Polynomial Time Algorithms for Computing a Minimum Hull Set in Distance-Hereditary and Chordal Graphs*

Mamadou Moustapha Kanté and Lhouari Nourine

Clermont-Université, Université Blaise Pascal, LIMOS, CNRS, France
{mamadou.kante,nourine}@isima.fr

Abstract. We give linear and polynomial time algorithms for computing a minimum hull-set in distance-hereditary and chordal graphs respectively. Prior to our result a polynomial time algorithm was only known for sub-classes of considered graph classes. Our techniques allow us to give at the same time a linear time algorithm for computing a minimum geodetic set in distance-hereditary graphs.

1 Introduction

In this paper we consider convexity related to shortest paths in graphs. For an undirected graph G and a subset X of the vertex-set of G, let $I[X]$ be the set of vertices that lie in a shortest path between two vertices of X. The *closure* of X is the smallest $X' \supseteq X$ such that $I[X'] = X'$. The *hull number* [16] of a graph G is defined as the minimum k such that there exists a set X of size k whose closure is the set of vertices of G. The notion of hull number has been introduced in [16] and has attracted many interest in the years [1,3,12,13]. Computing the minimum hull number of graphs is NP-complete [1,12] and polynomial time algorithms have been proposed for some graph classes: proper interval graphs, cographs, split graphs [12], cobipartite graphs [1].

In this paper we give polynomial time algorithms for computing a minimum hull set in distance-hereditary and chordal graphs. The computational complexity of the hull number of chordal graphs has been left open since [12] and to our knowledge no polynomial time algorithm for the computation of the hull number of distance-hereditary graphs is known. Surprisingly, the related notion *geodetic number* has been proven to be NP-complete in chordal graphs [14] and a polynomial time algorithm for interval graphs is open (polynomial time algorithms for split graphs and proper interval graphs are given in [14] and [15]).

Our linear time (in the size of the graph) algorithm for distance-hereditary graphs can be summarised as follows. We will first give a *monadic second-order* formula $HullSet(X)$ that holds in a connected distance-hereditary graph G if and only if X is a hull set of G. We will then use Courcelle et al.'s theorem

* M.M. Kanté is supported by the DORSO project, and L. Nourine by the DAG project, both of "Agence Nationale Pour la Recherche".

P. van Emde Boas et al. (Eds.): SOFSEM 2013, LNCS 7741, pp. 268–279, 2013.

stating that monadic second-order properties can be solved in linear time in distance-hereditary graphs (see [7]). In order to write the formula $HullSet(X)$ we will express in monadic second-order logic the property "z is in a shortest path between x and y" and we will use for that the well-known notion of *split decomposition* [8] and a characterisation of the property "being in a shortest path" by means of split decomposition.

The algorithm for chordal graphs is based upon the notion of *functional dependencies* which are constraints used in relational database design and specially in normalisation process [5].

Summary. Section 2 is devoted to preliminaries and basic facts. A linear time algorithm for distance-hereditary graphs, based on logical tools, is given in Section 3. Section 4 is devoted to the polynomial time algorithm for chordal graphs. We finish by some concluding remarks and open questions in Section 5.

2 Preliminaries

Graphs. If A and B are two sets, $A \backslash B$ denotes the set $\{x \in A \mid x \notin B\}$. The power set of a set V is denoted by 2^V. The size of a set A is denoted by $|A|$.

We refer to [10] for our graph terminology. A *graph* G is a pair (V_G, E_G) where V_G is the set of vertices and $E_G \subseteq (V_G \times V_G) \setminus (\{(x,x) \mid x \in V_G\})$ is the set of edges. A graph G is said to be *undirected* if $(x,y) \in E_G$ implies $(y,x) \in E_G$; an edge (x,y) is hence written xy (equivalently yx). For a graph G, we denote by $G[X]$, called the subgraph of G induced by $X \subseteq V_G$, the graph $(X, E_G \cap (X \times X))$. The *size* of a graph G, denoted by $\|G\|$, is defined as $|V_G| + |E_G|$.

A *path of length* k in a graph G from the vertex x to the vertex y is a sequence $(x = x_0, x_1, \ldots, x_k = y)$ such that the set $\{(x_i, x_{i+1}) \mid 0 \leq i \leq k-1\}$ is a subset of E_G; this path is said *chordless* if $(x_i, x_j) \notin E_G$ whenever $j > i+1$ or $j < i-1$. It is worth noticing that whenever G is undirected, if $P := (x_0, \ldots, x_k)$ is a path from x to y, $(x_k, \ldots, x_0)$ is also a path from y to x and we will say in this case that P is between x and y. A graph G is said *strongly connected* if for every pair (x,y) of vertices there exists a path from x to y and also a path from y to x. A *strongly connected component* of a graph G is a maximal strongly connected induced subgraph of G. Strongly connected undirected graphs are simply said to be *connected* and their strongly connected components are called *connected components*.

