Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Algorithms 57 (2005) 1–21
Journal of Algorithms
www.elsevier.com/locate/jalgor

Estimating all pairs shortest paths in restricted graph families: a unified approach ☆

Feodor F. Dragan

Department of Computer Science, Kent State University, Kent, OH 44242, USA

Received 20 July 2002

Available online 5 November 2004

Abstract

In this paper we show that a very simple and efficient approach can be used to solve the *all pairs almost shortest path* problem on the class of weakly chordal graphs and its different subclasses. Moreover, this approach works well also on graphs with small size of largest induced cycle and gives a unified way to solve the *all pairs almost shortest path* and *all pairs shortest path* problems on different graph classes including chordal, strongly chordal, chordal bipartite, and distance-hereditary graphs.

1. Introduction

Let $G = (V, E)$ be a finite, unweighted, undirected, connected and simple (i.e., without loops and multiple edges) graph. Let also $|V| = n$ and $|E| = m$. The *distance* $d(v, u)$ between vertices v and u is the smallest number of edges in a path connecting v and u. The *all pairs shortest path problem* is to compute $d(v, u)$ for all pairs of vertices v and u in G.

☆ Results of this paper were presented at the WG'01 conference [F.F. Dragan, Estimating all pairs shortest paths in restricted graph families: a unified approach (extended abstract), in: Proceedings of the 27th International Workshop "Graph-Theoretic Concepts in Computer Science" (WG'01), June 2001, in: Lecture Notes in Comput. Sci., vol. 2204, Springer, 2001, pp. 103–116].

E-mail address: dragan@cs.kent.edu (F.F. Dragan).

0196-6774/$ – see front matter
doi:10.1016/j.jalgor.2004.09.002

The all pairs shortest path (APSP) problem is one of the most fundamental graph problems. There has been a renewed interest in it recently (see [1,2,5,8,14,18,33,36–39,42–47] for general (unweighted, undirected) graphs, and [4,6,9,12,13,17,23,24,27,32,40,41] for special graph classes). For general graphs, the best result known is by Seidel [33], who showed that the APSP problem can be solved in $O(M(n)\log n)$ time where $M(n)$ denotes the time complexity for matrix multiplication involving small integers only. The currently best matrix multiplication algorithm is due to Coppersmith and Winograd [15] and has $O(n^{2.376})$ time bound. In contrast, the naive algorithm for APSP performs breadth-first searches from each vertex, and requires time $\Theta(nm)$ which is $\Theta(n^3)$ for dense graphs.

Given the fundamental nature of the APSP problem, it is important to consider the desirability of implementing the algorithms in practice. Unfortunately, fast matrix multiplication algorithms are far from being practical and suffer from large hidden constants in the running time bound. There is interest therefore in obtaining fast algorithms for the APSP problem that *do not use* fast matrix multiplication. The currently best *combinatorial* (i.e., in graph-theoretic terms) algorithm for the APSP problem is an $O(n^3/\log n)$ time algorithm obtained by Basch et al. [8]. This offers only a marginal improvement over the naive algorithm.

Since an algorithm for the APSP problem would yield an algorithm with similar time bound for Boolean matrix multiplication, obtaining a combinatorial $O(n^{3-\varepsilon})$ time algorithm for the APSP problem would be a major breakthrough. Nowadays, many researchers take the approach to consider the all pairs *almost* shortest path (APASP) problem in general graphs or to design (optimal) $O(n^2)$ time algorithms for special well structured graph classes (which are interesting from a practical point of view).

Awerbuch et al. [5] and Cohen [14] considered the problem of finding *stretch t all pairs paths*, where t is some fixed constant and a path is of stretch t if its length is at most t times the distance between the endvertices. They presented efficient algorithms for computing t-stretch paths for $t \geqslant 4$. A different approach was employed by Aingworth et al. [1]. They described a simple and elegant $O(n^{5/2}\sqrt{\log n})$ time algorithm for finding all distances in unweighted, undirected graphs with an *additive one-sided error* of at most 2. They also make the very important observation that the small distances, and not the long distances, are the hardest to approximate. Recently, Dor et al. [18] improved the result of Aingworth et al. [1] and obtained an $\widetilde{O}(\min\{n^{3/2}m^{1/2}, n^{7/3}\})$ time algorithm for finding all distances in unweighted, undirected graphs with an additive one-sided error of at most 2 (here, $\widetilde{O}(f)$ means $O(f\,\mathrm{polylog}\,n)$). A simple argument shows that computing all distances in G with an additive one-sided error of at most 1 is as hard as Boolean matrix multiplication (see [18]).

Besides, efficient algorithms have been developed for solving the APSP problem on some restricted classes of graphs. Optimal $O(n^2)$ time algorithms were proposed for interval graphs [4,27,32,41], circular arc graphs [4,41], permutation graphs [17], bipartite permutation graphs [12], strongly chordal graphs [6,17,23], chordal bipartite graphs [24], distance-hereditary graphs [17], and dually chordal graphs [9] (the definition of the various families of graphs will be presented in the next section). Optimal parallel algorithms for the APSP problem for some certain classes of graphs can be found in [12,17,23,40].

The problem of computing shortest path (respectively distance) between points avoiding obstacles in an environment is an important problem in computational geometry and

robot motion planning. The approach to the shortest path and other optimization problems in such metric spaces (e.g., in a simple rectilinear polygon endowed with some metric) taken by many researchers first finds a *visibility graph* of the metric space which contains a shortest path between any two given points. Two points are said to be *visible* to each other in the presence of obstacles if there exists a special (rectilinear, straight-line, etc.; depends on application) path between the two points that does not meet any of the obstacles. A visibility graph has vertices that correspond to the points on polygon and there exists an edge (link) between two vertices in the graph if and only if the points corresponding to them are visible to each other (in other formulation, an edge between two vertices exists if and only if the points corresponding to them are visible from a third point). The *link distance* [3] between any pair of points is the minimum number of links (edges) on a path between the corresponding vertices in the visibility graph. Motwani et al. [28,29] studied few classes of rectilinear polygons and showed that the "*staircase*" visibility graphs and the "*star*" visibility graphs corresponding to them are *permutation*, *chordal* or *weakly chordal*. More recently, Everett and Corneil [21] have shown that the visibility graphs of spiral polygons are *interval graphs*. Note that any interval graph is chordal, and any chordal graph as well as any permutation graph is weakly chordal [11].

The fact that the visibility graphs of some polygons are chordal, motivated researchers to look at distances in chordal graphs [10,13,23,40]. Han et al. showed in [23] that the APSP problem for a chordal graph G can be solved in $O(n^2)$ time if the *square* G^2 of G is already given. They also note that computing G^2 for chordal graphs is as hard as for general graphs (i.e., as hard as Boolean matrix multiplication), and present few subclasses of chordal graphs for which G^2 can be computed more efficiently. From results in [10] it follows that one can compute in $O(n^2)$ time all distances in chordal graphs with an additive error of at most 2. Even more, given a pair of vertices $x, y \in V$, after a simple linear time preprocessing, in only $O(1)$ time one can compute $d(x, y)$ with an additive error of at most 2. Sridhar et al. showed in [40] that using a more sophisticated linear time preprocessing of a chordal graph, the distance between any two vertices that is at most 1 greater than the shortest distance can be computed in $O(1)$ time. Unfortunately, methods used in [23,40] for chordal graphs use characteristic properties of those graphs and cannot be applied to more general weakly chordal graphs.

In this paper we investigate distances in weakly chordal graphs. We show that a very simple and efficient approach can be used to solve the APASP problem on the class of weakly chordal graphs and its different subclasses. Furthermore, we show that this approach works well also on graphs with small size of largest induced cycle and gives a unified way to solve the APASP and APSP problems on different graph classes including chordal, strongly chordal, chordal bipartite, and distance-hereditary graphs. Notice that any of these graph families can have a graph with a clique of size $\theta(n)$ [22,34]. Therefore, graphs from those families are generally not sparse and can contain as many as $\theta(n^2)$ edges. Thus, the naive algorithm for APSP, which performs breadth-first searches from each vertex, will result in a $\Theta(n^3)$ time algorithm.

The rest of this paper is organized as follows. We begin in Section 2 by presenting some definitions and a simple $O(n^2)$ time algorithm for computing all pairs almost shortest path distances in graphs with special vertex orderings. In Section 3, we prove that this algorithm

- computes all distances
  - in House-Hole-free graphs with an additive one-sided error of at most 2,
  - in House-Hole-Domino-free graphs (and hence in chordal graphs) with an additive one-sided error of at most 1, and
- solves the APSP problem for distance-hereditary graphs, chordal bipartite graphs, strongly chordal (and hence interval) graphs.

