# A Simple Paradigm for Graph Recognition : Application to Cographs and Distance Hereditary Graphs

G. Damiand, Michel Habib, Christophe Paul

# A Simple Paradigm for Graphs Recognition: Application to Cographs and Distance Hereditary Graphs

Guillaume DAMIAND Michel HABIB Christophe PAUL

LIRMM, France

**Abstract**

An easy way for graphs recognition algorithms is to use a two-steps process. First compute a characteristic feature as if the graph belong to that class. Secondly check whether the computed feature really defines the input graphs. Although in some cases the two steps can be merged. But separating them may yield to new and much more easily understood algorithms. In this paper we apply that paradigm to the cographs and distance hereditary graphs recognition problem.

## 1 Introduction

The design of efficient combinatorial algorithms still remains an art, and only few paradigms are available and can be re-used such as: greediness, divide and conquer or dynamic programming techniques .... In the particular case of graph recognition algorithms, i.e. algorithms which compute if a given graph $G$ belongs to a class $\mathcal{C}$ of graphs, let us consider a paradigm called *Compute as if and Check after* which seems to be very powerful to produce efficient and easy to implement algorithms. More precisely we use the following two steps recognition paradigm:

$\alpha$. Compute a characteristic feature as if the graph belongs to the class $\mathcal{C}$.
$\beta$. Check whether the computed feature really defines the input graph.

Some well known graphs recognition algorithms are already based on this principle. Perhaps the most famous examples are given by the recognition algorithms for chordal graphs, namely Lex-BFS [10] and MCS [11]. These algorithms compute an elimination ordering of the vertices that is a perfect elimination ordering if and only if the input graph is chordal. Similarly the very recent interval graphs recognition algorithms [3, 7] are also based on this paradigm. In some simple cases it could be more efficient to merge the two steps, unfortunately it is not always the case as shows the following very interesting example of comparability graphs recognition. Linear time algorithms [9] are now available to compute a characteristic feature, i.e. an orientation of $G$ which is transitive if and only if $G$ is a comparability graph. Up to now a linear test is still not known (i.e. testing in linear time if an oriented graph is transitive). Therefore in this case the two steps do not seem to have the same time-complexity, and the two

steps separation enlights where the bottleneck could be for time complexity. Of course in order to apply such a paradigm the given class of graphs $\mathcal{C}$ must be equipped with a characteristic feature, ideally a decomposition theorem or a representation theorem.

In this paper to enlight the above paradigm, we will focus on two families of graphs, namely: the cographs and the distance hereditary graphs and present two new recognition algorithms. Cographs are characterized by cotrees and distance hereditary graphs by pruning sequences. There already exits a linear time recognition algorithm for cographs [4]. But due to its incrementality it is rather complicated. Nevertheless applying the above paradigm yields to a simpler $O(n + m \log n)$ algorithm. For distance hereditary graphs, Damiand [**?**] finds a counter-example to Hammer and Maffray's algorithm [8]. Using the above paradigm, it can easily be repaired. The interested reader should refer to [5] for parallel recognition algorithms of these classes of graphs.

Section 2 introduces some well known definitions and results. Then in section 3, a new cographs recognition algorithm is presented. The repaired distance hereditary graphs recognition algorithm is designed in section 4.

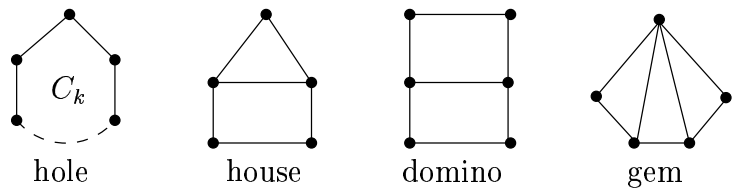
# 2 Preliminaries

All graphs considered in this paper are supposed to be undirected graphs with no loop and no multiple edge. Let $G = (V, E)$ be such a graph, we denote by $N(x)$ the *neighborhood* of a vertex $x$, and by $N[x] = N(x) \cup \{x\}$ the *closed neighborhood* of $x$. Two vertices $x$ and $y$ are *true twins* iff $N(x) = N(y)$, they are *false twins* iff $N[x] = N[y]$. A *pendant vertex* is a vertex of degree 1. The *distance* between two vertices $x$ and $y$, denoted by $d_G(x, y)$, is the length of a shortest path between $x$ and $y$.