The *distance* between two vertices x and y in an undirected graph G, denoted by $d_G(x,y)$, is the minimum k such that there exists a path of length k between x and y; if no such path exists then $d_G(x,y) = \infty$ (this does happen if G is not connected). Any path between two vertices x and y of length $d_G(x,y)$ is called a *shortest path* and is by definition a chordless path.

An undirected graph G is said *complete* if $E_G = (V_G \times V_G) \setminus (\{(x,x) \mid x \in V_G\})$ (it is denoted K_n if it has n vertices), and it is called a *cycle of length* n if its edge-set is the set $\{x_i x_{i+1} \mid 1 \leq i \leq n-1\} \cup \{x_1 x_n\}$ with $(x_1, \ldots, x_n)$ an ordering of its vertex-set. An undirected graph G is called *distance-hereditary* if for every

two vertices x and y of G all the chordless paths between x and y have the same length, and it is called *chordal* if it has no induced cycle of length greater than or equal to 4. A vertex x of an undirected graph G is called *simplicial* if $G[\{y \mid xy \in E_G\}]$ is a complete graph. A *tree* is an acyclic connected undirected graph and a *star* is a tree with a distinguished vertex adjacent to the other vertices (it is denoted S_n if it has n vertices).

Betweenness Relations. Let V be a finite set. A *betweenness relation on* V is a ternary relation $B \subseteq V^3$ such that for every $x, y, z \in V$, we have $B(x, z, y)$ holds if and only if $B(y, z, x)$ holds; z is said to be *between* x and y. Several betweenness relations, particularly in graphs, are studied in the literature (see [4,20]).

Let B be a betweenness relation on a finite set V. For every subset X of V, we let $B^o(X)$ be $\bigcup_{x,y \in X}\{z \in V \mid B(x, z, y)\}$. A subset X of V is said B-*convex* if $B^o(X) = X$ and its B-*convex hull*, denoted by $B^+(X)$, is the smallest B-convex set that contains X. A subset X of V is a B-*hull set* (resp. B-*geodetic set*) if $B^+(X) = V$ (resp. $B^o(X) = V$).

In this paper we deal with the following betweenness relation. For every undirected graph G, we define the betweenness relation $\mathcal{SP}_G$ on V_G where $\mathcal{SP}_G(x, z, y)$ holds if and only if z is in a shortest path between x and y. We are interested in computing the *hull number* of an undirected graph G defined as the size of a minimum $\mathcal{SP}_G$-hull set of V_G [16]. The computation of the hull number of a graph is NP-complete [12] and polynomial time algorithms exist for some graph classes (see for instance [1,12,13]). We give polynomial time algorithms for distance-hereditary and chordal graphs. Our techniques will allow us to derive a linear time algorithm for computing the *geodetic number* of a distance-hereditary graph G, defined as the $\mathcal{SP}_G$-geodetic set of V_G [14].

Fact 1. *A $\mathcal{SP}_G$-hull set of a non connected undirected graph G is the union of $\mathcal{SP}_G$-hull sets of its connected components. Similarly for $\mathcal{SP}_G$-geodetic sets.*

Proof. This follows from the fact that for every triple x, y, z of V_G, z is in a shortest path between x and y if and only if $d_G(x, y) = d_G(x, z) + d_G(z, y)$.

Fact 2. *Any $\mathcal{SP}_G$-hull set (or $\mathcal{SP}_G$-geodetic set) of an undirected graph G must contain the set of simplicial vertices of G.*

Proposition 3. *Let G be a connected undirected graph and let x be a vertex of G. Then, we can compute all the sets $\mathcal{SP}^o_G(\{x, y\})$, for all vertices $y \in V_G \setminus \{x\}$, in time $O(\|G\| + \sum_{y \in V_G \setminus \{x\}} \left|\mathcal{SP}^o_G(\{x, y\})\right|)$.*

Monadic Second-Order Logic. We refer to [7] for more information. A *relational signature* is a finite set $\mathcal{R} := \{R, S, T, \ldots\}$ of relation symbols, each of which given with an arity $ar(R) \geq 1$. A *relational $\mathcal{R}$-structure* $\mathfrak{A}$ is a tuple $(A, (R_{\mathfrak{A}})_{R \in \mathcal{R}})$ with $R_{\mathfrak{A}} \subseteq A^{ar(R)}$ for every $R \in \mathcal{R}$ and A is called its domain. Examples of relational structures are graphs that can be seen as relational

$\{edg\}$-structures with $ar(edg) = 2$, *i.e.*, a graph G is seen as the relational $\{edg\}$-structure (V_G, edg_G) with V_G its set of vertices and $edg_G(x, y)$ holds if and only if $(x, y) \in E_G$.