In that section we also show that the APSP problem for a House-Hole-free graph G can be solved in $O(n^2)$ time, if the square G^2 of G is given. All these classes are subclasses of the weakly chordal graphs. Finally, in Section 4 we consider graphs G with small size of largest induced cycle and show that our generic algorithm computes in $O(n^2)$ time all distances in G with an additive one-sided error of at most $k-1$, where k is the size of largest induced cycle in G. Hence, for weakly chordal graphs (even for Hole-free graphs), all distances can be computed in $O(n^2)$ time with an additive one-sided error of at most 3. For AT-free graphs, we show that all distances can be computed in $O(n^2)$ time with an additive

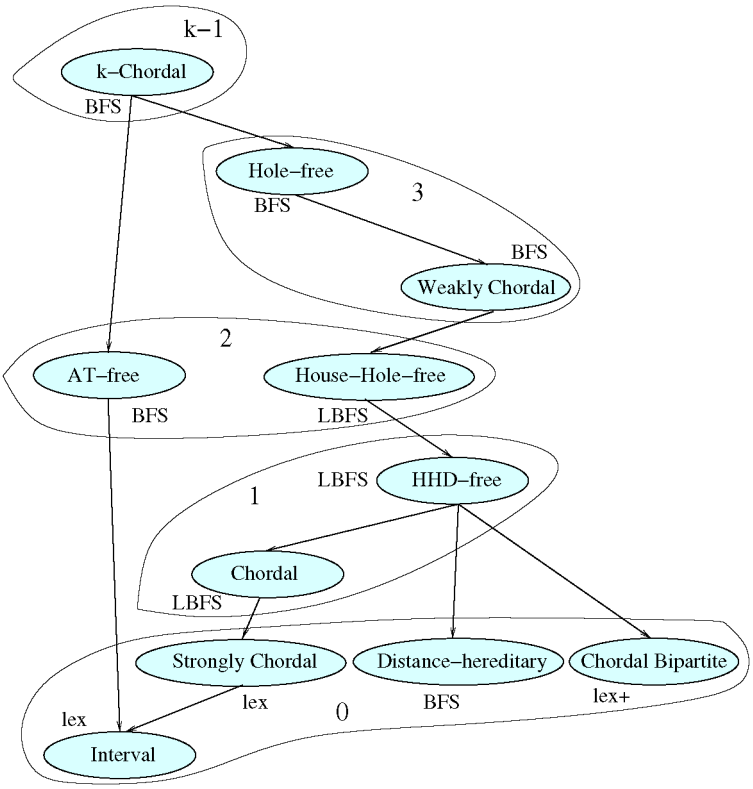

Fig. 1. The results of this paper and the hierarchy of the graph classes considered. Near each oval depicting a graph class we show which vertex ordering was used to achieve that bound on error. Graph classes are grouped with respect to the error in distance computations (shown by an integer). An arrow from class X to class Y indicates that Y is a subclass of X.

one-sided error of at most 2. Note that AT-free graphs include such well-known families of graphs as interval graphs, permutation graphs, trapezoid graphs and cocomparability graphs [16]. Figure 1 summarizes all results obtained in this paper.

2. Preliminaries and the algorithm

The (*open*) *neighborhood* of a vertex v is the set $N(v) = \{u \in V\colon uv \in E\}$ and the *closed neighborhood* is $N[v] = N(v) \cup \{v\}$. A *path* is a sequence of vertices $(v_0, \ldots, v_l)$ such that $v_i v_{i+1} \in E$ for $i = 0, \ldots, l-1$; its *length* is l. We say that this path connects vertices v_0 and v_l. An *induced path* is a path where $v_i v_j \in E$ if and only if $i = j-1$ and $j = 1, \ldots, l$. A *cycle* is a sequence of vertices $(v_0, \ldots, v_k)$ such that $v_0 = v_k$ and $v_i v_{i+1} \in E$ for $i = 0, \ldots, k-1$; its *length* is k. An *induced cycle* is a cycle where $v_i v_j \in E$ if and only if $|i-j| = 1$ (modulo k). A *hole* is an induced cycle of length at least 5.

We now define the various graph families that will be used in this paper. A graph G is *chordal* if there is no induced cycle of length greater than 3 in G. A *Hole-free* graph is a graph which contains no holes. A graph G is *weakly chordal* if both G and the complement of G are Hole-free graphs. A bipartite weakly chordal graph is called *chordal bipartite*. An *interval* graph is the intersection graph of intervals of a line (see Fig. 3 for an illustration). A graph is called *AT-free* if it does not have an *asteroidal triple*, i.e., a set of three vertices such that there is a path between any pair of them avoiding the closed neighborhood of the third. A graph is *House-Hole-free* (*HH-free*) if it contains no induced houses or holes and is *House-Hole-Domino-free* (*HHD-free*) if it contains no induced houses, holes or dominos (see Fig. 2). A *distance-hereditary graph* is a HHD-free graph which does not contain 3-fan as an induced subgraph. Finally, a graph $G = (V, E)$ is *strongly chordal* if it admits a *strong elimination ordering*, namely, an ordering $v_1, v_2, \ldots, v_n$ of V such that for each i, j, k and l, if $i < j$, $k < l$, $v_k, v_l \in N[v_i]$, and $v_k \in N[v_j]$, then $v_l \in N[v_j]$. For more information on these graph classes we refer the reader to [11,22].

For computing approximate distances in these graph classes we use the following simple $O(n^2)$ time algorithm. Let $G = (V, E)$ be a graph and $\sigma = [v_1, v_2, \ldots, v_n]$ be an ordering of the vertex set V of G. Let also $\sigma(v)$ be the number assigned to a vertex v in this ordering. Denote by $\mathrm{mn}(v)$ the *maximum neighbor* of v, i.e., the vertex of $N[v]$ with maximum number in σ. In the algorithm, the approximate distance from a vertex v_i to any non-neighbor v_j $(j > i)$ is computed via the maximum neighbor of v_i, using the formula $\hat{d}(v_i, v_j) := \hat{d}(\mathrm{mn}(v_i), v_j) + 1$.

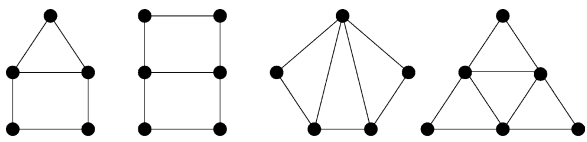

Fig. 2. Forbidden induced subgraphs.

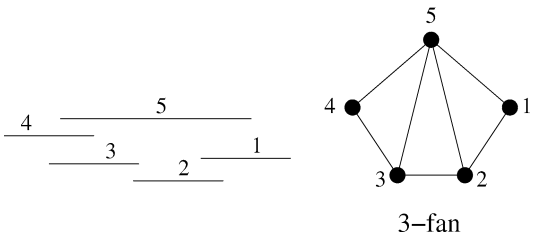

Fig. 3. An interval graph with its interval model. Vertices 1 and 2 are adjacent since the corresponding intervals intersect. Vertices 1 and 4 are not adjacent since the corresponding intervals do not intersect.

Algorithm APASP

for $i = 1$ **to** n **do** $\hat{d}(v_i, v_i) := 0$;
for $i = n - 1$ **downto** 1 **do**
  for $j = n$ **downto** $i + 1$ **do**
    if $v_i v_j \in E$ **then** $\hat{d}(v_i, v_j) := 1$ **else** $\hat{d}(v_i, v_j) := \hat{d}(\mathrm{mn}(v_i), v_j) + 1$;
return distances $\hat{d}$

We assume that the ordering σ used in the algorithm has the property that for all v_i with $i < n$, $\mathrm{mn}(v_i) \neq v_i$ (otherwise the algorithm will not work properly). This property holds for all orderings considered in this paper.

Note that this method is not new. It was already used to compute all pairs shortest path distances in strongly chordal graphs (see [17]) and in dually chordal graphs (see [9]). For both classes the characteristic orderings were employed. Here we show that this method gives the exact distances or a very good approximation of distances in many other graph classes.

Clearly, the algorithm can be modified to produce paths of length $\hat{d}$ rather than merely returning the approximate distances. Nevertheless, we present a separate procedure which for any two vertices u, v returns the path of length $\hat{d}(u, v)$ along which the APASP algorithm computes. Its complexity is $O(c \cdot \hat{d}(u, v))$, where c is the necessary time to verify the adjacency of two vertices.

Procedure (mn-path(u, v))

if u and v are adjacent **then return** (u, v)
else if $\sigma(u) < \sigma(v)$ **then return** $(u, \text{mn-path}(\mathrm{mn}(u), v))$
  else return $(\text{mn-path}(u, \mathrm{mn}(v)), v)$

The path between u and v returned by the procedure mn-path(u, v) will be called the *maximum-neighbor path*. Throughout this paper, by $d(u, v)$ we denote the (real) distance between u and v and by $\hat{d}(u, v)$ the approximate distance returned by the algorithm.

In this paper we will employ three types of vertex orderings: *breadth-first-search orderings, lexicographic-breadth-first-search orderings* of Rose et al. [35] and *lexical orderings* of Lubiw [26].

Let $\sigma = [v_1, v_2, \ldots, v_n]$ be any ordering of the vertex set of a graph G. We will write $a < b$ whenever in a given ordering σ vertex a has a smaller number than vertex b. More-

over, $\{a_1, \ldots, a_l\} < \{b_1, \ldots, b_k\}$ is an abbreviation for $a_i < b_j$ $(i = 1, \ldots, l;\ j = 1, \ldots, k)$. Let u be a vertex of G. We define the layers of G with respect to vertex u as follows: $L_i(u) = \{v\colon\ d(u, v) = i\}$ for $i = 0, 1, 2, \ldots$.

