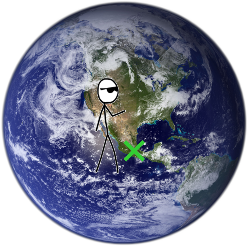
Algorithms for the Metric Dimension problem on directed graphs

Antoine Dailly, Florent Foucaud, Anni Hakanen
LIMOS, Clermont-Ferrand

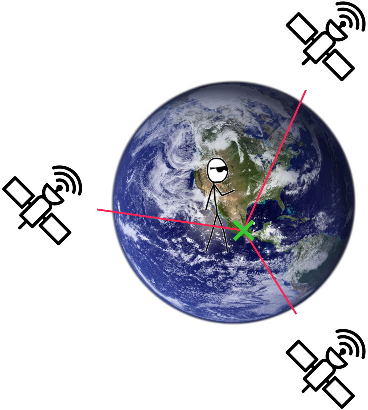
Séminaire Algo du GREYC
February 28, 2023

Where does it come from?

Where does it come from?

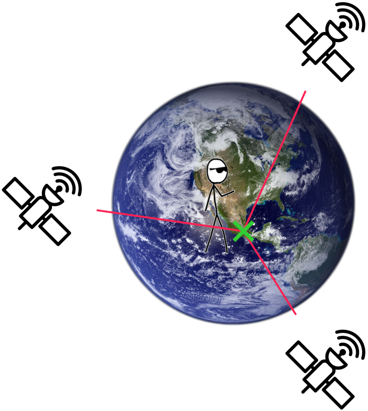


Where does it come from?



GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites for position

Where does it come from?



GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites for position

Question

How can we transpose this approach to graphs?

Metric Dimension

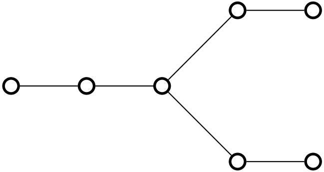
Definition

b resolves u and v if $\operatorname{dist}(b,u) \neq \operatorname{dist}(b,v)$

Metric Dimension

Definition

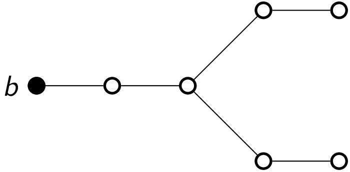
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

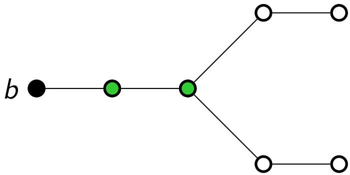
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

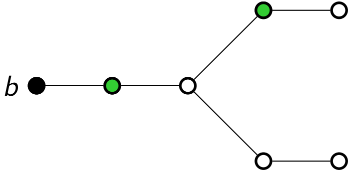
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

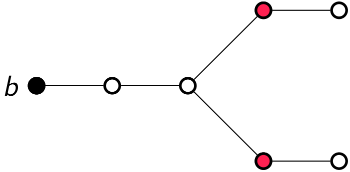
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

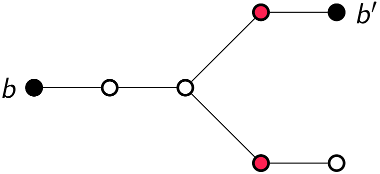
b resolves u and v if $\mathrm{dist}(b, u) \neq \mathrm{dist}(b, v)$



Metric Dimension

Definition

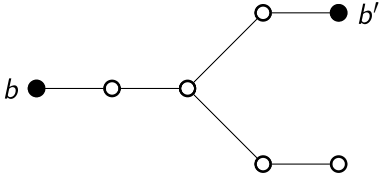
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



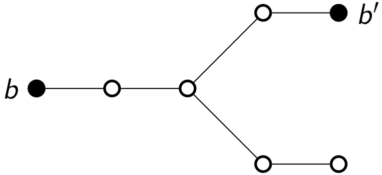
Resolving Set [Slater, 1975] [Harary & Melter, 1976]

$R \subseteq V(G)$ is a resolving set of G iff for every pair $\{u,v\}$, there is $b \in R$ that resolves u and v

Metric Dimension

Definition

b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Resolving Set [Slater, 1975] [Harary & Melter, 1976]

$R \subseteq V(G)$ is a resolving set of G iff for every pair $\{u,v\}$, there is $b \in R$ that resolves u and v

Metric Dimension

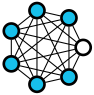
$\mathrm{MD}(G) =$ minimum size of a resolving set of G

Basic results

1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path 

Basic results

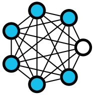
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



2. $\mathrm{MD}(G) = n-1 \Leftrightarrow G$ is K_n

Basic results

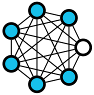
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



2. $\mathrm{MD}(G) = n - 1 \Leftrightarrow G$ is K_n
3. Trees?

Basic results

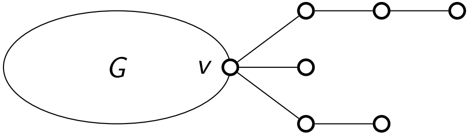
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



2. $\mathrm{MD}(G) = n - 1 \Leftrightarrow G$ is K_n
3. Trees?

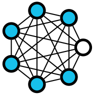
Legs

Paths with degree 2 inner vertices, and degree 1 and ≥ 3 end-points.



Basic results

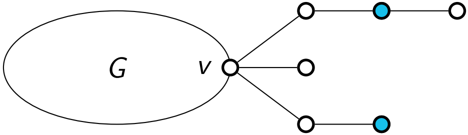
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



2. $\mathrm{MD}(G) = n - 1 \Leftrightarrow G$ is K_n
3. Trees?

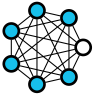
Legs

Paths with degree 2 inner vertices, and degree 1 and ≥ 3 end-points. If v has k legs, $k-1$ have ≥ 1 vertex in a resolving set.



Basic results

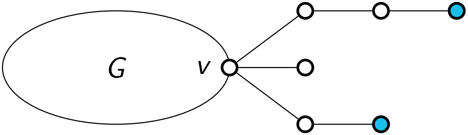
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



2. $\mathrm{MD}(G) = n - 1 \Leftrightarrow G$ is K_n
3. Trees?

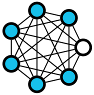
Legs

Paths with degree 2 inner vertices, and degree 1 and ≥ 3 endpoints. If v has k legs, $k-1$ have ≥ 1 vertex in a resolving set.



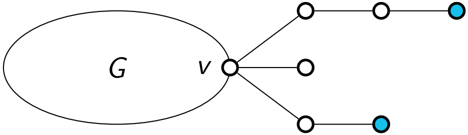
Simple leg rule: If v has $k \geq 2$ legs, select $k-1$ leg endpoints.

Basic results

1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path 
2. $\mathrm{MD}(G) = n-1 \Leftrightarrow G$ is K_n 
3. Trees? The simple leg rule gives an optimal resolving set [Slater, 1975]

Legs

Paths with degree 2 inner vertices, and degree 1 and ≥ 3 endpoints. If v has k legs, $k-1$ have ≥ 1 vertex in a resolving set.



Simple leg rule: If v has $k \geq 2$ legs, select $k-1$ leg endpoints.

Complexity results

Complexity results

Metric Dimension is difficult!

- NP-complete... [Garey & Johnson, 1979]

Complexity results

Metric Dimension is difficult!

- NP-complete... [Garey & Johnson, 1979]
- ... even on very simple graph classes: planar [Díaz *et al.*, 2012], split, bipartite [Epstein *et al.*, 2012], interval of diameter 2 [Foucaud *et al.*, 2017], bounded treewidth [Li & Pilipczuk, 2022]

Complexity results

Metric Dimension is difficult!

- NP-complete... [Garey & Johnson, 1979]
- ... even on very simple graph classes: planar [Díaz *et al.*, 2012], split, bipartite [Epstein *et al.*, 2012], interval of diameter 2 [Foucaud *et al.*, 2017], bounded treewidth [Li & Pilipczuk, 2022]
- W[2]-complete (so no $f(\mathrm{MD})n^k$ algorithm), even on subcubic graphs [Hartung & Nichterlein, 2013]

Complexity results

Metric Dimension is difficult!

- NP-complete... [Garey & Johnson, 1979]
- ... even on very simple graph classes: planar [Díaz *et al.*, 2012], split, bipartite [Epstein *et al.*, 2012], interval of diameter 2 [Foucaud *et al.*, 2017], bounded treewidth [Li & Pilipczuk, 2022]
- W[2]-complete (so no $f(\mathrm{MD})n^k$ algorithm), even on subcubic graphs [Hartung & Nichterlein, 2013]
- No polynomial-time algorithm can give better than a $\log(n)$ approximation factor, even on subcubic graphs [HN13]

Complexity results

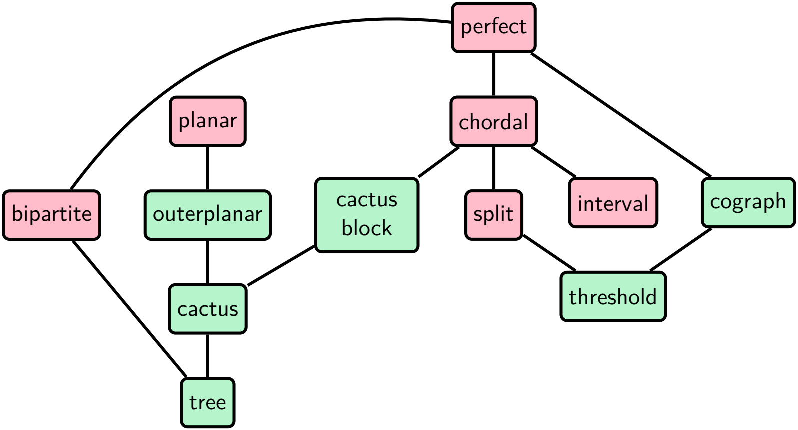
Metric Dimension is difficult!

