Greedy Algorithms for Optimization on Distance-Hereditary Graphs

Olivier Cogis and Eric Thierry
LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France.
Email: (ocogis,thierry)@lirmm.fr

Abstract

Distance-hereditary graphs can be reduced to a single vertex by repeatedly removing a pendant or a twin vertex from the graph. The sequence of removal is called a pruning sequence. When restricted to distance-hereditary graphs, several well-known $\mathcal{NP}$-complete problem turn to be polynomial, some in fact linear.

We focuss on pruning sequences considered as encoding the structure of distance-hereditary graphs. We give algorithms running along a single pass through a pruning sequence of a distance-hereditary graph, linear in the size of the pruning sequence, in order to solve three well-known problems: the maximal weighted stable set problem, a minimal weighted ab-separator problem and the diameter problem.

Keywords: graphs, distance-hereditary graphs, pruning sequence, optimization problems.

1 Introduction

Distance-hereditary graphs have been introduced by E. Howorka [17] as graphs such that the distance between two connected vertices is the same in every induced subgraph in which they remain connected. Many characterizations are known for this class: by means of forbidden subgraphs, properties of cycles, metric charaterizations (see [2] for a survey). Distance-hereditary graphs are known to be perfect graphs, while trees and cographs are well-studied classes of distance-hereditary graphs (see [4] for a survey).

Distance-hereditary graphs have been intensively studied from an algorithmic point of view, and it turns out that many problems can be solved in linear time and/or space complexity, including several well-known $\mathcal{NP}$-complete problems [1,3,5-7,9-14,16,18-23] As a matter of fact, distance-hereditary graphs have a clique-width ≤ 3 [15] with a linear construction of a 3-expression, yielding linear algorithms for a large amount of problems [8] (but to the expense of a large constant factor).

H.J. Bandelt and H.M. Mulder [2] provided a ***pruning sequence*** characterization of distance-hereditary graphs. Namely, whenever a graph G has a pendant vertex (a vertex with a unique neighbour) or a vertex with a twin (another vertex sharing any other neighbour), remove this vertex. Distance-hereditary graphs are those graphs reducible to a single vertex by means of a sequence of such vertex removals repeatedly applied to the graph. The sequence is then called a ***pruning sequence*** of the distance-hereditary graph. Given a distance-hereditary graph, a pruning sequence can be computed in linear time [10, 16]. And computing a pruning sequence not necessarily leading to a single vertex can be computed for any graph in linear time [20].

Using the pruning sequence of a distance-hereditary graph as the unique data encoding its structure, we focuss here on problems that can be solved by a single pass through the pruning sequence, such as computing a ***maximal weighted stable set***, a ***minimal weighted*** ab***-separator*** and a ***diametral path***.

Section 2 provides definitions needed in further sections and outlines a general reasoning scheme used to prove each of the three algorithms presented in Sections 3,4,5, which respectively solve the three problems mentionned above. We conclude in Section 6.

2 Definitions and a general optimization scheme

Let $G = (V, E)$ be an undirected graph (with no loop and no multiple edge). If $x \in V$, $N(x) = \{y \in V \mid (x, y) \in E\}$ is the neighborhood of x in G. An $x \in V$ such that $|N(x)| = 1$ is called a ***pendant vertex***. Vertices $x, y \in V$ such that $N(x)\backslash y = N(y)\backslash x$ are called ***twin vertices***: ***false twins*** if adjacent, ***true twins*** otherwise. A ***pruning sequence*** of a graph is a sequence of vertex removals such that at each step the removed vertex is a pendant vertex or a vertex that have a true or a false twin. As proved in [2], a graph is distance-hereditary iff it has a pruning sequence that leads to a single vertex. We use the following notations for a pruning sequence. We read the sequence from the left to the right and xTy means that x and y are true twins and that x is to be removed, xFy means that x and y are false twins and that x is to be removed, xPy means that x is a pendant vertex with y being its only neighbor and that x is to be removed. For instance, (aPb, dFb, eTc, ePb) is a pruning sequence of the distance-hereditary graph G shown on the figure 1 below. Note that when read from right to left, a pruning sequence can be used to reconstruct the graph. The number of vertices of the graph is denoted by n and the number of edges is denoted by m.

