big)\big)$ time and $O(deg_{G_v}(z_C))$ space. Since the sets of each partition $\mathcal{P}_{Z_i}$ of the set Z_i are disjoint and the sets Z_i are disjoint, we have that $O\big(\sum_{i=1}^{\ell} \sum_{P_j \in \mathcal{P}_{Z_i}} \sum_{z \in P_j} deg_{G_v}(z)\big) = O\big(|V(G_v)| + |E(G_v)|\big) = O(n + m)$. Additionally, since the sets $A(\ )$ are formed by concatenating some of the neighbors in G_v of one vertex z_C from each set P_j, we have that $O\big(\sum_{i=1}^{\ell} \sum_{P_j \in \mathcal{P}_{Z_i}} \sum_{z \in P_j} |A(z)|\big) = O\big(\sum_{i=1}^{\ell} \sum_{P_j \in \mathcal{P}_{Z_i}} \sum_{z \in P_j} deg_{G_v}(z)\big) = O\big(|V(G_v)| + |E(G_v)|\big) = O(n + m)$ as well. Thus, in total, Step 1.4 takes $O(n + m)$ time and space.

Since Steps 1.1-1.4 are executed for each vertex v of the input graph G and Step 2 takes $O(1)$ time, we have that the overall time complexity of Algorithm Recognize-HHD-free is:

$$\Big[\sum_{v \in V(G)} \Big(O(m + n\, deg(v)) \Big) + O(n + m) \Big] + O(1) \quad = \quad O(n\, m).$$

Therefore,

Theorem 3.2 *Let G be an undirected graph on n vertices and m edges. Then, Algorithm Recognize-HHD-free determines whether G is an HHD-free graph in $O(nm)$ time and $O(n + m)$ space.*

3.2 Providing a Certificate

Algorithm Recognize-HHD-free can be made to provide a certificate (a house, a hole, or a domino) whenever it decides that the input graph G is not HHD-free. The algorithm reports that the graph G is not HHD-free in three occasions, once in Step 1.2 and twice in Step 1.4. We describe next how we handle each of these cases.

Step 1.2: In this case, we have two non-neighbors x, y of v which have the same number of common neighbors with v and belong to different partition sets. Then, there exist vertices $u \in (N(x) \cap N(v)) - N(y)$ and $w \in (N(y) \cap N(v)) - N(x)$; we can find these vertices in $O(n)$ time using $O(n)$ space by traversing the adjacency lists of x and of y and marking the neighbors of x and y in two $O(n)$-size arrays, and then by traversing these two arrays. The vertices v, u, w, x, y induce a house or a C_5 depending on whether u, w are adjacent in G or not.

Step 1.4: In order to be able to efficiently produce a certificate when Algorithm Recognize-HHD-free reports that the input graph G is not HHD-free in Step 1.4, we do the following additional work:

W_1: Whenever, during the processing of a set P_j of a partition $\mathcal{P}_{Z_i}$, we need to union the sets of a partition $\mathcal{P}_{Z_k}$ containing the vertices in the set W, which is done by adding edges in the auxiliary multi-graph H_v (as explained in Section 3.1), we associate with each such edge the selected vertex z_C of P_j.

W_2: When processing a set P_j, we store with each element of the set $X - W$, which is added to $A(z_B)$ for $z_B \in W$, a reference to the selected vertex z_C of P_j; in this way, for each vertex z, each element of the set $A(z)$ carries a reference to a vertex of the set during whose processing this element was added to $A(z)$.

Note that this additional work does not increase asymptotically the time and space complexity of the algorithm.

In Step 1.4, Algorithm Recognize-HHD-free reports that the graph G is not HHD-free in the following two occasions:

1. There exists a vertex in a set P_j of a partition $\mathcal{P}_{Z_i}$ that is not adjacent to a vertex in X' or is adjacent to a vertex in $\left(\bigcup_{r=i+1}^{\ell} Z_r\right) - X'$. In any case, there exist vertices $z_A, z_Y, z_{Y'}, z_D$ of the graph G_v such that z_A corresponds to a connected component A of $G[S_p]$, $z_Y, z_{Y'} \in P_j$ correspond to distinct connected components Y, Y' of $G[S_i]$, z_D corresponds to a connected component D of $G[S_q]$ where $p < i < q$, $\{z_A z_Y, z_A z_{Y'}, z_{Y'} z_D\} \subseteq E(G_v)$, and $z_Y z_D \notin E(G_v)$. Additionally, because $p < i < q$, there exist vertices $u, w \in N(v)$ such that u is adjacent to all the vertices in S_i and to no vertex in S_p, and w is adjacent to all the vertices in S_q and to no vertex in S_i.

