Input-Sensitive Enumerations

Petr Golovach

Department of Informatics, University of Bergen

SGT 2018, Sète, France, 14.06.2018

Plan of the lectures

- Introduction to branching enumeration algorithms and their analysis.
- Advanced analysis of branching algorithms; the "Measure and Conquer" technique.
- Lower bounds.
- Conclusions and open problems.

Plan of the lectures

- Introduction to branching enumeration algorithms and their analysis.
- Advanced analysis of branching algorithms; the "Measure and Conquer" technique.
- Lower bounds.
- Conclusions and open problems.

Branching algorithms

The majority of input-sensitive enumeration algorithms are *Recursive branching algorithms*.

Branching algorithms

The majority of input-sensitive enumeration algorithms are *Recursive branching algorithms*.

Pros

- Branching algorithms could be simple and easy to implement.
- Typically, they use polynomial space.
- Could be efficient in practice.

Branching algorithms

The majority of input-sensitive enumeration algorithms are *Recursive branching algorithms*.

Pros

- Branching algorithms could be simple and easy to implement.
- Typically, they use polynomial space.
- Could be efficient in practice.

Cons

- Difficult to analyze.
- Could be difficult to apply for some classes of problems.

Branching algorithms

Branching algorithms are recursively applied to (specially tailored) instances of a problem using *reduction* and *branching* rules.

Branching algorithms

Branching algorithms are recursively applied to (specially tailored) instances of a problem using *reduction* and *branching* rules.

- *Reduction rules*
  - used to simplify instances,
  - typically reduce the size,
  - typically run in polynomial time.

Branching algorithms

Branching algorithms are recursively applied to (specially tailored) instances of a problem using *reduction* and *branching* rules.

- *Reduction rules*
  - used to simplify instances,
  - typically reduce the size,
  - typically run in polynomial time.
- *Branching rules*
  - solve the problem for an instance by recursively solving $t \geq 2$ (smaller) instances,
  - typically run in polynomial time (without recursive calls).

Search trees

Search trees are used to illustrate, understand and analyze branching algorithms:

- *Root*: assign the input to the root.
- *Node*: assign to each node a problem instance.
- *Child*: each instance produced by a branching rule is assigned to a child.
- *Leaf*: outputs are assigned to leaves.

Analysis of branching algorithms

Running time analysis:

- upper bound of the maximum number of leaves $L(s)$ of a search tree for an input of given size s,
- if each reduction and branching rule can be done in polynomial time, then we have running time $O^*(L(s))$,
- $L(s)$ gives a combinatorial upper bound for the number of enumerating objects.

Bounding the number of leaves

Let $\mathcal{A}$ be a recursive branching algorithm that uses a single branching rule that generates $r \geq 2$ instances of the problem.

We associate a **measure** $\mu(I)$ with each instance I of the problem.

Typically, for simple branching algorithms, $\mu(I)$ is integer, and often $\mu(I)$ is the number of vertices for graph problems.

Let $I_1, \ldots, I_r$ be the instances of the problem generated by the branching rule from I.

Assume that there are positive (integers) $t_1, \ldots, t_r$ such that

$$\mu(I_i) \leq \mu(I) - t_i \text{ for } i \in \{1, \ldots, r\}.$$

It is said that

$$b = (t_1, \ldots, t_r)$$

is the *branching vector* for the rule.

Bounding the number of leaves

Let $L(s)$ be the maximum number of leaves of a search tree for the instances I with $\mu(I) = s$.

We have that

$$L(s) \leq L(s - t_1) + \ldots + L(s - t_r).$$

Let $t = \max\{t_1, \ldots, t_r\}$. We associate with $b = (t_1, \ldots, t_r)$ the *characteristic polynomial*:

$$p(x) = x^t - x^{t-t_1} - \ldots - x^{t-t_r}.$$

Claim: $p(x)$ has a unique positive real root λ and $L(s) = O^*(\lambda^s)$.

It is said that $\lambda = \lambda(t_1, \ldots, t_r)$ is the *branching number* of b.

Bounding the number of leaves

Given a *branching vector*

$$b = (t_1, \ldots, t_r),$$

we solve the equation

$$x^t - x^{t-t_1} - \ldots - x^{t-t_r} = 0$$

for $t = \max\{t_1, \ldots, t_r\}$ and find the *branching number*

$$\lambda = \lambda(t_1, \ldots, t_r).$$

Then we obtain the bound

$$L(s) = O^*(\lambda^s).$$

Often we can show that $L(s) \leq \lambda^s$.

Analyzing collections of recurrences

We consider all branching rules and construct the family of branching vectors

$$\mathcal{B} = \{b^{(i)} \mid i \in \mathbb{N}\}$$

$$b^{(i)} = (t_1^{(i)}, \ldots, t_{r(i)}^{(i)}).$$

We compute

$$\lambda = \sup\{\lambda(b^{(i)}) \mid b^{(i)} \in \mathcal{B}\}.$$

Claim:

$$L(s) = O^*(\lambda^s).$$

Often it could be shown that $L(s) \leq \lambda^s$.

Analyzing collections of recurrences

The family of branching vectors $\mathcal{B}$ could be infinite.

Analyzing collections of recurrences

The family of branching vectors $\mathcal{B}$ could be infinite.

Nevertheless, it is usually sufficient to consider few "worst" branching vectors and find

$$\lambda = \max\{\lambda(b^{(i)}) \mid b^{(i)} \in \mathcal{B}\}.$$

Analyzing collections of recurrences

The family of branching vectors $\mathcal{B}$ could be infinite.

Nevertheless, it is usually sufficient to consider few "worst" branching vectors and find

$$\lambda = \max\{\lambda(b^{(i)}) \mid b^{(i)} \in \mathcal{B}\}.$$

Huge problem: The bound $O^*(\lambda^s)$ is limited by the worst branching number even if the corresponding branching occur in some few special cases.

Measure & Conquer

Fedor V. Fomin, Fabrizio Grandoni, Dieter Kratsch: A measure & conquer approach for the analysis of exact algorithms. J. ACM 56(5): 25:1-25:32 (2009)

Measure & Conquer

Fedor V. Fomin, Fabrizio Grandoni, Dieter Kratsch: A measure & conquer approach for the analysis of exact algorithms. J. ACM 56(5): 25:1-25:32 (2009)

Nerode Prize 2017:

Measure & Conquer

Fedor V. Fomin, Fabrizio Grandoni, Dieter Kratsch: A measure & conquer approach for the analysis of exact algorithms. J. ACM 56(5): 25:1-25:32 (2009)

Nerode Prize 2017:

Measure & Conquer

Fedor V. Fomin, Fabrizio Grandoni, Dieter Kratsch: A measure & conquer approach for the analysis of exact algorithms. J. ACM 56(5): 25:1-25:32 (2009)

Nerode Prize 2017:

Bounding the number of leaves

Let $\mathcal{A}$ be a recursive branching algorithm that uses a single branching rule that generates $r \geq 2$ instances of the problem.

We associate a **measure** $\mu(I)$ with each instance I of the problem.

Typically, for simple branching algorithms, $\mu(I)$ is integer, and often $\mu(I)$ is the number of vertices for graph problems.