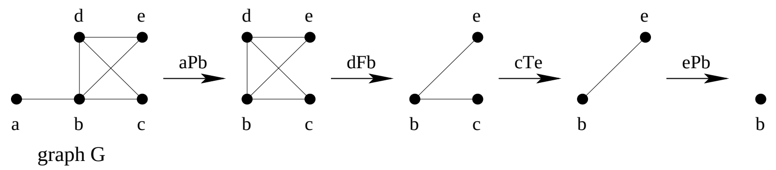

Figure 1: Example of a pruning sequence for the graph G

The three algorithms presented in the next three sections share some common features.
1- The input is a pruning sequence for some distance-hereditary together with a weight function defined on vertices.
2- The ouput is a solution to some optimization problem, expressed as the value of some maximum (resp. minimum) defined on the labelled graph together with some realizer of this value.
3- The algorithm is mainly a single pass through the given pruning sequence. According to the problem to be solved, the input graph G is relabelled (for the sake of simplicity, we still write G for the relabelled graph). Some set Λ_G is defined from G, and some weight function W_G is defined on Λ_G so that the optimal value v originally to be computed satisfies $v = \max\{W_G(\lambda)|\lambda \in \Lambda_G\}$ (resp. min). The key point to the proof of algorithm can be stated as follows. Assuming that one (twin or pendant) vertex deletion from the graph G yields the graph G', we define some updating of the labelling of G in order to produce a labelling of G', together with some function Φ from Λ_G into $\Lambda_{G'}$

such that $\mathcal{R}(\Phi)$ holds, where $\mathcal{R}(\Phi)$ stands for: Φ is surjective and for all S' in $\Lambda_{G'}$, $W_{G'}(\lambda') = \max\{W_G(\lambda) | \lambda \in \Phi^{-1}(\lambda')\}$. This ensures that $\max\{W_G(\lambda)|\lambda \in \Lambda_G\}$ remains constant along any sequence of vertex deletions induced by any pruning sequence of the distance-hereditary graph G.
4- The running time and space are in $\mathcal{O}(n)$.

3 The maximal weighted stable set problem

The ***maximal weighted stable set*** problem can be stated as: given some graph G where each vertex holds a positive weight, find a stable set which maximizes the sum of the weights of its vertices.

Algorithm 1: Maximal weighted stable set for distance-hereditary graphs

Input: A pruning sequence of the distance-hereditary graph G with a weight $w(x) > 0$ for each vertex x.
Output: A stable set S of G of maximum weight $\sum_{x \in S} w(x)$.
begin
  To each vertex x is associated a couple of weights $(p(x), q(x))$ and a couple $(P(x), Q(x))$ of subsets of V:
  foreach x *in* V **do**
    $p(x) \leftarrow w(x)$, $q(x) \leftarrow 0$;
    $P(x) \leftarrow \{x\}$, $Q(x) \leftarrow \emptyset$;
  while *the pruning sequence is not empty* **do**
    Let $x * y$, where $* \in \{F, T, P\}$, be the head of the pruning sequence. Update as follows:
    case xTy
      Replace the couple of weights of y by:
        $(\max(p(x)+p(y), p(x)+q(y), p(y)+q(x)), q(x)+q(y))$
      Among $p(x)+p(y)$,$p(x)+q(y)$ and $p(y)+q(x)$, choose one sum that is maximum, and replace the couple of sets associated to y as follows:
      case *it is* $p(x)+p(y)$*:* replace by $(P(x) \bigcup P(y), Q(x) \bigcup Q(y))$;
      case *it is* $p(x)+q(y)$*:* replace by $(P(x) \bigcup Q(y), Q(x) \bigcup Q(y))$;
      case *it is* $p(y)+q(x)$*:* replace by $(P(y) \bigcup Q(x), Q(x) \bigcup Q(y))$;
    case xFy
      Replace the couple of weights of y by:
        $(\max(p(x)+q(y), p(y)+q(x)), q(x)+q(y))$
      Among $p(x)+q(y)$ and $p(y)+q(x)$, choose one sum that is maximum, and replace the couple of sets associated to y as follows:
      case *it is* $p(x)+q(y)$*:* replace by $(P(x) \bigcup Q(y), Q(x) \bigcup Q(y))$;
      case *it is* $p(y)+q(x)$*:* replace by $(Q(x) \bigcup P(y), Q(x) \bigcup Q(y))$;
    case xPy
      Replace the couple of weights of y by:
        $(p(y)+q(x), \max(p(x)+q(y), q(x)+q(y)))$
      Among $p(x)+q(y)$ and $q(x)+q(y)$, choose one sum that is maximum, and replace the couple of sets associated to x as follows:
      case *it is* $p(x)+q(y)$*:* replace by $(P(y) \bigcup Q(x), P(x) \bigcup Q(y))$;
      case *it is* $q(x)+q(y)$*:* replace by $(P(y) \bigcup Q(x), Q(x) \bigcup Q(y))$;
    Remove the head of the pruning sequence and remove x from the graph.
  There is only one vertex x left, with weights $(p(x), q(x))$ and sets $(P(x), Q(x))$.
  if $p(x) \geq q(x)$ **then** return the set $P(x)$ and the weight $p(x)$
  else return the set $Q(x)$ and the weight $q(x)$.
end