We can find the vertices $z_A, z_Y, z_{Y'}, z_D$ by constructing a depth-first search tree of the subgraph $H_v[P_j]$, and by testing for any pair of adjacent vertices in the tree, whether they have the exact same neighbors in $\bigcup_{r=i+1}^{\ell} Z_r$ (note that (i) if this is true for every pair of adjacent vertices, then clearly all vertices in P_j have the same neighbors in $\bigcup_{r=i+1}^{\ell} Z_r$, and (ii) this test takes $O\left(\sum_{z \in P_j} deg_{G_v}(z)\right)$ time as in the algorithm's analysis). In our case, we will be able to find two adjacent vertices z_1, z_2 in the tree and a vertex $z_3 \in \bigcup_{r=i+1}^{\ell} Z_r$ such that z_2 is adjacent to z_3 in G_v whereas z_1 is not; then, $z_Y = z_1$, $z_{Y'} = z_2$, $z_D = z_3$, while z_A is the vertex associated with the tree edge $z_1 z_2$ (see W_1).

We consider the following two cases depending on whether z_A, z_D are adjacent in G_v or not:

1a. $z_A z_D \notin E(G_v)$: We traverse the adjacency lists of the vertices in Y until we find a vertex $y \in Y$ which is adjacent to a vertex in A. Next, we run breadth-first search in the subgraph $G[\{y\} \cup A \cup Y \cup D]$ starting at y until a vertex $z \in D$ is encountered; let ρ be the path in the breadth-first search tree connecting y to z. Clearly, ρ is chordless and because $z_A z_D \notin E(G_v)$ and $z_Y z_{Y'} \notin E(G_v)$, it is of the form $\rho = y x_1 \cdots x_r y'_1 \cdots y'_s z$ where $r, s \geq 1$, $x_1, \ldots, x_r \in A$, and $y'_1, \ldots, y'_s \in Y'$. If $r > 1$, then the vertices $u, y, x_1, \ldots, x_r, y'_1$ induce a hole in G. If $r = 1$ and $s > 1$, the vertices u, y, x_1, y'_1, y'_2 induce a house in G with y'_2 at its top. Finally, if $r = 1$ and $s = 1$, then if $uz \in E(G)$, the vertices u, y, x_1, y'_1, z induce a house with z at its top, whereas if $uz \notin E(G)$, the vertices v, u, y'_1, z, w induce a house or a C_5.

1b. $z_A z_D \in E(G_v)$: As in the previous case, we traverse the adjacency lists of the vertices in Y until we find a vertex $y \in Y$ which is adjacent to a vertex in A. Next, we run breadth-first search in the subgraph $G[\{y\} \cup A \cup D]$ starting at y until a vertex $z \in D$ is encountered; let ρ be the path in the breadth-first search tree connecting y to z. Again, ρ is chordless and because $z_Y z_D \notin E(G_v)$, it is of the form $\rho = y x_1 \cdots x_r z$ where $r \geq 1$ and $x_1, \ldots, x_r \in A$. (Note that the cases considered below follow the analysis of cases in the proof of Lemma 3.1; see also Figure 3.) Consider first that $uz \in E(G)$. If $r > 1$, the vertices $uz x_1 \cdots x_r y$ induce a hole in G.

If $r = 1$, then if $wx_1 \in E(G)$, the vertices v, w, x_1, y, u induce a house or a C_5 in G depending on whether $uw \in E(G)$ or not, whereas if $wx_1 \notin E(G)$, G contains a house induced by the vertices u, y, x_1, z, w or a domino induced by v, u, y, x_1, z, w depending on whether $uw \in E(G)$ or not. Suppose next that $uz \notin E(G)$. Then, if $wx_1 \in E(G)$, the vertices v, w, x_r, y, u induce a house or a C_5 depending on whether $uw \in E(G)$, whereas if $wx_1 \notin E(G)$, G contains the hole $wzx_1 \cdots x_r yu$ or $vwzx_1 \cdots x_r yu$ depending on whether $uw \in E(G)$.