Let $I_1, \ldots, I_r$ be the instances of the problem generated by the branching rule from I.

For a branching vector $b = (t_1, \ldots, t_r)$, we have that

$$\mu(I_i) \leq \mu(I) - t_i \text{ for } i \in \{1, \ldots, r\}.$$

For $\lambda = \lambda(t_1, \ldots, t_r)$ and $s = \mu(I)$,

$$L(s) = O^*(\lambda^s).$$

Bounding the number of leaves

Observe that to obtain the bound $L(s) = O^*(\lambda^s)$, we have not made any assumption about $\mu(I)$.

Bounding the number of leaves

Observe that to obtain the bound $L(s) = O^*(\lambda^s)$, we have not made any assumption about $\mu(I)$.

We can chose $\mu(I)$ to improve the running time analysis.

Bounding the number of leaves

Observe that to obtain the bound $L(s) = O^*(\lambda^s)$, we have not made any assumption about $\mu(I)$.

We can chose $\mu(I)$ to improve the running time analysis.

The measure $\mu(I)$ not necessarily should be integer.

Bounding the number of leaves

Observe that to obtain the bound $L(s) = O^*(\lambda^s)$, we have not made any assumption about $\mu(I)$.

We can chose $\mu(I)$ to improve the running time analysis.

The measure $\mu(I)$ not necessarily should be integer.

Assume that we consider an enumeration problem for graphs where the aim is to list all vertex subsets that satisfy a property P.

Claim: If $\mu(I) \leq n$ for the inputs containing n-vertex graphs, then

$$L(n) = O^*(\lambda^n).$$

Measure for graph problems

Assume that we consider an enumeration problem for graphs where the aim is to list all vertex subsets that satisfy a property P.

Measure for graph problems

Assume that we consider an enumeration problem for graphs where the aim is to list all vertex subsets that satisfy a property P.

We construct a weight function:

$$\omega\colon V(G) \to \mathbb{R}_{\geq 0}.$$

Measure for graph problems

Assume that we consider an enumeration problem for graphs where the aim is to list all vertex subsets that satisfy a property P.

We construct a weight function:

$$\omega\colon V(G) \to \mathbb{R}_{\geq 0}.$$

We set

$$\mu(G) = \sum_{v \in V(G)} \omega(v).$$

Measure for graph problems

Assume that we consider an enumeration problem for graphs where the aim is to list all vertex subsets that satisfy a property P.

We construct a weight function:

$$\omega\colon V(G) \to \mathbb{R}_{\geq 0}.$$

We set

$$\mu(G) = \sum_{v \in V(G)} \omega(v).$$

If $\omega(v) \leq 1$, then $\mu(G) \leq n$.

Measure for graph problems

Assume that we consider an enumeration problem for graphs where the aim is to list all vertex subsets that satisfy a property P.

We construct a weight function:

$$\omega\colon V(G) \to \mathbb{R}_{\geq 0}.$$

We set

$$\mu(G) = \sum_{v \in V(G)} \omega(v).$$

If $\omega(v) \leq 1$, then $\mu(G) \leq n$.

This is what we need to obtain that for n-vertex graphs,

$$L(n) = O^*(\lambda^n).$$

Minimal connected dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

Minimal connected dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A set D is a *connected dominating set* if

- D is a dominating set and
- $G[D]$ is connected.

Minimal connected dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A set D is a *connected dominating set* if

- D is a dominating set and
- $G[D]$ is connected.

A connected dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.

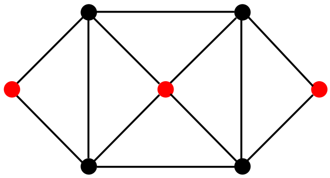
Minimal connected dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A set D is a *connected dominating set* if

- D is a dominating set and
- $G[D]$ is connected.

A connected dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.



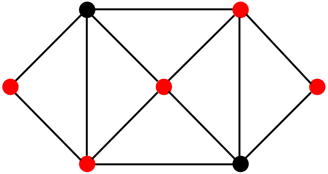
Minimal connected dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A set D is a *connected dominating set* if

- D is a dominating set and
- $G[D]$ is connected.

A connected dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.



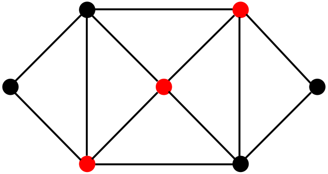
Minimal connected dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A set D is a *connected dominating set* if

- D is a dominating set and
- $G[D]$ is connected.

A connected dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.



Minimal connected dominating sets

Problem (Minimal CDS enumeration)

Input: *A graph G.*

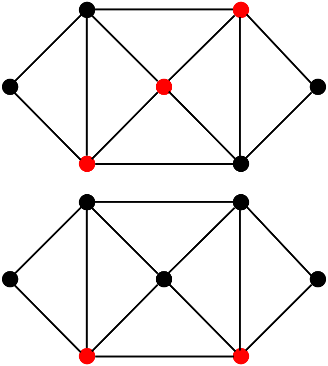
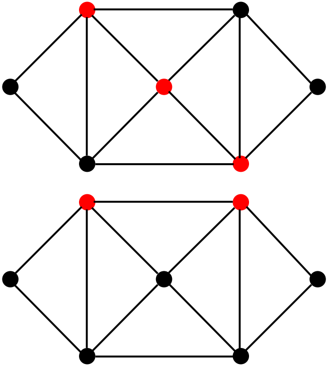
Task: *Enumerate all minimal connected dominating sets of G.*

Minimal connected dominating sets

Problem (Minimal CDS enumeration)

Input: *A graph G.*

Task: *Enumerate all minimal connected dominating sets of G.*

Minimal connected dominating sets

Problem (Minimal CDS enumeration)

Input: *A graph G.*

Task: *Enumerate all minimal connected dominating sets of G.*

Daniel Lokshtanov, Michal Pilipczuk, Saket Saurabh: Below all subsets for Minimal Connected Dominating Set. CoRR abs/1611.00840 (2016)

Minimal connected dominating sets

Problem (Minimal CDS enumeration)

Input: *A graph G.*

Task: *Enumerate all minimal connected dominating sets of G.*

Daniel Lokshtanov, Michal Pilipczuk, Saket Saurabh: Below all subsets for Minimal Connected Dominating Set. CoRR abs/1611.00840 (2016)

Theorem

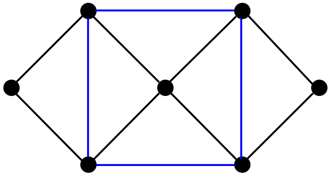
There is $\varepsilon > 0$ such that all minimal connected dominating sets of an n-vertex graph can be enumerated in time $2^{(1-\varepsilon)n}$.

Chordal graphs

A graph G is *chordal* if it has no induced cycle with at least 4 vertices.

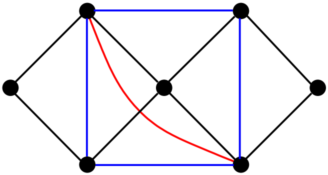
Chordal graphs

A graph G is *chordal* if it has no induced cycle with at least 4 vertices.