- NP-complete... [Garey & Johnson, 1979]
- ... even on very simple graph classes: planar [Díaz *et al.*, 2012], split, bipartite [Epstein *et al.*, 2012], interval of diameter 2 [Foucaud *et al.*, 2017], bounded treewidth [Li & Pilipczuk, 2022]
- W[2]-complete (so no $f(\mathrm{MD})n^k$ algorithm), even on subcubic graphs [Hartung & Nichterlein, 2013]
- No polynomial-time algorithm can give better than a $\log(n)$ approximation factor, even on subcubic graphs [HN13]

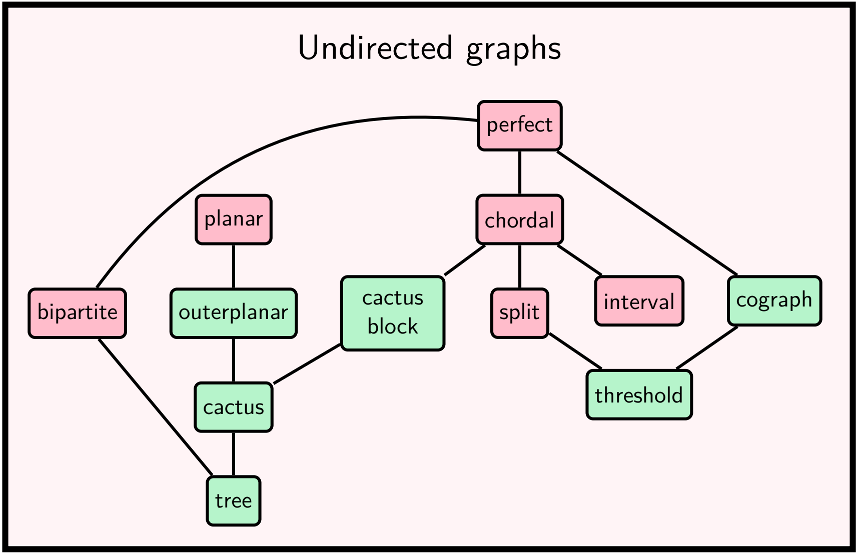
A few positive results...

- Linear-time: cographs [Epstein *et al.*, 2012], cactus block graphs [Hoffmann *et al.*, 2016]
- Polynomial-time: outerplanar graphs [Díaz *et al.*, 2012]
- FPT for bounded treelength [Belmonte *et al.*, 2015]

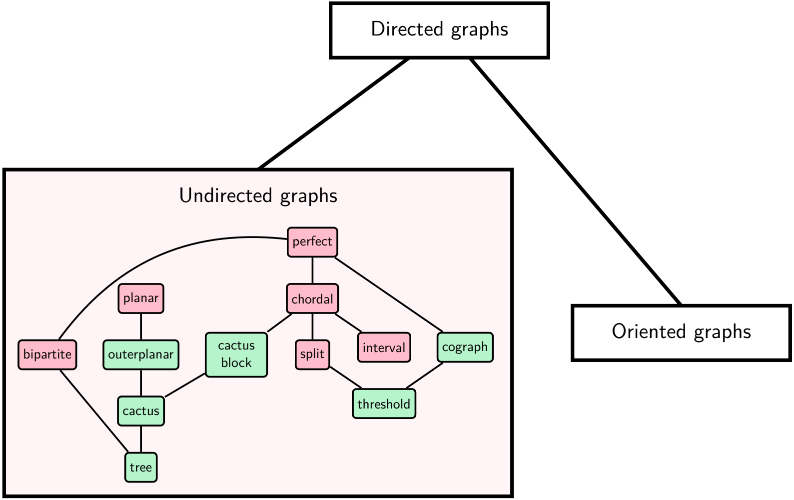
Inclusion diagram



Inclusion diagram

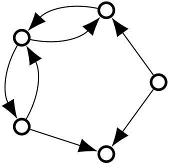


Inclusion diagram



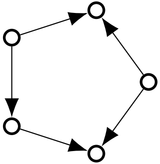
Definitions

- A directed graph may contain 2-cycles



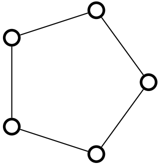
Definitions

- A directed graph may contain 2-cycles, an oriented graph cannot.



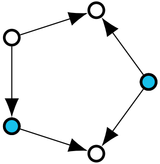
Definitions

- A directed graph may contain 2-cycles, an oriented graph cannot.
- If we remove the orientation, we obtain the underlying undirected graph



Definitions

- A directed graph may contain 2-cycles, an oriented graph cannot.
- If we remove the orientation, we obtain the underlying undirected graph

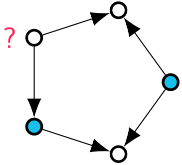


The definitions for Metric Dimension do not change:

- b resolves u and v if $\text{dist}(b,u) \neq \text{dist}(b,v)$
- $R \subseteq V(\overrightarrow{G})$ is a resolving set of G iff for every pair $\{u,v\}$, there is $b \in R$ that resolves u and v
- $\text{MD}(\overrightarrow{G})$ = minimum size of a resolving set of $\overrightarrow{G}$

Definitions

- A directed graph may contain 2-cycles, an oriented graph cannot.
- If we remove the orientation, we obtain the underlying undirected graph



The definitions for Metric Dimension do not change:

- b resolves u and v if $\text{dist}(b,u) \neq \text{dist}(b,v)$
- $R \subseteq V(\overrightarrow{G})$ is a resolving set of G iff for every pair $\{u,v\}$, there is $b \in R$ that resolves u and v
- $\text{MD}(\overrightarrow{G})$ = minimum size of a resolving set of $\overrightarrow{G}$

But there will be reachability problems!

Previous work

- Introduced in [Chartrand *et al.*, 2000] in a more constrained way: every vertex has to be reachable from **the whole** resolving set

Previous work

- Introduced in [Chartrand *et al.*, 2000] in a more constrained way: every vertex has to be reachable from **the whole** resolving set
- Study of $\mathrm{MD}(\overrightarrow{G})$ for "nice" oriented classes: Cayley digraphs [Fehr, 2006], tournaments [Lozano, 2013], orientations of wheels & fans [Pancahayani & Simanjuntak, 2014], De Brujin and Kautz digraphs [Rajan *et al.*, 2015]

Previous work

- Introduced in [Chartrand *et al.*, 2000] in a more constrained way: every vertex has to be reachable from **the whole** resolving set
- Study of MD($\overrightarrow{G}$) for "nice" oriented classes: Cayley digraphs [Fehr, 2006], tournaments [Lozano, 2013], orientations of wheels & fans [Pancahayani & Simanjuntak, 2014], De Brujin and Kautz digraphs [Rajan *et al.*, 2015]
- Relaxed definition we use introduced in [Araujo *et al.*, 2023+]: every vertex must be reachable from **some vertex** in the resolving set

Previous work

- Introduced in [Chartrand *et al.*, 2000] in a more constrained way: every vertex has to be reachable from **the whole** resolving set
- Study of MD($\overrightarrow{G}$) for "nice" oriented classes: Cayley digraphs [Fehr, 2006], tournaments [Lozano, 2013], orientations of wheels & fans [Pancahayani & Simanjuntak, 2014], De Brujin and Kautz digraphs [Rajan *et al.*, 2015]
- Relaxed definition we use introduced in [Araujo *et al.*, 2023+]: every vertex must be reachable from **some vertex** in the resolving set
- NP-complete on bipartite DAGs with maximum degree 8 and maximum distance 4 [Araujo *et al.*, 2023+]

Previous work

- Introduced in [Chartrand *et al.*, 2000] in a more constrained way: every vertex has to be reachable from **the whole** resolving set
- Study of $\text{MD}(\overrightarrow{G})$ for "nice" oriented classes: Cayley digraphs [Fehr, 2006], tournaments [Lozano, 2013], orientations of wheels & fans [Pancahayani & Simanjuntak, 2014], De Brujin and Kautz digraphs [Rajan *et al.*, 2015]
- Relaxed definition we use introduced in [Araujo *et al.*, 2023+]: every vertex must be reachable from **some vertex** in the resolving set
- NP-complete on bipartite DAGs with maximum degree 8 and maximum distance 4 [Araujo *et al.*, 2023+]
- Linear-time algorithm for orientations of trees [Araujo *et al.*, 2023+]

Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

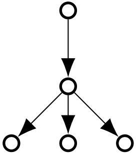
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in R



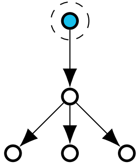
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in $R \Rightarrow$ Every source is in R



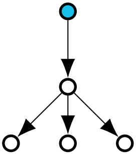
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in $R \Rightarrow$ Every source is in R
2. Resolving pairs of vertices



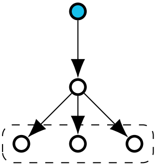
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in R $\Rightarrow$ Every source is in R
2. Resolving pairs of vertices $\Rightarrow$ For every set of k in-twins,



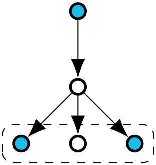
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in $R \Rightarrow$ Every source is in R
2. Resolving pairs of vertices $\Rightarrow$ For every set of k in-twins, $k-1$ of them are in R



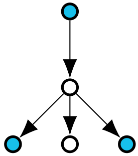
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in $R \Rightarrow$ Every source is in R
2. Resolving pairs of vertices $\Rightarrow$ For every set of k in-twins, $k-1$ of them are in R



Holds for all directed graphs!

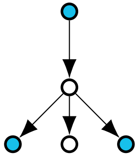
Algorithm for orientations of trees

Theorem [Araujo *et al.*, 2023+]

A minimum-size resolving set R of an orientation of a tree can be computed in linear time.

Proof

1. Every vertex must be reachable from at least one vertex in $R \Rightarrow$ Every source is in R
2. Resolving pairs of vertices $\Rightarrow$ For every set of k in-twins, $k-1$ of them are in R



Holds for all directed graphs!

3 The set R constructed this way is a resolving set

Our results

Theorem [D., Foucaud & Hakanen, 2023+]

Linear-time algorithms for minimum-size resolving sets of directed trees and orientations of unicyclic graphs.

