Estimating All Pairs Shortest Paths in Restricted Graph Families: A Unified Approach
(Extended Abstract)

Feodor F. Dragan

Dept. of Computer Science, Kent State University, Kent, Ohio 44242, USA
dragan@cs.kent.edu

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.

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 [1,2,4,6,12], [15,29] for general (unweighted, undirected) graphs, and [3,5,7,10,11,14,20,21,24], [28,32,33] for special graph classes). For general graphs, the best result known is by Seidel [29], 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 current best matrix multiplication algorithm is due to Coppersmith and Winograd [13] 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 the 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

A. Brandstädt and V.B. Le (Eds.): WG 2001, LNCS 2204, pp. 103–116, 2001.

by Basch et al [6]. 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 $O(n^{3-\epsilon})$ time algorithm for the APSP problem would be a major breakthrough. Nowadays, many researchers take approach to consider the all pairs *almost* shortest path (APASP) problem in general graphs or to design (optimal) $O(n^2)$ time algorithm for special well structured graph classes (which are interesting from practical point of view).

Awerbuch et al [4] and Cohen [12] 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 $t \geq 4$. 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 [15] improved the result of Aingworth et al [1] and obtained an $\tilde{O}(\min\{n^{3/2}m^{1/2}, n^{7/3}\})$ time algorithm for finding all distances in unweighted, undirected graphs with an additive one-sided error of at most 2 (here, $\tilde{O}(f)$ means $O(f \text{polylog } n)$). 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 [15]).

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 [3,24,28,33], circular arc graphs [3,33], permutation graphs [14], bipartite permutation graphs [10], strongly chordal graphs [5,14,20], chordal bipartite graphs [21], distance-hereditary graphs [14], and dually chordal graphs [7]. The optimal parallel algorithms for the APSP problem for some certain classes of graphs can be found in [10,14,20,32].

The approach to the shortest path and other optimization problems in 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. Motwani et al [25,26] studied two classes of rectilinear polygons and showed that the visibility graphs corresponding to them are *chordal* or *weakly chordal*, respectively. More recently, Everett and Corneil [18] have shown that the visibility graphs of spiral polygons are *interval graphs* (the definition of the various families of graphs will be presented in the next section).

This motivated researchers to look at distances in chordal graphs [8,11,20,32]. Han et al showed in [20] 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, and present few subclasses of chordal graphs for which G^2 can be computed more efficiently. From results in [8] it follows that within $O(n^2)$ time bound one can compute 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 [32] that using a more sophisticated linear time preprocessing of a chordal graph, the distance between any two vertices which is at most 1 greater than the shortest distance can be computed in $O(1)$ time.

In this paper 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.

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 this 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.

2 Preliminaries and the Algorithm

The *(open) neighborhood* of a vertex v is the set $N(v) = \{u \in V : uv \in E\}$ 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

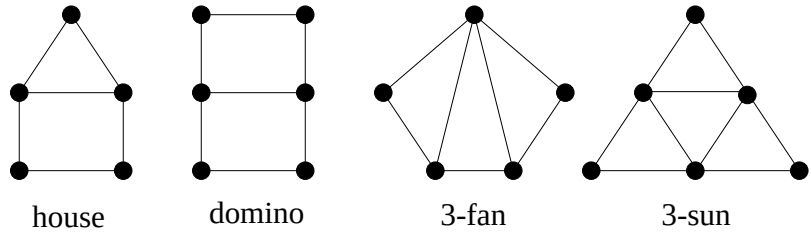

Fig. 1.

weakly chordal graph is called *chordal bipartite*. An *interval* graph is the intersection graph of intervals of a line. 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 Figure 1). 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 [9,19]

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 $mn(v)$ the *maximum neighbor* of v, i.e., the vertex of $N[v]$ with maximum number in σ.

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** $\hat{d}(v_i, v_j) := \hat{d}(mn(v_i), v_j) + 1$;
return distances $\hat{d}$.