Chordal graphs

A graph G is *chordal* if it has no induced cycle with at least 4 vertices.

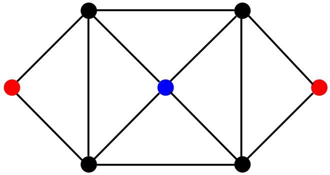


Simplicial vertices

A vertex v of a graph G is *simplicial* if $N_G(v)$ is a clique, that is, the neighbors of v are pairwise adjacent.

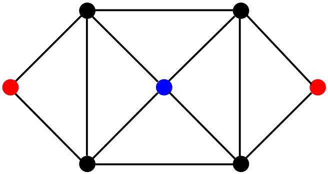
Simplicial vertices

A vertex v of a graph G is *simplicial* if $N_G(v)$ is a clique, that is, the neighbors of v are pairwise adjacent.



Simplicial vertices

A vertex v of a graph G is *simplicial* if $N_G(v)$ is a clique, that is, the neighbors of v are pairwise adjacent.



Lemma

Every chordal graph has a simplicial vertex.

Semi-simplicial vertices

For a vertex u of a graph G, denote by $S_G(u)$ the set of all simplicial vertices v of G such that $vu \in E(G)$.

Semi-simplicial vertices

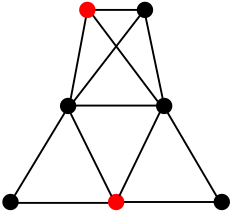
For a vertex u of a graph G, denote by $S_G(u)$ the set of all simplicial vertices v of G such that $vu \in E(G)$.

A vertex u is *semi-simplicial* if $S_G(u) \neq \emptyset$ and u is a simplicial vertex of $G - S_G(u)$.

Semi-simplicial vertices

For a vertex u of a graph G, denote by $S_G(u)$ the set of all simplicial vertices v of G such that $vu \in E(G)$.

A vertex u is *semi-simplicial* if $S_G(u) \neq \emptyset$ and u is a simplicial vertex of $G - S_G(u)$.



Semi-simplicial vertices

Lemma (Heggernes and Abu-Khzam, 2016)

Every connected chordal graph with at least two vertices has a semi-simplicial vertex.

Semi-simplicial vertices

Lemma (Heggernes and Abu-Khzam, 2016)

Every connected chordal graph with at least two vertices has a semi-simplicial vertex.

- If G is a complete graph with at least two vertices, then every vertex is semi-simplicial.

Semi-simplicial vertices

Lemma (Heggernes and Abu-Khzam, 2016)

Every connected chordal graph with at least two vertices has a semi-simplicial vertex.

- If G is a complete graph with at least two vertices, then every vertex is semi-simplicial.
- Suppose that G is not complete. Denote by S the set of all simplicial vertices of G.

Semi-simplicial vertices

Lemma (Heggernes and Abu-Khzam, 2016)

Every connected chordal graph with at least two vertices has a semi-simplicial vertex.

- If G is a complete graph with at least two vertices, then every vertex is semi-simplicial.
- Suppose that G is not complete. Denote by S the set of all simplicial vertices of G.
- Then $G' = G - S$ is not empty and has a simplicial vertex u.

Semi-simplicial vertices

Lemma (Heggernes and Abu-Khzam, 2016)

Every connected chordal graph with at least two vertices has a semi-simplicial vertex.

- If G is a complete graph with at least two vertices, then every vertex is semi-simplicial.
- Suppose that G is not complete. Denote by S the set of all simplicial vertices of G.
- Then $G' = G - S$ is not empty and has a simplicial vertex u.
- We have that u is a semi-simplicial vertex of G.

Contractions

Let D be a minimal connected dominating set of G.

Contractions

Let D be a minimal connected dominating set of G.

Suppose that $x, y \in D$ are adjacent.

Contractions

Let D be a minimal connected dominating set of G.

Suppose that $x, y \in D$ are adjacent.

Let $G' = G/xy$ and v_{xy} is the vertex of G' obtained from x and y.

Contractions

Let D be a minimal connected dominating set of G.

Suppose that $x, y \in D$ are adjacent.

Let $G' = G/xy$ and v_{xy} is the vertex of G' obtained from x and y.

Let $D' = (D \setminus \{x, y\}) \cup \{v_{xy}\}$.

Contractions

Let D be a minimal connected dominating set of G.

Suppose that $x, y \in D$ are adjacent.

Let $G' = G/xy$ and v_{xy} is the vertex of G' obtained from x and y.

Let $D' = (D \setminus \{x, y\}) \cup \{v_{xy}\}$.

We write that $D' = D/xy$

Contractions

Let D be a minimal connected dominating set of G.

Suppose that $x, y \in D$ are adjacent.

Let $G' = G/xy$ and v_{xy} is the vertex of G' obtained from x and y.

Let $D' = (D \setminus \{x, y\}) \cup \{v_{xy}\}$.

We write that $D' = D/xy$

Claim: D' is a minimal connected dominating set of G'.

Sketch of the proof

- We have that D' is a connected dominating set.

Sketch of the proof

- We have that D' is a connected dominating set.
- Assume that D' is not minimal. Then there is $u \in D'$ such that
  - $D' \setminus \{u\}$ is connected,
  - $D' \setminus \{u\}$ is a dominating set of G'.

Sketch of the proof

- We have that D' is a connected dominating set.
- Assume that D' is not minimal. Then there is $u \in D'$ such that
  - $D' \setminus \{u\}$ is connected,
  - $D' \setminus \{u\}$ is a dominating set of G'.
- If $u \neq v_{xy}$, then $D \setminus \{u\}$ is a connected dominating set of G.

Sketch of the proof

- We have that D' is a connected dominating set.
- Assume that D' is not minimal. Then there is $u \in D'$ such that
  - $D' \setminus \{u\}$ is connected,
  - $D' \setminus \{u\}$ is a dominating set of G'.
- If $u \neq v_{xy}$, then $D \setminus \{u\}$ is a connected dominating set of G.
- If $u = v_{xy}$, then either x or y is not a cut-vertex of $G[D]$.

Sketch of the proof

- We have that D' is a connected dominating set.
- Assume that D' is not minimal. Then there is $u \in D'$ such that
  - $D' \setminus \{u\}$ is connected,
  - $D' \setminus \{u\}$ is a dominating set of G'.
- If $u \neq v_{xy}$, then $D \setminus \{u\}$ is a connected dominating set of G.
- If $u = v_{xy}$, then either x or y is not a cut-vertex of $G[D]$.
- If x is not a cut-vertex of $G[D]$, then $D \setminus \{x\}$ is a connected dominating set of G.

Separators

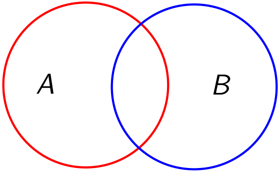
Observation: Let (A, B) be a *separation* of a connected graph G, that is, $A, B \subseteq V(G)$, $A \cup B = V(G)$ and there is no edge $uv \in E(G)$ with $u \in A \setminus B$ and $v \in B \setminus A$.

