Efficiently Enumerating Hitting Sets of Hypergraphs Arising in Data Profiling

Martin Schirneck

Joint work with Thomas Bläsius, Tobias Friedrich, Julius Lischeid, and Kitty Meeks,

to appear at ALENEX 2019.

Dagstuhl - 16 October 2018

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema,

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema, attributes (columns),

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema, attributes (columns), tuples (rows),

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema, attributes (columns), tuples (rows), values.

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema, attributes (columns), tuples (rows), values.
- Metadata: dependencies between attributes.

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema, attributes (columns), tuples (rows), values.
- Metadata: dependencies between attributes.
- Unique column combination (UCC): entries identify full tuple.

Data Profiling

Data profiling is the gathering of metadata from databases.

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

- Relational database: schema, attributes (columns), tuples (rows), values.
- Metadata: dependencies between attributes.
- Unique column combination (UCC): entries identify full tuple.

Task: Enumerate all inclusion-wise minimal UCCs.

From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

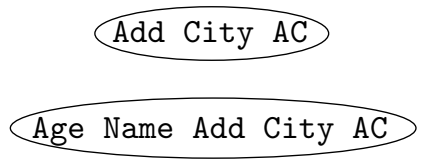
From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

Add City AC

From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

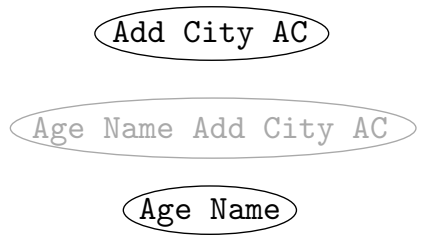


From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145

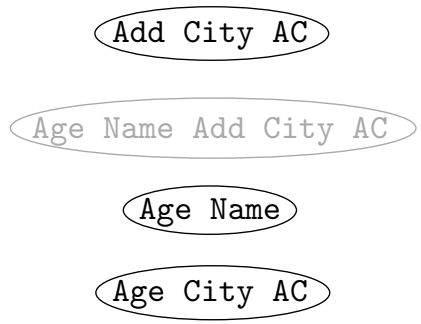
From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145



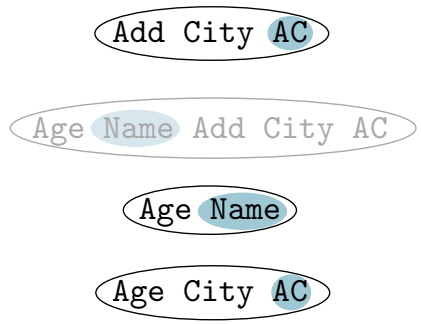
From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145



From UCCs to Hitting Sets

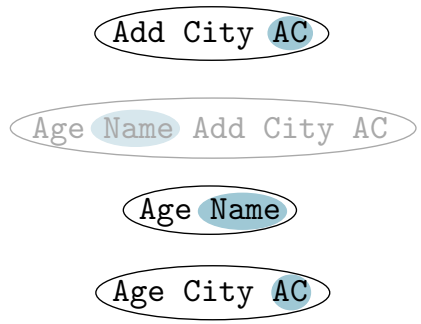
Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145



Minimal UCCs = minimal transversals of the hypergraph of difference sets.

From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145



Minimal UCCs = minimal transversals of the hypergraph of difference sets.

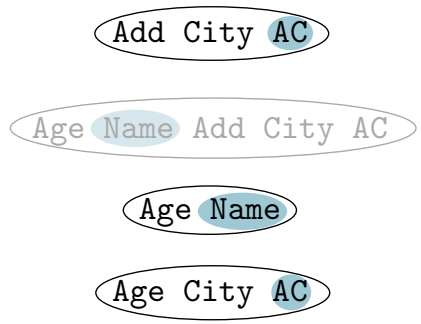
UCCs and the Transversal Hypergraph

There is a parsimonious polynomial reduction that preserves inclusions...

- ...from UCC to HittingSet. [Folklore]

From UCCs to Hitting Sets

Age	Name	Address	City	Area Code
47	Mustermann, Max	Mittelstraße 125	Potsdam	D-14467
47	Mustermann, Max	Oktavie-Allee 1	Wadern	D-66687
76	Doe, John	South Street 8	London	UK-W1K
90	Nightingale, Florence	South Street 8	London	UK-W1K
25	Menigmand, Morten	Trøjburgvej 24	Aarhus	DK-8200
33	Doe, John	South Street 8	Philadelphia	US-PA-19145



Minimal UCCs = minimal transversals of the hypergraph of difference sets.

UCCs and the Transversal Hypergraph

There is a parsimonious polynomial reduction that preserves inclusions...

- ...from UCC to HittingSet. [Folklore]
- ...from HittingSet to UCC. [Bläsius, Friedrich & Sch. 2016]

Enumeration with Polymonial Delay

Notation:

- n = number of vertices/attributes.
- m = number of hyperedges/minimal difference sets.
- k^* = size of the largest minimal hitting set/UCC.

Enumeration with Polymonial Delay

Notation:

- n = number of vertices/attributes.
- m = number of hyperedges/minimal difference sets.
- k^* = size of the largest minimal hitting set/UCC.

There is an enumeration algorithm for minimal hitting sets/UCCs that...

- ...has delay $O(m^{k^*+1}n^2)$, polynomial delay if k^* is a constant.

Enumeration with Polymonial Delay

Notation:

- n = number of vertices/attributes.
- m = number of hyperedges/minimal difference sets.
- k^* = size of the largest minimal hitting set/UCC.

There is an enumeration algorithm for minimal hitting sets/UCCs that...

- ...has delay $O(m^{k^*+1}n^2)$, polynomial delay if k^* is a constant.
  - Known: $k^* = O(1) \Rightarrow$ incremental-polynomial algorithm. [Eiter & Gottlob 1995]

Enumeration with Polymonial Delay

Notation:

- n = number of vertices/attributes.
- m = number of hyperedges/minimal difference sets.
- k^* = size of the largest minimal hitting set/UCC.

There is an enumeration algorithm for minimal hitting sets/UCCs that...

- ...has delay $O(m^{k^*+1}n^2)$, polynomial delay if k^* is a constant.
  - Known: $k^* = O(1) \Rightarrow$ incremental-polynomial algorithm. [Eiter & Gottlob 1995]
- ...uses space $O(mn)$.

Enumeration with Polymonial Delay

Notation:

- n = number of vertices/attributes.
- m = number of hyperedges/minimal difference sets.
- k^* = size of the largest minimal hitting set/UCC.

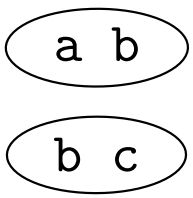
There is an enumeration algorithm for minimal hitting sets/UCCs that...

- ...has delay $O(m^{k^*+1}n^2)$, polynomial delay if k^* is a constant.
  - Known: $k^* = O(1) \Rightarrow$ incremental-polynomial algorithm. [Eiter & Gottlob 1995]
- ...uses space $O(mn)$.
- ...is fast in practice!

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

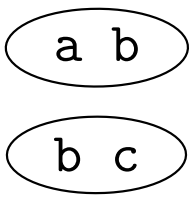


Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

X, Y



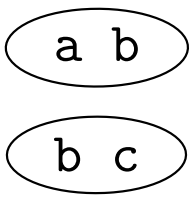
- Disjoint sets: partial solution X, excluded vertices Y

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

X, Y



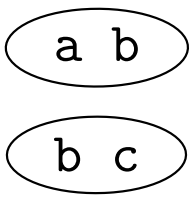
- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