We will use lower case variables $x, y, z, \ldots$ (resp. upper case variables $X, Y, Z, \ldots$) to denote elements of domains (resp. subsets of domains) of relational structures. Let $\mathcal{R}$ be a relational signature. The *atomic formulas over* $\mathcal{R}$ are $x = y$, $x \in X$ and $R(x_1, \ldots, x_{ar(R)})$ for $R \in \mathcal{R}$. The set $MS_{\mathcal{R}}$ of *monadic second-order formulas over* $\mathcal{R}$ is the set of formulas formed from atomic formulas over $\mathcal{R}$ with Boolean connectives $\wedge$, $\vee$, $\neg$, $\Longrightarrow$, $\Longleftrightarrow$, *element quantifications* $\exists x$ and $\forall x$, and *set quantifications* $\exists X$ and $\forall X$. An occurrence of a variable which is not under the scope of a quantifier is called a *free variable*. We will write $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ to express that the formula φ has $x_1, \ldots, x_m, Y_1, \ldots, Y_q$ as free variables and $\mathfrak{A} \models \varphi(a_1, \ldots, a_m, Z_1, \ldots, Z_q)$ to say that $\varphi(a_1, \ldots, a_m, Z_1, \ldots, Z_q)$ holds in $\mathfrak{A}$ when substituting $(a_1, \ldots, a_m) \in A^m$ to element variables $(x_1, \ldots, x_m)$ and $(Z_1, \ldots, Z_q) \in (2^A)^q$ to set variables $(Y_1, \ldots, Y_q)$ in $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$. The following is an example of a formula expressing that two vertices x and y are connected by a path

$$\forall X(x \in X \ \wedge \ \forall z, t(z \in X \ \wedge \ edg(z, t) \implies t \in X) \Longrightarrow y \in X).$$

If $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$ is a formula in $MS_{\mathcal{R}}$, we let opt_φ, with $opt \in \{\min, \max\}$, be the problem that consists in, given a relational $\mathcal{R}$-structure $\mathfrak{A}$, to finding a tuple $(Z_1, \ldots, Z_q)$ of $(2^A)^q$ such that

$$\sum_{1 \leq i \leq q} |Z_i| = opt \left\{ \sum_{1 \leq i \leq q} |W_j| \mid \mathfrak{A} \models \varphi(a_1, \ldots, a_m, W_1, \ldots, W_q) \right\}.$$

Many optimisation graph problems, *e.g.*, minimum dominating set, maximum clique, ..., correspond to opt_φ for some $MS_{\{edg\}}$ formula φ.

Let A be a finite set. An *A-coloured graph* is a graph with its edges and vertices labelled with elements in A. Let $\mathcal{R}_A$ be the relational signature $\{(edg_a)_{a \in A}, (nlab_a)_{a \in A}\}$ with $ar(edg_a) = 2$ and $ar(nlab_a) = 1$ for every $a \in A$. Every A-coloured graph G can be represented by the relational $\mathcal{R}_A$-structure $(V_G, (edg_{aG})_{a \in A}, (nlab_{aG})_{a \in A})$ where $edg_{aG}(x, y)$ holds if and only if $(x, y) \in E_G$ is labelled with $a \in A$ and $nlab_{aG}(x)$ holds if and only if $x \in V_G$ is labelled with $a \in A$.

Clique-width is a graph complexity measure introduced by Courcelle et al. and that is important in complexity theory (we refer to [7] for more information on clique-width). We recall the following important theorem that is the base of our algorithm for distance-hereditary graphs.

Theorem 4. *[7] Let A be a fixed finite set and let k be a fixed constant. For every $MS_{\mathcal{R}_A}$ formula $\varphi(x_1, \ldots, x_m, Y_1, \ldots, Y_q)$, opt_φ, for $opt \in \{\min, \max\}$, can be solved in time $O(f(k, |A|, \varphi) \cdot |V_G|)$, for some function f, in any A-coloured graph of clique-width at most k, provided the clique-width expression is given.*

3 Distance-Hereditary Graphs

We will first use the well-known notion of *split-decomposition* defined by Cunningham and Edmonds [8] to associate with every distance-hereditary graph G an A-coloured graph $\mathcal{S}_G$ for some finite set A. In a second step, we will prove that there exists a formula $Bet_{\mathcal{SP}}(x,z,y)$ in $MS_{\mathcal{R}_A}$ that holds in $\mathcal{S}_G$ if and only if z is in a shortest path in G between x and y. These two constructions combined with Theorem 4 and the next proposition will give rise to the linear time algorithm.

Proposition 5. *Let B be a betweenness relation on a finite set V and assume there exists a relational $\mathcal{R}$-structure $\mathfrak{A}$ with $V \subseteq A$, for some relational signature $\mathcal{R}$ that contains a relation $nlab_{\mathbf{V}}$ representing V. Assume also that there exists an $MS_{\mathcal{R}}$ formula $Bet(x,z,y)$ that holds in $\mathfrak{A}$ if and only if $B(x,z,y)$ holds. Then, there exist $MS_{\mathcal{R}}$ formulas $HullSet(X)$ and $GeodeticSet(X)$ expressing that X is a B-hull set and a B-geodetic set respectively.*

