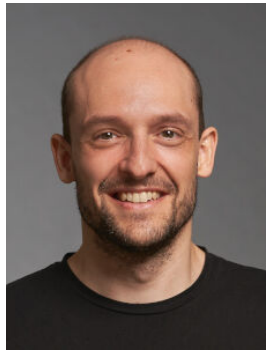
Parameterized Algorithms

Exercise 1 – Orga, Sheet 1, *H*-free Graphs

Elly, Jean-Pierre, Wendy

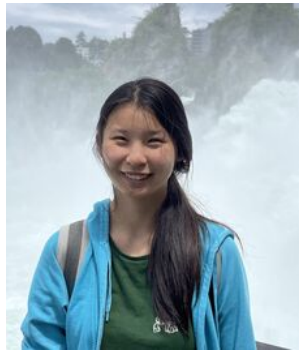
Orga Stuff



Thomas

Elly

Jean-Pierre

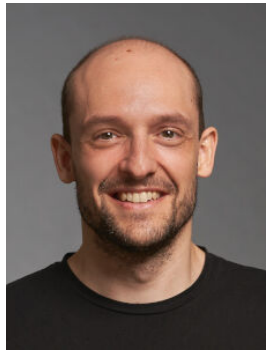


Wendy

Lecture

Exercise

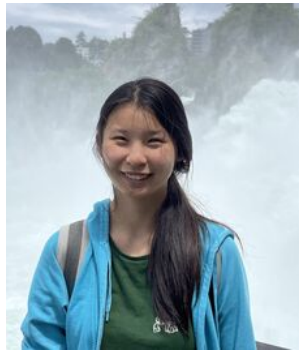
Orga Stuff



Thomas

Elly

Jean-Pierre



Wendy

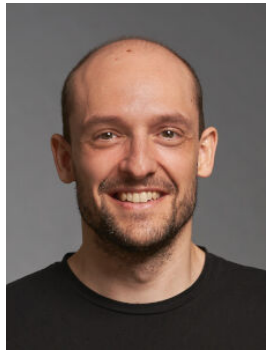
Lecture

Exercise

?

You

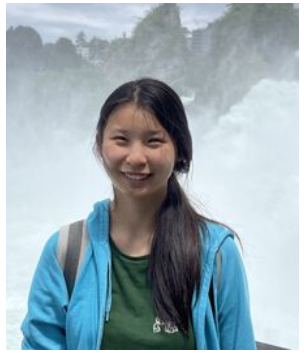
Orga Stuff



Thomas

Elly

Jean-Pierre



Wendy

You

Lecture

Exercise

Exercise Session

- discuss last exercise sheet
- get hints for next exercise sheet
- work on other problems, get different perspectives

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet
- practice writing solutions

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet
- practice writing solutions
  - choose proper abstraction level

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet
- practice writing solutions
  - choose proper abstraction level
  - structure that is easy to follow

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet
- practice writing solutions
  - choose proper abstraction level
  - structure that is easy to follow
  - give additional high-level explanation

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet
- practice writing solutions
  - choose proper abstraction level
  - structure that is easy to follow
  - give additional high-level explanation

Points System

0%: did not write anything

25%: had ideas, explained where you are stuck

50%: basic idea is correct (and explained)

75%: small/few details missing

100%: correct and well-written

More Orga Stuff

Exercise Sheets

- new sheet every two weeks (Thu – Wed)
- submission in teams via ILIAS (not handwritten, pls) unless your handwriting is really good
- coding submissions (bonus points)

Talk to each other, ask for hints on Discord, ask us in person!

send `!help join` to the scale bot

Goals

- at least $\frac{1}{2}$ of total points, $\frac{1}{4}$ of every sheet
- practice writing solutions
  - choose proper abstraction level
  - structure that is easy to follow
  - give additional high-level explanation

Points System

0%: did not write anything

25%: had ideas, explained where you are stuck

50%: basic idea is correct (and explained)