In a *breadth-first search* (*BFS*), started at a vertex u, the vertices of a graph G with n vertices are numbered from n to 1 in decreasing order. The vertex u is numbered by n and put on an initially empty queue of vertices. Then a vertex v at the head of the queue is repeatedly removed, and neighbors of v that are still unnumbered are consequently numbered and placed onto the queue. Clearly, BFS operates by proceeding vertices in layers: the vertices closest to the start vertex are numbered first, and the most distant vertices are numbered last. BFS may be seen to generate a rooted tree T with vertex u as the root. A vertex v is the *father* in T of exactly those neighbors in G which are inserted into the queue when v is removed.

An ordering σ of the vertex set of a graph G generated by a BFS will be called a *BFS-ordering* of G. Denote by $f(v)$ the farther of a vertex v with respect to σ. The following properties of a BFS-ordering will be used in what follows. Since all layers of V considered here are with respect to u, we will frequently use notation L_i instead of $L_i(u)$.

(P1) If $x \in L_i$, $y \in L_j$ and $i < j$, then $x > y$ in σ.
(P2) If $v \in L_q$ $(q > 0)$ then $f(v) \in L_{q-1}$ and $f(v)$ is the vertex from $N(v) \cap L_{q-1}$ with the largest number in σ, i.e., $f(v) = \mathrm{mn}(v)$.
(P3) If $x > y$, then either $\mathrm{mn}(x) > \mathrm{mn}(y)$ or $\mathrm{mn}(x) = \mathrm{mn}(y)$.
(P4) If $x, y, z \in L_j$, $x > y > z$ and $\mathrm{mn}(x)z \in E$, then $\mathrm{mn}(x) = \mathrm{mn}(y) = \mathrm{mn}(z)$ (in particular, $\mathrm{mn}(x)y \in E$).

Lexicographic breadth-first search (*LexBFS*), started at a vertex u, orders the vertices of a graph by assigning numbers from n to 1 in the following way. The vertex u gets the number n. Then each next available number k is assigned to a vertex v (as yet unnumbered) which has lexically largest vector $(s_n, s_{n-1}, \ldots, s_{k+1})$, where $s_i = 1$ if v is adjacent to the vertex numbered i, and $s_i = 0$ otherwise. An ordering of the vertex set of a graph generated by LexBFS we will call a *LexBFS-ordering*. It is well known that any LexBFS-ordering has property (P5) (cf. [25]). Moreover, any ordering fulfilling (P5) can be generated by LexBFS [20].

(P5) If $a < b < c$ and $ac \in E$ and $bc \notin E$ then there exists a vertex d such that $c < d$, $db \in E$ and $da \notin E$.

Lubiw in [26] has shown that any graph admits a *lexical ordering*, namely an ordering of V such that

(P6) If $a < b$ and $ac \in E$ and $bc \notin E$ then there exists a vertex d such that $c < d$, $d \in N[b]$ and $da \notin E$ $(d = b$ is allowed).

Clearly, any lexical ordering is a LexBFS-ordering, and any LexBFS-ordering is a BFS-ordering (but not conversely). Note also that for a given graph G, both a BFS-ordering and a LexBFS-ordering can be generated in linear time [22], while to date the fastest method—

doubly lexical ordering of the (closed) neighborhood matrix of G [26]—producing a lexical ordering of G takes $O(m \log n)$ [30] or $O(n^2)$ [34] time.

3. Distances in graphs from the weakly chordal hierarchy

In this section, we consider different subclasses of weakly chordal graphs. The weakly chordal graphs themselves will be considered in the next section. First we present the following simple but very useful lemma which holds for arbitrary graphs.

Lemma 1. *Let there exist integers $t \geqslant 2$ and $s \geqslant 0$ such that*

(1) *for all $x, y \in V$ with $x < y$ and $d(x, y) \geqslant t$, $d(x, y) = d(\mathrm{mn}(x), y) + 1$ holds,*
(2) *for all $x, y \in V$ with $d(x, y) < t$, $\hat{d}(x, y) \leqslant d(x, y) + s$ holds.*

Then $\hat{d}(x, y) \leqslant d(x, y) + s$ holds for all $x, y \in V$.

Proof. The proof is by induction on $d(x, y)$. Assume that $\hat{d}(x, y) \leqslant d(x, y) + s$ holds for all $x, y \in V$ with $d(x, y) < k$ $(k \geqslant t)$, and consider a pair x, y such that $d(x, y) = k$ and $x < y$. According to the APASP algorithm we have $\hat{d}(x, y) = \hat{d}(\mathrm{mn}(x), y) + 1$. Since $d(\mathrm{mn}(x), y) = d(x, y) - 1$, by induction, $\hat{d}(\mathrm{mn}(x), y) \leqslant d(\mathrm{mn}(x), y) + s$ holds. Therefore, $\hat{d}(x, y) \leqslant d(\mathrm{mn}(x), y) + s + 1 = d(x, y) + s$. □

Let $P = (x_0, x_1, \ldots, x_{k-1}, x_k)$ be an arbitrary path of G and σ be an ordering of the vertex set of this graph. The path P is *monotonic* (with respect to σ) if $x_0 < x_1 < \cdots < x_{k-1} < x_k$ holds whenever $x_0 < x_k$, and P is *convex* if there is an index i $(1 \leqslant i < k)$ such that $x_0 < x_1 < \cdots < x_{i-1} < x_i > x_{i+1} > \cdots > x_{k-1} > x_k$. Vertex x_i is called the *switching point* of the convex path P. Let now $P = (x_0, \ldots, x_k)$ be a shortest path of G connecting x_0 and x_k. We say that P is a *rightmost shortest path* if the sum $\sigma(x_0) + \sigma(x_1) + \cdots + \sigma(x_k)$ is largest among all shortest paths connecting x_0 and x_k.

3.1. Employing LexBFS-orderings

Let G be a HH-free graph and σ be a LexBFS-ordering of G. By P_4 we denote a path on four vertices.

Lemma 2 [19]. *G has no induced $P_4 = (c, a, b, d)$ with $\{a, b\} < \{c, d\}$ in σ.*

Corollary 1 [19]. *Let a, b, c be three distinct vertices of G such that $a < \{b, c\}$, $ab, ac \in E$ and $bc \notin E$. Then there is a vertex $d > \{b, c\}$ adjacent to b and c but not to a.*

Lemma 3 [19].

(1) *Every rightmost shortest path of G is either monotonic or convex.*
(2) *Let $P = (x_0, \ldots, x_k)$ be a rightmost shortest path in G which is convex and x_i be the switching point of P. Then $d(x_0, x_i) \geqslant d(x_k, x_i)$ if $x_0 < x_k$.*

Lemma 4. *Let x, y be a pair of vertices of G such that $x < y$ and $d(x, y) \geqslant 3$. Then $d(x, y) = d(\mathrm{mn}(x), y) + 1$ holds.*

Proof. Let $P = (x, x_1, x_2, \ldots, y)$ be a rightmost shortest path in G connecting vertices x and y. Since $x < y$ and $d(x, y) \geqslant 3$, by Lemma 3, we have $x < x_1 < x_2$. Assume $x_1 \neq \mathrm{mn}(x)$. Vertices $v = \mathrm{mn}(x)$ and x_2 cannot be adjacent, since otherwise we can replace x_1 in P with v and get a shortest path between x and y with larger sum, contradicting the choice of P. For path (v, x, x_1, x_2) we have $\{x, x_1\} < \{v, x_2\}$. Hence, by Lemma 2, v and x_1 are adjacent. Now we apply Corollary 1 to $x_1 < \{v, x_2\}$, $vx_2 \notin E$ and get a vertex u such that $u > \{v, x_2\}$, $uv, ux_2 \in E$ and $x_1u \notin E$. Furthermore, x cannot be adjacent to u because $v = \mathrm{mn}(x)$ and $u > v$. But then a house formed by x, v, x_1, u, x_2 is induced, that is impossible for a HH-free graph G. □

Unfortunately, this result cannot be extended to weakly chordal graphs. For any even integer $k \geqslant 4$, there exist a weakly chordal graph $G = (V, E)$, a LexBFS-ordering σ of G, and a pair of vertices $x, y \in V$ such that $x < y$ in σ, $d(x, y) = k$ but $d(x, y) < d(\mathrm{mn}(x), y) + 1$. Figure 4 gives such a weakly chordal graph for $k = 4$ (this example can easily be extended to a larger k). In all figures dark edges indicate the maximum-neighbor path between vertices numbered 1 and 2.

Lemma 5. *Let x, y be a pair of vertices of G such that $x < y$ and $d(x, y) = 2$. Then*

(1) *$d(x, y) = \hat{d}(x, y)$ if $\mathrm{mn}(x)y \in E$ or there exists a vertex $z \in N(x) \cap N(y)$ such that $z < y$;*
(2) *$d(x, y) \geqslant \hat{d}(x, y) - 1$ if there exists a vertex $z \in N(x) \cap N(y)$ such that $\mathrm{mn}(x)z \in E$.*

Proof. (1) Clearly, if $\mathrm{mn}(x)y \in E$, then $\hat{d}(\mathrm{mn}(x), y) = 1$ and $\hat{d}(x, y) = \hat{d}(\mathrm{mn}(x), y) + 1 = 2 = d(x, y)$. So, assume that $\mathrm{mn}(x)y \notin E$ and there is a vertex $z \in N(x) \cap N(y)$ such that $z < y$. For path $(\mathrm{mn}(x), x, z, y)$ we have $\{x, z\} < \{\mathrm{mn}(x), y\}$. Hence, by Lemma 2, $\mathrm{mn}(x)$ and z must be adjacent. Now we can proceed as in the proof of Lemma 4 and construct an induced house in HH-free graph G, which is impossible.