3.1 Split Decomposition

We will follow [6] (see also [18]) for our definitions. Two bipartitions $\{X_1, X_2\}$ and $\{Y_1, Y_2\}$ of a set V *overlap* if $X_i \cap Y_j \neq \emptyset$ for all $i, j \in \{1,2\}$. A *split* in a connected undirected graph G is a bipartition $\{X, Y\}$ of the vertex set V_G such that $|X|, |Y| \geq 2$, and there exist $X_1 \subseteq X$ and $Y_1 \subseteq Y$ such that $E_G = E_{G[X]} \cup E_{G[Y]} \cup X_1 \times Y_1$ (see Figure 1). A split $\{X, Y\}$ is *strong* if there is no other split $\{X', Y'\}$ such that $\{X, Y\}$ and $\{X', Y'\}$ overlap. Notice that not all graphs have a split. Those that do not have a split are called *prime*.

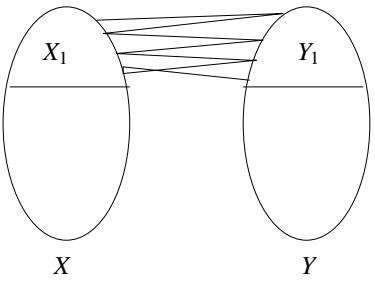

Fig. 1. A schematic view of a split

If $\{X, Y\}$ is a split of an undirected graph G, then we let $\mathcal{G}[X]$ and $\mathcal{G}[Y]$ be respectively $(X \cup \{h_X\}, E_{G[X]} \cup \{xh_X \mid x \in X \text{ and } \exists y \in Y,\ xy \in E_G\})$ and $(Y \cup \{h_Y\}, E_{G[Y]} \cup \{yh_Y \mid y \in Y \text{ and } \exists x \in X,\ xy \in E_G\})$ where h_X and h_Y are new vertices. The vertices h_X and h_Y are called *neighbour markers* of $\mathcal{G}[X]$

and $\mathcal{G}[Y]$. Notice that given $\mathcal{G}[X]$ and $\mathcal{G}[Y]$ with neighbour markers h_X and h_Y distinguished, we can reconstruct G as follows

$$V_G = (V_{\mathcal{G}[X]} \cup V_{\mathcal{G}[Y]}) \setminus \{h_X, h_Y\},$$
$$E_G = (E_{\mathcal{G}[X]} \cup E_{\mathcal{G}[Y]}) \setminus (\{xh_X \in E_{\mathcal{G}[X]}\} \cup \{xh_Y \in E_{\mathcal{G}[Y]}) \cup \{xy \mid x \in V_{\mathcal{G}[X]},\ y \in V_{\mathcal{G}[Y]} \text{ and } xh_X \in E_{\mathcal{G}[X]},\ yh_Y \in E_{\mathcal{G}[Y]}\}.$$

Fact 6. *Let $\{X, Y\}$ be a split in a connected undirected graph G and let $P := (x_0, \ldots, x_k)$ be a path in G.*

1. *If $x_0 \in X$ and $x_k \in Y$, then P is a shortest path if and only if there exists $x_i \in P$ such that $(x_0, \ldots, x_i, h_X)$ and $(h_Y, x_{i+1}, \ldots, x_k)$ are shortest paths in $\mathcal{G}[X]$ and $\mathcal{G}[Y]$ respectively.*
2. *If $x_0, x_k \in X$, then P is a shortest path if and only if either P is a shortest path in $\mathcal{G}[X]$ or $(x_0, \ldots, x_{i-1}, h_X, x_{i+1}, \ldots, x_k)$ is a shortest path in $\mathcal{G}[X]$ with $x_i \in Y$. Similarly for $x_0, x_k \in Y$.*

A *decomposition* of a connected undirected graph G is defined inductively as follows: $\{G\}$ is the only decomposition of size 1. If $\{G_1, \ldots, G_n\}$ is a decomposition of size n of G and G_i has a split $\{X, Y\}$, then $\{G_1, \ldots, G_{i-1}, \mathcal{G}_i[X], \mathcal{G}_i[Y], G_{i+1}, \ldots, G_n\}$ is a decomposition of size $n+1$. Notice that the decomposition process must terminate because the new graphs $\mathcal{G}_i[X]$ and $\mathcal{G}_i[Y]$ are smaller than G_i. The graphs G_i of a decomposition are called *blocks*. If two blocks have neighbour markers, we call them *neighbour blocks*.

A decomposition is *canonical* if and only if: (i) each block is either prime (called *prime block*), or is isomorphic to K_n (called *clique block*) or to a S_n (called *star block*) for $n \geq 3$, (ii) no two clique blocks are neighbour, and (iii) if two star blocks are neighbour, then either their markers are both centres or both not centres.

Theorem 7 ([8,9]). *Every connected undirected graph has a unique canonical decomposition, up to isomorphism. It can be obtained by iterated splitting relative to strong splits. This canonical decomposition can be computed in time $O(\|G\|)$ for every undirected graph G.*

The canonical decomposition of a connected undirected graph G constructed in Theorem 7 is called *split-decomposition* and we will denote it by $\mathcal{D}_G$.