75%: small/few details missing

100%: correct and well-written

Sheet 1

Branching Vector

builds on second lecture

Sheet 1

Branching Vector

builds on second lecture

Dominating Set

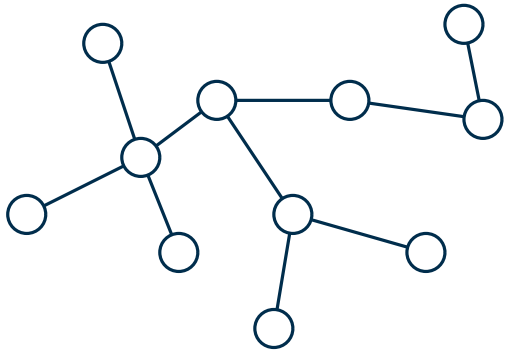
Sheet 1

Branching Vector

builds on second lecture

Dominating Set

a) on trees



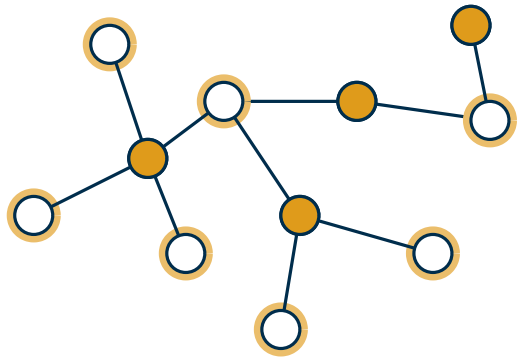
Sheet 1

Branching Vector

builds on second lecture

Dominating Set

a) on trees



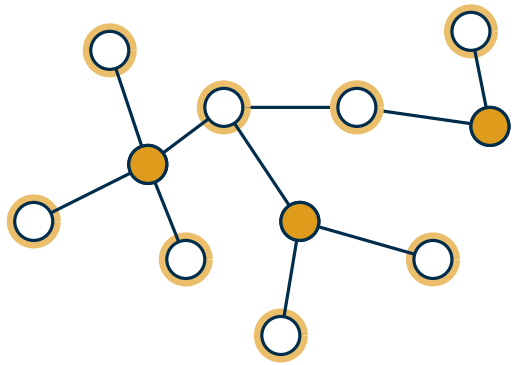
Sheet 1

Branching Vector

builds on second lecture

Dominating Set

a) on trees



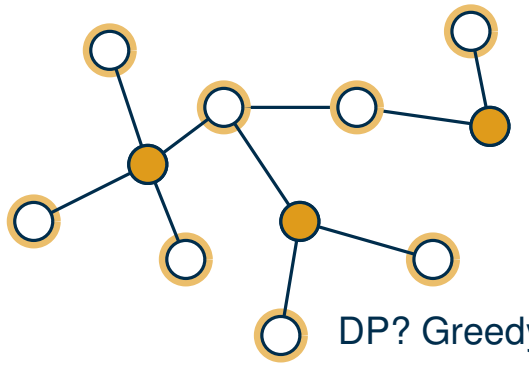
Sheet 1

Branching Vector