Our results

Theorem [D., Foucaud & Hakanen, 2023+]

Linear-time algorithms for minimum-size resolving sets of directed trees and orientations of unicyclic graphs.

Theorem [D., Foucaud & Hakanen, 2023+]

NP-complete for planar triangle-free DAGs of maximum degree 6.

Our results

Theorem [D., Foucaud & Hakanen, 2023+]

Linear-time algorithms for minimum-size resolving sets of directed trees and orientations of unicyclic graphs.

Theorem [D., Foucaud & Hakanen, 2023+]

NP-complete for planar triangle-free DAGs of maximum degree 6.

Theorem [D., Foucaud & Hakanen, 2023+]

FPT algorithm parameterized by directed modular width.

Directed trees (1) Necessary vertices

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

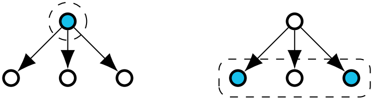
Directed trees (1) Necessary vertices

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm: two mandatory things

- Sources + resolving sets of in-twins



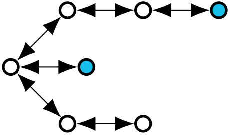
Directed trees (1) Necessary vertices

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm: two mandatory things

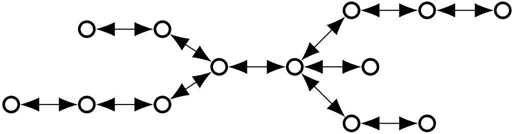
- Sources + resolving sets of in-twins
- Resolving legs of strongly connected components



Directed trees (2) Dummy vertices

Definition

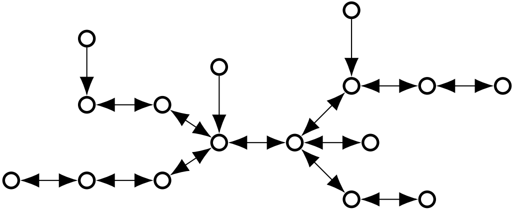
In a strongly connected component C



Directed trees (2) Dummy vertices

Definition

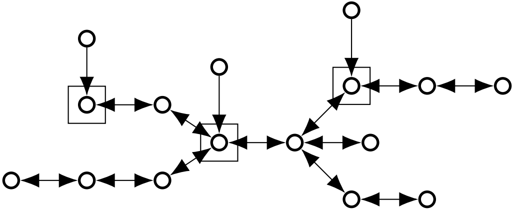
In a strongly connected component C, every $v \in C$ such that an arc $\overrightarrow{uv}$ with $u \notin C$ exists is a dummy vertex.



Directed trees (2) Dummy vertices

Definition

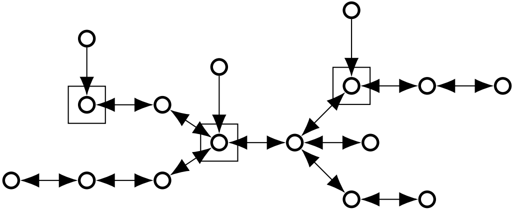
In a strongly connected component C, every $v \in C$ such that an arc $\overrightarrow{uv}$ with $u \notin C$ exists is a dummy vertex.



Directed trees (2) Dummy vertices

Definition

In a strongly connected component C, every $v \in C$ such that an arc $\overrightarrow{uv}$ with $u \notin C$ exists is a dummy vertex.

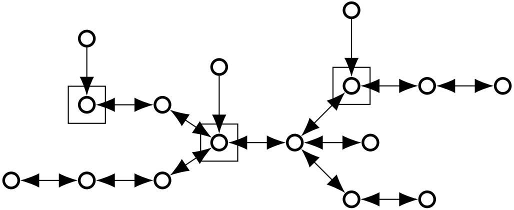


Every dummy vertex is a representative of the vertices in the resolving set behind the in-arc

Directed trees (2) Dummy vertices

Definition

In a strongly connected component C, every $v \in C$ such that an arc $\overrightarrow{uv}$ with $u \notin C$ exists is a dummy vertex.



Every dummy vertex is a representative of the vertices in the resolving set behind the in-arc

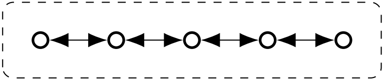
They act like degree ≥ 3 vertices for the purpose of legs

Directed trees (3) First problem: escalators

Definition

An escalator is a strongly connected component with:

- a path as an underlying graph

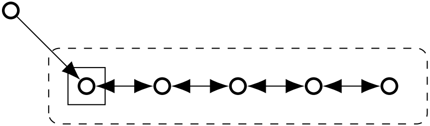


Directed trees (3) First problem: escalators

Definition

An escalator is a strongly connected component with:

- a path as an underlying graph
- only one in-arc from outside, at one end

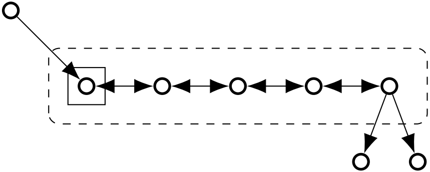


Directed trees (3) First problem: escalators

Definition

An escalator is a strongly connected component with:

- a path as an underlying graph
- only one in-arc from outside, at one end
- the only possible out-arcs to outside are at the other end

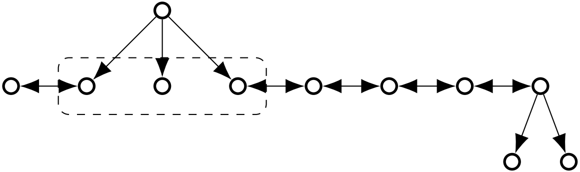


Directed trees (3) First problem: escalators

Definition

An escalator is a strongly connected component with:

- a path as an underlying graph
- only one in-arc from outside, at one end
- the only possible out-arcs to outside are at the other end



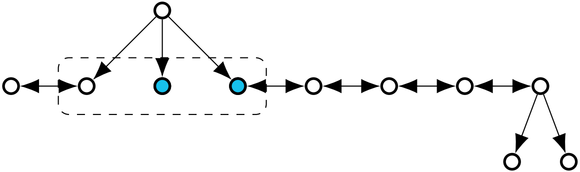
→ These are almost-in-twins

Directed trees (3) First problem: escalators

Definition

An escalator is a strongly connected component with:

- a path as an underlying graph
- only one in-arc from outside, at one end
- the only possible out-arcs to outside are at the other end



→ These are almost-in-twins

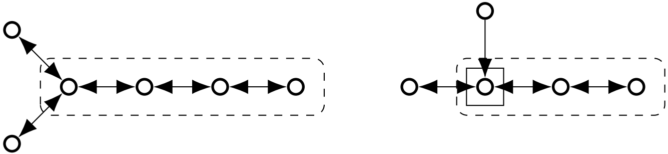
- For each set of k almost-in-twins, take $k-1$ in the resolving set

Directed trees (4) Second problem: special legs

Definition

In a strongly connected component, a special leg is a leg that:

- spans from a dummy or degree ≥ 3 (in the component) vertex

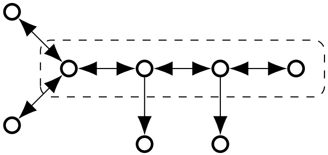


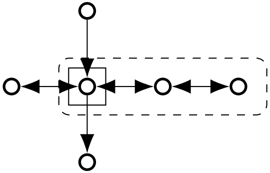
Directed trees (4) Second problem: special legs

Definition

In a strongly connected component, a special leg is a leg that:

- spans from a dummy or degree ≥ 3 (in the component) vertex
- has at least one out-arc from a vertex other than its endpoint



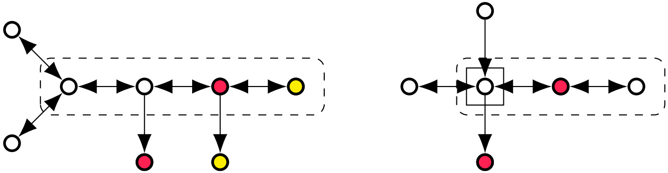


Directed trees (4) Second problem: special legs

Definition

In a strongly connected component, a special leg is a leg that:

- spans from a dummy or degree ≥ 3 (in the component) vertex
- has at least one out-arc from a vertex other than its endpoint



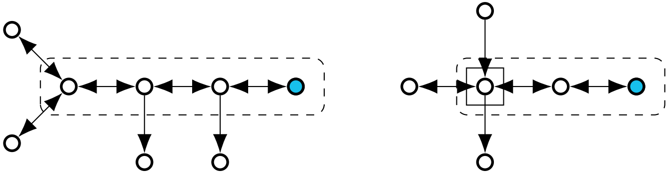
→ Conflict between pairs!

Directed trees (4) Second problem: special legs

Definition

In a strongly connected component, a special leg is a leg that:

- spans from a dummy or degree ≥ 3 (in the component) vertex
- has at least one out-arc from a vertex other than its endpoint

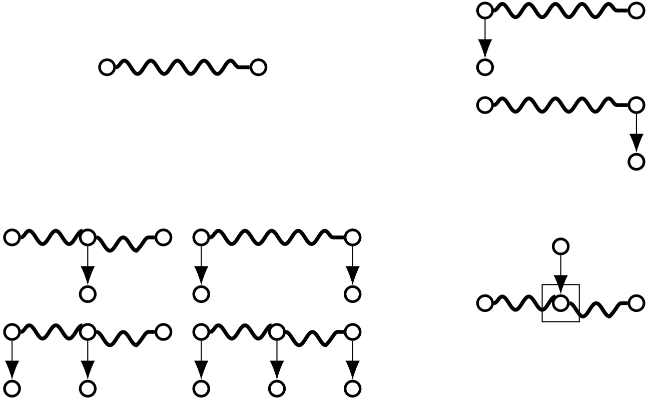


$\rightarrow$ Conflict between pairs!

