*algorithms*

MDPI

*Article*

# A Quasi-Hole Detection Algorithm for Recognizing $k$-Distance-Hereditary Graphs, with $k < 2$

**Serafino Cicerone**

Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, I-67100 L'Aquila, Italy; serafino.cicerone@univaq.it

**Abstract:** Cicerone and Di Stefano defined and studied the class of $k$-distance-hereditary graphs, i.e., graphs where the distance in each connected induced subgraph is at most $k$ times the distance in the whole graph. The defined graphs represent a generalization of the well known distance-hereditary graphs, which actually correspond to 1-distance-hereditary graphs. In this paper we make a step forward in the study of these new graphs by providing characterizations for the class of all the $k$-distance-hereditary graphs such that $k < 2$. The new characterizations are given in terms of both forbidden subgraphs and cycle-chord properties. Such results also lead to devise a polynomial-time recognition algorithm for this kind of graph that, according to the provided characterizations, simply detects the presence of quasi-holes in any given graph.

**Keywords:** distance-hereditary graphs; stretch number; recognition problem; forbidden subgraphs; hole detection



**Citation:** Cicerone, S. A Quasi-Hole Detection Algorithm for Recognizing $k$-Distance-Hereditary Graphs, with $k < 2$. *Algorithms* **2021**, *14*, 105. https://doi.org/10.3390/a14040105

Academic Editor: Frank Werner

Received: 10 February 2021
Accepted: 23 March 2021
Published: 25 March 2021



## 1. Introduction

Distance-hereditary graphs have been introduced by Howorka [1], and are defined as those graphs in which every connected induced subgraph is isometric; that is, the distance between any two vertices in the subgraph is equal to the one in the whole graph. Therefore, any connected induced subgraph of any distance-hereditary graph $G$ "inherits" its distance function from $G$. Formally:

**Definition 1** (from [1]). *A graph $G$ is a distance-hereditary graph if, for each connected induced subgraph $G'$ of $G$, the following holds: $d_{G'}(x,y) = d_G(x,y)$, for each $x, y \in G'$.*

This kind of graph have been rediscovered many times (e.g., see [2]). Since their introduction, dozens of papers have been devoted to them, and different kinds of characterizations have been found: metric, forbidden subgraphs, cycle/chord conditions, level/neighborhood conditions, generative, and more (e.g., see [3]). Among such results, the generative properties resulted as the most fruitful for algorithmic applications, since they allowed researchers to efficiently solve many combinatorial problems in the class of distance-hereditary graphs (e.g., see [4–9]).

From an applicative point of view, distance-hereditary graphs are mainly attractive due to their basic metric property. For instance, these graphs can model unreliable communication networks [10,11] in which vertex failures may occur: at a given time, if sender and receiver are still connected, any message can be still delivered without increasing the length of the path used to reach the receiver.

Since in communication networks this property could be considered too restrictive, in [12] the class of *k-distance-hereditary graphs* has been introduced. These graphs can model unreliable networks in which messages can eventually reach the destination traversing a path whose length is at most $k$ times the length of a shortest path computed in absence of vertex failures. The minimum $k$ a network guarantees regardless the failed vertices is called *stretch number*. Formally:

**Definition 2** (from [12]). *Given a real number $k \geq 1$, a graph $G$ is a* $k$-distance-hereditary graph *if, for each connected induced subgraph $G'$ of $G$, the following holds: $d_{G'}(x,y) \leq k \cdot d_G(x,y)$, for each $x, y \in G'$.*

The class of all the $k$-distance-hereditary graphs is denoted by $\mathrm{DH}(k)$. Concerning this class of graphs, the following relationships hold:

- $\mathrm{DH}(1)$ coincides with the class of distance-hereditary graphs;
- $\mathrm{DH}(k_1) \subseteq \mathrm{DH}(k_2)$, for each $k_1 \leq k_2$.

Additional results about the class hierarchy $\mathrm{DH}(k)$ can be found in [13,14]. It is worth to notice that this hierarchy is *fully general*; that is, for each arbitrary graph $G$ there exists a number $k$ such that $G \in \mathrm{DH}(k)$. It follows that the stretch number of $G$, denoted as $s(G)$, is the smallest number $t$ such that $G$ belongs to $\mathrm{DH}(t)$. In [12], it has been shown that the stretch number $s(G)$ of any connected graph $G$ can be computed as follows:

- the stretch number of any pair $\{u,v\}$ of distinct vertices is defined as $s_G(u,v) = D_G(u,v)/d_G(u,v)$, where $D_G(u,v)$ is the length of any longest induced path between $u$ and $v$, and $d_G(u,v)$ is the distance between the same pair of vertices;
- $s(G) = \max_{\{u,v\}} s_G(u,v)$.

It follows that for any non-trivial graph $G$ with $n \geq 4$ vertices, by simply maximizing $D(u,v)$ and minimizing $d(u,v)$, we get $s(G) \leq (n-2)/2$. From the above relationship about $s(G)$, we get that the stretch number is always a rational number. Interestingly, it has been shown that there are some rational numbers that cannot be stretch numbers. Formally, a positive rational number $t$ is called *admissible stretch number* if there exists a graph $G$ such that $s(G) = t$. The following result characterizes which numbers are admissible stretch numbers.

**Theorem 1** (from [14]). *A rational number $t$ is an admissible stretch number if and only if $t = 2 - \frac{1}{i}$, for some integer $i \geq 1$, or $t \geq 2$.*

Apart from the interesting general results found for the classes $\mathrm{DH}(k)$, the original motivation was studying how (if possible) to extend the known algorithmic results from the base class, namely $\mathrm{DH}(1)$, to $\mathrm{DH}(k)$ for some constant $k > 1$. According to Theorem 1, in this work we are interested in studying the class containing each graph $G$ such that $s(G) < 2$. Since this class contains graphs with stretch number *strictly* less than two, throughout this paper it will be denoted by $s\mathrm{DH}(2)$.

**Results.** In this work, we provide three results for the class $s\mathrm{DH}(2)$, namely two different characterizations and a recognition algorithm (notice that the characterizations have already been presented in [13] but with omitted proofs). The first characterization is based on listing all the minimal forbidden subgraphs for each graph in the class. It is interesting to observe the similarity with the corresponding result for the class $\mathrm{DH}(1)$:

- **(adapted from [2])** $G \in \mathrm{DH}(1)$ if and only if the following graphs are not induced subgraphs of $G$:
  - holes $H_n$, for each $n \geq 5$;
  - cycles $C_5$ with $cd(C_5) = 1$;
  - cycles $C_6$ with $cd(C_6) = 1$.
- **(this paper)** $G \in s\mathrm{DH}(2)$ if and only if the following graphs are not induced subgraphs of $G$:
  - holes $H_n$, for each $n \geq 6$;
  - cycles $C_6$ with $cd(C_6) = 1$;
  - cycles $C_7$ with $cd(C_7) = 1$;
  - cycles $C_8$ with $cd(C_8) = 1$.

Here we used the notion of "chord distance" $cd(C)$ to express the position of possible chords within any cycle $C$ (see Section 2 for a formal definition). Notice that in [14] a similar result has been provided for the generic class $\mathrm{DH}(2-\frac{1}{i})$, $i > 1$.

