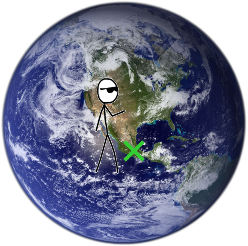
Algorithms and hardness for Metric Dimension on directed graphs

Antoine Dailly, Florent Foucaud, Anni Hakanen
LIMOS, Clermont-Ferrand
Funded by ANR GRALMECO

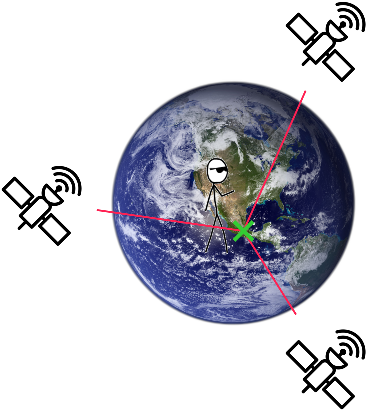
WG2023
June 30, 2023

Where does it come from?

Where does it come from?

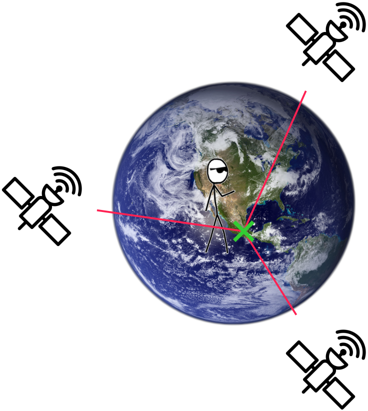


Where does it come from?



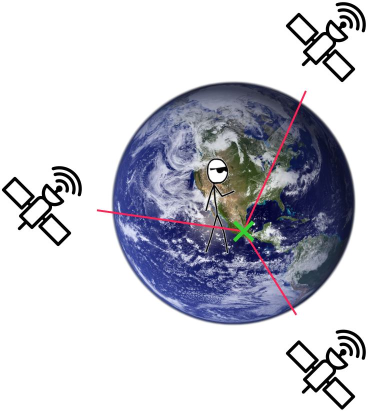
GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites

Where does it come from?

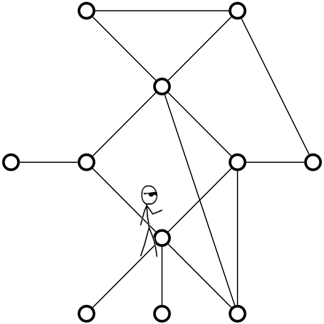


GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites

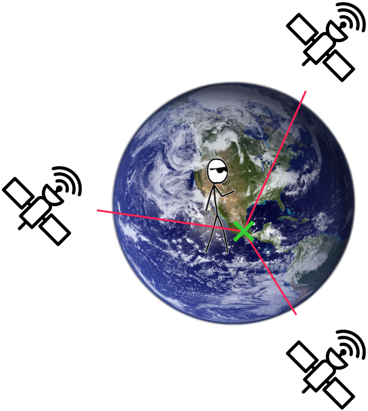
Where does it come from?



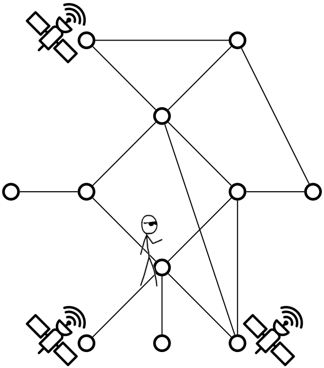
GPS, GLONASS, Galileo,
Beidou, IRNSS, QZSS: use
of at least four satellites



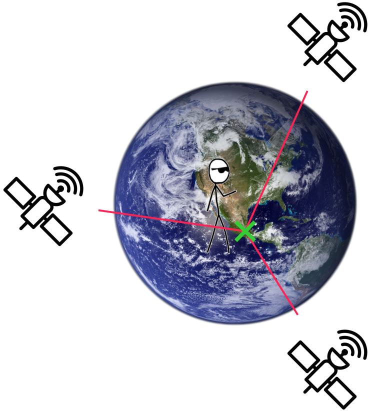
Where does it come from?