- Take the endpoint of each special leg

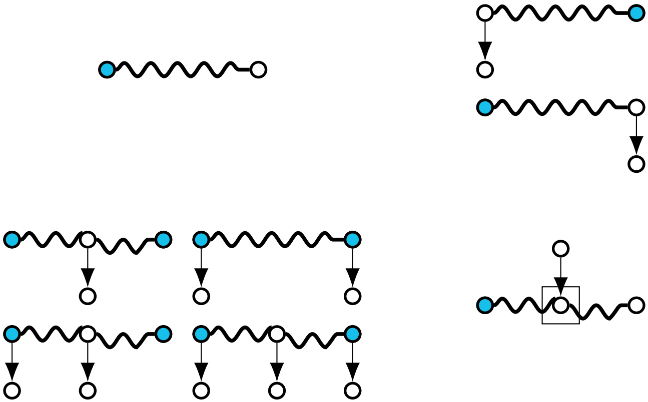
Directed trees (5) Third problem: some paths...

The strongly connected components whose underlying graph is a path (snake = any positive length) with the following patterns:



Directed trees (5) Third problem: some paths...

The strongly connected components whose underlying graph is a path (snake = any positive length) with the following patterns:



... require one or two endpoints.

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Mark the dummy vertices

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Mark the dummy vertices
   2.2 Solve the special paths cases (previous slide)

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
    2.1 Mark the dummy vertices
    2.2 Solve the special paths cases (previous slide)
    2.3 Take the endpoint of every special leg

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Mark the dummy vertices
   2.2 Solve the special paths cases (previous slide)
   2.3 Take the endpoint of every special leg
   2.4 Resolve the remaining standard legs

Directed trees (6) The final algorithm

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of a directed tree.

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Mark the dummy vertices
   2.2 Solve the special paths cases (previous slide)
   2.3 Take the endpoint of every special leg
   2.4 Resolve the remaining standard legs

This gives a resolving set... which we prove is minimum-size!

Orientations of unicyclic graphs

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of the orientation of a unicyclic graph.

Orientations of unicyclic graphs

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of the orientation of a unicyclic graph.

Algorithm

1. Take every source

3. Resolve each set of in-twins

Orientations of unicyclic graphs

Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of the orientation of a unicyclic graph.

Algorithm

1. Take every source

3. Resolve each set of in-twins with some priority

Orientations of unicyclic graphs

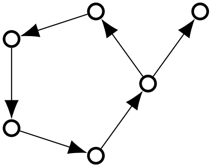
Theorem [D., Foucaud & Hakanen, 2023+]

There is a linear-time algorithm computing a minimum-size resolving set of the orientation of a unicyclic graph.

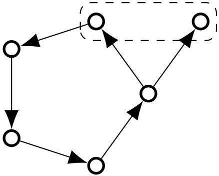
Algorithm

1. Take every source
2. Manage a few special cases (at most one more vertex)
3. Resolve each set of in-twins with some priority

No sink in the cycle

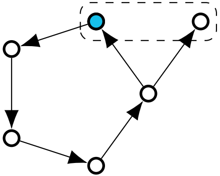


No sink in the cycle



Which in-twin?

No sink in the cycle

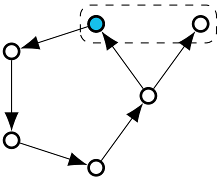


Which in-twin?

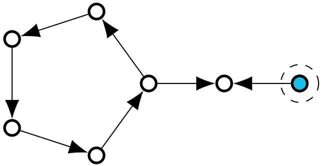
Priority

Give priority to in-twins in the cycle

No sink in the cycle



Which in-twin?

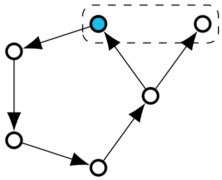


⚠ **Special case** (Reachability)

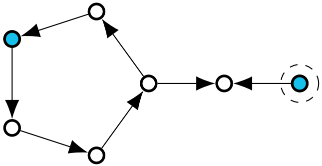
Priority

Give priority to in-twins in the cycle

No sink in the cycle



Which in-twin?

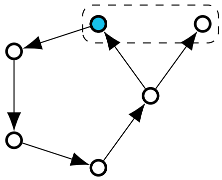


⚠**Special case** (Reachability)
$\Rightarrow$ Take one vertex from the cycle

Priority

Give priority to in-twins in the cycle

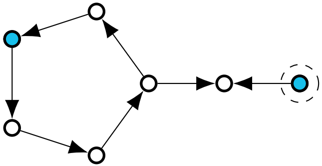
No sink in the cycle



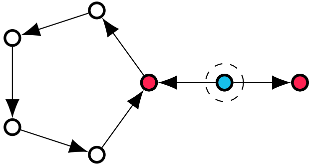
Which in-twin?

Priority

Give priority to in-twins in the cycle

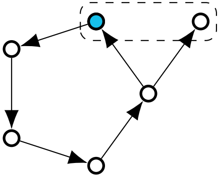


⚠ **Special case** (Reachability)
⇒ Take one vertex from the cycle



⚠ **Special case** (Unresolved pair)

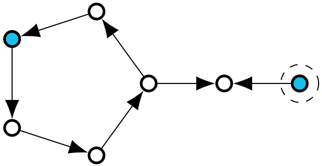
No sink in the cycle



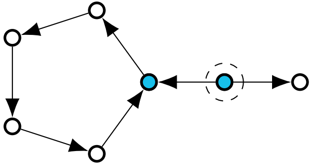
Which in-twin?

Priority

Give priority to in-twins in the cycle

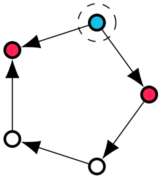


⚠**Special case** (Reachability)
⇒ Take one vertex from the cycle



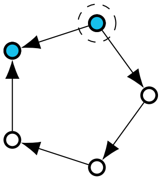
⚠**Special case** (Unresolved pair)
⇒ Take one unresolved vertex

One sink in the cycle



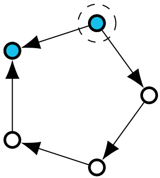
⚠**Special case**
(Unresolved pair)

One sink in the cycle

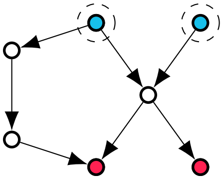


⚠**Special case**
(Unresolved pair)
⇒ Take one
unresolved vertex

One sink in the cycle

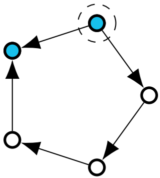


⚠**Special case**
(Unresolved pair)
⇒ Take one
unresolved vertex

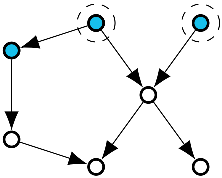


⚠**Special case**
(Unresolved pair)

One sink in the cycle

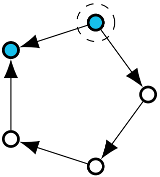


⚠**Special case**
(Unresolved pair)
⇒ Take one
unresolved vertex

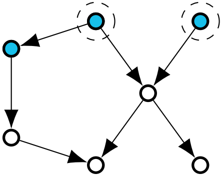


⚠**Special case**
(Unresolved pair)
⇒ Take one vertex
along the long path

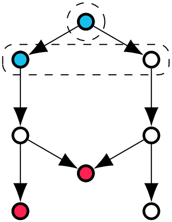
One sink in the cycle



⚠**Special case**
(Unresolved pair)
⇒ Take one
unresolved vertex

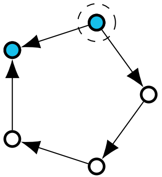


⚠**Special case**
(Unresolved pair)
⇒ Take one vertex
along the long path

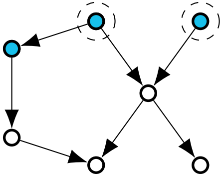


⚠**Special case**
(Unresolved pair)

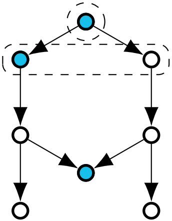
One sink in the cycle



⚠**Special case**
(Unresolved pair)
⇒ Take one
unresolved vertex

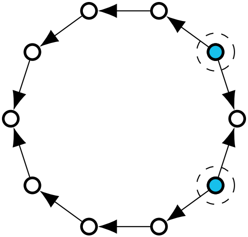


⚠**Special case**
(Unresolved pair)
⇒ Take one vertex
along the long path

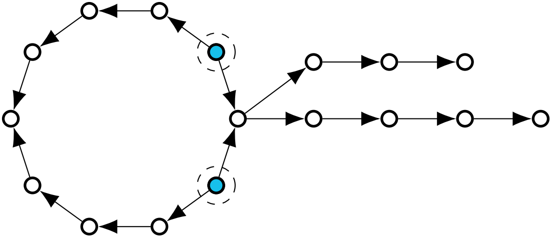


⚠**Special case**
(Unresolved pair)
⇒ Take the sink of
the cycle

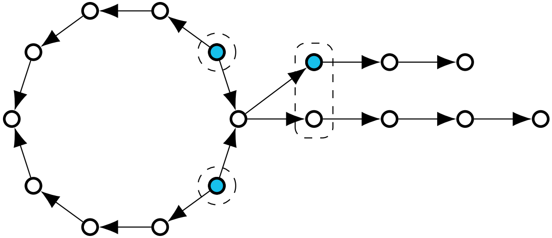
Two sinks in the cycle



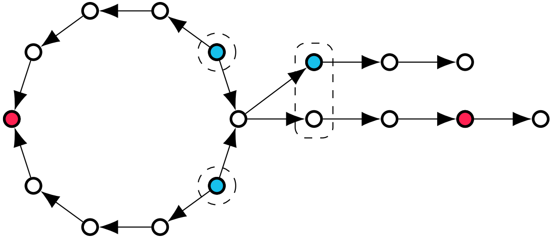
Two sinks in the cycle



Two sinks in the cycle

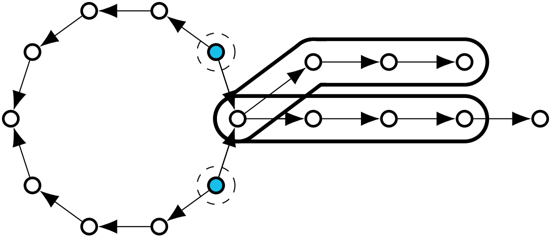


Two sinks in the cycle



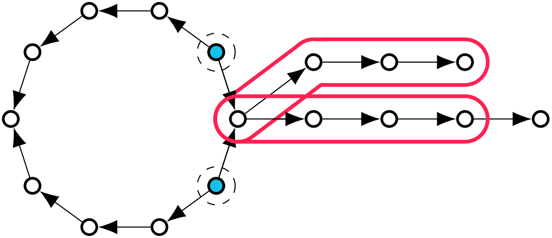
⚠**Special case** (Unresolved pair)