The second result is a characterization based on a cycle-chord property. As in the previous case, notice the similarity with the corresponding result for the class $\mathrm{DH}(1)$:

- **(from [12])** $G \in \mathrm{DH}(1)$ if and only if $cd(C_n) > 1$ for each cycle $C_n$, $n \geq 5$, of $G$;
- **(this paper)** $G \in s\mathrm{DH}(2)$ if and only if $cd(C_n) > 1$ for each cycle $C_n$, $n \geq 6$, of $G$.

The last result is a recognition algorithm for graphs belonging to $s\mathrm{DH}(2)$ that works in $O(n^2m^2)$ time and $O(m^2)$ space. Basically, this algorithm exploits the result based on the cycle-chord property and, as a consequence, simply detects *quasi-holes* in any graph. A quasi-hole is any cycle with at least five vertices and chord-distance at most one (i.e., all the possible chords of the cycle must be incident to the same vertex). This algorithm is obtained by adapting the algorithm provided in [15] for detecting holes (i.e., any cycle with at least five vertices and no chords).

**Outline.** The paper is organized as follows. In Section 2, we introduce notation and basic concepts used throughout the paper. Sections 3 and 4 are devoted to providing the characterization based on minimal forbidden subgraphs and cycle-chord conditions for graphs in $s\mathrm{DH}(2)$, respectively. In Section 5, we provide the algorithm for detecting quasi-holes and hence to solve the recognition problem for the class $s\mathrm{DH}(2)$. Finally, Section 6 provides some concluding remarks.

## 2. Notation and Basic Concepts

We consider finite, simple, loop-less, undirected, and unweighted graphs $G = (V, E)$ with vertex set $V$ and edge set $E$. A *subgraph* of $G$ is a graph having all its vertices and edges in $G$. Given $S \subseteq V$, the *induced subgraph* $G[S]$ of $G$ is the maximal subgraph of $G$ with vertex set $S$. Given $u \in V$, $N_G(u)$ denotes the set of *neighbors* of $u$ in $G$, and $N_G[u] = N_G(u) \cup \{u\}$.

A sequence of pairwise distinct vertices $(x_0, x_1, \ldots, x_k)$ is a *path* in $G$ if $(x_i, x_{i+1}) \in E$ for $0 \leq i < k$; vertex $x_i$, for each $0 < i < k$, is an *internal vertex* of that path. A *chord* of a path is any edge joining two non-consecutive vertices in the path, and a path is an *induced path* if it has no chords. We denote by $P_k$ any induced path with $k \geq 3$ vertices (e.g., an induced path on three vertices is denoted as $P_3$ whereas an induced path on four vertices is denoted as $P_4$). Two vertices $x$ and $y$ are *connected* in $G$ if there exists a path $(x, \ldots, y)$ in $G$. A graph is *connected* if every pair of vertices is connected.

A *cycle* in $G$ is a path $(x_0, x_1, \ldots, x_{k-1})$ where also $(x_0, x_{k-1}) \in E$. Two vertices $x_i$ and $x_j$ are *consecutive* in the cycle $(x_0, x_1, \ldots, x_{k-1})$ if $j = (i+1) \bmod k$ or $i = (j+1) \bmod k$. A *chord* of a cycle is an edge joining two non-consecutive vertices in the cycle. We denote by $C_k$ any cycle with $k \geq 3$ vertices, whereas $H_k$ denotes a *hole*, i.e., a cycle $C_k$, $k \geq 5$, without chords. The *chord distance* of a cycle $C_k$ is denoted by $cd(C_k)$ and is defined as the minimum number of consecutive vertices in $C_k$ such that every chord of $C_k$ is incident to some of such vertices (see Figure 1 for an example of chord distance). We assume $cd(H_k) = 0$.

The length of any shortest path between two vertices $x$ and $y$ in a graph $G$ is called *distance* and is denoted by $d_G(x, y)$. Moreover, the length of any longest induced path between them is denoted by $D_G(x, y)$. If $x$ and $y$ are distinct vertices, we use the symbols $p_G(x, y)$ and $P_G(x, y)$ to denote any shortest and any longest induced path between $x$ and $y$, respectively. Sometimes, when no ambiguity occurs, we also use $p_G(x, y)$ and $P_G(x, y)$ to denote the sets of vertices belonging to the corresponding paths. If $d_G(x, y) \geq 2$, then $\{x, y\}$ is a *cycle-pair* if there exist two induced paths $p_G(x, y)$ and $P_G(x, y)$ such that $p_G(x, y) \cap P_G(x, y) = \{x, y\}$. In other words, if $\{x, y\}$ is a cycle-pair, then there exist induced paths $p_G(x, y)$ and $P_G(x, y)$ such that the vertices in $p_G(x, y) \cup P_G(x, y)$ form a cycle in $G$; this cycle is denoted by $G[x, y]$. In Figure 1 $\{v_3, v_6\}$ is a cycle-pair that induces the cycle $(v_3, v_4, v_5, v_6, v_1)$; in particular, $G[v_3, v_6]$ is induced by $p_G(v_3, v_6) = (v_3, v_1, v_6)$ and $P_G(v_3, v_6) = (v_3, v_4, v_5, v_6)$. We use the symbol $\mathcal{S}(G)$ to denote the set containing all pairs $\{u, v\}$ of connected vertices that induce the stretch number of $G$, namely $\mathcal{S}(G) =$

$\{\{x,y\} : s_G(x,y) = s(G)\}$. The following lemma states that cycle-pairs are useful to determine the stretch number.

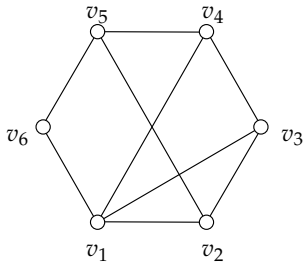


**Figure 1.** The chord distance of this $C_6$ graph is two because: (i) vertices $v_1$ and $v_2$ are consecutive in the cycle, (ii) every chord is incident to one of such vertices, and (iii) there is no other set with less than two vertices with the same properties.

**Lemma 1** (from [12]). *Let $G$ be a graph such that $s(G) > 1$. The following relationships hold:*

*(i) $d_G(u,v) \geq 2$ for each pair $\{u,v\}$ such that $\{u,v\} \in \mathcal{S}(G)$,*
*(ii) there exists a cycle-pair $\{u,v\}$ that induces the stretch number of $G$, that is $\{u,v\} \in \mathcal{S}(G)$.*

This lemma suggests that studying $s(G)$ concerns the analysis of cycles in $G$. In particular, if $\{u,v\}$ is a cycle-pair that belongs to $\mathcal{S}(G)$, then the cycle $G[u,v]$ is called *inducing-stretch cycle* for $G$. In Figure 1, the represented graph $G$ belongs to DH(3/2); moreover, both $\{v_3,v_5\}$ and $\{v_3,v_6\}$ are cycle-pairs in $\mathcal{S}(G)$, and $(v_1,v_3,v_4,v_5,v_6)$ is the corresponding inducing-stretch cycle.

## 3. A Characterization Based on Forbidden Subgraphs

A well known characterization based on *minimal forbidden subgraphs* has been provided for the class of distance-hereditary graphs.