(2) We may assume that $\mathrm{mn}(x)y \notin E$ and for the vertex $z \in N(x) \cap N(y)$ with $\mathrm{mn}(x)z \in E$, $z > y$ holds. We have $d(\mathrm{mn}(x), y) = 2$ and $y < z < \mathrm{mn}(x)$. Hence, by (1),

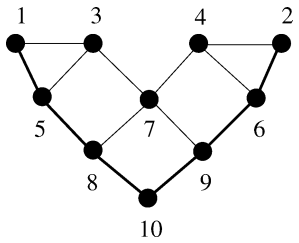

Fig. 4. A weakly chordal graph with a LexBFS-ordering; $d_{1,2} = d_{2,5} = 4$.

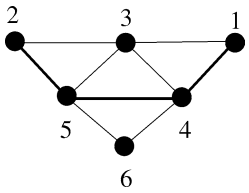

Fig. 5. A LexBFS-ordering of a 3-sun such that $\hat{d}_{1,2} - d_{1,2} = 1$.

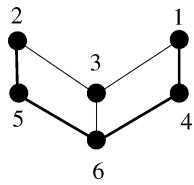

Fig. 6. A LexBFS-ordering of a domino such that $\hat{d}_{1,2} - d_{1,2} = 2$.

$\hat{d}(y, \mathrm{mn}(x)) = d(y, \mathrm{mn}(x))$, and therefore $\hat{d}(x, y) = \hat{d}(\mathrm{mn}(x), y) + 1 = d(\mathrm{mn}(x), y) + 1 = 3 = d(x, y) + 1$. □

Lemma 6. *Let x, y be a pair of vertices such that $x < y$, $d(x, y) = 2$, $\mathrm{mn}(x)y \notin E$, and there exists a vertex $z \in N(x) \cap N(y)$ with $\mathrm{mn}(x)z \notin E$ and $z > y$. Then G contains a domino as an induced subgraph and $d(x, y) = \hat{d}(x, y) - 2$ holds.*

Proof. Let $v = \mathrm{mn}(x)$, $u = \mathrm{mn}(z)$ and $w = \mathrm{mn}(y)$ be the maximum neighbors in σ of x, z and y, respectively. We have $x < y < z < v$, $vz, vy \notin E$. By property (P3), we have also $u \geqslant w > v$ and hence $xu, xw \notin E$. Since $\{x, z\} < \{v, u\}$, by Lemma 2, vertices v and u must be adjacent. If $uy \in E$ then we get an induced house in G, which is impossible. Therefore, $d(y, u) = 2$, $y < z < u$, and we can apply Lemma 5 and get $d(y, u) = \hat{d}(y, u) = 2$. That is, vertex $w = \mathrm{mn}(y)$ is adjacent to u. If now $wz \in E$ or $wv \in E$ then vertices x, y, z, v, u, w induce either a house or a cycle of length 5. Since these induced subgraphs are forbidden, w is neither adjacent to v nor to z, i.e., vertices x, y, z, v, u, w form an induced domino.

Now consider maximum neighbors $h = \mathrm{mn}(v)$, $s = \mathrm{mn}(u)$ and $t = \mathrm{mn}(w)$ of v, u and w, respectively. We had $d(v, w) = 2$, $v < w < u$ and $u \in N(v) \cap N(w)$. If also $hu, hw \notin E$ then, as we have shown above, vertices v, u, w, h, s, t will form an induced domino with edges $vu, vh, us, uw, wt, hs, st$. Since $v = \mathrm{mn}(x)$, $u = \mathrm{mn}(z)$, $w = \mathrm{mn}(y)$ and $v < w < u < h < t < s$, there cannot be edges with one end in $\{x, z, y\}$ and the other end in $\{h, s, t\}$. That is, the cycle $(x, z, y, w, t, s, h, v, x)$ of G of length 8 is induced. Since such cycles are forbidden in G, we must have $hu \in E$ or $hw \in E$.

If $hu \in E$ then vertices x, z, v, u, h induce a house, which is impossible (note that h is neither adjacent to x nor to z because $h > u = \mathrm{mn}(z) > v = \mathrm{mn}(x)$).

So, it remains only to consider the case when $hw \in E$. In this case, according to the APASP algorithm, we have $\hat{d}(h, w) = 1$ and therefore $\hat{d}(x, y) = \hat{d}(v, y) + 1 = \hat{d}(v, w) + 1 + 1 = \hat{d}(h, w) + 1 + 1 + 1 = 4$. Finally, $d(x, y) = 2 = 4 - 2 = \hat{d}(x, y) - 2$. □

Combining Lemmas 1, 4, 5, and 6 we obtain the following results for HH-free, HHD-free and chordal graphs.

Theorem 1. *Let G be a HH-free graph and σ be a LexBFS-ordering of G. Then, for all vertices $v, u \in V$, the distances returned in $\hat{d}$ by the algorithm APASP satisfy the inequality*

$$0 \leqslant \hat{d}(v, u) - d(v, u) \leqslant 2.$$

Corollary 2. *All distances in HH-free graphs with an additive one-sided error of at most* 2 *can be found in $O(n^2)$ time.*

Theorem 2. *Let G be a HHD-free graph and σ be a LexBFS-ordering of G. Then, for all vertices $v, u \in V$, the distances returned in $\hat{d}$ by the algorithm APASP satisfy the inequality*

$$0 \leqslant \hat{d}(v, u) - d(v, u) \leqslant 1.$$

Corollary 3. *All distances in HHD-free graphs with an additive one-sided error of at most* 1 *can be found in $O(n^2)$ time.*

Corollary 4 [40]. *All distances in chordal graphs with an additive one-sided error of at most* 1 *can be found in* $O(n^2)$ *time.*

Note that the bounds given in Theorems 1 and 2 are tight. Figure 5 shows a chordal graph with a LexBFS-ordering where the difference between $\hat{d}$ and d can be 1, Fig. 6 gives a similar example for HH-free graphs.

For the distance-hereditary graphs this method gives the exact distances.

Theorem 3. *Let G be a distance-hereditary graph and σ be a LexBFS-ordering of G. Then the distances returned in $\hat{d}$ by the algorithm APASP are the real distances, i.e., $\hat{d}(v,u) = d(v,u)$ for any $v,u \in V$.*

Proof. According to Lemmas 1, 4, 5, and 6, we have to consider only the case $d(v,u) = 2$, $v < u$, $\mathrm{mn}(v)u \notin E$ and there exists a vertex $z \in N(v) \cap N(u)$ such that $\mathrm{mn}(v)z \in E$, $z > u$.

Consider the maximum neighbor $w = \mathrm{mn}(u)$ of u. Since $v < u$ and $\mathrm{mn}(v)u \notin E$, by property (P3), we have $\mathrm{mn}(v) < w$, i.e., $v < u < z < \mathrm{mn}(v) < w$, and $wv \notin E$. By Lemma 2, vertex w has to be adjacent to $\mathrm{mn}(v)$ or z. If $\mathrm{mn}(v)w \in E$ then we get an induced house or an induced 3-fan depending on adjacency of w and z. Distance-hereditary graphs cannot contain such induced subgraphs. Hence, w is not adjacent to $\mathrm{mn}(v)$ but is adjacent to z. Now we apply Corollary 1 to $z < \{\mathrm{mn}(v), w\}$, $\mathrm{mn}(v)w \notin E$ and get a vertex t such that $t > w$, $\mathrm{mn}(v)t, tw \in E$ and $zt \notin E$. Since $t > w > \mathrm{mn}(v)$, vertex v is adjacent neither to t nor to w. Thus, we have created an induced house in G formed by $v, z, \mathrm{mn}(v), t, w$, contradicting the fact G is distance-hereditary. □

Later we will improve Theorem 3 by showing that even BFS-ordering is enough to get the exact distances in distance-hereditary graphs.

Corollary 5 [17]. *The all pairs shortest path problem in distance-hereditary graphs can be solved in* $O(n^2)$ *time.*

Lemma 4 allows us also to state the following interesting result. All what one needs to compute the exact distances in a HH-free graph G in $O(n^2)$ time is to know in advance the square G^2 of G. Recall that the square G^2 of $G = (V, E)$ is the graph with the same vertex set V where two vertices u and v ($u \neq v$) are adjacent if their distance in G is at most 2. Unfortunately, computing G^2 even for split graphs (a subclass of chordal graphs) is as hard as for general graphs.