builds on second lecture

Dominating Set

a) on trees



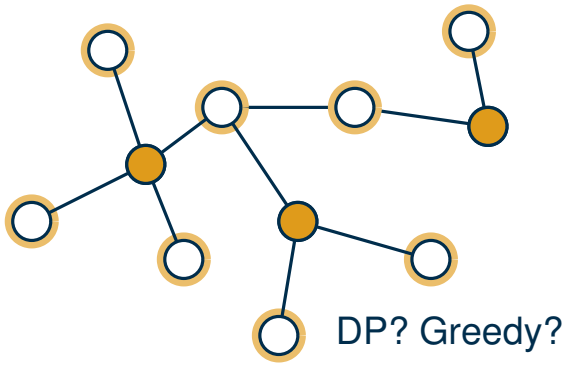
Sheet 1

Branching Vector

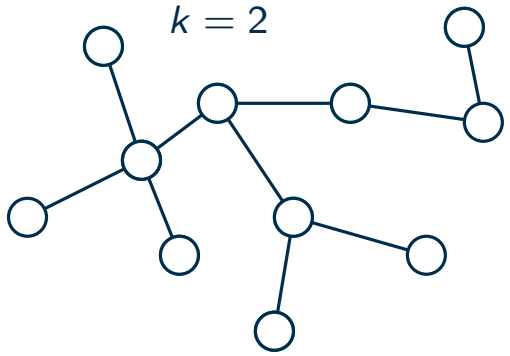
builds on second lecture

Dominating Set

a) on trees



b) k-dominating set on trees



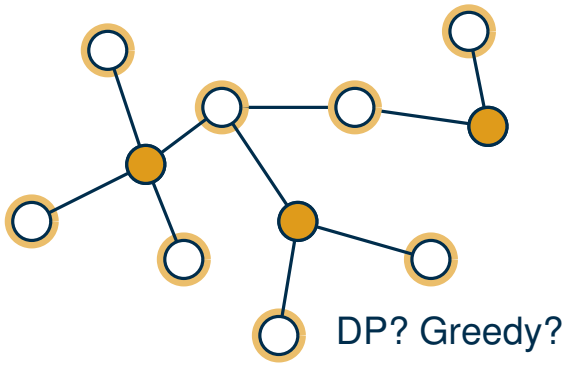
Sheet 1

Branching Vector

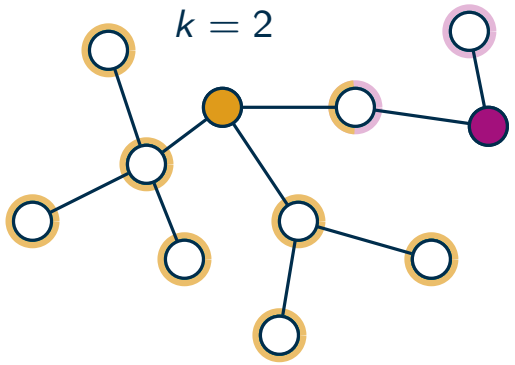
builds on second lecture

Dominating Set

a) on trees



b) k-dominating set on trees



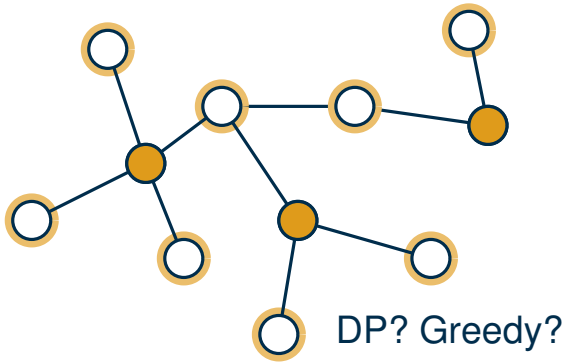
Sheet 1

Branching Vector

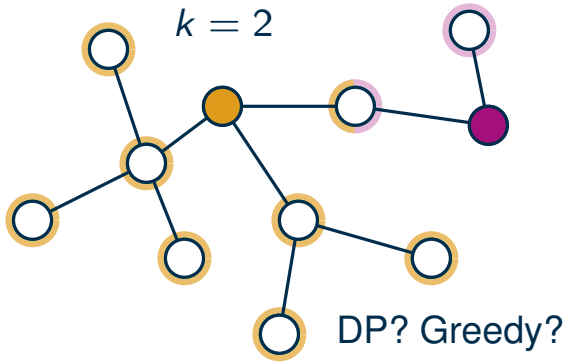
builds on second lecture

Dominating Set

a) on trees