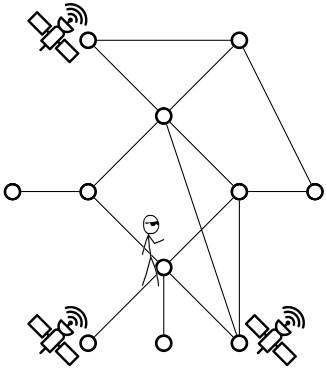
GPS, GLONASS, Galileo, Beidou, IRNSS, QZSS: use of at least four satellites



Where does it come from?



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



How many "satellites" would I need in a given graph?

Metric Dimension

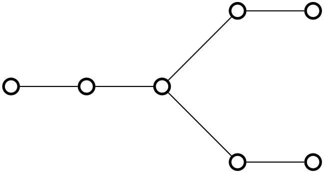
Definition

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

Metric Dimension

Definition

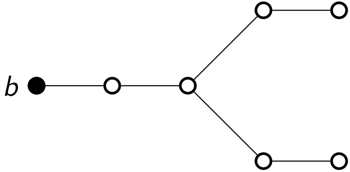
b resolves u and v if $\text{dist}(b,u) \neq \text{dist}(b,v)$



Metric Dimension

Definition

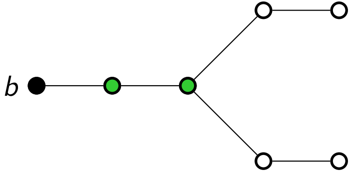
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

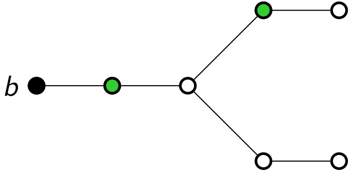
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

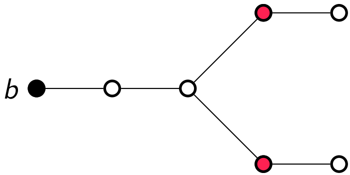
b resolves u and v if $\mathrm{dist}(b,u) \neq \mathrm{dist}(b,v)$



Metric Dimension

Definition

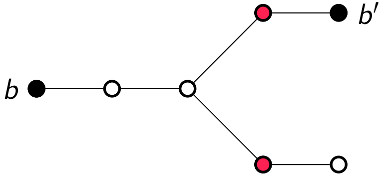
b resolves u and v if $\text{dist}(b,u) \neq \text{dist}(b,v)$



Metric Dimension

Definition

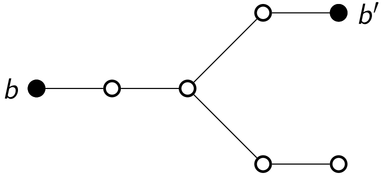
b resolves u and v if $\text{dist}(b,u) \neq \text{dist}(b,v)$



Metric Dimension

Definition

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



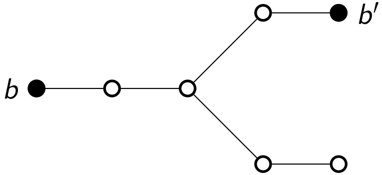
Resolving Set [Slater, 1975] [Harary & Melter, 1976]

$R \subseteq V(G)$ 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

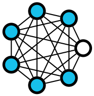
$\mathrm{MD}(G) =$ minimum size of a resolving set of G

Basic results

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

Basic results

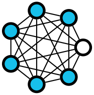
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



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

Basic results

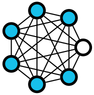
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



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

Basic results

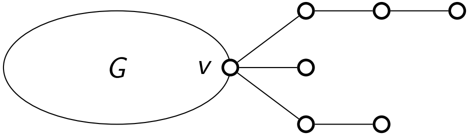
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



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

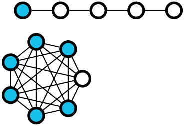
Legs

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



Basic results

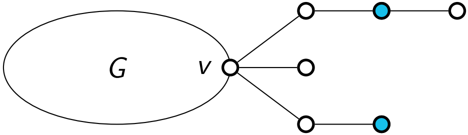
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



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

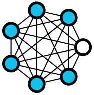
Legs

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

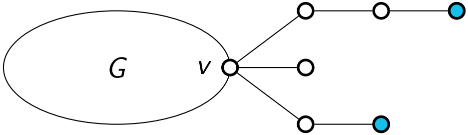
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



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

Legs

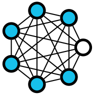
Paths with degree 2 inner vertices, and degree 1 and ≥ 3 endpoints. If v has k legs, $k-1$ have ≥ 1 vertex in a resolving set.



