A Structural View on Parameterizing Problems: Distance from Triviality

Jiong Guo   Falk Hüffner   Rolf Niedermeier

11th July 2006

Parameterization for hard problems

For exact algorithms for NP-hard problems, we probably have to accept exponential runtimes.

Approach: Try to confine the combinatorial explosion to some parameter k.

Parameterization for hard problems

For exact algorithms for NP-hard problems, we probably have to accept exponential runtimes.

Approach: Try to confine the combinatorial explosion to some parameter k.

Definition

For some *parameter* k of a problem, the problem is called *fixed-parameter tractable* with respect to k if there is an algorithm that solves it in $f(k) \cdot n^{O(1)}$.

Finding Parameters

Usually, many parameters are sensible.

Example

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k, is there $V' \subseteq V$ with $|V'| \leq k$ such that each edge has at least one endpoint in V'?

- Parameterization by solution size:
  If the vertex cover has size k:
  $O(1.3^k + kn)$ time algorithm

Finding Parameters

Usually, many parameters are sensible.

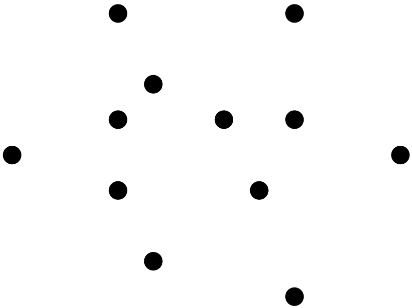
Example

Vertex Cover: Given a graph $G = (V, E)$ and an integer k, is there $V' \subseteq V$ with $|V'| \leq k$ such that each edge has at least one endpoint in V'?

- Parameterization by solution size:
  If the vertex cover has size k:
  $O(1.3^k + kn)$ time algorithm
- Parameterization by structure:
  If treewidth is bounded by w:
  $O(2^w \cdot n)$ time algorithm

2D-Traveling Salesman Problem

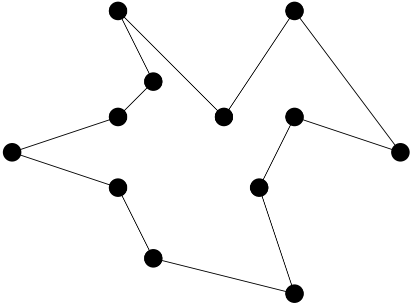
Given: n points from $\mathbf{R}^2$



2D-Traveling Salesman Problem

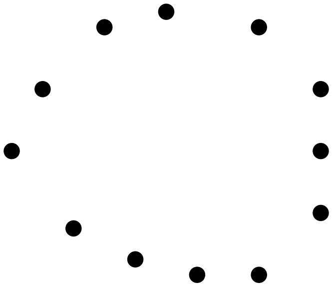
Given: n points from $\mathbf{R}^2$

Task: Find a minimal length tour through all points



Simple cases of the 2D-TRAVELING SALESMAN PROBLEM

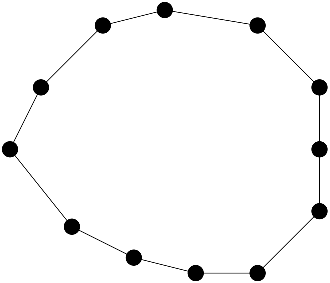
Trivial case: all vertices on the border of a convex region



Simple cases of the 2D-Traveling Salesman Problem

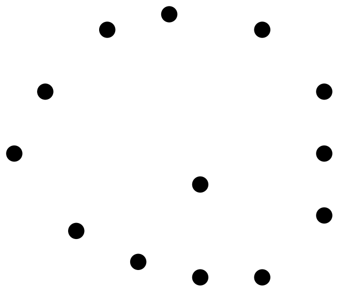
Trivial case: all vertices on the border of a convex region

- Walk all vertices in clockwise order



Simple cases of the 2D-Traveling Salesman Problem

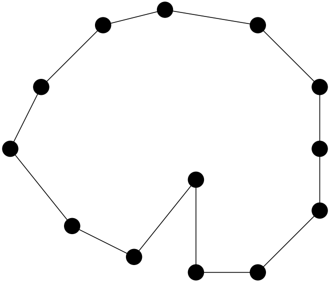
Nearly trivial case: one vertex inside the border of a convex region



Simple cases of the 2D-Traveling Salesman Problem

Nearly trivial case: one vertex inside the border of a convex region

- Few possibilities; polynomial time



Parameterized 2D-Traveling Salesman Problem

Generalized question:
How fast can we solve 2D-Traveling Salesman Problem for an instance with k points inside of the convex hull?

