Enumerating minimal dominating sets in triangle-free graphs

Jean-Florent Raymond
(TU Berlin)

Wednesday 31st October, 2018

Joint work with **Marthe Bonamy** (LaBRI, Bordeaux), **Oscar Defrain** (LIMOS, Clermont-Ferrand), and **Marc Heinrich** (LIRIS, Lyon).

Typical question:

given input I, list all objects of type X in I

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph
- minimal transversals of an hypergraph

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph
- minimal transversals of an hypergraph
- variable assignements satisfying a formula

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph
- minimal transversals of an hypergraph
- variable assignements satisfying a formula
- minimal story arc sets in a metabolic network

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph
- minimal transversals of an hypergraph
- variable assignements satisfying a formula
- minimal story arc sets in a metabolic network
- trains to Hamburg leaving tomorrow before 10:00

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph
- minimal transversals of an hypergraph
- variable assignements satisfying a formula
- minimal story arc sets in a metabolic network
- trains to Hamburg leaving tomorrow before 10:00
- shortest paths to Alexanderplatz

Typical question:

given input I, list all objects of type X in I

- cycles, cliques, VCs, dominating sets, etc. of a graph
- minimal transversals of an hypergraph
- variable assignements satisfying a formula
- minimal story arc sets in a metabolic network
- trains to Hamburg leaving tomorrow before 10:00
- shortest paths to Alexanderplatz

Problem: possibly many objects!

Input-sensitive: in terms of input size

Input-sensitive: in terms of input size

- [Fomin, Grandoni, and Pyatkin, 2008]:
  $O\left(1.7159^n\right)$-time for Minimal Dominating Set

Input-sensitive: in terms of input size

- [Fomin, Grandoni, and Pyatkin, 2008]:
  $O(1.7159^n)$-time for Minimal Dominating Set
- upper-bounds the number of objects

Input-sensitive: in terms of input size

- [Fomin, Grandoni, and Pyatkin, 2008]:
  $O(1.7159^n)$-time for Minimal Dominating Set
- upper-bounds the number of objects

Output-sensitive: in terms of input+output size

Two perspectives about complexity

Input-sensitive: in terms of input size

- [Fomin, Grandoni, and Pyatkin, 2008]:
  $O\left(1.7159^n\right)$-time for Minimal Dominating Set
- upper-bounds the number of objects

Output-sensitive: in terms of input+output size

- [Fredman and Khachiyan, 1996]:
  $O\left(s^{\log s}\right)$-time for Minimal Dominating Set
  $s = \text{poly}\,|G| + |\mathcal{D}(G)|$

n: input size

n: input size

- **output-polynomial:** poly$(n + s)$ to get all the solutions

  s: number of solutions

"Fast" output-sensitive algorithms

n: input size

- **output-polynomial:** $\text{poly}(n+s)$ to get all the solutions
  s: number of solutions

- **polynomial delay:** $\text{poly}(n)$ for each new solution

“FAST” OUTPUT-SENSITIVE ALGORITHMS

n: input size

- **output-polynomial:** $\text{poly}(n + s)$ to get all the solutions
  - s: number of solutions
- **incremental poly. delay:** $\text{poly}(n + p)$ for each new solution
  - p: size of the already output data
- **polynomial delay:** $\text{poly}(n)$ for each new solution

n: input size

- **output-polynomial:** $\mathrm{poly}(n+s)$ to get all the solutions

  s: number of solutions
- **incremental poly. delay:** $\mathrm{poly}(n+p)$ for each new solution

  p: size of the already output data
- **polynomial delay:** $\mathrm{poly}(n)$ for each new solution
- **linear delay:** ...

n: input size

- **output-polynomial:** $\text{poly}(n+s)$ to get all the solutions
  s: number of solutions
- **incremental poly. delay:** $\text{poly}(n+p)$ for each new solution
  p: size of the already output data
- **polynomial delay:** $\text{poly}(n)$ for each new solution
- **linear delay:** ...
- **constant delay:**...

n: input size

- **output-polynomial:** $\text{poly}(n + s)$ to get all the solutions
  s: number of solutions
- **incremental poly. delay:** $\text{poly}(n + p)$ for each new solution
  p: size of the already output data
- **polynomial delay:** $\text{poly}(n)$ for each new solution
- **linear delay:** ...
- **constant delay:**...