A graph $G$ is *distance hereditary* iff for any connected vertices $x$ and $y$ of a subgraph $H$ of $G$, then $d_G(x, y) = d_H(x, y)$. Here is a well known characterization of distance hereditary graphs :

**Theorem 1** *[8] The following conditions are equivalent:*

1. *$G$ is a distance hereditary graph*
2. *Any subgraph of $G$ has either a pendant vertex or a pair of twins*
3. *$G$ has no induced subgraph isomorph to a hole (chordless cycle of length $k \geqslant 5$), a domino, a house or a gem*



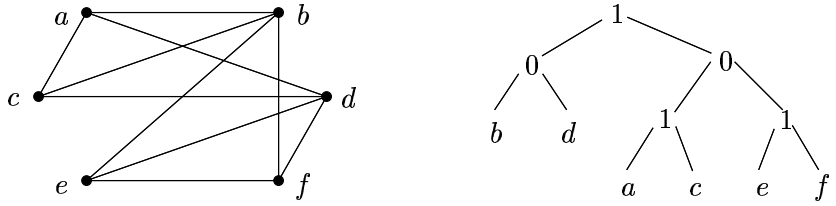

**Definition 1** *A* pruning sequence *is an ordering $\sigma = [(x_1, S, y) \dots (x_i, S, z) \dots x_n]$ such that for any $1 \leq i < n$:*

- *$(x_i, P, x_j)$ if in $G_i$, $N(x_i) = \{x_j\}$ ($x_i$ is a* pendant vertex*)*
- *$(x_i, T, x_j)$ if in $G_i$, $N(x_i) = N(x_j)$ ($x_i$ and $x_j$ are* true twins*)*
- *$(x_i, F, x_j)$ if in $G_i$, $N[x_i] = N[x_j]$ ($x_i$ and $x_j$ are* false twins*)*

The second point of theorem 1 shows that a graph $G = (V, E)$ is a distance hereditary graph iff there exists a pruning sequence of $V$.

**Definition 2** *Cographs is the smallest class of graphs containing the single vertex graph and closed under disjoint union and complement.*

Cographs are distance hereditary graphs : the vertices of a cograph can be pruned without the pendant vertex removal. They have been intensively studied. Moreover cographs are exactly the $P_4$-free graphs. They have a canonical tree-decomposition, called *cotree* (see above figure). The leaves of the cotree are the vertices of the cograph. Let $N$ be an internal node of the cotree and $T_N$ the subtree rooted at $N$. $N$ is labelled by 0 (resp. by 1) if the subgraph induced by the leaves of the subtree rooted at $N$ is *disconnected* (resp. *connected*)




# 3 Cographs recognition algorithm

Let us make two useful remarks. In a cograph, two vertices are adjacent iff their least common ancestor (LCA) in the cotree is labelled by 1. The internal nodes of a path from a leaf to the root are alternatively labelled by 1 and 0.

**Definition 3** *A set of vertices $M$ of a graph $G$ is a* module *iff for any $x$ and $y$ in $M$, $N(x) \setminus M = N(y) \setminus M$. A module $M$ is* prime *is no non-trivial subset of $M$ is a module.*

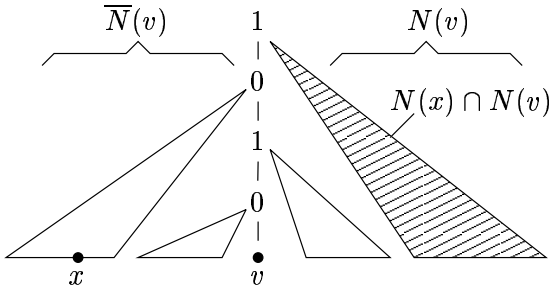
**Definition 4** *[2] A* factorizing permutation $\gamma$ *is a permutation of the vertices where the vertices of any module appear consecutively.*

For any internal node $N$, the set of leaves of $T_N$ is a module. For example $\{a, c, e, f\}$ in a module of the graph in figure **??** : all these vertices are adjacent to $b$ and $d$. The permutation $\gamma = [b, d, a, c, e, f]$ is a factorizing permutation.

