

The background of the slide is a complex network graph. It features a dense web of white nodes connected by thin, dark lines. The nodes are distributed across the frame, with a higher concentration in the center and left side. The background color transitions from a dark teal on the left to a dark blue on the right.

# Parameterized Algorithms

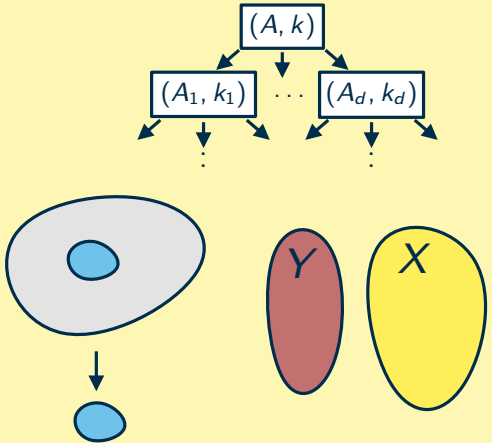
## Bounded Search Trees: Improved Branching

Thomas Bläsius

# Content

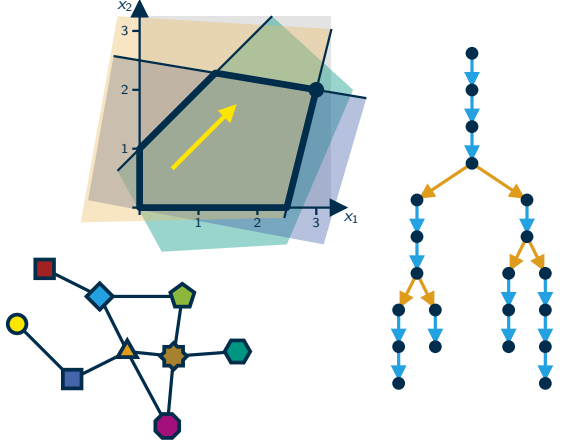
## Basic toolbox

- bounded search trees
- kernelization
- iterative compression



## Extended toolbox

- linear programs
- branch-and-reduce
- color coding



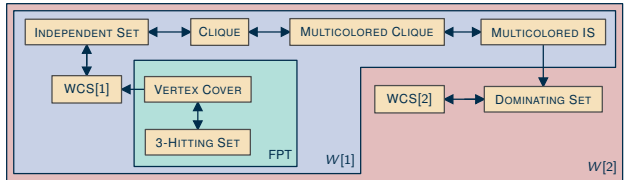
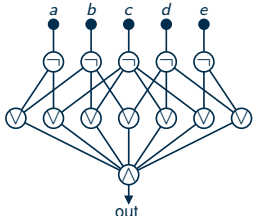
## Tree width

- dynamic programming
- chordal and planar graphs
- Courcelle's theorem



## Lower bounds

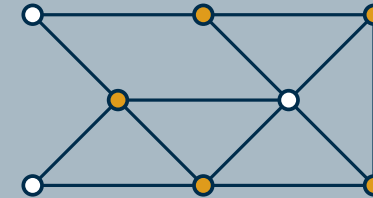
- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH



# Recap: Bounded search tree

## Problem: VERTEX COVER

Given a graph  $G = (V, E)$  and a parameter  $k$ . Does  $G$  have a vertex cover of size  $k$ ?

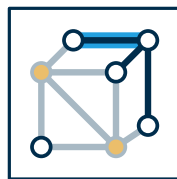
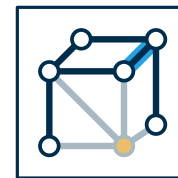


(vertex set  $V' \subseteq V$  with  $e \cap V' \neq \emptyset$  for all  $e \in E$ )

Is there a vertex cover of size at most  $k = 3$ ?



edge  $\{u, v\}$  is uncovered  
 $\Rightarrow u$  or  $v$  must be selected  
 $\rightarrow$  binary decision



solution of size 3

## Simple branching algorithm

- start:  $V' = \emptyset \rightarrow$  no edge covered
- step: branch on uncovered edge
- stop if: all edges covered or already  $k$  vertices selected