X, Y $\emptyset, \emptyset$

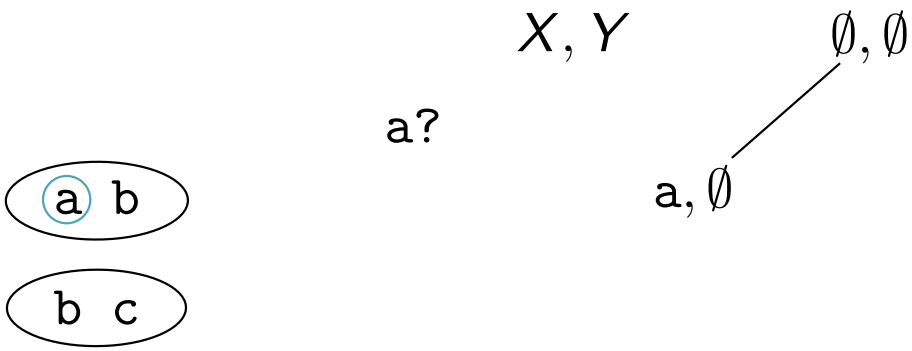


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

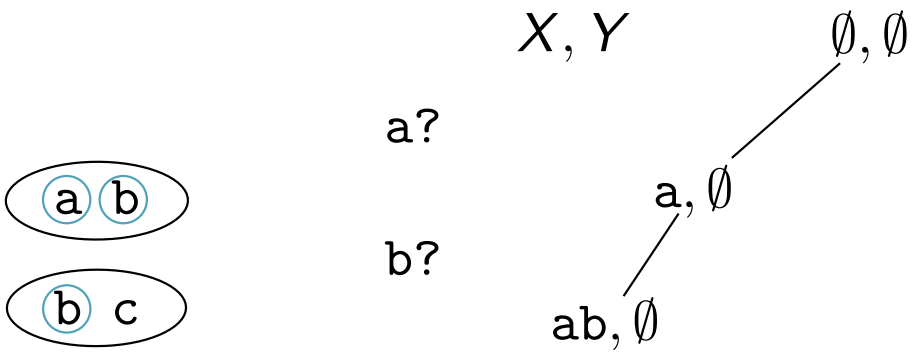


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

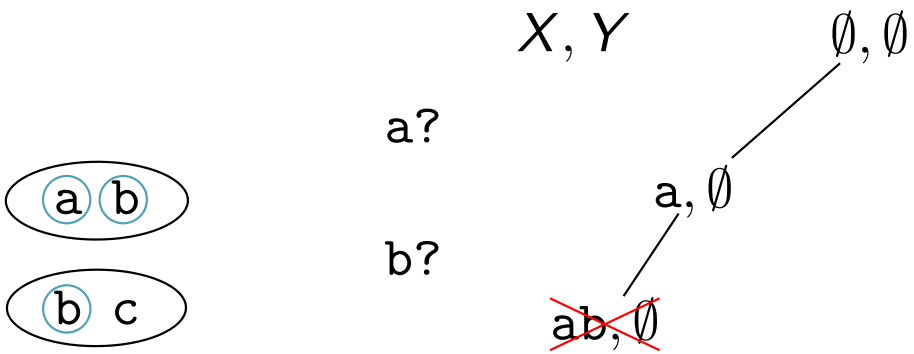


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

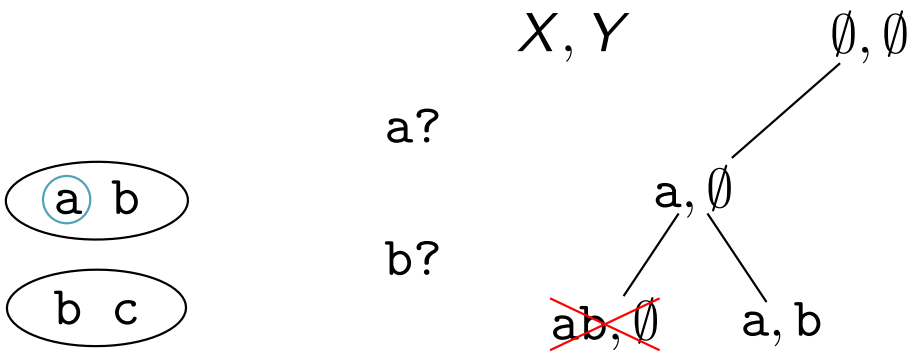


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

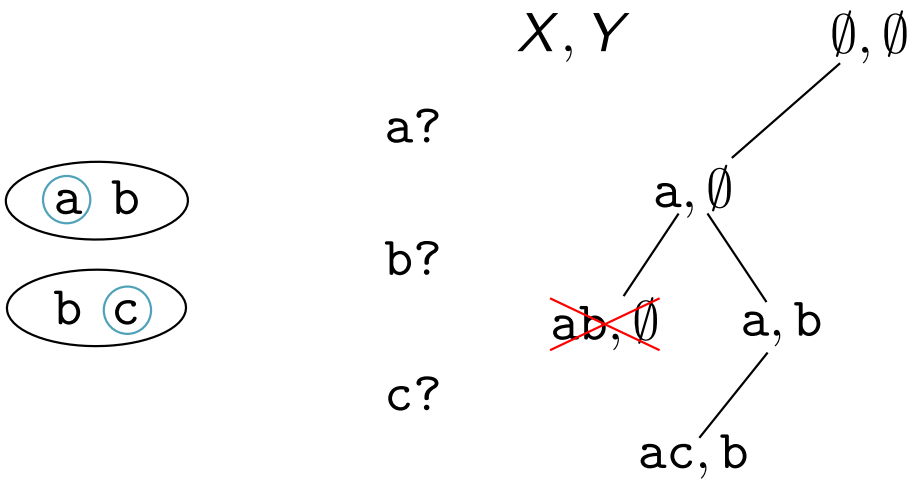


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

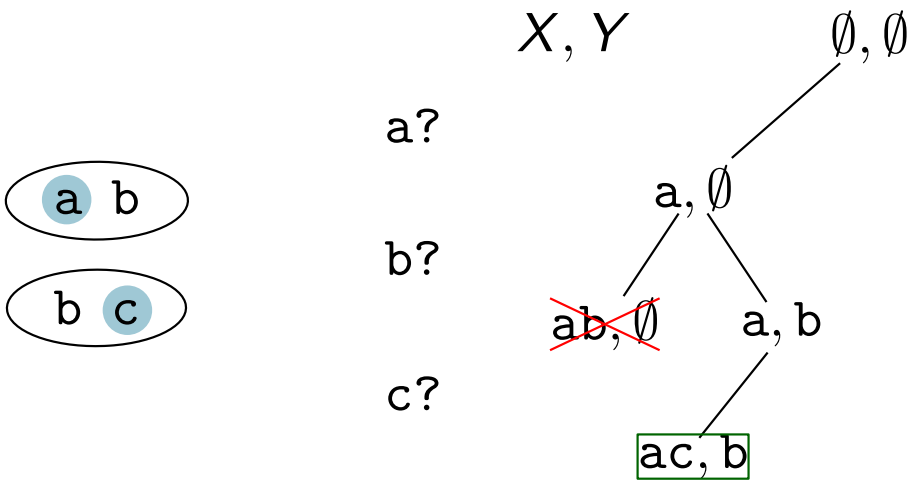


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

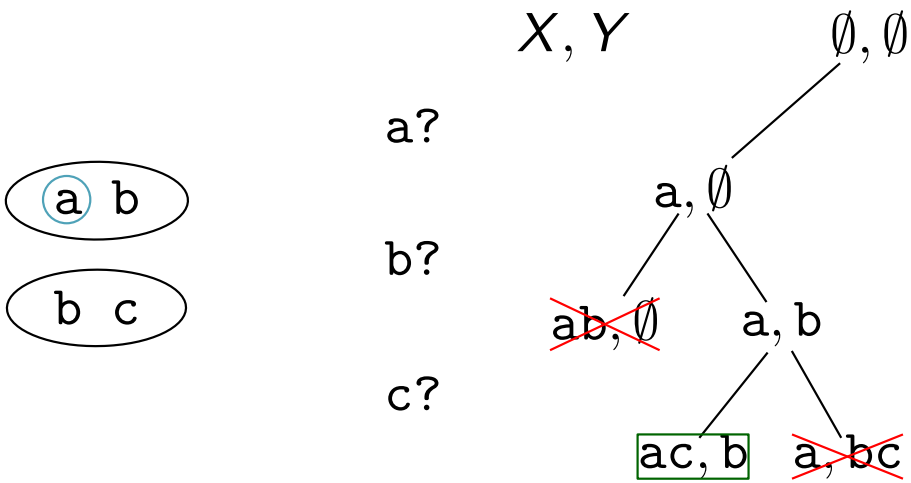


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

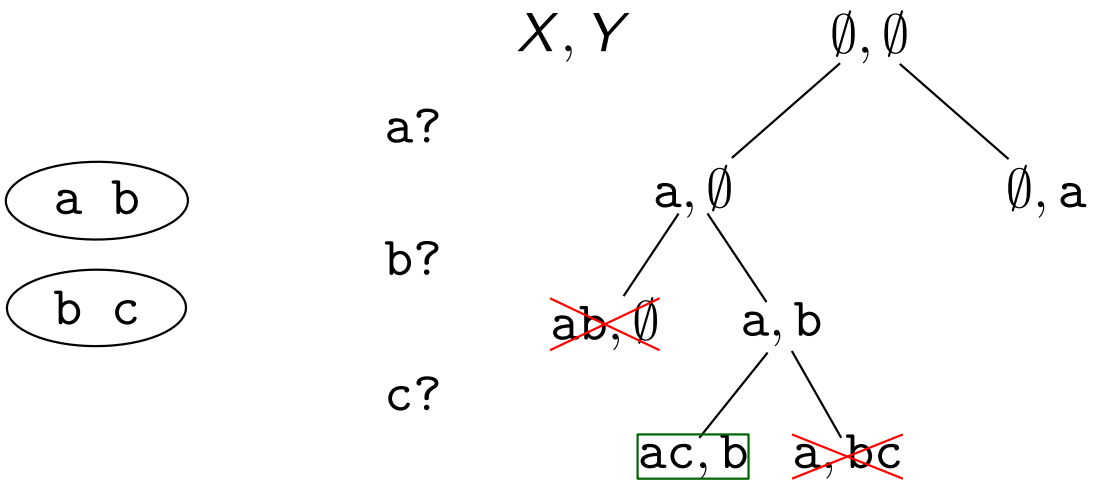


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

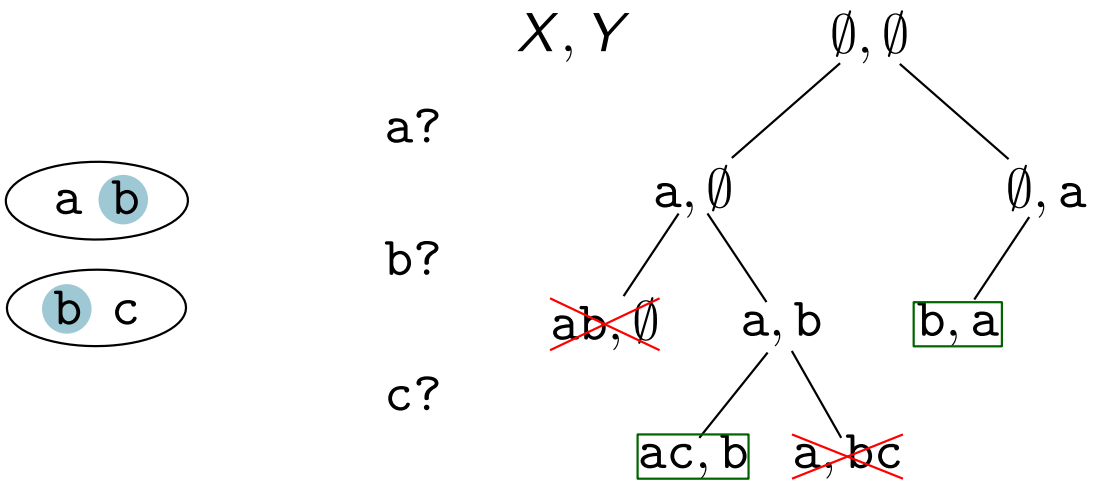


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.

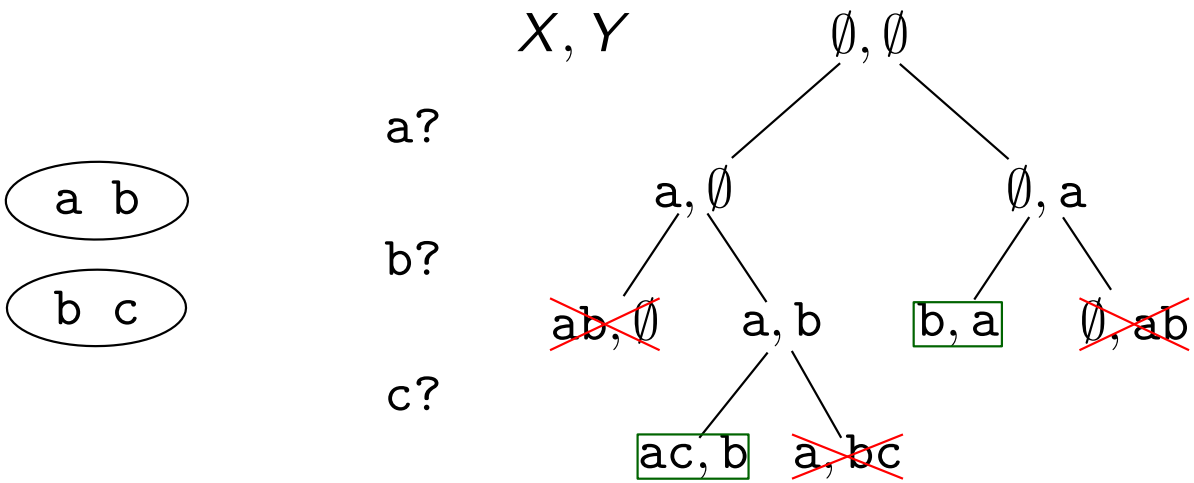


- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?
- Answers: EXTENDABLE, MINIMAL, or NOT EXTENDABLE

Backtracking Enumeration

a.k.a. the flashlight technique. [Mary & Strozecki 2016]

Idea: decision tree pruned by an extension oracle.



- Disjoint sets: partial solution X, excluded vertices Y
- Can X be extended to a minimal hitting set avoiding Y?
- Answers: EXTENDABLE, MINIMAL, or NOT EXTENDABLE

Extension Problem

Extension Problem for Minimal Hitting Sets

