ARBORICITY, h-INDEX and GRAPH ALGORITHMS

Min C. Lin Francisco Soulignac

Universidad de Buenos Aires, Argentina

Jayme L. Szwarcfiter

Universidade Federal do Rio de Janeiro, Brazil

Cornell University, May 2012

Theoretical Computer Science, to appear

Motivation

- Describe a new data structure useful for developing (dynamic) graph algorithms
- Apply the data structure to formulate new algorithms (static and dynamic) for several graph problems
- The proposed dynamic algorithms are the first of their kind in the literature
- The proposed static algorithms improve the complexities over the known algorithms, for sparse graphs (general improvement, for some algorithms)
- Main concepts: arboricity and h-index

Applications (Problems)

1. Listing all the cliques of size k.
2. Counting the number of induced subgraphs, isomorphic to any graph of four vertices.
3. Counting the number of K_4's, diamonds, paws and claws, containing a given vertex.
4. Recognizing diamond-free graphs
5. Finding simple, simplicial and dominated vertices.
6. Recognizing cop-win graphs, and finding a dismantling ordering.
7. Recognizing strongly chordal graphs, and finding a simple elimination ordering.

Contents

- Notation and basic propositions
- The h-graph data structure
- The 4-subgraph counting problem
- Dynamic recognition of diamond-free graphs
- Finding simple, simplicial and dominated vertices
- Recognizing cop-win graphs
- Recognizing strongly chordal graphs
- More applications

Related Results

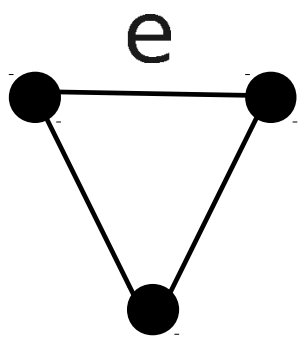
- Chiba and Nishizeki (1985)
- Kloks, Kratsch and Muller (2000)
- Eppstein and Spiro (2009)

Notation

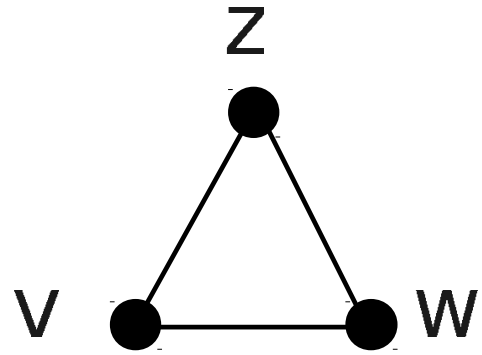
$G, V(G), E(G), N(v), N[v], n = |V(G)|, m = |E(G)|$

$N'(v)$, **edge neighborhood of** v,
set of edges having both extremes adjacent to v

$N(vw)$, **neighborhood of edge** vw,
set of vertices adjacent to both v, w



e ∈ N'(v)



z ∈ N(vw)