Theorem 1 *Algorithm 1 returns a stable set of maximum weight of the input graph. It can be implemented to run in $\mathcal{O}(n)$ time and space.*

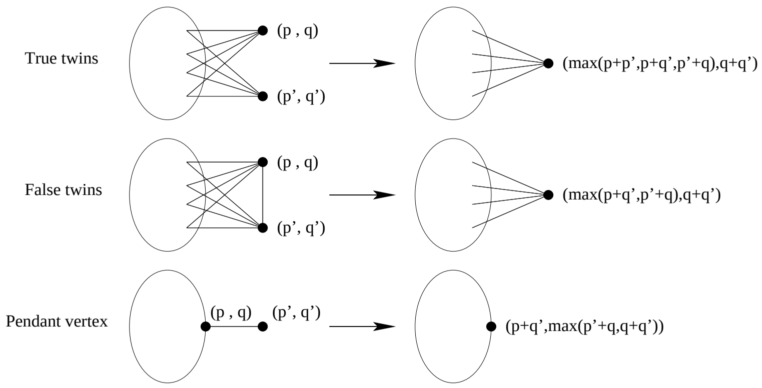

Figure 2: Updating the weights for the three different types of pruning in Algorithm 1.

Sketch of the proof: We follow the general scheme presented in section 2. We denote by Λ_G the collection of all the stable sets of the graph G. The weight function W_G is defined for all S in Λ_G by $W_G(S) = \sum_{x \in S} p(x) + \sum_{x \notin S} q(x)$. The function Φ from Λ_G to $\Lambda_{G'}$ corresponding to the updates of Algorithm 1 is defined by:

- If it is xTy or xFy, for all $S \in \Lambda_G$, $\Phi(S) = \begin{cases} S & \text{if } x \notin S \\ (S \backslash \{x\}) \bigcup \{y\} & \text{if } x \in S \end{cases}$
- If it is xPy, for all $S \in \Lambda_G$, $\Phi(S) = S \backslash \{x\}$.

Let G_0 be the initial distance-hereditary graph. Finding $\max\{W_{G_0}(S) | S \in \Lambda_{G_0}\}$ is exactly finding the maximum weighted stable set of G_0 because of the initialization step of Algorithm 1. Because the property $\mathcal{R}(\Phi)$ holds (a case by case proof omitted here), $\max\{W_G(S) | S \in \Lambda_G\} = \max\{W_{G_0}(S) | S \in \Lambda_{G_0}\}$ is an invariant of Algorithm 1. In order to prove that the subsets labelling the vertices achieve the optimum, here are some usefull invariants for Algorithm 1:

(1) For all vertices x and y in V, $x \neq y$, $(P(x) \cup Q(x)) \cap (P(y) \cup Q(y)) = \emptyset$.

