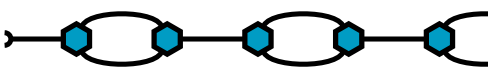
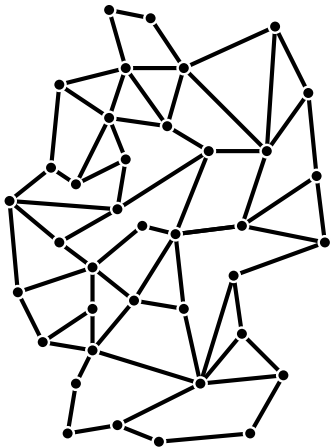
Faster, Higher, Easier: Toward a Systematic Study of Parameterized Vertex and Edge Selection Problems

Philipp Schepper



September 29, 2025

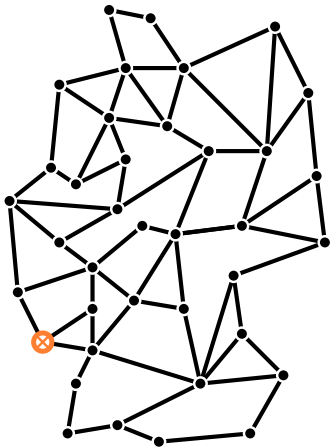
Fire Protection Train (FPT) Problem



Fire Protection Train (FPT) Problem

Task:

Find (few) locations to position the fire trains at.

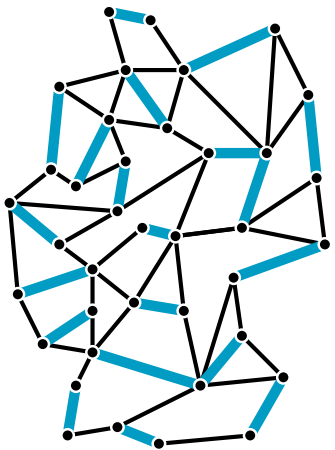


Fire Protection Train (FPT) Problem

Task:

Find (few) locations to position the fire trains at.

Moving Train

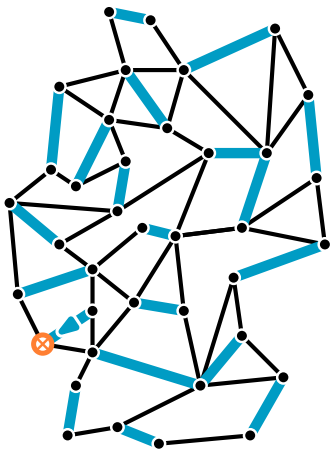


Fire Protection Train (FPT) Problem

Task:

Find (few) locations to position the fire trains at.

Moving Train

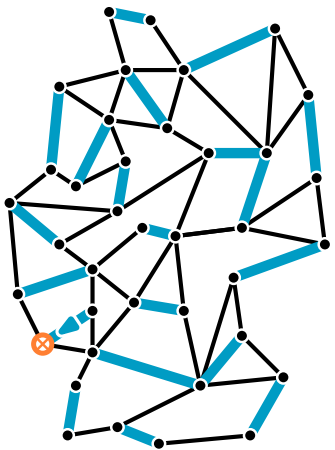


Fire Protection Train (FPT) Problem

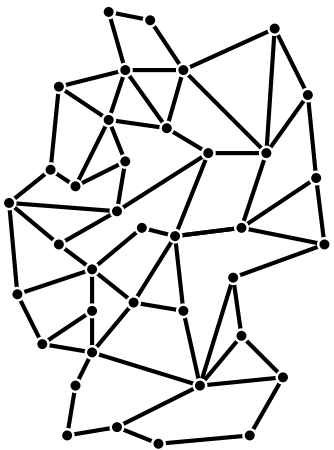
Task:

Find (few) locations to position the fire trains at.

Moving Train



Stationary Position

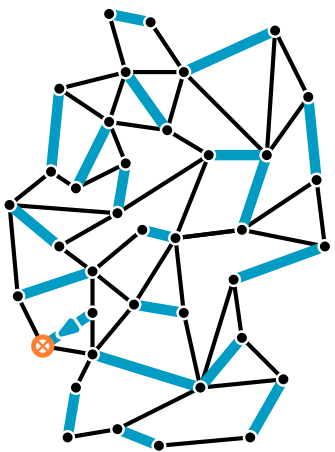


Fire Protection Train (FPT) Problem

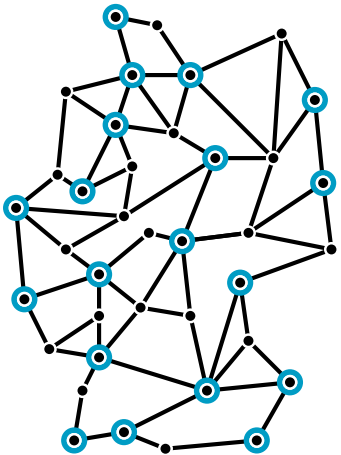
Task:

Find (few) locations to position the fire trains at.

Moving Train



Stationary Position

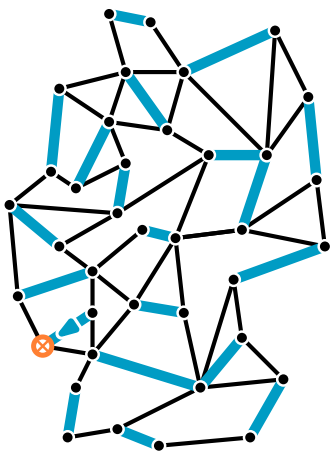


Fire Protection Train (FPT) Problem

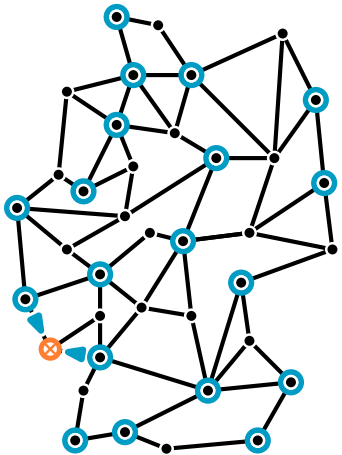
Task:

Find (few) locations to position the fire trains at.

Moving Train



Stationary Position



Fire Protection Train (FPT) Problem

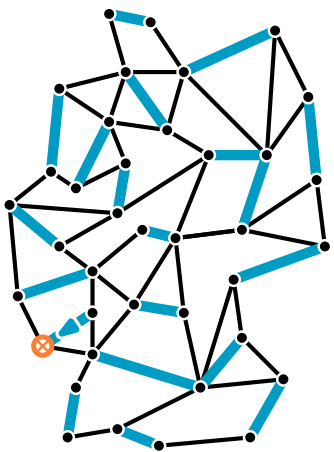
Task:

Find (few) locations to position the fire trains at.

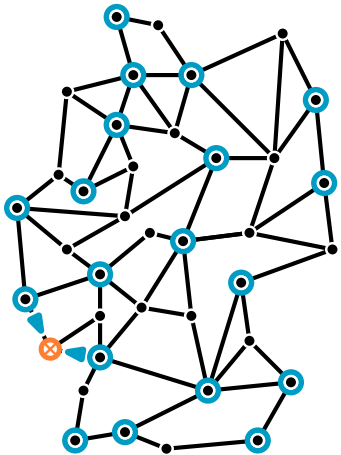
Next:

Formalize and solve the problem!

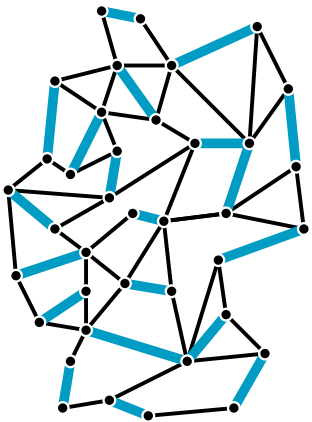
Moving Train

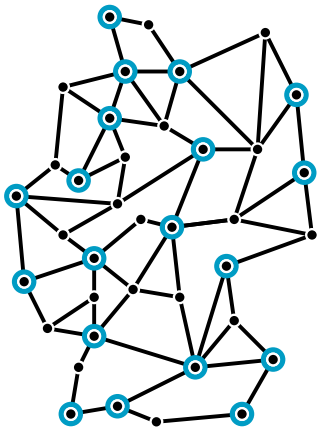


Stationary Position

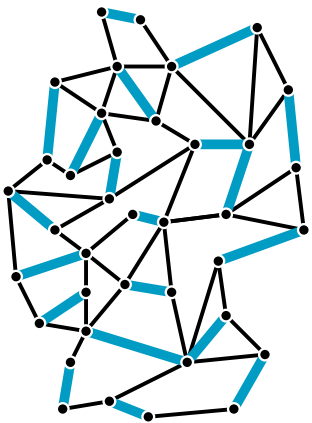


Fire Protection Train (FPT) Problem: Formalization



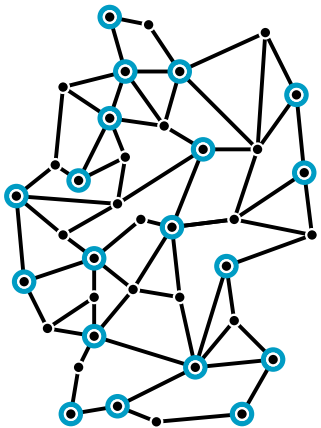


Fire Protection Train (FPT) Problem: Formalization

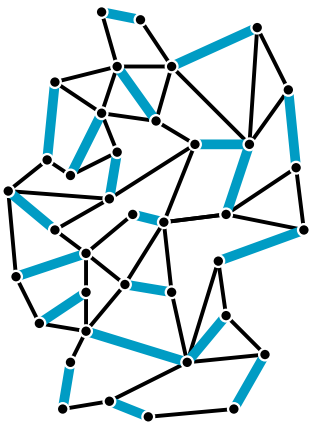


Edge Cover

Each vertex incident to ≥ 1 *selected edge*.



Fire Protection Train (FPT) Problem: Formalization



Edge Cover

Each vertex incident to

≥ 1 *selected edge*.

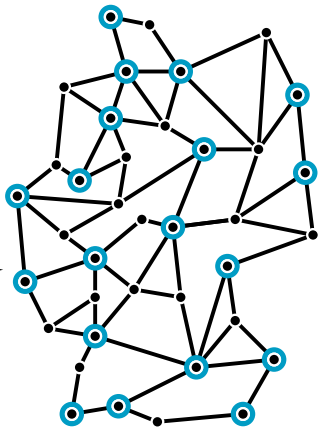
Total 2-Domination

Selected vertex:

≥ 1 selected neighbor.

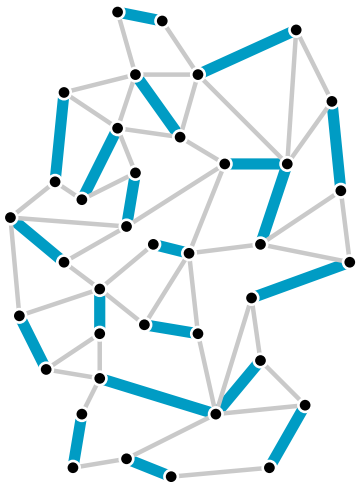
Unselected vertex:

≥ 2 selected neighbor.