Note: output space is not counted.

- $N(v)$: neighbors of v

Minimal dominating sets

- $N(v)$: neighbors of v
- $N[v] = N(v) \cup \{v\}$

Minimal dominating sets

- $N(v)$: neighbors of v
- $N[v] = N(v) \cup \{v\}$
- dominating set (DS): $D \subseteq V(G)$ s.t. $N[D] = V(G)$

  "D can see everybody"

- $N(v)$: neighbors of v
- $N[v] = N(v) \cup \{v\}$
- dominating set (DS): $D \subseteq V(G)$ s.t. $N[D] = V(G)$

  "D can see everybody"
- minimal dominating set: inclusion-wise minimal DS

- $N(v)$: neighbors of v
- $N[v] = N(v) \cup \{v\}$
- dominating set (DS): $D \subseteq V(G)$ s.t. $N[D] = V(G)$

  "D can see everybody"
- minimal dominating set: inclusion-wise minimal DS
- private neighbor of $v \in D$: vertex not dominated by $D \setminus \{v\}$

  (possibly v itself)

- $N(v)$: neighbors of v
- $N[v] = N(v) \cup \{v\}$
- dominating set (DS): $D \subseteq V(G)$ s.t. $N[D] = V(G)$
  "D can see everybody"
- minimal dominating set: inclusion-wise minimal DS
- private neighbor of $v \in D$: vertex not dominated by $D \setminus \{v\}$
  (possibly v itself)

Simple observation

$D \subseteq V(G)$ minimal dominating set iff

- D dominates G, and
- every $v \in D$ has a private neighbor (v is irredundant).

Goal: output-polynomial algorithms

Running time polynomial in (input size + output size)

Goal: output-polynomial algorithms

Running time polynomial in (input size + output size)

General case: open (best is quasi-polynomial)

Goal: output-polynomial algorithms
Running time polynomial in (input size + output size)

General case: open (best is quasi-polynomial)

Known: degenerate graphs, graphs of bounded clique-width or mim-width, split graphs, graphs of girth $\geqslant 7$, etc.

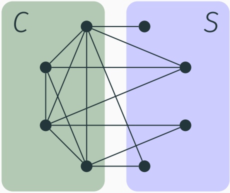
Goal: output-polynomial algorithms
Running time polynomial in (input size + output size)

General case: open (best is quasi-polynomial)

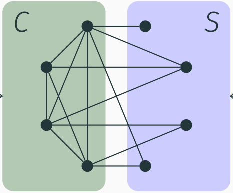
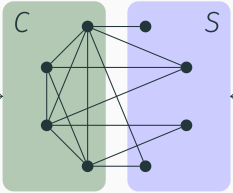
Known: degenerate graphs, graphs of bounded clique-width or mim-width, split graphs, graphs of girth $\geqslant 7$, etc.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)
There is an output-polynomial algorithm enumerating minimal dominating sets in triangle-free graphs.

Minimal dominating sets in split graphs



Minimal dominating sets in split graphs

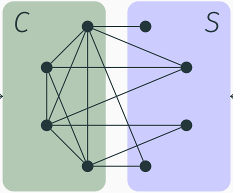
clique⟶  ⟵stable set clique⟶  ⟵stable set (maximal)

D minimal DS in G

$\Updownarrow$

$D \cap C$ is irredundant (every vertex has a private neighbor)

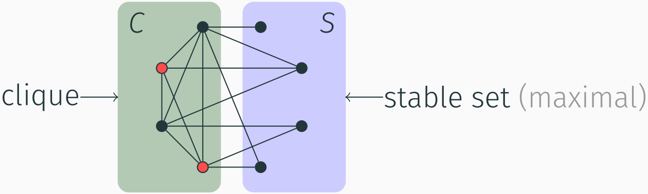
and ...