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

Basic results

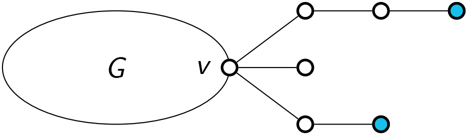
1. $\mathrm{MD}(G) = 1 \Leftrightarrow G$ is a path



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

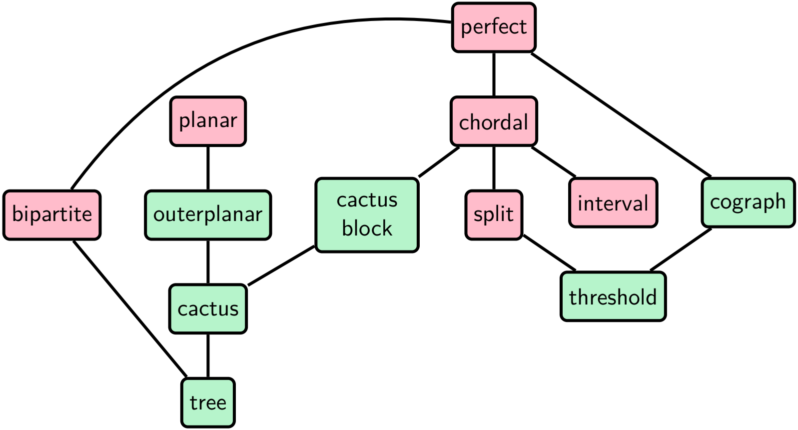
Legs

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.

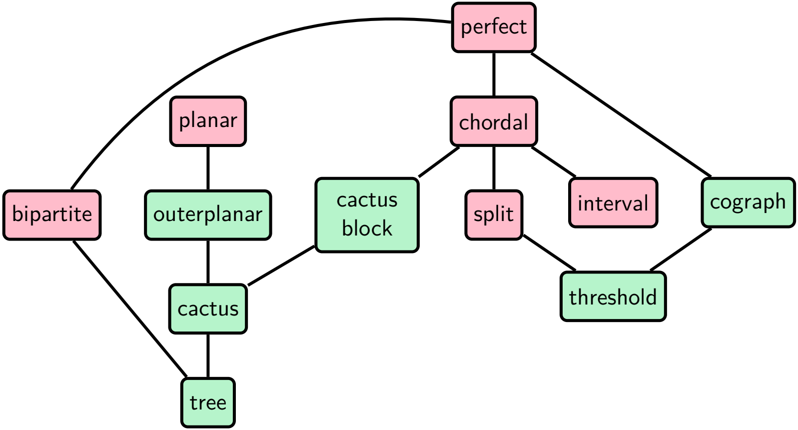


Simple leg rule: If v has $k \geq 2$ legs, select $k-1$ leg endpoints.