We assume that ordering σ used in the algorithm has the property that for all v_i with $i < n$, $mn(v_i) \neq v_i$ (otherwise 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 [14]) and in dually chordal graphs (see [7]). 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** $(u$,mn–path$(mn(u),v))$
else return (mn–path$(u,mn(v)),v)$

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 [31] and *lexical orderings* of Lubiw [23].

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. Moreover, $\{a_1, \cdots, a_l\} < \{b_1, \cdots, b_k\}$ is an abbreviation for $a_i < b_j$ $(i = 1, \cdots, l;\ j = 1, \cdots, k)$. Let u be a vertex of G. We define the layers of G with respect to vertex u as follows: $L_i(u) = \{v : d(u,v) = i\}$ 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 $N(v) \bigcap L_{q-1}$ with the largest number in σ, i.e., $f(v) = mn(v)$.

(P3) If $x > y$, then either $mn(x) > mn(y)$ or $mn(x) = mn(y)$.

(P4) If $x, y, z \in L_j$, $x > y > z$ and $mn(x)z \in E$, then $mn(x) = mn(y) = mn(z)$ (in particular, $mn(x)y \in E$).

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 as-

signed 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.[22]). Moreover, any ordering fulfilling (P5) can be generated by LexBFS [17].

(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 [23] 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 [19], while to date the fastest method – doubly lexical ordering of the (closed) neighborhood matrix of G [23] – producing a lexical ordering of G takes $O(m \log n)$ [27] or $O(n^2)$ [30] 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 itself 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 $t \geq 2$ and $s \geq 0$ such that*
(1) for all $x, y \in V$ with $x < y$ and $d(x, y) \geq t$, $d(x, y) = d(mn(x), y) + 1$ holds,
(2) for all $x, y \in V$ with $d(x, y) < t$, $\hat{d}(x, y) \leq d(x, y) + s$ holds.
Then $\hat{d}(x, y) \leq d(x, y) + s$ holds for all $x, y \in V$.

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

Let $P = (x_0, x_1, \cdots, x_{k-1}, x_k)$ 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 ($1 \leq i < k$) 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

$P = (x_0, \cdots, x_k)$ 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 ([16]). *G has no induced $P_4 = (c, a, b, d)$ with $\{a, b\} < \{c, d\}$ in σ.*

Corollary 1. *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 ([16]).
(1) Every rightmost shortest path of G is either monotonic or convex.
(2) Let $P = (x_0, \cdots, x_k)$ be a rightmost shortest path in G which is convex and x_i be the switching point of P. Then $d(x_0, x_i) \geq d(x_k, x_i)$ if $x_0 < x_k$.

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

Proof. Let $P = (x, x_1, x_2, \cdots, y)$ be a rightmost shortest path in G connecting vertices x and y. Since $x < y$ and $d(x, y) \geq 3$, by Lemma 3, we have $x < x_1 < x_2$. Assume $x_1 \neq mn(x)$. Vertices $v = mn(x)$ 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 $v = mn(x)$ 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 $k \geq 4$, 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 $d(x, y) < d(mn(x), y) + 1$. Figure 2 gives such a weakly chordal graph for $k = 4$ (this example can easily be extended to a larger k).

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

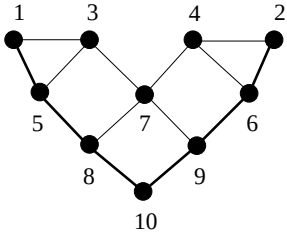

Fig. 2. A weakly chordal graph with a LexBFS–ordering; $d_{1,2}= d_{2,5} = 4$.

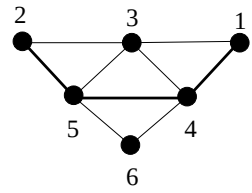

Fig. 3. A LexBFS–ordering of a 3-sun such that $\hat{d}_{1,2}-d_{1,2} = 1$.

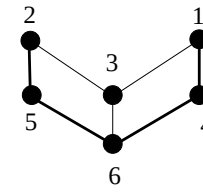

Fig. 4. A LexBFS–ordering of a domino such that $\hat{d}_{1,2}-d_{1,2} = 2$.