running time:  $O(2^k m)$

# Search tree with stronger bounds

## Can we be faster?

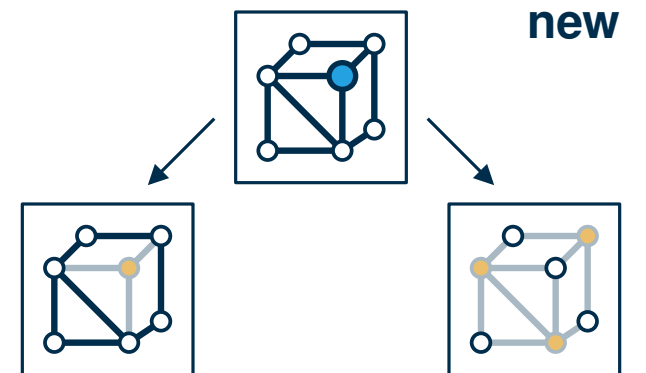
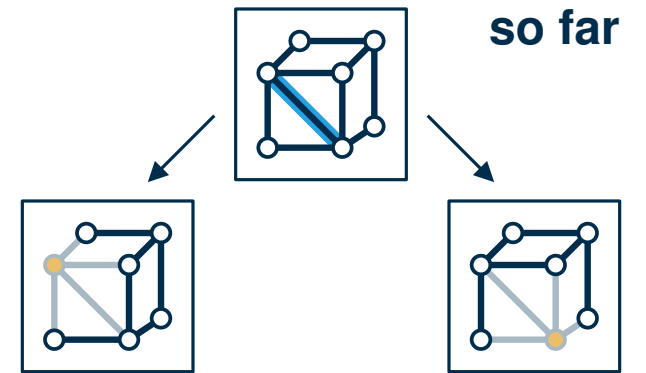
- goal: reduce the size of the search tree (so far:  $2^k$  leaves)
- branching rule so far: for edge  $\{u, v\}$  choose either  $u$  or  $v$
- new branching rule: for vertex  $v$  choose either  $v$  or  $N(v)$

## How many leaves does the resulting tree have?

- always choose a vertex  $v$  with degree  $\geq 2$
- upper bound for number of leaves:

$$T(k) = \begin{cases} T(k-1) + T(k-2), & \text{if } k \geq 2 \\ 1, & \text{otherwise} \end{cases}$$

- we get:  $T(k) \leq 1.6181^k$
- Why do we get 1.6181? Can we do better?  $\longrightarrow$  today



# Branching vectors

## Branching in general

- for instance  $(G, k)$  create  $\ell$  child instances  $(G_1, k_1), \dots, (G_\ell, k_\ell)$
- let  $d_i = k - k_i$  (parameter shrinks in child  $G_i$  by  $d_i$ )
- branching vector:  $(d_1, \dots, d_\ell)$

**Seen so far:** vector  $(1, 1) \Rightarrow$  tree size  $2^k$ , vector  $(1, 2) \Rightarrow$  tree size  $1.6181^k$

## How can we prove the tree size?

- guess the base  $\lambda$
- show  $T(k) \leq c \cdot \lambda^k$  via induction (more details in a moment)

## How can we guess the base?

- find root of a polynomial (more details in two moments)
- WolframAlpha helps with finding the root

# Tree size

## Example: branching vector (1, 2)

- recurrence:

$$T(k) = \begin{cases} T(k-1) + T(k-2), & \text{for } k \geq 2 \\ 1, & \text{otherwise} \end{cases}$$

- claim:  $T(k) \leq 1.6181^k$

## Proof

- initial step:  $k < 2 \Rightarrow T(k) = 1 \leq 1.6181^k$

- induction step:  $T(k) = T(k-1) + T(k-2)$

$$\leq 1.6181^{k-1} + 1.6181^{k-2}$$

$$= 1.6181^{k-2} \cdot \underbrace{(1.6181 + 1)}_{\leq 1.6181^2} \leq 1.6181^k$$

$$\leq 1.6181^2 \approx 2.6182$$

# Guessing the basis

## Example: branching vector (1, 2)

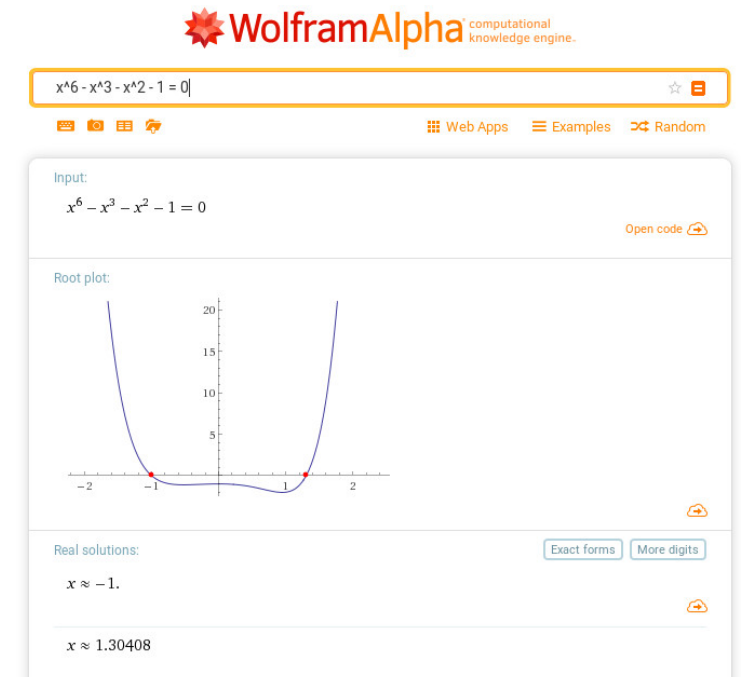
- recurrence:

$$T(k) = \begin{cases} T(k-1) + T(k-2), & \text{for } k \geq 2 \\ 1, & \text{otherwise} \end{cases}$$