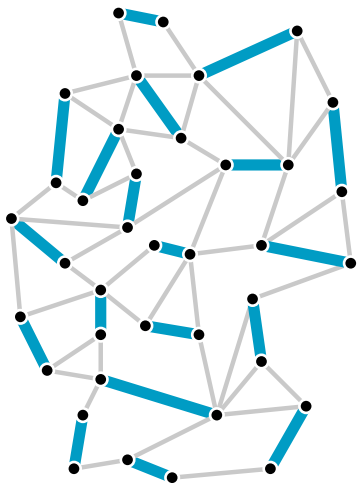
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	



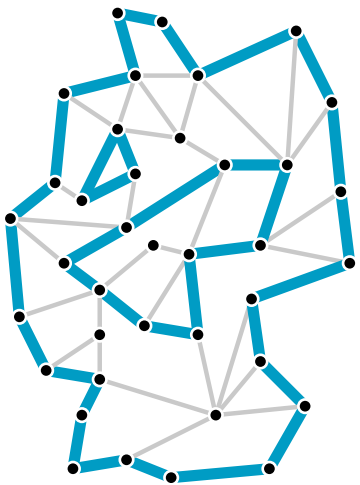
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	
Matching	≤ 1	
Perfect Matching	$= 1$	



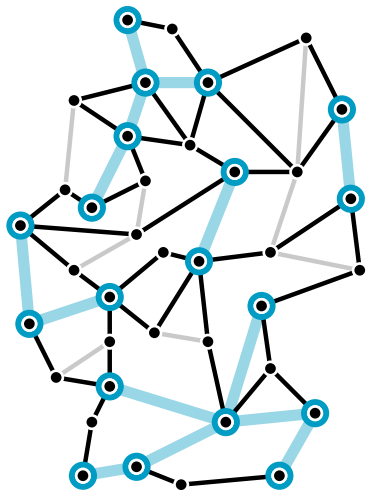
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	
Matching	≤ 1	
Perfect Matching	$= 1$	
Cycle Packing	$= 0$ or $= 2$	



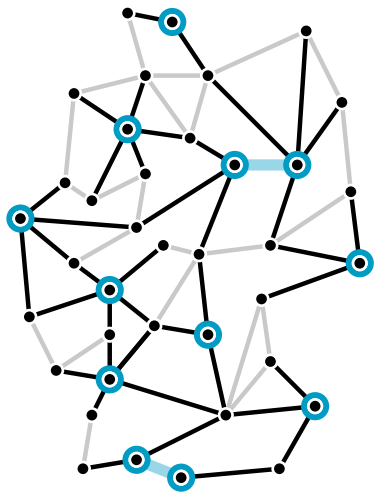
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	
Matching	≤ 1	
Perfect Matching	$= 1$	
Cycle Packing	$= 0$ or $= 2$	
Total 2-Domination	≥ 1	≥ 2



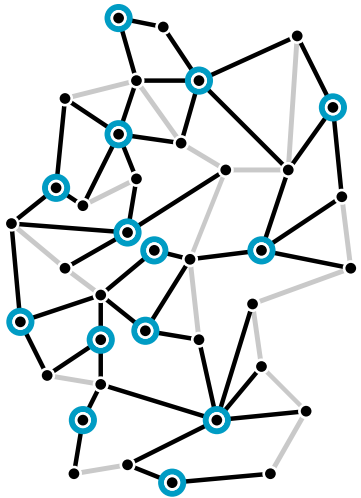
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	
Matching	≤ 1	
Perfect Matching	$= 1$	
Cycle Packing	$= 0$ or $= 2$	
Total 2-Domination	≥ 1	≥ 2
Dominating Set	≥ 0	≥ 1



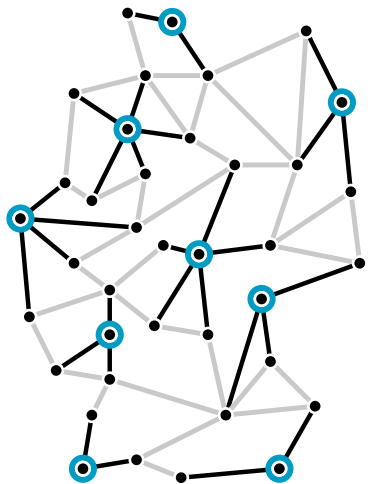
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	
Matching	≤ 1	
Perfect Matching	$= 1$	
Cycle Packing	$= 0$ or $= 2$	
Total 2-Domination	≥ 1	≥ 2
Dominating Set	≥ 0	≥ 1
Independent Set	$= 0$	≥ 0



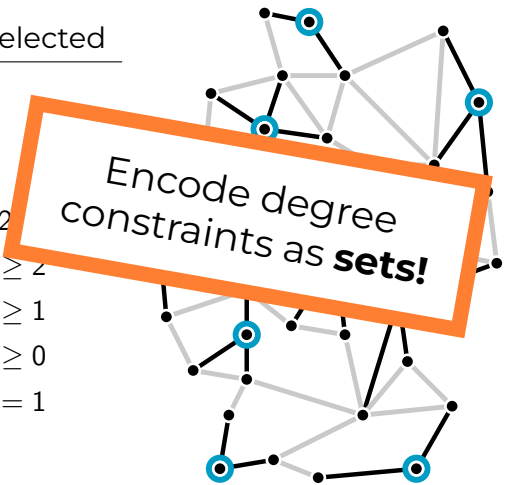
More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover	≥ 1	
Matching	≤ 1	
Perfect Matching	$= 1$	
Cycle Packing	$= 0$ or $= 2$	
Total 2-Domination	≥ 1	≥ 2
Dominating Set	≥ 0	≥ 1
Independent Set	$= 0$	≥ 0
Perfect Code	$= 0$	$= 1$



More Edge and Vertex Selection Problems

	Selected	Unselected
Edge Cover		≥ 1
Matching		≤ 1
Perfect Matching		$= 1$
Cycle Packing		$= 0$ or $= 2$
Total 2-Domination	≥ 1	≥ 2
Dominating Set	≥ 0	≥ 1
Independent Set	$= 0$	≥ 0
Perfect Code	$= 0$	$= 1$



Selecting Edges → Generalized Factor

Fix a set $\Lambda \subseteq \mathbb{N}$.

Λ-Factor [Lovász 1972]

Given: A simple graph G.

Is there an *edge set* $S \subseteq E(G)$ such that
for every vertex $v \in V(G)$ we have $\deg_S(v) \in \Lambda$?

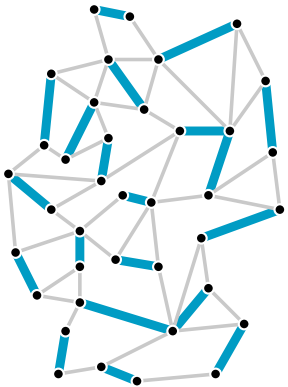
Selecting Edges → Generalized Factor

Fix a set $\Lambda \subseteq \mathbb{N}$.

Λ-Factor [Lovász 1972]

Given: A simple graph G.

Is there an *edge set* $S \subseteq E(G)$ such that
for every vertex $v \in V(G)$ we have $\deg_S(v) \in \Lambda$?



$\{1, 2, 3, \dots\}$-Factor

Selecting Edges → Generalized Factor

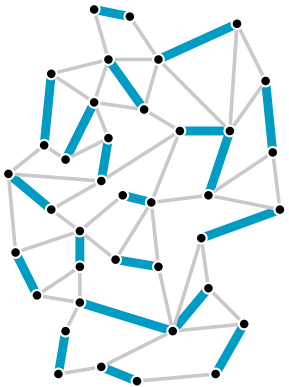
Fix a set $\Lambda \subseteq \mathbb{N}$.

Λ-Factor [Lovász 1972]

Given: A simple graph G.

Is there an *edge set* $S \subseteq E(G)$ such that
for every vertex $v \in V(G)$ we have $\deg_S(v) \in \Lambda$?

	Degree	Λ
Edge Cover	≥ 1	$\{1, 2, 3, \dots\}$
Matching	≤ 1	$\{0, 1\}$
Perfect Matching	$= 1$	$\{1\}$
Cycle Packing	$= 0$ or $= 2$	$\{0, 2\}$



$\{1, 2, 3, \dots\}$-Factor

Selecting Edges → Generalized Factor

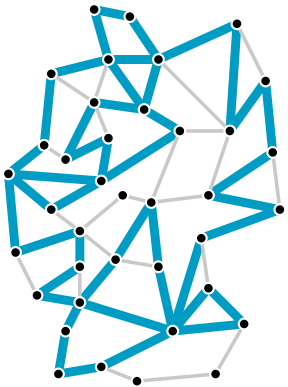
Fix a set $\Lambda \subseteq \mathbb{N}$.

Λ-Factor [Lovász 1972]

Given: A simple graph G.

Is there an *edge set* $S \subseteq E(G)$ such that
for every vertex $v \in V(G)$ we have $\deg_S(v) \in \Lambda$?

	Degree	Λ
Edge Cover	≥ 1	$\{1, 2, 3, \dots\}$
Matching	≤ 1	$\{0, 1\}$
Perfect Matching	$= 1$	$\{1\}$
Cycle Packing	$= 0$ or $= 2$	$\{0, 2\}$



$\{0, 2, 4, 6, \dots\}$-Factor

Selecting Vertices → Generalized Domination

Fix two sets $\sigma, \rho \subseteq \mathbb{N}$.

(σ, ρ)-DomSet [Telle 1994]

Given: A simple graph G.

Is there a *vertex set* $S \subseteq V(G)$ such that
– for each $v \in V(G) \cap S$, we have $|N(v) \cap S| \in \sigma$, and
– for each $v \in V(G) \setminus S$, we have $|N(v) \cap S| \in \rho$?

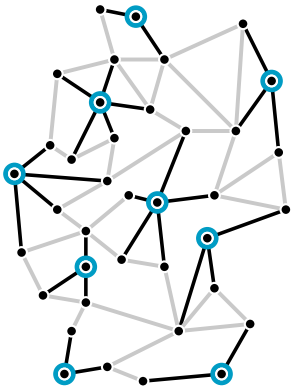
Selecting Vertices → Generalized Domination

Fix two sets $\sigma, \rho \subseteq \mathbb{N}$.

(σ, ρ)-DomSet [Telle 1994]

Given: A simple graph G.

Is there a *vertex set* $S \subseteq V(G)$ such that
– for each $v \in V(G) \cap S$, we have $|N(v) \cap S| \in \sigma$, and
– for each $v \in V(G) \setminus S$, we have $|N(v) \cap S| \in \rho$?



$(\{0\}, \{1\})$-DomSet

Selecting Vertices → Generalized Domination

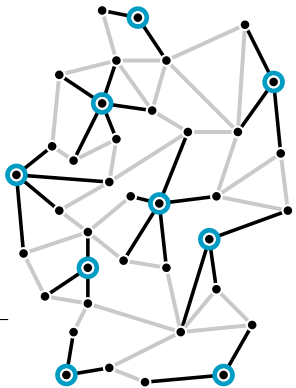
Fix two sets $\sigma, \rho \subseteq \mathbb{N}$.

(σ, ρ)-DomSet [Telle 1994]

Given: A simple graph G.

Is there a *vertex set* $S \subseteq V(G)$ such that
– for each $v \in V(G) \cap S$, we have $|N(v) \cap S| \in \sigma$, and
– for each $v \in V(G) \setminus S$, we have $|N(v) \cap S| \in \rho$?

	Sel.	Uns.	σ	ρ
Total 2-Dom.	≥ 1	≥ 2	$\{1, 2, \dots\}$	$\{2, 3, \dots\}$
Independent Set	$= 0$	≥ 0	$\{0\}$	$\mathbb{N}$
Dominating Set	≥ 0	≥ 1	$\mathbb{N}$	$\{1, 2, \dots\}$
Perfect Code	$= 0$	$= 1$	$\{0\}$	$\{1\}$

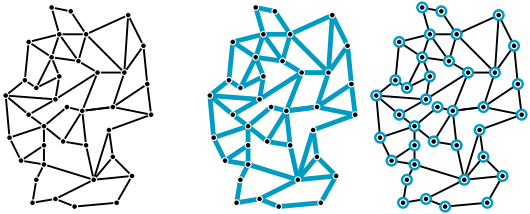


$(\{0\}, \{1\})$-DomSet

Solving the Problems

Few cases are trivial:

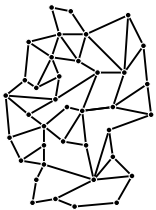
- $0 \in \Lambda$ or $0 \in \rho \rightarrow$ select nothing
- $\Lambda = \sigma = \mathbb{N} \rightarrow$ select everything

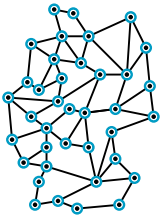


Solving the Problems

Few cases are trivial:

- $0 \in \Lambda$ or $0 \in \rho \rightarrow$ select nothing
- $\Lambda = \sigma = \mathbb{N} \rightarrow$ select everything





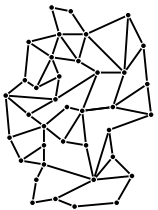
Our approach to the general cases:

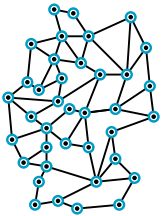
- Restrict input graphs

Solving the Problems

Few cases are trivial:

- $0 \in \Lambda$ or $0 \in \rho \to$ select nothing
- $\Lambda = \sigma = \mathbb{N} \to$ select everything





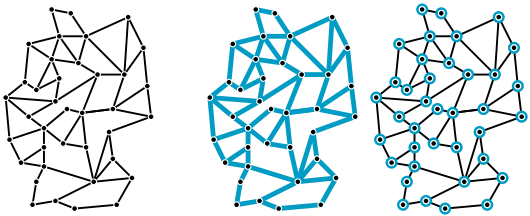
Our approach to the general cases:

- ~~Restrict input graphs~~
- Study parameterization by *treewidth*

Solving the Problems

Few cases are trivial:

- $0 \in \Lambda$ or $0 \in \rho \to$ select nothing
- $\Lambda = \sigma = \mathbb{N} \to$ select everything



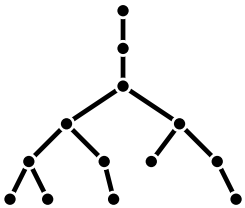
Our approach to the general cases:

- ~~Restrict input graphs~~
- Study parameterization by *treewidth*
- Consider *finite* and *cofinite* sets (i.e., the set or its complement is finite)
  $\rightsquigarrow$ covers most classical problems already

Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

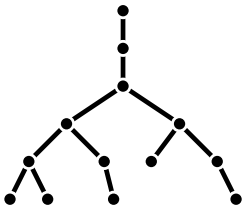


Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs

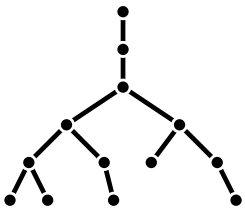


Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs



Treewidth …

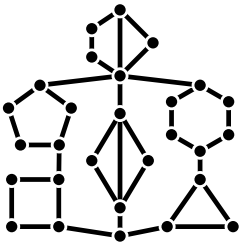
- measures how “tree-like” a graph is.

Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs



Treewidth …

- measures how “tree-like” a graph is.

Interlude: Treewidth

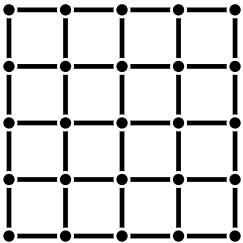
Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs

Treewidth …

- measures how “tree-like” a graph is.



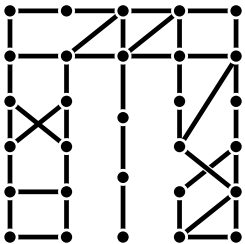
Large treewidth!

Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs



Treewidth …

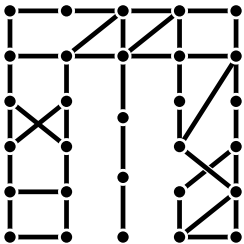
- measures how “tree-like” a graph is.

Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs



Treewidth …

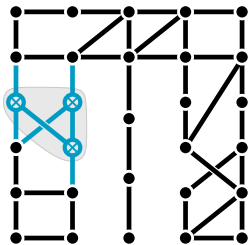
- measures how “tree-like” a graph is.
- has a definition that is hard to digest.

Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs



Treewidth …

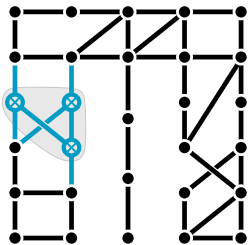
- measures how “tree-like” a graph is.
- has a definition that is hard to digest.
- is determined by special separators, referred to as “bags”.

Interlude: Treewidth

Motivation

Frequently straight-forward algorithms for trees

$\rightsquigarrow$ Extend to general graphs



Treewidth …

- measures how “tree-like” a graph is.
- has a definition that is hard to digest.
- is determined by special separators, referred to as “bags”.
- depends on some tree structure that allows for DPs (“tree decomposition”).

Solving the Problems: Known Results

Generalized Factor:

- Set Λ has no large “gap”: polynomial-time
  [Cornuéjols 1988]

Generalized Domination:

- No unified condition for polynomial-time cases (only individual results)

Solving the Problems: Known Results

Generalized Factor:

- Set Λ has no large "gap": polynomial-time [Cornuéjols 1988]
- Set is interval + 0: algorithm parameterized by treewidth [ACGMM 2018]

Generalized Domination:

- No unified condition for polynomial-time cases (only individual results)
- Finite or cofinite sets: algorithm parameterized by treewidth [vRBR 2009]

Solving the Problems: Known Results

Generalized Factor:

- Set Λ has no large "gap": polynomial-time [Cornuéjols 1988]
- Set is interval + 0: algorithm parameterized by treewidth [ACGMM 2018]

Generalized Domination:

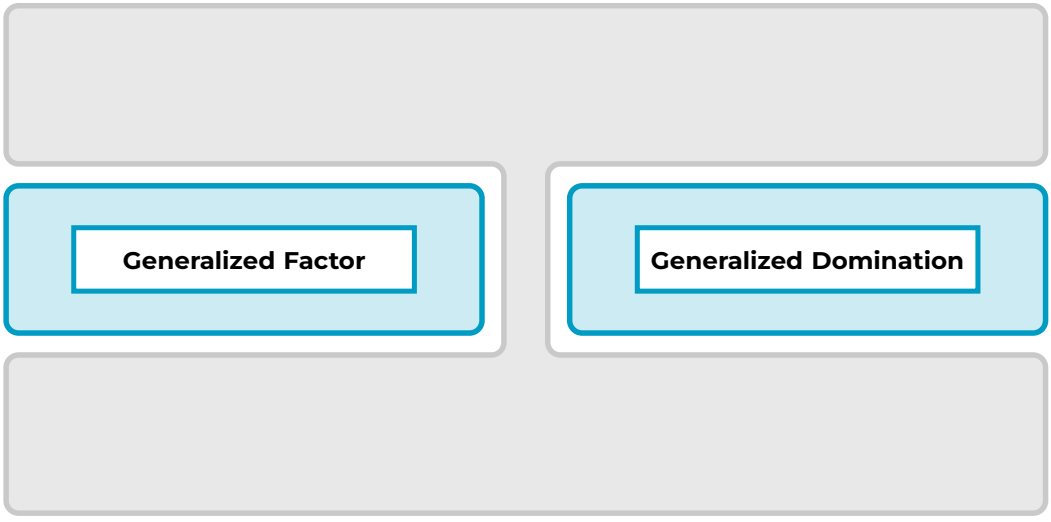
- No unified condition for polynomial-time cases (only individual results)
- Finite or cofinite sets: algorithm parameterized by treewidth [vRBR 2009]

What about more general sets?

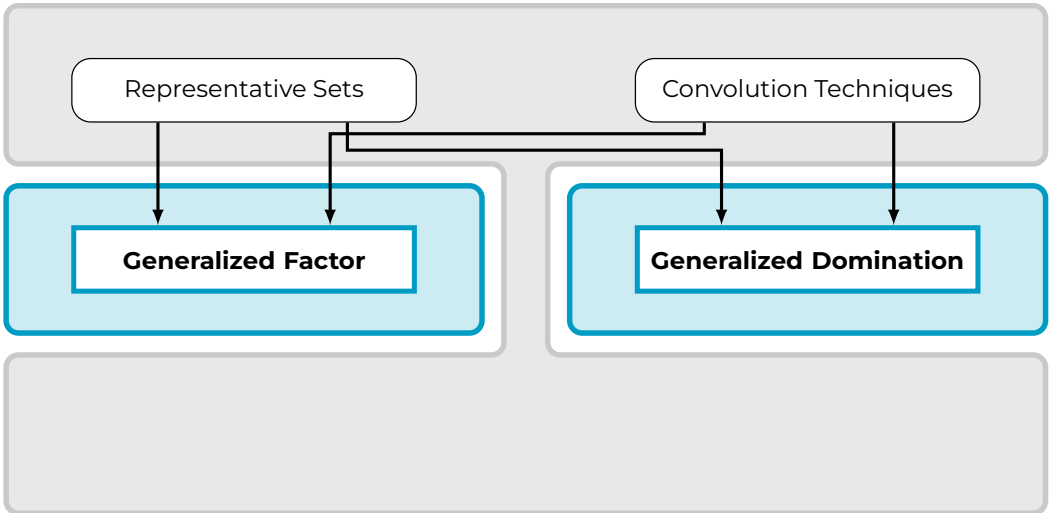
Are the known algorithms optimal?

What about counting the number of solutions?

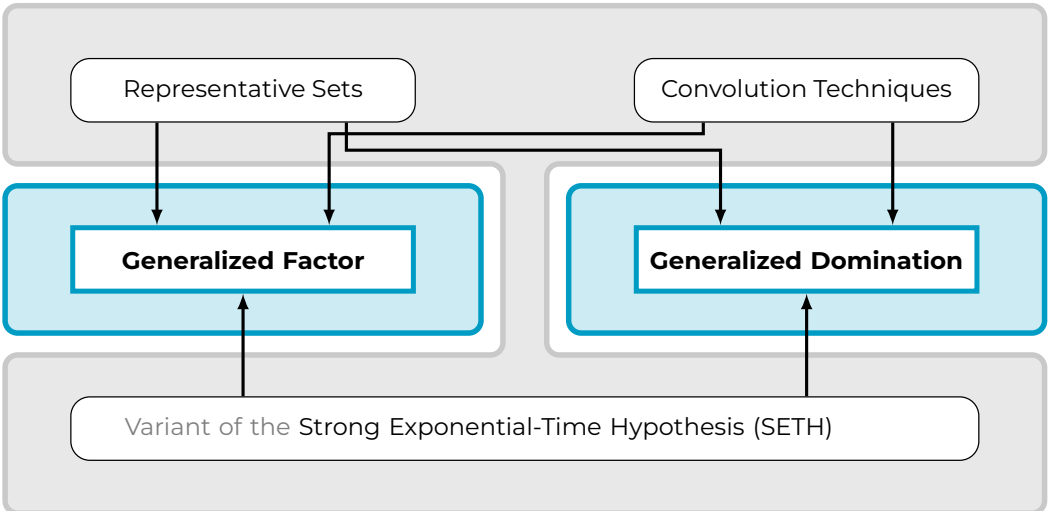
Outline



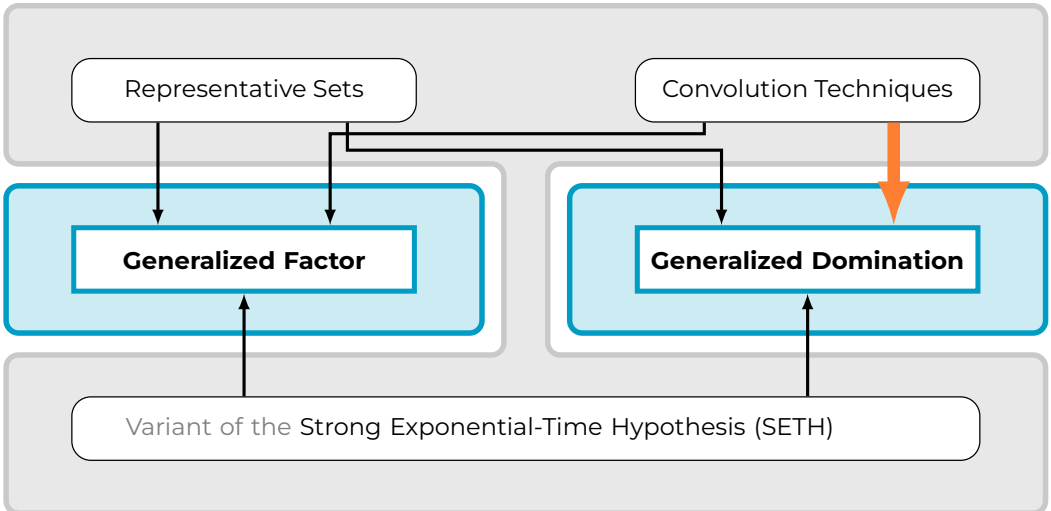
Outline


Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination

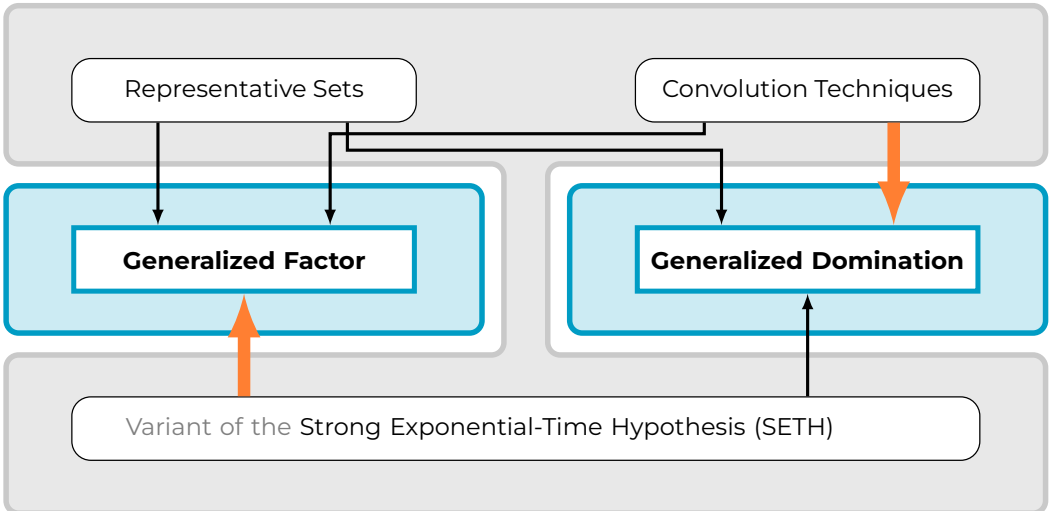
Outline


Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination
Variant of the Strong Exponential-Time Hypothesis (SETH)