[Deĭneko, Hoffmann, Okamoto&Woeginger, COCOON'04]

Theorem

2D-Traveling Salesman Problem *with k inner points can be solved in $O(2^k \cdot k^2 \cdot n)$ time.*

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Is there a coloring for a graph that originates from a

- split graph by adding k edges?

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Is there a coloring for a graph that originates from a

- split graph by adding k edges? — FPT

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Is there a coloring for a graph that originates from a

- split graph by adding k edges? — FPT
- split graph by adding k vertices?

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Is there a coloring for a graph that originates from a

- split graph by adding k edges? — FPT
- split graph by adding k vertices? — W[1]-hard

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Is there a coloring for a graph that originates from a

- split graph by adding k edges? — FPT
- split graph by adding k vertices? — W[1]-hard
- bipartite graph by adding k edges?

Negative Results for Distance from Triviality Parameterization

GRAPH COLORING [LEIZHEN CAI, DISCRETE APPL. MATH. 2003]
Is there a vertex coloring of a graph with c colors such that no edge joins vertices of equal colors?

- NP-complete in general, but polynomial time solvable on split graphs and bipartite graphs

Is there a coloring for a graph that originates from a

- split graph by adding k edges? — FPT
- split graph by adding k vertices? — W[1]-hard
- bipartite graph by adding k edges? — NP-c for $k \geq 3$

Scheme for Parameterization by Distance from Triviality

Assume that we study a hard problem.

1. Determine efficiently solvable special cases
   (e.g., the restriction to special graph classes)
   —the triviality.
2. Identify useful distance measures from the triviality
   (e.g., the treewidth of a graph)
   —the (structural) parameter.

Case Studies

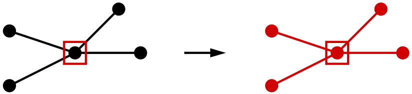
Power Dominating Set

Given a graph G, make all vertices become observed by choosing a set of vertices M to carry monitoring devices ($\square$).

Power Dominating Set

Given a graph G, make all vertices become observed by choosing a set of vertices M to carry monitoring devices ($\square$).

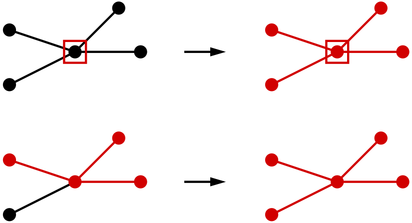
Observation rules:



Power Dominating Set

Given a graph G, make all vertices become observed by choosing a set of vertices M to carry monitoring devices (□).

Observation rules:



Power Dominating Set

- Power Dominating Set is NP-complete.

  [Haynes, Hedetniemi, Hedetniemi&Henning SIAM J. Discrete Math. 2002]

Power Dominating Set

- Power Dominating Set is NP-complete.

  [Haynes, Hedetniemi, Hedetniemi&Henning SIAM J. Discrete Math. 2002]

- Power Dominating Set is APX-hard and W[1]-hard with respect to the number of monitoring devices.

  [Kneis, Mölle, Richter&Rossmanith 2004]

  [Guo, Niedermeier&Raible FCT'05]

Power Dominating Set

- Power Dominating Set is NP-complete.

  [Haynes, Hedetniemi, Hedetniemi&Henning SIAM J. Discrete Math. 2002]
- Power Dominating Set is APX-hard and W[1]-hard with respect to the number of monitoring devices.

  [Kneis, Mölle, Richter&Rossmanith 2004]

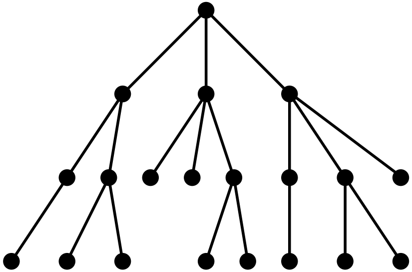
  [Guo, Niedermeier&Raible FCT'05]
- There is a linear time algorithm solving Power Dominating Set on trees.

Triviality: Trees.