A difficult problem

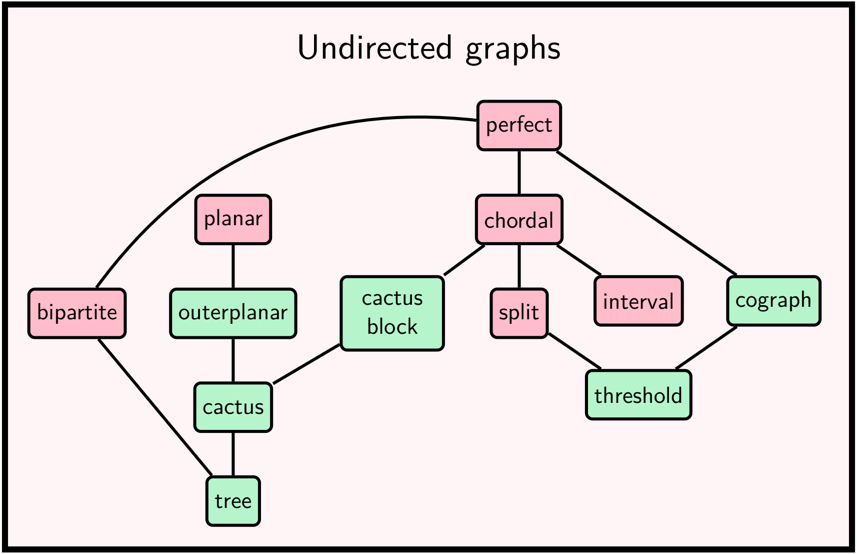


A difficult problem



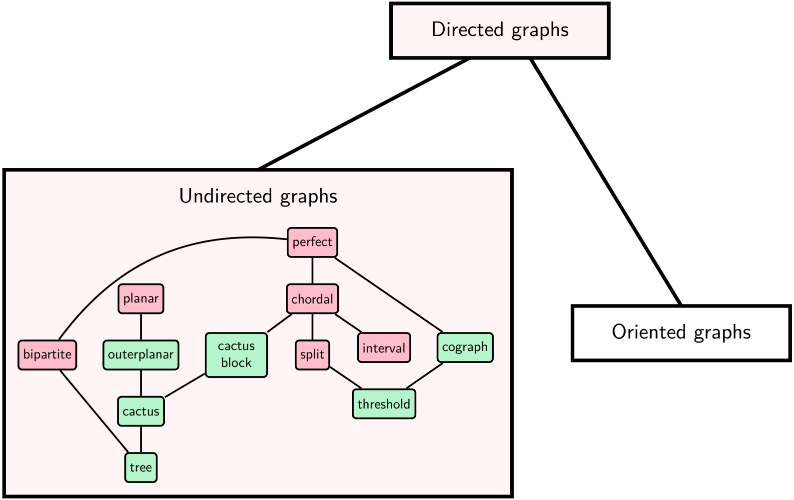
W[2]-complete and no better than $\log(n)$ approx in poly-time on subcubic graphs [HN13]

A difficult problem



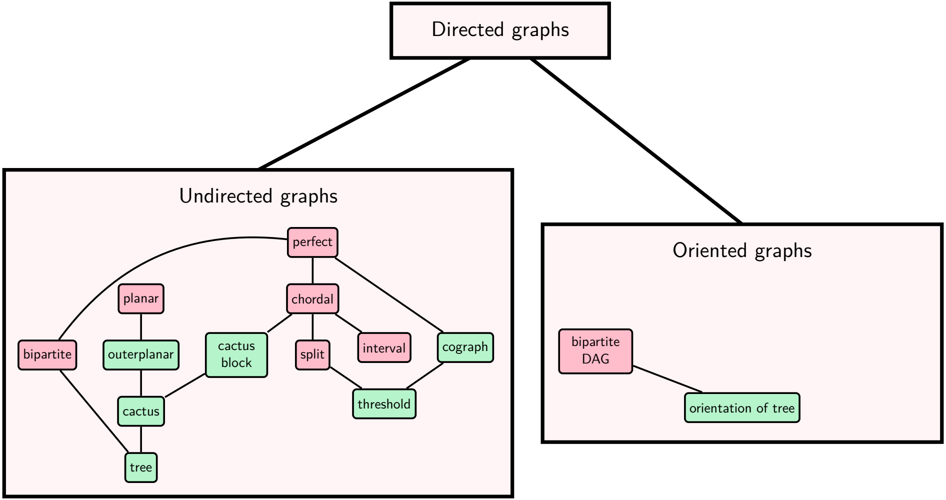
W[2]-complete and no better than $\log(n)$ approx in poly-time on subcubic graphs [HN13]