Notation

$d(v) = |N(v)|$, **degree of** v

$H(v) = \{w \in N(v)$ where $d(w) > d(v)\}$

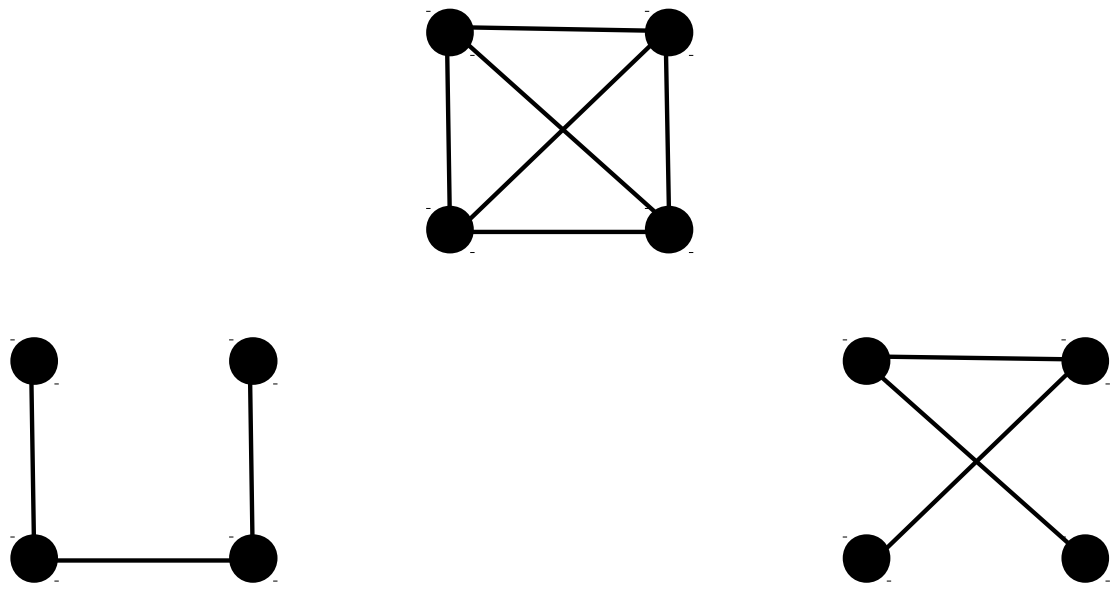
$d'(v) = |N'(v)|$, **edge-degree of** v

$d(vw) = |N(vw)|$, **degree of** vw

n^ω, time required to multiply two $n \times n$ matrices

Main Actor: Arboricity

$\alpha(G)$, **arboricity of** G,
minimum number of edge disjoint spanning forests which decomposes G



Main Actor: Arboricity

Characterized by Nash-Williams (1961)
Several graphs have bounded arboricity, as:
- graphs with bounded degrees
- graphs with bounded genus
- planar graphs have $\alpha \leq 3$
- outerplanar graphs have $\alpha \leq 2$

Main Actor: h-index

$h(G)$, **h-index of G,**
maximum h, s.t. G contains h vertices of degree at least h

Introduced by Eppstein and Spiro (2009)

"bibliographic" h-index, Hirsch (2005)

Basic Propositions

Theorem 1

$$\frac{\delta}{2} < \frac{m}{n-1} \leq \alpha(G) \leq h(G) \leq \sqrt{2m}$$

Proof $\alpha(G) \leq h(G)$

Induction on n.
When $n = 2$, trivial.
G, order $n > 2$
v vertex degree δ
$G' = G - v$. Then
$\alpha(G') \leq \alpha(G) \leq \alpha(G') + 1$ and
$h(G') \leq h(G) \leq h(G') + 1$

$\mathcal{D}'$, decomp G' into $h(G')$ forests

Case 1: $\delta < h(G)$

Then $\delta \leq h(G')$.
Induction hyp: $\alpha(G') \leq h(G')$. Then
Transform $\mathcal{D}'$ into a forest decomp of G:
- add v to each of the forests
- add one edge vw to exactly δ forests of $\mathcal{D}'$

Hence $\alpha(G) \leq h(G') \leq h(G)$

Case 2: $\delta \leq h(G)$

Then $\delta = h(G)$

If $h(G) = h(G')$ then $\delta = h(G')$, proceed as Case 1

Otherwise $h(G) = h(G') + 1$.

Transform $\mathcal{D}'$ into a decomp of G:

- add v as an isolated in each of the forests of $\mathcal{D}'$
- add a new forest F, where
  - (i) $d_F(v) = \delta$
  - (ii) the neighbors of v have degree 1
  - (iii) no other edges.