- find minimum  $\lambda > 0$  such that:  $T(k) = T(k-1) + T(k-2) \leq c \cdot \lambda^{k-1} + c \cdot \lambda^{k-2} \leq c \cdot \lambda^k$
- choose  $\lambda$ , such that:  $c \cdot \lambda^{k-1} + c \cdot \lambda^{k-2} = c \cdot \lambda^k$
- solve:  $\lambda^2 - \lambda - 1 = 0 \rightarrow \lambda = \frac{1+\sqrt{5}}{2} \leq 1.6181$

## General setting

- branching vector  $(d_1, \dots, d_\ell)$  with  $d = \text{maximum of the } d_i$
- solve:  $0 = \lambda^d - \lambda^{d-d_1} - \dots - \lambda^{d-d_\ell}$
- example: branching vector (3, 4, 6)
- solve:  $0 = \lambda^6 - \lambda^3 - \lambda^2 - 1$



# Back to VERTEX COVER

branching vector	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
base $b$	2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

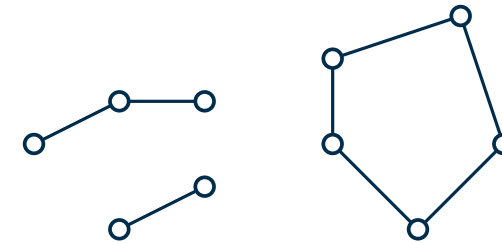
## Seen so far

- for an edge  $uv$ , choose  $u$  or  $v \rightarrow$  branching vector  $(1, 1)$
- for a vertex  $v$  with  $\deg(v) \geq 2$ , choose  $v$  or  $N(v) \rightarrow$  vector  $(1, 2)$

Why is there always a vertex  $v$  with  $\deg(v) \geq 2$ ?

## Can we improve?

- is there always a vertex  $v \in V$  with  $\deg(v) \geq 3$ ?
- if not:  $G$  consists of paths and cycles
- $\deg(v) \leq 2$  for all  $v \in V \Rightarrow$  solvable in polynomial time
- better branching: for a vertex  $v$  with  $\deg(v) \geq 3$ , choose  $v$  or  $N(v) \rightarrow$  vector  $(1, 3)$

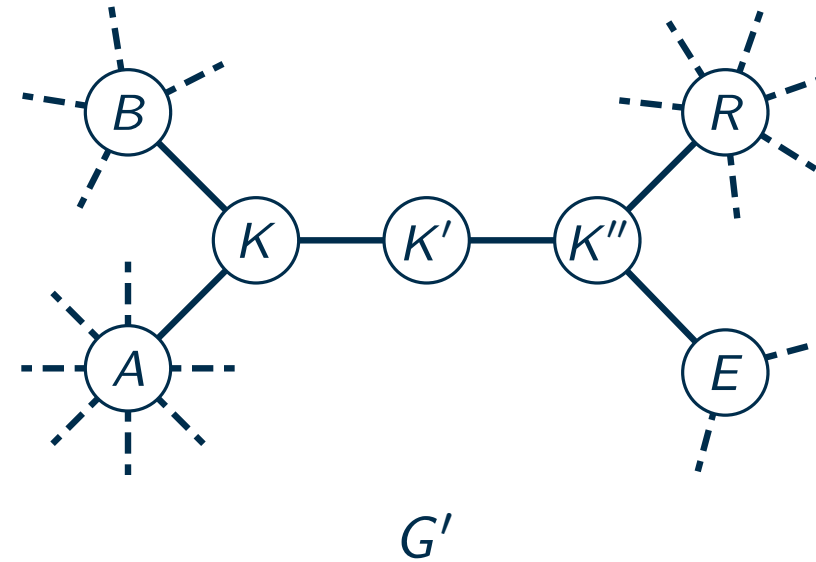
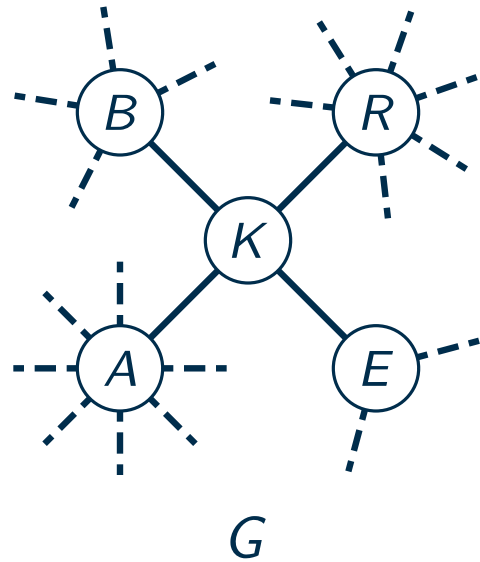


## Can we improve further?

- VERTEX COVER is NP-hard for graphs with maximum degree 3
- branching vector  $(1, 4)$  cannot be achieved as easily

