

Proposition Let G be a chordal graph and $\sigma: V(G) \rightarrow \{1, 2, \dots, n\}$ be an MAO in G with $\sigma(v_i) = i, \forall i \in 1, \dots, n, n = |V(G)|$.

Then $v_n, v_{n-1}, \dots, v_1$ is a PES in G .

Proof Let $\tilde{\sigma}: V(G) \rightarrow \{1, 2, \dots, n\}$ with $\tilde{\sigma}(v_i) = n - i + 1, \forall i \in 1, \dots, n$

We observe that $\tilde{\sigma}$ has the following property:

$$(*) \quad \forall u, v, w \in V(G) \quad \left. \begin{array}{l} \tilde{\sigma}(u) < \tilde{\sigma}(v) < \tilde{\sigma}(w) \\ \{u, w\} \in E(G), \{v, w\} \notin E(G) \end{array} \right\} \Rightarrow \begin{array}{l} \exists z \in V(G) \\ \text{with } \tilde{\sigma}(w) < \tilde{\sigma}(z) \\ \{v, z\} \in E(G) \\ \text{and } \{u, z\} \notin E(G) \end{array}$$

Indeed, the MAO label of v is smaller than that of u ($\sigma(v) < \sigma(u)$)

although u is adjacent to the already labelled w ($\sigma(w) < \sigma(u)$ and $\sigma(w) < \sigma(v)$) and v is not. Since v at least

as many neighbors with MAO label smaller than

$\sigma(v)$. As the number of neighbors of u with PSE-label $< \sigma(v)$. Thus there must exist a neighbor z of v with $\sigma(z) < \sigma(v)$ which is not a neighbor of u .

Came to far on May 13, 2020

Assume now that $\tilde{\sigma}$ is not a PSE. According to the previous proposition $\exists$ 2 vertices $v, w \in V(G), \{v, w\} \notin E(G)$ which are connected by a v - w -path P with $\tilde{\sigma}(x) < \min\{\tilde{\sigma}(v), \tilde{\sigma}(w)\}$

$\forall x \in P$. [1] Choose two such vertices v, w such that $\tilde{\sigma}(v)$ is as large as possible, and if there are many possibilities for w , choose it also such as $\tilde{\sigma}(w)$ is as large as possible. [2] Let P be the shortest v - w -path with the above property ($\forall x \in P, \tilde{\sigma}(x) < \min\{\tilde{\sigma}(v), \tilde{\sigma}(w)\}$).

Since $\{v, w\} \notin E(G)$, P contains at least one inner vertex.

Let $u \in P$ with $\{u, w\} \in E(G)$



We have $\tilde{\sigma}(v) > \tilde{\sigma}(w) > \tilde{\sigma}(u)$ by construction.

and $\{u, x\} \in E(G), \{v, w\} \notin E(G)$

According to (*) $\exists z \in V(G)$ with $\tilde{\sigma}(w) < \tilde{\sigma}(z), \{w, z\} \in E(G), \{u, z\} \notin E(G)$

But then ~~is~~ P_1 (extended by $\{w, z\}$) is a $v-z$ -path with $\tilde{\sigma}(x) < \min\{\tilde{\sigma}(z), \tilde{\sigma}(w)\}$, $\forall x \in P_1$ and according to the choice of v we have $\{z, w\} \in E(G)$. Hence P_1 and $\{z, w\}$ build a cycle C in G . This cycle must have a chord because G is chordal and the cycle contains 4 different vertices (v, w, u, z) . According to the choice of P (as a shortest path with the respective properties) every chord of C is incident with z . Let x be the ~~last~~ ^{first} vertex in the ~~$v-u$~~ ^{non-trivial $v-u$} path in C (i.e. the ~~$v-u$~~ path consisting more than 1 edge, ^{green}) which is adjacent to z . Since $\{u, z\} \notin E(G)$ we have $x \neq u$ and $(x, \dots, u, w, z)$ builds an induced cycle C_1 of length ≥ 4 . (blue) □

Summarizing we have:

- (*) A chordal graph has a PSE obtained by reversing the MAO.
- (**) If a graph is chordal, then it has a PSE.

These statements imply:

A graph is chordal $\Leftrightarrow$ the reversed MAO is a PSE.

So two steps are needed to check whether a given graph G is a PSE: Chordal

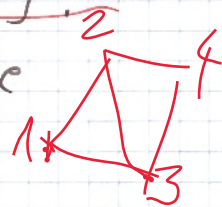
(1) Compute an MAO τ in G

(2) Check whether the reversed MAO ~~is~~, denoted by $\tilde{\tau}$, is a PSE.