clique⟶  ⟵stable set (maximal)

D minimal DS in G

$\Updownarrow$

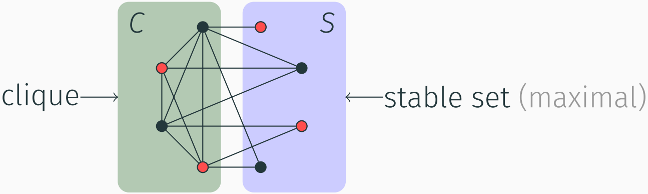
$D \cap C$ is irredundant (every vertex has a private neighbor)
and $D \cap S = \{$all vertices not dominated by $D \cap C\}$



D minimal DS in G

$\Updownarrow$

$D \cap C$ is irredundant (every vertex has a private neighbor)
and $D \cap S$ = {all vertices not dominated by $D \cap C$}

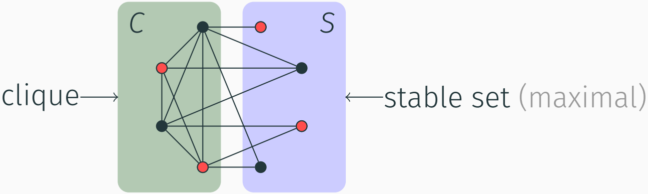


D minimal DS in G

$\Updownarrow$

$D \cap C$ is irredundant (every vertex has a private neighbor)

and $D \cap S = \{$all vertices not dominated by $D \cap C\}$



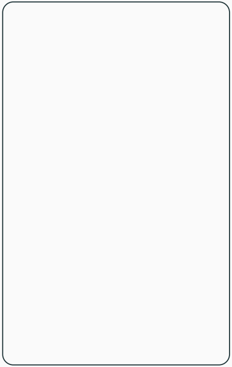
D minimal DS in G

$\Updownarrow$

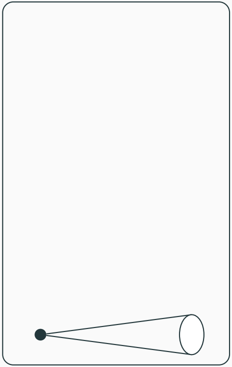
$D \cap C$ is irredundant (every vertex has a private neighbor)
and $D \cap S = \{$all vertices not dominated by $D \cap C\}$

Enumeration: complete in S every irredundant $X \subseteq C$
$\rightsquigarrow$ linear delay [Kanté, Limouzy, Mary, and Nourine, 2014]

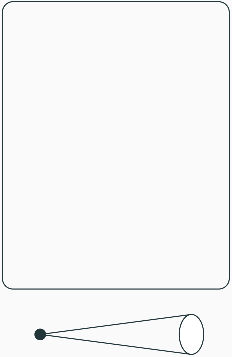
Goal: enumeration of minimal DS in triangle-free graphs



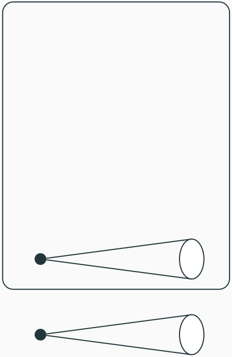
Goal: enumeration of minimal DS in triangle-free graphs



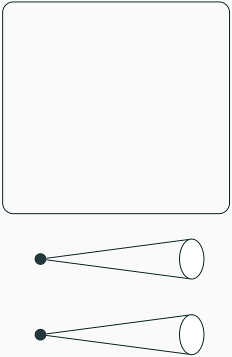
Goal: enumeration of minimal DS in triangle-free graphs



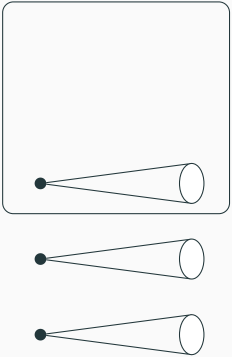
Goal: enumeration of minimal DS in triangle-free graphs