(2) For any vertex x in V, $\sum_{v \in P(x)} w(v) = p(x)$ and $\sum_{v \in Q(x)} w(v) = q(x)$

(3) For all vertices x and y in V, $x \neq y$, their associated sets verify:

$$\begin{cases} Q(x) \bigcup Q(y) \in \Lambda_{G_0}, & (i) \\ P(x) \bigcup Q(y) \in \Lambda_{G_0}, & (ii) \\ P(x) \bigcup P(y) \in \Lambda_{G_0} \text{ if } xy \notin E. & (iii) \end{cases}$$

(4) For any vertex x in V, $P(x)$ and $Q(x)$ are both in Λ_{G_0}.

As the algorithm uses nothing but a pruning sequence of the initial graph, it can be implemented to run in $\mathcal{O}(n)$ time and space. $\square$

4 The minimum weighted ab-separator problem

The ***minimum weighted*** ab***-separator problem*** can be stated as: given a graph G with at least two vertices and two fixed vertices a and b of G, find an a, b-separator (i.e.

a subset of vertices not including $\{a, b\}$ such that any path from a to b has a vertex in this set) which minimizes the sum of the weights of its vertices.

Algorithm 2: Minimum weighted a, b-separator for distance-hereditary graphs

Input: A pruning sequence of the distance-hereditary graph G with a weight $w(x) > 0$ for each vertex x, and two vertices a and b from this graph.
Output: An ab-separator S in G of minimum weight $\sum_{x \in S} w(x)$.

```
begin
    Mark a and b are marked and receive a infinite weight: p(a) ← ∞; p(b) ← ∞;
    Each non-marked vertex x receives a weight p(x) and a subset P(x) of V:
    p(a) ← ∞; p(b) ← ∞;
    P(a) ← ∅; P(b) ← ∅;
    foreach x in V\{a,b} do
        p(x) ← w(x);
        P(x) ← {x};
    Let x ∗ y, where ∗ ∈ {F,T,P}, be the head of the pruning sequence;
    while x and y are not both marked do
        if x is marked then
            Mark y;
            if xTy or xFy or (xPy and p(x) < p(y)) then
                p(y) ← p(x); P(y) ← P(x);
        else
            if y is not marked and (xTy or xFy) then
                p(y) ← p(x) + p(y);
                P(y) ← P(x) ⋃ P(y);
        Remove the head of the pruning sequence and remove x from the graph;
        Let x ∗ y, where ∗ ∈ {F,T,P}, be the new head of the pruning sequence;
    Both x and y are marked, there are two cases:
    case xFy or xPy
        From the two couples (P(x), p(x)) and (P(y), p(y)), return the set associated to the
        minimum weight;
    case xTy
        Using the remaining pruning sequence, compute ⋃_{v∈N(x)} P(v) and ∑_{v∈N(x)} p(v)) into
        variables sum and set as follows: Remove the head of the pruning sequence and remove
        x from the graph;
        Let v ∗ w, where ∗ ∈ {F,T,P}, be the new head of the pruning sequence;
        sum ← 0; set ← ∅;
        while the pruning sequence is not empty and not(v is marked and vPw) do
            case v is marked
                Mark w;
                if vFw then sum ← sum + p(w); set ← set ∪ P(w);
            case w is marked and (vPw or vFw): sum ← sum + p(w); set ← set ∪ P(w);
            case vFw and vTw: sum ← sum + p(w); set ← set ∪ P(w);
            Remove the head of the pruning sequence and remove v from the graph;
            Let v ∗ w, where ∗ ∈ {F,T,P}, be the new head of the pruning sequence;
        if v is marked and vPw then sum ← sum + p(w); set ← set ∪ P(w);
        From the three couples (P(x), p(x)), (P(y), p(y)) and (⋃_{v∈N(x)} P(v), ∑_{v∈N(x)} p(v)),
        return the set associated to the minimum weight;
end
```

Theorem 2 *Algorithm 2 returns an a, b-separator of minimum weight of the input graph. It can be implemented to run in $\mathcal{O}(n)$ time and space.*