2. For a set P_j of a partition $\mathcal{P}_{Z_i}$, we have that $\left(\bigcup_{z \in P_j} A(z)\right) - N_{G_v}(z_C) \neq \emptyset$, i.e., there exist vertices $z_Y \in P_j$ (z_Y corresponds to a component Y of $G[S_i]$) and $z' \in \left(\bigcup_{r=i+1}^{\ell} Z_r\right) \cup N_{G_v}(v)$ such that $z' \in A(z_Y)$ and $z' \notin N_{G_v}(z_C)$; note that since the algorithm has not stopped earlier, z_Y and z_C have the exact same neighbors in $\bigcup_{r=i+1}^{\ell} Z_r$, which implies that $z' \notin N_{G_v}(z_Y)$. Since $z' \in A(z_Y)$, by means of W_2, z' is associated with a vertex $z_A \in \bigcup_{r=1}^{i-1} Z_r$ such that $z_Y, z' \in N_{G_v}(z_A)$. If $z' \in N(v)$, we locate two vertices $x \in A$ and $y \in Y$ such that $xy \in E(G)$; because $z' \in N_{G_v}(z_A) - N_{G_v}(z_Y)$, z' is adjacent to x in G but is not adjacent to y. Moreover, there exists a vertex $w \in N(v)$ such that $w \in N(y) - N(x)$; then, the vertices v, z', w, x, y induce a house or a C_5 in G. Now, if $z' \notin N(v)$, then $z' = z_D$ corresponding to a component D of a subgraph $G[S_q]$ with $q > i$. This case is identical to Case 1b above.

It is not difficult to see that doing the work described above and checking the adjacencies in each of the aforementioned cases can be performed in $O(n + m)$ time using $O(n)$ space. Thus, we have:

Theorem 3.3 *Let G be an undirected graph on n vertices and m edges. Then, Algorithm Recognize-HHD-free can be augmented to produce a house, a hole, or a domino whenever it decides that G is not an HHD-free graph in $O(n + m)$ additional time and $O(n)$ additional space.*

4 Concluding Remarks

We have presented a recognition algorithm for the class of HHD-free graphs that runs in $O(n\,m)$ time and requires $O(n + m)$ space, where n is the number of vertices and m is the number of edges of the input graph. Moreover, we show how our algorithm can be augmented to yield, in $O(n + m)$ time and $O(n)$ space, a certificate (a house, a hole, or a domino) whenever it decides that the input graph is not HHD-free.

Despite the close relation between HHD-free and HH-free graphs, our results do not lead to an improvement in the recognition time complexity for HH-free graphs; therefore, we leave as an open problem the design of an $O(n\,m)$-time algorithm for recognizing HH-free graphs. Additionally, it would be interesting to obtain faster recognition algorithms for other related classes of graphs, such as, the brittle and the semi-simplicial graphs.

References

[1] A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph classes: A survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[2] V. Chvátal, Perfectly ordered graphs, *Annals of Discrete Math.* **21**, 63–65, 1984.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.

[4] E.M. Eschen, J.L. Johnson, J.P. Spinrad, and R. Sritharan, Recognition of some perfectly orderable graph classes, *Discrete Appl. Math.* **128**, 355–373, 2003.

[5] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Inc., 1980.

[6] R. Hayward, Meyniel weakly triangulated graphs I: co-perfect orderability, *Discrete Appl. Math.* **73**, 199–210, 1997.

[7] C.T. Hoàng, On the complexity of recognizing a class of perfectly orderable graphs, *Discrete Appl. Math.* **66**, 219–226, 1996.

[8] C.T. Hoàng and N. Khouzam, On brittle graphs, *J. Graph Theory* **12**, 391–404, 1988.