Theorem 4. *Let G be a HH-free graph and σ be a LexBFS-ordering of G. Let also the square G^2 of G be given. Then the all pairs shortest path problem on G can be solved in* $O(n^2)$ *time.*

Proof. Let $\sigma = [v_1, v_2, \ldots, v_n]$ be a LexBFS-ordering of G. Clearly (see Lemma 4), the following modification of APASP algorithm computes the exact distances in G.

for $i = 1$ **to** n **do** $d(v_i, v_i) := 0$;
for $i = n - 1$ **downto** 1 **do**
  for $j = n$ **downto** $i + 1$ **do**
    if $v_i v_j \in E(G)$ **then** $d(v_i, v_j) := 1$
    else if $v_i v_j \in E(G^2) \setminus E(G)$ **then** $d(v_i, v_j) := 2$
      else $d(v_i, v_j) := d(\mathrm{mn}(v_i), v_j) + 1$;
return distances d

If the adjacency matrix of G^2 is given, this algorithm clearly runs in $O(n^2)$ time. □

Corollary 6. *The all pairs shortest path problem on a HHD-free graph G can be solved in $O(n^2)$ time if G^2 is known.*

Corollary 7 [23]. *The all pairs shortest path problem on a chordal graph G can be solved in $O(n^2)$ time if G^2 is known.*

3.2. Employing lexical orderings

In this subsection we consider strongly chordal graphs and chordal bipartite graphs.

Theorem 5. *Let G be a strongly chordal graph and σ be a lexical ordering of G. Then the distances returned in $\hat{d}$ by the algorithm APASP are the real distances, i.e., $\hat{d}(v, u) = d(v, u)$ for any $v, u \in V$.*

Proof. Since any lexical ordering of a graph is a LexBFS-ordering and strongly chordal graphs are HH-free (even chordal) graphs, Lemmas 4 and 5 can be applied. According to those lemmas and Lemma 1, we have to consider only the case when $v < u$, $d(v, u) = 2$ and $\mathrm{mn}(v)u \notin E$. We claim that this case is impossible.

It is well known [26] that any lexical ordering of a strongly chordal graph is a strong elimination ordering. Consider a vertex $z \in N(v) \cap N(u)$. We have $zv, zu, \mathrm{mn}(v)v \in E$, $z < \mathrm{mn}(v)$, $v < u$. Since $\mathrm{mn}(v)u \notin E$, we get a contradiction with σ being a strong elimination ordering. □

Corollary 8 [6,17]. *The all pairs shortest path problem in strongly chordal graphs can be solved in $O(n^2)$ time.*

Since interval graphs are strongly chordal, we immediately conclude.

Corollary 9 [4,27,32,41]. *The all pairs shortest path problem in interval graphs can be solved in $O(n^2)$ time.*

Now we will present a similar result for chordal bipartite graphs. It is well known [26, 34] that a graph is chordal bipartite if and only if it admits a special vertex ordering called *Γ-free ordering*. The name came from the existence of a special ordering of rows and

columns of the bipartite adjacency matrix of a chordal bipartite graph. In graph terminology this can be stated as follows. For any chordal bipartite graph $G = (X \cup Y, E)$ there are orderings $\tau_X = [x_1, \ldots, x_{|X|}]$ and $\tau_Y = [y_1, \ldots, y_{|Y|}]$ such that for any $a, d \in X$, $b, c \in Y$ with $ab, ac, bd \in E$ and $a <_{\tau_X} d$, $b <_{\tau_Y} c$, the edge cd must exist in G. It is well known that doubly lexical ordering (see [26]) of the bipartite adjacency matrix of a chordal bipartite graph gives these orderings in $O(n^2)$ time [34]. Moreover, resulting orderings τ_X and τ_Y have property (P6). To be able to use Γ-free ordering in our distance computations, we will first transform orderings τ_X and τ_Y into one ordering $\sigma = [v_1, \ldots, v_n]$, where $n = |X| + |Y|$, which will be a LexBFS-ordering of G and will satisfy the property of a Γ-free ordering. Note that, a simpler idea to use doubly lexical ordering of the closed neighborhood matrix of G will not work here. Although the resulting ordering will be a LexBFS ordering of G, there are examples of chordal bipartite graphs for which not all such orderings have the Γ-free property. See Fig. 6 for an example; that ordering of a domino is lexical but there is no edge between 2 and 4 in a path $(2, 3, 1, 4)$.

A desired ordering $\sigma = [v_1, \ldots, v_n]$ one can get in the following way. First using the $O(n^2)$ time algorithm from [34] we get orderings τ_X and τ_Y. Then, starting at vertex $x_{|X|}$ (or at $y_{|Y|}$) we perform a LexBFS, breaking ties according to indexes in τ_X and τ_Y; if both vertices v and u can be chosen to be numbered next by LexBFS, we number first that vertex which has larger index in τ_X, τ_Y. Note that, if vertices cannot be distinguished by LexBFS, then they have a common neighbor and therefore both are either in X or in Y. We call the resulting ordering a *lexical+ ordering* of G.

The next lemma shows that σ preserves the order relations we had in τ_X, τ_Y. Clearly, σ can be found in total $O(n^2)$ time.

Lemma 7. *For any vertices $v, u \in X$ (respectively, $v, u \in Y$), $v <_\sigma u$ if and only if $v <_{\tau_X} u$ (respectively, $v <_{\tau_Y} u$).*

Proof. Let σ alternate the order relations we had in τ_X (or in τ_Y), and let a be a vertex of G with the largest index in σ for which there is a vertex b ($b <_\sigma a$) such that $a <_{\tau_X} b$ or $a <_{\tau_Y} b$ (depending on which part of G, X or Y, vertices a, b are from). Without loss of generality assume $a <_{\tau_X} b$, i.e., $a, b \in X$. According to our construction of σ, there must be a vertex c in G such that $a <_\sigma c$ and $ca \in E$, $cb \notin E$ (vertices a and b were distinguished from c and therefore a was numbered earlier (with larger number) in σ than b). We choose such c rightmost in σ. Since $a <_{\tau_X} b$ and c is adjacent to a but not to b, by property (P6), there must exist a vertex d in G such that $c <_{\tau_Y} d$ and $db \in E$, $da \notin E$. Since vertices a and b are distinguished from d and $b <_\sigma a$, according to our choice of c, $d <_\sigma c$ must hold. But that contradicts our choice of a; c has larger index than a in σ and $d <_\sigma c$ and $c <_{\tau_Y} d$. □

Corollary 10. *Any lexical+ ordering σ of a chordal bipartite graph $G = (V, E)$ is Γ-free, i.e., for any $a, b, c, d \in V$ such that $ab, ac, bd \in E$, $a <_\sigma d$, $b <_\sigma c$, $cd \in E$ must hold.*

Theorem 6. *Let G be a chordal bipartite graph and σ be a lexical+ ordering of G. Then the distances returned in $\hat{d}$ by the algorithm APASP are the real distances, i.e., $\hat{d}(v, u) = d(v, u)$ for any $v, u \in V$.*

Proof. Since σ is a LexBFS-ordering and chordal bipartite graphs are HH-free graphs, Lemmas 4 and 5 can be applied. According to those lemmas and Lemma 1, we have to consider only the case when $v <_\sigma u$, $d(v,u) = 2$ and $\mathrm{mn}(v)u \notin E$. Again we claim that this case is impossible. Consider a vertex $z \in N(v) \cap N(u)$. We have $zv, zu, \mathrm{mn}(v)v \in E$, $z <_\sigma \mathrm{mn}(v)$, $v <_\sigma u$. Since $\mathrm{mn}(v)u \notin E$, a contradiction with Corollary 10 arises. □

Corollary 11 [24]. *The all pairs shortest path problem in chordal bipartite graphs can be solved in $O(n^2)$ time.*

4. Distances in graphs with small size of largest induced cycle: employing BFS-orderings

In order to capture the notion of "small" induced cycles, we define a graph to be *k-chordal* if it has no induced cycles of size greater than k. Note that chordal graphs are precisely the 3-chordal graphs, weakly chordal graphs are 4-chordal graphs and AT-free graphs are 5-chordal.

For k-chordal graphs we obtained the following results. It is interesting to note that even simple BFS-orderings succeed in getting a good approximation for distances in k-chordal graphs (for small $k \geqslant 4$).

Theorem 7. *Let G be a k-chordal graph ($k \geqslant 4$) and σ be a BFS-ordering of G. Then, for all vertices $x, y \in V$, the distances returned in $\hat{d}$ by the algorithm APASP satisfy the inequality*

$$0 \leqslant \hat{d}(x,y) - d(x,y) \leqslant k-1.$$

Proof. Let T be the BFS-tree associated with BFS-ordering $\sigma = [v_1, \ldots, v_n]$. Recall that T is rooted at a vertex $u = v_n$ and the father of a vertex v ($v \neq u$) in T is denoted by $f(v)$. In this proof we will frequently use properties (P1)–(P4) of BFS-orderings and associated BFS-trees. Also, since vertex u is fixed, we will use L_i to denote layer $L_i(u)$ of G (and of T as well). Let $P_x = (u = x_0, x_1, \ldots, x_{t-1}, x_t = x)$ and $P_y = (u = y_0, y_1, \ldots, y_{s-1}, y_s = y)$ be the paths of T connecting the root u with x and y, respectively (see Fig. 7 for an illustration). By property (P2), $x_{i-1} = f(x_i) = \mathrm{mn}(x_i)$, for any $i = 1, \ldots, t$, and $y_{j-1} = f(y_j) = \mathrm{mn}(y_j)$, for any $j = 1, \ldots, s$. Moreover, $x_i < y_j$ and $y_i < x_j$, if $i > j$ (by property (P1) and since $x_i, y_i \in L_i$ and $x_j, y_j \in L_j$).