Two sinks in the cycle



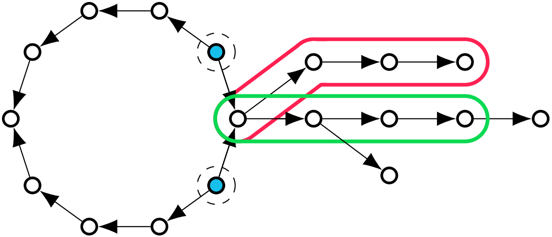
Those are **concerning paths**,

Two sinks in the cycle



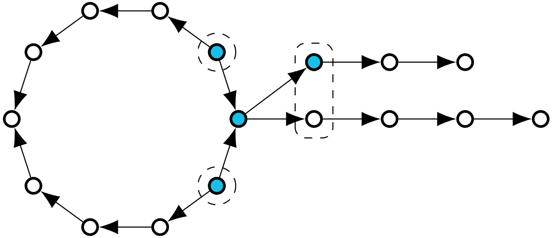
Those are **concerning paths**,
which can be either unfixable

Two sinks in the cycle



Those are **concerning paths**,
which can be either unfixable or fixable.

Two sinks in the cycle

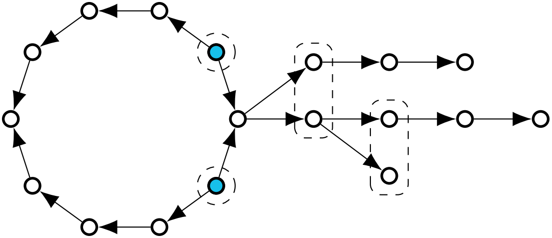


Those are **concerning paths**,
which can be either unfixable or fixable.

Special case & Priority

- If all the concerning paths are unfixable, then, take the sink

Two sinks in the cycle

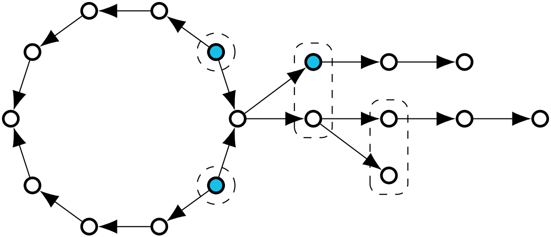


Those are **concerning paths**,
which can be either unfixable or fixable.

Special case & Priority

- If all the concerning paths are unfixable, then, take the sink
- Otherwise,

Two sinks in the cycle

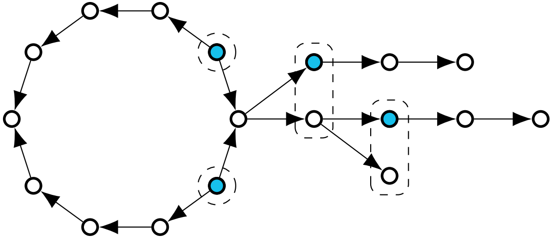


Those are **concerning paths**,
which can be either unfixable or fixable.

Special case & Priority

- If all the concerning paths are unfixable, then, take the sink
- Otherwise, **priority** to in-twins in unfixable paths

Two sinks in the cycle



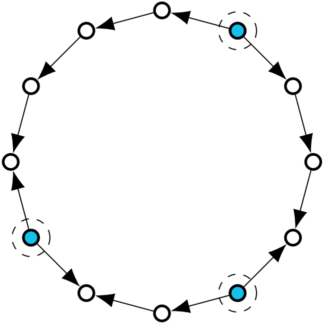
Those are **concerning paths**,
which can be either unfixable or fixable.

Special case & Priority

- If all the concerning paths are unfixable, then, take the sink
- Otherwise, **priority** to in-twins in unfixable paths, then concerning paths

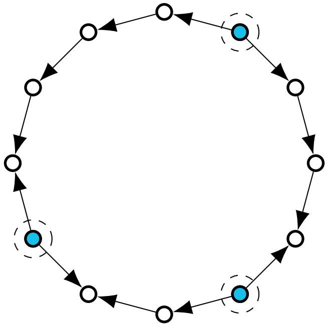
More than two sinks in the cycle

More than two sinks in the cycle



→ No problem!

More than two sinks in the cycle



→ No problem!

Linear-time algorithm

1. Take every source
2. Manage the special cases
3. Resolve each set of in-twins with some priority

NP-hardness (1) The gadgets

Theorem [D., Foucaud & Hakanen, 2023+]

DIRECTED METRIC DIMENSION is NP-complete for planar triangle-free DAGs of maximum degree 6.

NP-hardness (1) The gadgets

Theorem [D., Foucaud & Hakanen, 2023+]

Directed Metric Dimension is NP-complete for planar triangle-free DAGs of maximum degree 6.

Proof

Reduction from Vertex Cover on planar cubic biconnected undirected graphs [Mohar, 2001]

NP-hardness (1) The gadgets

Theorem [D., Foucaud & Hakanen, 2023+]

Directed Metric Dimension is NP-complete for planar triangle-free DAGs of maximum degree 6.

Proof

Reduction from Vertex Cover on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

NP-hardness (1) The gadgets

Theorem [D., Foucaud & Hakanen, 2023+]

DIRECTED METRIC DIMENSION is NP-complete for planar triangle-free DAGs of maximum degree 6.

Proof

Reduction from VERTEX COVER on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

If $uv \in$ perfect matching

NP-hardness (1) The gadgets

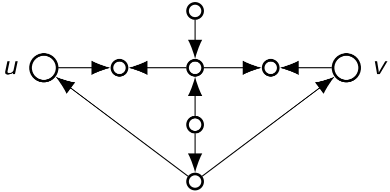
Theorem [D., Foucaud & Hakanen, 2023+]

Directed Metric Dimension is NP-complete for planar triangle-free DAGs of maximum degree 6.

Proof

Reduction from Vertex Cover on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

If $uv \in$ perfect matching



NP-hardness (1) The gadgets

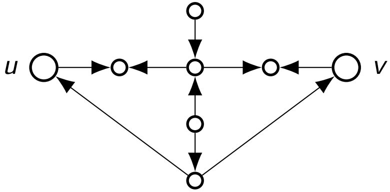
Theorem [D., Foucaud & Hakanen, 2023+]

Directed Metric Dimension is NP-complete for planar triangle-free DAGs of maximum degree 6.

Proof

Reduction from Vertex Cover on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

If $uv \in$ perfect matching



If $uv \notin$ perfect matching

NP-hardness (1) The gadgets

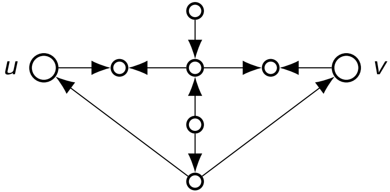
Theorem [D., Foucaud & Hakanen, 2023+]

Directed Metric Dimension is NP-complete for planar triangle-free DAGs of maximum degree 6.

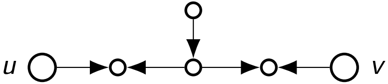
Proof

Reduction from Vertex Cover on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

If $uv \in$ perfect matching



If $uv \notin$ perfect matching



NP-hardness (1) The gadgets

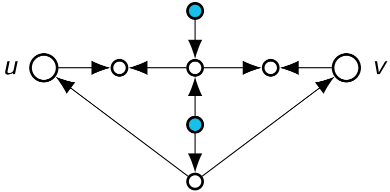
Theorem [D., Foucaud & Hakanen, 2023+]

DIRECTED METRIC DIMENSION is NP-complete for planar triangle-free DAGs of maximum degree 6.

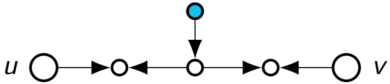
Proof

Reduction from VERTEX COVER on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

If $uv \in$ perfect matching



If $uv \notin$ perfect matching



NP-hardness (1) The gadgets

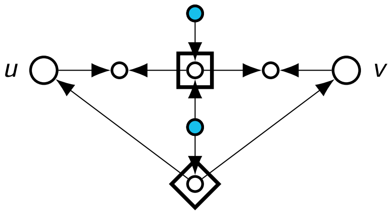
Theorem [D., Foucaud & Hakanen, 2023+]

DIRECTED METRIC DIMENSION is NP-complete for planar triangle-free DAGs of maximum degree 6.

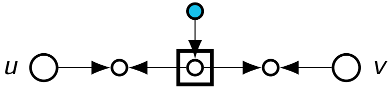
Proof

Reduction from VERTEX COVER on planar cubic biconnected undirected graphs [Mohar, 2001]
In particular, such graphs have a perfect matching [Petersen, 1891].