Separators

Observation: Let (A, B) be a *separation* of a connected graph G, that is, $A, B \subseteq V(G)$, $A \cup B = V(G)$ and there is no edge $uv \in E(G)$ with $u \in A \setminus B$ and $v \in B \setminus A$. If $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$, then $D \cap (A \cap B) \neq \emptyset$ for every minimal connected dominating set D.

Separators

Observation: Let (A, B) be a *separation* of a connected graph G, that is, $A, B \subseteq V(G)$, $A \cup B = V(G)$ and there is no edge $uv \in E(G)$ with $u \in A \setminus B$ and $v \in B \setminus A$. If $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$, then $D \cap (A \cap B) \neq \emptyset$ for every minimal connected dominating set D.



Enumeration of minimal CDS

Let G be a connected chordal graph.

Enumeration of minimal CDS

Let G be a connected chordal graph.

Let H is an *induced minor* of G (i.e., H is obtained by vertex deletions and edge contractions).

Enumeration of minimal CDS

Let G be a connected chordal graph.

Let H is an *induced minor* of G (i.e., H is obtained by vertex deletions and edge contractions).

For $X \subseteq V(H)$, $exp(X)$ denotes the set of vertices of G that are contracted to the vertices of X.

Enumeration of minimal CDS

Let G be a connected chordal graph.

Let H is an *induced minor* of G (i.e., H is obtained by vertex deletions and edge contractions).

For $X \subseteq V(H)$, $exp(X)$ denotes the set of vertices of G that are contracted to the vertices of X.

For $X \subseteq V(H)$, D is an X-minimal connected dominating set of H if

Enumeration of minimal CDS

Let G be a connected chordal graph.

Let H is an *induced minor* of G (i.e., H is obtained by vertex deletions and edge contractions).

For $X \subseteq V(H)$, $exp(X)$ denotes the set of vertices of G that are contracted to the vertices of X.

For $X \subseteq V(H)$, D is an X-minimal connected dominating set of H if

(i) $X \subseteq D$,

Enumeration of minimal CDS

Let G be a connected chordal graph.

Let H is an *induced minor* of G (i.e., H is obtained by vertex deletions and edge contractions).

For $X \subseteq V(H)$, $exp(X)$ denotes the set of vertices of G that are contracted to the vertices of X.

For $X \subseteq V(H)$, D is an X-minimal connected dominating set of H if

(i) $X \subseteq D$,

(ii) D is a connected dominating set, and

Enumeration of minimal CDS

Let G be a connected chordal graph.

Let H is an *induced minor* of G (i.e., H is obtained by vertex deletions and edge contractions).

For $X \subseteq V(H)$, $exp(X)$ denotes the set of vertices of G that are contracted to the vertices of X.

For $X \subseteq V(H)$, D is an X-minimal connected dominating set of H if

(i) $X \subseteq D$,

(ii) D is a connected dominating set, and

(ii) D is (inclusion) minimal subject to (i) and (ii).

Enumeration of minimal CDS

We construct the algorithm **Emum MCDS**(H, X).

Enumeration of minimal CDS

We construct the algorithm **Emum MCDS**(H, X).

Input: A graph H that is an induced minor of G and $X \subseteq V(H)$.

Enumeration of minimal CDS

We construct the algorithm **Emum MCDS**(H, X).

Input: A graph H that is an induced minor of G and $X \subseteq V(H)$.

Output: X-Minimal CDS D of H such that $exp(D)$ is a minimal CDS of G.

Enumeration of minimal CDS

We construct the algorithm **Emum MCDS**(H, X).

Input: A graph H that is an induced minor of G and $X \subseteq V(H)$.

Output: X-Minimal CDS D of H such that $exp(D)$ is a minimal CDS of G.

We generate all X-minimal CDS D of H, and for each D, we test whether $\exp(D)$ is minimal CDS of G.

Enumeration of minimal CDS

We construct the algorithm **Emum MCDS**(H, X).

Input: A graph H that is an induced minor of G and $X \subseteq V(H)$.

Output: X-Minimal CDS D of H such that $exp(D)$ is a minimal CDS of G.

We generate all X-minimal CDS D of H, and for each D, we test whether $\exp(D)$ is minimal CDS of G.

To enumerate minimal connected dominating sets of G, we call **Emum MCDS**$(G, \emptyset)$.

Enumeration of minimal CDS

We construct the algorithm **Emum MCDS**(H, X).

Input: A graph H that is an induced minor of G and $X \subseteq V(H)$.

Output: X-Minimal CDS D of H such that $exp(D)$ is a minimal CDS of G.

We generate all X-minimal CDS D of H, and for each D, we test whether $\exp(D)$ is minimal CDS of G.

To enumerate minimal connected dominating sets of G, we call **Emum MCDS**$(G, \emptyset)$.

The measure of the instance is $|V(H)|$.

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.

Note that the last step could be seen as a branching rule:

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.

Note that the last step could be seen as a branching rule: we branch on $k = |V(H)|$ instances $(H - N_H(v), \{v\})$.

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.

Note that the last step could be seen as a branching rule: we branch on $k = |V(H)|$ instances $(H - N_H(v), \{v\})$.

The branching vector is

$$\underbrace{(k, \ldots, k)}_{k}$$

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.

Note that the last step could be seen as a branching rule: we branch on $k = |V(H)|$ instances $(H - N_H(v), \{v\})$.

The branching vector is

$$(\underbrace{k, \ldots, k}_{k})$$

and the maximum branching number is $\lambda(3,3,3) = 3^{1/3} \approx 1.4423$ for $k = 3$.

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.

Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.
- If there are adjacent $x, y \in X$, then call **Emum MCDS**$(H/xy, X/xy)$.

Initial steps

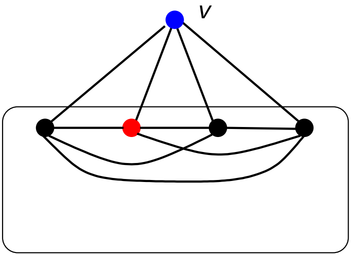
- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.
- If there are adjacent $x, y \in X$, then call **Emum MCDS**$(H/xy, X/xy)$.
- If H has a cut-vertex $v \notin X$, then call **Emum MCDS**$(H, X \cup \{v\})$.

Initial steps

- If H has a simplicial vertex v such that $N_H(v) \cap X \neq \emptyset$, then call **Emum MCDS**$(H - v, X)$.

Initial steps

- If H has a simplicial vertex v such that $N_H(v) \cap X \neq \emptyset$, then call **Emum MCDS**$(H - v, X)$.

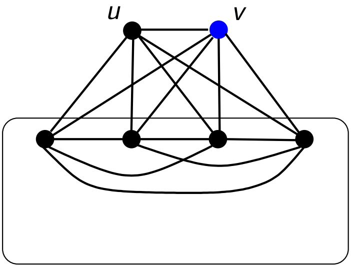


Initial steps

- If H has adjacent simplicial vertices u and v, then call **Emum MCDS**$(H - v, X)$.

Initial steps

- If H has adjacent simplicial vertices u and v, then call **Emum MCDS**$(H - v, X)$.