**Theorem 2** *[2] Given a factorizing permutation of a cograph, a cotree can be build in linear time.*

The presented algorithm computes a factorizing permutation iff the input graph is a cograph (this is step $\alpha$ of the paradigm). The reader should notice that a cotree can be build directly (without using theorem 2), but this version of the algorithm is simpler. The main idea is to use the first two remarks of this section. Given a vertex $v$, the leaves of the cotree $T$ can be rearranged such that $\overline{N}(v)$ is on the left of $v$ and $N(v)$ on the right (see the next figure). The problem is to gather the leaves of any subtree $T_N$ where $N$ ancestor of $v$ in $T$.

Let $x$ be a non-neighbor of $v$ (*ie* the LCA of $v$ and $x$ is a O-node $N$ on the path in $T$ From $v$ to the root). Let $N'$ be an ancestor of $v$ in $T$ labelled by 1. If $N'$ is a descendant of $N$, then the LCA of $x$ and any leaf $y$ of $N'$ is $N$ (thus $x$




and $y$ are non-adjacent). Therefore separating $N(v) \cap N(x)$ from $N(v) \cap \overline{N}(x)$ distinguishes the leaves of a subtree $T_{N'}$ ($N'$ descendant of $N$) from the other. Now to respect a factorizing permutation $N(x) \cap N(v)$ have to appear on the right of $\overline{N}(x) \cap N(v)$. The same argument holds when $x$ is a non-neighbor of $v$. To compute a factorizing permutation, we put $N(x) \cap N(v)$ on the right of $\overline{N}(x) \cap N(v)$. Since the class of cographs is an hereditary class, this process can be applied recursively.

**Algorithm 1:** Cograph($x$,$\mathcal{X}$)

**Input**: A set of vertices $\mathcal{X}$ of a graph $G$ and a vertex $x \in \mathcal{X}$

**Output**: A factorizing permutation iff $G[\mathcal{X}]$ is a cograph

**begin**

  **if** $|\mathcal{X}| = 1$ **then return** the ordered list $(x)$;

  **if** $|\overline{N}(x)| > |N(x)|$ **then** let $L = (N(x))$ and $L' = (\overline{N}(x))$;

  **else** let $L = (\overline{N}(x))$ and $L' = (N(x))$ be ordered lists;

1 **foreach** *vertex* $y \in \mathcal{Y}$ *a class of* $L$ **do**

    **foreach** *class* $\mathcal{Y}'$ *of* $L'$ **do**

      **if** $\mathcal{Y}' \cap N(y) \neq \mathcal{Y}'$ *and* $\mathcal{Y}' \cap N(y) \neq \emptyset$ **then**

        replace $\mathcal{Y}'$ in $L'$ by $\mathcal{Y}' \cap \overline{N}(y), \mathcal{Y}' \cap N(y)$;

2 **foreach** *class* $\mathcal{Y}'$ *of* $L'$ **do**

    Choose an arbitrary vertex $y' \in \mathcal{Y}'$;

    $pivot(\mathcal{Y}') = y'$;

    **foreach** *class* $\mathcal{Y}$ *of* $L$ **do**

      **if** $\mathcal{Y} \cap N(y') \neq \mathcal{Y}$ *and* $\mathcal{Y} \cap N(y') \neq \emptyset$ **then**

        replace $\mathcal{Y}$ in $L$ by $\mathcal{Y} \cap \overline{N}(y'), \mathcal{Y} \cap N(y')$;

    cograph($y'$,$\mathcal{Y}'$);

  **foreach** *class* $\mathcal{Y}$ *of* $L$ **do**

    choose an arbitrary vertex $y \in \mathcal{Y}$;

    cograph($y$,$\mathcal{Y}$);

  **if** $|\overline{N}(x)| > |N(x)|$ **then return** the ordered list $(L', \{x\}, L)$;

  **else return** the ordered list $(L, \{x\}, L')$;

**end**

To launch the process on a graph $G = (V, E)$, the first call to algorithm 1 is: *Cograph($v$,$V$)* where $v$ is an arbitrary vertex of $V$.