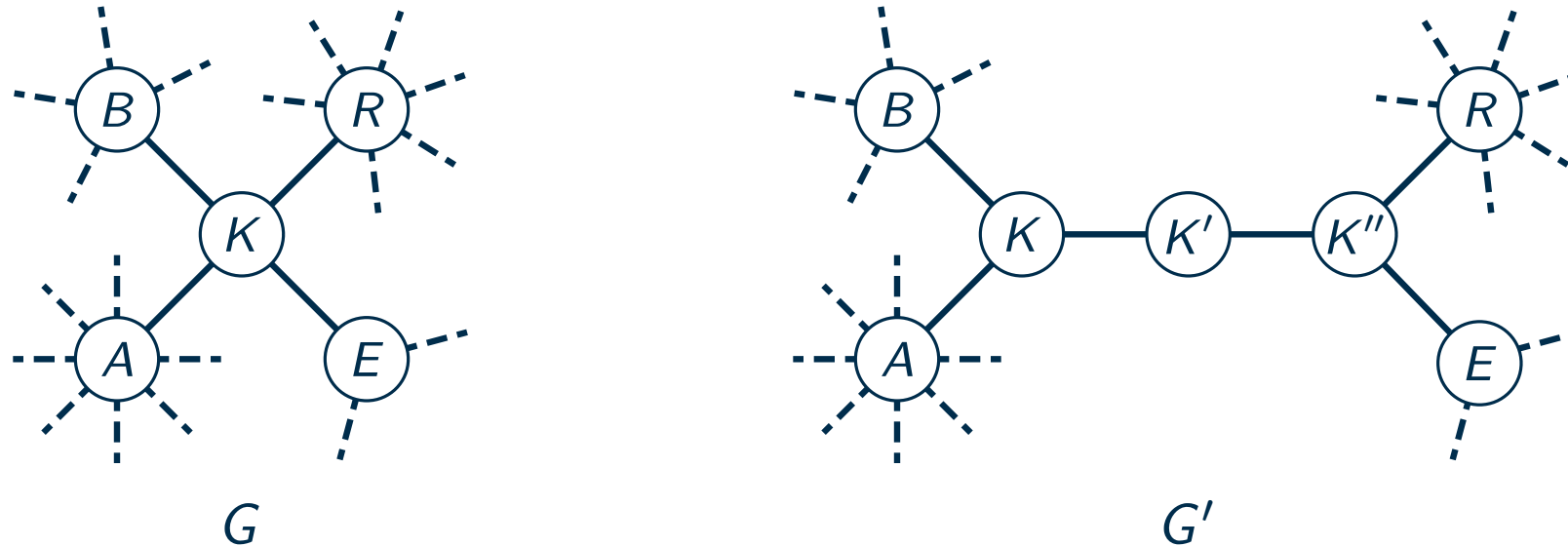
# What is happening here?

What does the following construction achieve?



# What is happening here?

What does the following construction achieve?



## Answer

- $G$  has a VC of size  $k \Leftrightarrow G'$  has a VC of size  $k + 1$
- degree of  $K$  is reduced from 4 to 3
- generalizes to higher degrees  $\rightarrow$  shows that VERTEX COVER is NP-hard for max-deg-3 graphs

# Eliminate $>$ Ignore

## Vertices with low degree

- so far: ignore vertex  $v$  if  $\deg(v) = 1$  or  $\deg(v) = 2$
- now: get completely rid of such vertices

## Why is that preferable?

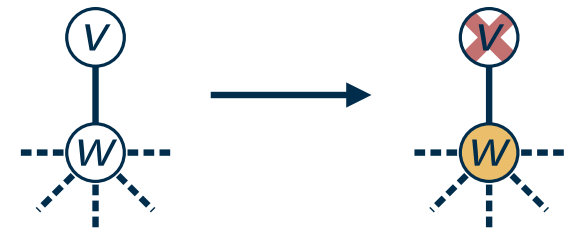
- shrinking the instance seems like a good idea
- restricts the class of possible instances  $\rightarrow \deg(v) \geq 3$  for all  $v \in V$  helps for later cases

**Rule 1:** there is a vertex  $v \in V$  with  $\deg(v) = 1$

- it never makes sense to choose  $v$
- instead, choose neighbor  $w$  of  $v$

## Does this count as branching?

- this is actually a reduction rule (or a branching rule with vector (1))
- no negative impact on the base in the running time



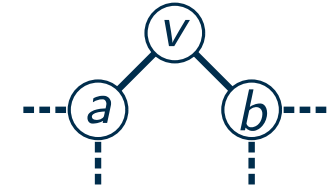
# Degree-2 vertices

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

**Rule 2:** there is a vertex  $v$  with  $\deg(v) = 2$

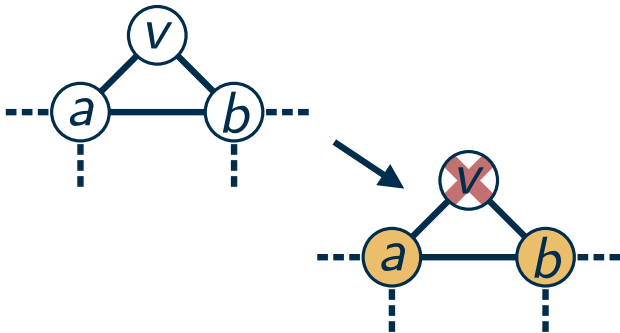
- if you choose  $a$ , you should also choose  $b$  (and vice versa)
- there are two options:
  - choose  $a$  and  $b$
  - exclude  $a$  and  $b \Rightarrow$  have to choose  $N(a) \cup N(b)$  (including  $v$ )

Why?