Hence $\alpha(G) \leq \alpha(G') + 1 \leq h(G') + 1 = h(G)$

Basic Propositions

Lemma 1 (Chiba and Nishizeki 1985)

$$\sum_{vw \in E(G)} \min\{d(v), d(w)\} \leq 2\alpha(G)m.$$

$\mathcal{F}$, decomposition of G into α disjoint spanning forests F_i.
Consider each tree T of $\mathcal{F}$ as rooted (arbitrary root).
Associate each edge e of T to its head vertex $h(e)$ in T. Then

$$\sum_{vw \in E(G)} \min\{d(v), d(w)\} \leq \sum_{1 \leq i \leq \alpha} \sum_{e \in E(F_i)} d(h(e))$$

$$\leq \sum_{1 \leq i \leq \alpha} \sum_{v \in V(G)} d(v) \leq 2\alpha m$$

Basic Propositions

Lemma 2 *Let $e_1, \ldots, e_m$ be an ordering of $E(G)$ for a graph G, and call $e_i = v_i w_i$. Denote by $|H_i(v)|$ the value of $|H(v)|$ in the subgraph of G that contains the edges $e_1, \ldots, e_i$, for every $1 \leq i \leq m$. Then,*

$$\sum_{i=1}^{m} |H_i(v_i)| \leq 4\alpha(G)m.$$

Lemma 3 *For every graph G,*

$$\sum_{v \in V(G)} \sum_{w \in N(v)} |H(w)| \leq 8\alpha(G)m.$$

Further Notation

$N(v, i)$, set of neighbors of v having degree i
$\mathcal{N} = \{N(v, i) | N(v, i) \neq \emptyset$ and $i < d(v)\}$

Vertex v can have at most $h(G)$ neighbors of degree at least $d(v) + 1$. Then $|H(v)| \leq h(G)$.

The h-graph Data Structure

For a graph G, the h-graph data structure consists of a triple for each vertex v:

$$(d(v), \mathcal{N}(v), H(v))$$

Supported Operations

Operation	Description	Complexity	
		One	All
Vertex insertion	Inserts new vertex v with given $N(v)$	$O(dh)$	$O(\alpha m)$
Vertex removal	Removes vertex v	$O(dh)$	$O(\alpha m)$
Edge insertion	Inserts new edge vw	$O(h)$	$O(\alpha m)$
Edge removal	Removes edge vw	$O(h)$	$O(\alpha m)$
Adjacency query	Queries if two vertices v, w are adjacent	$O(h)$	-
Finding $H(v)$	Returns set $H(v)$	$O(1)$	-
Finding $N'(v)$	Returns set $N'(v)$	$O(dh)$	$O(\alpha m)$
Constructing $G[N(v)]$	Constructs the graph $G[N(v)]$	$O(dh)$	$O(\alpha m)$

Operations - Brief Description

Insertion of edges. The algorithm for inserting a new edge vw into G has two phases. In the first one, we update the families $\mathcal{N}(z)$, for every $z \in N[v] \cup N[w]$. On the other hand, the second phase actually inserts the new edge into the sets $N(w, d_G(v) + 1)$ and $N(v, d_G(w) + 1)$, while updating the values of $d(v)$ and $d(w)$.

Insertion of vertices. For inserting a new vertex v, with given neighborhood $N(v)$, first, insert v as an isolated vertex, and then add the edges vw, for each $w \in N(n)$.

Operations - Brief Description

Removal of edges. For removing an edge vw, we undo the insertion process. But, this time, we start by the computations corresponding to the second phase. That is, first we physically remove the edge vw. Then we proceed to undoing the first phase. In this case, we need to update the families $\mathcal{N}_z$, for every $z \in N_G(v) \cup N_G(w)$.

Removal of vertices. Similarly as before, remove vertex v by removing all its incident edges first. At the end, remove v after becoming an isolated vertex.

Adjacency query. Querying whether two vertices v and w are adjacent is straightforward. Simply, traverse the set $H(z)$, where z is the vertex of least degree, between v and w.

Operations - Brief Description

Finding $H(v)$. Trivial, since in the data structure $H(v)$ belongs to the triple of v

Finding $N'(v)$. To construct $N'(v)$, we ought to traverse $N(v)$ and then $H(w)$, for each $w \in N(v)$.

Constructing $G[N(v)]$. Clearly, the subgraph of G induced by $N(v)$ is just the graph whose vertex set is $N(v)$, which is obtained from $\mathcal{N}(v)$ and $H(v)$, and whose edge set is $N'(v)$, which can be determined as above.