Sketch of the proof: We follow the general scheme presented in section 2. During the running of Algorithm 2 there are exactly two marked vertices, we denote them by z_G and

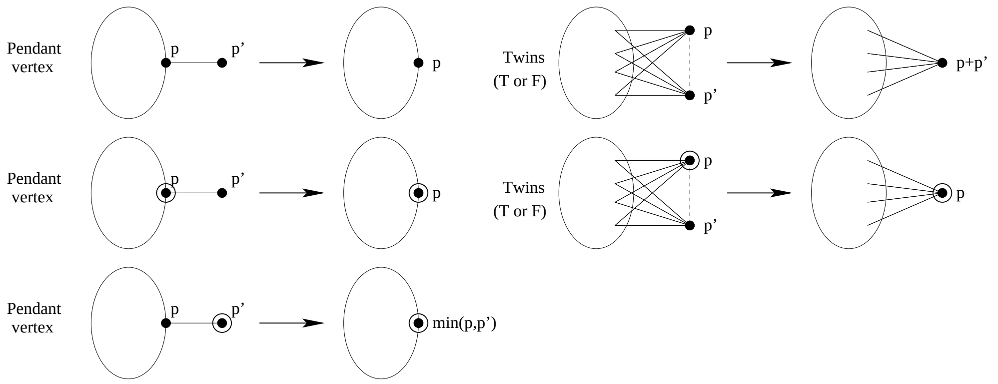

Figure 3: Updating the weights for the different types of pruning and marks in Algorithm 2. Two marked vertices are maintained (and exactly two), initialized to a and b. The marked vertices are circled and dotted lines join twin vertices (whether they are true or false twins is not relevant). We only show the non-terminal configurations (at most one of the two vertices involved in the deletion is marked).

z'_G. We denote by $\Lambda_G^{z_G z'_G}$ the collection of all the $z_G z'_G$-separators of G and by Λ_G the set $\{S \subseteq V | S \in \Lambda_G^{z_G, z'_G}$ or $S = \{z_G\}$ or $S = \{z'_G\}\}$. The weight function W_G is defined for all S in Λ_G by $W_G(S) = \sum_{x \in S} p(x)$. The function Φ from Λ_G to $\Lambda_{G'}$ corresponding to the updates of Algorithm 2 is defined by:

- If we have ((xTy or xFy) with y marked) or (xPy with x marked), then for all $S \in \Lambda_G$, $\Phi(S) = S \backslash \{x\}$.
- If we have xPy with x marked, then for all $S \in \Lambda_G$, $\Phi(S) = \left\{ \begin{array}{ll} \{y\} & \text{if } S = \{x\} \\ S & \text{otherwise} \end{array} \right.$
- If we have (xTy or xFy) with x marked, then for all $S \in \Lambda_G$,

$$\Phi(S) = \left\{ \begin{array}{ll} \{y\} & \text{if } S = \{x\} \\ S \backslash \{x\} & \text{otherwise} \end{array} \right.$$

- If we have (xTy or xFy) with x and y both unmarked, then for all $S \in \Lambda_G$,

$$\Phi(S) = \left\{ \begin{array}{ll} S \backslash \{y\} & \text{if } \{x, y\} \subseteq S \\ S \backslash \{x, y\} & \text{otherwise} \end{array} \right.$$

Let G_0 be the initial distance-hereditary graph. Finding $\min\{W_{G_0}(S) \mid S \in \Lambda_{G_0}\}$ is exactly finding the minimum weighted a, b-separator set of G_0 because of the initialization step of Algorithm 2. Because the property $\mathcal{R}(\Phi)$ holds (a case by case proof omitted here), $\min\{W_G(S) | S \in \Lambda_G\} = \min\{W_{G_0}(S) | S \in \Lambda_{G_0}\}$ is an invariant of Algorithm 2. In order to prove that the subsets labelling the vertices achieve the optimum, here are some usefull invariants for Algorithm 2:

(1) For all vertices x and y in V, $x \neq y$, $P(x) \bigcap P(y) = \emptyset$.