Claim 1. *For any vertex $v \in L_q$ ($q < t$) such that $v \neq x_q$ and $d(x_t, v) = d(x_t, x_q) = t - q$, $x_q > v$ holds.*

Proof of Claim 1. Let $P(x,v) = (x = v_t, v_{t-1}, \ldots, v_{q+1}, v_q = v)$ be a shortest path between x and v. Clearly, $v_i \in L_i$ for any $i \in \{q, q+1, \ldots, t-1, t\}$. Let $v_i = x_i$ be a common vertex of paths P_x and $P(x,v)$ closest to layer L_q. Then, neighbors v_{i-1} and x_{i-1} of x_i are distinct and, since $x_{i-1} = \mathrm{mn}(x_i)$, $v_{i-1} < x_{i-1}$ holds. Assume now that there is an index j

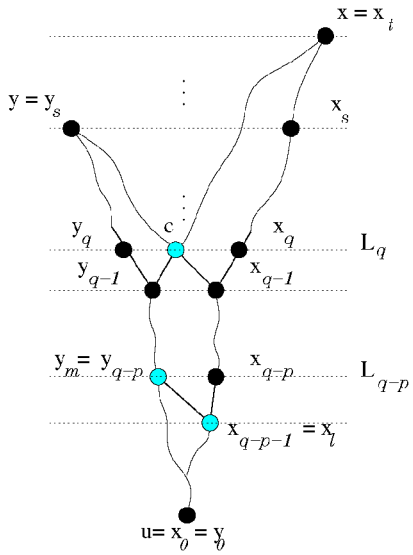

Fig. 7. Illustration to the proof of Theorem 7. A shortest path P between x and y containing c, the maximum-neighbor paths P_x and P_y joining u with x and y, respectively, and the maximum-neighbor path P' between x and y with the binding edge $x_l y_m$.

$(q < j < i)$ such that $v_j < x_j$ but $v_{j-1} > x_{j-1}$. By property (P3), from $v_j < x_j$ we conclude that $\mathrm{mn}(x_j) \geqslant \mathrm{mn}(v_j)$. But, since $x_{j-1} = \mathrm{mn}(x_j)$ and $\mathrm{mn}(v_j) \geqslant v_{j-1}$, a contradiction with $v_{j-1} > x_{j-1}$ arises. Thus, no such index j exists and therefore $x_q > v_q = v$. □

Analogously, for any vertex $v \in L_q$ $(q < s)$ such that $v \neq y_q$ and $d(y_s, v) = d(y_s, y_q) = s - q$, $y_q > v$ must hold.

As we mentioned earlier, the algorithm APASP computes the value $\hat{d}(x, y)$ along the maximum-neighbor path which consists of a subpath $P'_x = (x_l, \ldots, x_{t-1}, x_t = x)$ of P_x and a subpath $P'_y = (y_m, \ldots, y_{s-1}, y_s = y)$ of P_y such that $x_l y_m \in E$ and therefore $|l - m| \leqslant 1$ (see the algorithm APASP and the procedure mn-path).

More precisely, the maximum-neighbor path $P' = (x = x_t, x_{t-1}, \ldots, x_l, y_m, \ldots, y_{s-1}, y_s = y)$ between x and y is constructed by the procedure mn-path(x, y) in the following way. Assume, without loss of generality, that $x < y$ in σ and hence $t \geqslant s$. It constructs first a subpath $(x = x_t, x_{t-1}, \ldots, x_{s+1})$ (it can be empty if $t = s$). Then, if $x_{s+1} y_s \in E$, it stops and returns the path $(x = x_t, x_{t-1}, \ldots, x_{s+1}, y_s)$. Otherwise, it advances towards $x_s = f(x_{s+1}) = \mathrm{mn}(x_{s+1})$ and reaches layer L_s containing vertex y_s. In each current layer L_i $(i \leqslant s)$, it does the following:

- stops and returns path $(x = x_t, x_{t-1}, \ldots, x_{i+1}, x_i, y_i, y_{i+1}, \ldots, y_s = y)$, if $x_i y_i \in E$;
- advances towards $x_{i-1} = f(x_i) = \mathrm{mn}(x_i)$ on path P_x and then stops and returns path $(x = x_t, \ldots, x_{i+1}, x_i, x_{i-1}, y_i, y_{i+1}, \ldots, y_s = y)$, if $x_i < y_i$ in σ and $x_i y_i \notin E$, $x_{i-1} y_i \in E$;

- advances towards $x_{i-1} = f(x_i) = \mathrm{mn}(x_i)$ on path P_x and then towards $y_{i-1} = f(y_i) = \mathrm{mn}(y_i)$ on path P_y, if $x_i < y_i$ in σ and $x_i y_i, x_{i-1} y_i \notin E$;
- advances towards $y_{i-1} = f(y_i) = \mathrm{mn}(y_i)$ on path P_y and then stops and returns path $(x = x_t, \ldots, x_{i+1}, x_i, y_{i-1}, y_i, y_{i+1}, \ldots, y_s = y)$, if $x_i > y_i$ in σ and $x_i y_i \notin E$, $y_{i-1} x_i \in E$;
- advances towards $y_{i-1} = f(y_i) = \mathrm{mn}(y_i)$ on path P_y and then towards $x_{i-1} = f(x_i) = \mathrm{mn}(x_i)$ on path P_x, if $x_i > y_i$ in σ and $x_i y_i, y_{i-1} x_i \notin E$.

From this we can conclude that the maximum-neighbor path is always an induced path and the *binding edge* $x_l y_m$ of P' is an edge $x_i y_j$ of G such that $x_i \in P_x$, $y_j \in P_y$ and the sum $i + j$ is maximal. Indeed, assume there is an edge $x_{i'} y_{j'}$, $x_{i'} \in P_x$, $y_{j'} \in P_y$ with $i' + j' > i + j$. Since this edge is overlooked by the mn-path procedure, we have either $i' = j' - 1$ and $y_{j'} < x_{j'}$ or $j' = i' - 1$ and $x_{i'} < y_{i'}$. Without loss of generality, assume that $i' = j' - 1$ and $y_{j'} < x_{j'}$ hold. But then, by properties (P2) and (P3), $\mathrm{mn}(x_{j'}) = \mathrm{mn}(y_{j'}) = x_{i'}$ and, according to mn-path procedure, at least edge $x_{j'} x_{i'}$ $(= x_{j'} \mathrm{mn}(y_{j'}))$ will be chosen by the mn-path procedure as the binding edge of P'.

Consider now a shortest path P of G connecting vertices x and y and let q be the smallest index such that $L_q \cap P \neq \emptyset$ (see Fig. 7 for an illustration). Let also c be a vertex from $L_q \cap P$. Since T is a BFS-tree rooted at u, we have $d(x_t, c) \geqslant d(x_t, x_q) = t - q$ and $d(y_s, c) \geqslant d(y_s, y_q) = s - q$. We want to show that $\mathrm{length}(P') - \mathrm{length}(P) = \hat{d}(x, y) - d(x, y)$ is not larger that $k - 1$. We will distinguish between two cases: $k = 2p + 2$ and $k = 2p + 3$ $(p \geqslant 1)$.

If there is an edge between vertices of subpaths $(x_{q-p}, \ldots, x_{t-1}, x)$ and $(y_{q-p}, \ldots, y_{s-1}, y)$, then $\hat{d}(x, y) \leqslant t - (q - p) + 1 + s - (q - p) = (t - q) + (s - q) + 2p + 1 \leqslant d(x_t, c) + d(y_s, c) + 2p + 1 = d(x, y) + 2p + 1$. Therefore, $\hat{d}(x, y) - d(x, y) \leqslant 2p + 1 \leqslant k - 1$ and we are done.

Thus, we may assume that no edge exists between those subpaths. In a subgraph of G induced by vertices $\{x_{q-1}, x_q, \ldots, x_{t-1}\} \cup P \cup \{y_{s-1}, \ldots, y_q, y_{q-1}\}$, consider a shortest path P'' connecting x_{q-1} and y_{q-1}. Clearly, $\mathrm{length}(P'') \geqslant 2$ and no inner vertex of this path is adjacent to a vertex from L_i $(i \leqslant q - 2)$.

Assume now, without loss of generality, that $x_{q-p} < y_{q-p}$. If x_{q-p-1}, which is $f(x_{q-p}) = \mathrm{mn}(x_{q-p})$, is adjacent to y_{q-p}, then $\hat{d}(x, y) = t - (q - p - 1) + 1 + s - (q - p) = (t - q) + (s - q) + 2p + 2 \leqslant d(x_t, c) + d(y_s, c) + 2p + 2 = d(x, y) + 2p + 2$. Therefore, $\hat{d}(x, y) - d(x, y) \leqslant 2p + 2$ and we are done in case $k = 2p + 3$ and in case $k = 2p + 2$ with $d(x_t, c) > d(x_t, x_q) = t - q$ or $d(y_s, c) > d(y_s, y_q) = s - q$. It remains to consider the following two cases.

Case 1. $x_{q-p-1} y_{q-p} \notin E$.