b) k-dominating set on trees



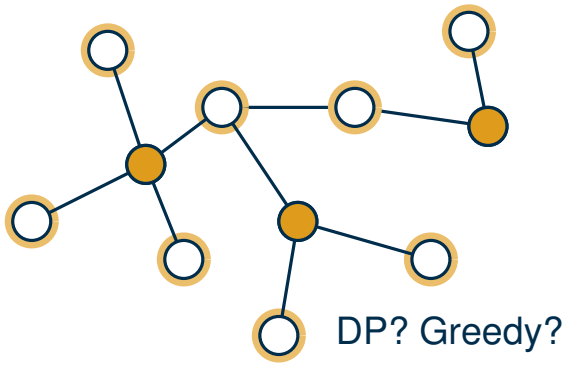
Sheet 1

Branching Vector

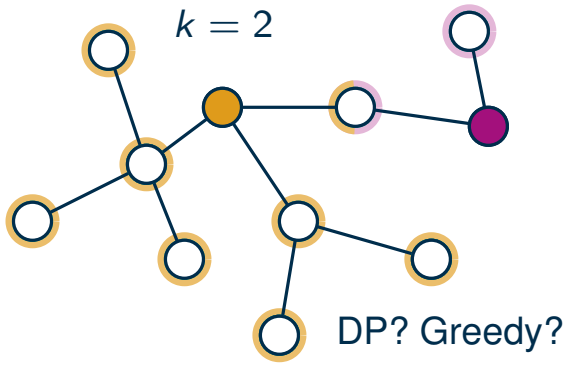
builds on second lecture

Dominating Set

a) on trees



b) k-dominating set on trees



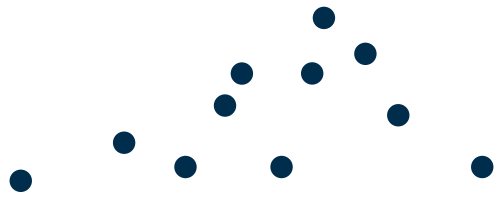
c) FPT in max deg + solution size on general graphs

Sheet 1

Branching Vector

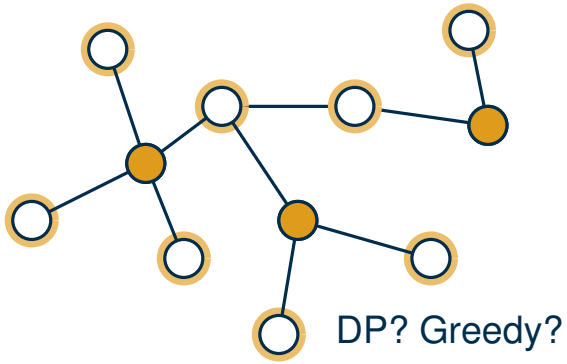
builds on second lecture

Line Cover

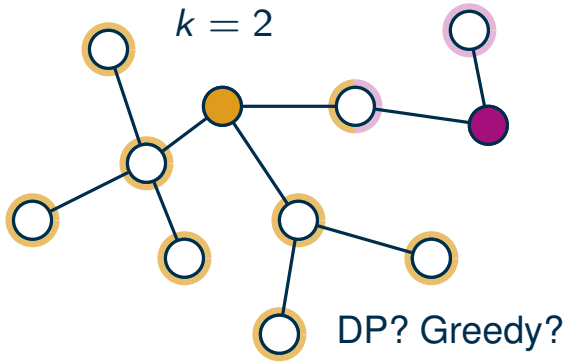


Dominating Set

a) on trees



b) k-dominating set on trees



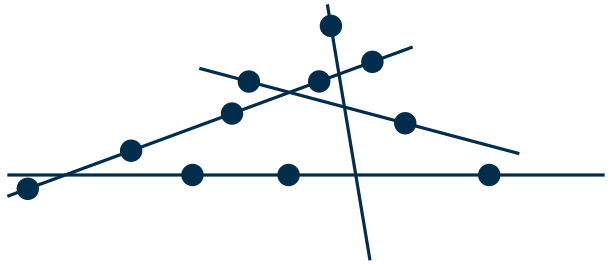
c) FPT in max deg + solution size on general graphs