Edge Insertion - vw

Two phases:

1) Create set $\mathtt{N}(v, d_G(v))$, move the vertices with degree $d_G(v)$ from $\mathtt{H}(v)$ to $\mathtt{N}(v, d_G(v))$, and move v from $\mathtt{N}(z, d_G(v))$ to $\mathtt{N}(z, d_G(v)+1)$, for every $z \in H(v)$. Apply analogous operations for w.

2) insert v at the end of $\mathtt{N}(w, d_G(v)+1)$ and w at the end of $\mathtt{N}(v, d_G(w)+1)$, update the values of $\mathtt{d}(v)$ and $\mathtt{d}(w)$.

Edge Insertion - Complexity

First phase:
$O(|H(v)| + |H(w)|)$

Second phase:
$O(min\{d(v), d(w), h(G)\})$

Total complexity:
$O(|H(v)| + |H(w)| + min\{d(v), d(w), h(G)\})$

Vertex insertion

Complexity for inserting one vertex v:

$O(d(v)h(G))$

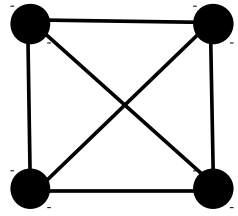
Constructing the h-graph data structure

Insert all n vertices and m edges.

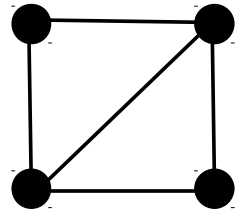
Complexity: $O(\alpha m)$

The 4-subgraph counting problem

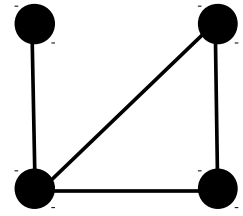
CONNECTED



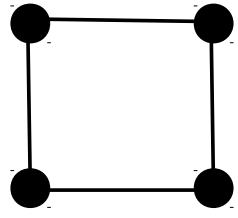
K_4



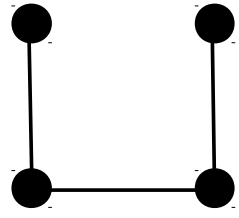
diamond



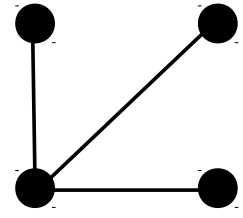
paw



C_4



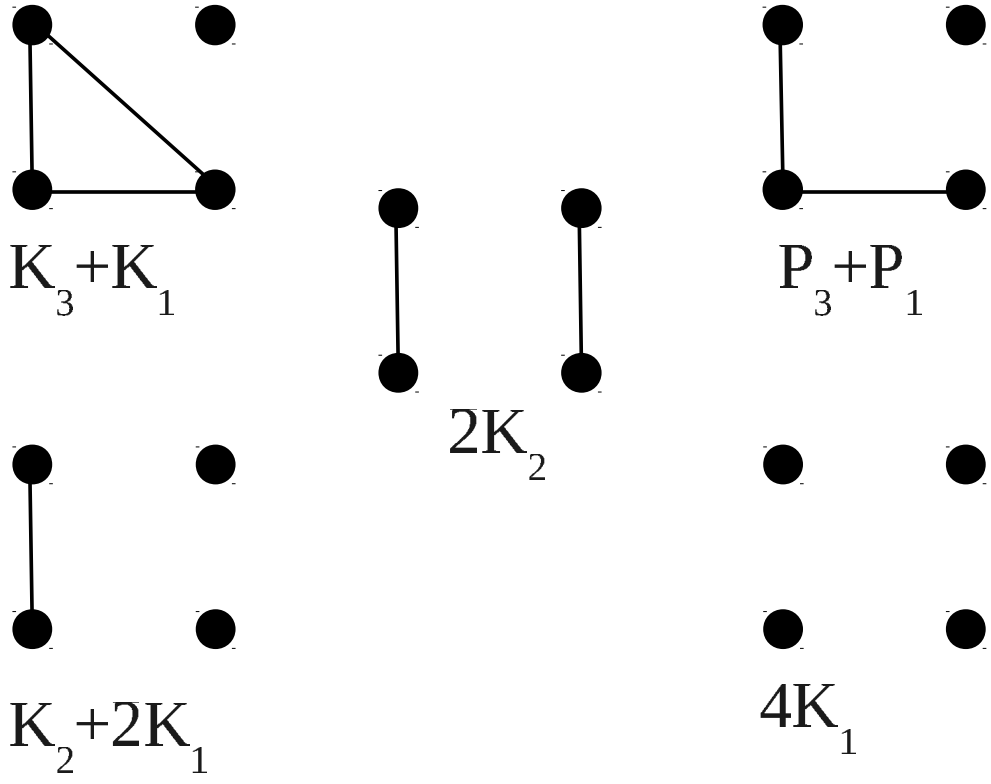
P_4



claw

The 4-subgraph counting problem

DISCONNECTED



./

The known method

Theorem 2 (Kloks Kratsch & Muller 2000) *Let $\tilde{H}$ be a connected graph on four vertices such that there is an $O(t(G))$ time algorithm counting the number of induced $\tilde{H}$'s in a graph G. Then, there is an $O(n^\omega + t(G))$ time algorithm counting the number of induced H's of G for all connected graphs H on four vertices.*

The known method