3.2 Definition of $Bet_{\mathcal{SP}}(x, z, y)$ and the Linear Time Algorithm

We let A be the set $\{s, \epsilon, \mathbf{V}, \mathbf{M}\}$. For every connected undirected graph G we associate the A-coloured graph $\mathcal{S}_G$ where $V_{\mathcal{S}_G} = \bigcup_{G_i \in \mathcal{D}_G} V_{G_i}$, $E_{\mathcal{S}_G} = \left(\bigcup_{G_i \in \mathcal{D}_G} E_{G_i}\right) \cup \{xy \mid x, y \text{ are neighbour markers}\}$, a vertex x is labelled $\mathbf{V}$ if and only if $x \in V_G$, otherwise it is labelled $\mathbf{M}$, and an edge xy is labelled s if and only if $xy \in E_{G_i}$ for some $G_i \in \mathcal{D}_G$, otherwise it is labelled ϵ. Figure 2 gives an example of the graph $\mathcal{S}_G$.

A path $(x_0, x_1, \ldots, x_k)$ in $\mathcal{S}_G$ is said *alternating* if, for every $1 \leq i \leq k-1$, the labels of the edges $x_{i-1}x_i$ and x_ix_{i+1} are different.

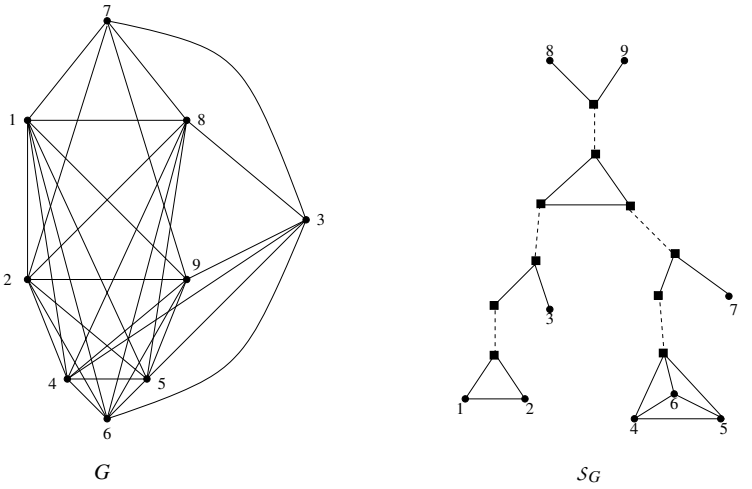

Fig. 2. Solid lines are edges labelled by s and the dashed ones are labelled by ϵ. The vertices labelled by **V** are those with a number and those labelled by **M** are the rest.

Lemma 8. *[6] Let G be a connected undirected graph. Then, $xy \in E_G$ if and only if there exists an alternating path between x and y in $\mathcal{S}_G$. This alternating path is moreover unique.*

The following gives a characterisation of the property "z is in a shortest path in G between x and y" with respect to shortest paths in $\mathcal{S}_G$.

Proposition 9. *Let G be a connected undirected graph and let x, y, z be vertices of G. Then, $\mathcal{SP}_G(x, z, y)$ holds if and only if*

(i) $\mathcal{SP}_{\mathcal{S}_G}(x, z, y)$ holds, or
(ii) there exists a marker h such that $\mathcal{SP}_{\mathcal{S}_G}(x, h, y)$ holds and there exists an alternating path between h and z in $\mathcal{S}_G$ starting with an edge labelled by ϵ.

We now return to distance-hereditary graphs. We will use the following theorem that characterises distance-hereditary graphs (among the several ones).

Theorem 10 ([2]). *A connected undirected graph is a distance-hereditary graph if and only if each block of its split decomposition is either a clique or a star block.*

As a corollary we get the following (the proof is an easy induction on the size of the split decomposition).

Corollary 11. *Let G be a connected distance-hereditary graph. Then, a sequence $P := (x_0, \ldots, x_k)$ is a shortest path in $\mathcal{S}_G$ if and only if P is a chordless path in $\mathcal{S}_G$. Moreover, a shortest path between two vertices in $\mathcal{S}_G$ is unique.*

We can therefore prove the following.

Proposition 12. *There exists an $MS_{\mathcal{R}_A}$ formula $Bet_{\mathcal{SP}}(x, z, y)$ that expresses that z is in a shortest path between x and y in connected distance-hereditary graphs.*

By Corollary 11 the graph obtained from $\mathcal{S}_G$ by forgetting the labels of the vertices and of the edges is a distance-hereditary graph, and then has clique-width at most 3 (see [19]). We can in fact prove that the A-coloured graph $\mathcal{S}_G$ has clique-width at most 3 as stated below (its proof is an easy induction).