Initial steps

- If X is a minimal CDS of H, then check whether $exp(X)$ is a minimal CDS of G and output X if this holds.
- If $X = \emptyset$ and H is a complete graph, then consider all $D = \{v\}$ for $v \in V(H)$.
- If there are adjacent $x, y \in X$, then call **Emum MCDS**$(H/xy, X/xy)$.
- If H has a cut-vertex $v \notin X$, then call **Emum MCDS**$(H, X \cup \{v\})$.
- If H has a simplicial vertex v such that $N_H(v) \cap X \neq \emptyset$, then call **Emum MCDS**$(H - v, X)$.
- If H has adjacent simplicial vertices u and v, then call **Emum MCDS**$(H - v, X)$.

Branching

If the initial steps cannot be applied, then

Branching

If the initial steps cannot be applied, then

- H is not complete,

Branching

If the initial steps cannot be applied, then

- H is not complete,
- simplicial vertices of H are pairwise non-adjacent,

Branching

If the initial steps cannot be applied, then

- H is not complete,
- simplicial vertices of H are pairwise non-adjacent,
- the minimum degree of H is at least 2.

Branching

If the initial steps cannot be applied, then

- H is not complete,
- simplicial vertices of H are pairwise non-adjacent,
- the minimum degree of H is at least 2.

We select a semi-simplicial vertex x and branch depending on $|S_H(x)|$:

- $|S_H(x)| = 1$,
- $|S_H(x)| = 2$,
- $|S_H(x)| \geq 3$.

Branching for $|S_H(x)| = 1$

Let y be the unique vertex of $S_H(x)$.

Branching for $|S_H(x)| = 1$

Let y be the unique vertex of $S_H(x)$.

We consider two cases:

- $d_H(y) = 2$,
- $d_H(y) \geq 3$.

Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

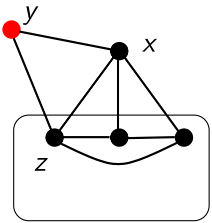
Let z be the neighbor of y distinct from x.

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H/xy - z, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H/yz - x, (X \cup \{z\})/yz)$.

Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

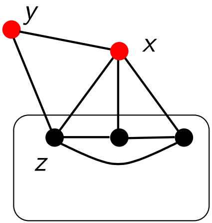
- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H/xy - z, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H/yz - x, (X \cup \{z\})/yz)$.



Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

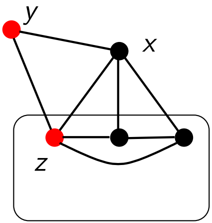
- If $y \in X$, then
  (i) call **Emum MCDS**$(H/xy - z, (X \cup \{x\})/xy)$,
  (ii) call **Emum MCDS**$(H/yz - x, (X \cup \{z\})/yz)$.



Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

- If $y \in X$, then
  (i) call **Emum MCDS**$(H/xy - z, (X \cup \{x\})/xy)$,
  (ii) call **Emum MCDS**$(H/yz - x, (X \cup \{z\})/yz)$.



Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

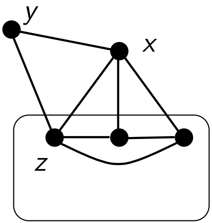
Let z be the neighbor of y distinct from x.

- If $y \notin X$, then
  (i) call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,
  (ii) call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.

Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

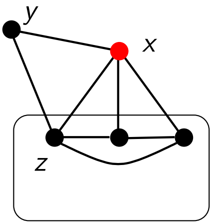
- If $y \notin X$, then
  (i) call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,
  (ii) call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.



Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

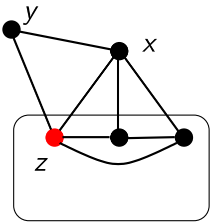
- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.



Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.



Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H/xy - z, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H/yz - x, (X \cup \{z\})/yz)$.
- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.

Branching for $|S_H(x)| = 1$ and $d_H(y) = 2$

Let z be the neighbor of y distinct from x.

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H/xy - z, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H/yz - x, (X \cup \{z\})/yz)$.
- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.

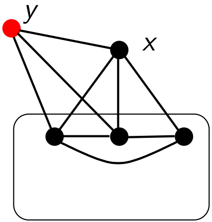
The branching vector is $(2, 2)$ and $\lambda(2, 2) = 2^{1/2} \approx 1.4143$.

Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H(y) \setminus \{x\})/xy, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.

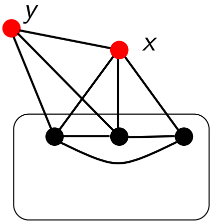
Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H(y) \setminus \{x\})/xy, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.



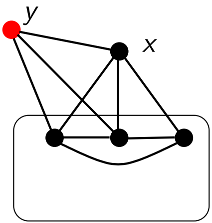
Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H(y) \setminus \{x\})/xy, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.



Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H(y) \setminus \{x\})/xy, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.

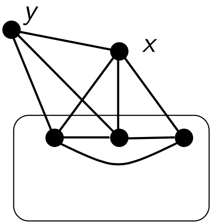


Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}), X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.

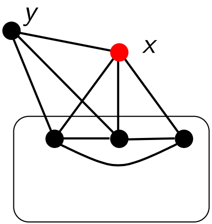
Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}), X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.



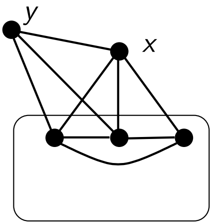
Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}), X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.



Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \notin X$, then
  (i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}), X \cup \{x\})$,
  (ii) call **Emum MCDS**$(H - x, X)$.



Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \in X$, then
  (i) call **Emum MCDS**$(H - (N_H(y) \setminus \{x\})/xy, (X \cup \{x\})/xy)$,
  (ii) call **Emum MCDS**$(H - x, X)$.
- If $y \notin X$, then
  (i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}), X \cup \{x\})$,
  (ii) call **Emum MCDS**$(H - x, X)$.

Branching for $|S_H(x)| = 1$ and $d_H(y) \geq 3$

- If $y \in X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H(y) \setminus \{x\})/xy, (X \cup \{x\})/xy)$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.
- If $y \notin X$, then
  - **(i)** call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}), X \cup \{x\})$,
  - **(ii)** call **Emum MCDS**$(H - x, X)$.

The worst branching vector is $(3, 1)$ for $d_H(y) = 3$ and $\lambda(3, 1) \approx 1.4656$.

Branching for $|S_H(x)| = 2$

Let y and z be the neighbors of x, $d_H(y) \leq d_H(z)$.

Branching for $|S_H(x)| = 2$

Let y and z be the neighbors of x, $d_H(y) \leq d_H(z)$.

Observation: if for an X-minimal connected dominating set D, it holds that $x \in D \setminus X$, then

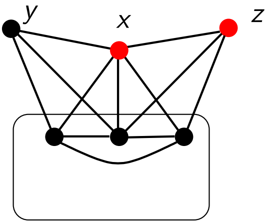
- either $(N_H(y) \setminus \{x\}) \cap D = \emptyset$
- or $(N_H(z) \setminus \{x\}) \cap D = \emptyset$.

Branching for $|S_H(x)| = 2$

Let y and z be the neighbors of x, $d_H(y) \leq d_H(z)$.