[9] C.T. Hoàng and R. Sritharan, Finding houses and holes in graphs, *Theoret. Comput. Sci.* **259**, 233–244, 2001.

[10] B. Jamison and S. Olariu, On the semi-perfect elimination, *Adv. Appl. Math.* **9**, 364–376, 1988.

[11] R.M. McConnell and J. Spinrad, Linear-time modular decomposition and efficient transitive orientation, *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'94)*, 536–545, 1994.

[12] M. Middendorf and F. Pfeiffer, On the complexity of recognizing perfectly orderable graphs, *Discrete Math.* **80**, 327–333, 1990.

[13] S.D. Nikolopoulos and L. Palios, Algorithms for P_4-comparability graph recognition and acyclic P_4-transitive orientation, *Algorithmica* **39**, 95–126, 2004.

[14] S.D. Nikolopoulos and L. Palios, Recognizing HH-free, HHD-free, and Welsh-Powell opposition graphs, *Discrete Math. and Theoret. Comput. Sci.* **8**, 65–82, 2006.

[15] S. Olariu, Weak bipolarizable graphs, *Discrete Math.* **74**, 159–171, 1989.

[16] D.J. Rose, R.E. Tarjan, and G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* **5**, 266–283, 1976.

Appendix

Proofs of Lemma 3.3 and Theorem 3.1

Lemma 3.3 *Let G be a graph, v a vertex of G, G_v the auxiliary graph described earlier in terms of G and v, and Z_i $(i = 1, 2, \ldots, \ell)$ the sets of vertices of G_v defined in Eqn. (1). Suppose that there exist vertices $z_A \in Z_i$ and $z_{A'} \in Z_j$ of G_v corresponding to connected components A, A' of subgraphs $G[S_i]$ and $G[S_j]$, respectively, where $i < j$, such that $z_A z_{A'} \in E(G_v)$ and there exists a vertex $x \in \left(\bigcup_{r=j+1}^{\ell} Z_r\right) \cup N(v)$ such that $x \in N_{G_v}(z_A) - N_{G_v}(z_{A'})$. Then if Algorithm Recognize-HHD-free is run on G, it reports that G is not HHD-free and stops.*

Proof: Suppose that Algorithm Recognize-HHD-free is run on G. If the algorithm stops at any time before reaching Step 2, then it has reported that G is not HHD-free, as desired. If it does not stop, it eventually processes the sets of the partition $\mathcal{P}_{Z_i}$, and in particular the set P_A to which z_A belongs, computes the set $X'_A = N_{G_v}(z_C) \cap \left(\bigcup_{r=i+1}^{\ell} Z_r\right)$ where z_C is the selected vertex from P_A, and checks that all the vertices in P_A are adjacent to precisely X'_A; if this is not so, then the algorithm reports that G is not HHD-free, otherwise, we have that $N_{G_v}(z_A) \cap \left(\bigcup_{r=i+1}^{\ell} Z_r\right) = N_{G_v}(z_C) \cap \left(\bigcup_{r=i+1}^{\ell} Z_r\right)$. Thus, if $x \in \bigcup_{r=j+1}^{\ell} Z_r$, then $x \in X'_A$ because $x \in N_{G_v}(z_A)$. Since $z_{A'} \in N_{G_v}(z_A) \cap \left(\bigcup_{r=i+1}^{\ell} Z_r\right)$, it follows that $z_{A'} \in X'_A$, that is, the set X'_A is not empty, and the algorithm finds the minimum-index set S_k such that X'_A contains a vertex corresponding to a component of the subgraph $G[S_k]$, computes the set W_A of vertices in X'_A that correspond to components of $G[S_k]$, computes $X_A = X'_A \cup \big(N_{G_v}(z_C) \cap N(v)\big) = X'_A \cup \big(N_{G_v}(z_A) \cap N(v)\big)$, selects a vertex z_B in W_A, and concatenates the set $X_A - W_A$ to $A(z_B)$. Then, $k \leq j$, and $x \in A(z_B)$ because $x \in X_A - W_A$; if $x \in \bigcup_{r=j+1}^{\ell} Z_r$ then $x \in X'_A$, and $x \notin W_A$ since $k \leq j$, whereas if $x \in N(v)$ then $x \in N_{G_v}(z_A) \cap N(v)$.