Outline


Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination
Variant of the Strong Exponential-Time Hypothesis (SETH)

Outline

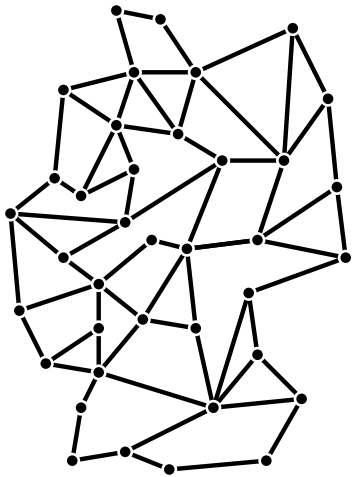

Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination
Variant of the Strong Exponential-Time Hypothesis (SETH)

Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

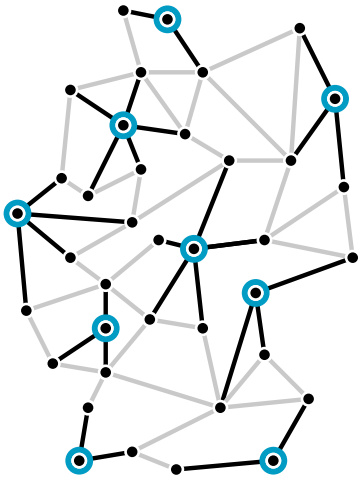
Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$



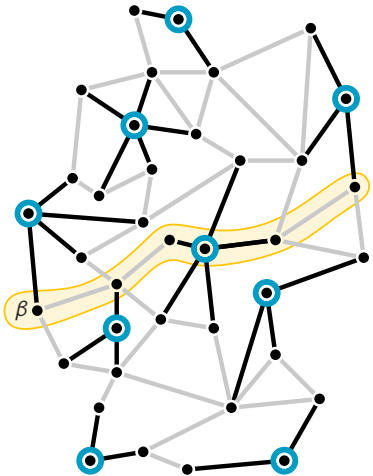
Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$



Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$



Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

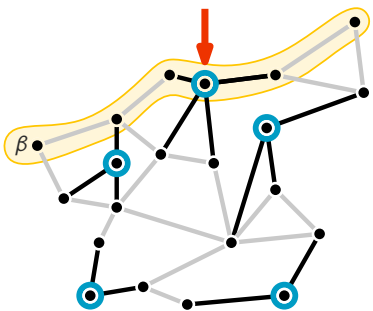
Each vertex in β can be

Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

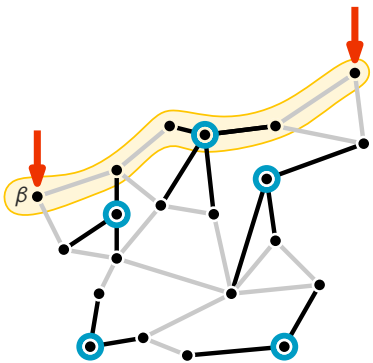
Each vertex in β can be

1. *selected* with 0 selected neighbors,



Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

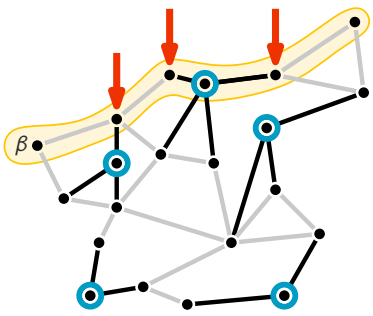


Each vertex in β can be

1. *selected* with 0 selected neighbors,
2. *unselected* with 0 selected neighbors, or

Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$



Each vertex in β can be

1. *selected* with 0 selected neighbors,
2. *unselected* with 0 selected neighbors, or
3. *unselected* with 1 selected neighbor.

Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

Each vertex in β can be

1. *selected* with 0 selected neighbors,
2. *unselected* with 0 selected neighbors, or
3. *unselected* with 1 selected neighbor.

- Otherwise, solution is invalid!

Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

Each vertex in β can be

1. *selected* with 0 selected neighbors,
2. *unselected* with 0 selected neighbors, or
3. *unselected* with 1 selected neighbor.

- Otherwise, solution is invalid!

$\rightsquigarrow$ 3 states for each vertex

Solving (σ, ρ)-DomSet: Idea

Case Study: Perfect Code = (σ, ρ)-DomSet with $\sigma = \{0\}$ and $\rho = \{1\}$

Each vertex in β can be

1. *selected* with 0 selected neighbors,
2. *unselected* with 0 selected neighbors, or
3. *unselected* with 1 selected neighbor.

- Otherwise, solution is invalid!

$\rightsquigarrow$ 3 states for each vertex

$\rightsquigarrow$ $3^{|\beta|}$ states for bag β

Solving (σ, ρ)-DomSet: Algorithm

Recall: $3^{|\beta|}$ states for each bag β.

Solving (σ, ρ)-DomSet: Algorithm

Recall: $3^{|\beta|}$ states for each bag β.

The algorithm:

DP over the given tree decomposition

Solving (σ, ρ)-DomSet: Algorithm

Recall: $3^{|\beta|}$ states for each bag β.

The algorithm:

DP over the given tree decomposition

\+ Efficient convolution techniques to combine states at join nodes

Solving (σ, ρ)-DomSet: Algorithm

Recall: $3^{|\beta|}$ states for each bag β.

The algorithm:

DP over the given tree decomposition

$+$ Efficient convolution techniques to combine states at join nodes

$=$ $3^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$ algorithm

Solving (σ, ρ)-DomSet: Algorithm

Recall: $3^{|\beta|}$ states for each bag β.

The algorithm:

DP over the given tree decomposition

$+$ Efficient convolution techniques to combine states at join nodes

$=$ $3^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$ algorithm

Are $3^{\mathrm{tw}(G)}$ states actually possible?

Can we do better?

Solving (σ, ρ)-DomSet: Algorithm

Recall: $3^{|\beta|}$ states for each bag β.

The algorithm:

DP over the given tree decomposition

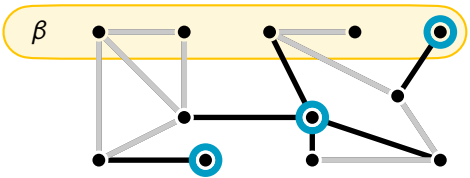
$+$ Efficient convolution techniques to combine states at join nodes

$=$ $3^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$ algorithm

Are $3^{\mathrm{tw}(G)}$ states actually possible? **No!**

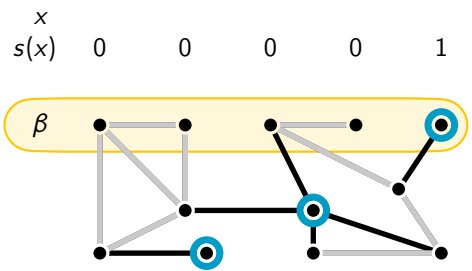
Can we do better? **Yes!**

Solving (σ, ρ)-DomSet: Structural Insights



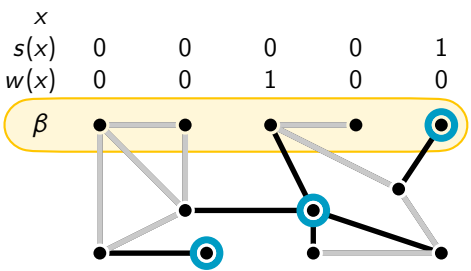
Solving (σ, ρ)-DomSet: Structural Insights

- Define $\{0, 1\}$-vector encoding selection status $\rightsquigarrow$ selection-vector s



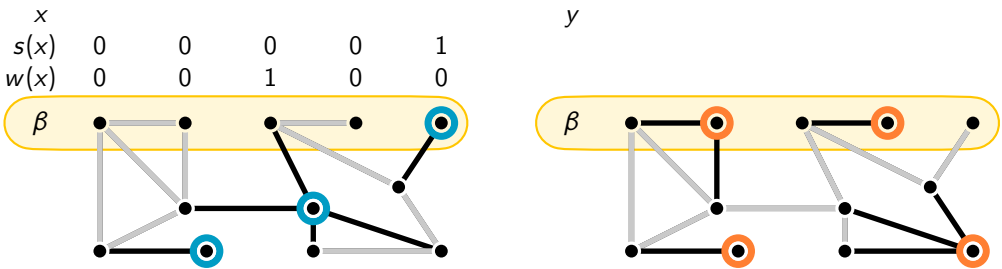
Solving (σ, ρ)-DomSet: Structural Insights

- Define $\{0,1\}$-vector encoding selection status $\rightsquigarrow$ selection-vector s
- Define vector with number of selected neighbors $\rightsquigarrow$ weight-vector w



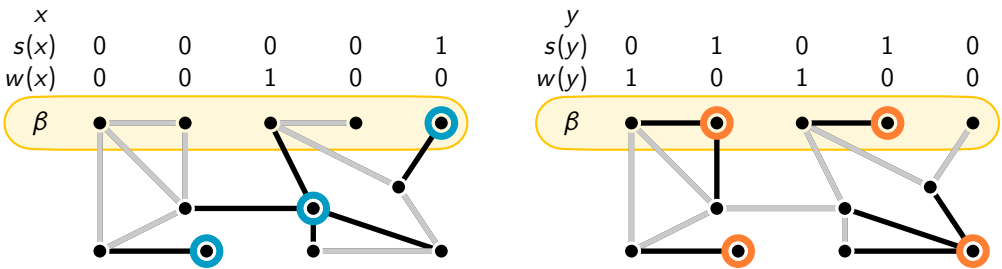
Solving (σ, ρ)-DomSet: Structural Insights

- Define $\{0,1\}$-vector encoding selection status $\rightsquigarrow$ selection-vector s
- Define vector with number of selected neighbors $\rightsquigarrow$ weight-vector w
- Consider two solutions:



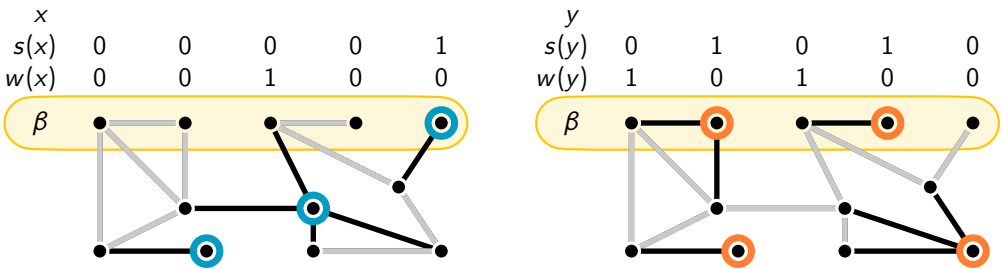
Solving (σ, ρ)-DomSet: Structural Insights

- Define $\{0,1\}$-vector encoding selection status $\rightsquigarrow$ selection-vector s
- Define vector with number of selected neighbors $\rightsquigarrow$ weight-vector w
- Consider two solutions:



Solving (σ, ρ)-DomSet: Structural Insights

- Define $\{0,1\}$-vector encoding selection status $\rightsquigarrow$ selection-vector s
- Define vector with number of selected neighbors $\rightsquigarrow$ weight-vector w
- Consider two solutions:



- Vectors are orthogonal: $s(x) \cdot w(y) = s(y) \cdot w(x) = 0$

$\rightsquigarrow$ Not too many vectors/solutions can satisfy this property

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Problem: Convolution still considers all $3^{\mathrm{tw}(G)}$ possible states.

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Problem: Convolution still considers all $3^{\mathrm{tw}(G)}$ possible states.

Adjusted algorithm:

1. Use orthogonality to compress vectors

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Problem: Convolution still considers all $3^{\mathrm{tw}(G)}$ possible states.