In this case the binding edge $x_l y_m$ of the maximum-neighbor path P' is too far from layer L_q ($l \leqslant q - p - 1$ and $m \leqslant q - p - 1$ hold), and we can create an induced cycle in G of length at least $2p + 3$ (see Fig. 7). Indeed, since no edge can exist between inner vertices of path P'' and vertices of a subpath $P''' = (x_{q-2}, \ldots, x_l, y_m, \ldots, y_{q-2})$ of P', paths P'' and P''' together with edges $x_{q-1} x_{q-2}$, $y_{q-1} y_{q-2}$ form an induced cycle in G of length $\geqslant \mathrm{length}(P'') + p + 1 + p \geqslant 2 + 2p + 1 = 2p + 3$. As G is k-chordal, no induced cycles of

length greater than k are allowed. Therefore, we conclude that $k = 2p+3$, length$(P'') = 2$ and $x_{q-p-1}y_{q-p-1} \in E$, i.e., $l = m = q - p - 1$. Furthermore, if $d(x_t, c) > d(x_t, x_q) = t - q$ or $d(y_s, c) > d(y_s, y_q) = s - q$, then from $\hat{d}(x, y) = t - (q - p - 1) + 1 + s - (q - p - 1) = (t - q) + (s - q) + 2p + 3 < d(x_t, c) + d(y_s, c) + 2p + 3 = d(x, y) + 2p + 3$, we deduce that $\hat{d}(x, y) - d(x, y) \leqslant 2p + 2 = k - 1$. So, we may assume $d(x_t, c) = d(x_t, x_q) = t - q$, $d(y_s, c) = d(y_s, y_q) = s - q$, and hence $P \cap L_q = \{c\}$ and, by Claim 1, $c < \{x_q, y_q\}$. From length$(P'') = 2$, we have also $x_{q-1}c, y_{q-1}c \in E$. Now assuming, without loss of generality, that $x_q < y_q$, and having $c < x_q < y_q$, mn$(y_q)c \in E$ and mn$(y_q)x_q \notin E$, we arrive at a contradiction with property (P4).

Case 2. $k = 2p + 2$ and $x_{q-p-1}y_{q-p} \in E$.

In this case we have $d(x_t, c) = d(x_t, x_q) = t - q$, $d(y_s, c) = d(y_s, y_q) = s - q$, and hence again $P \cap L_q = \{c\}$ and, by Claim 1, $c < \{x_q, y_q\}$. Inner vertices of a shortest path P'' are not adjacent to any vertex from $P''' = (x_{q-2}, \ldots, x_{q-p-1}, y_{q-p}, \ldots, y_{q-2})$ (note that if $p = 1$ then $P''' = (x_{q-2})$). To avoid an induced cycle of G of length $\geqslant 2p + 3$ (formed by vertices of P'' and P'''), we must have length$(P'') = 2$, i.e., $x_{q-1}c, y_{q-1}c \in E$ (see Fig. 7 for an illustration). Assuming again, without loss of generality, that $x_q < y_q$, and having $c < x_q < y_q$, mn$(y_q)c \in E$ and mn$(y_q)x_q \notin E$, we arrive at the same contradiction with property (P4). □

Corollary 12. *Let G be a distance-hereditary graph and σ be a BFS-ordering of G. Then the distances returned in $\hat{d}$ by the algorithm APASP are the real distances, i.e., $\hat{d}(v, u) = d(v, u)$ for any $v, u \in V$.*

Proof. In the proof of Theorem 7 we have shown that, for any vertices v, u, the maximum-neighbor path connecting them is an induced path. Since any induced path in distance-hereditary graphs is a shortest one (see [7]) and the length of the maximum-neighbor path connecting v and u is $\hat{d}(v, u)$, we have $\hat{d}(v, u) = d(v, u)$. □

Corollary 13. *All distances in k-chordal graphs with an additive one-sided error of at most $k - 1$ can be found in $O(n^2)$ time.*

Corollary 14. *All distances in hole-free graphs with an additive one-sided error of at most 3 can be found in $O(n^2)$ time.*

Corollary 15. *All distances in weakly chordal graphs with an additive one-sided error of at most 3 can be found in $O(n^2)$ time.*

The result can be strengthened further for AT-free graphs.

Theorem 8. *Let G be an AT-free graph and σ be a BFS-ordering of G. Then, for all vertices $x, y \in V$, the distances returned in $\hat{d}$ by the algorithm APASP satisfy the inequality*

$$0 \leqslant \hat{d}(x, y) - d(x, y) \leqslant 2.$$

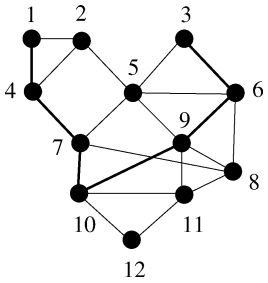

Fig. 8. A weakly chordal graph with a LexBFS-ordering; $\hat{d}_{1,3} - d_{1,3} = 3$.

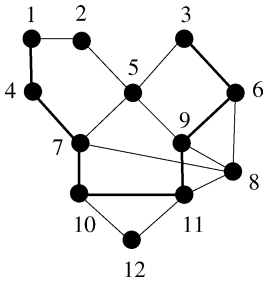

Fig. 9. A 5-chordal graph with a LexBFS-ordering; $\hat{d}_{1,3} - d_{1,3} = 4$.

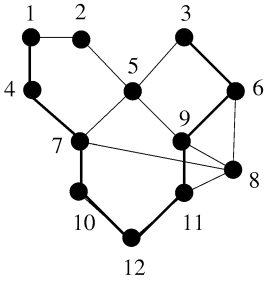

Fig. 10. A 6-chordal graph with a LexBFS-ordering; $\hat{d}_{1,3} - d_{1,3} = 5$.

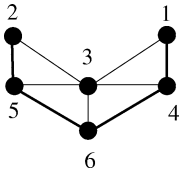

Fig. 11. A chordal graph with a BFS-ordering; $\hat{d}_{1,2} - d_{1,2} = 2$.

Proof. As in the proof of Theorem 7, let P be a shortest path of G connecting vertices x and y, c be a vertex from P closest to u, and $P_x = (u = x_0, x_1, \ldots, x_{t-1}, x_t = x)$, $P_y = (u = y_0, y_1, \ldots, y_{s-1}, y_s = y)$, $P' = (x = x_t, x_{t-1}, \ldots, x_l, y_m, \ldots, y_{s-1}, y_s = y)$ be the maximum-neighbor paths between x and u, y and u, x and y, respectively. Assume also $c \in L_q$ and, without loss of generality, $x_q < y_q$ (see Fig. 7).

If there is no edge between a subpath $P'_x = (x_{q-1}, \ldots, x_{t-1}, x)$ of P_x and a subpath $P'_y = (y_{q-1}, \ldots, y_{s-1}, y)$ of P_y, then vertices x_q, y_q and u form an asteroidal triple (note that path $(u = x_0, x_1, \ldots, x_{q-1}, x_q)$ avoids the closed neighborhood of y_q, path $(u = y_0, y_1, \ldots, y_{q-1}, y_q)$ avoids the closed neighborhood of x_q, and vertices x_q, y_q can be joined by a path in $\{x_q, x_{q+1}, \ldots, x_{t-1}\} \cup P \cup \{y_{s-1}, \ldots, y_{q+1}, y_q\}$ which avoids the closed neighborhood of u). Since this is impossible in AT-free graphs, the binding edge $x_l y_m$ of P' must be between paths P'_x and P'_y, i.e., $l \geqslant q-1$ and $m \geqslant q$ (recall, that we have assumed $x_q < y_q$).

Thus, $\hat{d}(x, y) \leqslant t-(q-1)+1+s-q = (t-q)+(s-q)+2 \leqslant d(x_t, c)+d(y_s, c)+2 = d(x, y)+2$. Therefore, $\hat{d}(x, y) - d(x, y) \leqslant 2$ and we are done. □

In Figs. 8, 9 and 10 we present three graphs with BFS-orderings which show that the bound stated in Theorem 7 is tight at least for 4-, 5-, and 6-chordal graphs. Since all orderings shown in these figures are even LexBFS-orderings, considering LexBFS instead of BFS in Theorem 7 will not give any better bound (for $k \geqslant 4$). In contrast, for $k = 3$,

i.e., for chordal graphs, a LexBFS-ordering gives a better result than a BFS-ordering. It turns out that a BFS-ordering is not enough to get an additive one-sided error of at most 1 for all chordal graphs (see Fig. 11 for an example). The domino with a LexBFS-ordering presented in Fig. 6 shows that the bound in Theorem 8 for AT-free graphs is tight.

Corollary 16. *All distances in AT-free graphs with an additive one-sided error of at most* 2 *can be found in* $O(n^2)$ *time.*

Note that distances in k-chordal graphs were already considered in [13]. It was shown that for any k-chordal graph $G = (V, E)$ there exists a tree $T = (V, F)$ such that $|d_G(u, v) - d_T(u, v)| \leqslant \lfloor k/2 \rfloor + \alpha$ for all $u, v \in V$, where $\alpha = 1$ if $k \neq 4, 5$, and $\alpha = 2$ otherwise. Such a tree T can be constructed in linear time $O(n + m)$.

