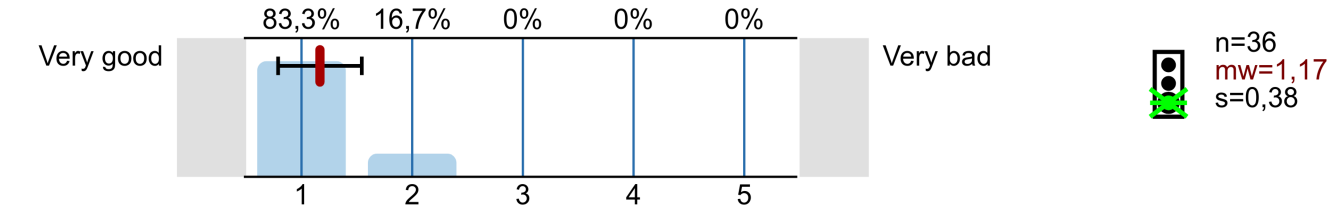
Parameterized Algorithms

Exercise 7 – Sheet 6, *W*-Hierarchy

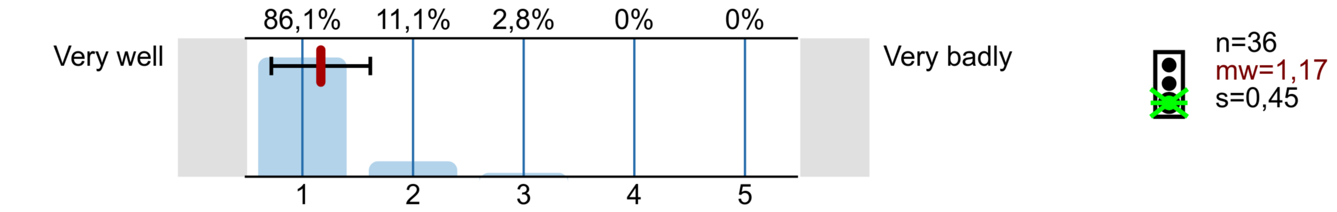
Elly, Jean-Pierre, Wendy

Evaluation Results

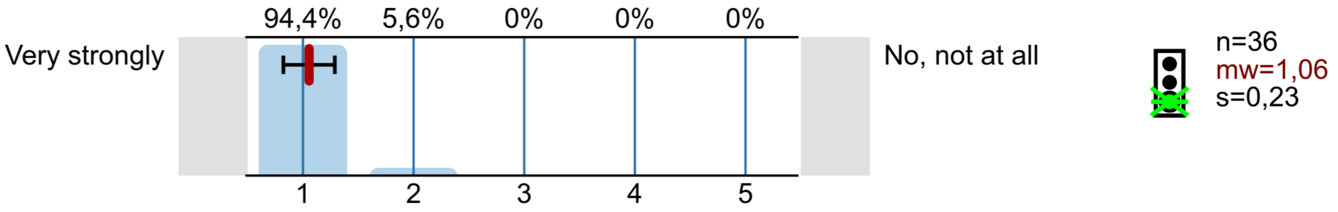
Please rate the course as a whole



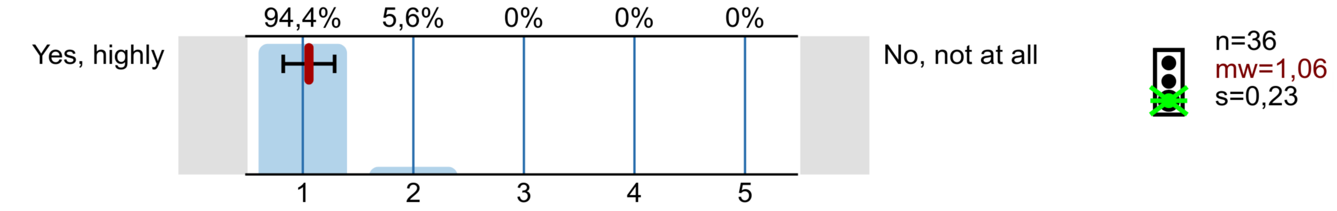
How is the course structured?



Does the lecturer appear dedicated and motivated during the course?

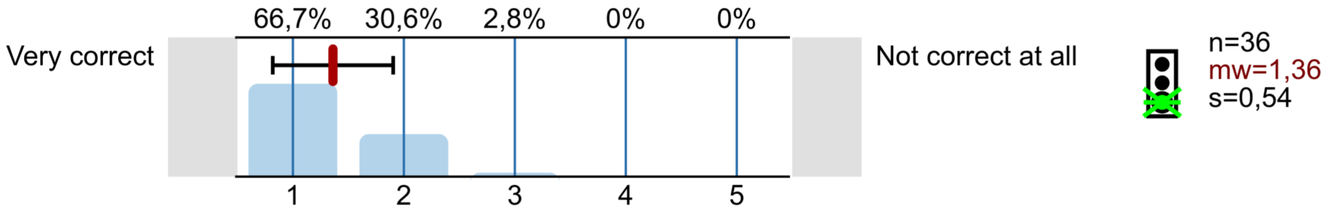


Is the lecturer responsive to questions and concerns of the students?

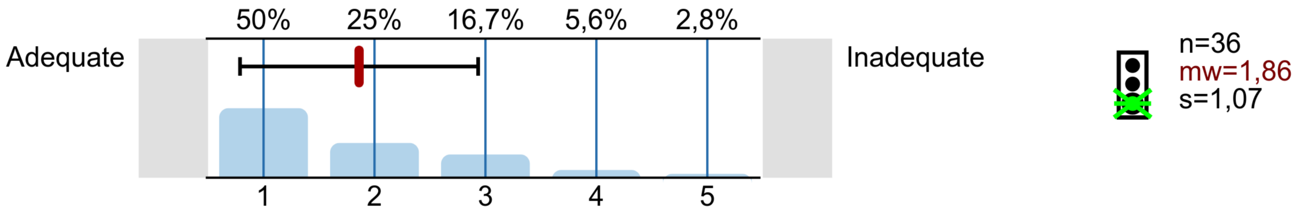


Evaluation Results

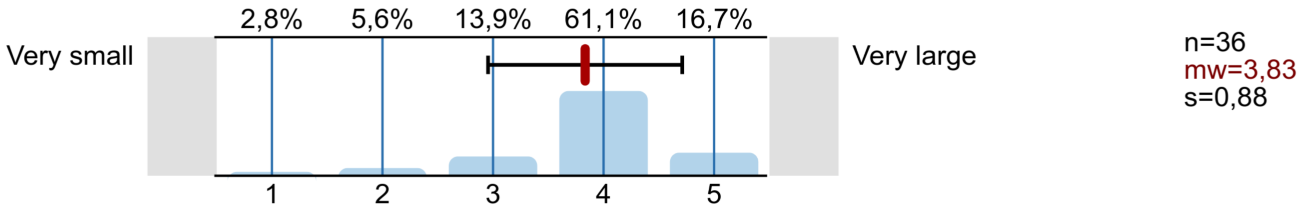
I learn a lot in this course.