Adjusted algorithm:

1. Use orthogonality to compress vectors
2. Use convolution techniques to obtain a faster DP

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Problem: Convolution still considers all $3^{\mathrm{tw}(G)}$ possible states.

Adjusted algorithm:

1. Use orthogonality to compress vectors
2. Use convolution techniques to obtain a faster DP
3. Use orthogonality to decompress vectors again

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Problem: Convolution still considers all $3^{\mathrm{tw}(G)}$ possible states.

Adjusted algorithm:

1. Use orthogonality to compress vectors
2. Use convolution techniques to obtain a faster DP
3. Use orthogonality to decompress vectors again

$\rightsquigarrow$ Algorithm with runtime $\mathbf{2}^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$ before: $\mathbf{3}^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$

Solving (σ, ρ)-DomSet: A Faster Algorithm

Key Result

Only $2^{\mathrm{tw}(G)}$ of the $3^{\mathrm{tw}(G)}$ states might have solutions!

Problem: Convolution still considers all $3^{\mathrm{tw}(G)}$ possible states.

Adjusted algorithm:

1. Use orthogonality to compress vectors
2. Use convolution techniques to obtain a faster DP
3. Use orthogonality to decompress vectors again

$\rightsquigarrow$ Algorithm with runtime $\mathbf{2}^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$ before: $\mathbf{3}^{\mathrm{tw}(G)} \cdot \mathrm{poly}(|G|)$

We extend this to other sets as well!

Solving (σ, ρ)-DomSet: Our Contribution

For finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$, we define a constant $c_{\sigma,\rho}$ such that

Solving (σ, ρ)-DomSet: Our Contribution

For finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$, we define a constant $c_{\sigma,\rho}$ such that

Theorem

We solve (σ, ρ)-DomSet in time $(c_{\sigma,\rho})^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$
if a tree decomposition of width tw is given.

Solving (σ, ρ)-DomSet: Our Contribution

For finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$, we define a constant $c_{\sigma,\rho}$ such that

Theorem

We solve (σ, ρ)-DomSet in time $(c_{\sigma,\rho})^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$
if a tree decomposition of width tw is given.

Algorithm works for

- (exact) decision version,
- minimization and maximization version, and
- counting version.

Solving (σ, ρ)-DomSet: Our Contribution

For finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$, we define a constant $c_{\sigma,\rho}$ such that

Theorem

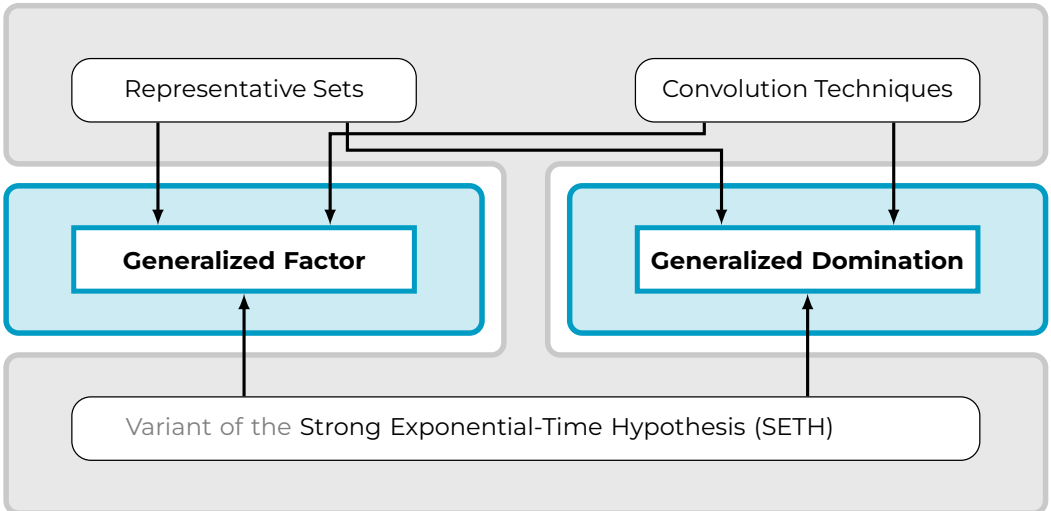
We solve (σ, ρ)-DomSet in time $(c_{\sigma,\rho})^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$
if a tree decomposition of width tw is given.

Algorithm works for

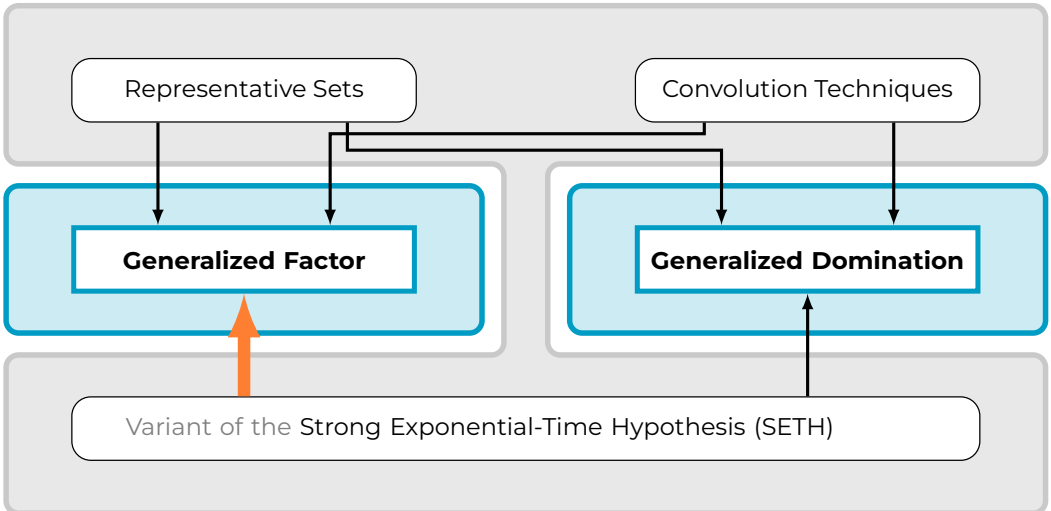
- (exact) decision version,
- minimization and maximization version, and
- counting version.

Next goal: Check if constant $c_{\sigma,\rho}$ is optimal.

Outline


Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination
Variant of the Strong Exponential-Time Hypothesis (SETH)

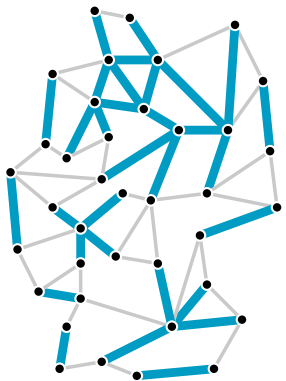
Outline


Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination
Variant of the Strong Exponential-Time Hypothesis (SETH)

Λ-Factor: Upper Bound

For simplicity, consider finite sets for now.

Each vertex can have $\max \Lambda + 1$ intermediate states.



$\Lambda = \{1, 4\}$

Λ-Factor: Upper Bound

For simplicity, consider finite sets for now.

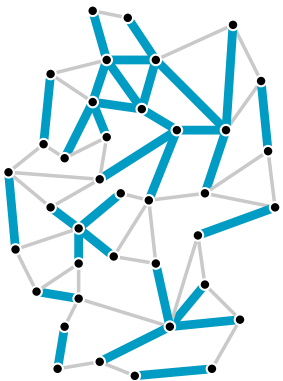
Each vertex can have $\max \Lambda + 1$ intermediate states.

Theorem

For every finite set $\Lambda \subseteq \mathbb{N}$:

We solve Λ-Factor in time $(\max \Lambda + 1)^{\mathsf{tw}} \cdot \mathsf{poly}(|G|)$

if a tree decomposition of width tw is given.



$\Lambda = \{1, 4\}$

Λ-Factor: Upper Bound

For simplicity, consider finite sets for now.

Each vertex can have $\max \Lambda + 1$ intermediate states.

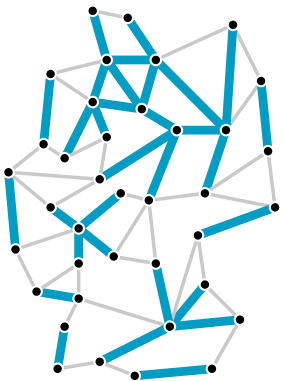
Theorem

For every finite set $\Lambda \subseteq \mathbb{N}$:

We solve Λ-Factor in time $(\max \Lambda + 1)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ if a tree decomposition of width tw is given.

Prove:

No algorithm with running time $(\max \Lambda + 1 - \varepsilon)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ exists.



$\Lambda = \{1, 4\}$

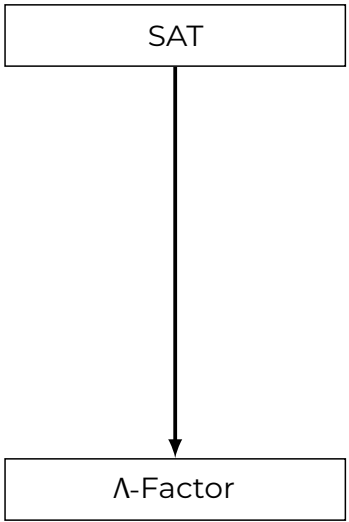
Λ-Factor: Idea for the Lower Bound

- *Un*conditionally extremely difficult

Λ-Factor: Idea for the Lower Bound

- *Un*conditionally extremely difficult
- Use difficulty of well-studied problems

⇝ Suitable reduction gives good lower bound



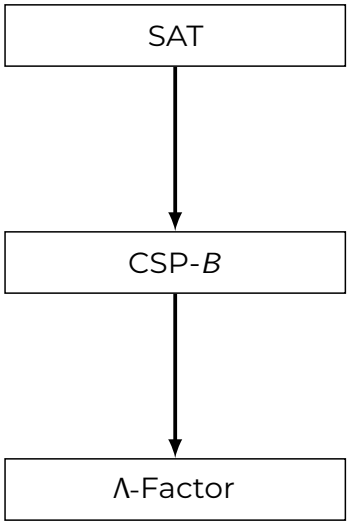
Λ-Factor: Idea for the Lower Bound

- *Un*conditionally extremely difficult
- Use difficulty of well-studied problems

⇝ Suitable reduction gives good lower bound

- Use better suited, more flexible CSP-*B* problem (variables can take *B* values)

  [Lampis 2020 & 2025]



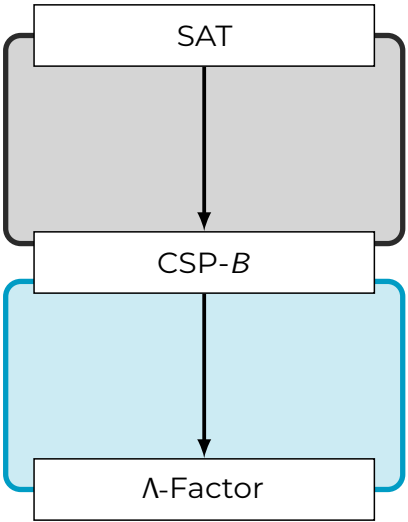
Λ-Factor: Idea for the Lower Bound

- *Un*conditionally extremely difficult
- Use difficulty of well-studied problems

⇝ Suitable reduction gives good lower bound

- Use better suited, more flexible CSP-*B* problem (variables can take *B* values) [Lampis 2020 & 2025]

⇝ Simplified reductions that are easier to understand



Λ-Factor: Steps of the Lower Bound

Prove: No algorithm with running time $(\max \Lambda + 1 - \varepsilon)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ exists.

Λ-Factor: Steps of the Lower Bound

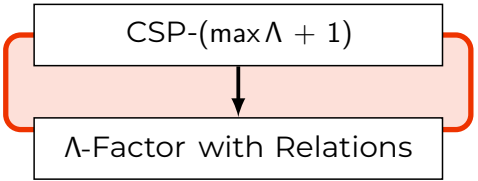
Prove: No algorithm with running time $(\max \Lambda + 1 - \varepsilon)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ exists.