If $uv \in$ perfect matching

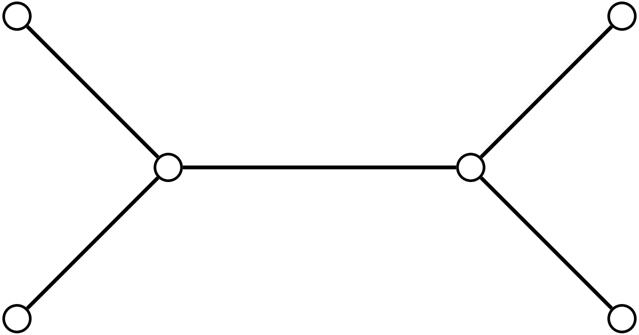


If $uv \notin$ perfect matching



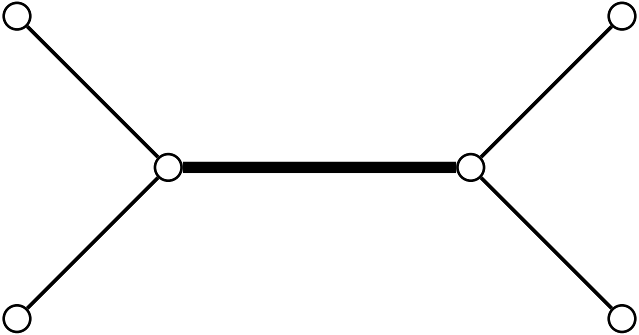
NP-hardness (2) Combining gadgets

We start from a planar cubic graph



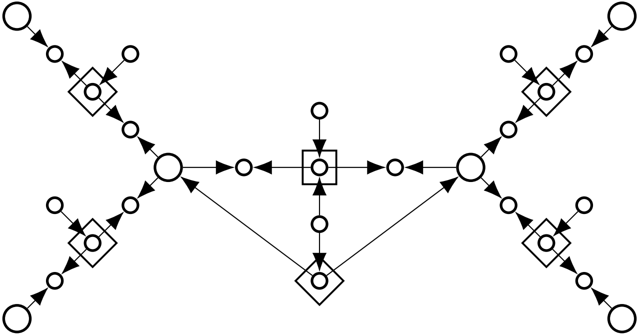
NP-hardness (2) Combining gadgets

We start from a planar cubic graph and a perfect matching



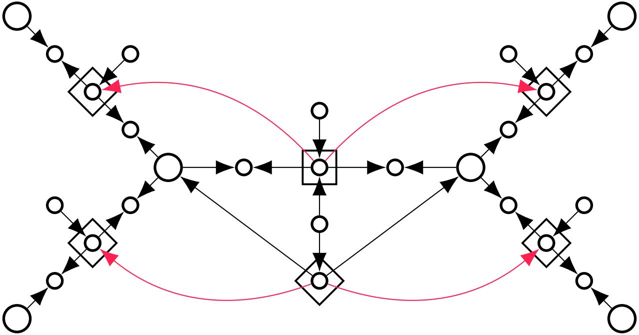
NP-hardness (2) Combining gadgets

We start from a planar cubic graph and a perfect matching



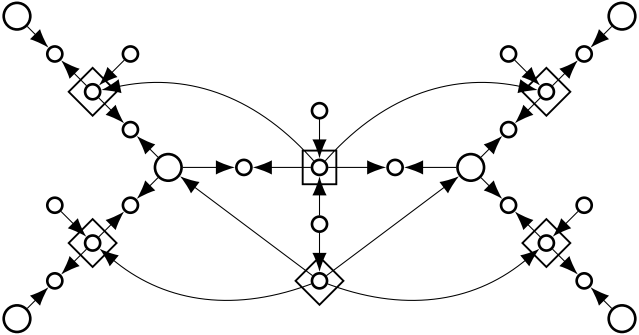
NP-hardness (2) Combining gadgets

We start from a planar cubic graph and a perfect matching



NP-hardness (2) Combining gadgets

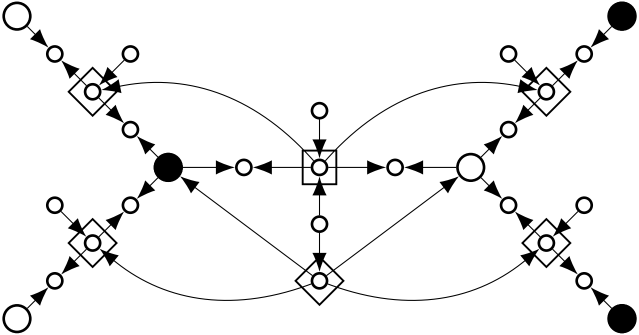
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftrightarrow$ MD $\leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

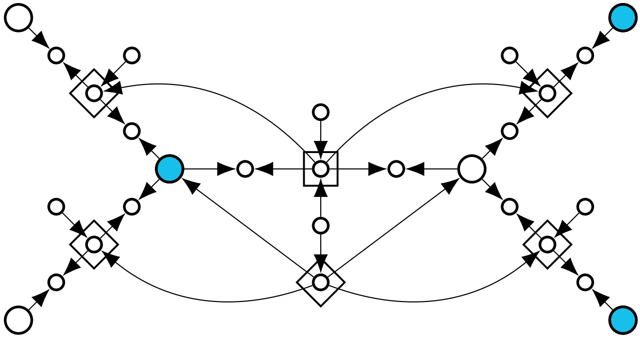
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Rightarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

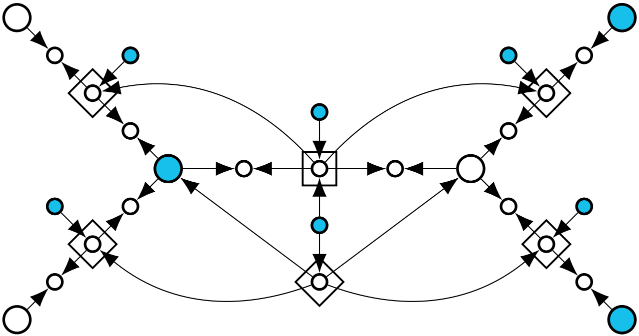
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Rightarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

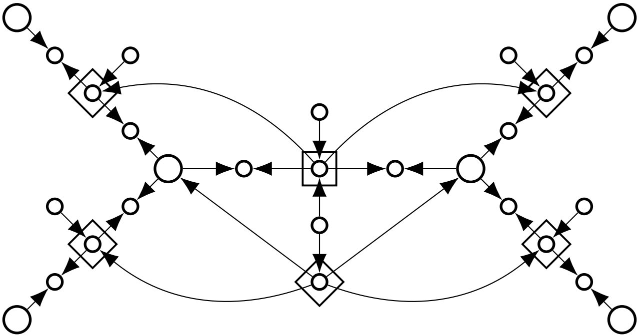
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Rightarrow \text{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

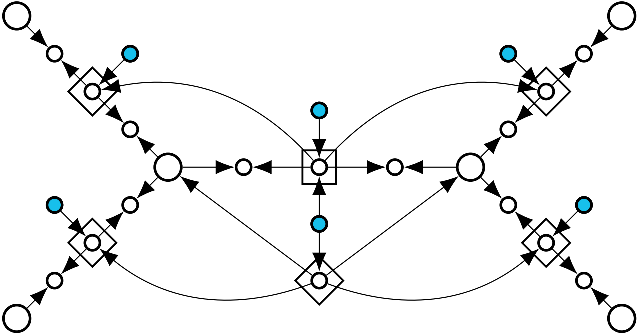
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

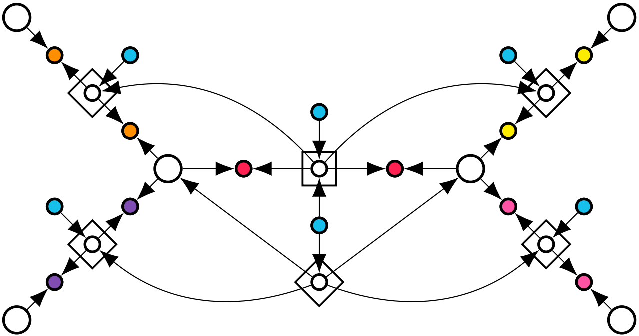
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

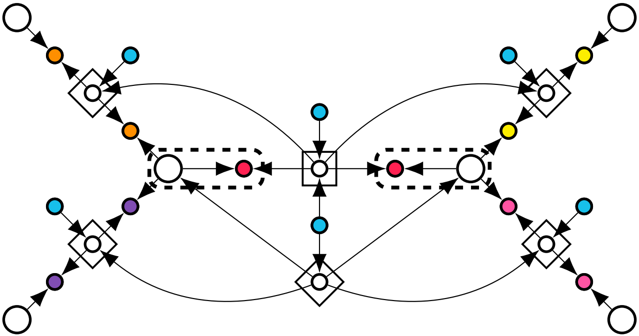
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \text{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

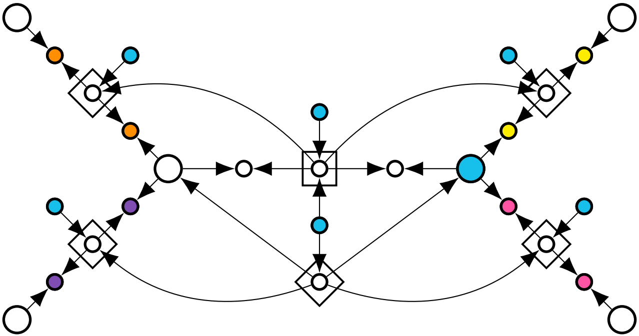
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \text{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

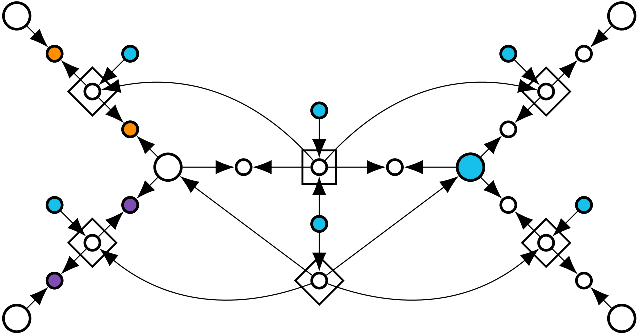
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



$$\text{Vertex cover} \leq k \Leftarrow \text{MD} \leq k + \frac{4|E|}{3}$$

NP-hardness (2) Combining gadgets

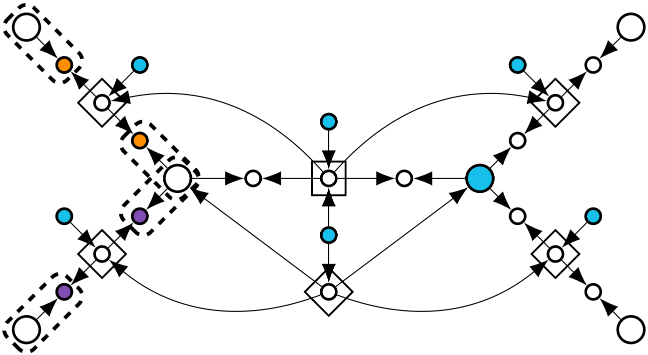
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k$ $\Leftarrow$ $\mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