Sheet 1

Branching Vector

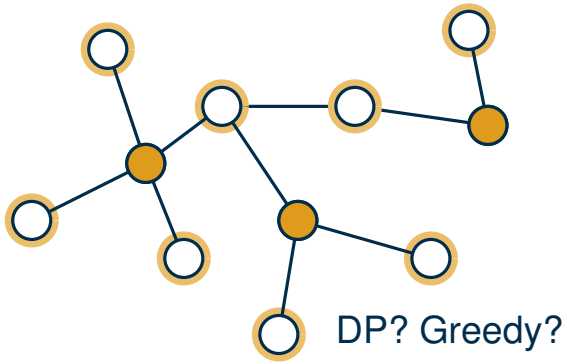
builds on second lecture

Line Cover

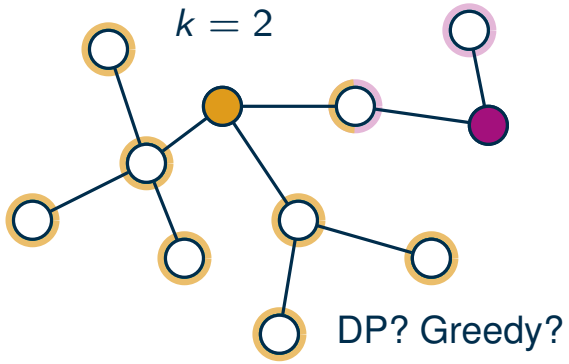


Dominating Set

a) on trees



b) k-dominating set on trees



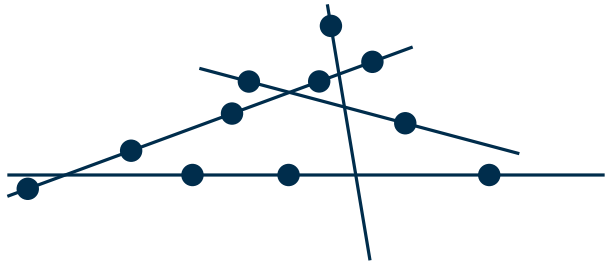
c) FPT in max deg + solution size on general graphs