CSP-$(\max \Lambda + 1)$

Λ-Factor

Λ-Factor: Steps of the Lower Bound

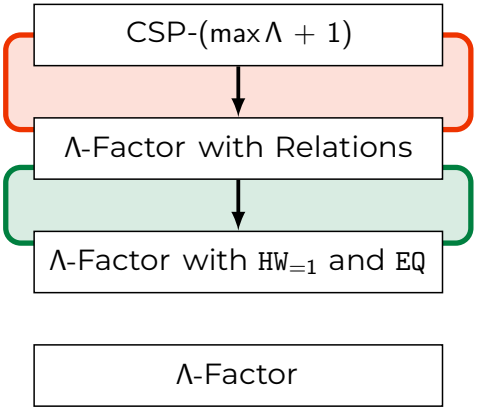
Prove: No algorithm with running time $(\max \Lambda + 1 - \varepsilon)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ exists.



Encode the variable assignment by edge selections

Λ-Factor: Steps of the Lower Bound

Prove: No algorithm with running time $(\max \Lambda + 1 - \varepsilon)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ exists.

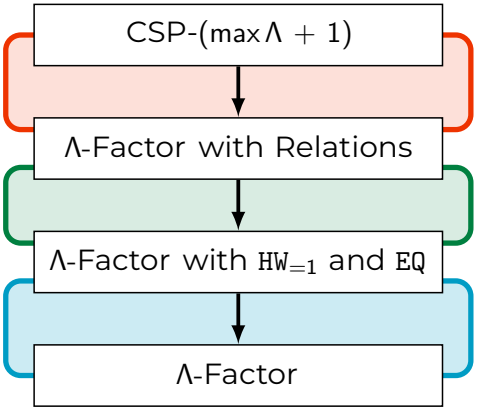


Encode the variable assignment by edge selections

Model general relations by two simple relations

Λ-Factor: Steps of the Lower Bound

Prove: No algorithm with running time $(\max \Lambda + 1 - \varepsilon)^{\mathrm{tw}} \cdot \mathrm{poly}(|G|)$ exists.



Encode the variable assignment
by edge selections

Model general relations
by two simple relations

Exploit properties of Λ
to realize the remaining relations

Λ-Factor: Our Contribution

Fix a finite or cofinite set $\Lambda \subseteq \mathbb{N}$.

Define $\operatorname{top}\Lambda := \max\Lambda$ if Λ is finite and $\operatorname{top}\Lambda := \max(\mathbb{Z} \setminus \Lambda) + 1$ if Λ is cofinite.

Example: $\operatorname{top}\{1, 4\} = 4$ and $\operatorname{top}\{1, 4, 5, 6, \dots\} = 4$

Λ-Factor: Our Contribution

Fix a finite or cofinite set $\Lambda \subseteq \mathbb{N}$.

Define $\operatorname{top}\Lambda := \max\Lambda$ if Λ is finite and $\operatorname{top}\Lambda := \max(\mathbb{Z} \setminus \Lambda) + 1$ if Λ is cofinite.

Theorem (Upper Bound)

We solve Λ-Factor in time $(\operatorname{top}\Lambda + 1)^{\mathrm{tw}} \cdot \operatorname{poly}(|G|)$ if a tree decomposition of width tw is given.

Λ-Factor: Our Contribution

Fix a finite or cofinite set $\Lambda \subseteq \mathbb{N}$.

Define $\operatorname{top}\Lambda := \max\Lambda$ if Λ is finite and $\operatorname{top}\Lambda := \max(\mathbb{Z} \setminus \Lambda) + 1$ if Λ is cofinite.

Theorem (Upper Bound)

We solve the *counting version* of Λ-Factor in time $(\operatorname{top}\Lambda + 1)^{\mathsf{tw}} \cdot \operatorname{poly}(|G|)$ if a tree decomposition of width tw is given.

Λ-Factor: Our Contribution

Fix a finite or cofinite set $\Lambda \subseteq \mathbb{N}$.

Define $\operatorname{top}\Lambda := \max\Lambda$ if Λ is finite and $\operatorname{top}\Lambda := \max(\mathbb{Z} \setminus \Lambda) + 1$ if Λ is cofinite.

Theorem (Upper Bound)

We solve the *counting version* of Λ-Factor in time $(\operatorname{top}\Lambda + 1)^{\mathsf{tw}} \cdot \operatorname{poly}(|G|)$ if a tree decomposition of width tw is given.

Theorem (Lower Bound)

Unless #SETH fails, Λ-#Factor has no $(\operatorname{top}\Lambda + 1 - \varepsilon)^{\mathsf{tw}} \cdot \operatorname{poly}(|G|)$ algorithm for non-poly-time sets even if a tree decomposition of width tw is given.

Λ-Factor: Our Contribution

Fix a finite or cofinite set $\Lambda \subseteq \mathbb{N}$.

Define $\operatorname{top}\Lambda := \max\Lambda$ if Λ is finite and $\operatorname{top}\Lambda := \max(\mathbb{Z} \setminus \Lambda) + 1$ if Λ is cofinite.

Theorem (Upper Bound)

We solve the *counting version* of Λ-Factor in time $(\operatorname{top}\Lambda + 1)^{\mathsf{tw}} \cdot \operatorname{poly}(|G|)$ if a tree decomposition of width tw is given.

Theorem (Lower Bound)

Unless #SETH fails, Λ-#Factor has no $(\operatorname{top}\Lambda + 1 - \varepsilon)^{\mathsf{tw}} \cdot \operatorname{poly}(|G|)$ algorithm for non-poly-time sets even if a tree decomposition of width tw is given.

Decision version:

- **Finite set:** Similar lower bound
- **Cofinite set:** Improved algorithm but no matching lower bound

(σ, ρ)-DomSet: Our Contribution

Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$.

Theorem (Upper Bound)

We solve the *counting version* of (σ, ρ)-DomSet in time $(c_{\sigma,\rho})^{\mathsf{tw}} \cdot \mathrm{poly}(|G|)$ if a tree decomposition of width tw is given.

(σ, ρ)-DomSet: Our Contribution

Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$.

Theorem (Upper Bound)

We solve the *counting version* of (σ, ρ)-DomSet in time $(c_{\sigma,\rho})^{\mathsf{tw}} \cdot \mathrm{poly}(|G|)$ if a tree decomposition of width tw is given.

Theorem (Lower Bound)

Unless #SETH fails, (σ, ρ)-#DomSet has no $(c_{\sigma,\rho} - \varepsilon)^{\mathsf{tw}} \cdot \mathrm{poly}(|G|)$ algorithm for non-poly-time sets even if a tree decomposition of width tw is given.

(σ, ρ)-DomSet: Our Contribution

Fix two finite or cofinite sets $\sigma, \rho \subseteq \mathbb{N}$.

Theorem (Upper Bound)

We solve the *counting version* of (σ, ρ)-DomSet in time $(c_{\sigma,\rho})^{\mathsf{tw}} \cdot \mathrm{poly}(|G|)$ if a tree decomposition of width tw is given.

Theorem (Lower Bound)

Unless #SETH fails, (σ, ρ)-#DomSet has no $(c_{\sigma,\rho} - \varepsilon)^{\mathsf{tw}} \cdot \mathrm{poly}(|G|)$ algorithm for non-poly-time sets even if a tree decomposition of width tw is given.

Decision version:

- **Finite sets:** Similar lower bound
- **Cofinite sets:** Improved algorithm but *no* lower bound

Summary

Problem		Deciding
Λ-Factor	Λ finite	tight
	Λ cofinite	improved algorithm, gap remains

Summary

Problem		Deciding
Λ-Factor	Λ finite	tight
	Λ cofinite	improved algorithm, gap remains
(σ, ρ)-DomSet	σ and ρ finite	tight
	σ or ρ cofinite	improved algorithm, no lower bound

Summary

Problem		Deciding	Counting
Λ-Factor	Λ finite	tight	tight
	Λ cofinite	improved algorithm, gap remains	
(σ, ρ)-DomSet	σ and ρ finite	tight	tight
	σ or ρ cofinite	improved algorithm, no lower bound	

Summary

Problem		Deciding	Counting
Λ-Factor	Λ finite	tight	tight
	Λ cofinite	improved algorithm, gap remains	
(σ, ρ)-DomSet	σ and ρ finite	tight	tight
	σ or ρ cofinite	improved algorithm, no lower bound	

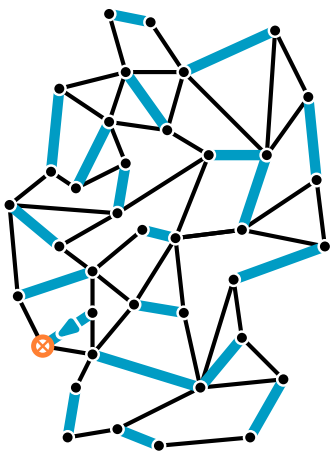
Open questions: Close gaps? More general sets? Other parameters?

Fire Protection Train (FPT) Problem

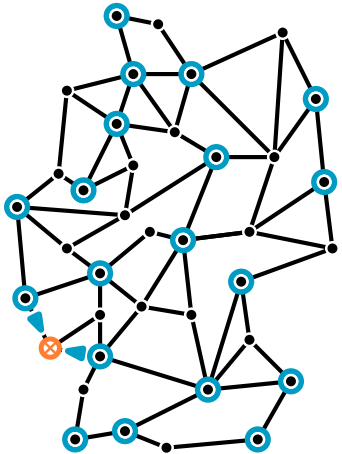
Task:

Find (few) locations to position the fire trains at.

Moving Train

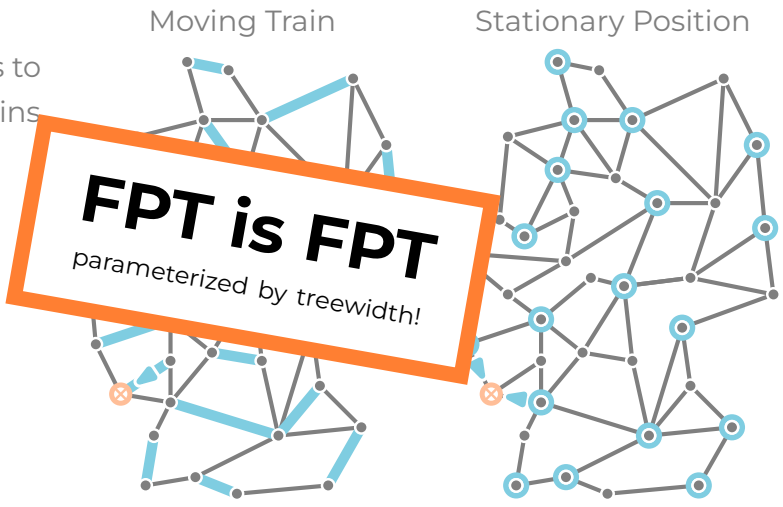


Stationary Position



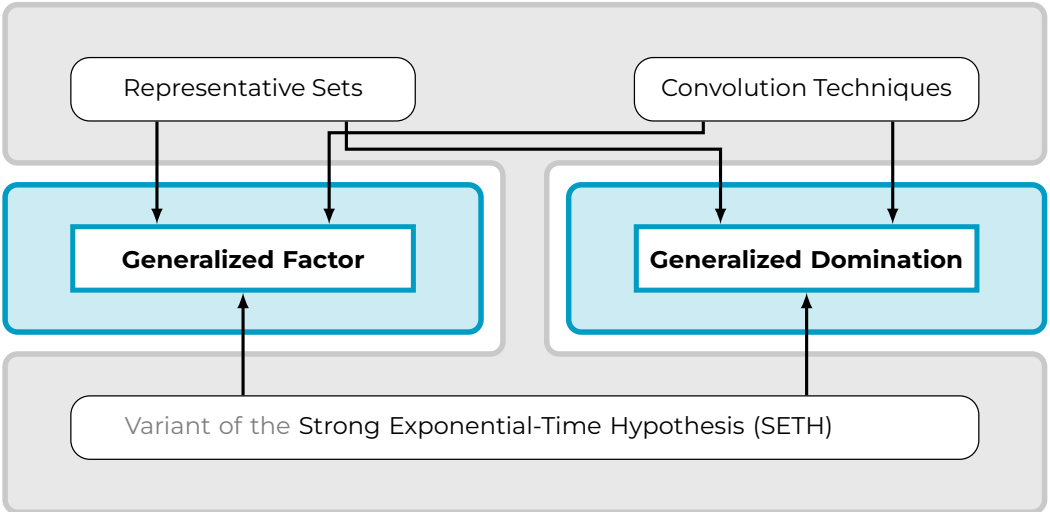
Fire Protection Train (FPT) Problem

Task:
Find (few) locations to position the fire trains at.