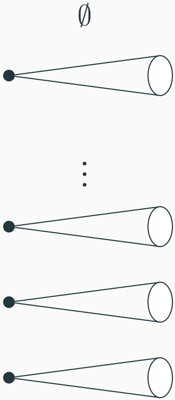
Goal: enumeration of minimal DS in triangle-free graphs



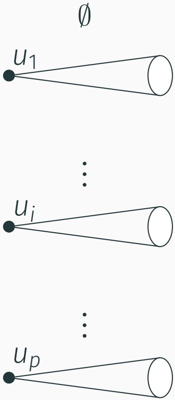
Goal: enumeration of minimal DS in triangle-free graphs



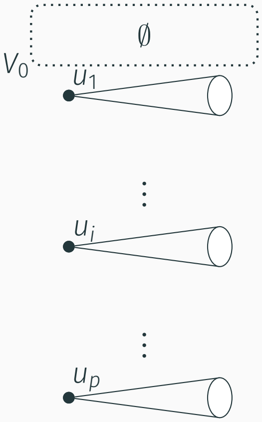
Goal: enumeration of minimal DS in triangle-free graphs



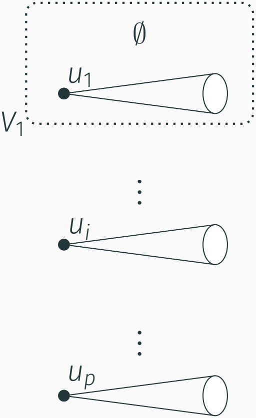
Goal: enumeration of minimal DS in triangle-free graphs



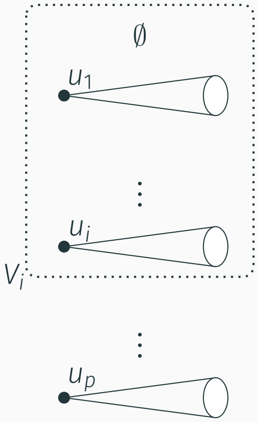
Goal: enumeration of minimal DS in triangle-free graphs



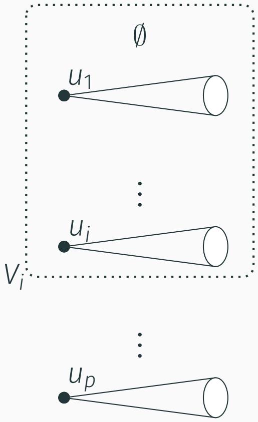
Goal: enumeration of minimal DS in triangle-free graphs



Goal: enumeration of minimal DS in triangle-free graphs



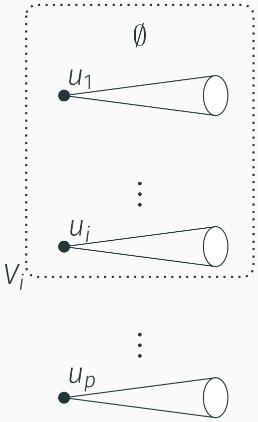
Goal: enumeration of minimal DS in triangle-free graphs



Plan:

1. enumerate the minimal DS of V_i
2. use them to enumerate those of V_{i+1}

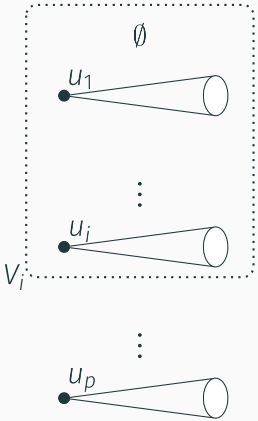
Goal: enumeration of minimal DS in triangle-free graphs



Plan:

1. enumerate the minimal DS of V_i
   with vertices of G
2. use them to enumerate those of V_{i+1}