Sheet 1

Branching Vector

builds on second lecture

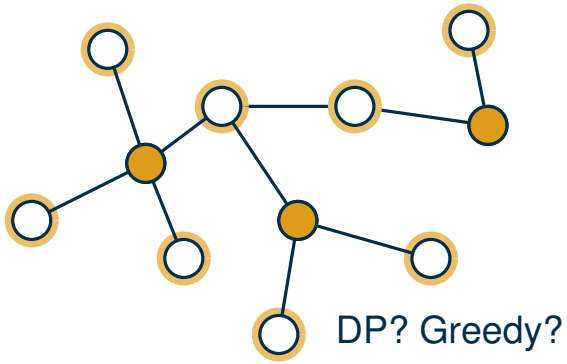
Line Cover



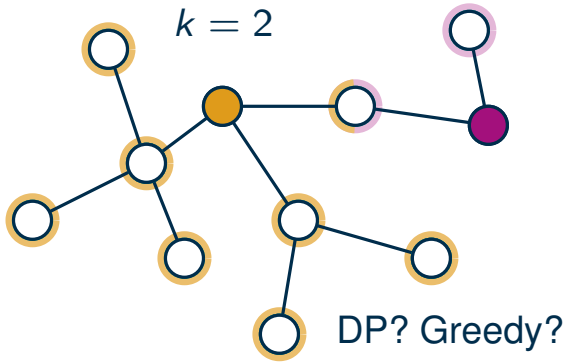
find reduction rules

Dominating Set

a) on trees



b) k-dominating set on trees



c) FPT in max deg + solution size on general graphs

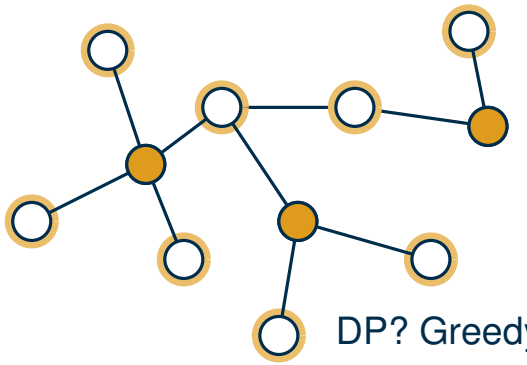
Sheet 1

Branching Vector

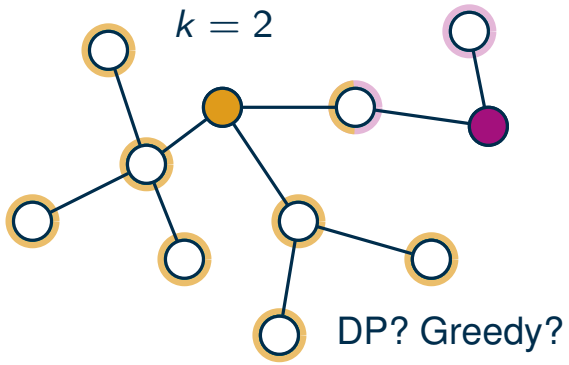
builds on second lecture

Dominating Set

a) on trees

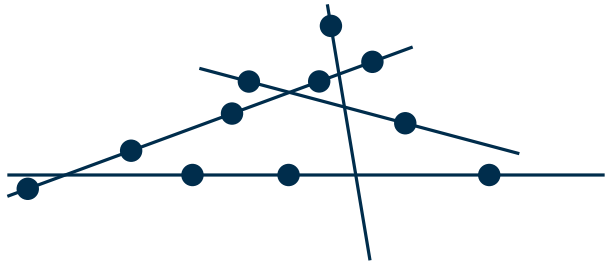


b) k-dominating set on trees

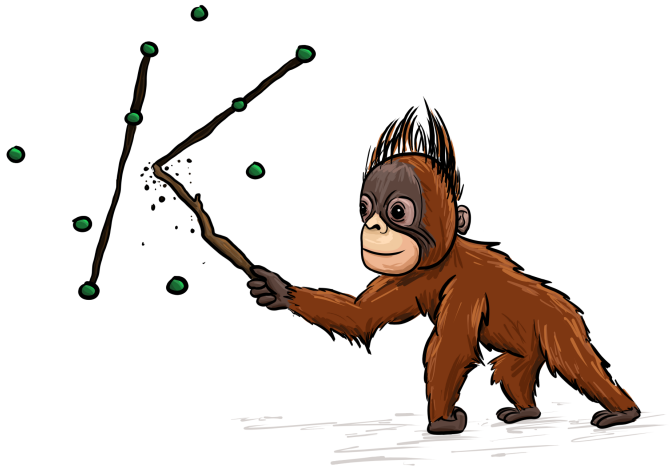


c) FPT in max deg + solution size on general graphs

Line Cover



find reduction rules

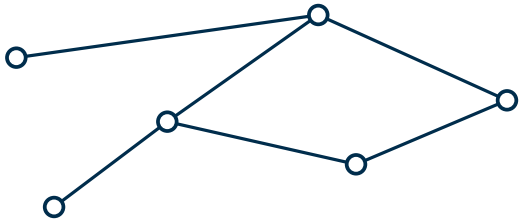


Tim Domnick, 2024/25

Techniques

Bounded Search Tree

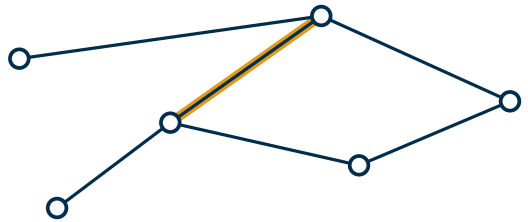
Vertex Cover



Techniques

Bounded Search Tree

Vertex Cover

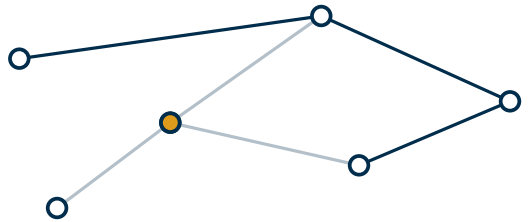


- cover an edge $\rightarrow$ two options to choose

Techniques

Bounded Search Tree

Vertex Cover

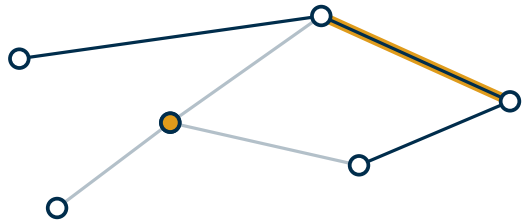


- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)

Techniques

Bounded Search Tree

Vertex Cover

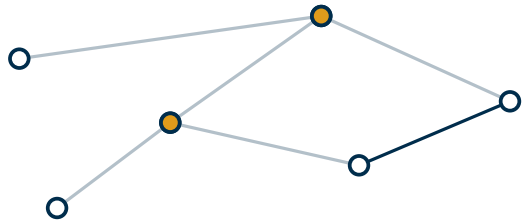


- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)