**Theorem 2** (from [2]). *A graph $G$ is a distance-hereditary graph if and only if it does not contain, as an induced subgraph, any of the following graphs: the hole $H_n$, $n \geq 5$, the house, the fan, and the domino (cf. Figure 2).*

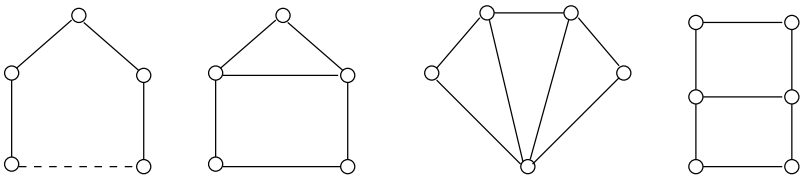

**Figure 2.** The minimal forbidden subgraphs of distance-hereditary graphs: from left to right, the *hole*, the *house*, the *fan*, and the *domino*. Dashed lines represent paths of length at least one.

This result can be easily reformulated, and simplified, by using the notion of chord distance. In particular, it is possible to characterize in a compact way all the forbidden subgraphs by using just the notion of chord distance as follows:

- *$G$ is a distance-hereditary graph if and only if the following graphs are not induced subgraphs of $G$:*
  *(i) $H_n$, for each $n \geq 5$;*
  *(ii) cycles $C_5$ with $cd(C_5) = 1$;*
  *(iii) cycles $C_6$ with $cd(C_6) = 1$.*

It is worth to notice that in this way we do not consider the minimal subgraphs only (cf. Figure 3).

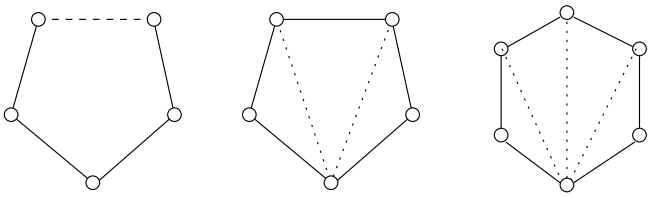

**Figure 3.** The forbidden subgraphs of DH(1) expressed according to the notion of chord distance. Dashed lines represent paths of length at least one. Dotted lines represent chords that may or may not exist.

In the following we provide a characterization similar to that of Theorem 2 for any graph $G \in s\mathrm{DH}(2)$. Before giving such a result, we need to recall the following technical lemma.

**Lemma 2.** *Let $G$ be a graph and let $G[x,y]$ be an inducing-stretch cycle of $G$ defined by the induced paths $P_G(x,y) = (x, u_1, u_2, \ldots, u_{p-1}, y)$ and $p_G(x,y) = (x, v_1, v_2, \ldots, v_{q-1}, y)$. If $d(x,y) \geq 3$ then $v_1$ must be incident to chords of the cycle $G[x,y]$.*

**Proof.** Since $G[x,y]$ is an inducing-stretch cycle of $G$, then $s(G) = \frac{p}{q}$. If $v_1$ is not incident to any chords of $G[x,y]$, then the induced paths $P_G(v_1,y) = (v_1, x, u_1, u_2, \ldots, u_{p-1}, y)$ and $p_G(v_1,y) = (v_1, v_2, \ldots, v_{q-1}, y)$ imply $s_G(v_1,y) = \frac{p+1}{q-1} > \frac{p}{q}$, a contradiction. □

Let $G$ be any graph. According to Lemma 1, let us consider an inducing-stretch cycle $G[x,y]$ of $G$. Assume that $G[x,y]$ is formed by the vertices of the induced paths $P_G(x,y) = (x, u_1, u_2, \ldots, u_{p-1}, y)$ and $p_G(x,y) = (x, v_1, v_2, \ldots, v_{q-1}, y)$. Since $P_G(x,y)$ and $p_G(x,y)$ are induced paths, each chord of $G[x,y]$ (if any) joins vertices $v_i$ and $u_j$, with $1 \leq i \leq q-1$ and $1 \leq j \leq p-1$. When some vertex $v_i$ is incident to chords of $G[x,y]$, we denote by $(v_i, u_{l_i})$ and $(v_i, u_{r_i})$ the *leftmost* and *rightmost* chords of $v_i$, respectively. Formally, the indices $l_i$ and $r_i$ are defined as follows:

- $l_i = \min\{i' \mid 1 \leq i' \leq p-1 \text{ and } (v_i, u_{i'}) \text{ is a chord of } G[x,y]\}$
- $r_i = \max\{i' \mid 1 \leq i' \leq p-1 \text{ and } (v_i, u_{i'}) \text{ is a chord of } G[x,y]\}$

**Theorem 3.** *Let $G$ be a graph. $G \in s\mathrm{DH}(2)$ if and only if the following graphs are not induced subgraphs of $G$:*

*(i)* *$H_n$, for each $n \geq 6$;*
*(ii)* *cycles $C_6$ with $cd(C_6) = 1$;*
*(iii)* *cycles $C_7$ with $cd(C_7) = 1$;*
*(iv)* *cycles $C_8$ with $cd(C_8) = 1$.*

**Proof.** ($\Rightarrow$) Each provided hole and cycle has stretch number greater or equal to 2, and hence it cannot be an induced subgraph of $G$.

($\Leftarrow$) We prove that if $s(G) \geq 2$, then $G$ contains one of the subgraphs in items $(i)$, $(ii)$, $(iii)$, or $(iv)$, or $G$ contains a proper induced subgraph $G'$ such that $s(G') \geq 2$. In the latter case, we can recursively apply to $G'$ the following proof.

According to Lemma 1, consider an inducing-stretch cycle $G[x,y]$ of $G$ and assume it is formed by the vertices of the induced paths $P_G(x,y) = (x, u_1, u_2, \ldots, u_{p-1}, y)$ and $p_G(x,y) = (x, v_1, v_2, \ldots, v_{q-1}, y)$. Notice that, since $P_G(x,y)$ and $p_G(x,y)$ are induced paths, each possible chord of $G[x,y]$ joins vertices $v_i$ and $u_j$, with $1 \leq i \leq q-1$ and $1 \leq j \leq p-1$.

Since $\frac{p}{q} \geq 2$ by hypotheses, then $q \geq 2$ by Item $(i)$ of Lemma 1, and hence $p \geq 4$. According to the value of $q$, we analyze two different cases:

$q = 2$: In this case, if $G[x,y]$ is chordless, then it corresponds to a hole as described in Item $(i)$. If the chord distance of $G[x,y]$ is equal to 1, all chords are incident to $v_1$. According to $p$, we have:

$p = 4$: $G[x, y]$ corresponds to the cycle in Item $(ii)$;

$p = 5$: $G[x, y]$ corresponds to the cycle in Item $(iii)$;

$p = 6$: $G[x, y]$ corresponds to the cycle in Item $(iv)$;

$p \geq 7$: Let $(v_1, u_{l_1})$ be the leftmost chord of $v_1$. If $l_1 \geq 4$ the cycle $(v_1, x, u_1, u_2, \ldots, u_{l_1})$ corresponds to the cycle in Item $(i)$. When $l_1 \leq 3$, consider the subgraph $G'$ induced by the vertices in the cycle $(v_1, u_{l_1}, u_{l_1+1}, \ldots, u_{p-1}, y)$. The induced paths $P' = (u_{l_1}, u_{l_1+1}, \ldots, u_{p-1}, y)$ and $p' = (u_{l_1}, v_1, y)$ provide the following lower bound for $s_{G'}$:

$$s_{G'}(u_{l_1}, y) \geq \frac{p-l_1}{2} \geq \frac{7-3}{2} = 2.$$

Hence, $G'$ is a proper subgraph of $G$ with $s(G') \geq 2$. The statement follows by recursively applying to $G'$ this proof.