Let X, Y be disjoint set of vertices, $X \cap Y = \emptyset$.

(i) Is there a minimal hitting set H s.t. $X \subseteq H$ and $H \cap Y = \emptyset$?

(ii) If so, is $H = X$?

Extension Problem

Extension Problem for Minimal Hitting Sets

Let X, Y be disjoint set of vertices, $X \cap Y = \emptyset$.

(i) Is there a minimal hitting set H s.t. $X \subseteq H$ and $H \cap Y = \emptyset$?

(ii) If so, is $H = X$?

NP-complete in general, but tractable if $|X|$ is small. [Boros, Gurvich & Hammer 1998]

Extension Problem

Extension Problem for Minimal Hitting Sets

Let X, Y be disjoint set of vertices, $X \cap Y = \emptyset$.

(i) Is there a minimal hitting set H s.t. $X \subseteq H$ and $H \cap Y = \emptyset$?

(ii) If so, is $H = X$?

NP-complete in general, but tractable if $|X|$ is small. [Boros, Gurvich & Hammer 1998]

No hope for an FPT-algorithm:

- W[3]-complete when parameterized by $|X|$.

Extension Problem

Extension Problem for Minimal Hitting Sets

Let X, Y be disjoint set of vertices, $X \cap Y = \emptyset$.

(i) Is there a minimal hitting set H s.t. $X \subseteq H$ and $H \cap Y = \emptyset$?

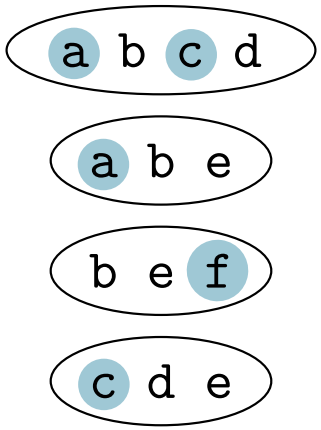
(ii) If so, is $H = X$?

NP-complete in general, but tractable if $|X|$ is small. [Boros, Gurvich & Hammer 1998]

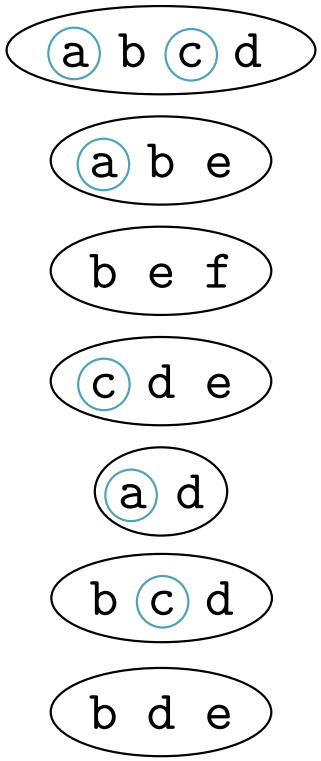
No hope for an FPT-algorithm:

- W[3]-complete when parameterized by $|X|$.
- Under ETH: not solvable in time $f(|X|) \cdot (m+n)^{o(|X|)}$ for any f.