Suppose that $k < j$. Then, $z_{A'}$ also belongs to $A(z_B)$. If Algorithm Recognize-HHD-free does not stop early, it eventually processes the set P_B of the partition $\mathcal{P}_{Z_k}$ to which z_B belongs, and checks whether the set $\left(\bigcup_{z \in P_B} A(z)\right) - N_{G_v}(z') \neq \emptyset$, where z' is the vertex selected from P_B. Since $z_{A'}, x \in A(z_B)$, we have that $z_{A'}, x \in \bigcup_{z \in P_B} A(z)$. If there exists a vertex in P_B that is not adjacent to $z_{A'}$ or x, then the algorithm reports that the input graph G is not HHD-free and stops. Otherwise, we have that the vertex $z_B \in Z_k$ is adjacent to both $z_{A'}$ and x in the graph G_v, and the conditions for the vertices $z_B, z_{A'}, x$ are identical to the conditions described in the statement of this lemma for $z_A, z_{A'}, x$. Moreover, since $k > i$ and the difference $j - i$ is finite, this case may occur a finite number of times. Eventually, either Algorithm Recognize-HHD-free will stop prematurely, or $k = j$ and Algorithm Recognize-HHD-free processes the set $P_{A'}$ of the partition $\mathcal{P}_{Z_j}$ to which $z_{A'}$ belongs. In the latter case, let $z_{C'}$ be the vertex selected from $P_{A'}$ during the processing of $P_{A'}$. If $x \in N_{G_v}(z_{C'})$ then the algorithm stops since $x \in N_{G_v}(z_{C'}) \cap \left(\bigcup_{r=j+1}^{\ell} Z_r\right)$ and $x \notin N_{G_v}(z_{A'})$. If $x \notin N_{G_v}(z_{C'})$, the algorithm again stops because $\left(\bigcup_{z \in P_{A'}} A(z)\right) - N_{G_v}(z_{C'}) \neq \emptyset$ (recall that $x \in A(z_B)$ and note that $z_B \in P_{A'}$ where z_B is the vertex that was selected during the processing of the set P_A containing z_A so that the vertices in $X_A - W_A$ are concatenated to $A(z_B)$); thus, $x \in A(z_B)$ and $x \notin N_{G_v}(z_{C'})$, and Algorithm Recognize-HHD-free reports that the input graph G is not HHD-free and stops. ∎

Theorem 3.1 *When Algorithm Recognize-HHD-free is run on a graph G, it reports that G is not HHD-free if and only if G is indeed not HHD-free.*

Proof: ($\Longrightarrow$) Suppose that the algorithm prints that G is not HHD-free while processing vertex v. This may happen either in Step 1.2 or in Step 1.4: In Step 1.2, it happens only if there exist two non-neighbors x, y of v such that $xy \in E(G)$, x, y belong to different sets of the partition $\mathcal{S}_v$, and

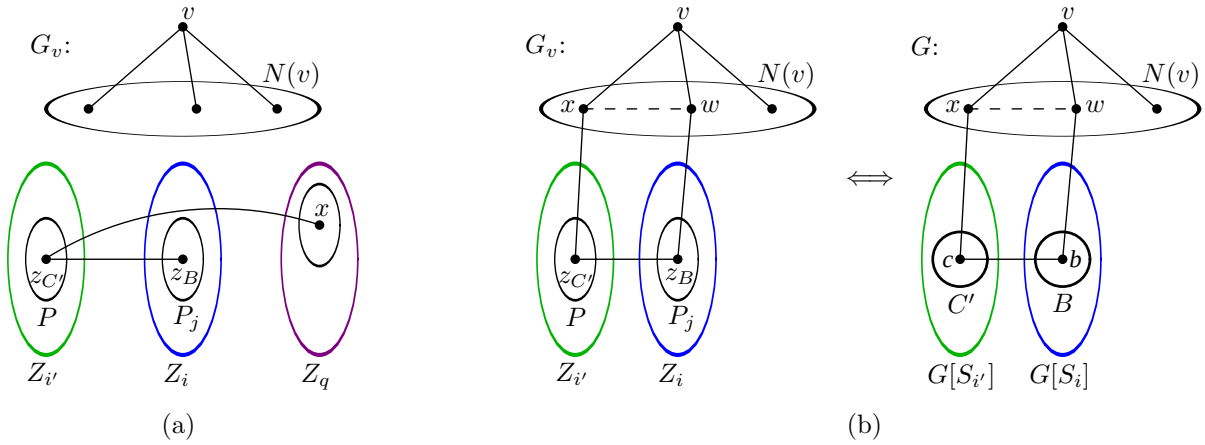

Figure 7: For the proof of Theorem 3.1.