Observation: if for an X-minimal connected dominating set D, it holds that $x \in D \setminus X$, then

- either $(N_H(y) \setminus \{x\}) \cap D = \emptyset$
- or $(N_H(z) \setminus \{x\}) \cap D = \emptyset$.

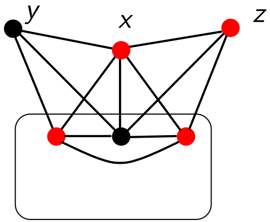


Branching for $|S_H(x)| = 2$

Let y and z be the neighbors of x, $d_H(y) \leq d_H(z)$.

Observation: if for an X-minimal connected dominating set D, it holds that $x \in D \setminus X$, then

- either $(N_H(y) \setminus \{x\}) \cap D = \emptyset$
- or $(N_H(z) \setminus \{x\}) \cap D = \emptyset$.



Branching for $|S_H(x)| = 2$

We consider the cases:

- $d_H(y) = d_H(z) = 2$,
- $d_H(y) = 2$ and $d_H(z) \geq 3$,
- $d_H(y), d_H(z) \geq 3$,

Branching for $|S_H(x)| = 2$

We consider the cases:

- $d_H(y) = d_H(z) = 2$,
- $d_H(y) = 2$ and $d_H(z) \geq 3$,
- $d_H(y), d_H(z) \geq 3$,

and branch depending on the inclusion of y and z in X.

Branching for $|S_H(x)| = 2$

Let $d_H(y), d_H(z) \geq 3$ and $y, z \notin X$.

Branching for $|S_H(x)| = 2$

Let $d_H(y), d_H(z) \geq 3$ and $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,
(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,
(iii) call **Emum MCDS**$(H - x, X)$.

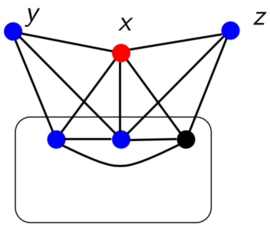
Branching for $|S_H(x)| = 2$

Let $d_H(y), d_H(z) \geq 3$ and $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,

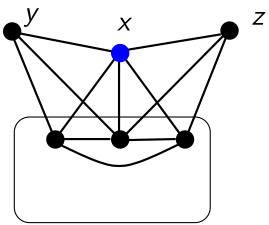
(iii) call **Emum MCDS**$(H - x, X)$.



Branching for $|S_H(x)| = 2$

Let $d_H(y), d_H(z) \geq 3$ and $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,
(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,
(iii) call **Emum MCDS**$(H - x, X)$.



Branching for $|S_H(x)| = 2$

Let $d_H(y), d_H(z) \geq 3$ and $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,
(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,
(iii) call **Emum MCDS**$(H - x, X)$.

Branching for $|S_H(x)| = 2$

Let $d_H(y), d_H(z) \geq 3$ and $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,
(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,
(iii) call **Emum MCDS**$(H - x, X)$.

The worst branching vector $(4, 4, 1)$ for $d_H(y) = d_H(z) = 3$ and $\lambda(4, 4, 1) \approx 1.5437$.

Branching for $|S_H(x)| \geq 3$

Assume that $S_H(x) \cap X = \emptyset$.

Branching for $|S_H(x)| \geq 3$

Assume that $S_H(x) \cap X = \emptyset$.

(i) call **Emum MCDS**$(H - S_H(x), X \cup \{x\})$,
(ii) call **Emum MCDS**$(H - x, X)$.

Branching for $|S_H(x)| \geq 3$

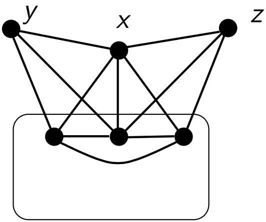
Assume that $S_H(x) \cap X = \emptyset$.

(i) call **Emum MCDS**$(H - S_H(x), X \cup \{x\})$,
(ii) call **Emum MCDS**$(H - x, X)$.

The worst branching vector is $(3, 1)$ for $|S_H(x)| = 3$ and $\lambda(3, 1) \approx 1.4656$.

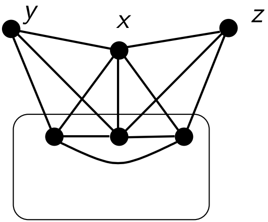
Introducing measure

We have that the worst case is $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$.



Introducing measure

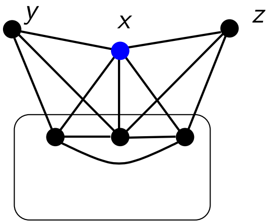
We have that the worst case is $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$.



The branching vector is $(4, 4, 1)$ and $\lambda(4, 4, 1) \approx 1.5437$.

Introducing measure

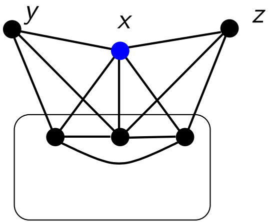
We have that the worst case is $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$.



If we delete x, then the vertices y and z get degree 2 in the obtained graph.

Introducing measure

We have that the worst case is $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$.



If we delete x, then the vertices y and z get degree 2 in the obtained graph.

Simplicial verices of degree 2 are good!

Introducing measure

Let $0 < \varepsilon < 1$.

Introducing measure

Let $0 < \varepsilon < 1$.

We set

$$\omega(v) = \begin{cases} 1 - \varepsilon & \text{if } v \text{ is a simplicial vertex of degree 2} \\ 1 & \text{otherwise} \end{cases}$$

Improving branching numbers

Let $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$, and assume that $y, z \notin X$.

Improving branching numbers

Let $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$, and assume that $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,

(iii) call **Emum MCDS**$(H - x, X)$.

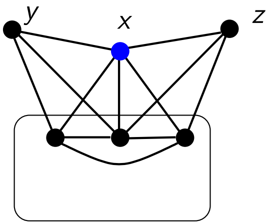
Improving branching numbers

Let $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$, and assume that $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,

(iii) call **Emum MCDS**$(H - x, X)$.



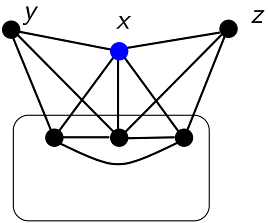
Improving branching numbers

Let $|S_H(x)| = 2$ and $d_H(y) = d_H(z) = 3$ for $\{y, z\} = S_H(x)$, and assume that $y, z \notin X$.

(i) call **Emum MCDS**$(H - (N_H[y] \setminus \{x\}) \cup \{z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - (N_H[z] \setminus \{x\}) \cup \{y\}, X \cup \{x\})$,

(iii) call **Emum MCDS**$(H - x, X)$.



The branching vector is $(4, 4, 1 + 2\varepsilon)$ and $\lambda(4, 4, 1 + 2\varepsilon) < \lambda(4, 4, 1)$.

Some branching numbers grow

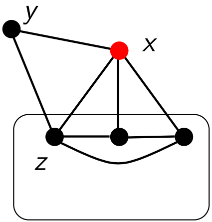
Let $|S_H(x)| = 1$ and $d_H(y) = 2$ for $\{y\} = S_H(x)$. Let $N_H(y) = \{y, z\}$ and assume that $y \notin X$.