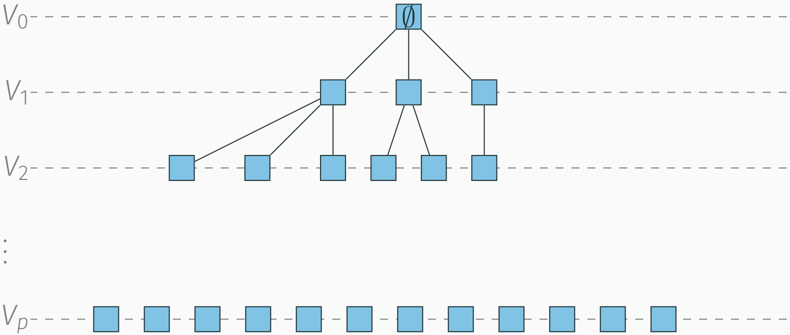
Goal: enumeration of minimal DS in triangle-free graphs

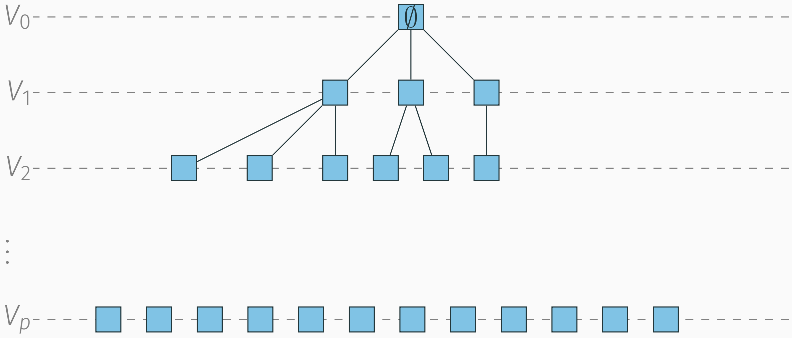


Plan:

1. enumerate the minimal DS of V_i
   with vertices of G
2. use them to enumerate those of V_{i+1}
   extend each minimal DS of V_i
   to a minimal DS of V_{i+1}

Growing partial minimal dominating sets

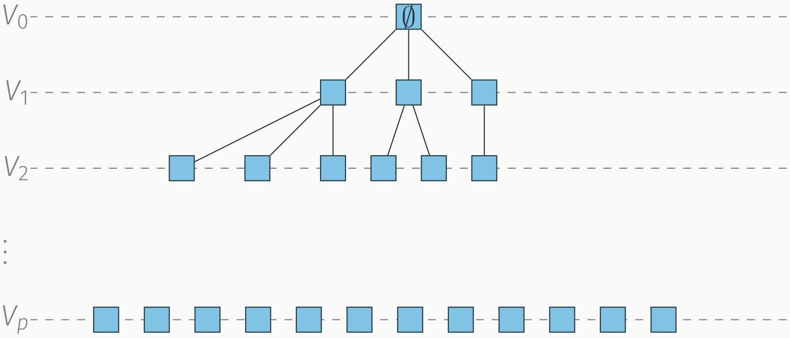




Important properties:

- no cycle

Growing partial minimal dominating sets



Important properties:

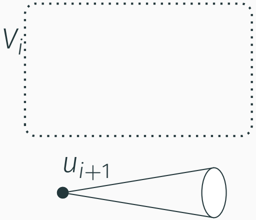
- no cycle
- no leaf before level p

From V_i to V_{i+1}

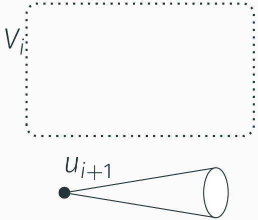
Goal: extend a minimal DS D of V_i to a minimal DS of V_{i+1}

From V_i to V_{i+1}

Goal: extend a minimal DS D of V_i to a minimal DS of V_{i+1}



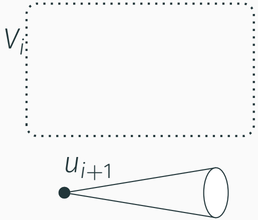
Goal: extend a minimal DS D of V_i to a minimal DS of V_{i+1}



Observation:

- possibly D dominates V_{i+1}
- if not then $D \cup \{u_{i+1}\}$ does.