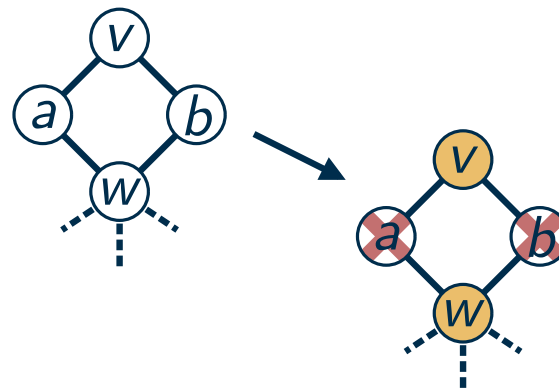
**Case 1:**  $ab \in E$

- excluding  $a$  and  $b$  is not ok
- have to choose  $a$  and  $b$



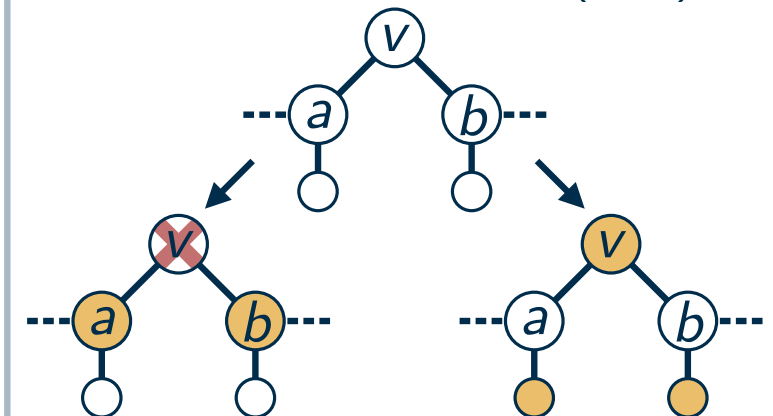
**Case 2:**  $N(a) \cup N(b) = \{v, w\}$

- never good to choose  $a$  and  $b$



**Case 3:**  $|N(a) \cup N(b)| \geq 3$

- branch with vector (2, 3)

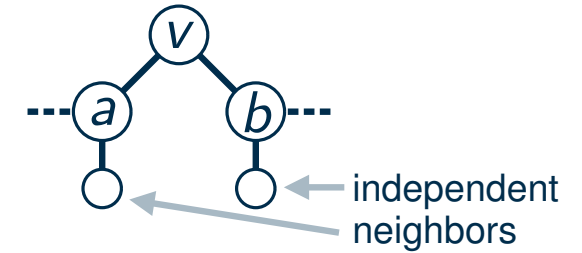
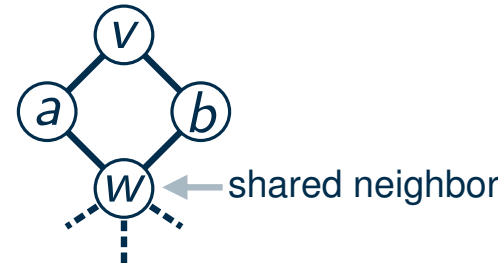
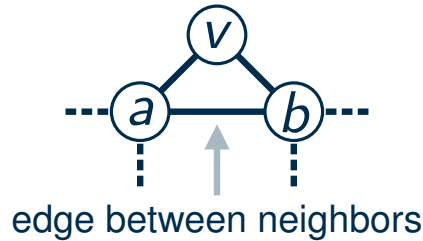


# Useful case-distinction

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

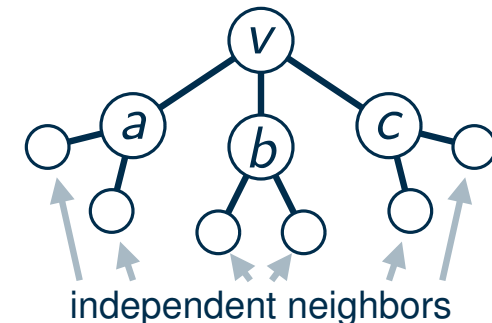
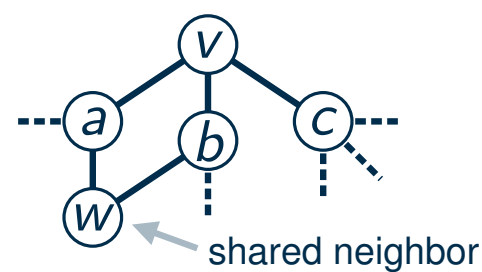
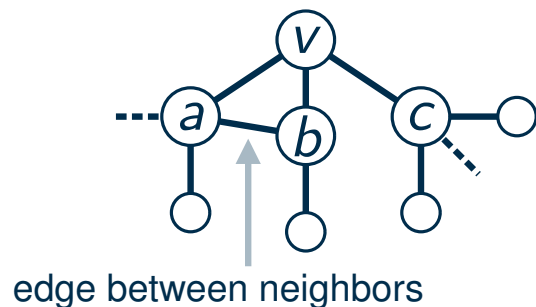
## Case distinction for degree-2 vertices