Power Dominating Set on Trees

Idea for the linear time algorithm:

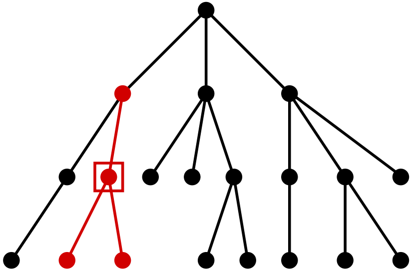
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

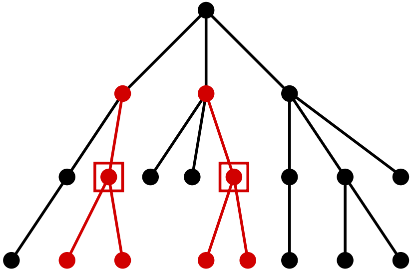
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

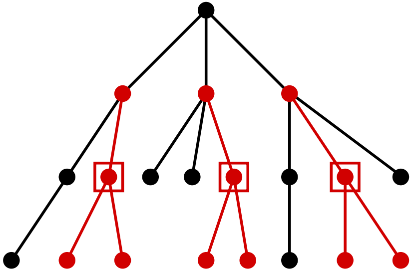
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

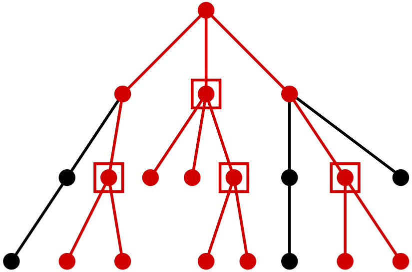
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

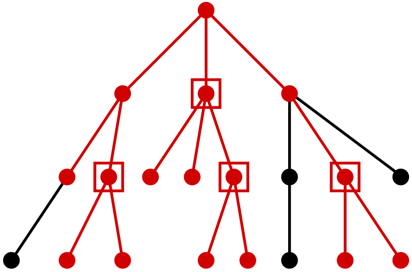
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

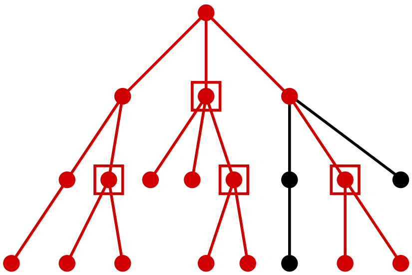
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

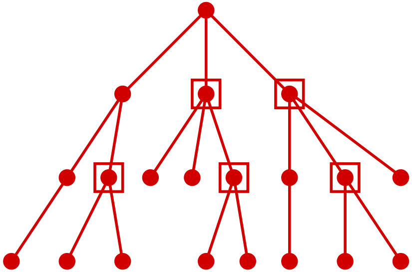
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Trees

Idea for the linear time algorithm:

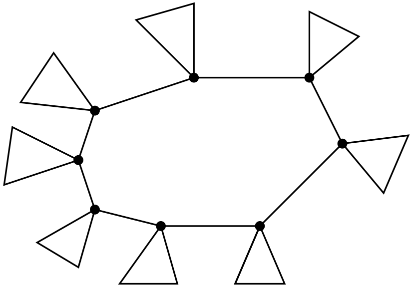
- Work layer-wise bottom-up from the leaves.
- Place a monitoring device in vertices with at least two unobserved children.



Power Dominating Set on Almost Trees

Distance from Triviality: Number of edges added.

First we consider a single added edge.

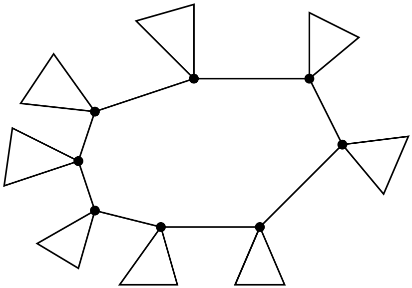


Power Dominating Set on Almost Trees

Distance from Triviality: Number of edges added.

First we consider a single added edge.

- Treat trees with linear time algorithm.
- We can prune observed edges and singleton vertices.

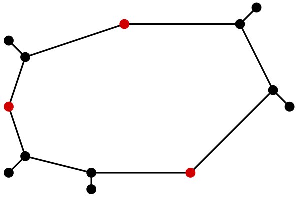


Power Dominating Set on Almost Trees

Distance from Triviality: Number of edges added.

First we consider a single added edge.

- Treat trees with linear time algorithm.
- We can prune observed edges and singleton vertices.

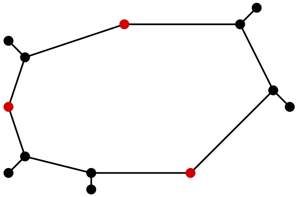


Power Dominating Set on Almost Trees

Distance from Triviality: Number of edges added.