Proof. (1) Clearly, if $mn(x)y \in E$, then $\hat{d}(mn(x), y)=1$ and $\hat{d}(x, y)= \hat{d}(mn(x), y) +1 = 2 = d(x, y)$. So, assume that $mn(x)y \notin E$ and there is a vertex $z \in N(x) \bigcap N(y)$ such that $z < y$. For path $(mn(x), x, z, y)$ we have $\{x, z\} < \{mn(x), y\}$. Hence, by Lemma 2, $mn(x)$ 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 $mn(x)y \notin E$ and for the vertex $z \in N(x) \bigcap N(y)$ with $mn(x)z \in E$, $z > y$ holds. We have $d(mn(x), y) = 2$ and $y < z < mn(x)$. Hence, by (1), $\hat{d}(y, mn(x)) = d(y, mn(x))$, and therefore $\hat{d}(x, y) = \hat{d}(mn(x), y) + 1 = d(mn(x), y) + 1 = 3 = d(x, y) + 1$. □

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

Proof. Let $v = mn(x)$, $u = mn(z)$ and $w = mn(y)$ 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 $u \geq w > v$ 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, $d(y, u) = 2$ and we can apply Lemma 5 to $y < z < u$ and get $d(y, u) = \hat{d}(y, u) = 2$. That is, vertex $w = mn(y)$ 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 adjacent neither to v nor to z, i.e. vertices x, y, z, v, u, w form an induced domino.

Now consider maximum neighbors $h = mn(v)$, $s = mn(u)$ and $t = mn(w)$ of v, u and w, respectively. We had $d(v, w) = 2$, $v < w < u$ and $u \in N(v) \bigcap N(w)$. 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 $v = mn(x)$, $u = mn(z)$, $w = mn(y)$ 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 adjacent neither to x nor to z because $h > u = mn(z) > v = mn(x)$).

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*

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

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*

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

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 ([32]). *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 3 shows a chordal graph with a LexBFS–ordering where the difference between $\hat{d}$ and d can achieve 1, Figure 4 gives a similar example for HH-free graphs. In all figures dark edges indicate the maximum-neighbor path between vertices numbered 1 and 2.

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 $d(v, u) = 2$, $v < u$, $mn(v)u \notin E$ and there exists a vertex $z \in N(v) \bigcap N(u)$ such that $mn(v)z \in E$, $z > u$.

Consider the maximum neighbor $w = mn(u)$ of u. Since $v < u$ and $mn(v)u \notin E$, by property (P3), we have $mn(v) < w$, i.e., $v < u < z < mn(v) < w$, and $wv \notin E$. By Lemma 2, vertex w has to be adjacent to $mn(v)$ or z. If

$mn(v)w \in E$ 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 $mn(v)$ but is adjacent to z. Now we apply Corollary 1 to $z < \{mn(v), w\}$, $mn(v)w \notin E$ and get a vertex t such that $t > w$, $mn(v)t, tw \in E$ and $zt \notin E$. Since $t > w > mn(v)$, vertex v is adjacent neither to t nor to w. Thus, we have created an induced house in G formed by $v, z, mn(v), t, w$, contradicting the fact G is distance-hereditary. □

Corollary 5 ([14]). *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 is 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 $v_iv_j \in E(G)$ **then** $d(v_i, v_j) := 1$
    else if $v_iv_j \in E(G^2) \setminus E(G)$ **then** $d(v_i, v_j) := 2$
      else $d(v_i, v_j) := d(mn(v_i), v_j) + 1$;
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 ([20]). *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 $mn(v)u \notin E$. We claim that this case is impossible.

It is well-known [23] that any lexical ordering of a strongly chordal graph is a strong elimination ordering. Consider a vertex $z \in N(v) \bigcap N(u)$. We have $zv, zu, mn(v)v \in E$, $z < mn(v), v < u$. Since $mn(v)u \notin E$, a contradiction with σ is a strong elimination ordering arises. □

Corollary 8 ([14,5]). *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 ([3,24,28,33]). *The all pairs shortest path problem in interval graphs can be solved in $O(n^2)$ time.*

In a similar way we can prove the following (proof is omitted).