Appendix

Outline


Representative Sets
Convolution Techniques
Generalized Factor
Generalized Domination
Variant of the Strong Exponential-Time Hypothesis (SETH)

(σ, ρ)-DomSet: Structured Pairs

m-structured (σ, ρ)

For $m \geq 2$, (σ, ρ) is m-structured if there are α and β such that

for all $s \in \sigma$ we have $s \equiv \alpha \bmod m$ and for all $r \in \rho$ we have $r \equiv \beta \bmod m$.

(σ, ρ)-DomSet: Structured Pairs

m-structured (σ, ρ)

For $m \geq 2$, (σ, ρ) is m-structured if there are α and β such that
for all $s \in \sigma$ we have $s \equiv \alpha \bmod m$ and for all $r \in \rho$ we have $r \equiv \beta \bmod m$.

Examples:

σ	ρ	m-structured for
$\{1, 3\}$	$\{4\}$	$m = 2$
$\{0, 4\}$	$\{1, 9\}$	$m = 2, 4$
$\{0\}$	$\{1\}$	every $m \geq 2$
$\{0, 3\}$	$\{1, 5\}$	no $m \geq 2$
$\{d\}$	$\mathbb{N}$	no $m \geq 2$

(σ, ρ)-DomSet: Structured Pairs

m-structured (σ, ρ)

For $m \geq 2$, (σ, ρ) is m-structured if there are α and β such that
for all $s \in \sigma$ we have $s \equiv \alpha \bmod m$ and for all $r \in \rho$ we have $r \equiv \beta \bmod m$.

Examples:

σ	ρ	m-structured for
$\{1, 3\}$	$\{4\}$	$m = 2$
$\{0, 4\}$	$\{1, 9\}$	$m = 2, 4$
$\{0\}$	$\{1\}$	every $m \geq 2$
$\{0, 3\}$	$\{1, 5\}$	no $m \geq 2$
$\{d\}$	$\mathbb{N}$	no $m \geq 2$

If (σ, ρ) is m-structured for some $m \geq 2$, we show an improved algorithm.

(σ, ρ)-DomSet: Our Contribution (Formalized)

Definition for finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$

- $c_{\sigma,\rho} \coloneqq \operatorname{top}\sigma + \operatorname{top}\rho + 2$ if (σ, ρ) is not $\mathtt{m}$-structured,
- $c_{\sigma,\rho} \coloneqq \max(\operatorname{top}\sigma, \operatorname{top}\rho) + 2$ if $\operatorname{top}\sigma = \operatorname{top}\rho$ is even and (σ, ρ) is 2-structured but not $\mathtt{m} \geq 3$-structured, and
- $c_{\sigma,\rho} \coloneqq \max(\operatorname{top}\sigma, \operatorname{top}\rho) + 1$ otherwise.

(σ, ρ)-DomSet: Our Contribution (Formalized)

Definition for finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$

- $c_{\sigma,\rho} := \operatorname{top}\sigma + \operatorname{top}\rho + 2$ if (σ, ρ) is not $\mathtt{m}$-structured,
- $c_{\sigma,\rho} := \max(\operatorname{top}\sigma, \operatorname{top}\rho) + 2$ if $\operatorname{top}\sigma = \operatorname{top}\rho$ is even and (σ, ρ) is 2-structured but not $\mathtt{m} \geq 3$-structured, and
- $c_{\sigma,\rho} := \max(\operatorname{top}\sigma, \operatorname{top}\rho) + 1$ otherwise.

Theorem (Upper Bound)

We can solve (σ, ρ)-#DomSet in time $(c_{\sigma,\rho})^{\mathtt{tw}} \cdot \operatorname{poly}(|G|)$

if a tree decomposition of width $\mathtt{tw}$ is given.

(σ, ρ)-DomSet: Our Contribution (Formalized)

Definition for finite or cofinite $\sigma, \rho \subseteq \mathbb{N}$

- $c_{\sigma,\rho} := \operatorname{top}\sigma + \operatorname{top}\rho + 2$ if (σ, ρ) is not $\mathtt{m}$-structured,
- $c_{\sigma,\rho} := \max(\operatorname{top}\sigma, \operatorname{top}\rho) + 2$ if $\operatorname{top}\sigma = \operatorname{top}\rho$ is even and (σ, ρ) is 2-structured but not $\mathtt{m} \geq 3$-structured, and
- $c_{\sigma,\rho} := \max(\operatorname{top}\sigma, \operatorname{top}\rho) + 1$ otherwise.

Theorem (Upper Bound)

We can solve (σ, ρ)-#DomSet in time $(c_{\sigma,\rho})^{\mathtt{tw}} \cdot \operatorname{poly}(|G|)$

if a tree decomposition of width $\mathtt{tw}$ is given.

Theorem (Lower Bound)

Unless #SETH fails, (σ, ρ)-#DomSet has no $(c_{\sigma,\rho} - \varepsilon)^{\mathtt{tw}} \cdot \operatorname{poly}(|G|)$ algorithm

for non-trivial sets, even if a tree decomposition of width $\mathtt{tw}$ is given.

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

- Fix a vertex v.

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

- Fix a vertex v.
- Fix three solutions with degree 0, 256, and 1024 for v, respectively.

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

- Fix a vertex v.
- Fix three solutions with degree 0, 256, and 1024 for v, respectively.
- A future extension increases the degree of v by f.

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

- Fix a vertex v.
- Fix three solutions with degree 0, 256, and 1024 for v, respectively.
- A future extension increases the degree of v by f.
- One of these three solutions can be combined with f as $\{0 + f, 256 + f, 1024 + f\} \neq \{0, 1024\} = \mathbb{N} \setminus \Lambda$.

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

- Fix a vertex v.
- Fix three solutions with degree 0, 256, and 1024 for v, respectively.
- A future extension increases the degree of v by f.
- One of these three solutions can be combined with f as $\{0 + f, 256 + f, 1024 + f\} \neq \{0, 1024\} = \mathbb{N} \setminus \Lambda$.

$\rightsquigarrow$ $|\mathbb{N} \setminus \Lambda| + 1 = |\{0, 1024\}| = \mathbf{3}$ solutions suffice instead of $\mathbf{1026}$!

Representative Sets: Idea

Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 1024\}$: $1026^{\mathsf{tw}(G)} \cdot \mathsf{poly}(|G|)$ algorithm.

Can we improve this? Are *all* partial solutions relevant?

- Fix a vertex v.
- Fix three solutions with degree 0, 256, and 1024 for v, respectively.
- A future extension increases the degree of v by f.
- One of these three solutions can be combined with f as $\{0 + f, 256 + f, 1024 + f\} \neq \{0, 1024\} = \mathbb{N} \setminus \Lambda$.

$\rightsquigarrow$ $|\mathbb{N} \setminus \Lambda| + 1 = |\{0, 1024\}| = \mathbf{3}$ solutions suffice instead of $\mathbf{1026}$!

Idea: Exploit this for a faster algorithm.

Representative Sets

- Store only a *representative set* of the possible partial solutions!
  $\rightsquigarrow (|\mathbb{N} \setminus \Lambda| + 1)^{\mathsf{tw}}$ solutions suffice instead of $(\max(\mathbb{N} \setminus \Lambda) + 2)^{\mathsf{tw}}$.

Representative Sets

- Store only a *representative set* of the possible partial solutions!
  $\rightsquigarrow (|\mathbb{N} \setminus \Lambda| + 1)^{\mathsf{tw}}$ solutions suffice instead of $(\max(\mathbb{N} \setminus \Lambda) + 2)^{\mathsf{tw}}$.
- Combine this with classical DP.

Representative Sets

- Store only a *representative set* of the possible partial solutions!
  $\rightsquigarrow (|\mathbb{N} \setminus \Lambda| + 1)^{\mathsf{tw}}$ solutions suffice instead of $(\max(\mathbb{N} \setminus \Lambda) + 2)^{\mathsf{tw}}$.
- Combine this with classical DP.

Theorem

Fix a finite set $X \subseteq \mathbb{N}$. We can solve $(\mathbb{N} \setminus X)$-Factor in time $(|X| + 1)^{4\mathsf{tw}} \cdot \mathsf{poly}(|G|)$ if a tree decomposition of width tw is given.

Representative Sets

- Store only a *representative set* of the possible partial solutions!
  $\rightsquigarrow (|\mathbb{N} \setminus \Lambda| + 1)^{\mathsf{tw}}$ solutions suffice instead of $(\max(\mathbb{N} \setminus \Lambda) + 2)^{\mathsf{tw}}$.
- Combine this with classical DP.

Theorem

Fix a finite set $X \subseteq \mathbb{N}$. We can solve $(\mathbb{N} \setminus X)$-Factor in time $(|X| + 1)^{4\mathsf{tw}} \cdot \mathsf{poly}(|G|)$ if a tree decomposition of width tw is given.

Algorithm improves from 1026^{tw} to $3^{4\mathsf{tw}} = 81^{\mathsf{tw}}$.

Representative Sets

- Store only a *representative set* of the possible partial solutions!
  $\rightsquigarrow (|\mathbb{N} \setminus \Lambda| + 1)^{\mathsf{tw}}$ solutions suffice instead of $(\max(\mathbb{N} \setminus \Lambda) + 2)^{\mathsf{tw}}$.
- Combine this with classical DP.

Theorem

Fix a finite set $X \subseteq \mathbb{N}$. We can solve $(\mathbb{N} \setminus X)$-Factor in time $(|X| + 1)^{4\mathsf{tw}} \cdot \mathsf{poly}(|G|)$ if a tree decomposition of width tw is given.

Algorithm improves from 1026^{tw} to $3^{4\mathsf{tw}} = 81^{\mathsf{tw}}$.

We show a similar result for (σ, ρ)-DomSet.

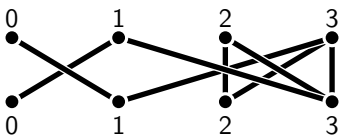
Note: Incompatible with counting version (“every solution matters”)!

Half-Induced Matchings

Definition (Half-Induced Matching)

Graph $G = (L \cup R, E)$ has a half-induced matching of size ℓ if there are $a_1, \ldots, a_\ell \in L$ and $b_1, \ldots, b_\ell \in R$ such that

- $(a_i, b_i) \in E$ for all i, and
- $(a_i, b_j) \notin E$ for all $i < j$.

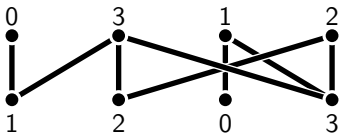


Half-Induced Matchings

Definition (Half-Induced Matching)

Graph $G = (L \cup R, E)$ has a half-induced matching of size ℓ if there are $a_1, \ldots, a_\ell \in L$ and $b_1, \ldots, b_\ell \in R$ such that

- $(a_i, b_i) \in E$ for all i, and
- $(a_i, b_j) \notin E$ for all $i < j$.

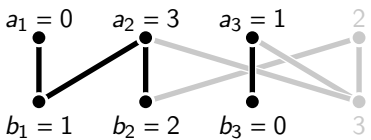


Half-Induced Matchings

Definition (Half-Induced Matching)

Graph $G = (L \cup R, E)$ has a half-induced matching of size ℓ if there are $a_1, \ldots, a_\ell \in L$ and $b_1, \ldots, b_\ell \in R$ such that

- $(a_i, b_i) \in E$ for all i, and
- $(a_i, b_j) \notin E$ for all $i < j$.

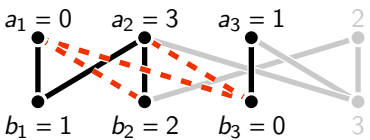


Half-Induced Matchings

Definition (Half-Induced Matching)

Graph $G = (L \cup R, E)$ has a half-induced matching of size ℓ if there are $a_1, \ldots, a_\ell \in L$ and $b_1, \ldots, b_\ell \in R$ such that

- $(a_i, b_i) \in E$ for all i, and
- $(a_i, b_j) \notin E$ for all $i < j$.