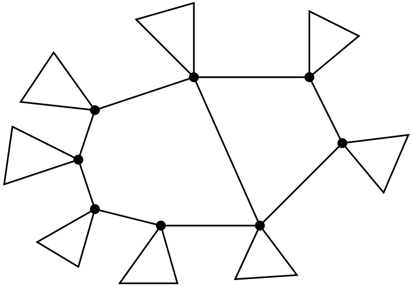
First we consider a single added edge.

- Treat trees with linear time algorithm.
- We can prune observed edges and singleton vertices.
- Branch on first vertex for placing a monitoring device, solve the rest in linear time.



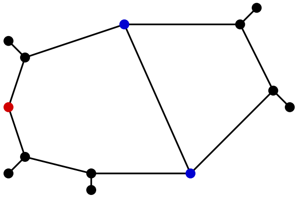
Power Dominating Set on Almost Trees

- Power Dominating Set on a tree with k edges added



Power Dominating Set on Almost Trees

- Power Dominating Set on a tree with k edges added
- After treating trees, we additionally have joints.

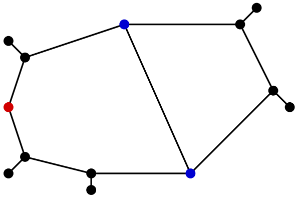


Power Dominating Set on Almost Trees

- Power Dominating Set on a tree with k edges added
- After treating trees, we additionally have joints.

Branch for each joint x:

- x gets a monitoring device
- x does not get a monitoring device
  - Branch further according to the local effect of x



Power Dominating Set on Almost Trees

Observation: The number of joints is bounded by $2k$.
Therefore, the number of branches depends only on k, not on n:

Theorem

Power Dominating Set *for a graph which originates from a tree with* k *edges added is fixed-parameter tractable with respect to* k.

Clique

Input: *A graph G and a nonnegative integer s.*
Question: *Does G contain a clique, that is, a complete subgraph, of size s?*

CLIQUE

Input: *A graph G and a nonnegative integer s.*
Question: *Does G contain a clique, that is, a complete subgraph, of size s?*

- NP-complete

CLIQUE

Input: *A graph G and a nonnegative integer s.*
Question: *Does G contain a clique, that is, a complete subgraph, of size s?*

- NP-complete
- APX-hard

Clique

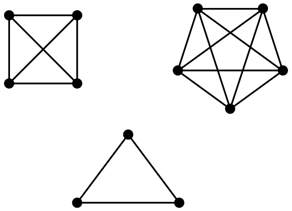
Input: *A graph G and a nonnegative integer s.*
Question: *Does G contain a clique, that is, a complete subgraph, of size s?*

- NP-complete
- APX-hard
- W[1]-hard with respect to s

CLIQUE on Cluster Graphs: Trivial Case

Definition

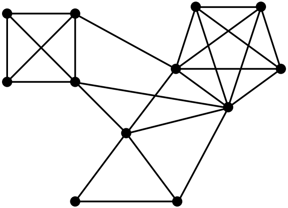
A *cluster graph* is a graph where every connected component is a clique.



Triviality: Cluster graphs.

CLIQUE on Nearly Cluster Graphs

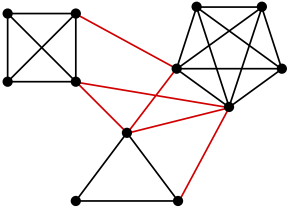
Distance from Triviality: k edges added.



Solving CLIQUE:

CLIQUE on Nearly Cluster Graphs

Distance from Triviality: k edges added.

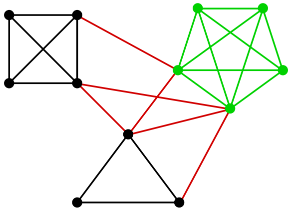


Solving CLIQUE:

- Find the k added edges: $O(1.53^k + n^3)$ time [GRAMM et al., Algorithmica 2004].

Clique on Nearly Cluster Graphs

Distance from Triviality: k edges added.

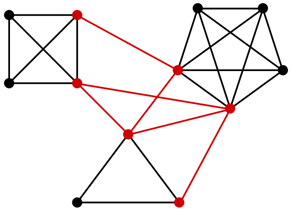


Solving Clique:

- Find the k added edges: $O(1.53^k + n^3)$ time [Gramm et al., Algorithmica 2004].
- Find the largest clique in the underlying cluster graph.

Clique on Nearly Cluster Graphs

Distance from Triviality: k edges added.