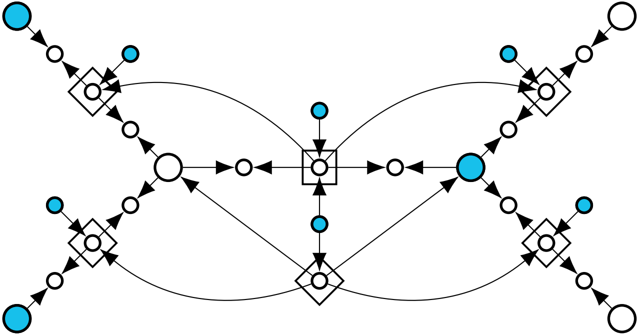
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

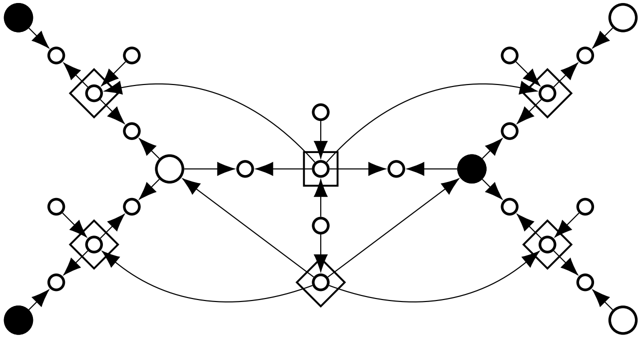
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

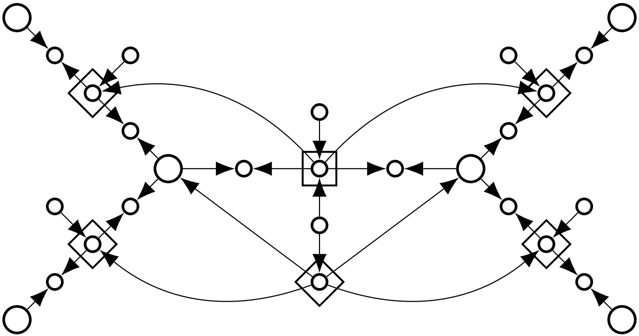
We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



Vertex cover $\leq k \Leftarrow \mathrm{MD} \leq k + \frac{4|E|}{3}$

NP-hardness (2) Combining gadgets

We start from a planar cubic graph and a perfect matching
We obtain a planar triangle-free DAG of max degree 6



$$\text{Vertex cover} \le k \Leftrightarrow \text{MD} \le k + \frac{4|E|}{3}$$

Parameterized complexity

Theorem [D., Foucaud & Hakanen, 2023+]

There is an $\mathcal{O}(n^3+m)+\mathcal{O}(t^5 2^{t^2} n)$ algorithm computing the metric dimension of a digraph of order n, size m and directed modular width at most t.

Parameterized complexity

Theorem [D., Foucaud & Hakanen, 2023+]

There is an $\mathcal{O}(n^3+m)+\mathcal{O}(t^5 2^{t^2} n)$ algorithm computing the metric dimension of a digraph of order n, size m and directed modular width at most t.

Algorithm

Generalized from [Belmonte *et al.*, 2017]

Parameterized complexity

Theorem [D., Foucaud & Hakanen, 2023+]

There is an $\mathcal{O}(n^3+m)+\mathcal{O}(t^5 2^{t^2} n)$ algorithm computing the metric dimension of a digraph of order n, size m and directed modular width at most t.

Algorithm

Generalized from [Belmonte *et al.*, 2017]

1. Compute all the distances [Floyd-Warshall]
2. Obtain an optimal modular decomposition [McConnell & de Montgolfier, 2005]

Parameterized complexity

Theorem [D., Foucaud & Hakanen, 2023+]

There is an $\mathcal{O}(n^3+m)+\mathcal{O}(t^5 2^{t^2} n)$ algorithm computing the metric dimension of a digraph of order n, size m and directed modular width at most t.

Algorithm

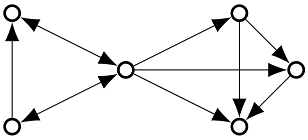
Generalized from [Belmonte *et al.*, 2017]

1. Compute all the distances [Floyd-Warshall]
2. Obtain an optimal modular decomposition [McConnell & de Montgolfier, 2005]
3. Start from the trivial modules, and combine them (dynamic programming)

Modular decompositions

Definition [Gallai, 1967] (and many others)

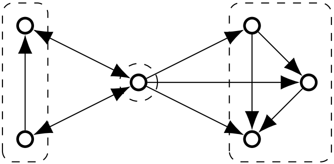
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.



Modular decompositions

Definition [Gallai, 1967] (and many others)

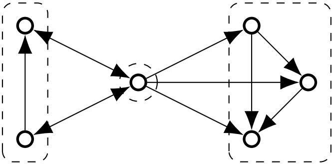
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.



Modular decompositions

Definition [Gallai, 1967] (and many others)

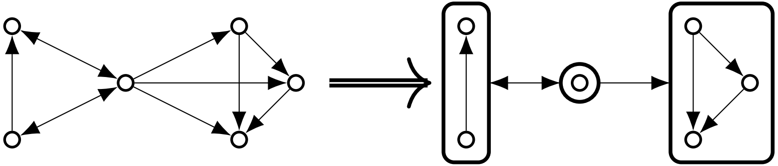
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.



Modular decompositions

Definition [Gallai, 1967] (and many others)

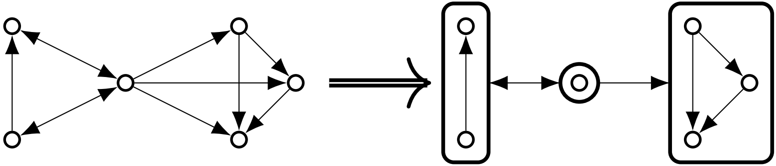
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.



Modular decompositions

Definition [Gallai, 1967] (and many others)

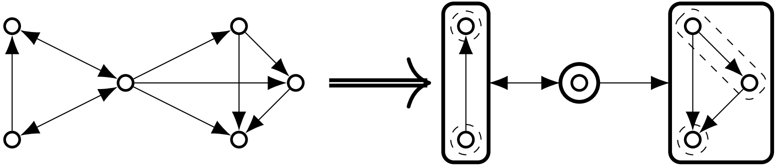
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.



Modular decompositions

Definition [Gallai, 1967] (and many others)

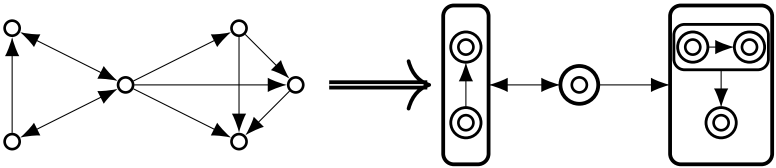
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.



Modular decompositions

Definition [Gallai, 1967] (and many others)

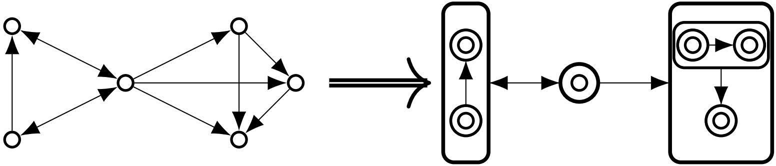
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.



Modular decompositions

Definition [Gallai, 1967] (and many others)

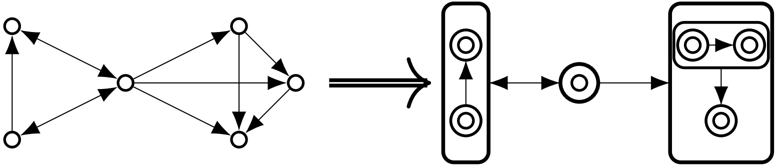
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.
The width of a decomposition is the **max** number of modules in one factorization step.



Modular decompositions

Definition [Gallai, 1967] (and many others)

A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.
The width of a decomposition is the **max** number of modules in one factorization step.

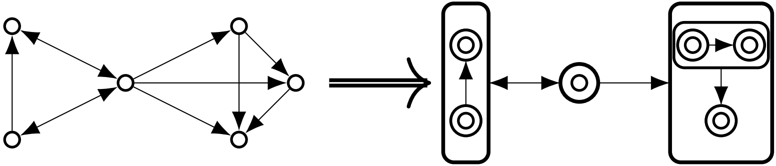


width 3

Modular decompositions

Definition [Gallai, 1967] (and many others)

A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.
The width of a decomposition is the **max** number of modules in one factorization step.
The modular width is the **min** width over all decompositions.