A difficult problem



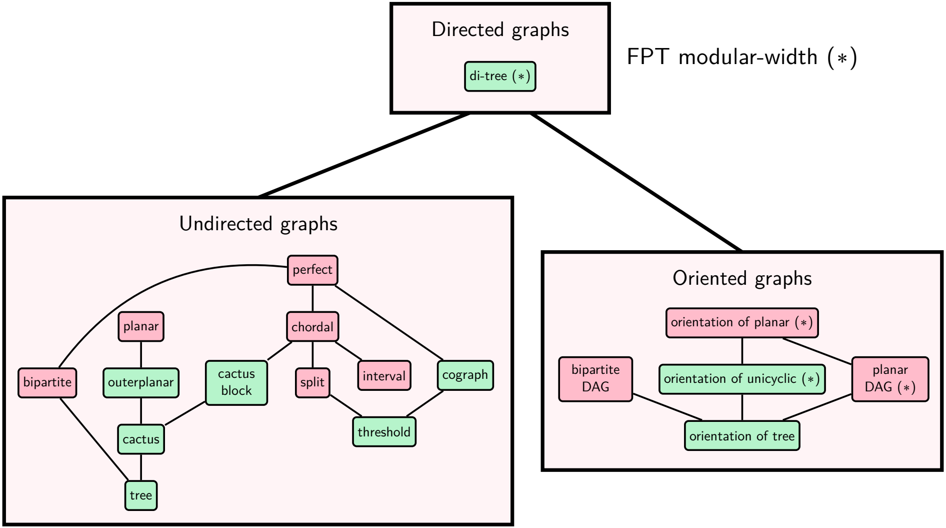
W[2]-complete and no better than $\log(n)$ approx in poly-time on subcubic graphs [HN13]

A difficult problem



W[2]-complete and no better than $\log(n)$ approx in poly-time on subcubic graphs [HN13]

A difficult problem $(*)$ = our results



W[2]-complete and no better than $\log(n)$ approx in poly-time on subcubic graphs [HN13]

Our results

Theorem [D., Foucaud & Hakanen, 2023]

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

Theorem [D., Foucaud & Hakanen, 2023]

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

Theorem [D., Foucaud & Hakanen, 2023]

FPT algorithm for Metric Dimension parameterized by directed modular width.

Di-trees

Definition

From a tree, transform every edge into an arc or a 2-cycle.

Di-trees

Definition

From a tree, transform every edge into an arc or a 2-cycle.

Algorithm: some mandatory vertices

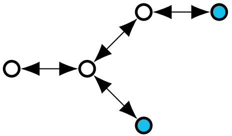
Di-trees

Definition

From a tree, transform every edge into an arc or a 2-cycle.

Algorithm: some mandatory vertices

- Simple leg rule in strongly connected components



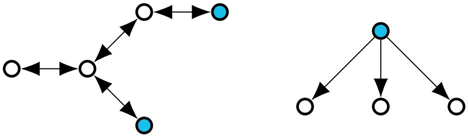
Di-trees

Definition

From a tree, transform every edge into an arc or a 2-cycle.

Algorithm: some mandatory vertices

- Simple leg rule in strongly connected components
- Sources



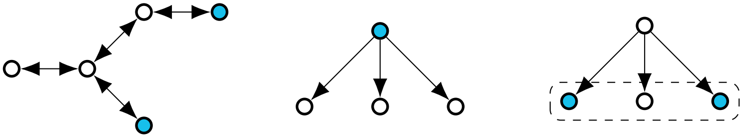
Di-trees

Definition

From a tree, transform every edge into an arc or a 2-cycle.

Algorithm: some mandatory vertices

- Simple leg rule in strongly connected components
- Sources
- Resolving sets of in-twins



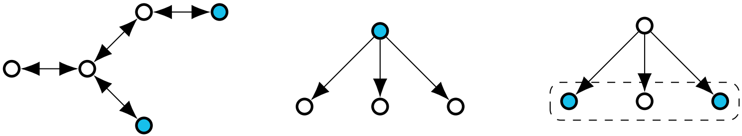
Di-trees

Definition

From a tree, transform every edge into an arc or a 2-cycle.

Algorithm: some mandatory vertices

- Simple leg rule in strongly connected components
- Sources
- Resolving sets of in-twins



... but some refinement is needed!

Refining in-twins

Almost-in-twins

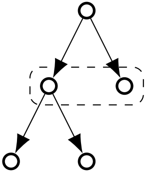
Vertices that share the same in-neighbour and:

Refining in-twins

Almost-in-twins

Vertices that share the same in-neighbour and:

- either are not in a nontrivial strongly connected component;

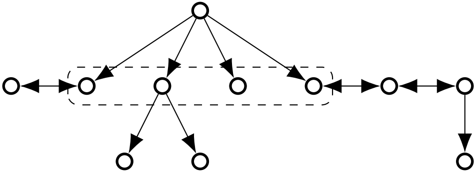


Refining in-twins

Almost-in-twins

Vertices that share the same in-neighbour and:

- either are not in a nontrivial strongly connected component;
- or are the endpoint of a so-called *escalator*.