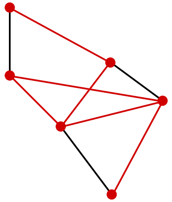


Solving Clique:

- Find the k added edges: $O(1.53^k + n^3)$ time [Gramm et al., Algorithmica 2004].
- Find the largest clique in the underlying cluster graph.
- Find the largest clique in the subgraph induced by the vertices that gained in degree: $O(1.22^{2k}) = O(1.49^k)$ time [Robson, J. Algorithms 1986].

Clique on Nearly Cluster Graphs

Distance from Triviality: k edges added.

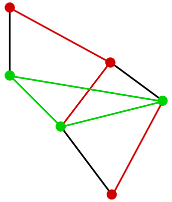


Solving Clique:

- Find the k added edges: $O(1.53^k + n^3)$ time [Gramm et al., Algorithmica 2004].
- Find the largest clique in the underlying cluster graph.
- Find the largest clique in the subgraph induced by the vertices that gained in degree: $O(1.22^{2k}) = O(1.49^k)$ time [Robson, J. Algorithms 1986].

Clique on Nearly Cluster Graphs

Distance from Triviality: k edges added.



Solving Clique:

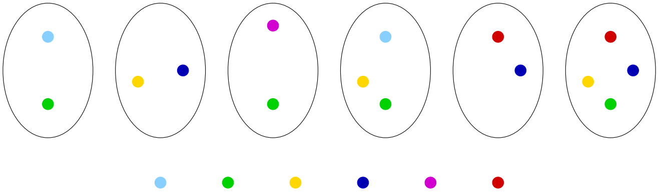
- Find the k added edges: $O(1.53^k + n^3)$ time [Gramm et al., Algorithmica 2004].
- Find the largest clique in the underlying cluster graph.
- Find the largest clique in the subgraph induced by the vertices that gained in degree: $O(1.22^{2k}) = O(1.49^k)$ time [Robson, J. Algorithms 1986].

Clique on Nearly Cluster Graphs

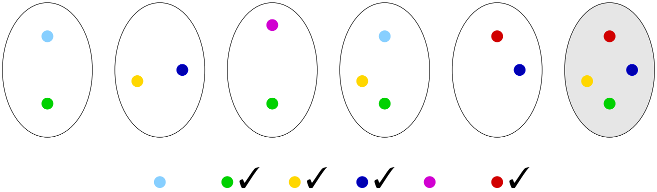
Theorem

Clique for a cluster graph with k edges added can be solved in $O(1.53^k + n^3)$ time.