Some branching numbers grow

Let $|S_H(x)| = 1$ and $d_H(y) = 2$ for $\{y\} = S_H(x)$. Let $N_H(y) = \{y, z\}$ and assume that $y \notin X$.

(i) call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.

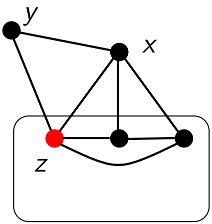


Some branching numbers grow

Let $|S_H(x)| = 1$ and $d_H(y) = 2$ for $\{y\} = S_H(x)$. Let $N_H(y) = \{y, z\}$ and assume that $y \notin X$.

(i) call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.

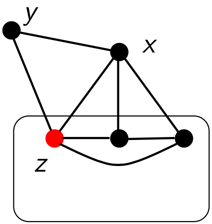


Some branching numbers grow

Let $|S_H(x)| = 1$ and $d_H(y) = 2$ for $\{y\} = S_H(x)$. Let $N_H(y) = \{y, z\}$ and assume that $y \notin X$.

(i) call **Emum MCDS**$(H - \{y, z\}, X \cup \{x\})$,

(ii) call **Emum MCDS**$(H - \{x, y\}, X \cup \{z\})$.



The branching vector is $(2 - \varepsilon, 2 - \varepsilon) > \lambda(2, 2)$.

Balancing branching numbers

Let $\lambda_i(\varepsilon)$ be the branching numbers considered as functions of ε.

Balancing branching numbers

Let $\lambda_i(\varepsilon)$ be the branching numbers considered as functions of ε.

We have to find

$$\lambda = \min_{\varepsilon} \max_{i} \lambda_i(\varepsilon)$$

Balancing branching numbers

Let $\lambda_i(\varepsilon)$ be the branching numbers considered as functions of ε.

We have to find

$$\lambda = \min_{\varepsilon} \max_{i} \lambda_i(\varepsilon)$$

We consider two families of branching numbers.

Balancing branching numbers

Let $\lambda_i(\varepsilon)$ be the branching numbers considered as functions of ε.

We have to find

$$\lambda = \min_{\varepsilon} \max_{i} \lambda_i(\varepsilon)$$

We consider two families of branching numbers.

$$\mathcal{B} = \{\lambda_i(\varepsilon) \mid \lambda_i(\varepsilon) \text{ is a decreasing function}\}$$

Balancing branching numbers

Let $\lambda_i(\varepsilon)$ be the branching numbers considered as functions of ε.

We have to find

$$\lambda = \min_{\varepsilon} \max_{i} \lambda_i(\varepsilon)$$

We consider two families of branching numbers.

$$\mathcal{B} = \{\lambda_i(\varepsilon) \mid \lambda_i(\varepsilon) \text{ is a decreasing function}\}$$

and

$$\mathcal{B}' = \{\lambda_i(\varepsilon) \mid \lambda_i(\varepsilon) \text{ is an increasing function}\}.$$

Balancing branching numbers

Let $\lambda_i(\varepsilon)$ be the branching numbers considered as functions of ε.

We have to find

$$\lambda = \min_{\varepsilon} \max_{i} \lambda_i(\varepsilon)$$

We consider two families of branching numbers.

$$\mathcal{B} = \{\lambda_i(\varepsilon) \mid \lambda_i(\varepsilon) \text{ is a decreasing function}\}$$

and

$$\mathcal{B}' = \{\lambda_i(\varepsilon) \mid \lambda_i(\varepsilon) \text{ is an increasing function}\}.$$

Eventually, we have to solve some equation

$$\lambda_i(\varepsilon) = \lambda_j(\varepsilon)$$

for $\lambda_i \in \mathcal{B}$ and $\lambda_j \in \mathcal{B}'$.

Balancing branching numbers

For $(4, 4, 1 + 2\varepsilon)$ and $(2 - \varepsilon, 2 - \varepsilon)$, we have

Balancing branching numbers

For $(4, 4, 1 + 2\varepsilon)$ and $(2 - \varepsilon, 2 - \varepsilon)$, we have

$$\begin{cases} x^4 - x^{4-(1-2\varepsilon)} - 2 = 0 \\ x^{2-\varepsilon} - 2 = 0. \end{cases}$$

Balancing branching numbers

For $(4, 4, 1 + 2\varepsilon)$ and $(2 - \varepsilon, 2 - \varepsilon)$, we have

$$\begin{cases} x^4 - x^{4-(1-2\varepsilon)} - 2 = 0 \\ x^{2-\varepsilon} - 2 = 0. \end{cases}$$

$\varepsilon \approx 0.21199\ldots$ and $x \approx 1.47353\ldots$

Enumeration of minimal CDS for chordal graphs

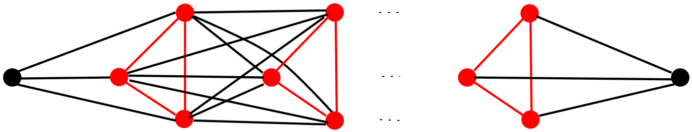
Theorem

An n-vertex chordal graph has at most 1.4736^n minimal connected dominating sets and these sets can be enumerated in time $O(1.4736)$.

Enumeration of minimal CDS for chordal graphs

Theorem

An n-vertex chordal graph has at most 1.4736^n minimal connected dominating sets and these sets can be enumerated in time $O(1.4736)$.



There are chordal graphs with at least $3^{(n-1)/3} = \Omega(1.4422^n)$ minimal CDS

Measure & Conquer

Suppose that we consider the enumeration problem where the aim is to list all subsets of verices that satisfy a certain property.

Measure & Conquer

Suppose that we consider the enumeration problem where the aim is to list all subsets of verices that satisfy a certain property.

Then the typical strategy is the following.

Measure & Conquer

Suppose that we consider the enumeration problem where the aim is to list all subsets of verices that satisfy a certain property.

Then the typical strategy is the following.

- define a weight function

$$\omega \colon V(G) \to \mathbb{R}_{\geq 0}.$$

Measure & Conquer

Suppose that we consider the enumeration problem where the aim is to list all subsets of verices that satisfy a certain property.

Then the typical strategy is the following.

- define a weight function

$$\omega\colon V(G) \to \mathbb{R}_{\geq 0}.$$

- Set the measure of an instance

$$\mu(G) = \sum_{v \in V(G)} \omega(v).$$

Weight function

- Usually, the weight $\omega(v)$ is a function of the degree, that is,

$$\omega(v) = w(d(v)).$$

Weight function

- Usually, the weight $\omega(v)$ is a function of the degree, that is,

$$\omega(v) = w(d(v)).$$

- Typically,

$$0 \leq w(0) < w(1) < \ldots w(k) = \ldots = 1.$$

Weight function

- Usually, the weight $\omega(v)$ is a function of the degree, that is,

$$\omega(v) = w(d(v)).$$

- Typically,

$$0 \leq w(0) < w(1) < \ldots w(k) = \ldots = 1.$$

- The main idea: the removal of a vertex decreases the degrees of the neighbors.