Algorithms for computing the number of $K_4's$ of a graph:

$O(n + m^{1.61})$, $O(n + \alpha^2 m)$
Chiba and Nishizeki (1985)
Kloks, Kratsch and Muller (2000)

Counting connected subgraphs of size 4:

$O(n^\omega + min\{m^{1.61}, \alpha^2 m\})$

The proposed method

Theorem 3 *Let $\tilde{H}$ be a graph on four vertices such that there is an $O(t(G))$ time algorithm counting the number of induced $\tilde{H}$'s in a graph G. Then, there is an $O(n + \alpha(G)m + t(G))$ time algorithm counting the number of induced H's of G for every graph H on four vertices.*

Unknows

Given G, find:

$k = \#\ K_4$'s $\qquad$ $d = \#$ diamonds

$s = \#\ C_4$'s $\qquad$ $p = \#\ P_4$'s

$q = \#$ paws $\qquad$ $y = \#$ claws

$\overline{k} = \#\ 4K_1$'s $\qquad$ $\overline{d} = \#\ K_2 + 2K_1$'s

$\overline{s} = \#\ 2K_2$'s $\qquad$ $\overline{q} = \#\ P_3 + K_1$'s

$\overline{y} = \#\ K_3 + K_1$'s

Notation

For a graph G,

$\overline{m} = |E(\overline{G})|$

$\overline{d(v)}$ = degree of v in $\overline{G}$

$\delta(v, w)$ = number vertices adjacent to v and not w

L = set of vertices, any two of them forming a C_4 with two fixed non-adjacent vertices v, w.

$\mathcal{S}$ = set of triples (v, w, L)

System of Equations (1-5)

$$\sum_{(v,w,L)\in\mathcal{S}} \binom{|L|}{2} = 3k + d + s \tag{1}$$

$$\sum_{vw\in E(G)} \binom{d(vw)}{2} = 6k + d \tag{2}$$

$$\sum_{vw\in E(G)} \delta(v,w)\delta(w,v) = 4s + p \tag{3}$$

$$\sum_{vw\in E(G)} \left(\binom{\delta(v,w)}{2} + \binom{\delta(w,v)}{2} \right) = q + 3y \tag{4}$$

$$\sum_{vw\in E(G)} \binom{d(v) + d(w) - d(vw) - 2}{2} = 6k + 5d + 4s + p + 3q + 3y \tag{5}$$