$q \geq 3$: In this case, according to Lemma 2, $v_1$ must be incident to chords. We now analyze two cases with respect to the value of $r_1$, $(v_1, u_{r_1})$ being the rightmost chord of $v_1$:

$r_1 \geq 4$: Consider the subgraph $G''$ induced by the vertices in the cycle $(v_1, x, u_1, u_2, \ldots, u_{r_1})$. In this case, the induced paths $P'' = (x, u_1, u_2, \ldots, u_{r_1})$ and $p'' = (x, v_1, u_{r_1})$ provide the following lower bound for $s_{G''}$: $s_{G''}(x, u_{r_1}) \geq r_1/2 \geq 2$. Hence, $G''$ is a proper subgraph of $G$ with $s(G'') \geq 2$. The statement follows by recursively applying to $G''$ this proof.

$r_1 \leq 3$: in this case the induced paths $P''' = (v_1, u_{r_1}, u_{r_1+1}, \ldots, u_{p-1}, y)$ and $p''' = (v_1, v_2, \ldots, v_{q-1}, y)$ provide the following lower bound for $s_G(v_1, y)$:

$$s_G(v_1, y) \geq \frac{p-2}{q-1}.$$

Since $\frac{p-2}{q-1} \geq \frac{p}{q}$ is equivalent to $\frac{p}{q} \geq 2$ (which holds by hypothesis), then the subgraph $G'''$ induced by the vertices in both $P'''$ and $p'''$ is a proper subgraph of $G$ with stretch $p^*/q^* \geq 2$ and $q^* = q - 1$. Hence, the statement follows by recursively applying to $G'''$ this proof.

This concludes the proof. □

Figures 3 and 4 summarize the characterizations based on forbidden subgraphs for classes DH(1) and $s$DH(2), respectively. Figure 5 provides the list of all the *minimal* forbidden subgraphs of any graph in $s$DH(2).

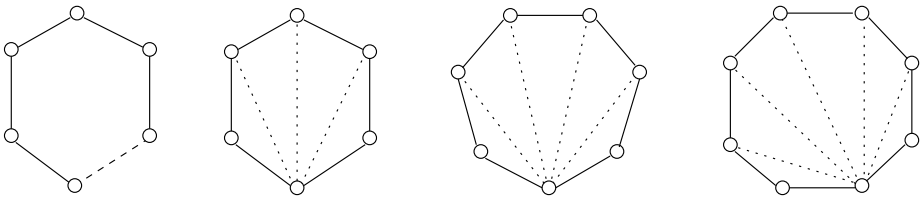

**Figure 4.** The forbidden subgraphs of graphs having stretch number less than 2. Dashed (dotted, respectively) lines represent paths of length at least one (chords that may or may not exist, respectively).

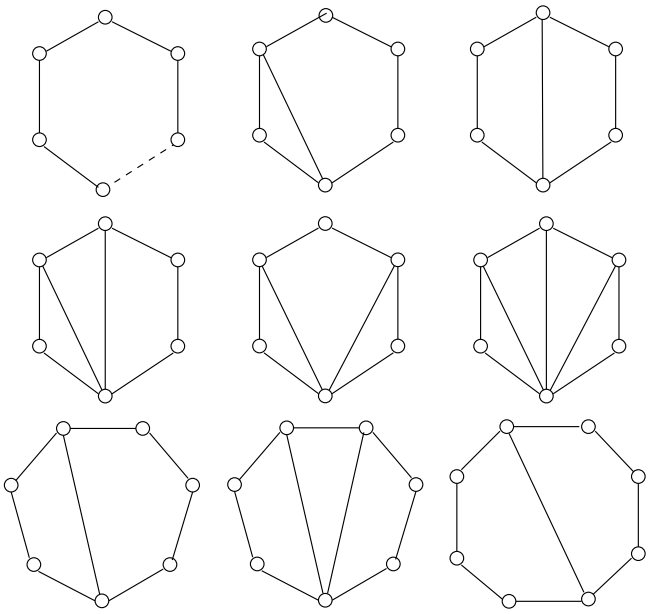

**Figure 5.** The *minimal* forbidden subgraphs of any graph with stretch number less than 2. Dashed lines represent paths of length at least one.

## 4. A Characterization Based on Cycle-Chord Conditions

For the class of distance-hereditary graphs, Howorka provided the following well known characterization based on cycle-chord conditions.

**Theorem 4** (from [1]). *Let $G$ be a graph. $G \in \mathrm{DH}(1)$ if and only if each cycle $C_n$, $n \geq 5$, of $G$ has two crossing chords.*

In [12], this result has been reformulated in terms of chord distance:

**Theorem 5** (from [12]). *Let $G$ be a graph. $G \in \mathrm{DH}(1)$ if and only if $cd(C_n) > 1$ for each cycle $C_n$, $n \geq 5$, of $G$.*

In the remainder of this section, we provide a similar characterization for graphs belonging to $s\mathrm{DH}(2)$.

**Lemma 3.** *Let $G$ be a graph. If $s(G) = 2$ then $G$ contains, as induced subgraph, a cycle $C_6$ with chord distance at most 1.*

**Proof.** According to Lemma 1, consider an inducing-stretch cycle $G[x, y]$ of $G$. Since $s(G) = 2$, assume that $G[x, y]$ is formed by the vertices of the induced paths $P_G(x, y) = (x, u_1, u_2, \ldots, u_{2s-1}, y)$ and $p_G(x, y) = (x, v_1, v_2, \ldots, v_{s-1}, y)$, with $s \geq 2$.

If $s = 2$ then the proof is concluded. In fact, cycle $G[x, y]$ has 6 vertices and every chord of $G[x, y]$ (if any) is incident to $v_1$.

In the remainder of the proof assume $s \geq 3$. In this case, according to Lemma 2, $v_1$ is incident to chords of $G[x, y]$. Let $(v_1, u_{r_1})$ be the rightmost chord incident to $v_1$. We analyze different cases according to the value of $r_1$.

- Assume $r_1 > 4$. In this case, the induced paths $(x, u_1, u_2, \ldots, u_{r_1})$ and $(x, v_1, u_{r_1})$ provide a stretch number $s_G(x, u_{r_1}) \geq \frac{r_1}{2} > 2$, a contradiction.
- Assume $r_1 \leq 2$. In this case, the induced paths $(v_1, u_{r_1}, u_{r_1+1}, \ldots, u_{2s-1}, y)$ and $(v_1, v_2, \ldots, v_{s-1}, y)$ provide the following lower bound on $s_G(v_1, y)$:

$$s_G(v_1, y) \geq \frac{2s - r_1 + 1}{s - 1} \geq \frac{2s - 2 + 1}{s - 1} = 2 + \frac{1}{s - 1}.$$

This contradicts $s(G) = 2$.