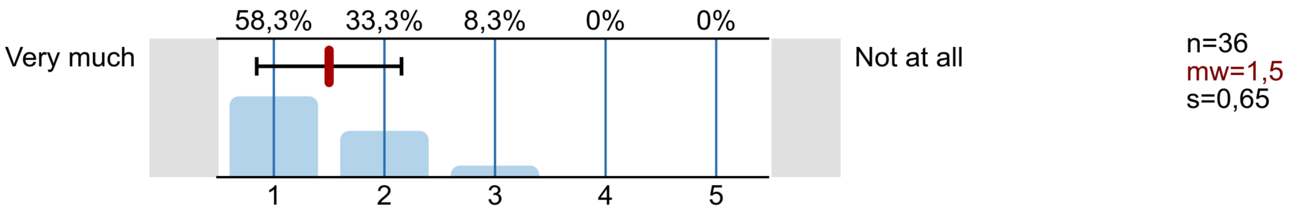
The amount of work required for this course is...



How large is the amount of work for this course?

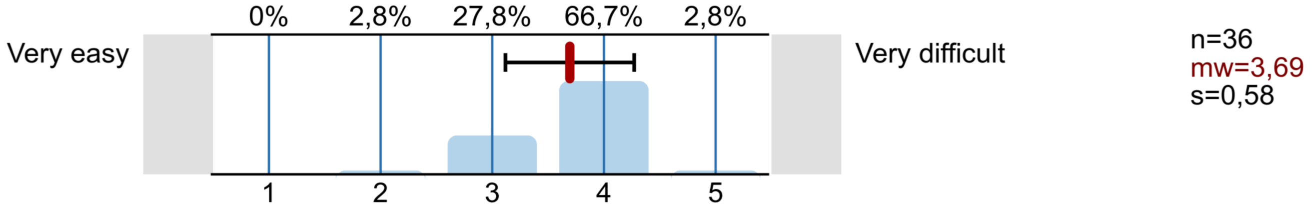


How much do you like attending this course?

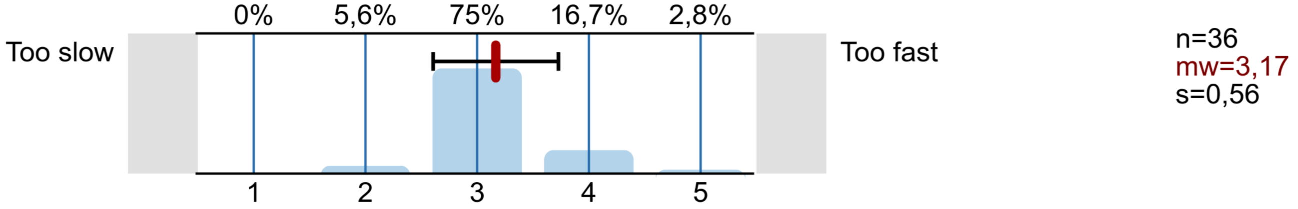


Evaluation Results

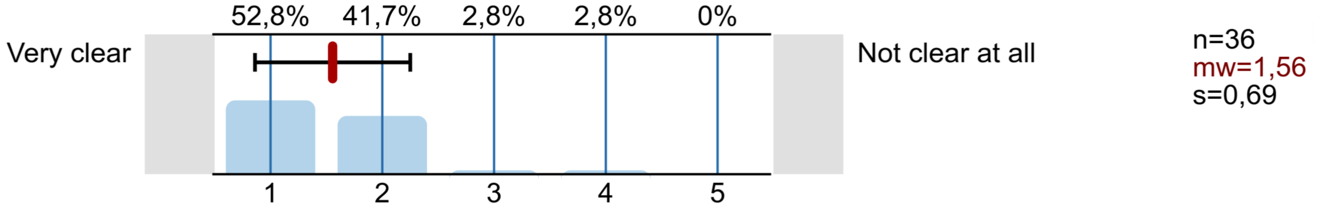
Difficulty of Contents



Speed

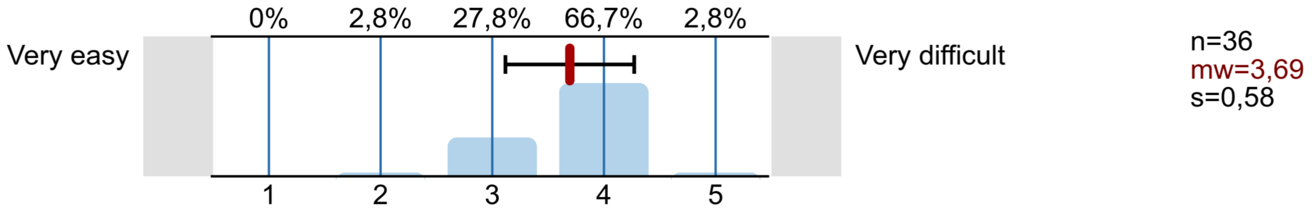


Clarity (using helpful examples)

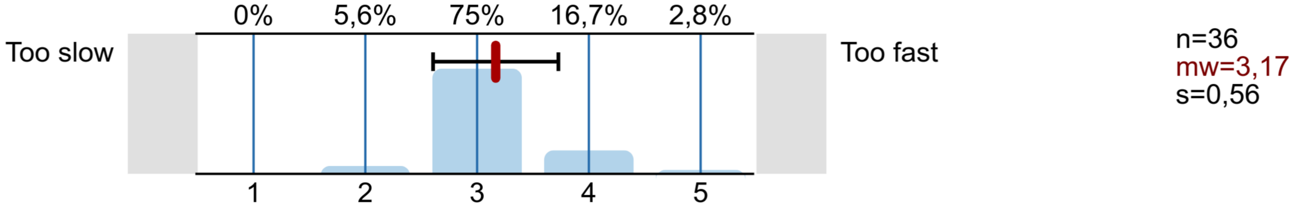


Evaluation Results

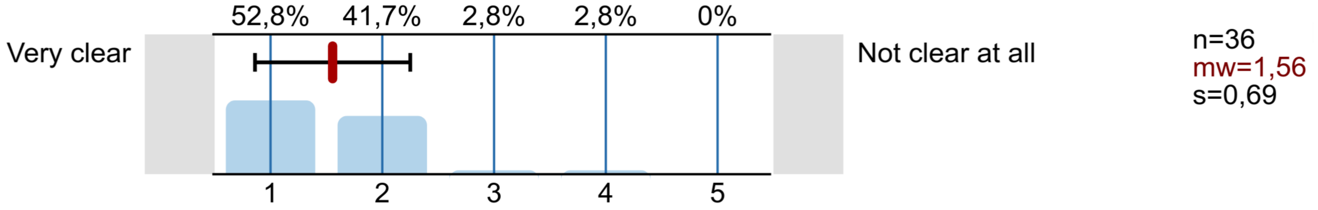
Difficulty of Contents



Speed



Clarity (using helpful examples)