System of Equations (6-10)

$$\sum_{v \in V(G)} d'(v)(n-3) = 12k + 6d + 3q + 3\bar{y} \tag{6}$$

$$\sum_{v \in V(G)} \binom{d(v)}{2}(n-3) = 12k + 8d + 4s + 2p + 5q + 3y + \bar{q} + 3\bar{y} \tag{7}$$

$$\binom{m}{2} - \sum_{v \in V(G)} \binom{d(v)}{2} = 3k + 2d + 2s + p + q + \bar{s} \tag{8}$$

$$\binom{\overline{m}}{2} - \sum_{v \in V(G)} \binom{\overline{d(v)}}{2} = s + p + 3\bar{k} + 2\bar{d} + 2\bar{s} + \bar{q} \tag{9}$$

$$\binom{n}{4} = k + d + s + p + q + y + \bar{k} + \bar{d} + \bar{s} + \bar{q} + \bar{y} \tag{10}$$

./

Equation (2)

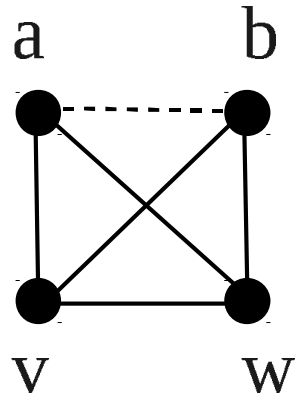
$$\sum_{vw \in E(G)} \binom{d(vw)}{2} = 6k + d$$

Let $vw \in E(G)$ and $\binom{d(vw)}{2} = \ell$

$\Rightarrow \exists \ell$ pairs of vertices $a, b \in V(G), a \neq b$, s.t.

$a, b \in N(v) \cap N(w) \Rightarrow$

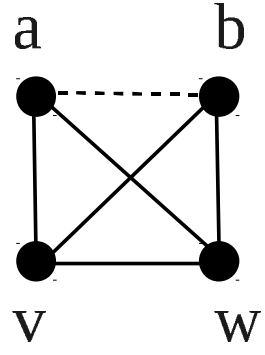
a, b, v, w induce a K_4 (if $a, b \in E(G)$) or a diamond, otherwise.



Equation (2)

- Each K_4 is counted 6 times (each edge of the K_4 takes the role of vw)
- Each diamond is counted exactly once (when v, w are the vertices of degree 3 of the diamond). Then

$$\sum_{vw \in E(G)} \binom{d(vw)}{2} = 6k + d$$



Constant terms

To compute the constant (left side) terms of the equations we require:
$O(n + \alpha m)$, for equations (1-6)

$O(n + m)$, for equations (7-10)

Solving the system

Claim 1 *The equations are linearly independent*

There are 10 equations and 11 variables:

$$k, d, s, p, q, y, \overline{k}, \overline{d}, \overline{s}, \overline{q}, \overline{y}$$

Adding an 11th equation of the form

$$value = k$$

the system becomes triangular as:
Each unknown is then determined by the values of the variables that precede it in the sequence.

By fixing the number of subgraphs $\tilde{H}$, the system can be solved.

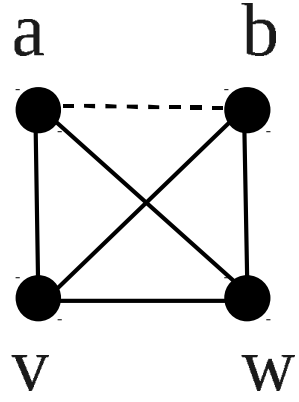
Recognizing diamond-free graphs

Algorithms:

$O(n^{\omega} + m^{\frac{3}{2}})$, Kloks, Kratsch and Muller (2000)

$O(m^{\frac{3}{2}})$, Eisenbrand and Grandoni (2004)

Note: By employing the method of counting subgraphs of size 4, we obtain an algorithm of complexity $O(\alpha^2 m)$



A simple algorithm

Theorem 4 *G is diamond-free iff $G[N[v]$ is a disjoint union of cliques, for every $v \in V(G)$*

Note: Constructing $G[N[v]]$ is one of the basic operations supported by the h-graph data structure.

We can recognize diamond-free graphs in $O(\alpha m)$ time.