Proposition 13. *For every connected distance-hereditary graph G, the A-coloured graph $\mathcal{S}_G$ has clique-width at most 3. Moreover, a clique-width expression of $\mathcal{S}_G$ can be computed in time $O(\|\mathcal{S}_G\|)$.*

Theorem 14. *For every distance-hereditary graph G one can compute in time $O(\|G\|)$ a minimum $\mathcal{SP}_G$-hull set and a minimum $\mathcal{SP}_G$-geodetic set of G.*

Proof. From Fact 1 it is enough to prove the theorem for connected distance-hereditary graphs. So assume that G is connected. By Theorem 10 one can compute in time $O(\|G\|)$ the split decomposition $\mathcal{D}_G$ of G. By [18, Lemma 2.2] we have $\|\mathcal{S}_G\| = O(\|G\|)$, and therefore one can compute $\mathcal{S}_G$ in time $O(\|G\|)$. By Theorem 4 and Propositions 5, 12 and 13, one can compute a minimum $\mathcal{SP}_G$-hull set and a minimum $\mathcal{SP}_G$-geodetic set of G in time $O(\|\mathcal{S}_G\|) = O(\|G\|)$. □

4 Chordal Graphs

The algorithm for chordal graphs is based on the notion of *functional dependencies*, borrowed from database community. Before introducing them, let us recall some properties of chordal graphs.

Lemma 15. *[11] Every chordal graph has at least two simplicial vertices.*

A *perfect elimination ordering* of a graph G is an ordering $\sigma := (x_1, \ldots, x_n)$ of V_G such that for every $1 \leq i \leq n$, the vertex x_i is simplicial in $G[\{x_i, \ldots, x_n\}]$. The following is a well-known result.

Theorem 16. *[17,21] A graph G is chordal if and only if it has a perfect elimination ordering. Moreover, a perfect elimination ordering can be computed in time $O(\|G\|)$.*

We now define *functional dependencies*. Let V be a finite set. A *functional dependency* on V is a pair (X, y), often written $X \to y$, where $X \subseteq V$ is called the *premise*, and $y \in V$ is called the *conclusion*. If $X \to z$ is a functional dependency with $X = \{x, y\}$ (or $X = \{x\}$), we will write $xy \to z$ (or $x \to z$) instead of $\{x, y\} \to z$ (or $\{x\} \to z$). A *model* for $X \to y$ is a subset F of V such that $y \in F$ whenever $X \subseteq F$.

A set Σ of functional dependencies on V is called an *implicational system on* V. If an implicational system is composed only of functional dependencies with premises of size 1 we call it a *unit-premise implicational system*. A set $X \subseteq V$ is

said Σ-*closed* if it is a model for each functional dependency in Σ. The Σ-*closure* of a set X, noted $\Sigma(X)$, is the smallest Σ-closed set that contains X. A *superkey* for Σ is a subset K of V such that $\Sigma(K) = V$ and a *key* is an inclusionwise minimal superkey.

If Σ is an implicational system on a finite set V, we let $\mathscr{G}(\Sigma)$, called the *dependence graph* of Σ, be the directed graph (V', E') with

$$V' := V \cup \{P_X \mid X \text{ is a premise in } \Sigma\},$$
$$E' := \bigcup_{X \to y \in \Sigma} \Big(\big(\{(x, P_X) \mid x \in X\} \big) \cup \{(P_X, y)\} \Big).$$

One observes that $\mathscr{G}(\Sigma)$ has size $O\left(|V| + \sum_{X \to y \in \Sigma} \big(|X| + 1\big) \right)$ and can be computed in time $O\left(|V| + \sum_{X \to y \in \Sigma} \big(|X| + 1\big) \right)$.

Proposition 17. *Let Σ be a unit-premise implicational system on a finite set V. Then a minimum key of Σ can be computed in time $O(|V| + |\Sigma|)$.*

We now borrow some ideas from [22]. Let Σ be an implicational system on a finite set V. A strongly connected component S of $\mathscr{G}(\Sigma)$ is called a *source component* if for all $x, y \in V_{\mathscr{G}(\Sigma)}$, if $(x, y) \in E_{\mathscr{G}(\Sigma)}$ and $y \in V_S$, then $x \in V_S$. That means that all edges between a vertex in S and a vertex in $V_{\mathscr{G}(\Sigma)} \setminus S$ is always from S.

Let R be a subset of V. We let Σ^R, called the *restriction of Σ to R*, be the implicational system on R defined as $\{X \to y \in \Sigma \mid X \cup \{y\} \subseteq R\}$. A *contraction* of Σ to R is the implicational system $\overline{\Sigma^R}$ on $V \setminus \Sigma(R)$ defined as

$$\big\{X \to y \mid X \cup S \to y \in \Sigma, \text{ and } S \subseteq \Sigma(R) \text{ and } X \cup \{y\} \subseteq V \setminus \Sigma(R)\big\} \bigcup \big\{X \to y \in \Sigma \mid X \cup \{y\} \subseteq V \setminus \Sigma(R)\big\}.$$

Proposition 18. *[22] Let Σ be an implicational system on a finite set V. Let S be a source component of $\mathscr{G}(\Sigma)$. Then each minimum key of Σ is a union of a minimum key of Σ^{V_S} and of a minimum key of $\overline{\Sigma^{V_S}}$. Moreover, all such unions are minimum keys of Σ.*

If Σ is an implicational system on V, then $x \in V$ is called an *extreme point* if x is not the conclusion of any functional dependency in Σ. It is straightforward to verify that any extreme point is a source component in $\mathscr{G}(\Sigma)$ and then is in any minimum key of Σ. (Notice that a similar notion of extreme points has been already used in the literature, see for instance [4].)