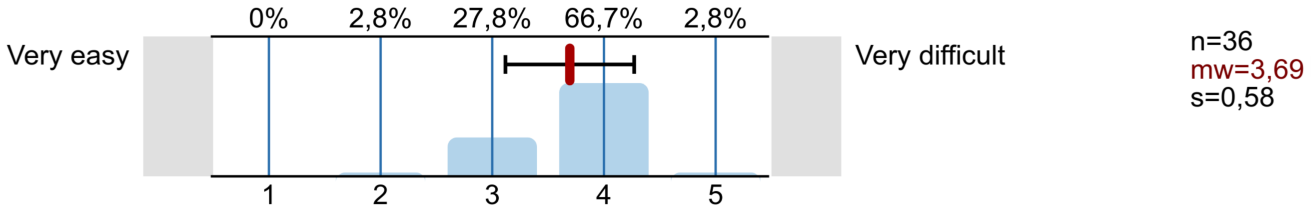
+

−

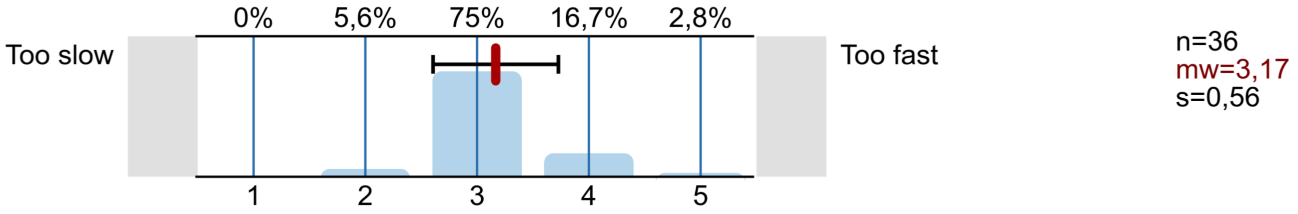
- sheets are too difficult/time-consuming

Evaluation Results

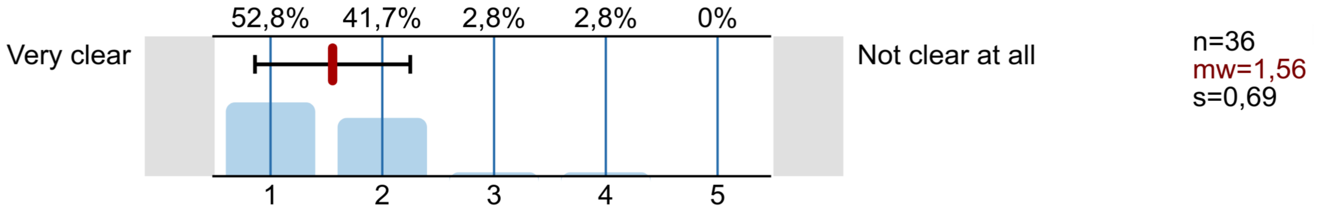
Difficulty of Contents



Speed



Clarity (using helpful examples)



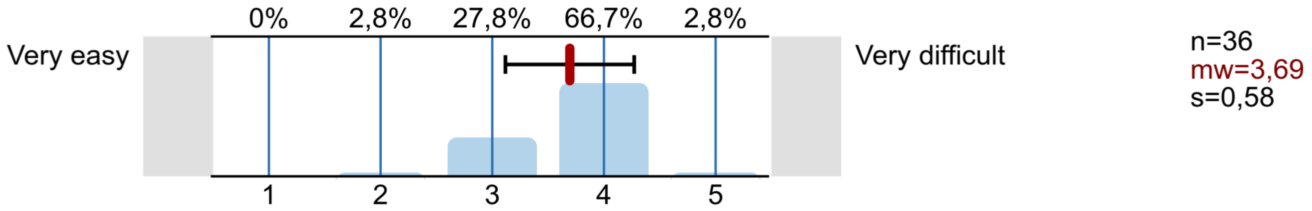
+

−

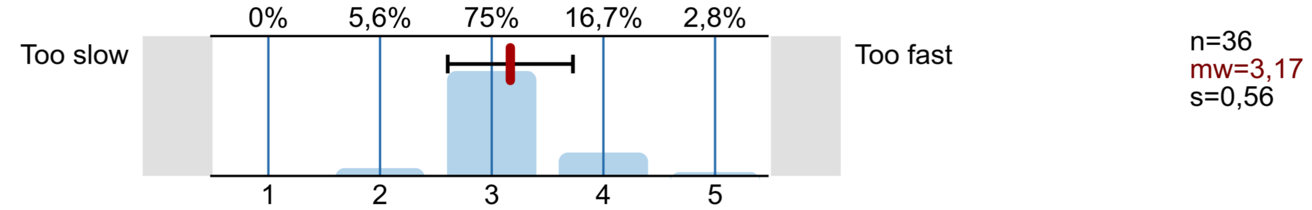
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading

Evaluation Results

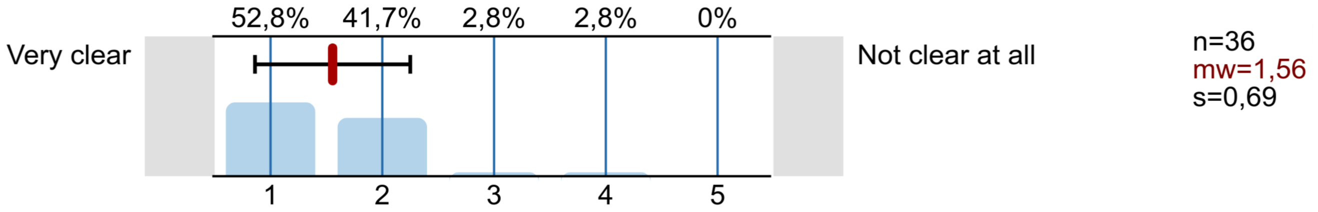
Difficulty of Contents



Speed



Clarity (using helpful examples)