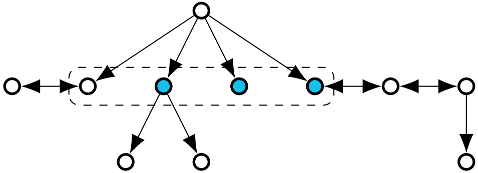


Refining in-twins

Almost-in-twins

Vertices that share the same in-neighbour and:

- either are not in a nontrivial strongly connected component;
- or are the endpoint of a so-called *escalator*.



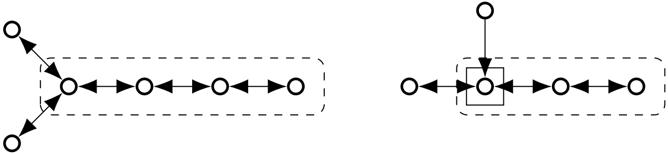
$\Rightarrow$ For each set of k almost-in-twins, take $k-1$ in the resolving set

Refining legs

Definition

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

- spans from a degree ≥ 3 (in the component) vertex or a vertex with an in-arc coming from outside the component

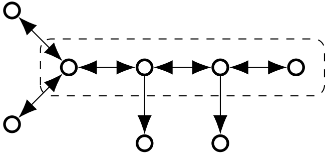


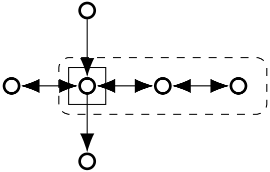
Refining legs

Definition

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

- spans from a degree ≥ 3 (in the component) vertex or a vertex with an in-arc coming from outside the component
- has at least one out-arc from a vertex other than its endpoint and no other in-arc from outside



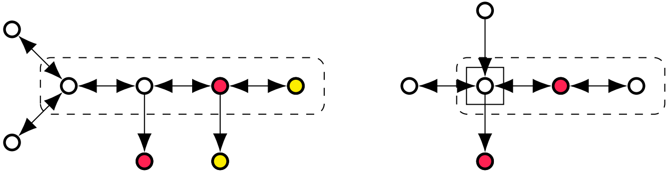


Refining legs

Definition

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

- spans from a degree ≥ 3 (in the component) vertex or a vertex with an in-arc coming from outside the component
- has at least one out-arc from a vertex other than its endpoint and no other in-arc from outside



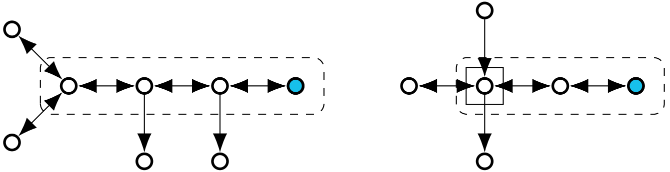
→ Conflict between pairs!

Refining legs

Definition

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

- spans from a degree ≥ 3 (in the component) vertex or a vertex with an in-arc coming from outside the component
- has at least one out-arc from a vertex other than its endpoint and no other in-arc from outside



→ Conflict between pairs!

⇒ Take the endpoint of each special leg

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

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

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

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

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Solve some special cases

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Solve some special cases
   2.2 Take the endpoint of every special leg

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Solve some special cases
   2.2 Take the endpoint of every special leg
   2.3 Resolve the remaining standard legs

The algorithm for di-trees

Theorem [D., Foucaud & Hakanen, 2023]

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

Algorithm

1. Take every source, resolve each set of almost-in-twins
2. For each strongly connected component
   2.1 Solve some special cases
   2.2 Take the endpoint of every special leg
   2.3 Resolve the remaining standard legs

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

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

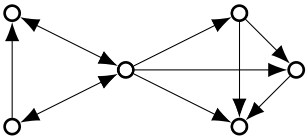
Generalized from [Belmonte *et al.*, 2017]

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)

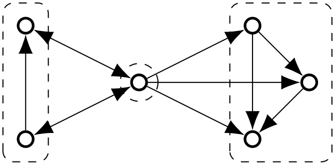
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.