Finding the True Witnesses

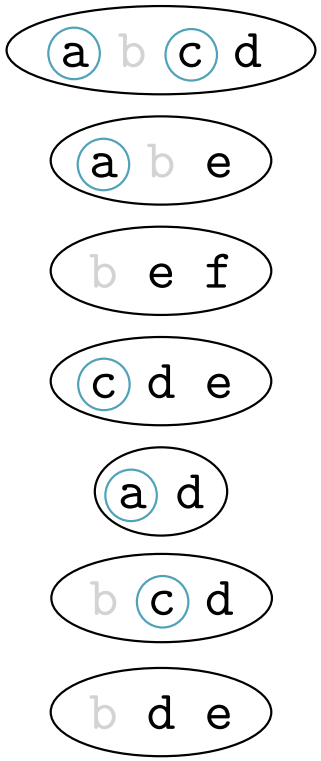


Finding the True Witnesses



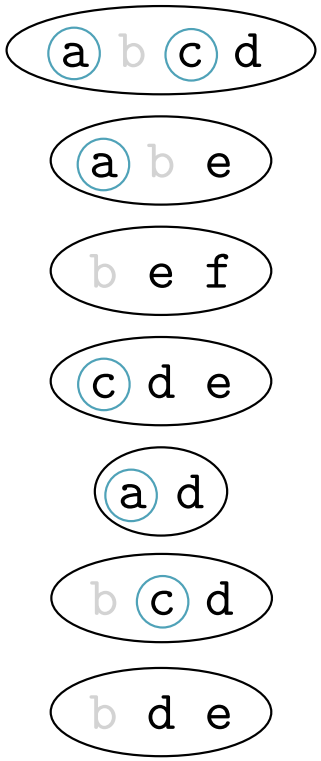
$X = \{a,c\}$

Finding the True Witnesses



$$X = \{a,c\}$$
$$Y = \{b\}$$

Finding the True Witnesses

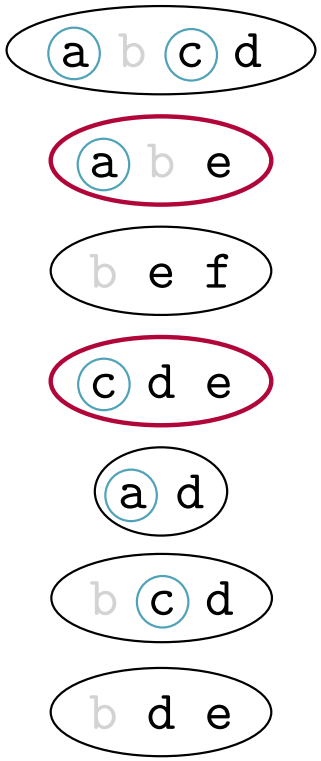


$X = \{a,c\}$

$Y = \{b\}$

- Exactly one element of X: potential witness.

Finding the True Witnesses

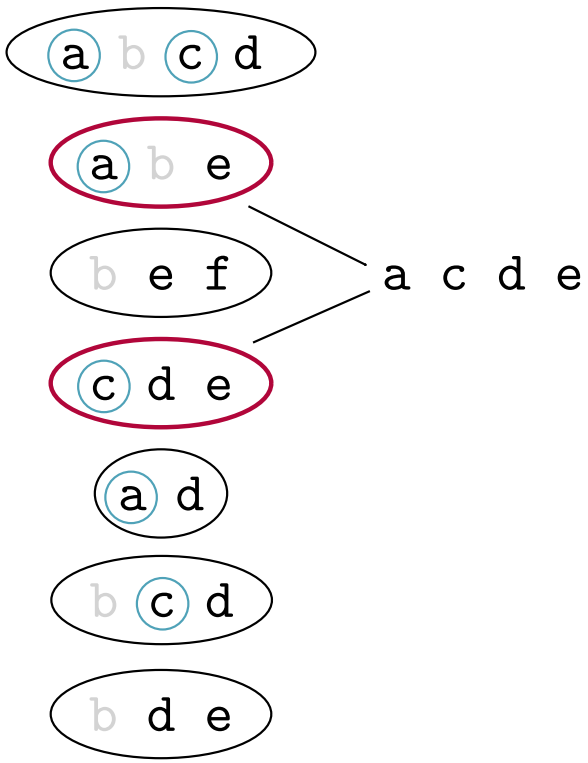


$X = \{a,c\}$

$Y = \{b\}$

- Exactly one element of X: potential witness.

Finding the True Witnesses

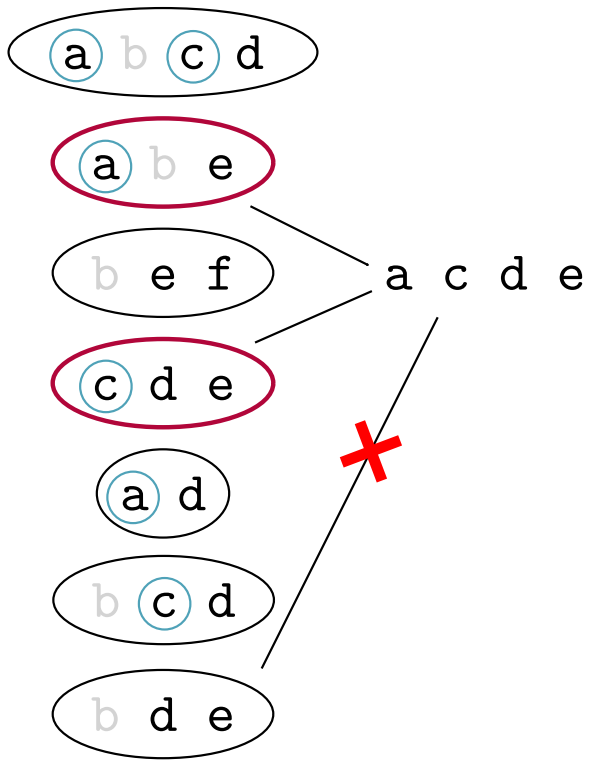


$X = \{a,c\}$

$Y = \{b\}$

- Exactly one element of X: potential witness.

Finding the True Witnesses

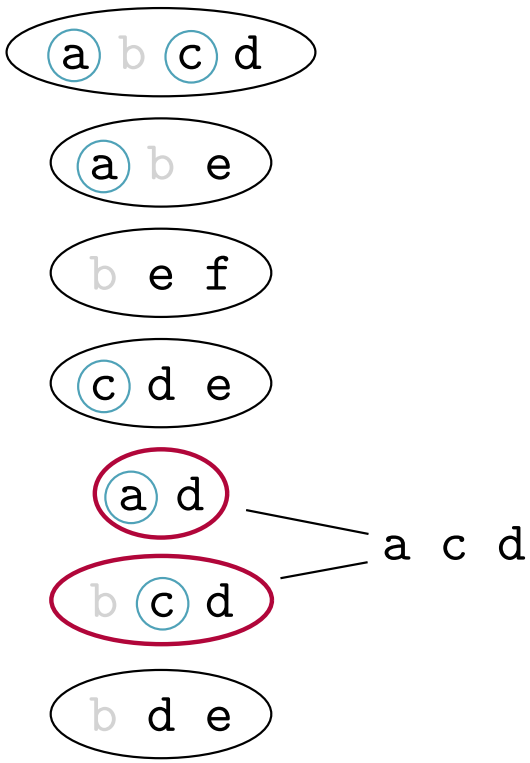


$X = \{a,c\}$

$Y = \{b\}$

- Exactly one element of X: potential witness.

Finding the True Witnesses

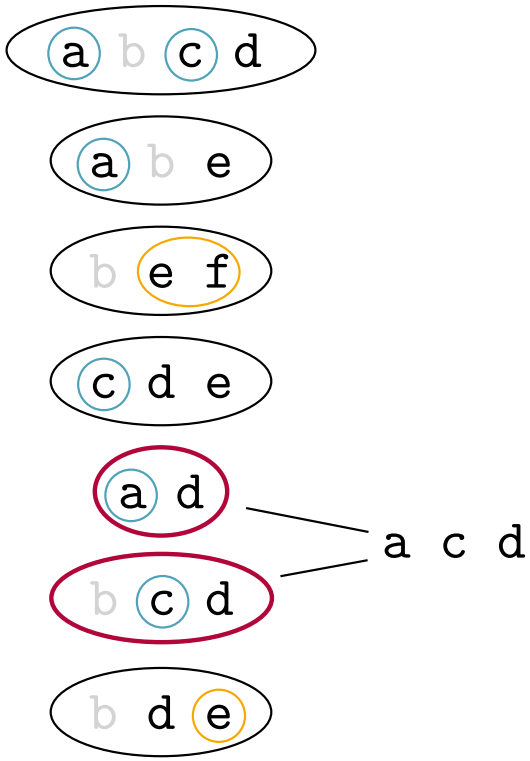


$X = \{a,c\}$

$Y = \{b\}$

- Exactly one element of X: potential witness.