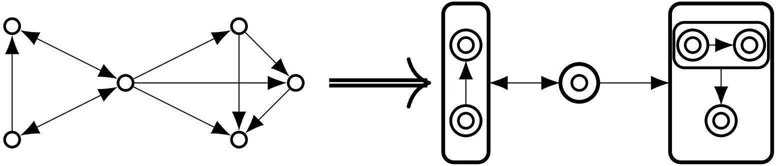


width 3

Modular decompositions

Definition [Gallai, 1967] (and many others)

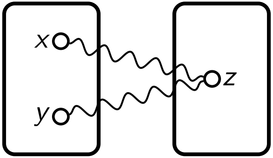
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.
The width of a decomposition is the **max** number of modules in one factorization step.
The modular width is the **min** width over all decompositions.



mw 3 width 3

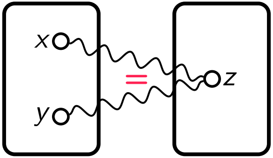
Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$,



Properties of the modular decomposition

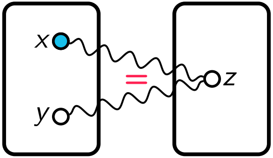
1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\mathrm{dist}(x,z) = \mathrm{dist}(y,z)$ and $\mathrm{dist}(z,x) = \mathrm{dist}(z,y)$



Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

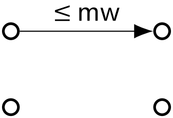


Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width

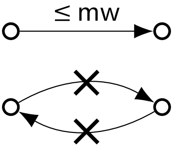


Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width or infinite



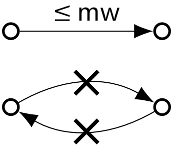
Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\mathrm{dist}(x,z) = \mathrm{dist}(y,z)$ and $\mathrm{dist}(z,x) = \mathrm{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\mathrm{mw})$

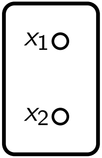


Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution
2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\text{mw})$
3. Given vertices $x_1, x_2 \in M_i$,



Properties of the modular decomposition

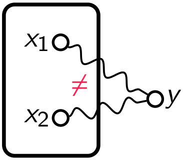
1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\text{mw})$

3. Given vertices $x_1, x_2 \in M_i$, if $\text{dist}(x_1, y) \neq \text{dist}(x_2, y)$,

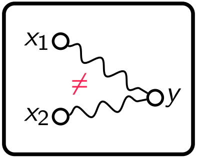


Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution
2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\text{mw})$
3. Given vertices $x_1, x_2 \in M_i$, if $\text{dist}(x_1, y) \neq \text{dist}(x_2, y)$, then $y \in M_i$



Properties of the modular decomposition

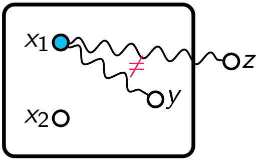
1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\text{mw})$

3. Given vertices $x_1, x_2 \in M_i$, if $\text{dist}(x_1, y) \neq \text{dist}(x_2, y)$, then $y \in M_i$ and one of x_1, x_2 will resolve y and $z \notin M_i$



Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x, z) = \text{dist}(y, z)$ and $\text{dist}(z, x) = \text{dist}(z, y)$

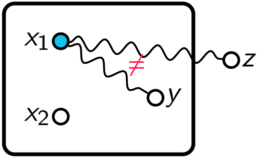
   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\text{mw})$

3. Given vertices $x_1, x_2 \in M_i$, if $\text{dist}(x_1, y) \neq \text{dist}(x_2, y)$, then $y \in M_i$ and one of x_1, x_2 will resolve y and $z \notin M_i$

   $\Rightarrow$ Combining local solutions is "easy" in this case



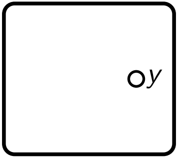
Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\mathrm{dist}(x,z) = \mathrm{dist}(y,z)$ and $\mathrm{dist}(z,x) = \mathrm{dist}(z,y)$

   $\Rightarrow$ All nontrivial modules contain a vertex in the solution
2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\mathrm{mw})$
3. Given vertices $x_1, x_2 \in M_i$, if $\mathrm{dist}(x_1, y) \neq \mathrm{dist}(x_2, y)$, then $y \in M_i$ and one of x_1, x_2 will resolve y and $z \notin M_i$

   $\Rightarrow$ Combining local solutions is "easy" in this case



But what if, for some $y \in M_i$,

Properties of the modular decomposition

1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

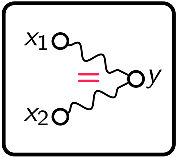
   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

2. The distance between vertices is either bounded by the modular width or infinite

   $\Rightarrow$ Allows us to bound DP steps by $f(\text{mw})$

3. Given vertices $x_1, x_2 \in M_i$, if $\text{dist}(x_1,y) \neq \text{dist}(x_2,y)$, then $y \in M_i$ and one of x_1, x_2 will resolve y and $z \notin M_i$

   $\Rightarrow$ Combining local solutions is "easy" in this case



But what if, for some $y \in M_i$,
$\text{dist}(x_1,y) = \text{dist}(x_2,y)$ for every $x_1, x_2 \in M_i$?

d-constant vertices

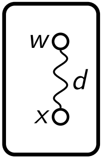
Definition

In a module M, a vertex x is d-constant if $\mathrm{dist}(w, x) = d$ for every $w \in M_R$ (where M_R is the local solution).

d-constant vertices

Definition

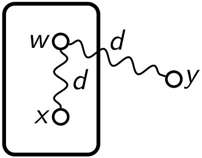
In a module M, a vertex x is d-constant if $\mathrm{dist}(w, x) = d$ for every $w \in M_R$ (where M_R is the local solution).



d-constant vertices

Definition

In a module M, a vertex x is d-constant if $\text{dist}(w, x) = d$ for every $w \in M_R$ (where M_R is the local solution).

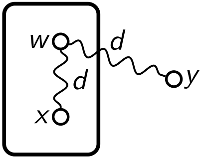


⚠ The local solution M_R does not resolve x and y!

d-constant vertices

Definition

In a module M, a vertex x is d-constant if $\text{dist}(w, x) = d$ for every $w \in M_R$ (where M_R is the local solution).



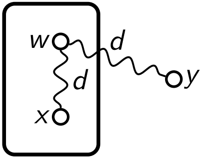
⚠ The local solution M_R does not resolve x and y!

⇒ We need to keep track of all d-constant vertices...

d-constant vertices

Definition

In a module M, a vertex x is d-constant if $\mathrm{dist}(w, x) = d$ for every $w \in M_R$ (where M_R is the local solution).



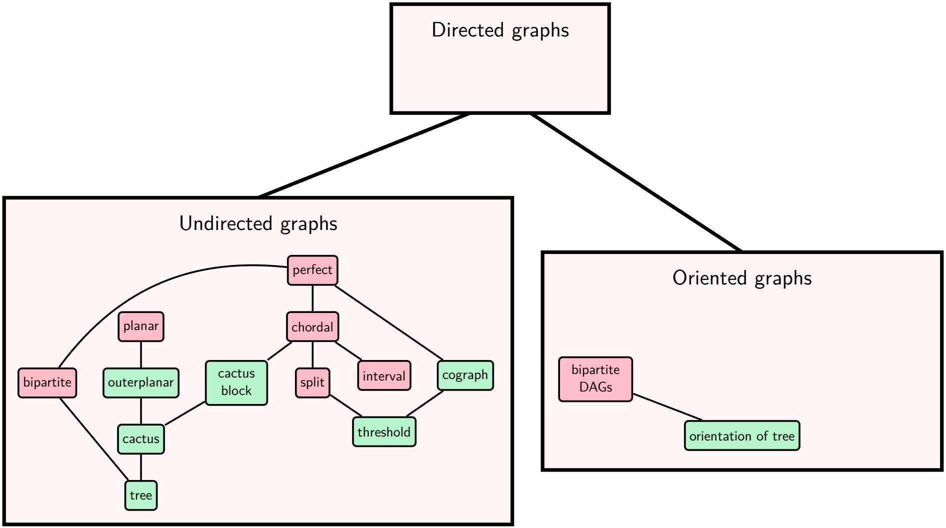
⚠The local solution M_R does not resolve x and y!

$\Rightarrow$ We need to keep track of all d-constant vertices...

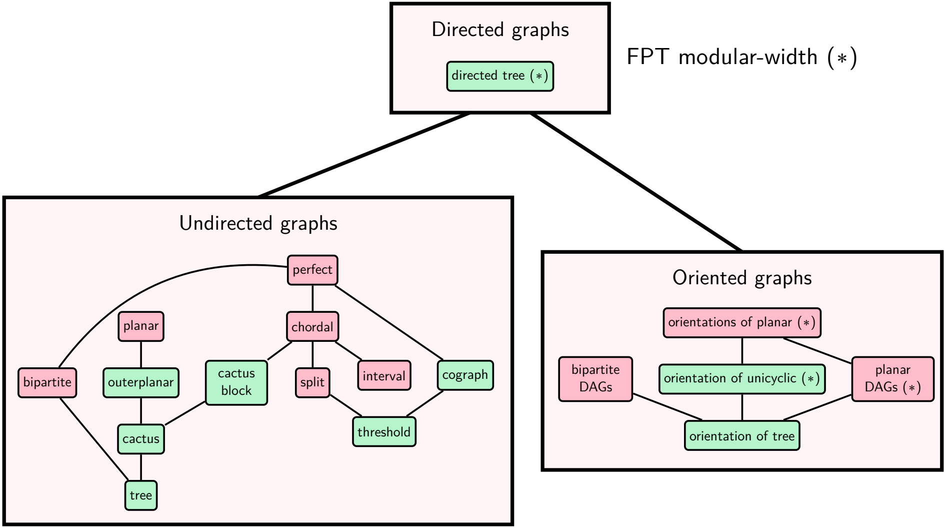
... but $d \in \{1, \ldots, \mathrm{mw}, \infty\}$ so their number is **bounded by** $\mathrm{mw}+1$ for each factor!

$\Rightarrow$ We can brute-force them when combining local solutions.

New inclusion diagram:



New inclusion diagram: $(*)$ = our results



Final words

Our contribution to Metric Dimension on directed graphs

- NP-completeness for a very restricted class
- Linear-time algorithms (directed trees, orientations of unicyclic)
- FPT algorithm using modular decomposition

Final words

Our contribution to Metric Dimension on directed graphs

- NP-completeness for a very restricted class
- Linear-time algorithms (directed trees, orientations of unicyclic)
- FPT algorithm using modular decomposition

Future work

1. Orientations of/Directed outerplanar?
2. DAGs of maximum distance 2?
3. Other parameterizations? Practical implementation?

Final words

Our contribution to Metric Dimension on directed graphs

- NP-completeness for a very restricted class
- Linear-time algorithms (directed trees, orientations of unicyclic)
- FPT algorithm using modular decomposition

Future work

1. Orientations of/Directed outerplanar?
2. DAGs of maximum distance 2?
3. Other parameterizations? Practical implementation?