5. Conclusion

In this paper we presented a very simple and efficient approach for solving the APASP problem on weakly chordal graphs and its subclasses. The same approach worked well also on graphs with small size of largest induced cycle. It gave a unified way to solve the APSP and APASP problems on different graph classes, including weakly chordal, House-Hole-free, House-Hole-Domino-free, chordal, AT-free, strongly chordal, chordal bipartite, and distance hereditary graphs. With one shot we obtained many known results on distances in particular graph classes.

We conclude this paper with few open problems.

1. For which other graph classes (with special vertex orderings) can this approach give a good approximation of distances?
2. Can one characterize those vertex orderings along which the APASP algorithm will compute the exact distances (or approximate distances with an additive one-sided error of at most 1, 2, ...)?
3. Can the maximum-neighbor path concept developed here be used for designing good routing and/or distance labeling schemes in those graph classes? (For definitions and concepts of routing schemes and distance labeling schemes see the recent book of Peleg [31].)

Acknowledgments

This research was partially supported by the DFG. The author is grateful to an anonymous referee for many helpful suggestions.

References

[1] D. Aingworth, C. Chekuri, P. Indyk, R. Motwani, Fast estimation of diameter and shortest paths (without matrix multiplications), SIAM J. Comput. 28 (1999) 1167–1181.

[2] N. Alon, Z. Galil, O. Margalit, On the exponent of the all pairs shortest path problem, in: Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science, 1991, pp. 569–575.
[3] E.M. Arkin, J.S.B. Mitchell, S. Suri, Optimal link path queries in a simple polygon, in: Proceedings of the 3rd Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 1992), 1992, pp. 269–279.
[4] M.J. Atallah, D.Z. Chen, D.T. Lee, An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications, in: Proceedings of the European Symposium on Algorithms, in: Lecture Notes in Comput. Sci, vol. 726, 1993, pp. 13–24.
[5] B. Awerbuch, B. Berger, L. Cowen, D. Peleg, Near-linear cost sequential and distributed constructions of sparse neighborhood covers, in: Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, 1993, pp. 638–647.
[6] V. Balachandhran, C. Pandu Rangan, All-pairs-shortest length on strongly chordal graphs, Discrete Appl. Math. 69 (1996) 169–182.
[7] H.-J. Bandelt, H.M. Mulder, Distance-hereditary graphs, J. Combin. Theory Ser. B 41 (1986) 182–208.
[8] J. Basch, S. Khanna, R. Motwani, On diameter verification and Boolean matrix multiplication, Technical Report STAN-CS-95-1544, Department of Computer Science, Stanford University, 1995.
[9] A. Brandstädt, V.D. Chepoi, F.F. Dragan, The algorithmic use of hypertree structure and maximum neighborhood orderings, Discrete Appl. Math. 82 (1998) 43–77.
[10] A. Brandstädt, V.D. Chepoi, F.F. Dragan, Distance approximating trees for chordal and dually chordal graphs, J. Algorithms 30 (1999) 166–184.
[11] A. Brandstädt, V.B. Le, J.P. Spinrad, Graph Classes: A Survey, SIAM Monogr. Discrete Math. Appl., 1999.
[12] L. Chen, Solving the shortest-paths problem on bipartite permutation graphs efficiently, Inform. Process. Lett. 55 (1995) 259–264.
[13] V.D. Chepoi, F.F. Dragan, A note on distance approximating trees in graphs, European J. Combin. 21 (2000) 761–766.
[14] E. Cohen, Fast algorithms for constructing t-spanners and paths with stretch t (extended abstract), in: Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, 1993, pp. 648–658.
[15] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progression, in: Proceedings of the 19th ACM Symposium on Theory of Computing, 1987, pp. 1–6.
[16] D.G. Corneil, S. Olariu, L. Stewart, Linear time algorithms for dominating pairs in asteroidal triple-free graphs, SIAM J. Comput. 28 (1999) 1284–1297.
[17] E. Dahlhaus, Optimal (parallel) algorithms for the all-to-all vertices distance problem for certain graph classes, in: Proceedings of the International Workshop "Graph-Theoretic Concepts in Computer Science", in: Lecture Notes in Comput. Sci., vol. 657, 1992, pp. 60–69.
[18] D. Dor, S. Halperin, U. Zwick, All pairs almost shortest paths, in: Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 452–461.
[19] F.F. Dragan, Almost diameter of a house-hole free graph in linear time via LexBFS, Discrete Appl. Math. 95 (1999) 223–239.
[20] F.F. Dragan, F. Nicolai, A. Brandstädt, LexBFS-orderings and powers of graphs, in: Proceedings of the International Workshop "Graph-Theoretic Concepts in Computer Science", in: Lecture Notes in Comput. Sci., vol. 1197, 1997, pp. 166–180.
[21] H. Everett, D.G. Corneil, Recognizing visibility graphs of spiral polygons, J. Algorithms 11 (1990) 1–26.
[22] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
[23] K. Han, C.N. Sekharan, R. Sridhar, Unified all-pairs shortest path algorithms in the chordal hierarchy, Discrete Appl. Math. 77 (1997) 59–71.
[24] C.-W. Ho, J.-M. Chang, Solving the all-pairs-shortest-length problem on chordal bipartite graphs, Inform. Process. Lett. 69 (1999) 87–93.
[25] B. Jamison, S. Olariu, On the semi-perfect elimination, Adv. in Appl. Math. 9 (1988) 364–376.
[26] A. Lubiw, Doubly lexical orderings of matrices, SIAM J. Comput. 16 (1987) 854–879.
[27] P. Mirchandani, A simple $O(n^2)$ algorithm for the all-pairs shortest path problem on an interval graph, Networks 27 (1996) 215–217.
[28] R. Motwani, A. Ragunathan, H. Saran, Perfect graphs and orthogonally convex covers, SIAM J. Discrete Math. 2 (1989) 371–392.
[29] R. Motwani, A. Ragunathan, H. Saran, Covering orthogonal polygons with star polygons: the perfect graphs approach, J. Comput. System Sci. 40 (1990) 19–48.

[30] R. Paige, R.E. Tarjan, Three partition refinement algorithms, SIAM J. Comput. 16 (1987) 973–989.
[31] D. Peleg, Distributed Computing—A Locality-sensitive Approach, SIAM, Philadelphia, PA, 2000.
[32] R. Ravi, M.V. Marathe, C. Pandu Rangan, An optimal algorithm to solve the all-pair shortest path problem on interval graphs, Networks 22 (1992) 21–35.
[33] R. Seidel, On the all-pair-shortest-path problem, in: Proceedings of the 24th ACM Symposium on Theory of Computing, 1992, pp. 745–749.
[34] J.P. Spinrad, Doubly lexical ordering of dense 0–1-matrices, Inform. Process. Lett. 45 (1993) 229–235.
[35] D. Rose, R.E. Tarjan, G. Lueker, Algorithmic aspects on vertex elimination on graphs, SIAM J. Comput. 5 (1976) 266–283.
[36] S. Pettie, A faster all-pairs shortest path algorithm for real-weighted sparse graphs, in: Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP 2002), Malaga, Spain, July 8–13, 2002, in: Lecture Notes in Comput. Sci., vol. 2380, 2002, pp. 85–97.
[37] S. Pettie, On the comparison-addition complexity of all-pairs shortest paths, in: Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC 2002), Vancouver, BC, Canada, November 21–23, 2002, in: Lecture Notes in Comput. Sci., vol. 2518, 2002, pp. 32–43.
[38] S. Pettie, V. Ramachandran, Computing shortest paths with comparisons and additions, in: Proceedings of the 13th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2002), San Francisco, CA, USA, January 6–8, 2002, ACM/SIAM, 2002, pp. 267–276.
[39] S. Pettie, A new approach to all-pairs shortest paths on real-weighted graphs, Theoret. Comput. Sci. 312 (2004) 47–74.
[40] R. Sridhar, K. Han, N. Chandrasekharan, Efficient algorithms for shortest distance queries on special classes of polygons, Theoret. Comput. Sci. 140 (1995) 291–300.
[41] R. Sridhar, D. Joshi, N. Chandrasekharan, Efficient algorithms for shortest distance queries on interval, directed path and circular arc graphs, in: Proceedings of the 5th International Conference on Computing and Information, 1993, pp. 31–35.
[42] M. Thorup, Undirected single-source shortest paths with positive integer weights in linear time, J. ACM 46 (1999) 362–394.
[43] M. Thorup, Floats, integers, and single source shortest paths, J. Algorithms 35 (2000) 189–201.
[44] M. Thorup, Compact oracles for reachability and approximate distances in planar digraphs, in: Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001), Las Vegas, Nevada, USA, October 14–17, 2001, IEEE Computer Society, 2001, pp. 242–251.
[45] M. Thorup, Integer priority queues with decrease key in constant time and the single source shortest paths problem, in: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC 2003), San Diego, CA, USA, June 9–11, 2003, ACM, 2003, pp. 149–158.
[46] M. Thorup, Fully-dynamic all-pairs shortest paths: faster and allowing negative cycles, in: Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT 2004), Humlebaek, Denmark, July 8–10, 2004, in: Lecture Notes in Comput. Sci., vol. 3111, 2004, pp. 384–396.
[47] M. Thorup, U. Zwick, Approximate distance oracles, in: Proceedings on the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001), Heraklion, Crete, Greece, July 6–8, 2001, ACM, 2001, pp. 183–192.