Finding the True Witnesses

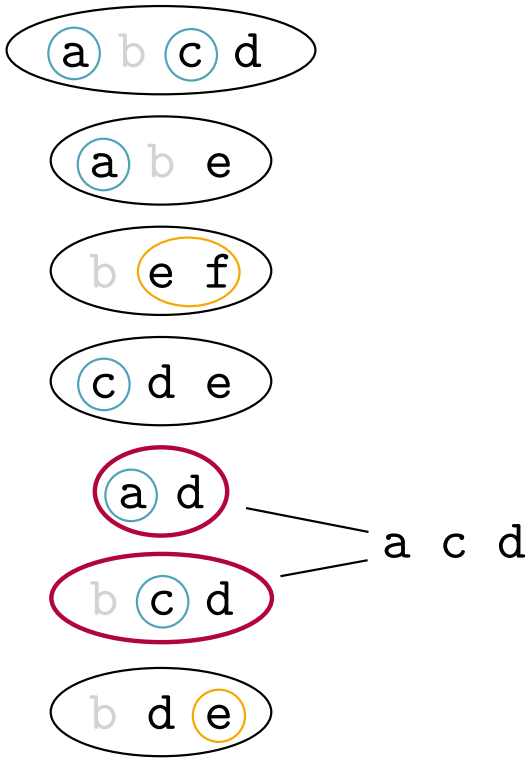


$X = \{a,c\}$

$Y = \{b\}$

- Exactly one element of X: potential witness.

Finding the True Witnesses

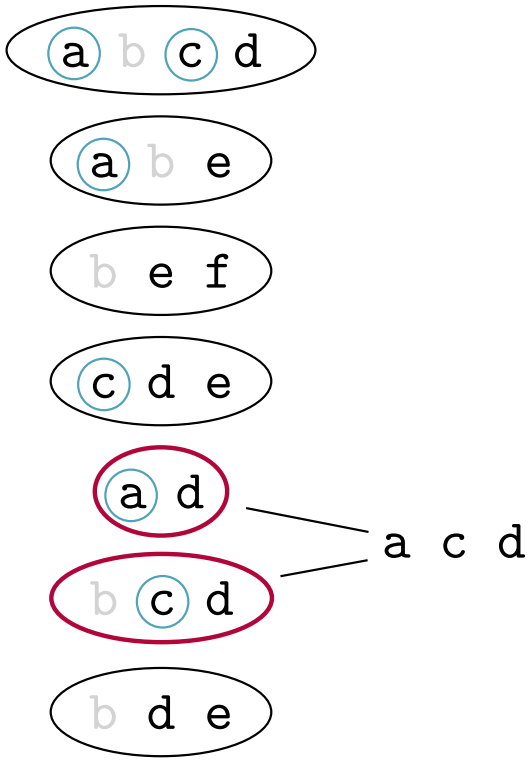


$X = \{\mathtt{a},\mathtt{c}\}$

$Y = \{\mathtt{b}\}$

- Exactly one element of X: potential witness.
- X is extendable iff there are potential witnesses $(E_x)_{x \in X}$ s.t. the union $\bigcup_{x \in X} E_x$ does not contain an unhit edge.

Finding the True Witnesses



$X = \{\mathtt{a},\mathtt{c}\}$

$Y = \{\mathtt{b}\}$

$\mathcal{W}_\mathtt{a} = \{\{\mathtt{a},\mathtt{e}\},\{\mathtt{a},\mathtt{d}\}\}$

$\mathcal{W}_\mathtt{c} = \{\{\mathtt{c},\mathtt{d},\mathtt{e}\},\{\mathtt{c},\mathtt{d}\}\}$

$\mathcal{U} = \{\{\mathtt{e},\mathtt{f}\},\{\mathtt{d},\mathtt{e}\}\}$

- Exactly one element of X: potential witness.
- X is extendable iff there are potential witnesses $(E_x)_{x\in X}$ s.t. the union $\bigcup_{x\in X} E_x$ does not contain an unhit edge.

The Extension Oracle

```
1  if X = ∅ then
2      if V\Y is a hitting set then return EXTEND.;
3      else return NOT EXTEND.;
4  initialise set system 𝒰 = ∅;
5  foreach x ∈ X do initialise set system 𝒲_x = ∅;
6  foreach edge E do
7      if E ∩ X = {x} then add E\Y to 𝒲_x;
8      if E ∩ X = ∅ then add E\Y to 𝒰;
9  if ∃x ∈ X : 𝒲_x = ∅ then return NOT EXTEND.;
10 if 𝒰 = ∅ then return MINIMAL;
11 foreach (E_{x_1}, ..., E_{x_|X|}) ∈ 𝒲_{x_1} × ··· × 𝒲_{x_|X|} do
12     W ← ⋃_{i=1}^{|X|} E_{x_i};
13     if ∀U ∈ 𝒰 : U ⊈ W then return EXTEND.;
14 return NOT EXTEND.;
```

The Extension Oracle

```
1  if X = ∅ then
2      if V\Y is a hitting set then return EXTEND.;
3      else return NOT EXTEND.;
4  initialise set system 𝒰 = ∅;
5  foreach x ∈ X do initialise set system 𝒲_x = ∅;
6  foreach edge E do
7      if E ∩ X = {x} then add E\Y to 𝒲_x;
8      if E ∩ X = ∅ then add E\Y to 𝒰;
9  if ∃x ∈ X : 𝒲_x = ∅ then return NOT EXTEND.;
10 if 𝒰 = ∅ then return MINIMAL;
11 foreach (E_{x_1}, ..., E_{x_|X|}) ∈ 𝒲_{x_1} × ··· × 𝒲_{x_|X|} do
12     W ← ⋃_{i=1}^{|X|} E_{x_i};
13     if ∀U ∈ 𝒰 : U ⊈ W then return EXTEND.;
14 return NOT EXTEND.;
```

The Extension Oracle

```
1  if X = ∅ then
2      if V\Y is a hitting set then return EXTEND.;
3      else return NOT EXTEND.;
4  initialise set system 𝒰 = ∅;
5  foreach x ∈ X do initialise set system 𝒲_x = ∅;
6  foreach edge E do
7      if E ∩ X = {x} then add E\Y to 𝒲_x;
8      if E ∩ X = ∅ then add E\Y to 𝒰;
9  if ∃x ∈ X : 𝒲_x = ∅ then return NOT EXTEND.;
10 if 𝒰 = ∅ then return MINIMAL;
11 foreach (E_{x_1}, ..., E_{x_|X|}) ∈ 𝒲_{x_1} × ··· × 𝒲_{x_|X|} do
12     W ← ⋃_{i=1}^{|X|} E_{x_i};
13     if ∀U ∈ 𝒰 : U ⊈ W then return EXTEND.;
14 return NOT EXTEND.;
```

The Extension Oracle

```
1  if X = ∅ then
2      if V\Y is a hitting set then return EXTEND.;
3      else return NOT EXTEND.;
4  initialise set system 𝒰 = ∅;
5  foreach x ∈ X do initialise set system 𝒲_x = ∅;
6  foreach edge E do
7      if E ∩ X = {x} then add E\Y to 𝒲_x;
8      if E ∩ X = ∅ then add E\Y to 𝒰;
9  if ∃x ∈ X : 𝒲_x = ∅ then return NOT EXTEND.;
10 if 𝒰 = ∅ then return MINIMAL;
11 foreach (E_{x_1}, ..., E_{x_|X|}) ∈ 𝒲_{x_1} × ··· × 𝒲_{x_|X|} do
12     W ← ⋃_{i=1}^{|X|} E_{x_i};
13     if ∀U ∈ 𝒰 : U ⊈ W then return EXTEND.;
14 return NOT EXTEND.;
```

The Extension Oracle

```
1  if X = ∅ then
2      if V\Y is a hitting set then return EXTEND.;
3      else return NOT EXTEND.;
4  initialise set system 𝒰 = ∅;
5  foreach x ∈ X do initialise set system 𝒲_x = ∅;
6  foreach edge E do
7      if E ∩ X = {x} then add E\Y to 𝒲_x;
8      if E ∩ X = ∅ then add E\Y to 𝒰;
9  if ∃x ∈ X : 𝒲_x = ∅ then return NOT EXTEND.;
10 if 𝒰 = ∅ then return MINIMAL;
11 foreach (E_{x_1}, ..., E_{x_|X|}) ∈ 𝒲_{x_1} × ··· × 𝒲_{x_|X|} do
12     W ← ⋃_{i=1}^{|X|} E_{x_i};
13     if ∀U ∈ 𝒰 : U ⊄ W then return EXTEND.;
14 return NOT EXTEND.;
```

From Run Time...

- Dominant brute-force phase: $O(m^{|X|} \cdot mn)$.

From Run Time...

- Dominant brute-force phase: $O(m^{|X|} \cdot mn)$.
- Matches conditional lower bound.

From Run Time...

- Dominant brute-force phase: $O(m^{|X|} \cdot mn)$.
- Matches conditional lower bound.