Techniques

Bounded Search Tree

Vertex Cover

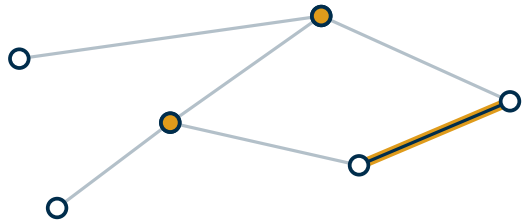


- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)

Techniques

Bounded Search Tree

Vertex Cover

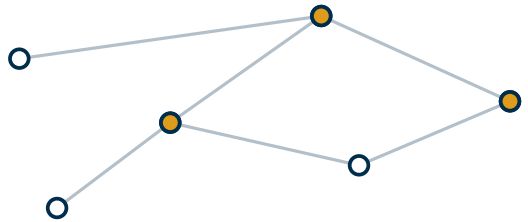


- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)

Techniques

Bounded Search Tree

Vertex Cover

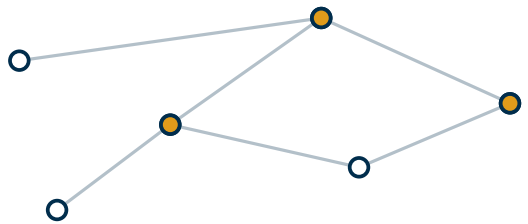


- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time

Techniques

Bounded Search Tree

Vertex Cover

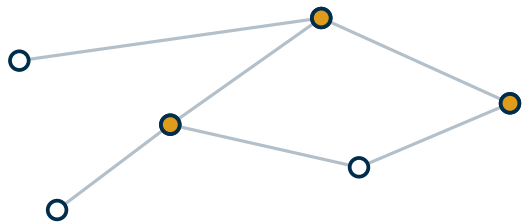


- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

Techniques

Bounded Search Tree

Vertex Cover



- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

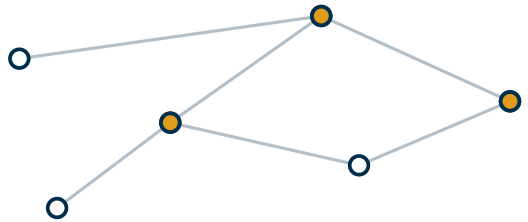
To prove

- in search tree:
  a child yes-instance $\Leftrightarrow$ parent yes-instance

Techniques

Bounded Search Tree

Vertex Cover



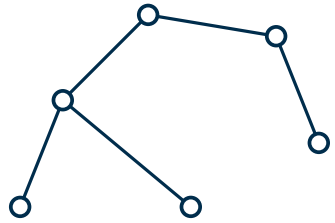
- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

To prove

- in search tree:
  a child yes-instance $\Leftrightarrow$ parent yes-instance

Kernelization

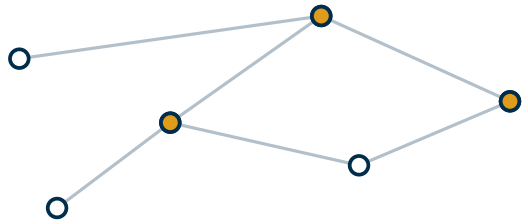
Vertex Cover on trees



Techniques

Bounded Search Tree

Vertex Cover



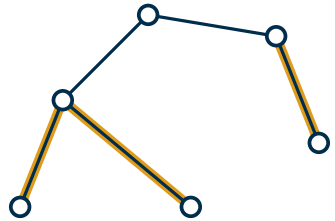
- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

To prove

- in search tree:
  a child yes-instance $\Leftrightarrow$ parent yes-instance

Kernelization

Vertex Cover on trees

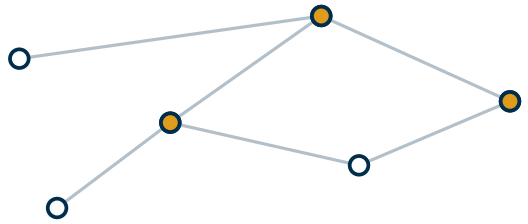


- edges to leaves need to be covered

Techniques

Bounded Search Tree

Vertex Cover



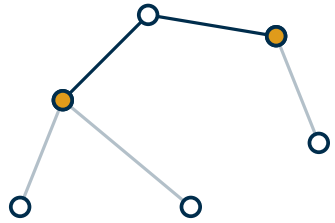
- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