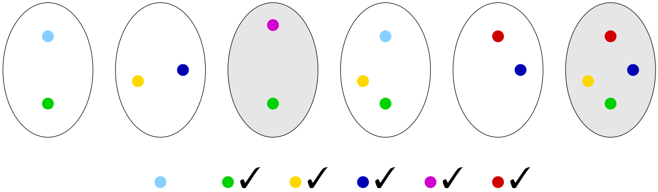
Set Cover



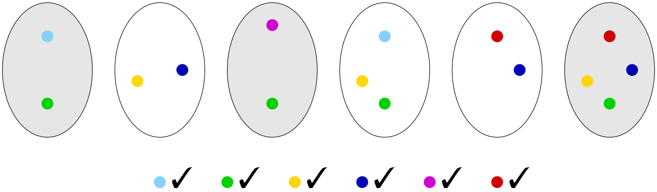
Set Cover



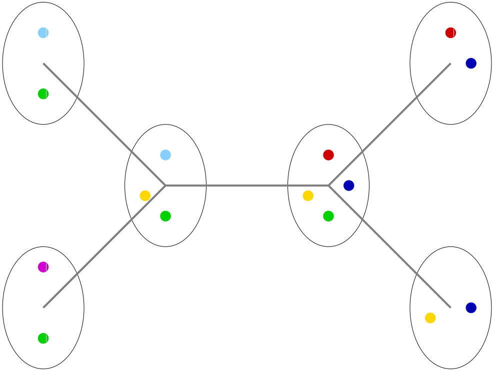
Set Cover



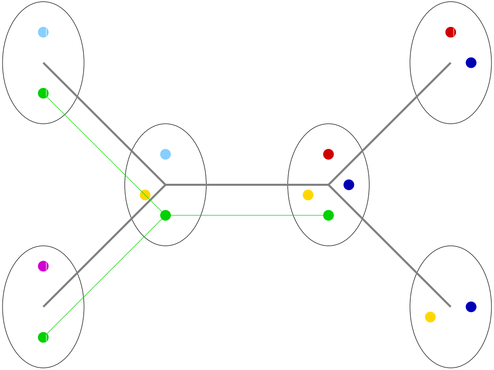
Set Cover



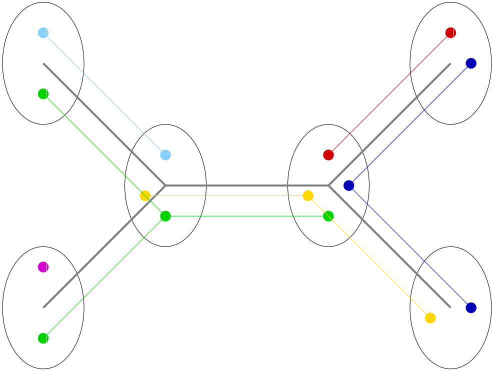
Tree-Like Set Cover



Tree-Like Set Cover



Tree-Like Set Cover



Parameterizing Tree-Like Weighted Set Cover

[Guo&Niedermeier, Manuscript, June 2004]

- Tree-like Weighted Set Cover is NP-complete, even with bounded number of occurrences per element.
- Tree-like Weighted Set Cover can be solved in polynomial time if the underlying tree is a path.

Parameterizing Tree-Like Weighted Set Cover

[Guo&Niedermeier, Manuscript, June 2004]

- Tree-like Weighted Set Cover is NP-complete, even with bounded number of occurrences per element.
- Tree-like Weighted Set Cover can be solved in polynomial time if the underlying tree is a path.

Triviality: Subset trees that are paths.

Parameterizing Tree-Like Weighted Set Cover

[Guo&Niedermeier, Manuscript, June 2004]

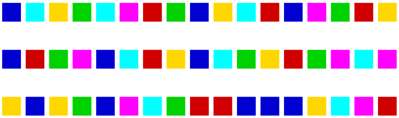
- Tree-like Weighted Set Cover is NP-complete, even with bounded number of occurrences per element.
- Tree-like Weighted Set Cover can be solved in polynomial time if the underlying tree is a path.

Triviality: Subset trees that are paths.
Distance from Triviality: Number of leaves of the subset tree.

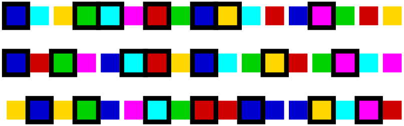
Theorem

Tree-like Weighted Set Cover *with occurrence bounded by d can be solved in $O(2^{dk^2} \cdot m^2 n)$ time, where k denotes the number of the leaves of the subset tree.*

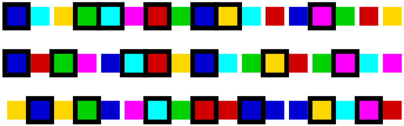
Longest Common Subsequence



Longest Common Subsequence

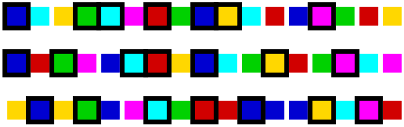


Longest Common Subsequence



- Longest Common Subsequence is NP-complete and W[1]-hard for parameter "number of strings"

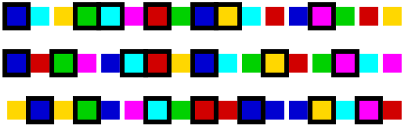
Longest Common Subsequence



- Longest Common Subsequence is NP-complete and W[1]-hard for parameter "number of strings"
- Longest Common Subsequence can be solved in polynomial time if all strings are permutations of $1 \dots n$.

Triviality: Strings are permutations.

Longest Common Subsequence



- Longest Common Subsequence is NP-complete and W[1]-hard for parameter "number of strings"
- Longest Common Subsequence can be solved in polynomial time if all strings are permutations of $1 \dots n$.

Triviality: Strings are permutations.
Distance from Triviality: Maximum occurrence number.

Theorem

Longest Common Subsequence *of k strings can be solved in* $O(2^{2k \log s} \cdot k \cdot n^2)$ *time, where s denotes the maximum occurrence number of a letter in an input string.*

Summary

Distance from triviality—a natural way of parameterizing a hard problem X:

1. Determine efficiently solvable special cases of X—the triviality.
2. Identify useful distance measures from the triviality—the (structural) parameter.

- Mostly structural results: How can we extend the range of tractability?
- Might also lead to efficient practical implementations if the parameter is small.