How to perform these 2 steps efficiently?

(1) An MAO τ can be computed in $O(m + n \log n)$ time by using Fibonacci heaps.

Indeed let $|G| = n$ and $m := |E(G)|$.



set $\alpha(v) := 0 \quad \forall v \in V(G); \text{ list} := \{v_1, \dots, v_n\}$

for $i=1$ to n do

begin

(1) select $v \in \text{list}$ with $\alpha(v) = \max \{ \alpha(x) : x \in \text{list} \}$

(2) set $\tilde{\sigma}(v) := i$; delete v from list;

(3) set $\alpha(u) := \alpha(u) + 1 \quad \forall u \in \text{list mit } \{u, v\} \in E(G)$
 $\#$ of currently labelled neighbors $\neq$

end

(3) takes $O(\deg(v))$ time, hence altogether $O(m)$ for all vertices in $V(G)$.

(1)+(2) take $O(\log n)$ time with Fibonacci heaps (finding the element of the list with the largest key and deleting it from the list).

Once an MAO σ has been constructed, we set

$$\tilde{\sigma}(\sigma^{-1}/(n-i+1)) = i \quad \forall i \in \overline{1, n}$$

where $\sigma^{-1}(n-i+1)$ is the vertex $v \in V$ mapped by σ to $n-i+1$, i.e. $\sigma(v) = n-i+1$.

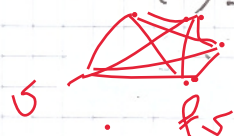
(2) Then we check whether $\tilde{\sigma}$ is a PSE, this can be done in $O(m+n)$ time

The upper neighborhoods $UN(\tilde{\sigma}^{-1}(i))$ can be computed in a total of $O(m+n)$ time for all $i \in \overline{1, n}$, if G is specified in terms of the adjacency list $\text{Adj}(v)$, $\forall v \in V(G)$:

to this end we just go through the list $\text{Adj}(v)$ and check/select all vertices u in this list for which $\tilde{\sigma}(u) > \tilde{\sigma}(v)$.

It is more difficult to efficiently check whether $UN(v)$ is a clique $\forall v \in V(G)$; it should be avoided checking the existence of the same edge many times. This can

be realized by introducing the so called UN-leader f_v $\forall v \in V(G)$:
 $f_v = \arg \min \{ \tilde{\sigma}(x) : x \in UN(v) \}$.



Clearly $(*) UN(v) \setminus \{f_v\} \subseteq UN(f_v)$ must hold $\forall v \in V(G)$

This is a ~~partial~~ requirement of " $UN(v)$ induces a clique". The idea is to collect ^{in a set} $A(f_v)$ for which the adjacency to f_v for some $v \in V(G)$ needs to be checked, and perform the adjacency test of all elements of $A(f_v)$ to f_v at once (of course $UN(v) \cap f_v \subseteq A(f_v)$ holds $\forall v \in V$). This procedure is implemented by the function $PSE(G, \sigma)$ below. It takes as input a graph G and a labelling $\sigma: V(G) \rightarrow \{1, 2, \dots, |G|\}$ and checks whether σ is a PSE.

(12)

function $PSE(G, \sigma): \text{boolean}$

begin

for $v \in V(G)$ $A(v) := \emptyset$

for $i = 1$ to n do

begin

$u := \sigma^{-1}(i)$

$UN := \{w \in \text{Adj}(u) : \sigma^{-1}(w) > \sigma^{-1}(u)\}$

if $A(v) \neq UN$ then

begin

| $PSE := \text{FALSE}; \text{EXIT};$

end

if $UN \neq \emptyset$ then

begin

| $f := \text{argmin} \{ \sigma(w) : w \in UN \}$

| $A(f) := A(f) \cup (UN \setminus \{f\})$

end

End /* for i */

$PSE := \text{TRUE}$

end Correctness :

Correctness: trivial because all required adjacencies for $UN(v)$ being a clique are checked in turn;

In order let e.g. $UN(v) = \{x_1, \dots, x_k\}$ with
 $\sigma(x_1) < \sigma(x_2) < \dots < \sigma(x_k)$ for some $v \in V(G)$ and
some $k \in \mathbb{N}$.

then $UN(v)$ induces a clique $\Leftrightarrow$

$$\begin{aligned} UN(v) \setminus \{x_1\} &\subseteq UN(x_1) \quad \checkmark & x_1 \\ UN(v) \setminus \{x_1, x_2\} &\subseteq UN(x_2) \quad \checkmark & x_2 \\ & & \vdots \\ UN(v) \setminus \{x_1, \dots, x_{k-1}\} &\subseteq UN(x_{k-1}) \quad \checkmark & x_{k-1} \end{aligned}$$

and all these inclusions are tested in the course of the algorithm, in the iteration $\sigma(x_1), \sigma(x_2), \dots$, and $\sigma(x_k)$, respectively.