**Theorem 3** *Let $G = (V, E)$ be a graph. The algorithm 1 computes a factorizing permutation of $G$ iff $G$ is a cograph. It runs in $O(n + m \log n)$.*

The idea of the correctness has already been explained. Let us give the outlines of the complexity analysis. In loop 1, the neighborhood of each vertex of $L$ is used. Since the size of $L$ is less than half of the size of $\mathcal{X}$, in the whole process the neighborhood of a given vertex can be visited $\log n$ time. Loop 2 just uses the neighborhood of one vertex $y'$. Then $N(y')$ will be used once more to recursively launch the process. It means that in the whole process the neighborhood of each vertex is visited at most $\log n + 2$ times. This yields to the $O(n + m \log n)$ complexity.

# 4 Distance hereditary graphs algorithm

The expected result of Hammer and Maffray's algorithm [8] is to build a pruning sequence iff the input graph is distance hereditary. But it fails on the domino or the house if it starts with a degree 3 vertex. It means that the algorithm answers *"yes, there exist a pruning sequence"* although the domino and the house are not distance hereditary graphs.

To recognize distance hereditary graphs, let us use our two steps paradigm. The first step computes a pruning sequence as if the input graph is distance hereditary and the second just check its correctness. Cographs are distance hereditary graphs. Using the cotree, Hammer and Maffray proposed a linear algorithm that computes a pruning sequence of a cograph: if two vertices are sons of a same 0-node (resp. 1-node) then are true twins (resp. false twins). So algorithm 2 joined to the algorithm 4 can be the $\beta$ step of the paradigm for the cograph recongition.

**Algorithm 2:** Prune-cograph($G$,$j$)
**Input**: A cograph $G$
**Output**: A pruning sequence $\sigma$ iff $G$ is cograph
**begin**
  Compute a cotree $T$ of $G$;
  Let $S$ be the nodes of $T$ having only leaves as descendant;
  **while** $S \neq \emptyset$ **do**
    Pick an arbitrary node $N \in S$;
    Pick an arbitrary son $x$ of $N$;
    **For any** *son* $y \neq x$ *of* $N$ **do**
      **if** $N$ *is a 1-node* **then** $\sigma(j) \leftarrow (yTx)$;
      **else** $\sigma(j) \leftarrow (yFx)$;
      $j \leftarrow j + 1$;
    Replace $N$ by $x$ in $T$;
    **if** *the* $father(N)$ *has only leaves as descendant* **then**
      add $father(N)$ to $S$;
**end**

**Theorem 4** *[8] Algorithm 2 computes a pruning sequence of a cograph $G$ in $O(n + m)$.*

To get a linear complexity for algorithm 2, we have to build a cotree in linear time. Clearly, it can not be done by algorithm 1. But the expected complexity can be achieved using the classical Corneil, Pearl and Stewart's algorithm [4]. The rest of algorithm 2 can be compared to a postfix search in a tree. Since the size of the cotree is $O(n+m)$, the whole complexity is $O(n+m)$.

The distance hereditary graphs recognition algorithm is based on theorem 5. Let us introduce some notations. Let $G$ be a connected graph and $L_1, \ldots, L_k$ be the distance layout from an arbitrary vertex $v$ of $G$. For any vertex $x$ and for $1 \leq i \leq k$, we note by $N_i(x) = N(x) \cap L_i$. Let $x$ be a vertex of $L_i$, we call *inner degree* of $x$ the cardinality of $N_{i-1}(x)$.