Goal: extend a minimal DS D of V_i to a minimal DS of V_{i+1}



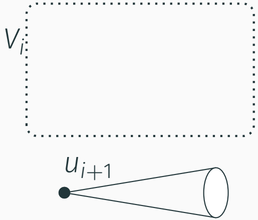
Observation:

- possibly D dominates V_{i+1}
- if not then $D \cup \{u_{i+1}\}$ does.

extension is always possible

From V_i to V_{i+1}

Goal: extend a minimal DS D of V_i to a minimal DS of V_{i+1}



Observation:

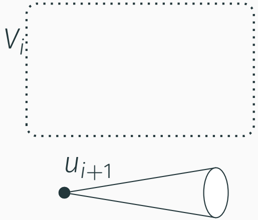
- possibly D dominates V_{i+1}
- if not then $D \cup \{u_{i+1}\}$ does.

extension is always possible

candidate extension: minimal X s.t. $D \cup X$ dominates V_{i+1}

From V_i to V_{i+1}

Goal: extend a minimal DS D of V_i to a minimal DS of V_{i+1}



Observation:

- possibly D dominates V_{i+1}
- if not then $D \cup \{u_{i+1}\}$ does.

extension is always possible

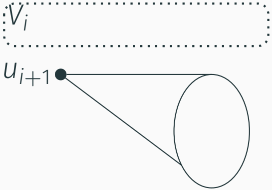
candidate extension: minimal X s.t. $D \cup X$ dominates V_{i+1}

Lemma: $|\text{candidate extensions of } D| \leqslant |\text{minimal DS of } G|$
(so we can try them all even if only few *work*)

Which are the candidate extensions of D?

D: minimal DS of V_i

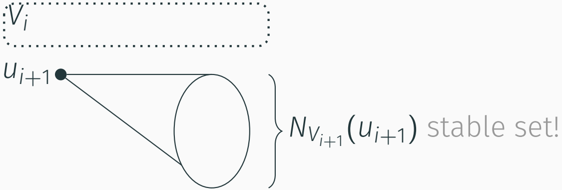
Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}



Which are the candidate extensions of D?

D: minimal DS of V_i

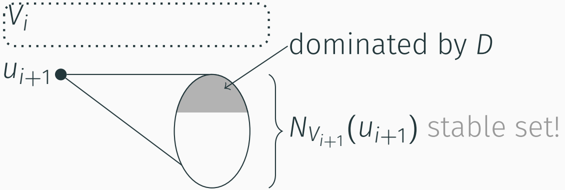
Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}



Which are the candidate extensions of D?

D: minimal DS of V_i

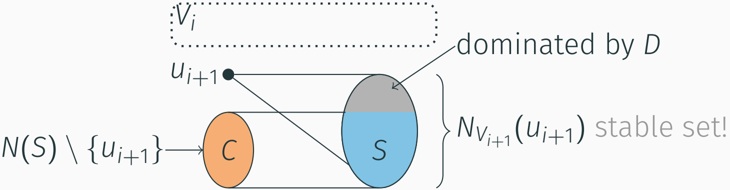
Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}



Which are the candidate extensions of D?

D: minimal DS of V_i

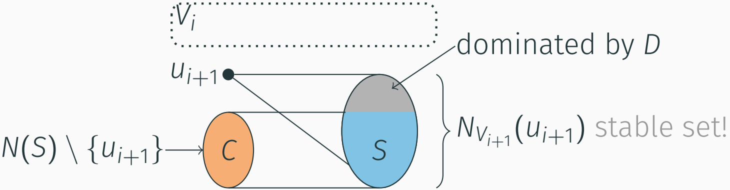
Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}



Which are the candidate extensions of D?

D: minimal DS of V_i

Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}

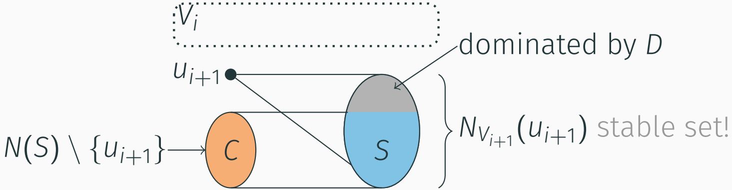


- if u_{i+1} dom. by D: they only have to dominate S

Which are the candidate extensions of D?