Modular decompositions

Definition [Gallai, 1967] (and many others)

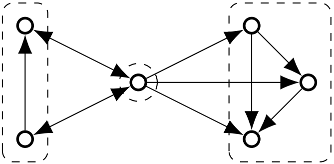
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.



Modular decompositions

Definition [Gallai, 1967] (and many others)

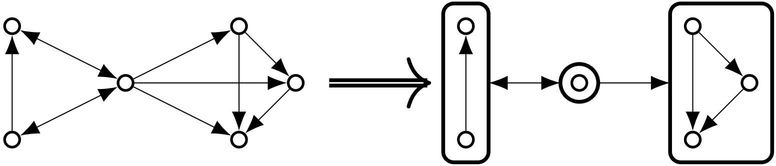
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.



Modular decompositions

Definition [Gallai, 1967] (and many others)

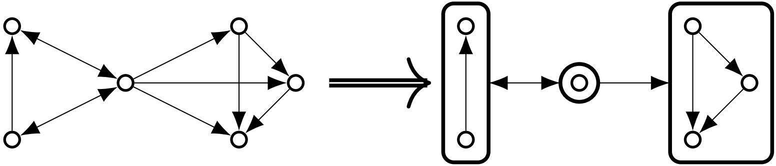
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.



Modular decompositions

Definition [Gallai, 1967] (and many others)

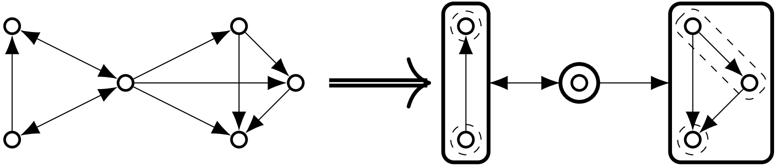
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.



Modular decompositions

Definition [Gallai, 1967] (and many others)

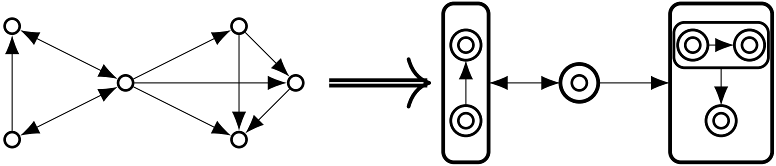
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.



Modular decompositions

Definition [Gallai, 1967] (and many others)

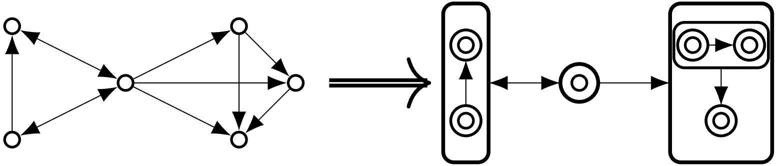
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.



Modular decompositions

Definition [Gallai, 1967] (and many others)

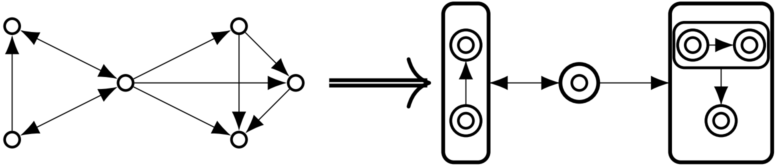
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.
The width of a decomposition is the **max** number of modules in one factorization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

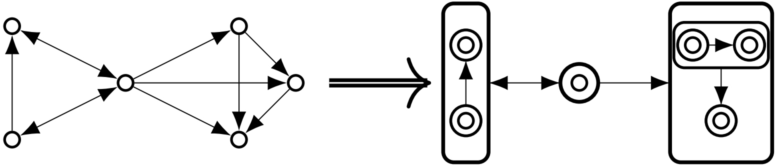


width 3

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.