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 $v, u \in V$.*

Corollary 10 ([21]). *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, which we present here without proofs. It is interesting to note that even simple BFS–orderings succeed in getting a good approximation for distances in k-chordal graphs (for small $k \geq 4$).

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

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

Corollary 11. *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 12. *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 13. *All distances in weakly chordal graphs with an additive one-sided error of at most 3 can be found in $O(n^2)$ time.*

This result can be strengthened further for AT-free graphs. We can prove the following.

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

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

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

In Figures 5, 6 and 7 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 figures are even LexBFS–orderings,

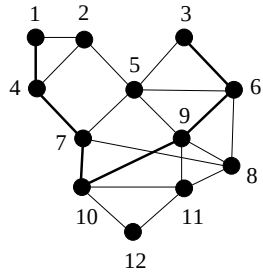

Fig. 5. A weakly chordal graph with a LexBFS–ordering; $\hat{d}_{1,3} - d_{1,3} = 3$.

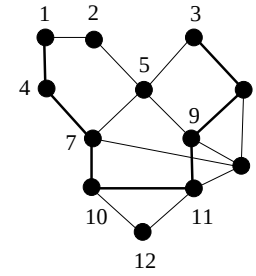

Fig. 6. A 5-chordal graph with a LexBFS–ordering; $\hat{d}_{1,3} - d_{1,3} = 4$.

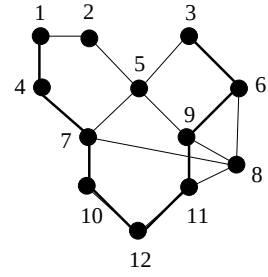

Fig. 7. A 6-chordal graph with a LexBFS–ordering; $\hat{d}_{1,3} - d_{1,3} = 5$.

considering LexBFS instead of BFS in Theorem 7 will not give any better bound (for $k \geq 4$). Furthermore, the domino with a LexBFS–ordering presented in Figure 4 shows that the bound in Theorem 8 for AT-free graphs is tight, too.

Note that distances in k-chordal graphs were already considered in [11]. It was shown that for any k-chordal graph $G = (V, E)$ there exists a tree $T = (V, F)$ such that $|d_G(u, v) - d_T(u, v)| \leq \lfloor k/2 \rfloor + \alpha$ 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 $O(n + m)$.

Acknowledgements: This research was partially supported by the DFG.

References

1. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani, Fast estimation of diameter and shortest paths (without matrix multiplications), *SIAM J. on Computing*, 28 (1999), 1167–1181.
2. N. Alon, Z. Galil, and O. Margalit, On the exponent of the all pairs shortest path problem, Proceedings of the *32nd Annual IEEE Symposium on Foundations of Computer Science*, 1991, 569–575.
3. M. J. Atallah, D. Z. Chen, and D. T. Lee, An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications, Proceedings of the *European Symposium on Algorithms, Lecture Notes in Computer Science*, 726 (1993), 13–24.
4. B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, Near-linear cost sequential and distributed constructions of sparse neighborhood covers, Proceedings of the *34th Annual IEEE Symposium on Foundations of Computer Science,* 1993, 638–647.
5. V. Balachandhran, and C. Pandu Rangan, All-pairs-shortest length on strongly chordal graphs, *Discrete Applied Math.*, 69 (1996), 169–182.
6. J. Basch, S. Khanna, and R. Motwani, On diameter verification and Boolean matrix multiplication, *Technical Report STAN-CS-95-1544*, Department of Computer Scince, Stanford University, 1995.
7. A. Brandstädt, V. D. Chepoi, and F. F. Dragan, The algorithmic use of hypertree structure and maximum neighborhood orderings, *Discrete Applied Math.*, 82 (1998), 43–77.
8. A. Brandstädt, V. D. Chepoi, and F. F. Dragan, Distance approximating trees for chordal and dually chordal graphs, *Journal on Algorithms,* 30 (1999), 166–184.
9. A. Brandstädt, V. B. Le, and J. P. Spinrad, Graph classes: a survey, *SIAM monographs on discrete mathematics and applications,* 1999.
10. L. Chen, Solving the shortest-paths problem on bipartite permutation graphs efficiently, *Information Processing Letters*, 55 (1995), 259–264.
11. V. D. Chepoi, and F. F. Dragan, A note on distance approximating trees in graphs, *Europ. J. Combinatorics*, 21 (2000), 761–766.
12. E. Cohen, Fast algoruthms for constructing t-spanners and paths with stretch t (extended abstract), Proceedings of the *34th Annual IEEE Symposium on Foundations of Computer Science,* 1993, 648–658.
13. D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progression, Proceedings of the *19th ACM Symposium on Theory of Computing*, 1987, 1–6.