It follows that either $r_1 = 4$ or $r_1 = 3$. In the first case the cycle $(v_1, x, u_1, u_2, u_3, u_4)$ represents the requested cycle $C_6$: chords of $G[x, y]$ (if any) are all incident to $v_1$. In the second case consider the induced paths $(v_1, u_{r_1}, u_{r_1+1}, \ldots, u_{2s-1}, y)$ and $(v_1, v_2, \ldots, v_{s-1}, y)$. These paths induce the following lower bound on $s_G(v_1, y)$:

$$s_G(v_1, y) \geq \frac{2s - r_1 + 1}{s - 1} = \frac{2s - 3 + 1}{s - 1} = 2.$$

Hence, the above paths induce a proper subgraph $G'$ of $G$ with stretch number 2. Hence, this proof can be recursively applied to $G'$. □

**Lemma 4.** *Let $G$ be a graph. $s(G) \geq 2$ if and only if $G$ contains, as an induced subgraph, a cycle $C_n$, $n \geq 6$, with chord distance at most 1.*

**Proof.** ($\Leftarrow$) Trivial.

($\Rightarrow$) If $s(G) = 2$, then it is sufficient to use Lemma 3. Now, let us assume that $s(G) = p/q > 2$ such that $p$ and $q$ are coprime. By Lemma 1, if $G[x, y]$ is an inducing-stretch cycle of $G$, according to the hypotheses, we may assume that $G[x, y]$ is formed by the vertices of the induced paths $P_G(x, y) = (x, u_1, u_2, \ldots, u_{p \cdot s-1}, y)$ and $p_G(x, y) = (x, v_1, v_2, \ldots, v_{q \cdot s-1}, y)$, with $s \geq 1$.

If $d(x, y) = 2$, then $G[x, y]$ contains at least 6 vertices and all its chords (if any) are incident to $v_1$. Then, $G[x, y]$ corresponds to the requested cycle.

In the remainder, assume that $d(x, y) \geq 3$. In this case, by Lemma 2, vertex $v_1$ is incident to chords of $G[x, y]$: let $(v_1, u_{r_1})$ be the rightmost chord incident to it.

If $r_1 \leq 3$, then the two induced paths $(v_1, u_{r_1}, u_{r_1+1}, \ldots, u_{p \cdot s-1}, y)$ and $(v_1, v_2, \ldots, v_{q \cdot s-1}, y)$ provide the following lower bound for $s_G(v_1, y)$:

$$s_G(v_1, y) \geq \frac{p \cdot s - r_1 + 1}{q \cdot s - 1}.$$

Now we show that

$$\frac{p \cdot s - r_1 + 1}{q \cdot s - 1} > \frac{p}{q}. \tag{1}$$

It can be easily observed that Equation (1) is equivalent to

$$\frac{p}{q} > r_1 - 1. \tag{2}$$

Since $r_1 \leq 3$ and $p/q > 2$ by hypothesis, then Equation (2) holds. This implies that $s_G(v_1, y) > p/q$, a contradiction.

Then, it follows that $r_1 \geq 4$. In this case, $C = (x, u_1, u_2, \ldots, u_{r_1}, v_1)$ is an induced cycle with $r_1 + 2 \geq 6$ vertices and chord distance at most 1 (In $C$, all the possible chords are incident to $v_1$). This concludes the proof. □

This lemma can be reformulated so that it directly provides a characterization for the graphs under consideration.

**Theorem 6.** *Let $G$ be a graph. $G \in s\mathrm{DH}(2)$ if and only if $cd(C_n) > 1$ for each cycle $C_n$, $n \geq 6$, of $G$.*

Compare Theorems 5 and 6 to observe the similarity between the cycle-chord characterizations of graphs with stretch number equal to 1 and graphs with stretch number less than 2, respectively.

## 5. Recognition Algorithm

The distance-hereditary graphs, i.e., graphs in DH(1), can be recognized in linear time [16], while the recognition problem for the generic class DH($k$), $k$ not fixed, is co-NP-complete [12]. For small and fixed values of $k$, in [14] a partial answer to this basic problem is given. In particular, Lemma 1 states that for $k < 2$, only specific rational numbers may act as stretch numbers. In [14], a characterization for each class DH($2 - 1/i$), $i > 1$, has been provided, and such a characterization led to a polynomial time algorithm for the recognition problem for the class DH($2 - 1/i$), with fixed $i > 1$. Unfortunately, the running time of this algorithm is bounded by $O(n^{3i+2})$.

In this section, we propose a polynomial-time algorithm for solving the recognition problem for the class $s$DH(2) according to the following approach. Lemma 4 provides a characterization for all graphs not belonging to $s$DH(2). It is based on detecting whether a given graph $G$ contains or not an induced cycle $C_n$, $n \geq 6$, with chord distance at most 1. Now, assume that we have an algorithm $\mathcal{A}$ returning *true* if and only if a given graph $G$ contains such a cycle. Then, to recognize whether $G \in s$DH(2) we can simply use $\mathcal{A}$ on $G$ and certify the membership if and only if $\mathcal{A}$ return *false*. In the remainder of this section we show that such an algorithm $\mathcal{A}$ can be defined.

### 5.1. An Existing Hole Detection Algorithm

We remind that $H_k$ denotes a hole, i.e., a chordless cycle with $k \geq 5$ vertices. In [15], Nikolopoulos and Palios provided the following result about the hole detection problem.

**Theorem 7** (from [15]). *Given any connected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, it is possible to determine whether $G$ contains a hole in $O(m^2)$ time and $O(nm)$ space.*

They also extended their result to larger versions of holes.

**Corollary 1** (from [15]). *Let $G = (V, E)$ be a connected graph with $|V| = n$ and $|E| = m$, and let $k \geq 5$ be a constant. It is possible to determine whether $G$ contains a hole on at least $k$ vertices in $O(nm^{p-1})$ time and $O(m^{p-1})$ space if $k = 2p$, and in $O(n + m^p)$ time and $O(nm^{p-1})$ space if $k = 2p + 1$.*

Therefore, according to this corollary, it is possible to check whether $G$ contains a hole $H_k$, with $k \geq 6$ vertices, in $O(nm^2)$ time and $O(m^2)$ space.

### 5.2. Quasi-Hole Detection Algorithm

We call *quasi-hole* any cycle $C_k$ such that $k \geq 5$ and $cd(C_k) \leq 1$. In what follows, we show that the hole-detection algorithms recalled in Theorem 7 and Corollary 1 can be adapted to detect quasi-holes in any connected graph $G$. This adapted version is called `QuasiHoleDetection` and it is described in pseudo-code as shown in Algorithms 1 and 2. The strategy behind `QuasiHoleDetection` is based on the following result:

**Lemma 5.** *A connected graph $G$ contains a quasi-hole if and only if there exists a cycle $(v_0, v_1, \ldots, v_\ell)$, $\ell \geq 4$, in $G$ such that each path $(v_i, v_{i+1}, v_{i+2}, v_{i+3})$, $i = 1, \ldots, \ell - 3$, is a $P_4$ of $G$.*

**Proof.** ($\Rightarrow$) If $G$ contains a quasi-hole $C_k$ then the vertices of $C_k$ form a cycle fulfilling the conditions of the statement (where $v_0$ is the only vertex incident to possible chords of the cycle).