**Theorem 5** *[1] Let $G$ be a connected graph and $L_1, \ldots, L_k$ be the distance layout from an arbitrary vertex $v$ of $G$. Then $G$ is a* distance hereditary graphs *iff the following conditions are verified for any $1 \leq i \leq k$:*

1. *If $x$ and $y$ belong to the same connected component of $G[L_i]$, then $N_{i-1}(x) = N_{i-1}(y)$*
2. *$G[L_i]$ is a cograph*
3. *If $x \in N_{i-1}(u)$ and $y \in N_{i-1}(u)$ are in different connected components $X$ and $Y$ of $G[L_{i-1}]$, then $X \cup Y \subseteq N(u)$ and $N_{i-2}(x) = N_{i-2}(y)$*
4. *If $x, y$ are in the different connected components of $G[L_i]$, then $N_{i-1}(x)$ and $N_{i-1}(y)$ are either disjoint or comparable for the inclusion order*
5. *If $x \in N_{i-1}(u)$ and $y \in N_{i-1}(u)$ are in the same connected component $C$ of $G[L_{i-1}]$, then the vertices of $C$ non-adjacent to $u$ are either adjacent to both $x$ and $y$ or none of them*

Algorithm 3 is the $\alpha$ step of the recognition paradigm for distance hereditary graphs. The $\beta$ step will be given by algorithm 4.

**Theorem 6** *Let $G$ be a graph. Algorithm 3 computes in linear time a pruning sequence of $G$ iff $G$ is distance hereditary.*

**Proof:** If the computed sequence is a pruning sequence, then $G$ is a distance hereditary graph (theorem 1). So we just have to prove the converse. During the $i$-th loop 1, all the vertices of the sets $L_j$ for $i < j \leq k$ has been removed.

By theorem 5.1, each connected component of $G[L_i]$ is a module. Since $G[L_i]$ is a cograph (theorem 5.2), twins in $G[L_i]$ are also twins in $G$. Thus in loop 2, we can contract each connected component $cc$ of $G[L_i]$ and build of pruning sequence of $G[cc]$ with the algorithm 2.

At this step $L_i$ is a stable set. Thus (loop 5) the remaining vertices of $L_i$ with inner degree 1 can be removed as pendant vertices.

Now by theorem 5.4, the neighborhood of two distinct vertices of $L_i$ are either disjoint or comparable. Let us consider a linear extension of the inclusion of these neighborhoods (ordering the vertices with respect to their inner degree produces such a linear extension). Let $x$ be the first vertex in this ordering. Let $u$ and $v$ be two distinct vertices of $N_{i-1}(x)$. By theorem 5.4, $N_i(u) = N_i(v)$. If $u$ and $v$ are in distinct connected component $cc(u)$ and $cc(v)$ of $L_{i-1}$, then $N_{i-2}(u) = N_{i-2}(v)$ (theorem 5.3). Moreover, $cc(u)$ and $cc(v)$ are included in $N_{i-1}(x)$. Finally, if $u$ and $v$ are in the same connected component $cc$ of $L_{i-1}$, theorem 5.4 shows that $N_{i-2}(u) = N_{i-2}(v)$ and by theorem 5.5, if $w \in cc$ is not adjacent to $x$ then $w$ is adjacent to both $u$ and $v$ or none of them. Therefore

**Algorithm 3:** Prune($G$)

**Input**: A graph $G = (V, E)$

**Output**: A pruning sequence $\sigma$ iff $G$ is a distance hereditary graph

```
begin
    pick an arbitrary vertex v;
    j ← 1;
    compute the distance layout L_1, ..., L_k from v;
1   for i = k downto 1 do
2       For each connected component cc of G[L_i] do
3           contract cc into a single vertex;
            prune-cograph(G[cc],j);
            j ← j + |cc| − 1;
4       sort the vertices of G[L_i] by increasing inner degree;
5       For each vertex x of L_i with inner degree 1 do
            Let y the only neighbor of x;
            σ(j) ← (xPy);
            j ← j + 1;
        if i ≠ 1 then
6           For each x ∈ L_i taken in increasing inner degree order do
7               contract N_{i−1}(x) into a single vertex;
                prune-cograph(G[N_{i−1}(x)],j);
                j ← j + |N_{i−1}(x)| − 1;
                Let y the only neighbor of x;
                σ(j) ← (xPy);
                j ← j + 1;
end
```