To prove

- in search tree:
  a child yes-instance $\Leftrightarrow$ parent yes-instance

Kernelization

Vertex Cover on trees

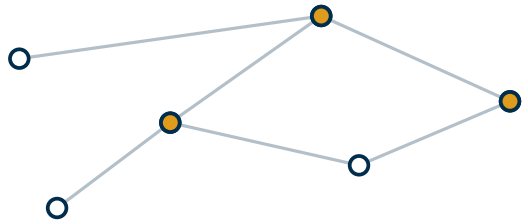


- edges to leaves need to be covered
- choosing parent is always better

Techniques

Bounded Search Tree

Vertex Cover



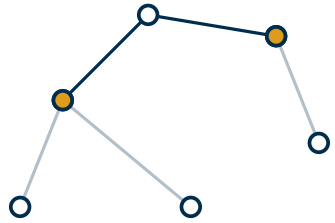
- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

To prove

- in search tree:
  a child yes-instance $\Leftrightarrow$ parent yes-instance

Kernelization

Vertex Cover on trees

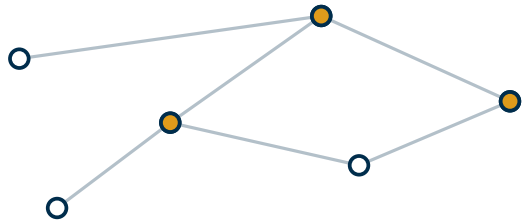


- edges to leaves need to be covered
- choosing parent is always better
- apply reduction rules until small kernel

Techniques

Bounded Search Tree

Vertex Cover



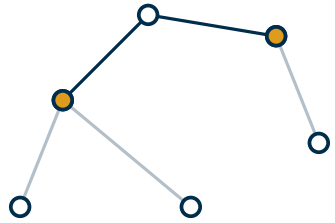
- cover an edge $\rightarrow$ two options to choose
- choice makes progress (k decreases)
- $\text{options}^{\text{depth}} \cdot \text{poly}(n) = 2^k \cdot \text{poly}(n)$ time
- vertex cover in $O(1.6181^k \cdot \text{poly}(n))$ possible

To prove

- in search tree:
  a child yes-instance $\Leftrightarrow$ parent yes-instance

Kernelization

Vertex Cover on trees



- edges to leaves need to be covered
- choosing parent is always better
- apply reduction rules until small kernel

To prove

- reduction rules correct
- kernel is small

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?


-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?


-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

H-FREE EDGE DELETION

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?


-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

H-FREE EDGE DELETION

(induced)
H-FREE EDGE ADDITION

(induced)
H-FREE EDGE EDITION

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?


-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

H-FREE EDGE DELETION

(induced)
H-FREE EDGE ADDITION

(induced)
H-FREE EDGE EDITION

finite set of graphs → $\mathcal{F}$-FREE VERTEX DELETION

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?

How to find H in G?


-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

H-FREE EDGE DELETION

(induced)
H-FREE EDGE ADDITION

(induced)
H-FREE EDGE EDITION

finite set of graphs → $\mathcal{F}$-FREE VERTEX DELETION

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?

How to find H in G?


-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


with common vertex?


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

H-FREE EDGE DELETION

(induced)
H-FREE EDGE ADDITION

(induced)
H-FREE EDGE EDITION

finite set of graphs → **$\mathcal{F}$-FREE VERTEX DELETION**

H-free Graphs

H-FREE VERTEX DELETION
Given: Graph G, parameter k
Does V' exist with $|V'| \leq k$ vertices such that H does not appear as a subgraph in $G - V'$?

How to find H in G?

-FREE VD

- search tree
- improve (??)


-FREE VD

- search tree
- ... improve


with common vertex?


-FREE VD

- what about $K_{1,d}$?
- search tree
- kernelization

R1: vertex with super high degree?
R2: delete edge between adjacent vertices with small degrees

How many bad vertices are still ok?
How many vertices with small degree?

H-FREE EDGE DELETION

(induced)
H-FREE EDGE ADDITION

(induced)
H-FREE EDGE EDITION

finite set of graphs → $\mathcal{F}$-FREE VERTEX DELETION