+

- implementation tasks

−

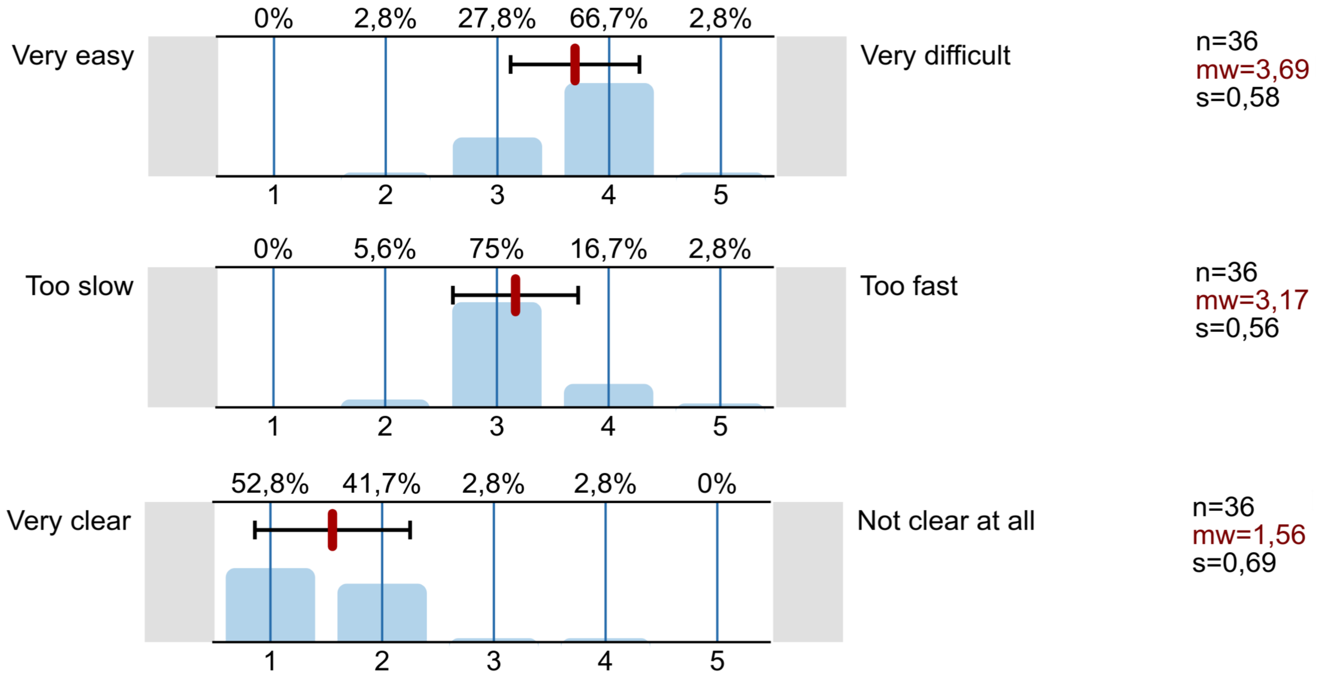
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks

Evaluation Results

Difficulty of Contents

Speed

Clarity (using helpful examples)



+

- implementation tasks

−

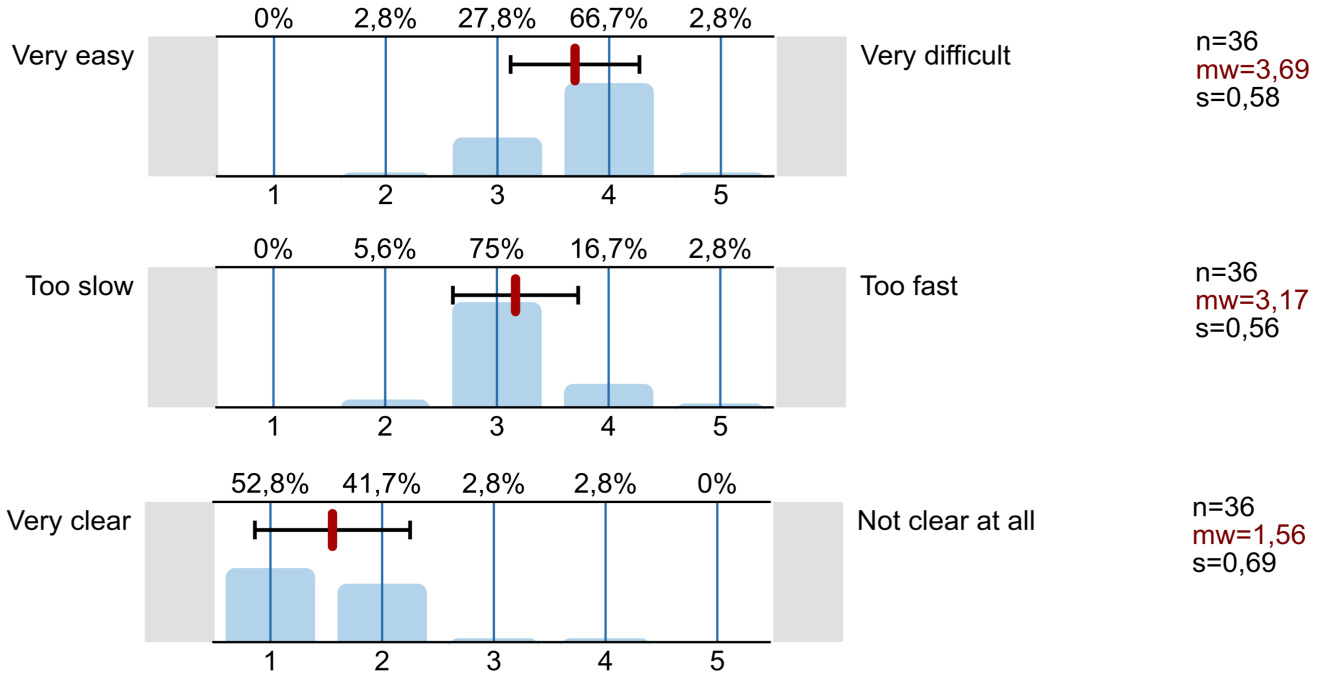
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks

Evaluation Results

Difficulty of Contents

Speed

Clarity (using helpful examples)



+

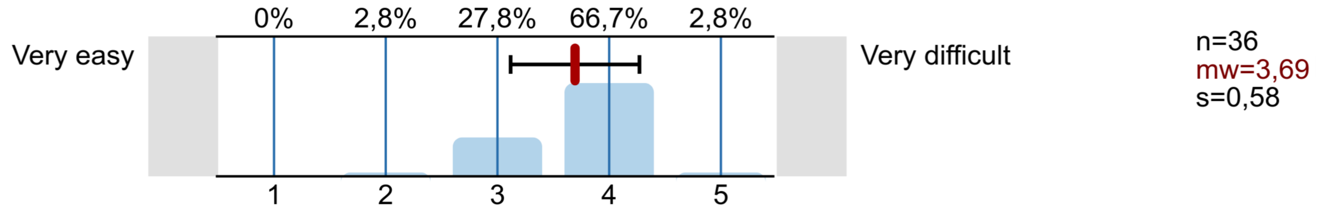
- implementation tasks
- discussion of old sheet

−

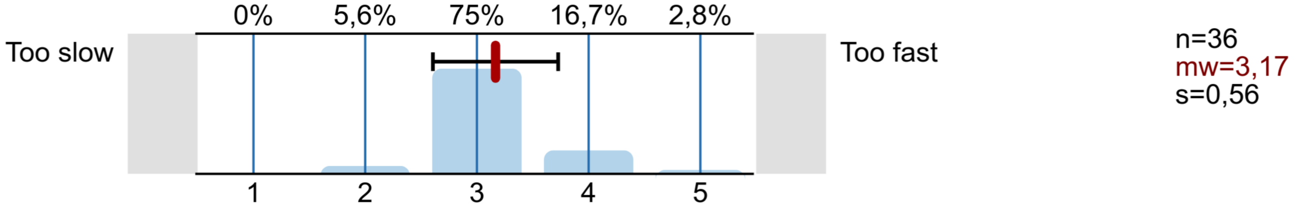
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks
- discussion of old sheet too long