D: minimal DS of V_i

Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}

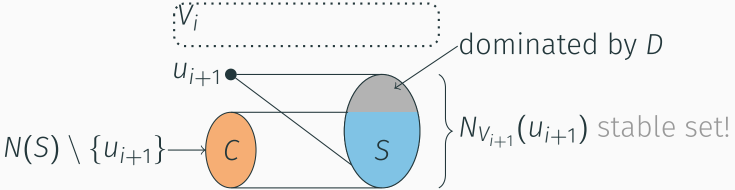


- if u_{i+1} dom. by D: they only have to dominate S
  $\rightarrow$ exactly the minimal DS of $\mathrm{Split}(C, S)$

Which are the candidate extensions of D?

D: minimal DS of V_i

Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}

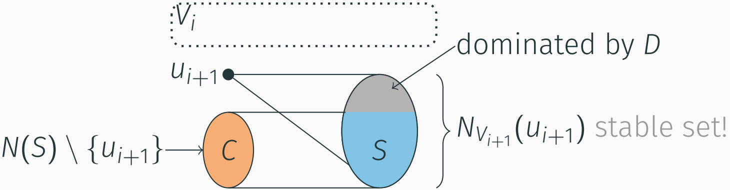


- if u_{i+1} dom. by D: they only have to dominate S
  $\rightarrow$ exactly the minimal DS of $\mathrm{Split}(C, S)$
- or u_{i+1} not dom. by D: they should dom. u_{i+1} as well

WHICH ARE THE CANDIDATE EXTENSIONS OF D?

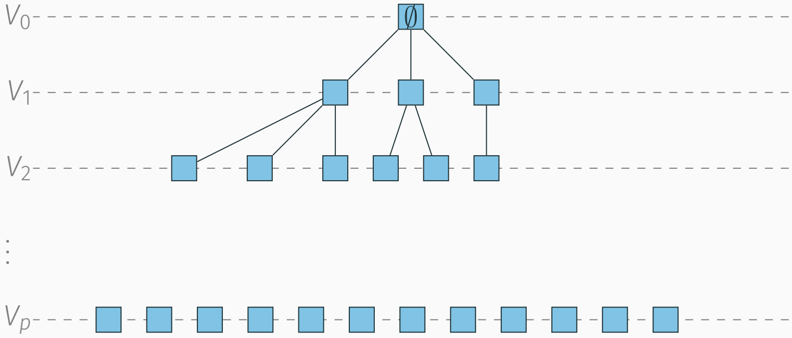
D: minimal DS of V_i

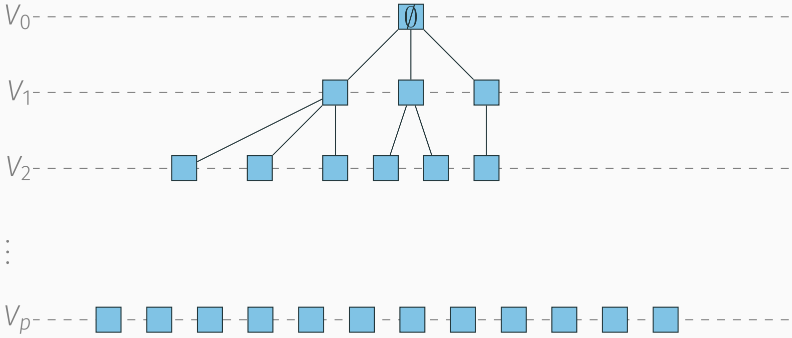
Candidate ext.: minimal X s.t. $D \cup X$ dominates V_{i+1}