- cases in a sense capture, how independent the neighborhood of  $v$  is



- independent neighborhood  $\Rightarrow$  large union  $N(a) \cup N(b) \Rightarrow$  good branching vector
- dependent neighborhood  $\Rightarrow$  more structure  $\Rightarrow$  can be argued away

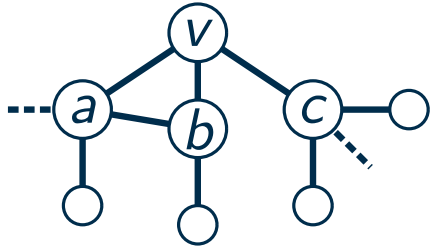
**Goal for degree-3 vertices:** similar case distinction but use that  $\deg(u) \geq 3$  for all  $u \in V$



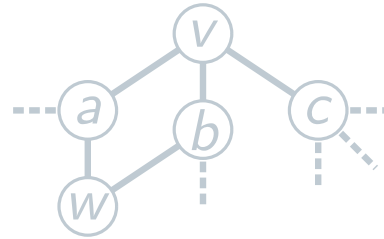
# Degree-3 vertices

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

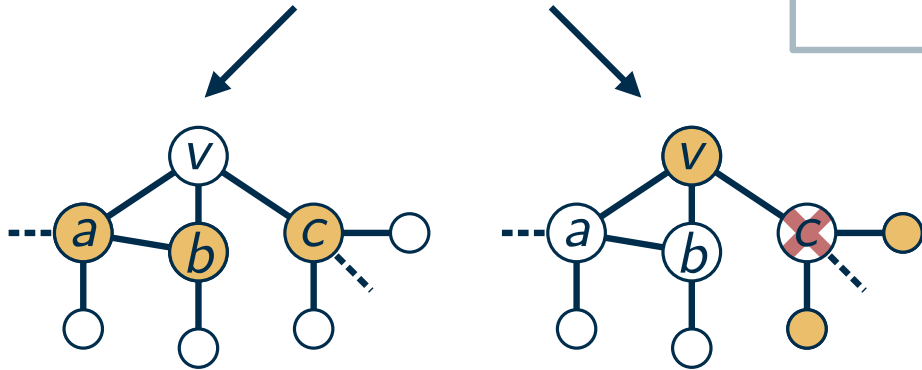
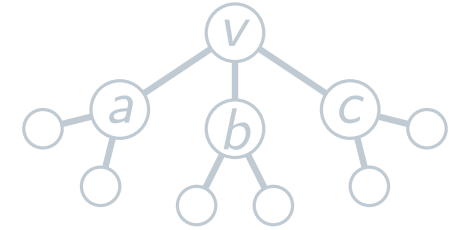
**Case 1:**  $ab \in E$



**Case 2:**  $w \in N(a) \cup N(b)$



**Case 3:** no edge among  $a, b, c$  and  $v$  is only common neighbor



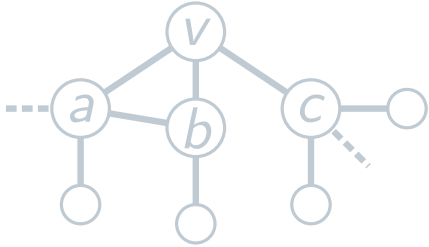
## Branching for case 1

- select or exclude  $c$
- selecting  $c$  yields degree-2 situation  $\Rightarrow$  select  $a$  and  $b$
- exclude  $c \Rightarrow$  select  $N(c)$  (note:  $\deg(c) \geq 3$ )
- branching vector:  $(3, 3)$

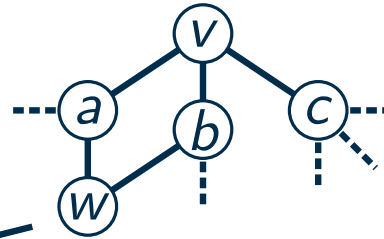
# Degree-3 vertices

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

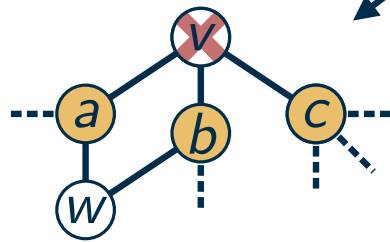
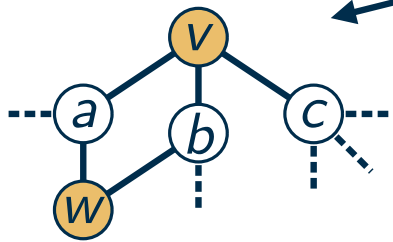
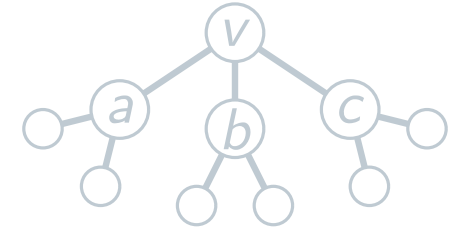
**Case 1:**  $ab \in E$