$|N(x) \cap N(v)| = |N(y) \cap N(v)|$; but then, in accordance with Observation 3.1, G contains an induced house or C_5. In Step 1.4, the algorithm may print that G is not HHD-free in two places:

- if there exists a vertex, say, $z_{C'} \in P_j$, that is not adjacent in the graph G_v to a vertex in X' or is adjacent to a vertex in $\left(\bigcup_{r=i+1}^{\ell} Z_r\right) - X'$: The set P_j, which contains z_C and $z_{C'}$, has been formed by unions of disjoint subsets of the set Z_i of vertices of G_v (corresponding to connected components of the subgraph $G[S_i]$) whenever there exists a vertex, say, z (corresponding to a component of a $G[S_{i'}]$, where $i' < i$), such that z is adjacent in G_v to elements of these subsets. Thus, for z_C and $z_{C'}$ to end up in the same set P_j given that they initially belonged to different sets, there is a sequence of vertices $z_{A_1}, z_{A_2}, \ldots, z_{A_h} \in \bigcup_{r=1}^{i-1} Z_r$ such that $z_C \in N_{G_v}(z_{A_1})$, $z_{C'} \in N_{G_v}(z_{A_h})$, and for each $p = 1, 2, \ldots, h$, $|N_{G_v}(z_{A_p}) \cap Z_i| \geq 2$ and $\left(N_{G_v}(z_{A_p}) \cap \left(\bigcup_{t=1}^{p-1} N_{G_v}(z_{A_t})\right)\right) \cap Z_i \neq \emptyset$). Then, Corollary 3.2 (statement (ii)) applies and implies that if G was HHD-free, the vertices z_C and $z_{C'}$ would have the exact same neighbors in $\left(\bigcup_{r=i+1}^{\ell} Z_r\right) \cup N(v)$; as this is not the case, the graph G is not HHD-free.

- if there exists a vertex $x \in \left(\bigcup_{z \in P_j} A(z)\right) - N_{G_v}(z_C)$, where z_C is the vertex selected from the set P_j (of vertices corresponding to connected components of the subgraph $G[S_i]$) that is currently being processed: The vertex x belongs to $\bigcup_{z \in P_j} A(z)$ because it has been added to some $A(z_B)$, where $z_B \in P_j$; this addition was done earlier in Step 1.4 for a value i' of the index of the for-loop (i.e. $i' < i$) while processing a subset P of the set $Z_{i'}$ of vertices of G_v that correspond to components of $G[S_{i'}]$; let $z_{C'}$ be the vertex selected from P during its processing. Then, $z_{C'} z_B \in E(G_v)$ and $x z_{C'} \in E(G_v)$. Since $\bigcup_{z \in P_j} A(z) \subseteq \left(\bigcup_{r=i+1}^{\ell} Z_r\right) \cup N(v)$, we distinguish the following two cases:
  If $x \in \bigcup_{r=i+1}^{\ell} Z_r$, then $x \in Z_q$ where $q > i$ (see Figure 7(a)), and if G was HHD-free, Corollary 3.1 would apply to the vertices $z_{C'} \in Z_{i'}$, $z_B \in Z_i$, and $x \in Z_q$ where $i' < i < q$, and would imply that $x z_B \in E(G_v)$; this leads to a contradiction since $x \notin N_{G_v}(z_C)$, and the vertices z_B, z_C belong to P_j and have the same neighbors in $\bigcup_{r=i+1}^{\ell} Z_r$ (otherwise, the algorithm would have stopped). Thus, G is not HHD-free.
  Now, suppose that $x \in N(v)$ (see Figure 7(b)); then, because $x z_{C'} \in E(G_v)$ and $x z_C \notin E(G_v)$, vertex x is adjacent in G to all the vertices in $S_{i'}$ and to no vertex in S_i. Since $z_{C'} z_B \in E(G_v)$, there exist vertices $c \in C' \subseteq S_{i'}$ and $b \in B \subseteq S_i$ such that $bc \in E(G)$. Moreover, since $i' < i$, there exists a vertex $w \in N(v)$ such that $w \in N(b) - N(c)$. Then, the vertices v, x, w, b, c induce a house or a C_5 in G.