Time complexity: All operations except for the test $A(v) \subseteq UN$ can be performed in $O(n+m)$ time (if UN is implemented as a single linked list).
(n repetitions of the for loop with $O(\text{Adj}(v))$ in repetition $\sigma(v)$, $\forall v \in V(G)$).

In order to implement the above test efficiently we use an n -dimensional vector TEST with the initialisation $TEST[i] = 0, \forall i \in \overline{1, n}$.

Then $A(v) \subseteq UN$ can be performed as follows:

```
IS-SUBSET := TRUE
for  $x \in UN$  set  $test(x) = 1$ 
for  $y \in A(v)$  do
  begin
    if  $Test(y) = 0$  then IS-SUBSET = FALSE
  end
Return IS-SUBSET
```

Since the overall number of elements appearing in all UN is bounded by $|E(G)| = m$ and the set $A(v)$ are filled with elements of UN s, in the ^{overall} number of operations involving $A(v)$ also bounded by $|E(G)|$.
We can think of $A(v)$ as a multiset where the ^{new} elements are added at its end even if they are already contained in $A(v)$.
 $A(v)$ is also impl. as a single linked list.

Summarizing we obtain

Theorem Chordal graphs on n vertices and m edges can be recognized in $O(m + n \log n)$ time. If an MAO is given, the recognition can be done in $O(m + n)$ time.

Linear algorithms to determine $\chi(G)$, $\omega(G)$, $\alpha(G)$ and $\theta(G)$ for a chordal graph G with PSE σ

Consider a chordal graph G and a PSE σ in it.

The set $C_v = \{v\} \cup N(v)$ define a clique $\forall v \in V(G)$, by definition of PSE. Moreover among these cliques are all maximal cliques, i.e.

if maximal clique $C \exists v \in V(G)$ with $C = C_v$.

In fact let $v = \arg \min \{v(x) : x \in C\}$. Then $C \subseteq \{v\} \cup N(v)$ and since C is a maximal clique the inclusion above is fulfilled with equality. Thus we get

Observation (Fulkerson 1965, Gross 1965)

A chordal graph on n vertices has at most n maximal cliques (with equality holding iff $E(G) = \emptyset$).

easy to see starting with v_n the last vertex in PSE.

(PSE: $v_1 \dots v_n$). If the graph has n maximal cliques, then each $\{v\} \cup N(v)$ is a max. clique (and they are pairwise disjoint). Thus $\{v_n\} \cup N(v_n) = \{v_n\}$ in a max. clique $\Rightarrow N(v_n) = \emptyset$.

Repeat the arg. with $G[\{v_1 \dots v_{n-1}\}]$, get $N(v_{n-1}) = \emptyset$ and so on.

Computing $\omega(G)$: Find the largest clique among $v \cup N(v)$ let it be $v^* \cup N(v^*)$. Then $\omega(G) = |v^* \cup N(v^*)|$.

Thus if we have a PSE $\omega(G)$ can be computed in linear time.

Computing $\chi(G)$: Exercises

Determining a maximum stable set and $\alpha(G)$ and $\Theta(G)$ and

set $v_i = \sigma^{-1}(i) \quad \forall i \in \overline{n}, n := |G|$. an optimal clique cover

set $s_1 = v_1$ and define s_k inductively by

$\forall k \geq 2 \quad s_k = \argmin \{ \sigma(v) : \sigma(v) > \sigma(s_{k-1}) \text{ and } v \notin \bigcup_{i=1}^{k-1} UN(s_i) \}$

then s_k is the vertex with the smallest label among all vertices with label larger than s_{k-1} which are not adjacent to any of $s_1, \dots, s_{k-1}$.

let ℓ be the largest index of some vertex s defined in this (since the set of vertices is finite, the above process terminates).

Then $S = \{s_1, \dots, s_\ell\}$ is a stable set in G because $\forall i, j$
 $i < j \Rightarrow \{v_i, v_j\} \notin E(G)$ because s_j is not neighbored to any of $s_1, \dots, s_i, \dots, s_{j-1}$.