Evaluation Results

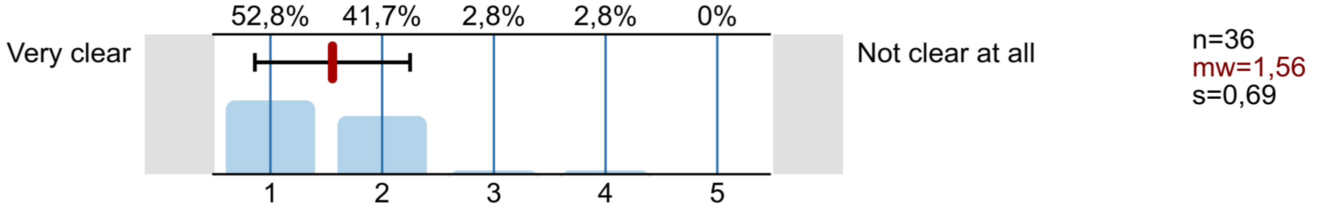
Difficulty of Contents



Speed



Clarity (using helpful examples)



+

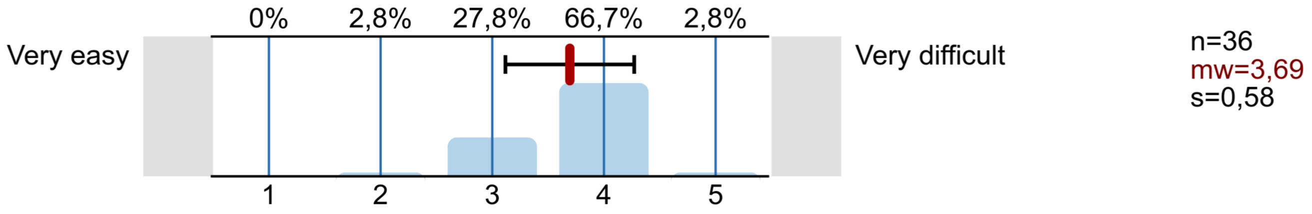
- implementation tasks
- discussion of old sheet

−

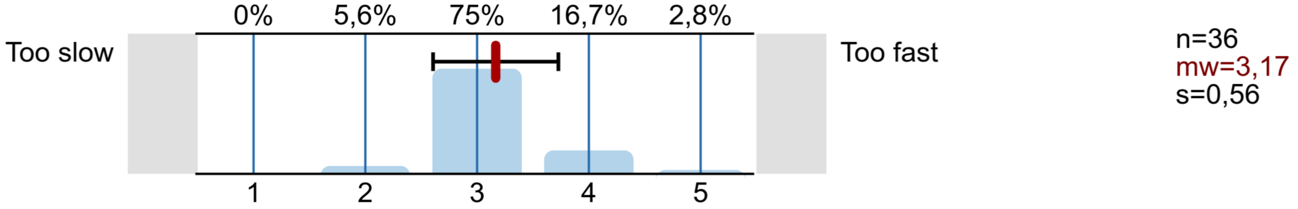
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks
- discussion of old sheet too long

Evaluation Results

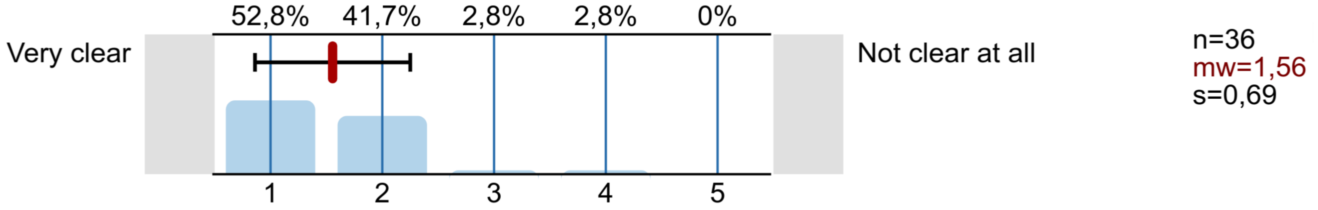
Difficulty of Contents



Speed



Clarity (using helpful examples)



+

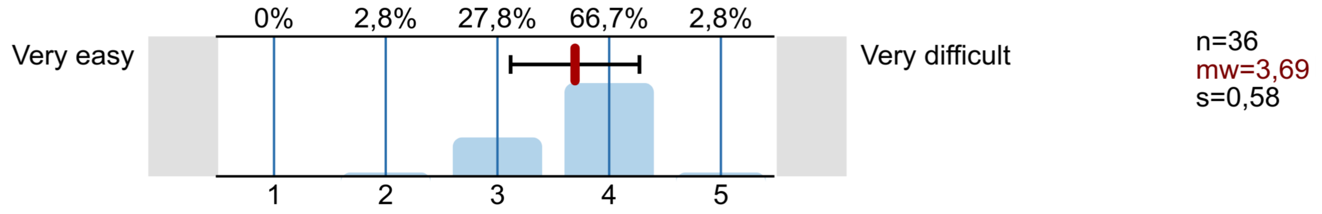
- implementation tasks
- discussion of old sheet
- good ratio between old and new problems

−

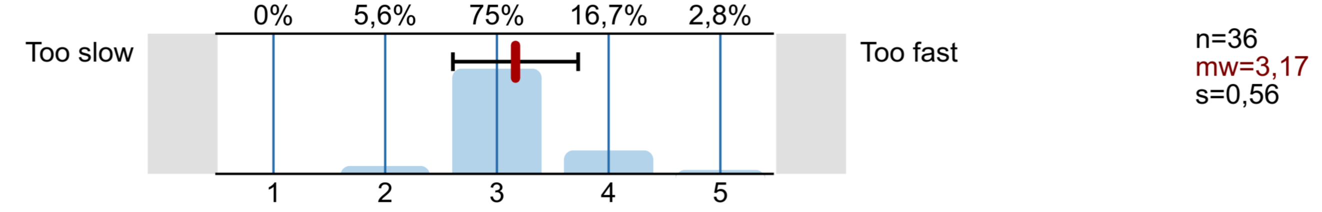
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks
- discussion of old sheet too long

Evaluation Results

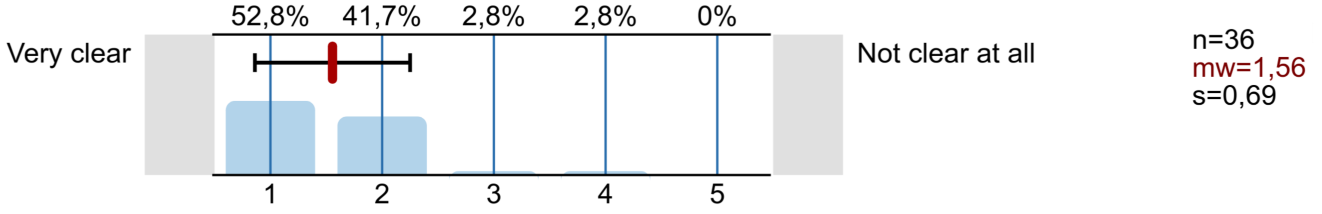
Difficulty of Contents



Speed



Clarity (using helpful examples)