($\Leftarrow$) Suppose that $G$ admits cycles as described in the statement, and let $C = (v_0, v_1, \ldots, v_\ell)$ be *the shortest* among such cycles. We now show that $(i)$ $C$ has at least 5 vertices and $(ii)$ $cd(C) \leq 1$:

$(i)$ Since $C$ fulfills the conditions of the statement, then $C$ contains at least 5 vertices;

$(ii)$ Suppose by contradiction that $cd(C) > 1$. Then, there must exist chords $(v_i, v_j)$ with both $v_i$ and $v_j$ different from $v_0$. To each chord $(v_i, v_j)$ not incident on $v_0$, we associate

a "length" defined as $length(v_i, v_j) = |j - i|$. Now, let $(v_l, v_r)$, with $l < r$, be a chord with minimum length. By definition, $0 < l < r \leq \ell$ holds. Since $(v_l, v_{l+1}, v_{l+2}, v_{l+3})$ is a $P_4$, then $r \geq l + 4$, and hence $C' = (v_l, v_{l+1}, \ldots, v_r)$ results to be a cycle with at least 5 vertices. Moreover, between $v_i$ and $v_j$, for each $l \leq i < i + 2 \leq j \leq r$, $(i, j) \neq (l, r)$, cannot exist an edge, otherwise it would be a chord with length smaller than $length(v_l, v_r)$.

Since $C'$ is a cycle with at least 5 vertices and with chord distance zero, then it contradicts the fact that $C$ is the shortest among the cycles fulfilling the conditions of the statement. Hence, $cd(C) \leq 1$.

Since both the properties at points $(i)$ and $(ii)$ hold, it follows that $C$ is a quasi-hole. □

**Algorithm 1:** A quasi-hole detection algorithm.

**Algorithm:** `QuasiHoleDetection`
**Input** : a connected undirected graph $G = (V, E)$
**Output:** "true" if $G$ contains a quasi-hole, "false" otherwise.

```
1  calculate the adjacency matrix M[ ] of G ;
2  foreach v1 ∈ V do
3      set each entry of the arrays walked_P3[ ] and AP[ ] to 0;
4      base ← v1;
5      AP[v1] ← 1;
6      foreach (v2, v3) ∈ E do
7          if (v1, v2) ∈ E and v1 ≠ v3 then
8              AP[v2] ← 1;
9              Visit(base, v1, v2, v3);
10             AP[v2] ← 0;
11         end
12         if (v1, v3) ∈ E and v1 ≠ v2 then
13             AP[v3] ← 1;
14             Visit(base, v1, v3, v2);
15             AP[v3] ← 0;
16         end
17     end
18     AP[v1] ← 0;
19 end
20 print "false".
```

The above lemma is used by the provided algorithm for the detection of quasi-holes in $G$. To this end, we associate to $G$ a directed graph $G^+$ defined as follows:

- $\{v_{abc} \mid (a, b, c)$ *is a* $P_3$ *in the graph* $G\}$ is the vertex set of $G^+$;
- $\{(v_{abc}, v_{bcd}) \mid (a, b, c, d)$ *is a* $P_4$ *in the graph* $G\}$ is the edge set of $G^+$.

If $(a, b, c)$ is a path $P_3$ of $G$, then both the vertices $v_{abc}$ and $v_{cba}$ belong to $G^+$. In a similar way, if $(a, b, c, d)$ is a path $P_4$ of $G$, then the edges $(v_{abc}, v_{bcd})$ and $(v_{dcb}, v_{cba})$ must be contained in $G^+$. Hence, visiting $G^+$ is equivalent to proceeding along $P_4$s of $G$. It follows that the conditions of Lemma 5 on $G$ can be verified by performing a revised DFS on $G^+$ (cf. [17]). In turn, the following lemma holds:

**Lemma 6.** *Let $G$ be any connected graph, and let $G^+$ be its associated directed graph. By performing a DFS on $G^+$, if the DFS-path is $v_{u_0u_1u_2}, v_{u_1u_2u_3}, \ldots, v_{u_{k-2}u_{k-1}u_k}$, where $u_i \neq u_j$ for each $0 \leq i < j < k$ and $u_k = u_\ell$ for some $\ell$ such that $0 \leq \ell < k$, then $u_\ell, u_{\ell+1}, \ldots, u_{k-1}$ are vertices forming a cycle in $G$ that fulfill Lemma 5. Conversely, if $G$ contains a quasi-hole, the DFS on $G^+$ will meet a sequence of vertices in $G^+$ whose corresponding $P_3$s in $G$ produce a path as the path $(v_1, v_2, \ldots, v_\ell)$ in the cycle as in Lemma 5.*

**Algorithm 2:** A recursive procedure used by `QuasiHoleDetection` to perform an adapted DFS.

**Procedure:** procedure `Visit`
**Input** : four vertices *base*, $u_1$, $u_2$, and $u_3$ of $G$

```
1  AP[u_3] ← 1;
2  walked_P_3[(u_1, u_2), u_3] ← 1;
3  foreach (u_3, u_4) ∈ E \ {(u_3, u_2)} do
4      if u_4 = base then
5          if AP.size ≥ 5 then
               // the active path determines a quasi-hole
6              print "true" ;
7              exit;
8          else
9              break
10         end
11     else
12         if (u_2, u_4) ∉ E and (u_1 = base or (u_1, u_4) ∉ E) then
               // here, when u_1 ≠ base, (u_1, u_2, u_3, u_4) forms a P_4 in G
13             if AP[u_4] = 1 then
                   // the active path determines a hole
14                 print "true" ;
15                 exit;
16             end
17             if walked_P_3[(u_2, u_3), u_4] = 0 then
18                 Visit(base, u_2, u_3, u_4);
19             end
20         end
21     end
22 end
23 AP[u_3] ← 0;
```

By following the same strategy used in [15], to reduce the space complexity required by $G^+$, the DFS on $G^+$ is simulated by performing a revised DFS directly on $G$. This revised DFS on $G$ is implemented by Algorithm `QuasiHoleDetection` (cf. Figure 1).

At Line 1, the algorithm computes the adjacency matrix $M[\,]$ of $G$ from its adjacency-list (we assume that $G$ is provided as input according to this representation). $M[\,]$ is used to check the adjacency in constant time. At Line 2, each vertex $v_1$ of $G$ is checked against the following possible role: $v_1$ belongs to a quasi-hole $C$ and all the chords of $C$, if any, are adjacent to $v_1$. To perform this check, at Line 6 we consider each edge $(v_2, v_3)$ in $G$: if this edge, along with $(v_1, v_2)$ (cf. Line 7) or $(v_1, v_3)$ (cf. Line 12), form a path with three vertices, then the algorithm tries to extend this path into the requested cycle by recursively calling the Procedure `Visit` (see Algorithm 2).

`Visit` works according to Lemma 5: in any step, it attempts to extend a path $P_3$ defined by $(u_1, u_2, u_3)$ into $P_4$s of the form $(u_1, u_2, u_3, u_4)$; then, for each such $P_4$, the procedure proceeds by extending the $P_3$ formed by $(u_2, u_3, u_4)$ into $P_4$s of the form $(u_2, u_3, u_4, u_5)$, and so on. In this situation, the *active-path* is first extended from $(u_1, u_2, u_3)$ to $(u_1, u_2, u_3, u_4)$, then to $(u_1, u_2, u_3, u_4, u_5)$ and so on. In case of backtracking, the last vertex is removed of the current active-path. By proceeding in this way, two cases may occur:

- the initial vertex $v_1$ (called *base* in the algorithm) is added again to the active-path (cf. Line 4). If the length of the active-path is 5 or more (cf. Line 5), then the graph contains a cycle fulfilling the conditions of Lemma 5 and hence a quasi-hole is found;

- at the end of the active-path there is a vertex different from *base* but already inserted in the active-path (cf. Lines 12–13). In this case, again the conditions of Lemma 5 apply, but now we are sure that a hole is found.