**Case 2:**  $w \in N(a) \cup N(b)$



**Case 3:** no edge among  $a, b, c$  and  $v$  is only common neighbor



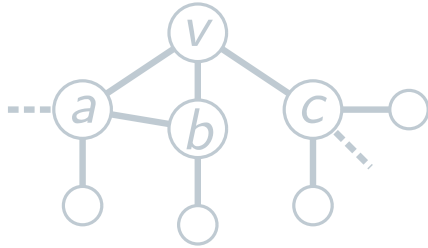
## Branching for case 2

- 4-cycle  $\rightarrow$  must select at least two opposite corners
- branching: select  $\{v, w\}$  or  $\{a, b\}$
- selecting  $\{a, b\}$  yields degree-1 situation  $\Rightarrow$  select  $c$  and exclude  $v$
- branching vector:  $(2, 3)$

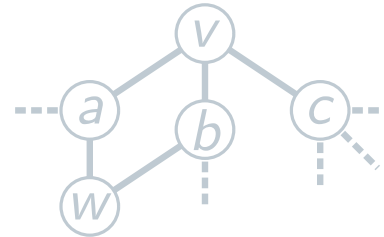
# Degree-3 vertices

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

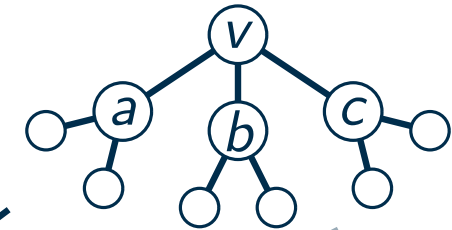
Case 1:  $ab \in E$



Case 2:  $w \in N(a) \cup N(b)$

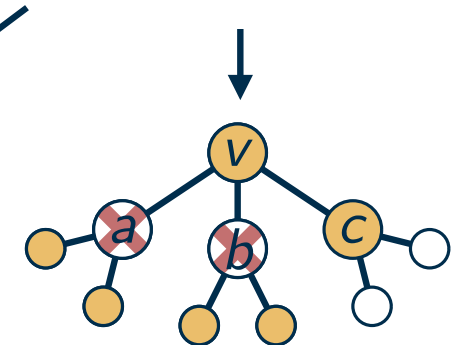
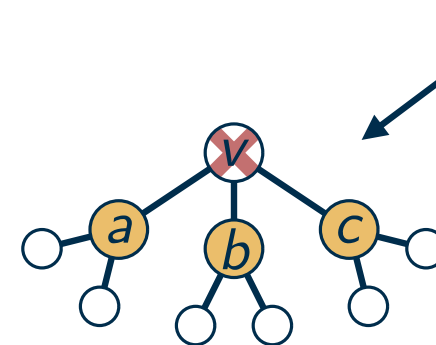
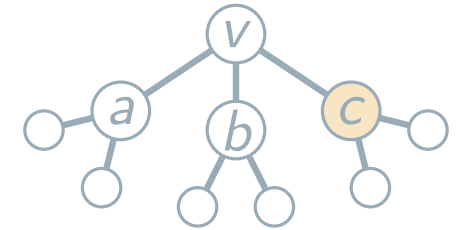
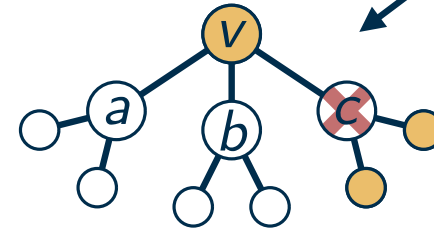


Case 3: no edge among  $a, b, c$  and  $v$  is only common neighbor



## Branching for case 3

- select or exclude  $c$
- exclude  $c \Rightarrow$  select  $N(c)$  (note:  $\deg(c) \geq 3$ )
- selecting  $c$  yields degree-2 situation:
  - select  $a$  and  $b$
  - exclude  $a$  and  $b$
- branching vector:  $(3, 3, 6)$



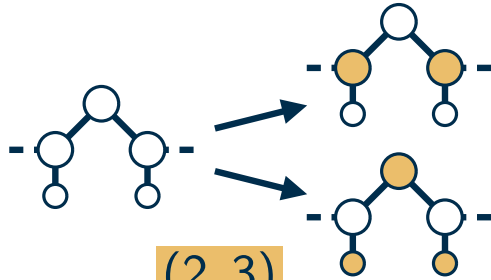
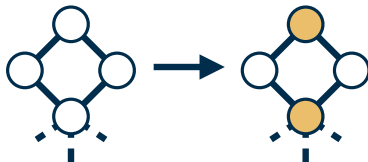
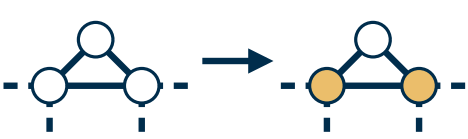
# Summary