Moreover $\mathcal{J} = \{ \underbrace{s_k \cup UN(s_k)}_{C_k} \}_{1 \leq k \leq \ell}$ is a clique cover of

G because $s_k \cup UN(s_k)$ is a clique $\forall k \in \overline{\ell}$ and if there is a $v \in V(G)$ not in $\bigcup_{k=1}^{\ell} \underbrace{s_k \cup UN(s_k)}_{C_k}$

then let $\overline{\sigma(v)}$ be an index r, k .

$\sigma(s_1) \leq \dots < \sigma(s_i) < \sigma(v) < \sigma(s_{i+1}) \dots < \sigma(s_\ell)$

Then in iteration $i+1$ of the procedure above v would have been selected instead of s_{i+1} . ∇

So $\mathcal{J}$ is a clique cover with $|\mathcal{J}| = |S| \Rightarrow$

$\Theta(G) \leq \alpha(G)$. Since $\Theta(G) \geq \alpha(G)$ always holds

we have $\Theta(G) = \alpha(G)$ and thus both the clique cover $\mathcal{J}$ and the stable set S are optimal. $\square$

Two other families of perfect graphs

(16)

Comparability graphs

Def A graph $G=(V,E)$ is called a comparability graph iff there exists a partial order $P_{\leq}$ on $V(G)$ such that $\forall \{u,v\} \in E(G)$ u and v are in relation $P_{\leq}$, i.e. either $u P_{\leq} v$ or $v P_{\leq} u$.

Example Every bipartite graph $G=(V \cup U, E)$ is a comparability graph: define a partial order $P_{\leq}$ on $V \cup U$ by $\forall x,y \in U \cup V$ $x P_{\leq} y \iff (x,y) \in E, x \in U, y \in V$ or $x=y$.
 $x P_{\leq} y \wedge y P_{\leq} z \Rightarrow x P_{\leq} z$

Homework: check that the above relation $P_{\leq}$ is a partial order on $U \cup V$.
 $x P_{\leq} y \wedge y P_{\leq} x \Rightarrow x=y$

Lemma A graph G is a comparability graph iff G is transitively orientable, i.e. $\exists$ orientation $O: E \rightarrow V \times V$ of its edges such that $(x,y) \in O \wedge (y,z) \in O \Rightarrow (x,z) \in O$.
Proof: exercises

Example A transitive orientation of the edges of a bipartite graph would orient every edge from U to V .

There exists an $O(n+m)$ algorithm which computes a transitive orientation of the edges of an (undirected) comparability graph.

J.P. Spinrad, P_4 trees and substitution decomposition, Discrete Applied Mathematics 38, 1992, 263-281.

Theorem Let $P_{\leq}$ be a partial order on a finite set M .

The length of a longest chain in $(M, P_{\leq})$ equals the min. number of antichains in a covering of M by antichains in $(M, P_{\leq})$.

(no proof) is the dual formulation of the Dilworth's theorem. Noting that the largest antichain has the same size as the smallest chain decomposition.
Corollary Comparability graphs are perfect.

Proof: Cliques correspond to chains and stable set correspond to antichains in comp. graphs.

Interval graphs

17

Def A graph G is called an interval graph if it is the intersection graph of a family of closed intervals $I = ([a_i, b_i])_{i \in I}$ on the real line, where $a_i, b_i \in \mathbb{R}$, $a_i \leq b_i$ and I an index set, i.e. $\exists$ bijection $f: V(G) \rightarrow I$ such that $\{u, w\} \in E(G) \Leftrightarrow f(u) \cap f(w) \neq \emptyset$.

Proposition: Interval graphs are chordal
(proof in the exercise)

Theorem (Gilmore, Hoffman 1964)

For a given graph $G = (V, E)$ the following statements are equivalent

- (i) G is an interval graph for $k \geq 4$,
- (ii) G contains no induced C_k ✓ and $\bar{G}$ is a comparability graph ✓

(iii) The maximum cliques of G can be ordered linearly such that $\forall v \in V$ all maximal cliques containing v build a contiguous interval in that ordering.

In other words, $\exists$ an ordering $C_1, C_2, \dots, C_k$ of all max. cliques in G such that $\forall v \in V(G)$, $\exists i_v, k_v$ with $C_{i_v}, C_{i_v+1}, \dots, C_{i_v+k_v}$ being all max. cliques which contain v .

(no proof)

k. Simon, A simple linear time algorithm to recognize interval graphs, H. Bieri and H. Noltemeier, eds.,

Lecture Notes in Computer Science 553, 1991, 289-308

Proposition: Interval graphs can be recognized in linear time i.e. in $O(m+n)$ where n is the number of vertices and m is the number of edges.