+

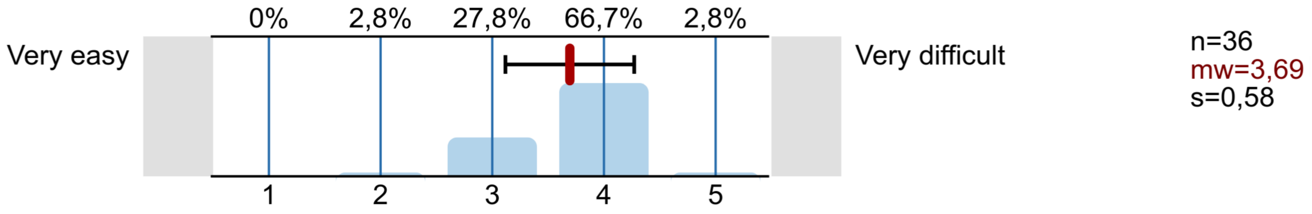
- implementation tasks
- discussion of old sheet
- good ratio between old and new problems

–

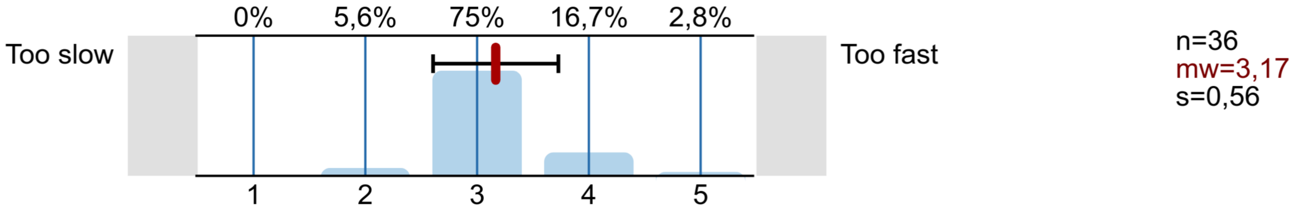
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks
- discussion of old sheet too long
- too little encouragement to work together

Evaluation Results

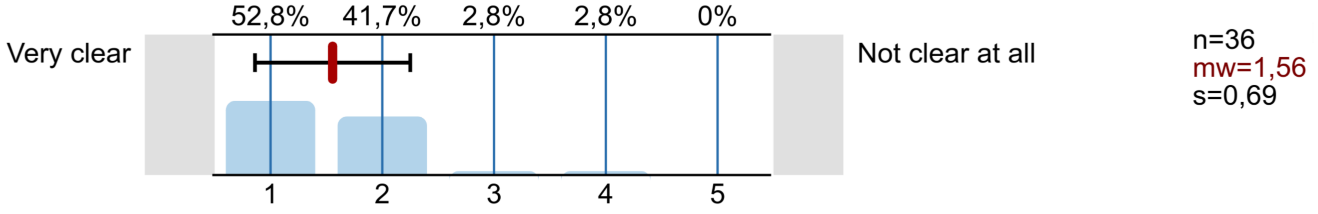
Difficulty of Contents



Speed



Clarity (using helpful examples)



+

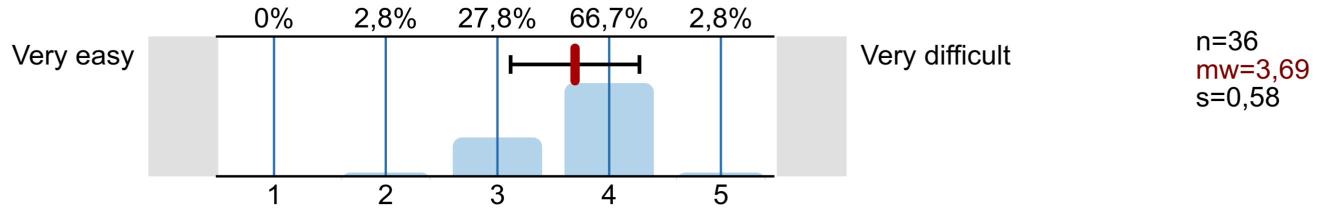
- problem solving and personal advice
- implementation tasks
- discussion of old sheet
- good ratio between old and new problems

−

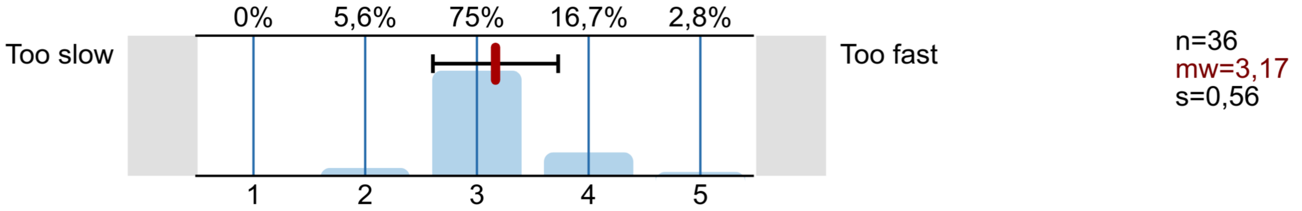
- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks
- discussion of old sheet too long
- too little encouragement to work together

Evaluation Results

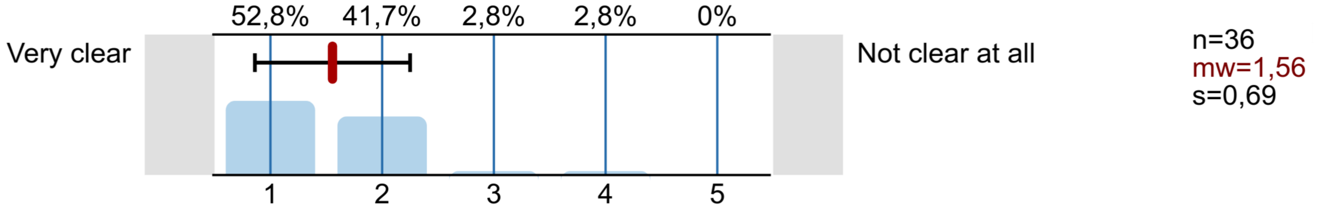
Difficulty of Contents



Speed



Clarity (using helpful examples)