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

**Rule 1:**  $\deg(v) = 1$

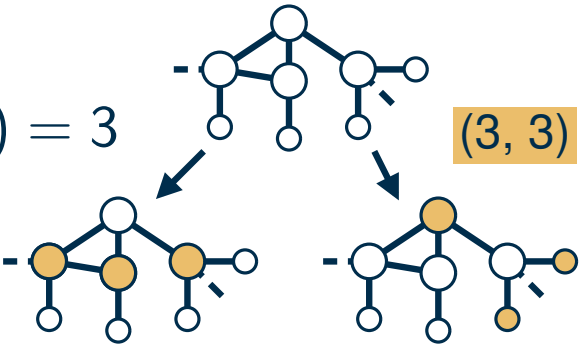


**Rule 2:**  $\deg(v) = 2$

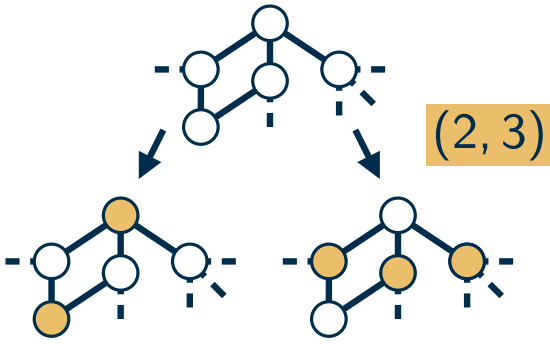


(2, 3)

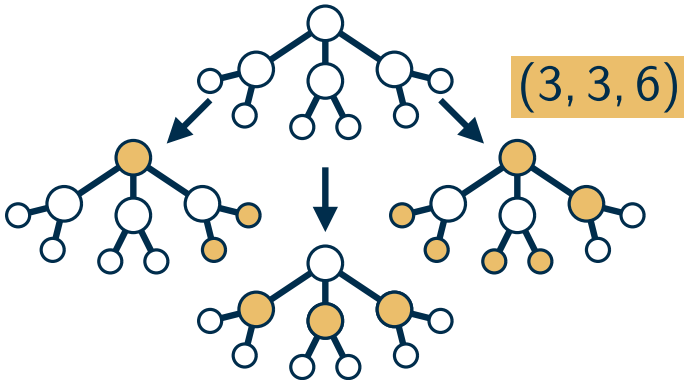
**Rule 3:**  $\deg(v) = 3$



(3, 3)

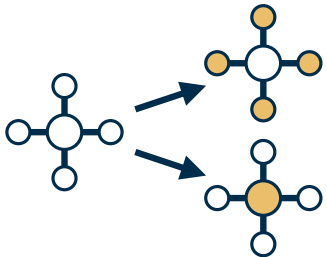


(2, 3)



(3, 3, 6)

**Rule 4:**  $\deg(v) = 4$



(1, 4)

worst branching vector: (1, 4)

$\Rightarrow$  running time:  $1.381^k \cdot n^{O(1)}$

# Can we do better?

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(3, 3)	(3, 3, 6)	(3, 4, 6)
2	1.618	1.466	1.381	1.325	1.325	1.26	1.342	1.305

**Rule 1:** there is  $v \in V$  with  $\deg(v) = 1$

**Rule 2:** there is  $v \in V$  with  $\deg(v) = 2$

(2, 3)

**Rule 3:** there is  $v \in V$  with  $\deg(v) = 3$

(3, 3, 6)

(3, 3)

(2, 3)

**Rule 4:** there is  $v \in V$  with  $\deg(v) \geq 5$

(1,  $\deg(v)$ )

**Rule 5:** all vertices have degree 4

(1, 4)

**Claim:** the total running time is  $1.342^k \cdot n^{O(1)}$

## Intuitive argument

- rule 5 happens at most once on every path from the root to a leaf
- costs only a constant factor
- we can ignore the branching vector (1, 4)

# Wrap-Up & Literature

## Theorem

VERTEX COVER parameterized by solution size  $k$  can be solved in  $1.342^k \cdot n^{O(1)}$  time.

## Bounded search tree

- better branching  $\rightarrow$  smaller base
- branching vectors help with the analysis
- “worst” branching vector  $\rightarrow$  may yield suboptimal bound (e.g., branch often with (5, 5); rarely with (1, 1))

## Improving the base

- best since 2010:  $O(1.2738^k + kn)$

[doi.org/10.1016/j.tcs.2010.06.026](https://doi.org/10.1016/j.tcs.2010.06.026)

- result from 2024:  $O^*(1.25284^k)$

<https://doi.org/10.4230/LIPIcs.STACS.2024.40>

- also see:

<http://fpt.wikidot.com/fpt-races>

Problem	$f(k)$	vertices in kernel	Reference/Comments
Vertex Cover	$1.2529^k$	$2k$	42
Connected Vertex Cover	$2^k$	no $k^{O(1)}$	26, randomized algorithm
Multiway Cut	$2^k$	not known	21, 38: $O(k^{s+1})$ -vertex kernel with $s$ terminals
Directed Multiway Cut	$2^{O(k^3)}$	no $k^{O(1)}$	34
Almost-2-SAT (VC-PM)	$2.3146^k$	$O(k^6)$	37, 38, randomized kernel
Multicut	$2^{O(k^3)}$	no $k^{O(1)}$	22, 35