Example 1. Let $V := \{1, 2, 3, 4, 5, 6\}$ and let $\Sigma := \{\{1, 2\} \to 3, \{2, 4\} \to 1, \{1, 3, 5\} \to 6, \{1, 4\} \to 3, 1 \to \{2, 4\}\}$. 5 is an extreme point, and indeed it must be in any key of Σ, but 6 cannot be in any key. Examples of keys are $\{1, 5\}, \{2, 4, 5\}$, and examples of Σ-closed sets are $\{2\}, \{1, 2, 3, 4\}, \{3, 6\}$.

We can now explain the algorithm for connected chordal graphs. We will associate with every graph G the implicational system on V_G

$$\Sigma_G := \bigcup_{x,y \in V} \{xy \to z \mid z \in \mathcal{SP}^o_G(\{x,y\}) \setminus \{x,y\}\}.$$

One notices that a vertex x of a graph G is simplicial if and only if x is an extreme point in Σ_G.

Fact 19. *Let G be an undirected graph. Then, $\mathcal{SP}^+_G(X) = \Sigma_G(X)$ for every $X \subseteq V_G$. Then, K is a minimum $\mathcal{SP}_G$-hull set of G if and only if K is a minimum key of Σ_G.*

According to Proposition 18, our strategy is recursively.

1. Take a source component S of Σ_G and decompose Σ_G into $\Sigma_G^{V_S}$ and $\overline{\Sigma_G^{V_S}}$.
2. Compute a minimum key K_1 of $\Sigma_G^{V_S}$.
3. Compute a minimum key K_2 of $\overline{\Sigma_G^{V_S}}$.
4. Return $K_1 \cup K2$.

The first difficulty is here when the source component is not a single element. The second difficulty is to iterate this process in $\Sigma := \overline{\Sigma_G^{V_S}}$. We overcome these difficulties by first constructing an elimination ordering $\sigma := (x_1, \ldots, x_n)$ of G and the algorithm treats the vertices of G, in this order, and decides at each step whether the current vertex can be in a minimum key (that is updated at each step). At the beginning Σ is set up to Σ_G. At each step i, if x_i is in the closure of the already computed key, then we go to the next step (the current vertex cannot be in a minimum key). Otherwise, if x_i is an extreme point in Σ, then it is a source component and then we can include it in a minimum key of Σ and use Proposition 18 to update Σ for the next iteration. If x_i is not an extreme point, then since it is a simplicial vertex in $G[\{x_i, \ldots, x_n\}]$, it cannot be a conclusion in any functional dependencies with premises of size 2 in Σ. So, that means it appears as a conclusion in functional dependencies with premises of size 1 in Σ (those latter are created during the process by Proposition 18). Since we cannot decide whether to include x_i in the computed minimum key, we update Σ, according to Lemma 20 below, in order to remove x_i in the premises of functional dependencies in Σ. At the end of the n iterations, Σ is reduced to a unit-premise implicational system and by Proposition 17 we can compute minimum keys of such implicational systems.

Lemma 20. *Let $\Sigma := \Sigma_1 \cup \Sigma_2$ be an implicational system on a finite set V with $\Sigma_1 := \{z \to t \in \Sigma\}$ and $\Sigma_2 := \{zt \to y \in \Sigma\}$. Let x in V be an extreme point in Σ_2 and not in Σ_1 and let $\Sigma' := \big(\Sigma \setminus \{xz \to y \in \Sigma\}\big) \cup \big(\{tz \to y \mid xz \to y, t \to x \in \Sigma\}\big)$. Then, any minimum key K' of Σ' is a minimum key of Σ. Conversely, to any minimum key K of Σ, one can associate a minimum key K' of Σ'.*

We can therefore state the following.

Theorem 21. *For any connected chordal graph G, one can compute a minimum $\mathcal{SP}_G$-hull set of G in time $O\left(\|G\| + \sum_{x,y \in V_G} \left|\mathcal{SP}^o_G(\{x,y\})\right|\right)$, which is less than $O(|V_G|^3)$. Therefore, one can compute a minimum $\mathcal{SP}_G$-hull set of any chordal graph G in time $O(|V_G|^3)$.*

5 Concluding Remarks

In this paper we have given a linear time algorithm and a cubic-time algorithm for computing a minimum $\mathcal{SP}_G$-hull set in distance-hereditary and chordal graphs respectively. The techniques used to get these two algorithms are different from the ones known in the literature, specially those techniques based on functional dependencies borrowed from database community. We hope these techniques will be fruitful to obtain new algorithms for graph classes where the complexity of computing a minimum $\mathcal{SP}_G$-hull set is open. We can cite among them graph classes of bounded tree-width and more generally those of bounded clique-width, weakly chordal graphs, degenerate graphs, ...