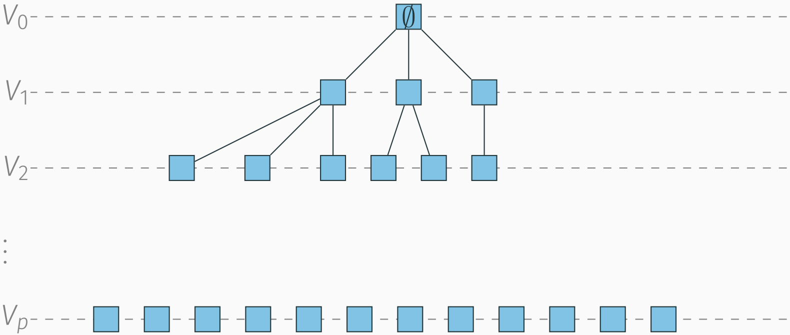
- if u_{i+1} dom. by D: they only have to dominate S
  $\rightarrow$ exactly the minimal DS of $\mathrm{Split}(C, S)$
- or u_{i+1} not dom. by D: they should dom. u_{i+1} as well
  $\rightarrow$ irredundant $\{t\} \cup Q$ s.t. $\begin{cases} t \in N(u_{i+1}) \\ Q \subseteq C \text{ minimal DS of } \mathrm{Split}(C, S) \end{cases}$

The algorithm





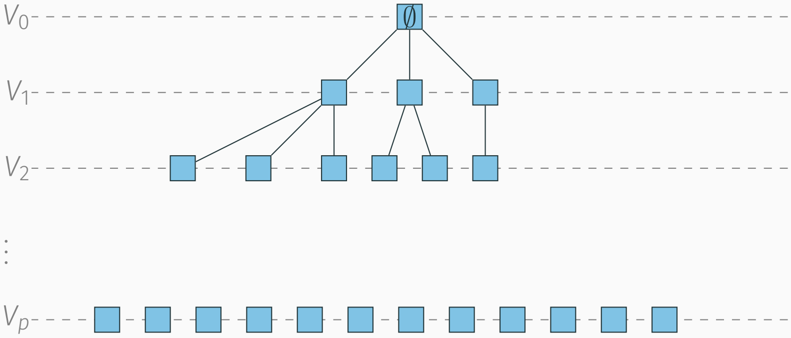
For each D minimal DS of V_i:



For each D minimal DS of V_i:

- compute all candidate extensions;

The algorithm



For each D minimal DS of V_i:

- compute all candidate extensions;
- only keep the $X \cup D$'s that are minimal and children of D.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

The set $\mathcal{D}(G)$ of minimal dominating sets of any triangle-free graph G can be enumerated in time $\mathrm{poly}(|G|) \cdot |\mathcal{D}(G)|^2$ and polynomial space.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

The set $\mathcal{D}(G)$ of minimal dominating sets of any triangle-free graph G can be enumerated in time $\mathrm{poly}(|G|) \cdot |\mathcal{D}(G)|^2$ and polynomial space.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

Deciding if a vertex set X can be extended into a minimal dominating set is NP-complete in bipartite graphs.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

The set $\mathcal{D}(G)$ of minimal dominating sets of any triangle-free graph G can be enumerated in time $\mathrm{poly}(|G|) \cdot |\mathcal{D}(G)|^2$ and polynomial space.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

Deciding if a vertex set X can be extended into a minimal dominating set is NP-complete in bipartite graphs.

- complexity improvements?

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

The set $\mathcal{D}(G)$ of minimal dominating sets of any triangle-free graph G can be enumerated in time $\mathrm{poly}(|G|) \cdot |\mathcal{D}(G)|^2$ and polynomial space.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

Deciding if a vertex set X can be extended into a minimal dominating set is NP-complete in bipartite graphs.

- complexity improvements?
- extensions to other classes?

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

The set $\mathcal{D}(G)$ of minimal dominating sets of any triangle-free graph G can be enumerated in time $\mathrm{poly}(|G|) \cdot |\mathcal{D}(G)|^2$ and polynomial space.

Theorem (Bonamy, Defrain, Heinrich, R., 2018+)

Deciding if a vertex set X can be extended into a minimal dominating set is NP-complete in bipartite graphs.

- complexity improvements?
- extensions to other classes?

Thank you!