$N_{i-1}(x)$ is a module. Since it is contained in $L_{i-1}$, $G[N_{i-1}(x)]$ is a cograph. Twins in $G[N_{i-1}(x)]$ are twins in $G$: $N_{i-1}(x)$ can be contracted into a single vertex and compute a pruning sequence of $G[N_{i-1}(x)]$.

Respecting the linear extension of the neighborhoods, the previous argument can be applied for all remaining vertices of $L_i$. That ends the correctness proof.

Let us now give some ideas for the complexity issues. Computing the distance layouts can be done in $O(n + m)$ via a breadth first search. At line 3, each connected component of $G[L_i]$ into a single vertex can be contracted in $O(G[l_i])$. Sorting the vertices of $L_i$ with respect to their inner degree (line 4) can also be done in linear by a bucket sort. For each distance layout, the global complexity of line 7 is $O(G[L_i \cup L_{i-1}])$. Thus the whole complexity is $O(n + m)$. □

To make the recognition procedure complete, let us now present the verification algorithm ($\beta$ step of the recognition paradigm). Each vertex and his neighborhood are visited at most twice. This yields to a linear complexity.

**Theorem 7** *Distance hereditary graphs can be recognized in linear time*

**Algorithm 4:** Verification-step($G$,$\gamma$)
**Input**: A graph $G = (V, E)$ and a sequence $\gamma = [s_1, \ldots, s_n]$
**Output**: True iff $\gamma$ is a pruning sequence of $G$
**begin**
  **for** $j = n$ *downto 2* **do**
    $xSy \leftarrow s_j$;
    **if** $S$=$P$ **then**
      **if** $|N(x)| \neq 1$ *or* $(x, y) \notin E$ **then return** false;
    **else**
      **if** $N(x) \cap \{x_1, \ldots, x_{j-1}\} \neq N(y) \cap \{x_1, \ldots, x_{j-1}\}$ **then**
        **return** false;
    **return** true;
**end**

# 5 Conclusions

Based on the *Compute as if and Check after* paradigm, two new recognition algorithms for cographs and distance hereditary graphs have been presented. These algorithms seem to be simpler than the already known algorithms. For distance hereditary graphs, we put right Hammer and Maffray's algorithm. For cographs, although the optimal complexity is not achieved, the data structures of presented algorithm are simpler than those of Corneil Pearl and Stewart algorithm. The next problem is now to improve this complexity without using complicated data structures.

Many other applications of this paradigm could have been quoted (as for example the recognition of distributive lattices [6] in linear time) and we hope that it will be helpful in the near future to design new algorithms.

# References


[1] H.J. Bandelt and H.M Mulder. Distance hereditary graphs. *Journ. of Combin. Theory serie B*, 41:182–208, 1986.

[2] C. Capelle. *Decomposition de graphes et permutations factorisantes*. PhD thesis, Univ. de Montpellier II, 1996.

[3] D.G Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm. Extended abstract, 1997.

[4] D.G. Corneil, Y. Pearl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal of Computing*, 14(4):926–934, Novenber 1985.

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

[6] M. Habib et L. Nourine. On some tree representation for distributive lattices. Technical Report 93042, LIRMM, 93.

[7] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Sc.* to appear.

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

[9] R.M. McConnell and J.P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of undirected graphs. In *SODA*, pages 19–35, 1997.

[10] Donald J. Rose, R. Endre Tarjan, and George S. Leuker. Algorithmic aspects of vertex elimination on graphs. *SIAM Jour. of Comp.*, 5(2):266–283, 1976.

[11] R.E. Tarjan and M. Yannakakis. Simple linear algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergaphs. *SIAM Jour. of Comp.*, 13:566–579, 1984.