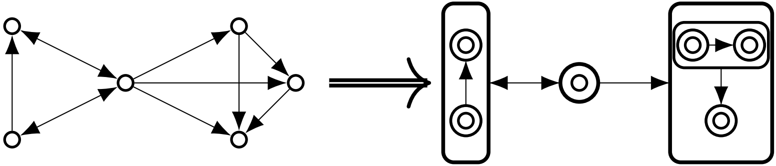


width 3

Modular decompositions

Definition [Gallai, 1967] (and many others)

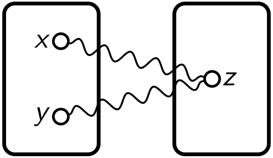
A module is a set X of vertices such that every vertex **outside** of X sees vertices of X in the same way.
A factorization is the graph of the modules.
A modular decomposition is obtained by repeating factorizations.
The width of a decomposition is the **max** number of modules in one factorization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



mw 3 width 3

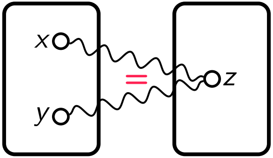
Properties of the modular decomposition

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



Properties of the modular decomposition

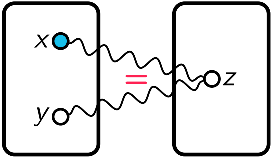
1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\mathrm{dist}(x, z) = \mathrm{dist}(y, z)$ and $\mathrm{dist}(z, x) = \mathrm{dist}(z, y)$



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

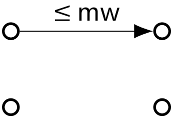


Properties of the modular decomposition

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

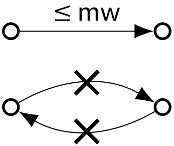
   $\Rightarrow$ All nontrivial modules contain a vertex in the solution

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



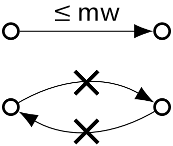
Properties of the modular decomposition

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 $f(\mathrm{mw})$

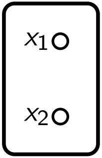


Properties of the modular decomposition

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 $x_1, x_2 \in M_i$,

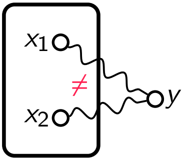


Properties of the modular decomposition

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

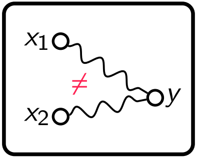
1. Given vertices $x, y \in M_i$ and $z \in M_j$, $\text{dist}(x,z) = \text{dist}(y,z)$ and $\text{dist}(z,x) = \text{dist}(z,y)$

   $\Rightarrow$ 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 $y \in M_i$

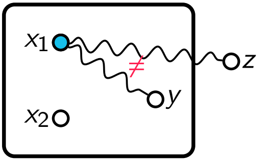


Properties of the modular decomposition

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 $z \notin M_i$



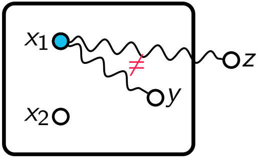
Properties of the modular decomposition

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



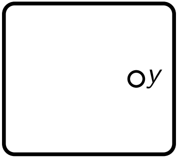
Properties of the modular decomposition

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 $y \in M_i$,

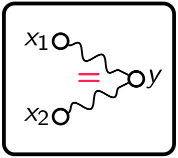
Properties of the modular decomposition

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

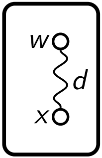
Definition

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

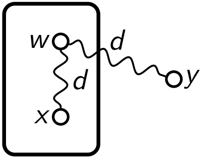
In a module M, a vertex x is d-constant if $\mathrm{dist}(w, x) = d$ for every $w \in M_R$ (where M_R is the local solution).



d-constant vertices

Definition

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

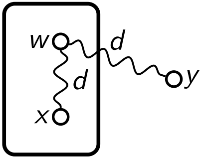


⚠ The local solution M_R does not resolve x and y!

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).



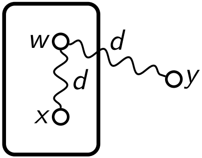
⚠ The local solution M_R does not resolve x and y!

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

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!

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

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

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

Final words

Our contribution to Metric Dimension on directed graphs

- NP-completeness for a very restricted class
- Linear-time algorithms (di-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 (di-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 (di-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?