(2) For any vertex x in V such that $p(x) < \infty$, we have: $\sum_{v \in P(x)} w(v) = p(x)$.

(3) Let z_G and z'_G be the marked vertices of G. The current graph verifies:

$$\begin{cases} \forall S \in \Lambda_G^{z_G z'_G},\ \bigcup_{x\in S} P(x) \in \Lambda_{G_0}^{ab}, & (i) \\ \forall x \in \{z_G, z'_G\},\ p(x) < \infty \text{ implies } P(x) \in \Lambda_{G_0}^{ab}. & (ii) \end{cases}$$

As Algorithm 1, Algorithm 2 can be implemented to run in $\mathcal{O}(n)$ time and space. □

5 The computation of the diameter

The diametral path problem can be stated as: given a graph G, find a path which maximizes over pairs of vertices the length of the shortest paths connecting two vertices. Such a path is called a diametral path and its length the diameter of the graph. When distance-hereditary graphs are concerned the problem can be restated as: find a chordless path of maximum length.

Algorithm 3 solves the problem as restated and makes use of the following definitions and notations. The lists represent paths of the initial graph. For a path S of vertices $x_1, x_2, \ldots, x_k$, we denote by $head(S)$ the first vertex x_1, by $tail(S)$ the last vertex x_k and by $length(S)$ the length of the path which is $k-1$. We denote by $S+S'$ the concatenation of two paths S and S' (when $tail(S)$ and $head(S')$ are adjacent vertices), and by S^{-1} the inverted path from path S.

Algorithm 3: Computation of the diameter for distance-hereditary graphs

Input: A pruning sequence of the distance-hereditary graph G.
Output: The diameter of G and a diametral path D.
begin
  To each vertex x is associated a weight $p(x)$ and a list $P(x)$;
  foreach x *in* V **do** $p(x) \leftarrow 0$; $P(x) \leftarrow \emptyset$;
  $diam \leftarrow 0$; $D \leftarrow \emptyset$;
  while *the pruning sequence is not empty* **do**
    Let $x * y$, where $* \in \{F, T, P\}$, be the head of the pruning sequence;
    case xTy
      if $p(x) + p(y) + 2 > diam$ **then**
        $diam \leftarrow p(x) + p(y) + 2$;
1   Choose one vertex v in $N(y)$ and $D \leftarrow P(x)^{-1} + v + P(y)$;
      if $p(x) > p(y)$ **then** $P(y) \leftarrow P(x)$;
      $p(y) \leftarrow \max(p(x), p(y))$;
    case xFy
      if $p(x) + p(y) + 1 > diam$ **then**
        $diam \leftarrow p(x) + p(y) + 1$;
        $D \leftarrow P(x)^{-1} + P(y)$;
      if $p(x) > p(y)$ **then** $P(y) \leftarrow P(x)$;
      $p(y) \leftarrow \max(p(x), p(y))$;
    case xPy
      if $p(x) + p(y) + 1 > diam$ **then**
        $diam \leftarrow p(x) + p(y) + 1$;
        $D \leftarrow P(x)^{-1} + P(y)$;
      if $p(x) + 1 > p(y)$ **then** $P(y) \leftarrow P(x)$;
      $p(y) \leftarrow \max(p(x) + 1, p(y))$;
    Remove the head of the pruning sequence and remove x from the graph.;
  Return the value $diam$ as the diameter and D as the chordless path of length $diam$.
end