Dynamic algorithm

1. Insertion of vertices
2. Removal of vertices
3. Insertion of edges
4. Removal of edges

Insertion of vertices

G, diamond-free; v new vertex, $N(v) \subseteq V(G)$

Is $G \cup \{v\}$ diamond-free ?

C, clique of G

* v is **edge-adjacent** to C: $E(C) \cap N'(v) \neq \emptyset$
* v is **fully edge-adjacent** C: $E(C) \cap N'(v) = E(C)$

Theorem 5 *$G \cup \{v\}$ is diamond-free iff the following holds for every maximal clique C to which v is edge-adjacent:*

1. *v is fully edge-adjacent to C, and*
2. *if v is edge-adjacent to a maximal clique $C' \neq C$, then $V(C') \cap V(C) = \emptyset$.*

Complexities

- Insertion of vertex v:
  $O(d(v)h(G))$
- Removal of vertex v:
  $O(d(v)h(G))$
- Insertion of edge vw:
  $O(d(v) + d(w))$
- Removal of edge vw:
  $O(min\{d(v) + d(w)\} + |H(v)| + |H(w)|)$

Simple, simplicial and dominated vertices

- v is **dominated** by w: $N[v] \subseteq N[w]$
- v, w are **comparable**: v is dominated by w, and w is dominated by v
- v is **simplicial**: v is dominated by all its neighbors
- v is **simple**: v is simplicial and its neighbors are pairwise comparable

Alg: dominated and simplicial vertices

v **dominated** by w iff $w \in N(v)$ and $d(v) - d(vw) = 1$

1. For each edge $vw \in E(G)$, compute $d(vw)$
2. For $v \in V(G)$
   $D(v) := \emptyset$ (set of vertices that dominate v)
   for $v \in N(v)$
   if $d(v) - d(vw) = 1$ then include w in $D(v)$
3. If $|D(v)| = d(v)$ then v is simplicial.

Complexity: $O(\alpha m)$

$O(m^{1.41})$, Kloks, Kratsch and Muller (2000)

Algorithm: simple vertices

For each simplicial vertex v
  for each edge $wz \in N'(v)$
    if $z \in D(w)$ and $w \in D(z)$ then v is simple

Complexity $O(\alpha m)$

Dynamic algorithms - Dominated vertices

D_G, family of dominated vertices of G
v, new vertex, $N(v) \subseteq V(G)$
$H = G \cup \{v\}$
Find D_H

Lemma 4 *A vertex $w \neq v$ is dominated in $H = G \cup \{v\}$ iff at least one of the following statements holds:*

- $w \notin N(v)$ *and* w *is dominated in* G
- $d_H(w) - d_H(vw) = 1$
- $d_H(w) - d_H(wz) = 1$, *for some* $wz \in N'_H(v)$

Complexity: $O(d(v)h(G))$

Simplicial vertices

Lemma 5 *A vertex $w \neq v$ is simplicial in $H = G \cup \{v\}$ if and only if one of the following statements is true:*

- $w \notin N(v)$ *and* w *is simplicial in* G*, or*
- w *is simplicial in* G*, and* $d(w) - d(vw) = 1$.

Complexity: $O(d(v)h(G))$

Simple vertices

Lemma 6 *Two vertices w and z of G are comparable in $G \cup \{v\}$ if and only if they are comparable in G and either $\{w, z\} \subseteq N(v)$ or $\{w, z\} \cap N(v) = \emptyset$.*

Complexity: $O(\alpha m)$

Recognizing cop-win graphs

Cop-win order of G:
$v_1, \ldots, v_n$, where v_i dominated in $G[\{v_i, \ldots, v_n\}]$,
$1 \leq i \leq n$

Cop-win graph: admits a cop-win order

Introduced by Poston (1971)
Also known as **dismantlable** graphs

Recognition: $O(min\{nm, n^3/\log n\})$ Spinrad (2004)

Dismantlings

A **dismantling** of a graph G is a graph H obtained by iteratively removing one dominated vertex, until no dominated vertices remain.