...to Delay

Claim: Largest solution has constant size k^* $\Rightarrow$ polynomial delay.

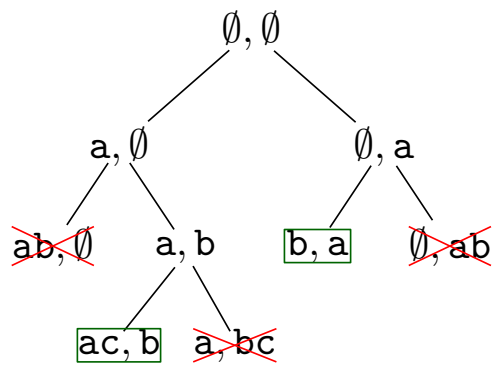
From Run Time...

- Dominant brute-force phase: $O(m^{|X|} \cdot mn)$.
- Matches conditional lower bound.

...to Delay

Claim: Largest solution has constant size k^* $\Rightarrow$ polynomial delay.

- Maximum distance between leaves in $O(n)$.



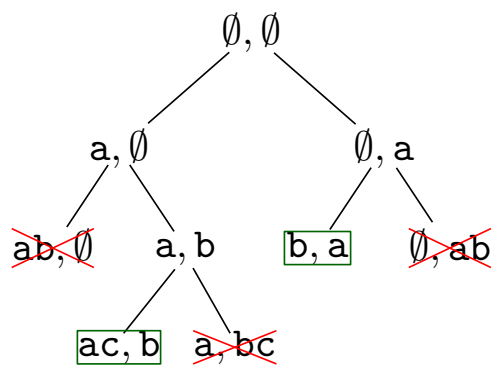
From Run Time...

- Dominant brute-force phase: $O(m^{|X|} \cdot mn)$.
- Matches conditional lower bound.

...to Delay

Claim: Largest solution has constant size k^* $\Rightarrow$ polynomial delay.

- Maximum distance between leaves in $O(n)$.
- If $|X| \geq k^*$, oracle answer is either NOT EXTENDABLE or MINIMAL.



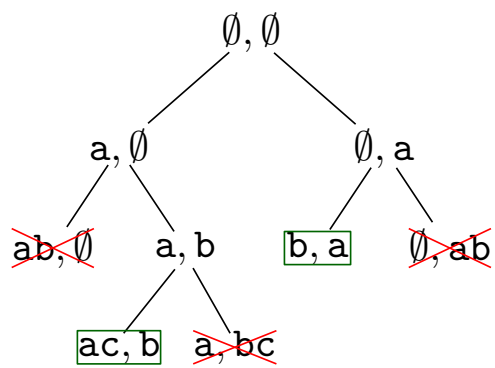
From Run Time...

- Dominant brute-force phase: $O(m^{|X|} \cdot mn)$.
- Matches conditional lower bound.

...to Delay

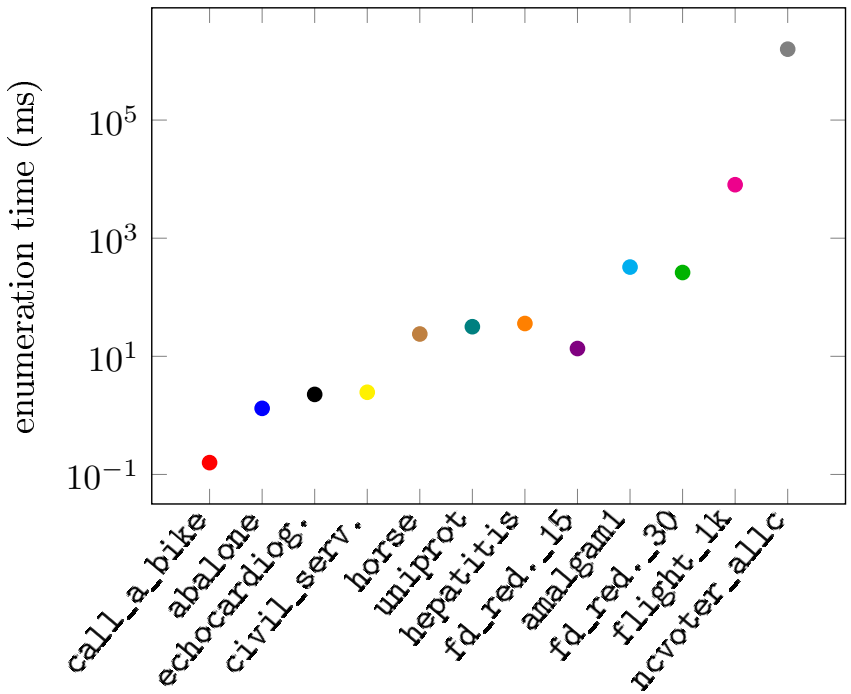
Claim: Largest solution has constant size $k^* \Rightarrow$ polynomial delay.

- Maximum distance between leaves in $O(n)$.
- If $|X| \geq k^*$, oracle answer is either NOT EXTENDABLE or MINIMAL.
- Maximum delay of $O(n) \cdot O(m^{k^*+1}n)$.



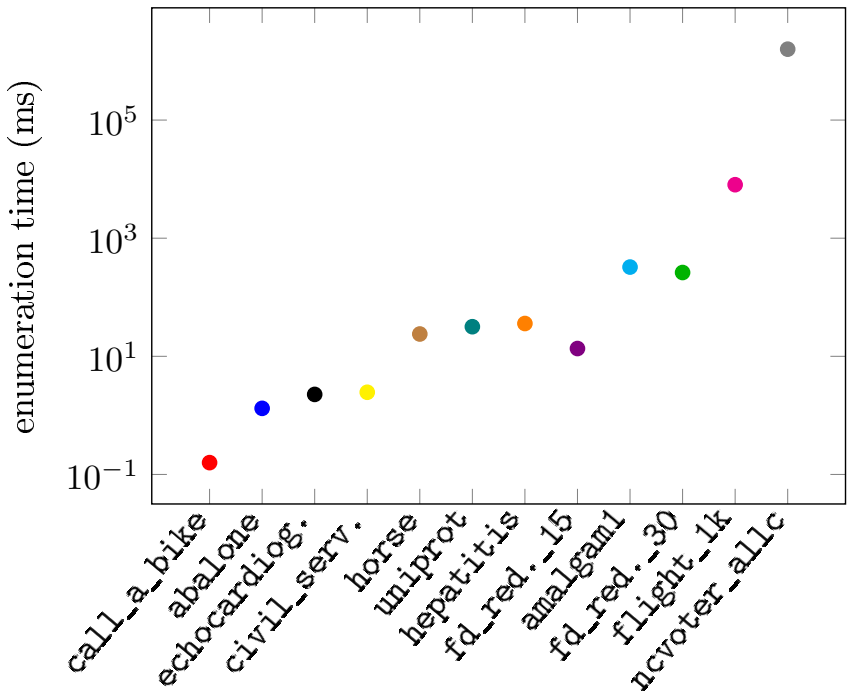
Theory and Practice

Setup: 10+2 databases on 2x 2.60GHz CPUs & 256GB RAM.



Theory and Practice

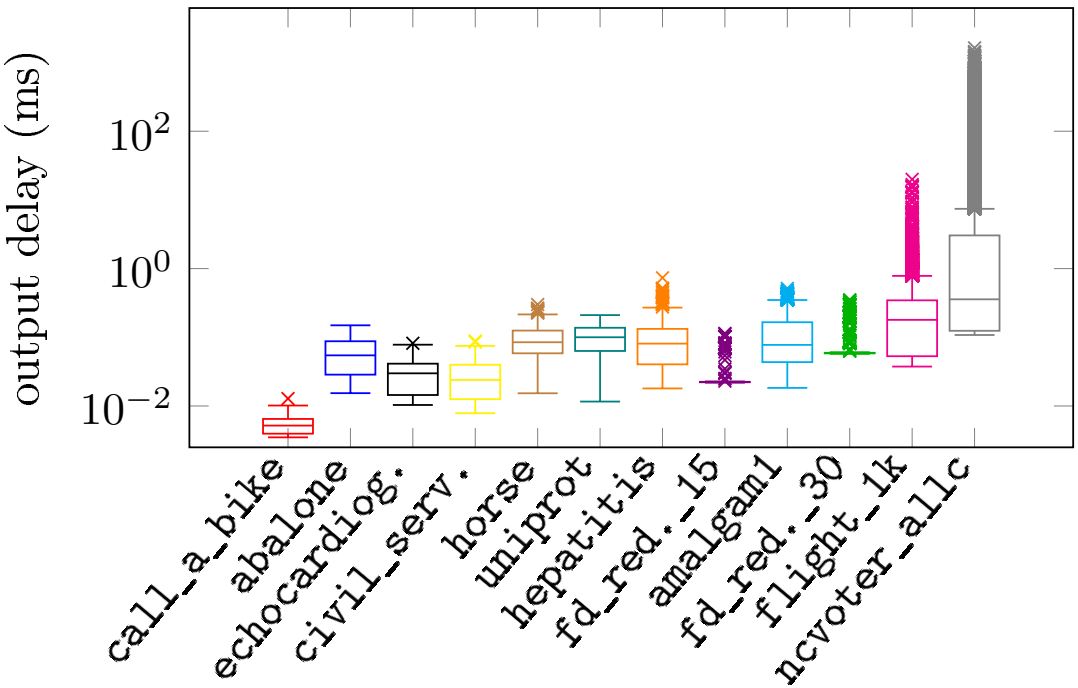
Setup: 10+2 databases on 2x 2.60GHz CPUs & 256GB RAM.