14. E. Dahlhaus, Optimal (parallel) algorithms for the all-to-all vertices distance problem for certain graph classes, Proceedings of the *International Workshop "Graph-Theoretic Concepts in Computer Science"*, *Lecture Notes in Computer Science*, 657 (1992), 60–69.
15. D. Dor, S. Halperin, and U. Zwick, All pairs almost shortest paths, Proceedings of the *37th Annual IEEE Symposium on Foundations of Computer Science*, 1996, 452–461.
16. F. F. Dragan, Almost diameter of a hose-hole free graph in linear time via LexBFS, *Discrete Applied Math.*, 95 (1999), 223-239.
17. F.F. Dragan, F. Nicolai and A. Brandstädt, LexBFS–orderings and powers of graphs, Proceedings of the *International Workshop "Graph-Theoretic Concepts in Computer Science"*, *Lecture Notes in Computer Science*, 1197 (1997), 166–180.
18. H. Everett and D.G. Corneil, Recognizing visibility graphs of spiral polygons, *Journal on Algorithms,* 11 (1990), 1–26.
19. M. C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, *Academic Press*, New York, 1980.
20. K. Han, Chandra N. Sekharan and R. Sridhar, Unified all-pairs shortest path algorithms in the chordal hierarchy, *Discrete Applied Math.*, 77 (1997), 59–71.
21. Chin-Wen Ho and Jou-Ming Chang, Solving the all-pairs-shortest-length problem on chordal bipartite graphs, *Information Processing Letters*, 69 (1999), 87–93.
22. B. Jamison and S. Olariu, On the semi–perfect elimination, *Advances in Applied Math.,* 9 (1988), 364–376.
23. A. Lubiw, Doubly lexical orderings of matrices, *SIAM J. on Computing,* 16 (1987), 854–879.
24. P. Mirchandani, A simple $O(n^2)$ algorithm for the all-pairs shortest path problem on an interval graph, *Networks*, 27 (1996), 215–217.
25. R. Motwani, A. Ragunathan and H. Saran, Perfect graphs and ortogonally convex covers, *SIAM J. Discrete Math.,* 2 (1989), 371–392.
26. R. Motwani, A. Ragunathan and H. Saran, Covering orthogonal polygons with star polygons: the perfect graphs approach, *J. of Computer and System Sciences*, 40 (1990), 19–48.
27. R. Paige and R.E. Tarjan, Three partition refinement algorithms, *SIAM J. on Computing*, 16 (1987), 973–989.
28. R. Ravi, M. V. Marathe, and C. Pandu Rangan, An optimal algorithm to solve the all-pair shortest path problem on interval graphs, *Networks*, 22 (1992), 21–35.
29. R. Seidel, On the all-pair-shortest-path problem, Proceedings of the *24th ACM Symposium on Theory of Computing*, 1992, 745–749.
30. J.P. Spinrad, Doubly lexical ordering of dense 0–1– matrices, *Information Processing Letters*, 45 (1993), 229–235.
31. D. Rose, R.E. Tarjan and G. Lueker, Algorithmic aspects on vertex elimination on graphs, *SIAM J. on Computing*, 5 (1976), 266–283.
32. R. Sridhar, K. Han, and N. Chandrasekharan, Efficient algorithms for shortest distance queries on special classes of polygons, *Theoretical Computer Science*, 140 (1995), 291–300.
33. R. Sridhar, D. Joshi, and N. Chandrasekharan, Efficient algorithms for shortest distance queries on interval, directed path and circular arc graphs, Proceedings of the *5th International Conference on Computing and Information*, 1993, 31–35.