Observation 1 *: All dismantlings of a graph are isomorphic.*

Proposed algorithm

Direct application

1. find the set D of dominated vertices of G
2. while $D \neq \emptyset$,
   choose $v \in D$
   remove v from D
   update D
3. H, remaining graph
4. if $|V(H) = 1|$ then G is cop-win, otherwise it is not

Complexity: $O(\alpha m)$

Recognizing strongly chordal graphs

Simple elimination order of G:
$v_1, \ldots, v_n$, where v_i is a simple vertex in $G[\{v_i, \ldots, v_n\}]$, $1 \leq i \leq n$

Strongly chordal graphs: admits a simple elimination order

Introduced by Farber (1983)
Recognition algorithms: $O(n^2)$, $O(m \log n)$
Paige & Tarjan (1987), Spinrad (1993), Spinrad (2003)

Observation 2 *Every strongly chordal graph has a simple vertex v and $G \setminus \{v\}$ is strongly chordal*

Proposed algorithm

Direct application

1. find the set Q of simple vertices of G
2. while $Q \neq \emptyset$
   choose $v \in Q$
   remove v from G
   update Q
3. F, remaining graph
4. if F has no vertices then G is strongly chordal, otherwise it is not.

Complexity: $O(\alpha m)$

Summary - dynamic algorithms

Problem	**Inserting** v	**Removing** v
Listing all cliques of size k	$O(kdh\alpha^{k-3})$	-
Recognizing diamond-free graphs	$O(dh)$	$O(dh)$
Finding dominated vertices	$O(dh)$	$O(dh)$
Finding simplicial vertices	$O(dh)$	$O(dh)$
Finding simple vertices	$O(m)$	$O(m)$

Summary - static Algorithms

- Listing the cliques of size k

  $O(k\alpha^{k-2}m)$, Chiba & Nishizeki (1985)

  Proposed: $O(k\alpha^{k-2}m)$

- Counting subgraphs size 4:

  $O(n^{\omega} + min\{m^{1.61}, \alpha^2 m\})$, Kloks Kratsch & Muller(2000)

  Proposed: $O(min\{m^{1.61}, \alpha^2 m\})$

- Recognizing diamond-free graphs

  $O(m^{3/2})$, Eisenbrand & Grandoni (2004)

  Proposed: $O(\alpha m)$

Summary - static Algorithms

- Finding simple, simplicial and dominated vertices
  $O(m^{1.41})$ Kloks, Kratsch & Muller(2000)
  Proposed: $O(\alpha m)$
- Recognizing cop-win graphs
  $O(min\{nm, n^3/\log n\})$ Spinrad (2004)
  Proposed: $O(\alpha m)$
- Recognizing strongly chordal graphs
  $O(min\{n^2, m \log n\})$, Paige & Tarjan (1987), Spinrad (1993)
  Proposed: $O(\alpha m)$

More applications

$\mathcal{C}$, class of graphs
G **locally** $\mathcal{C}$: all vertex neighborhoods induce graphs belonging to $\mathcal{C}$

Theorem 6 *Let $\mathcal{C}$ be a class of graphs which can be recognized in $O(m)$ time. Then locally $\mathcal{C}$ graphs can be recognized in $O(\alpha m)$ time*

Example: diamond-free graphs $\cong$ locally "disjoint union of cliques" graphs

More applications

- It is possible to recognize gem-free graphs in $O(\alpha m)$ time
  Note: gem-free graphs $\cong$ locally cographs
- It is possible to recognize wheel-free graphs in $O(\alpha m)$ time
  Note: wheel-free graphs $\cong$ locally forests
- It is possible to recognize locally chordal graphs in $O(\alpha m)$ time

More applications

- The h-graph data structure can be extended to handle directed graphs
- A transitive orientation is a property that can be formulated in terms of neighborhoods, suited to be handled by h-graph data structures
- It would then be possible to recognize transitive orientations in $O(\alpha m)$ time
- Comparability graphs could then be recognized in $O(\alpha m)$ time.

Questions

- More operations ?
- Faster operations ?
- More applications ?

The End

THANK YOU

./