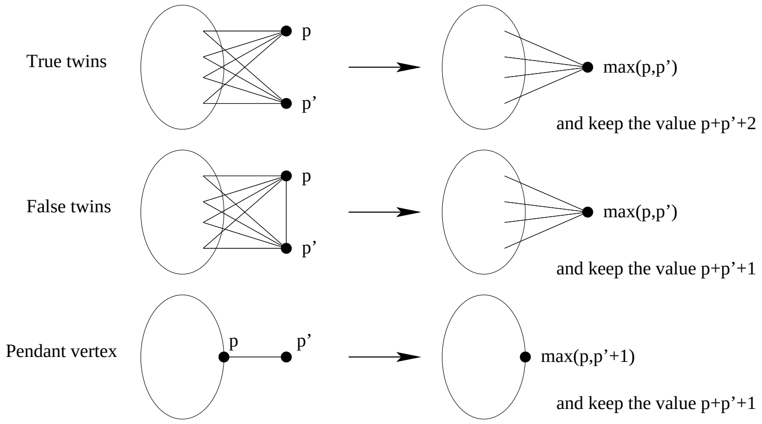

Figure 4: Updating the weights for the different types of pruning in Algorithm 3. The value which is kept is a candidate for the value of the diameter (it is compared to the variable *diam* in algorithm 3).

Theorem 3 *Algorithm 3 returns the diameter of the input graph and a diametral path. It can be implemented to run in $\mathcal{O}(n)$ time and space.*

Sketch of the proof: We follow the general scheme presented in section 2. We denote by Λ^*_G the collection of all the chordless paths of G and by Λ_G the collection $\Lambda^*_G \bigcup \{D\}$. The weight function W_G is defined for all S in Λ_G by $W_G(S) = p(head(S)) + length(S) + p(tail(S))$ if $S \neq D$, and $W_G(D) = diam$. Assuming that a vertex deletion from the graph G (with the list D and weight $diam$) yields the graph G' (with the list D' and weight $diam'$), the function Φ from Λ_G to $\Lambda_{G'}$ corresponding to the updates of Algorithm 3 is defined for all $S \in \Lambda_G$ by:

$$\Phi(S) = \begin{cases} D' & \text{if } S = D \text{ or } x, y \in S, \\ (S\backslash\{x\})\bigcup\{y\} & \text{if } x \in S \text{ and } x \neq head(S), x \neq tail(S), \\ S\backslash\{x\} & \text{if } x = head(S) \text{ or } x = tail(S), \\ S & \text{otherwise.} \end{cases}$$

Let G_0 be the initial distance-hereditary graph. Finding $\max\{W_{G_0}(S) \mid S \in \Lambda_{G_0}\}$ is exactly finding the maximum length of a chordless path of G_0 because of the initialization step of Algorithm 3. Because the property $\mathcal{R}(\Phi)$ holds (a case by case proof omitted here), $\max\{W_G(S)|S \in \Lambda_G\} = \max\{W_{G_0}(S)|S \in \Lambda_{G_0}\}$ is an invariant of Algorithm 2. In order to prove that the lists labelling the vertices and the list D achieve the optimum, here are some usefull invariants for Algorithm 2:

(1) For all vertices x and y in V, $x \neq y$, $P(x) \bigcap P(y) = \emptyset$.

(2) For any vertex x in V, we have $P(x) \in \Lambda^*_{G_0}$ and $length(P(x)) = p(x)$.

(3) For all vertices x and y in V, for all vertices s in $P(x)$ and t in $P(y)$,

$$st \in E \text{ in } G_0 \implies s = head(P(x)),\ t = head(P(y)) \text{ and } xy \in E.$$

(4) $D \in \Lambda^*_{G_0}$ and $length(D) = diam$.

As we only have a pruning sequence, we cannot directly choose v at step 1 in $\mathcal{O}(1)$. However the search for v can continue during the rest of the pruning process, not modifying the overall complexity which remains $\mathcal{O}(n)$ time and space. $\square$

6 Conclusion

We have presented three algorithms showing that pruning sequences encompass enough structure of distance-hereditary graphs, so that well-known optimization problems (some being $\mathcal{NP}$-complete in the general case) can be solved by a single pass through such sequences. Therefore these algorithms have a complexity in $\mathcal{O}(n)$ time and space. This leads us to wonder which other problems should apply to being greedily solved using pruning sequences. The answer might not be obvious, for instance distance-hereditary graphs are known to be *perfectly orderable* graphs for which there exist *perfect orders* inducing a greedy algorithm which computes an optimal coloring of the graph in $\mathcal{O}(n)$. It turns out that perfect orders are not pruning sequences and seem to be more efficient when coloring is at stake.