($\Longleftarrow$) Now, suppose that the graph G is not HHD-free; we will show that Algorithm Recognize-HHD-free will report that.

Suppose that the vertices v, u, w, x, y form a cycle $vuxyw$ in G and induce a C_5 or a house with v at its top. If Algorithm Recognize-HHD-free does not stop early (if it does so, it correctly reports that the graph G is not HHD-free), it eventually executes the body of the for-loop of Step 1 for the vertex v. The vertex adjacencies in the above mentioned house or C_5 imply that the vertices x, y do not belong to the same partition set of the partition $\mathcal{S}_v$. Suppose that x and y belong to the connected components A and B of the subgraphs $G[S_i]$ and $G[S_j]$, respectively; moreover, without loss of generality, suppose that $i < j$. These facts imply that $z_A z_B \in E(G_v)$ and that $u \in \big(N_{G_v}(z_A) - N_{G_v}(z_B)\big) \cap N(v)$; then, Lemma 3.3 applies asserting that the algorithm will report that the input graph G is not HHD-free and will stop.

Suppose that G contains a hole $vua_1a_2 \cdots a_h w$, where $h \geq 2$. Again, if Algorithm Recognize-HHD-free does not stop early (and reports that the graph G is not HHD-free), it eventually executes the body of the for-loop of Step 1 for the vertex v. Note that the vertices $a_1, a_2, \ldots, a_h$ are all non-neighbors of v; let $A_1, A_2, \ldots, A_h$, respectively, be the connected components of the subgraphs of G induced by the partition sets of the partition $\mathcal{S}_v$ to which the vertices $a_1, a_2, \ldots, a_h$ belong (note that the components A_1 and A_h differ from each other and from all other components, whereas $A_2, A_3, \ldots, A_{h-1}$ are not necessarily distinct). Then, the graph G_v contains vertices $z_{A_1}, z_{A_2}, \ldots, z_{A_h}$ corresponding to the above connected components, and because G contains the path $a_1a_2 \cdots a_h$, the subgraph $G_v[\{z_{A_1}, z_{A_2}, \ldots, z_{A_h}\}]$ is connected; let ρ be a chordless path in this subgraph from z_{A_1} to z_{A_h}. Because none of the vertices $a_2, \ldots, a_{h-1}$ are adjacent to either u or w, the vertices v, u, w, and the vertices of the path ρ form a hole in G_v; let it be $vz_1z_2 \cdots z_t$, where $z_1 = u$, $z_t = w$, and $t \geq 4$; note that no two consecutive vertices in the subpath $z_2 \cdots z_{t-1}$ correspond to connected components of the same subgraph $G[S_i]$ since no two such vertices are adjacent in G_v. If $t = 4$, then the graph G contains a C_5 and as proved earlier, Algorithm Recognize-HHD-free reports that G is not HHD-free and stops. Now suppose that $t > 4$. Let j, k be such that the vertices z_2, z_3 correspond to connected components of the subgraphs $G[S_j]$ and $G[S_k]$, respectively; since z_2, z_3 are adjacent, clearly $j \neq k$; moreover, if $j < k$, then Lemma 3.3 applies to the edge z_2z_3 of G_v and because $u \in \big(N_{G_v}(z_2) - N_{G_v}(z_3)\big) \cap N(v)$, it implies that the algorithm will report that the input graph G is not HHD-free and will stop. If $k < j$, then there exist vertices z_i $(3 \leq i \leq t-2)$ such that the vertices z_{i-1}, z_i, z_{i+1} correspond to connected components of the subgraphs $G[S_p], G[S_j], G[S_q]$, respectively, where $j < p$ and $j < q$. For any such vertex z_i for which $p \neq q$, we can use Lemma 3.3: if, without loss of generality, we assume that $p < q$, then $z_iz_{i-1} \in E(G_v)$ and $z_{i+1} \in \big(N_{G_v}(z_i) - N_{G_v}(z_{i-1})\big) \cap \bigcup_{r=j+1}^{\ell} Z_r$. The only remaining case is if for each such vertex z_i, we have that $p = q$. Then, there exists such a vertex z_i such that not both z_{i-2} and z_{i+2} belong to $\bigcup_{r=1}^{p-1} Z_r$. If not, then all the vertices z_i, $i = 2, 3, \ldots, t-1$, would correspond to connected components of subgraphs $G[S_r]$ with $r \leq \min\{j, j'\}$, where j, j' are such that the vertices z_2 and z_{t-1} correspond to connected components of $G[S_j]$ and $G[S_{j'}]$, respectively; this, however, contradicts the fact that $\max\{j, j'\} > \min\{j, j'\}$ since $j \neq j'$ due to the adjacencies in the hole $vz_1z_2 \cdots z_t$. So, suppose without loss of generality that $z_{i-2} \in \big(\bigcup_{r=p+1}^{\ell} Z_r\big) \cup N(v)$; note that $z_{i-2} \notin Z_p$ because $z_{i-1} \in Z_p$ and z_{i-1}, z_{i-2} are adjacent in G_v. Then, when Algorithm Recognize-HHD-free processes the set of the partition $\mathcal{P}_{Z_p}$ containing both z_{i-1} and z_{i+1}, it finds that z_{i-2} is adjacent to z_{i-1} but not to z_{i+1}; then, Algorithm Recognize-HHD-free reports that G is not HHD-free and stops.