Weight function

- Usually, the weight $\omega(v)$ is a function of the degree, that is,

$$\omega(v) = w(d(v)).$$

- Typically,

$$0 \leq w(0) < w(1) < \ldots w(k) = \ldots = 1.$$

- The main idea: the removal of a vertex decreases the degrees of the neighbors.
- To determine k and $w(0), \ldots, w(k-1)$, we have to solve an auxiliary optimization problem.

Enumeration of minimal dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

Enumeration of minimal dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.

Enumeration of minimal dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.

Theorem (Fomin, Grandoni, Pyatkin, Stepanov, 2008)

An n-vertex graph has at most 1.7159^n *minimal dominating sets and these sets can be enumerated in time* $O(1.7159^n)$.

Enumeration of minimal dominating sets

A set of vertices D of a graph G is a *dominating set* if every $v \in V(G)$ is either in D or is adjacent to a vertex of D.

A dominating set is *(inclusion) minimal* if for every $Y \subset X$, Y is not a connected dominating set.

Theorem (Fomin, Grandoni, Pyatkin, Stepanov, 2008)

An n-vertex graph has at most 1.7159^n minimal dominating sets and these sets can be enumerated in time $O(1.7159^n)$.

Lower bound: there are graphs with $15^{n/6}$ (1.5704^n) minimal dominating sets.

Minimal set covers

Let $\mathcal{S}$ be a family of subsets over a universe $\mathcal{U}$.

Minimal set covers

Let $\mathcal{S}$ be a family of subsets over a universe $\mathcal{U}$.

$\mathcal{S}^* \subseteq S$ is a *set cover* if for every $u \in \mathcal{U}$ there is $S \in \mathcal{S}^*$ that *covers* u, that is, $u \in S$.

Minimal set covers

Let $\mathcal{S}$ be a family of subsets over a universe $\mathcal{U}$.

$\mathcal{S}^* \subseteq S$ is a *set cover* if for every $u \in \mathcal{U}$ there is $S \in \mathcal{S}^*$ that *covers* u, that is, $u \in S$.

A set cover $\mathcal{S}^*$ is *(inclusion) minimal* if every $\hat{\mathcal{S}} \subset \mathcal{S}^*$ is not a set cover.

Minimal set covers

Let $\mathcal{S}$ be a family of subsets over a universe $\mathcal{U}$.

$\mathcal{S}^* \subseteq S$ is a *set cover* if for every $u \in \mathcal{U}$ there is $S \in \mathcal{S}^*$ that *covers* u, that is, $u \in S$.

A set cover $\mathcal{S}^*$ is *(inclusion) minimal* if every $\hat{\mathcal{S}} \subset \mathcal{S}^*$ is not a set cover.

Let G be a graph and

$$\mathcal{U} = V(G) \text{ and } \mathcal{S} = \{N_G[v] \mid v \in V(G)\}.$$

Minimal set covers

Let $\mathcal{S}$ be a family of subsets over a universe $\mathcal{U}$.

$\mathcal{S}^* \subseteq S$ is a *set cover* if for every $u \in \mathcal{U}$ there is $S \in \mathcal{S}^*$ that *covers* u, that is, $u \in S$.

A set cover $\mathcal{S}^*$ is *(inclusion) minimal* if every $\hat{\mathcal{S}} \subset \mathcal{S}^*$ is not a set cover.

Let G be a graph and

$$\mathcal{U} = V(G) \text{ and } \mathcal{S} = \{N_G[v] \mid v \in V(G)\}.$$

Observation: $D \subseteq V(G)$ is a minimal dominating set of G if and only if $\mathcal{S}^* = \{N_G[v] \mid v \in D\}$ is a minimal set cover for $(\mathcal{S}, \mathcal{U})$.

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S},\mathcal{U})$ is defined as follows:

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S}, \mathcal{U})$ is defined as follows:

- every set $S \in \mathcal{S}$ has weight $\alpha_{|S|}$,

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S},\mathcal{U})$ is defined as follows:

- every set $S \in \mathcal{S}$ has weight $\alpha_{|S|}$,
- every $u \in \mathcal{U}$ has weight $\beta_{|u|}$ where $|u|$ denotes the *frequency* of u, that is the number of sets in $\mathcal{S}$ containing u.

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S},\mathcal{U})$ is defined as follows:

- every set $S \in \mathcal{S}$ has weight $\alpha_{|S|}$,
- every $u \in \mathcal{U}$ has weight $\beta_{|u|}$ where $|u|$ denotes the *frequency* of u, that is the number of sets in $\mathcal{S}$ containing u.

The measure

$$\mu(\mathcal{S},\mathcal{U}) = \sum_{S\in\mathcal{S}} \alpha_{|S|} + \sum_{u\in\mathcal{U}} \beta_{|u|}.$$

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S},\mathcal{U})$ is defined as follows:

- every set $S \in \mathcal{S}$ has weight $\alpha_{|S|}$,
- every $u \in \mathcal{U}$ has weight $\beta_{|u|}$ where $|u|$ denotes the *frequency* of u, that is the number of sets in $\mathcal{S}$ containing u.

The measure

$$\mu(\mathcal{S},\mathcal{U}) = \sum_{S\in\mathcal{S}} \alpha_{|S|} + \sum_{u\in\mathcal{U}} \beta_{|u|}.$$

- $1 \leq \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 = \alpha_i$ for $i \geq 5$,

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S},\mathcal{U})$ is defined as follows:

- every set $S \in \mathcal{S}$ has weight $\alpha_{|S|}$,
- every $u \in \mathcal{U}$ has weight $\beta_{|u|}$ where $|u|$ denotes the *frequency* of u, that is the number of sets in $\mathcal{S}$ containing u.

The measure

$$\mu(\mathcal{S},\mathcal{U}) = \sum_{S\in\mathcal{S}} \alpha_{|S|} + \sum_{u\in\mathcal{U}} \beta_{|u|}.$$

- $1 \leq \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 = \alpha_i$ for $i \geq 5$,
- $0 = \beta_1 < \beta_2 < \beta_3 < \beta_4 < \beta_5 = 1 = \beta_j$ for $j \geq 6$.

Measure & Conquer for minimal set covers

The weight of $(\mathcal{S},\mathcal{U})$ is defined as follows:

- every set $S \in \mathcal{S}$ has weight $\alpha_{|S|}$,
- every $u \in \mathcal{U}$ has weight $\beta_{|u|}$ where $|u|$ denotes the *frequency* of u, that is the number of sets in $\mathcal{S}$ containing u.

The measure

$$\mu(\mathcal{S},\mathcal{U}) = \sum_{S\in\mathcal{S}} \alpha_{|S|} + \sum_{u\in\mathcal{U}} \beta_{|u|}.$$

- $1 \leq \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 = \alpha_i$ for $i \geq 5$,
- $0 = \beta_1 < \beta_2 < \beta_3 < \beta_4 < \beta_5 = 1 = \beta_j$ for $j \geq 6$.

The weights $\alpha_1, \ldots, \alpha_4$ and $\beta_2, \ldots, \beta_5$ are found by making use of the *quasiconvex programming*.