References

[1] H.-J. Bandelt, A. D'Atri, M. Moscarini, H.-M. Mulder, and A. Schultze. Operations on distance-hereditary graphs. Technical report, Istituto di Analisi dei Sistemi ed Informatica, 1988. R. 226, 7/1988.

[2] H.-J. Bandelt and H. M. Mulder. Distance-hereditary graphs. *J. Combin. Theory*, Ser. B, 41:182–208, 1986.

[3] A. Brandstaädt and F. F. Dragan. A linear-time algorithm for connected r-domination and steiner tree on distance-hereditary graphs. *Networks*, 31:177–182, 1998.

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

[5] H.J. Broersma, E. Dahlhaus, and T. Kloks. A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics*, 99(1-3):367–400, 2000.

[6] M. S. Chang, S. Y. Hsieh, and G. H. Chen. Dynamic programming on distance-hereditary graphs. In *Proceedings of ISAAC'97*, pages 344–353, 1997.

[7] D. G. Corneil, F. F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. In *WG'98*, pages 192–202, 1998.

[8] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. *Theory of Computing Systems*, 33:125–150, 2000.

[9] E. Dahlhaus. Efficient parallel recognition algorithms of cographs and distance hereditary graphs. *Discrete Applied Mathematics*, 57(1):29–44, 1995.

[10] G. Damiand, M. Habib, and C. Paul. A simple paradigm for graphs recognition : application to cographs and distance hereditary graphs. To appear in Theoretical Computer Science.

[11] A. D'Atri and M. Moscarini. Distance-hereditary graphs, steiner trees, and connected domination. *SIAM Journal on Computing*, 17(3):521–538, June 1988.

[12] F. F. Dragan. Dominating cliques in distance-hereditary graphs. In *"Algorithm Theory - SWAT'94 " 4th Scandinavian Workshop on Algorithm Theory*, LNCS 824, pages 370–381, 1994.

[13] F. F. Dragan. A central vertex of a distance-hereditary graph in linear time. In *Abstracts of the 8th Meeting of the EURO Working Group on Locational Analysis*, page page 11, September 1995.

[14] F. F. Dragan and F. Nicolai. Lexbfs-orderings of distance-hereditary graphs with application to the diametral pair problem. *Discrete Appl. Math.*, 98:191–207, 2000. Technical Report SM-DU-303 , University of Duisburg 1995.

[15] M.C. Golumbic and U. Rotics. On the clique-width of perfect graph classes. In *Proceedings of WG'99*, LNCS 1665, pages 135–147, 1999.

[16] P. Hammer and F. Maffray. Completely separable graphs. *Discrete Applied Mathematics*, 27:85–99, 1990.

[17] E. Howorka. A characterization of distance-hereditary graphs. *Quart. J. Math. Oxford*, Ser. 2, 28:417–420, 1977.

[18] S.-Y. Hsieh, C.-W. Ho, T.-S. Hsu, M.-T. Ko, and G.-H. Chen. Characterization of efficiently solvable problems on distance-hereditary graphs. In *ISAAC'98*, pages 257–266, 1998.

[19] S.-Y. Hsieh, C.-W. Ho, T.-S. Hsu, M.-T. Ko, and G.-H. Chen. Efficient parallel algorithms on distance-hereditary graphs. *Parallel Processing Letters*, 9(1):43–52, 1999.

[20] J.-M. Lanlignel, O. Raynaud, and E. Thierry. Pruning graphs with digital search trees. application to distance hereditary graphs. In *Proceedings of STACS'2000*, LNCS 1770, pages 529–541, 2000.

[21] H. Müller and F. Nicolai. Polynomial time algorithms for hamiltonian problems on distance-hereditary graphs. *Information Processing Letters*, 46(5):225–230, 1993.

[22] O. Oellermann. Computing the average distance of a distance hereditary graph in linear time. *Congressus Numerantium*, 103:219–223, 1994.

[23] O. Oellermann and J. Spinrad. On a polynomial algorithm for testing whether a graph is 3-steiner distance hereditary. *Inform. Proc. Letters*, 55:149–154, 1995.