- 23 to 200k solutions = enumeration times 0.25ms to 27min.

Theory and Practice

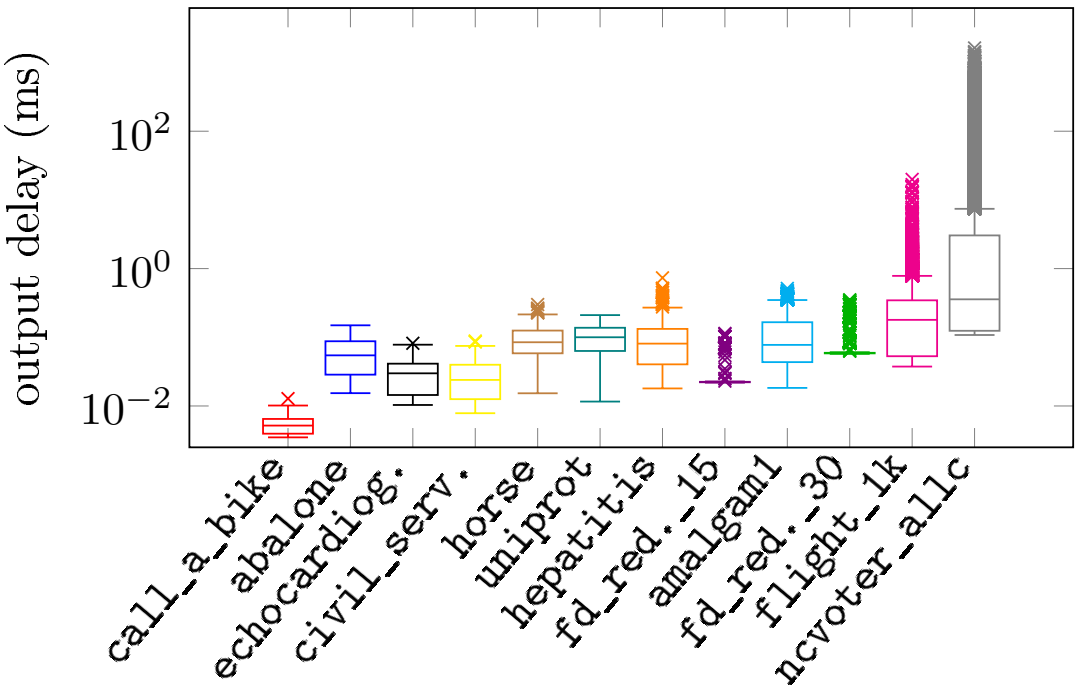
Setup: 10+2 databases on 2x 2.60GHz CPUs & 256GB RAM.



- 23 to 200k solutions = enumeration times 0.25ms to 27min.

Theory and Practice

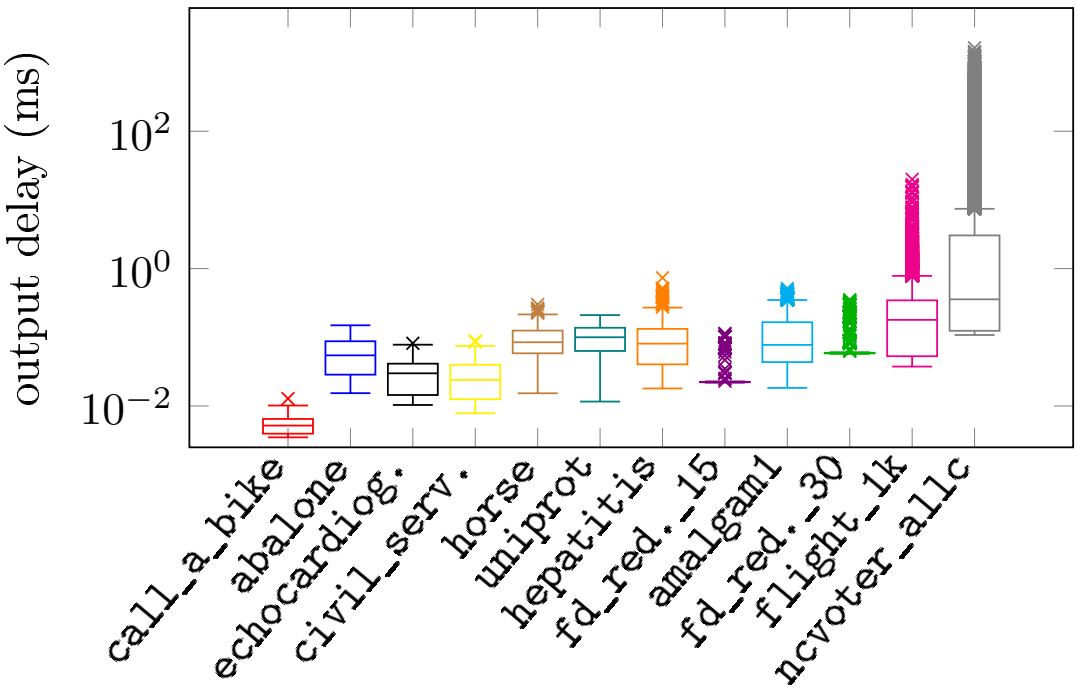
Setup: 10+2 databases on 2x 2.60GHz CPUs & 256GB RAM.



- 23 to 200k solutions = enumeration times 0.25ms to 27min.
- `ncvoter_allc` (88 cols., 100k rows): $n = 82$, $m = 448$, $k^* = 15$.

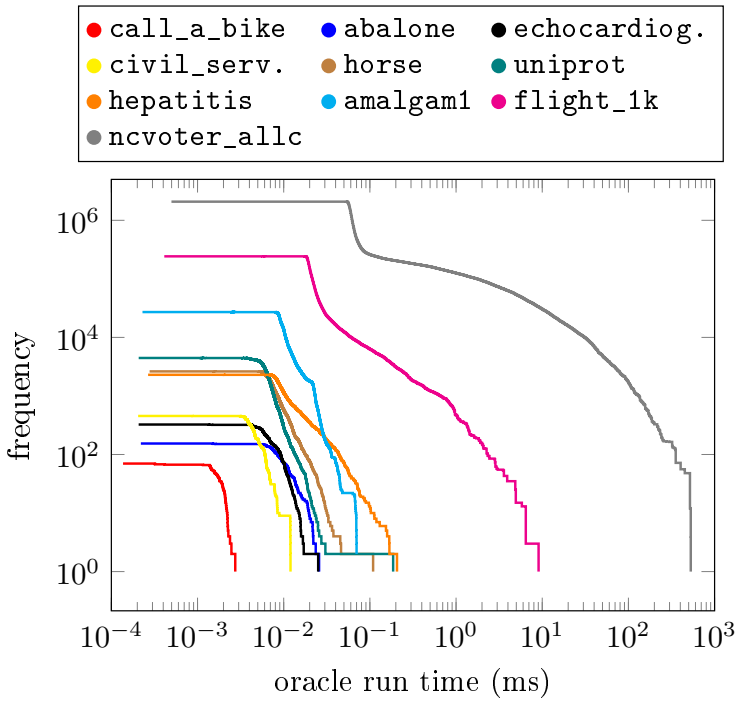
Theory and Practice

Setup: 10+2 databases on 2x 2.60GHz CPUs & 256GB RAM.