Suppose that G contains a domino D induced by the cycle $vudefw$ with a single chord uf (i.e., v is a corner vertex of D). Again, if Algorithm Recognize-HHD-free does not stop early (in which case, it reports that the input graph G is not HHD-free), it will eventually execute the body of the for-loop of Step 1 for the vertex v. The vertex adjacencies in the domino D imply that the vertices d, e, f belong to distinct partition sets of the partition $\mathcal{S}_v$; let D, E, F be the connected components of

the subgraphs $G[S_h], G[S_i], G[S_j]$ to which d, e, f belong, respectively, where $h \neq i$, $h \neq j$, and $i \neq j$. Because $de \in E(G)$ and $ef \in E(G)$, then $\{z_D z_E, z_E z_F\} \subseteq E(G_v)$. If $z_D z_F \in E(G_v)$, then there exist vertices $x \in D$ and $y \in E$ such that $xy \in E(G)$; but then, the vertices v, u, x, y, w induce a house in G with x at its top, and as proved earlier for the case of an induced house or C_5, the algorithm will report that the graph G is not HHD-free and will stop. So let us assume that $z_D z_F \notin E(G_v)$. If $h < i$, Lemma 3.3 applies to the edge $z_D z_E$ of G_v, and because $u \in \big(N_{G_v}(z_D) - N_{G_v}(z_E)\big) \cap N(v)$, it implies that the algorithm will report that G is not HHD-free and will stop. The same conclusion is obtained in a similar fashion if $j < i$; note that $z_F z_E \in E(G_v)$ and $u, w \in \big(N_{G_v}(z_F) - N_{G_v}(z_E)\big) \cap N(v)$. Thus, $i < h$ and $i < j$. If additionally $h < j$, then Lemma 3.3 applies to the edge $z_E z_D$ of G_v, and because $z_F \in \big(N_{G_v}(z_E) - N_{G_v}(z_D)\big) \cap \bigcup_{r=h+1}^{\ell} Z_r$, it implies that the algorithm will report that G is not HHD-free and will stop; the same conclusion is also obtained if $j < h$ for the edge $z_E z_F$ because $z_D \in \big(N_{G_v}(z_E) - N_{G_v}(z_F)\big) \cap \bigcup_{r=j+1}^{\ell} Z_r$. ∎