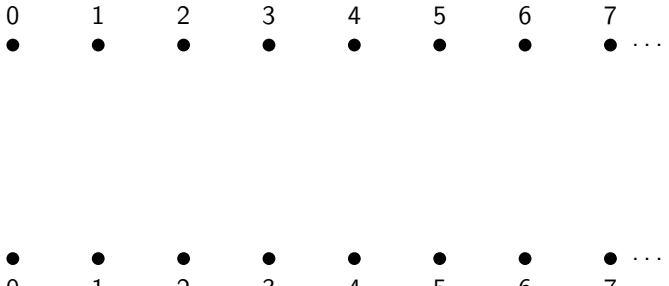


Half-induced matching of size 3

Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

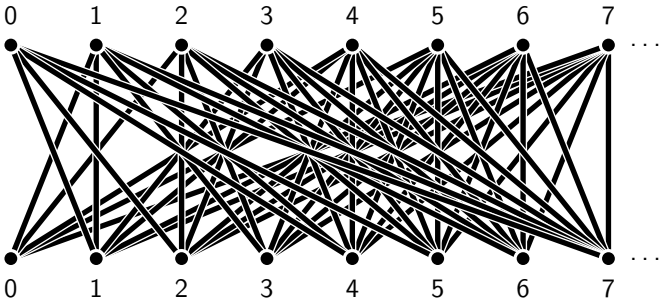
Draw edge from i to j if $i + j \in \Lambda$:



Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

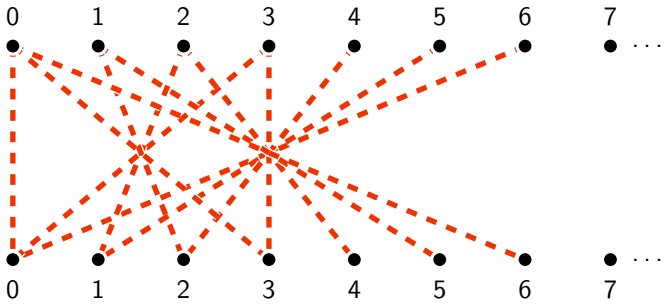
Draw edge from i to j if $i + j \in \Lambda$:



Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

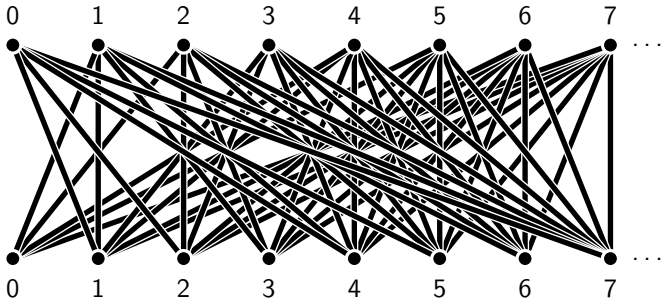
Draw edge from i to j if $i + j \in \Lambda$:



Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

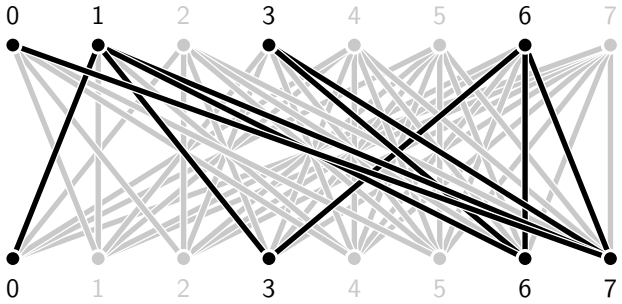
Draw edge from i to j if $i + j \in \Lambda$:



Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

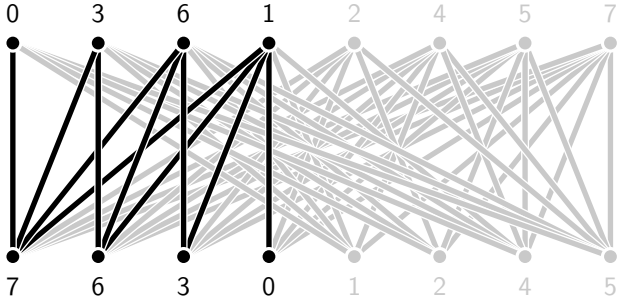
Draw edge from i to j if $i + j \in \Lambda$:



Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

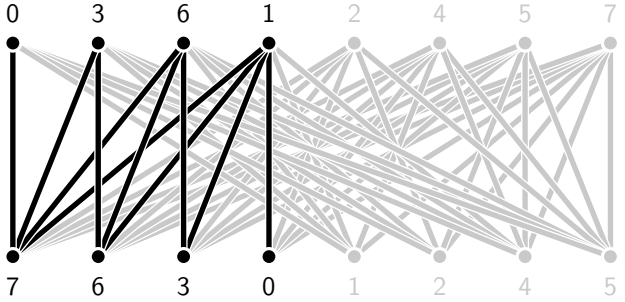
Draw edge from i to j if $i + j \in \Lambda$:



Half-Induced Matchings: Application

Consider Λ-Factor with $\Lambda = \mathbb{N} \setminus \{0, 3, 6\}$

Draw edge from i to j if $i + j \in \Lambda$:



Our observation:

A larger half-induced matching provides a better lower bound!

Convolution Techniques

Extend algorithm by van Rooij in multiple steps: [van Rooij 2020]

- 𝕊 **Cyclic convolution:**
  Extend existing algorithm by computation of prime numbers
- 𝔸 **Normal addition:**
  Use cyclic convolution and encode checksum by 𝕃 to detect overflows
- 𝕃 **Normal addition with large numbers:**
  Use cyclic convolution and encode checksum in binary to detect overflows
- 𝕋 **Truncated addition:**
  Use normal addition and zeta- and Möbius-transform

Lower Bound: Known Hypotheses

Strong Exponential Time Hypothesis (SETH) [IP 2001, CIP 2009]

For all $\delta > 0$, there is a $k \geq 3$ such that satisfiability of k-CNF formulas on n variables cannot be solved in time $(2 - \delta)^n$.

Constraint Satisfaction Hypothesis (CSPH) [Lampis 2020]

For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that q-CSP-B with n variables cannot be solved in time $(B - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$.

Primal Pathwidth SETH (pw-SETH) [Lampis 2025]

For all $\varepsilon > 0$, there is no algorithm which takes as input a 3-SAT instance ϕ and a path decomposition of its primal graph of width pw and correctly decides if ϕ is satisfiable in time $(2 - \varepsilon)^{\mathrm{pw}} \cdot |\phi|^{\mathcal{O}(1)}$.

Lower Bound: New Hypothesis

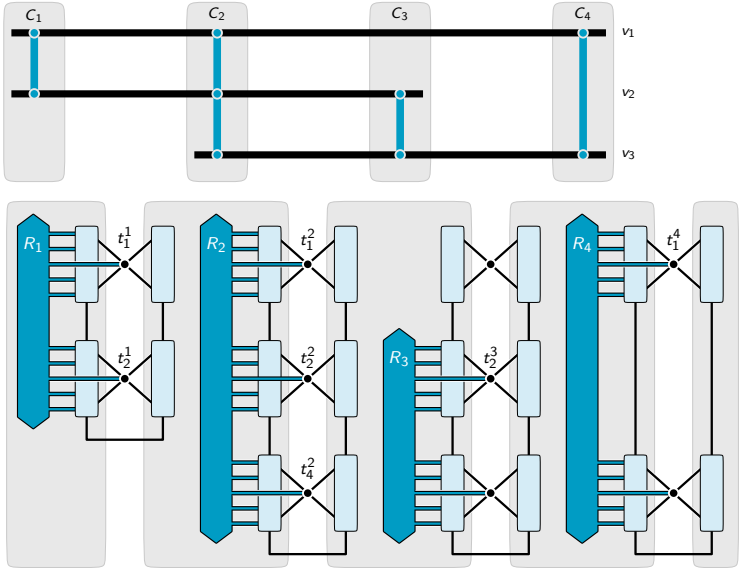
mono-pw-CSPH

For all $B \geq 2$ and $\varepsilon > 0$, there exists a q such that no algorithm can, for a q-CSP-B instance ϕ that is given with

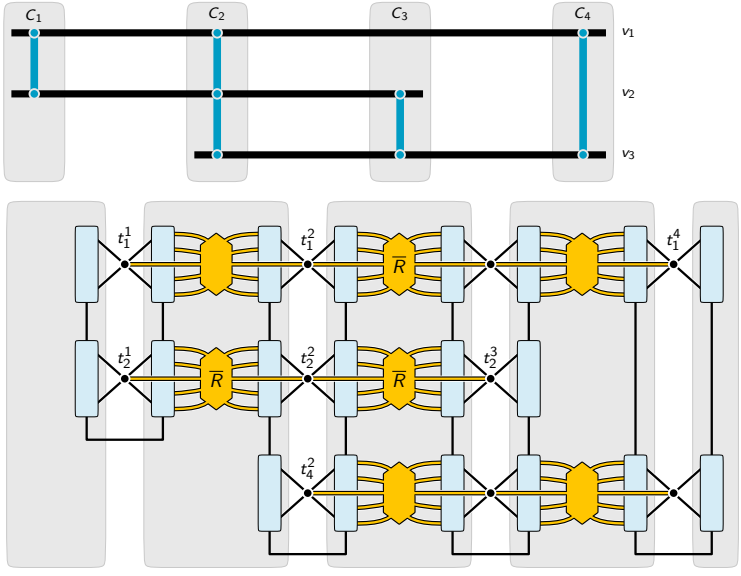
- a partition $V_1 \dot{\cup} V_2$ of the variables,
- a nice path decomposition of the primal graph of ϕ of width w
- where every bag contains at most $\mathcal{O}(B \cdot \log|V_1|)$ variables from V_2,
- an injective mapping c from the constraints of ϕ to the bags,
- the promise that every satisfying multi-assignment that is consistent on V_2 is also consistent on V_1,

decide if there is a monotone satisfying multi-assignment that is consistent on V_2 in time $(B-\varepsilon)^w \cdot |\phi|^{\mathcal{O}(1)}$.

Lower Bound: High-Level Construction



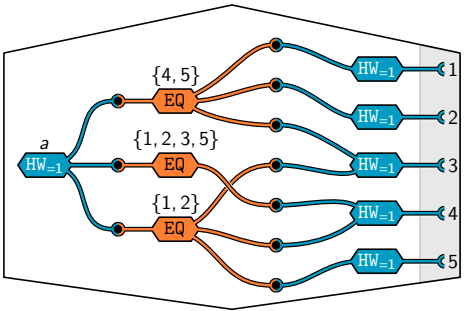
Lower Bound: High-Level Construction



Lower Bound: Realizing Relations

To realize arbitrary relations it suffices to

- force a single vertex/edge (of a group) to be selected $\rightsquigarrow \texttt{HW}_{=1}$ and
- ensure that a group of vertices/edges is consistently selected (all or none) $\rightsquigarrow \texttt{EQ}$.

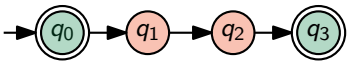


Summary (Formalized)

Problem	Algorithm	Lower Bound	
(σ, ρ)-#DomSet	$(c_{\sigma,\rho})^{\mathrm{tw}}$	no $(c_{\sigma,\rho} - \varepsilon)^{\mathrm{pw}}$	
(σ, ρ)-DomSet	$(c_{\sigma,\rho})^{\mathrm{tw}}$	no $(c_{\sigma,\rho} - \varepsilon)^{\mathrm{pw}}$	σ and ρ finite
	$(\mathrm{cost}_{\sigma,\rho})^{\mathcal{O}(\mathrm{tw})}$	open	σ or ρ cofinite
#AntiFactor$_x$	n^{tw}	no $n^{\mathrm{pw}-\varepsilon}$	
Λ-#Factor	$(\mathrm{top}\,\Lambda + 1)^{\mathrm{tw}}$	no $(\mathrm{top}\,\Lambda + 1 - \varepsilon)^{\mathrm{pw}}$	
Λ-Factor		no $(\mathrm{top}\,\Lambda + 1 - \varepsilon)^{\mathrm{pw}}$	Λ finite
		no $(h - \varepsilon)^{\mathrm{pw}}$	Λ cofinite with half-induced matching of size h
AntiFactor$_x$	$(x+1)^{\mathcal{O}(\mathrm{tw})}$	no $(x + 1 - \varepsilon)^{\mathrm{pw}}$	x degrees forbidden

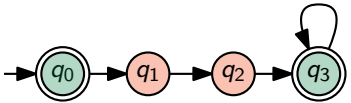
More General Sets: From States to Automata

Finite set:



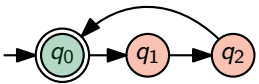
$\tau = \{0, 3\}$

Cofinite set:



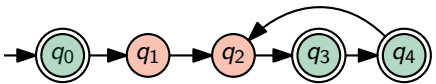
$\tau = \{0, 3, 4, 5, 6, \dots\}$

Periodic set (residue class)**:**



$\tau = \{0, 3, 6, 9, \dots\}$

Infinite set (ultimately periodic)**:**



$\tau = \{0, 3, 4, 6, 7, 9, 10, \dots\}$

Next steps:

Connection to *faster* algorithms? Efficient convolution techniques?