We conclude by pointing out that our techniques for distance-hereditary graphs can be applied to compute a minimum $\mathcal{SP}_G$-hull set or $\mathcal{SP}_G$-geodetic set in other graph classes. We have for instance the following.

Proposition 22. *Let k be a fixed integer. Let G be an undirected graph such that $\mathcal{G}(\Sigma_G)$ has clique-width at most k. Then, one can compute in time $O(|V_G|^6)$ a minimum $\mathcal{SP}_G$-hull set and a minimum $\mathcal{SP}_G$-geodetic set of G.*

A question that arises then is which graphs have dependence graphs of bounded clique-width?

The logical tools can be also applied to other betweenness relations. For instance, for an undirected graph G, we let $\mathcal{P}_G$ be the betweenness relation where $\mathcal{P}_G(x,z,y)$ holds if and only if z is in a chordless path between x and y. We can prove the following.

Proposition 23. *Let k be a fixed positive integer. Let G be an undirected graph of clique-width at most k. Then, one can compute in time $O(|V_G|^3)$ a minimum $\mathcal{P}_G$-hull set and a minimum $\mathcal{P}_G$-geodetic set of G.*

References

1. Araújo, J., Campos, V., Giroire, F., Nisse, N., Sampaio L., Soares, R.P.: On the hull number of some graph classes. Technical report (2011)
2. Bouchet, A.: Transforming trees by successive local complementations. J. Graph Theory 12(2), 195–207 (1988)
3. Chartrand, G., Fink, J.F., Zhang, P.: The hull number of an oriented graph. International Journal of Mathematics and Mathematical Sciences (36), 2265–2275 (2003)

4. Chvátal, V.: Antimatroids, betweenness, convexity. In: Cook, W.J., Lovász, L. (eds.) Research Trends in Combinatorial Optimization, pp. 57–64. Springer (2009)
5. Codd, E.F.: Further normalization of the data base relational model. In: Rustin, R. (ed.) Data Base Systems, pp. 33–64. Prentice Hall, Englewood Cliffs (1972)
6. Courcelle, B.: The monadic second-order logic of graphs XVI: Canonical graph decompositions. Logical Methods in Computer Science 2(2) (2006)
7. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic: a Language Theoretic Approach. Encyclopedia of Mathematics and its Applications, vol. 138. Cambridge University Press (2012)
8. Cunningham, W.H., Edmonds, J.: A combinatorial decomposition theory. Canadian Journal of Mathematics 32, 734–765 (1980)
9. Dahlhaus, E.: Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. J. Algorithms 36(2), 205–240 (2000)
10. Diestel, R.: Graph Theory, 3rd edn. Springer (2005)
11. Dirac, G.A.: On rigid circuit graphs. Abhandlungen Aus Dem Mathematischen Seminare der Universität Hamburg 25(1-2), 71–76 (1961)
12. Dourado, M.C., Gimbel, J.G., Kratochvíl, J., Protti, F., Szwarcfiter, J.L.: On the computation of the hull number of a graph. Discrete Mathematics 309(18), 5668–5674 (2009)
13. Dourado, M.C., Protti, F., Rautenbach, D., Szwarcfiter, J.L.: On the hull number of triangle-free graphs. SIAM J. Discrete Math. 23(4), 2163–2172 (2010)
14. Dourado, M.C., Protti, F., Rautenbach, D., Szwarcfiter, J.L.: Some remarks on the geodetic number of a graph. Discrete Mathematics 310(4), 832–837 (2010)
15. Ekim, T., Erey, A., Heggernes, P., van 't Hof, P., Meister, D.: Computing Minimum Geodetic Sets of Proper Interval Graphs. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 279–290. Springer, Heidelberg (2012)
16. Everett, M.G., Seidman, S.B.: The hull number of a graph. Discrete Mathematics 57(3), 217–223 (1985)
17. Fulkerson, D.R., Gross, O.A.: Incidence Matrices and Interval Graphs. Pacific J. Math. 15(3), 835–855 (1965)
18. Gavoille, C., Paul, C.: Distance labeling scheme and split decomposition. Discrete Mathematics 273(1-3), 115–130 (2003)
19. Golumbic, M.C., Rotics, U.: On the clique-width of some perfect graph classes. Int. J. Found. Comput. Sci. 11(3), 423–443 (2000)
20. Pelayo, I.M.: On convexity in graphs. Technical report (2004)
21. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. 5(2), 266–283 (1976)
22. Saiedian, H., Spencer, T.: An efficient algorithm to compute the candidate keys of a relational database schema. Comput. J. 39(2), 124–132 (1996)