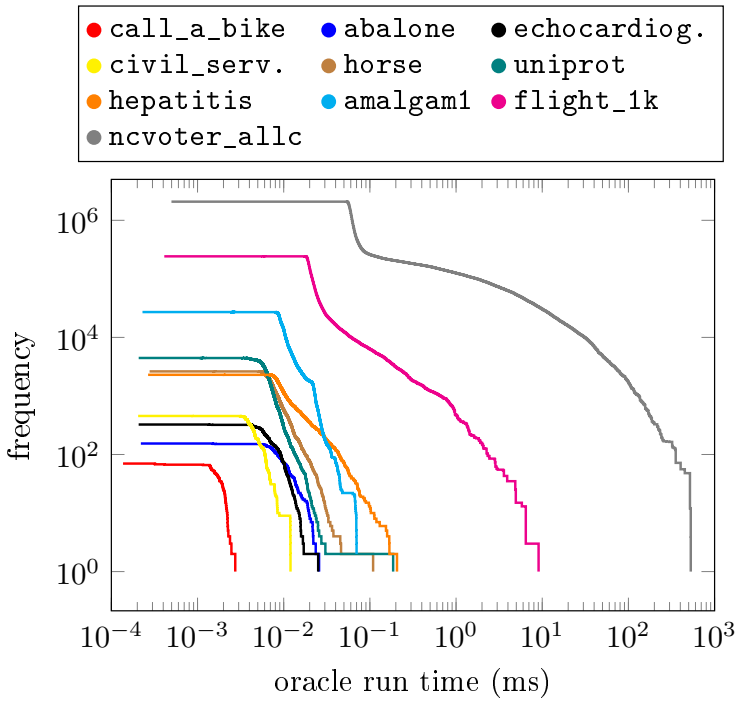


- 23 to 200k solutions = enumeration times 0.25ms to 27min.
- `ncvoter_allc` (88 cols., 100k rows): $n = 82$, $m = 448$, $k^* = 15$.
  - Maximum delay of 1.6s, but median at 0.35ms.

Oracle Run Times

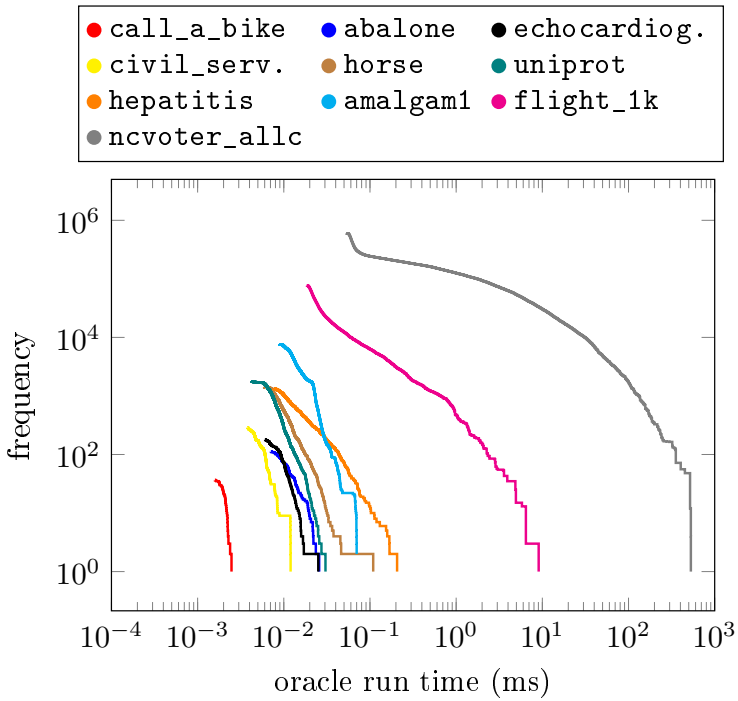


Oracle Run Times



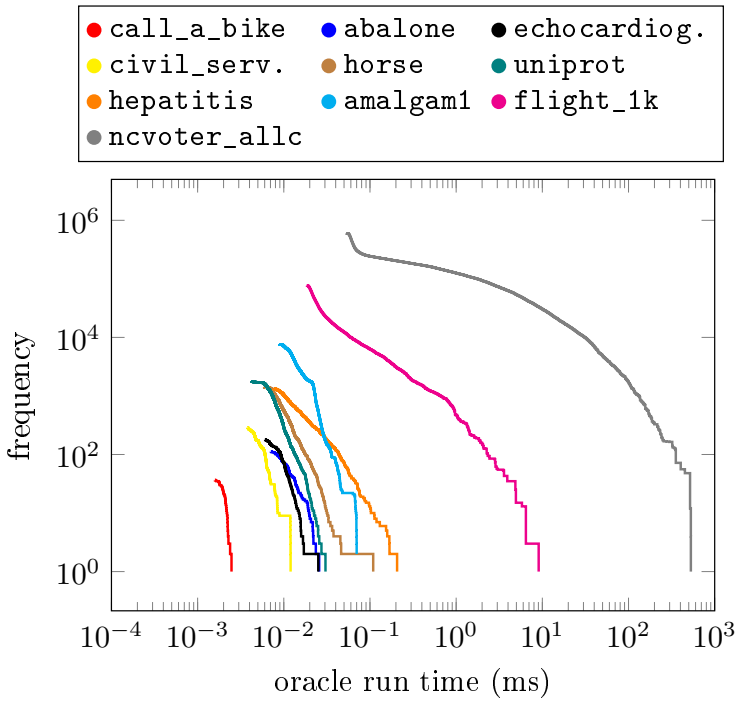
- Half the calls are trivial ($X = \emptyset$) or easy ($\mathcal{W}_x = \emptyset$ or $\mathcal{U} = \emptyset$).

Oracle Run Times



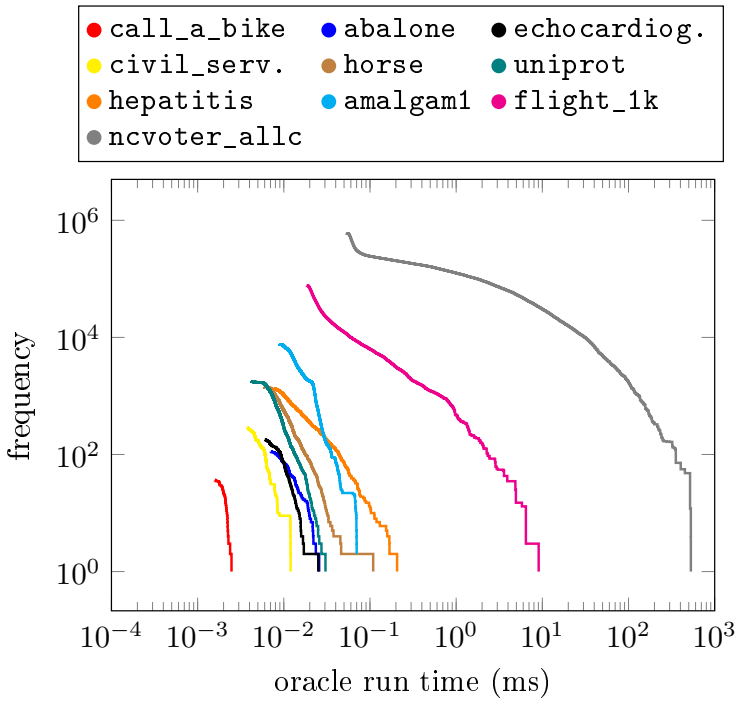
- Half the calls are trivial ($X = \emptyset$) or easy ($\mathcal{W}_x = \emptyset$ or $\mathcal{U} = \emptyset$).

Oracle Run Times



- Half the calls are trivial ($X = \emptyset$) or easy ($\mathcal{W}_x = \emptyset$ or $\mathcal{U} = \emptyset$).
- Brute-force calls exhibit power-law behavior.

Oracle Run Times



- Half the calls are trivial ($X = \emptyset$) or easy ($\mathcal{W}_x = \emptyset$ or $\mathcal{U} = \emptyset$).
- Brute-force calls exhibit power-law behavior.

Practice: The algorithm rarely hits the worst case.

Conclusion

1. Hitting set enumeration with polynomial delay is possible if the largest solution has constant size.
2. The extension oracle is a natural W[3]-complete problem.
3. Enumeration is fast on hypergraphs arising in data profiling.

Conclusion

1. Hitting set enumeration with polynomial delay is possible if the largest solution has constant size.
2. The extension oracle is a natural W[3]-complete problem.
3. Enumeration is fast on hypergraphs arising in data profiling.

Future Work

- Preprocessing seems to be the real bottleneck.

Conclusion

1. Hitting set enumeration with polynomial delay is possible if the largest solution has constant size.
2. The extension oracle is a natural W[3]-complete problem.
3. Enumeration is fast on hypergraphs arising in data profiling.

Future Work

- Preprocessing seems to be the real bottleneck.
- Understand the structure of difference sets.

Conclusion

1. Hitting set enumeration with polynomial delay is possible if the largest solution has constant size.
2. The extension oracle is a natural W[3]-complete problem.
3. Enumeration is fast on hypergraphs arising in data profiling.

Future Work

- Preprocessing seems to be the real bottleneck.
- Understand the structure of difference sets.

Thank you.