It is worth to remark that the ongoing active-path $P$ on $G$ and the ongoing DFS-path $P^+$ on $G^+$ contain exactly the same vertices: the elements of $P$ correspond to the vertices of the $P_3$s associated with the elements of $P^+$ (in $P$, the repeated vertices of $G$ in adjacent $P_3$s are present only once).

We now explain the role of the additional data structures $AP[\cdot]$ and $walked\_P_3[(\cdot,\cdot),\cdot]$. The former is an auxiliary array of size $n$ used to check if a vertex appears in the "active path" computed so far; given $u$, $AP[u]$ is equal to 1 if $u$ appears in the active path, 0 otherwise. Concerning the latter, during the visit on $G^+$, vertices that correspond to path $P_3$s of $G$ are recorded so that they are not "visited" again. The entry $walked\_P_3[(u_1,u_2),u_3]$ equals one if and only if the vertices $u_1,u_2,u_3$ induce $(u_1,u_2,u_3)$ as a path $P_3$ of $G$ already encountered during the DFS, otherwise it equals zero. Since $walked\_P_3[(\cdot,\cdot),\cdot]$ has entries $walked\_P_3[(u_1,u_2),u_3]$ and $walked\_P_3[(u_2,u_1),u_3]$ for each edge $(u_1,u_2) \in E$ and for each $u_3 \in V$, then its size is $2m \cdot n$. Notice that `Visit` registers the entry of $walked\_P_3[\,]$ at the beginning, thus avoiding another execution on the same path $P_3$. In this way, `Visit`() is executed exactly once for each path $P_3$ of $G$.

Notice that the description of `Visit`() assures that starting from a $P_3$ formed by $(u_1,u_2,u_3)$ we proceed to a $P_3$ formed by $(u_2,u_3,u_4)$ only if $(u_1,u_2,u_3,u_4)$ is a path $P_4$ of $G$. The only exception is when $u_1$ coincides with the starting vertex $v_1$ selected at Line 2 by `QuasiHoleDetection`: in such a case $(u_1,u_2,u_3,u_4)$ may have chords from $u_1$. For this purpose, the initial vertex $v_1$ is assigned to the variable *base* (cf. Line 4 of the main algorithm) and it is later passed to `Visit` (cf. Lines 9 and 14 of the main algorithm).

We can now provide the following statement:

**Theorem 8.** *Given any connected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, it is possible to determine whether $G$ contains a quasi-hole in $O(nm^2)$ time and $O(nm)$ space.*

**Proof.** According to the above description of `QuasiHoleDetection`, its correctness follows from Lemmas 5 and 6, and from the inherent execution of DFS on $G^+$. In the remainder of the proof we analyze the complexity of the algorithm about the required time and space.

As $G$ is a connected graph, we get $n = O(m)$. Concerning the data structures used by the algorithm, we assume that from any edge $(v_1,v_2)$ it is possible to access in constant time both its endpoints; alike, from any entry in the adjacency matrix $M[\,]$ of $G$ corresponding to $v_1$ and $v_2$ it is possible to access in constant time the edge $(v_1,v_2)$.

Consider first the time complexity of performing the revised DFS of $G$. The visit starts at Line 6, and proceeds by recursive calls to `Visit`. This recursive procedure checks each path $(u_1,u_2,u_3)$ of $G$ which is a $P_3$ and tries to extend it into a $P_4$ of the form $(u_1,u_2,u_3,u_4)$. Notice that each set of vertices $\{u_1,u_2,u_3,u_4\}$ where $(u_1,u_2,u_3)$ is a $P_3$ and $u_4$ is adjacent to $u_3$ is uniquely characterized by the ordered pair $((u_1,u_2),(u_3,u_4))$ where $(u_1,u_2)$ and $(u_3,u_4)$ are ordered pairs of adjacent vertices in $G$. Hence, the time required to perform the whole visit according to the recursive executions of `Visit` is $O(m^2)$. We can now determine the time complexity of `QuasiHoleDetection`. Step at Line 1 clearly takes $O(n^2)$ time. The subsequent loop at Line 2 is repeated $O(n)$ times, and for each step the algorithm requires $O(nm)$ time for the initialization at Line 3 and, as described before, $O(m^2)$ time for visiting $G$ according to the recursive calls to `Visit`.

It follows that the final time complexity is $O(nm^2)$. The algorithm requires $O(nm)$ space: $O(n)$ and $O(nm)$ for the arrays $AP[\,]$ and $walked\_P_3[\,]$, respectively, and $O(n^2)$ for the adjacency matrix $M[\,]$ and the adjacency-list used to represent $G$. □

### *5.3. Detecting Quasi-Hole on at Least k Vertices*

As in [15], the strategy described above to define a quasi-hole detection algorithm can be generalized to built algorithms for the detection of quasi-holes on at least $k$ ver-

tices, with $k \geq 5$. For any input graph $G$, we consider the following family of directed graphs $G^{(t)}$:

- $\{v_{u_1u_2\cdots u_{t-1}} \mid (u_1, u_2, \ldots, u_{t-1})$ *is an induced path* $P_{t-1}$ in $G\}$ is the vertex set of $G^{(t)}$,
- $\{(v_{u_1u_2\cdots u_{t-1}}, v_{u_2u_3\cdots u_t}) \mid (u_1, u_2, \ldots, u_t)$ *is an induced path* $P_t$ in $G\}$ is the edge set of $G^{(t)}$.

By definition, $G \equiv G^{(2)}$ and $G^+ \equiv G^{(4)}$ where $G^+$ is the direct graph associated to $G$ in Section 5.2. Therefore, in the same way that running DFS on $G^+ \equiv G^{(4)}$ allowed us to detect quasi-holes (on at least five vertices), running DFS on $G^{(k-1)}$ allows us to detect (extended) quasi-holes on at least $k$ vertices, for each constant $k \geq 5$. This is ensured by the following statement, which represents a generalization of Lemma 5:

**Lemma 7.** *Given a constant $k \geq 5$, a graph $G$ contains a quasi-hole on at least $k$ vertices if and only if $G$ contains a cycle $(u_0, u_1, \ldots, u_t)$, with $t \geq k-1$, such that $(u_i, u_{i+1}, \ldots, u_{i+k-2})$ is an induced path $P_{k-1}$ of $G$ for each $i = 1, 2, \ldots, t-k+2$.*

Lemmas 6 and 7 induce the following statement:

**Corollary 2.** *Let $G$ be a connected graph and let $k \geq 5$ be a constant. Assume that a DFS is executed on $G^{(k-1)}$, the directed graph associated to $G$. If the active path computed by the DFS is $v_{u_0u_1\cdots u_{k-3}}, v_{u_1u_2\cdots u_{k-2}}, \ldots, v_{u_{r-k+3}u_{r-k+4}\cdots u_r}$, where $u_i \neq u_j$ for all $0 \leq i < j < r$, and $u_r = u_p$ for some $p$ such that $0 \leq p < r$, then $u_p, u_{p+1}, \ldots, u_{r-1}$ are vertices forming a cycle in $G$ that fulfill the conditions of Lemma 7. Conversely, if $G$ contains a quasi-hole on at least $k$ vertices, the DFS on $G^{(k-1)}$ will meet a sequence of vertices whose associated $P_{k-2}$s in $G$ form a path as the path $(u_1, u_2, \ldots, u_t)$ in the cycle of Lemma 7.*

Additionally, in this situation we do not build $G^{(k-1)}$ since we implicitly run DFS on this associated graph. In particular, we process each unvisited $P_{k-2}$ of $G$ as follows: we try to extend the induced path $P_{k-2}$ formed by $(u_0, u_1, \ldots, u_{k-3})$ into $P_{k-1}$s of the form $(u_0, u_1, \ldots, u_{k-3}, u_{k-2})$; then, for each such $P_{k-1}$, we proceed by extending the $P_{k-2}$ $(u_1, u_2, \ldots, u_{k-2})$ into $P_{k-1}$s, and so on. Since there exist $O(m^a)$ induced paths on $2a$ vertices and $O(nm^a)$ on $2a+1$ vertices, and it requires $O(k)$ time to detect whether a vertex extends a $P_{k-1}$ into a $P_k$, we have the following corollary:

**Corollary 3.** *Let $G = (V, E)$ be a connected graph with $|V| = n$ and $|E| = m$, and let $k \geq 5$ be a constant. By implicitly running DFS on $G^{(k-1)}$ it is possible to detect whether $G$ contains a quasi-hole on at least $k$ vertices in $O(n^2m^{p-1})$ time when $k = 2p$, and in $O(n^2 + nm^p)$ time when $k = 2p+1$.*

The space required is $O(m^{p-1})$ when $k = 2p$, and $O(nm^{p-1})$ when $k = 2p+1$. According to Lemma 4 and Corollary 3, we finally get the following result:

**Theorem 9.** *Let $G = (V, E)$ be a connected graph with $|V| = n$ and $|E| = m$. It is possible to recognize whether $G \in s\mathrm{DH}(2)$ in $O(n^2m^2)$ time and $O(m^2)$ space.*

## 6. Conclusions

In this paper, we studied the class $s\mathrm{DH}(2)$. It contains each graph $G$ with stretch number less than two, that is $s(G) < 2$. These graphs form a superclass of the well studied distance-hereditary graphs, which corresponds to graphs with stretch number equal to one.

For the class $s\mathrm{DH}(2)$ we provided: (1) a characterization based on listing all the minimal forbidden subgraphs, (2) a characterization based on cycle-chord properties, and (3) a recognition algorithm that works in $O(n^2m^2)$ time and $O(m^2)$ space. This algorithm exploits the result based on the cycle-chord property to detects quasi-holes in a graph; it is a simple adaptation of the algorithm provided in [15] for detecting holes.

The characterizations found seem to suggest that the graphs in $s\mathrm{DH}(2)$ and those in $\mathrm{DH}(1)$ may be really similar in structure and hence properties. As a consequence, it would be interesting to determine whether the class $s\mathrm{DH}(2)$ can be also characterized according to generative operations (we remind that the generative properties resulted as the most fruitful for devising efficient algorithms for distance-hereditary graphs). This problem has been partially addressed in [18,19].

On the contrary, Theorem 1 could suggest that graphs with stretch number greater or equal to two may have a completely different structure with respect to those in $\mathrm{DH}(1)$.

Another possible extension of this work could be to investigate in the class $s\mathrm{DH}(2)$ other specific combinatorial problems that have been solved in the class of distance-hereditary graphs.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Howorka, E. Distance-hereditary graphs. *Q. J. Math.* **1977**, *28*, 417–420. [CrossRef]
2. Bandelt, H.J.; Mulder, H.M. Distance-hereditary graphs. *J. Comb. Theory Ser. B* **1986**, *41*, 182–208. [CrossRef]
3. Brandstädt, A.; Le, V.B.; Spinrad, J.P. *Graph Classes: A Survey*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1999.
4. Brandstädt, A.; Dragan, F.F. A linear-time algorithm for connected r-domination and Steiner tree on distance-hereditary graphs. *Networks* **1998**, *31*, 177–182. [CrossRef]
5. Chang, M.S.; Hsieh, S.Y.; Chen, G.H. Dynamic Programming on Distance-Hereditary Graphs. In *International Symposium on Algorithms and Computation*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1350, pp. 344–353.
6. Gioan, E.; Paul, C. Dynamic distance hereditary graphs using split decomposition. In *International Symposium on Algorithms and Computation*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4835, pp. 41–51.
7. Lin, C.; Ku, K.; Hsu, C. Paired-Domination Problem on Distance-Hereditary Graphs. *Algorithmica* **2020**, *82*, 2809–2840. [CrossRef]
8. Nicolai, F.; Szymczak, T. Homogeneous sets and domination: A linear time algorithm for distance-hereditary graphs. *Networks* **2001**, *37*, 117–128. [CrossRef]
9. Rao, M. Clique-width of graphs defined by one-vertex extensions. *Discret. Math.* **2008**, *308*, 6157–6165. [CrossRef]
10. Cicerone, S.; Di Stefano, G.; Flammini, M. Compact-Port Routing Models and Applications to Distance-Hereditary Graphs. *J. Parallel Distrib. Comput.* **2001**, *61*, 1472–1488. [CrossRef]
11. Esfahanian, A.H.; Oellermann, O.R. Distance-hereditary graphs and multidestination message-routing in multicomputers. *J. Comb. Math. Comb. Comput.* **1993**, *13*, 213–222.
12. Cicerone, S.; Di Stefano, G. Graphs with bounded induced distance. *Discret. Appl. Math.* **2001**, *108*, 3–21. [CrossRef]
13. Cicerone, S. Characterizations of Graphs with Stretch Number less than 2. *Electron. Notes Discret. Math.* **2011**, *37*, 375–380. [CrossRef]
14. Cicerone, S.; Di Stefano, G. Networks with small stretch number. *J. Discret. Algorithms* **2004**, *2*, 383–405. [CrossRef]
15. Nikolopoulos, S.D.; Palios, L. Detecting Holes and Antiholes in Graphs. *Algorithmica* **2007**, *47*, 119–138. [CrossRef]
16. Hammer, P.L.; Maffray, F. Completely separable graphs. *Discret. Appl. Math.* **1990**, *27*, 85–99. [CrossRef]
17. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 2nd ed.; The MIT Press and McGraw-Hill Book Company: New York, NY, USA, 2001.
18. Cicerone, S. Using Split Composition to Extend Distance-Hereditary Graphs in a Generative Way—(Extended Abstract). In *International Conference on Theory and Applications of Models of Computation*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6648, pp. 286–297. [CrossRef]
19. Cicerone, S. On Building Networks with Limited Stretch Factor. In *Web, Artificial Intelligence and Network Applications, Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications, AINA, Caserta, Italy, 15–17 Aprli 2020*; Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2020; Volume 1150, pp. 926–936. [CrossRef]