+

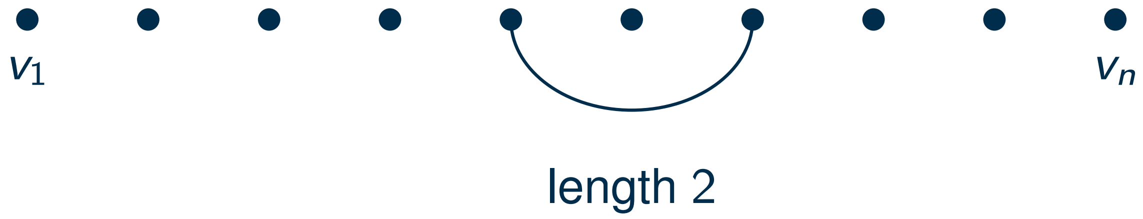
- problem solving and personal advice
- hints for new sheet
- implementation tasks
- discussion of old sheet
- good ratio between old and new problems

−

- sheets are too difficult/time-consuming
- problem solving more fun than paper reading
- implementation tasks
- discussion of old sheet too long
- too little encouragement to work together

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

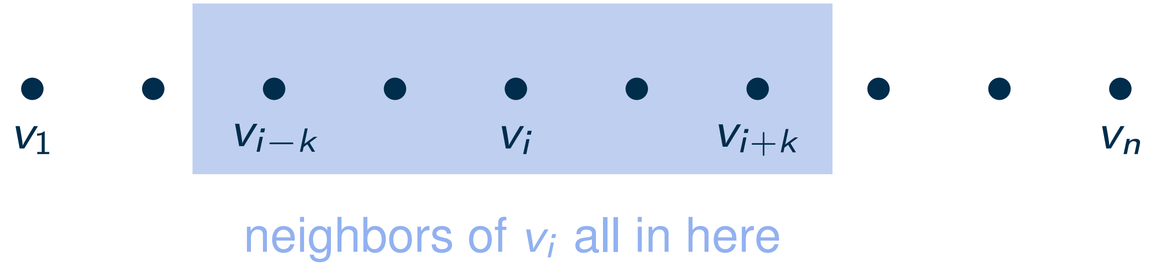


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

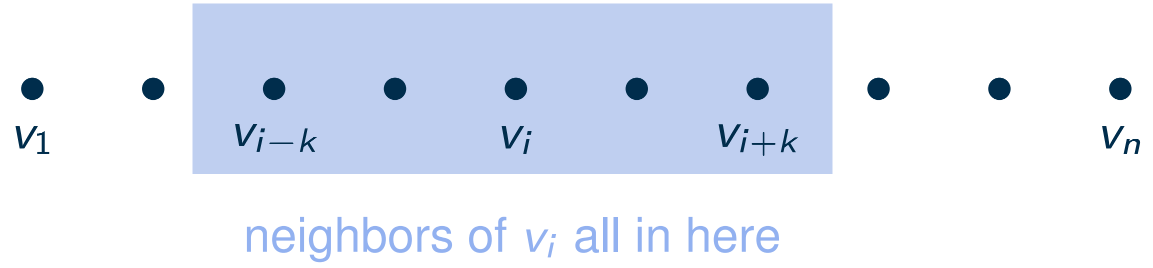
Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.



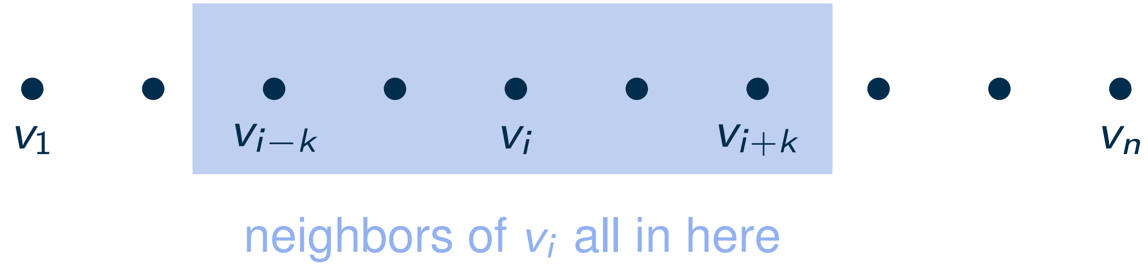
Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. **Provide an FPT algorithm** to find a minimum dominating set.



Sheet 6 - Dominating Set on Special Graphs

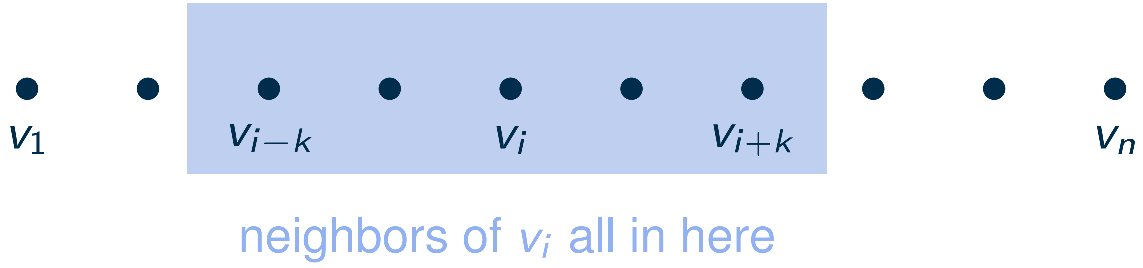
Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.



Courcelle?

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.



Courcelle?

can be used to show containment in FPT, but only implicitly gives us algorithm...

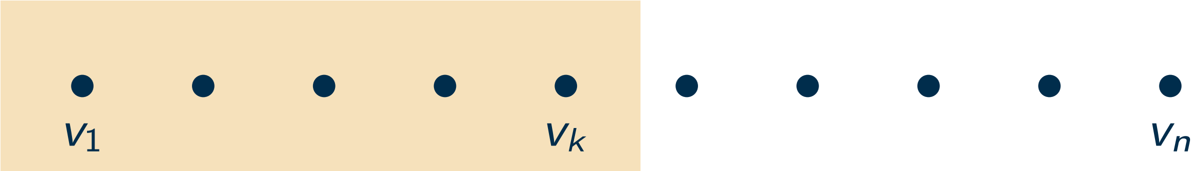
Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

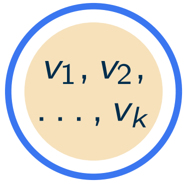
- Nice path decomposition of width $k - 1$:

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

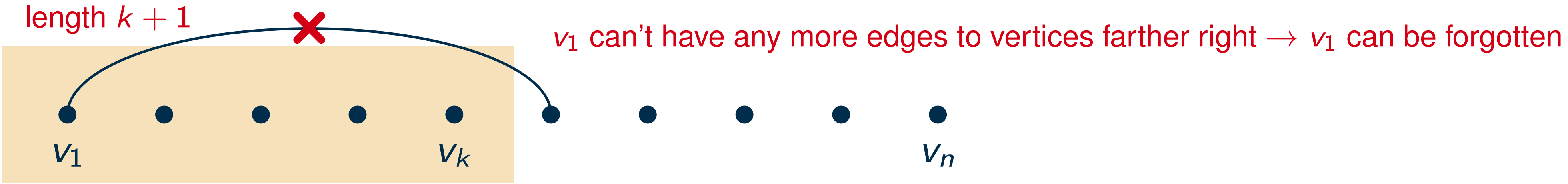


- Nice path decomposition of width $k - 1$:

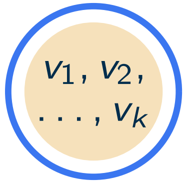


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

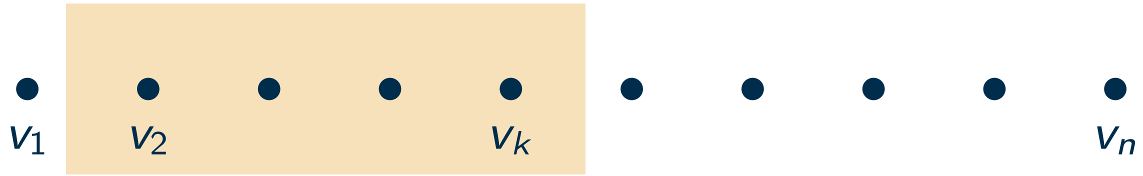


- Nice path decomposition of width $k-1$:

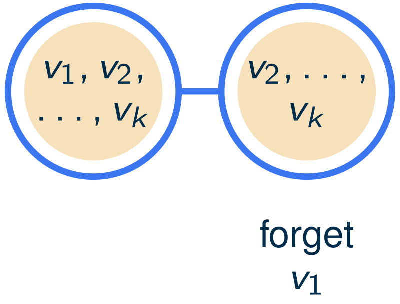


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

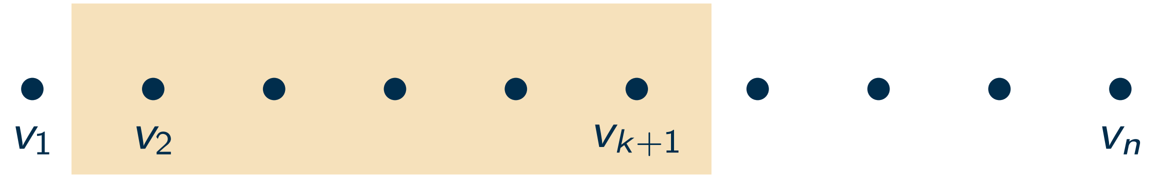


- Nice path decomposition of width $k - 1$:

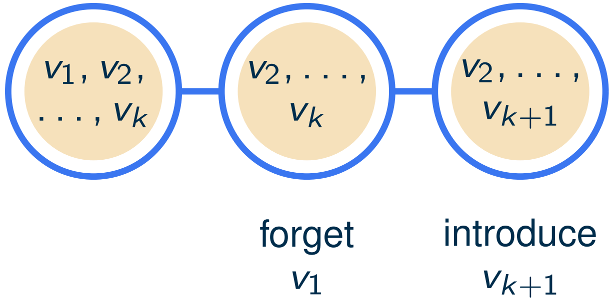


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

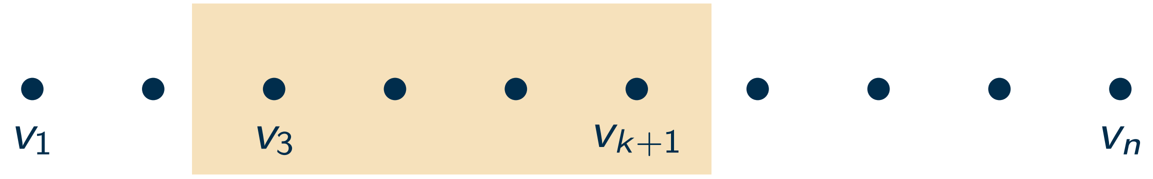


- Nice path decomposition of width $k-1$:

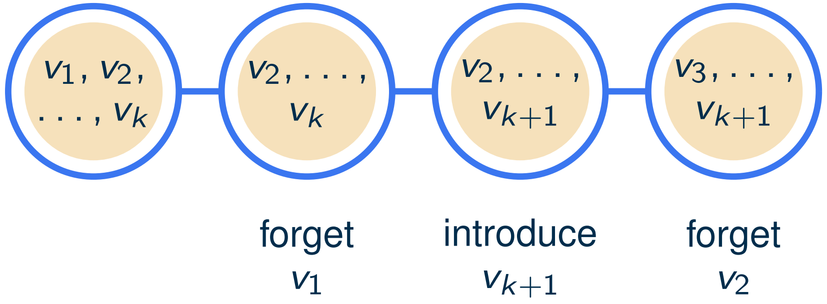


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

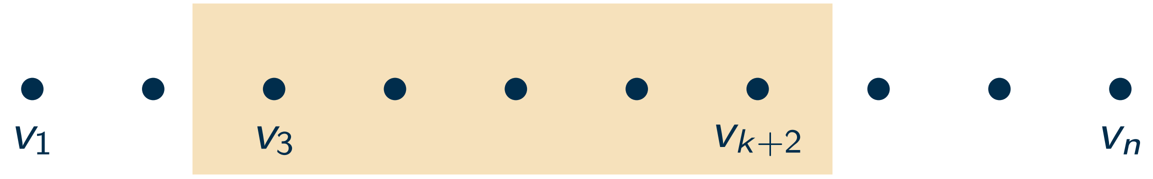


- Nice path decomposition of width $k-1$:

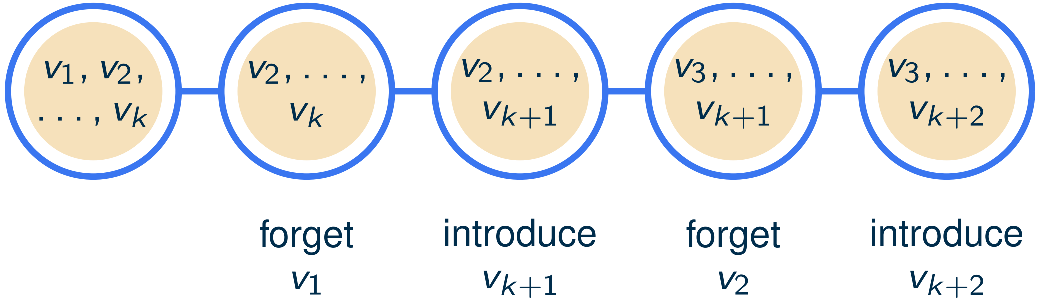


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

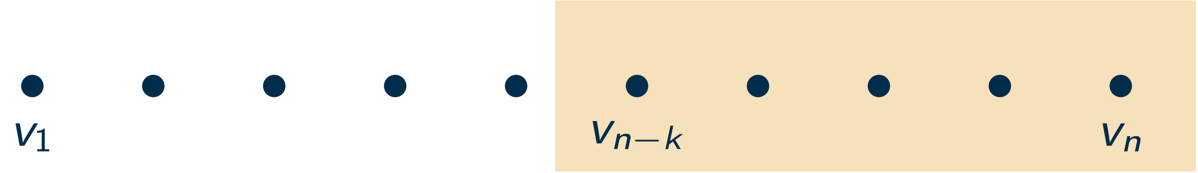


- Nice path decomposition of width $k - 1$:

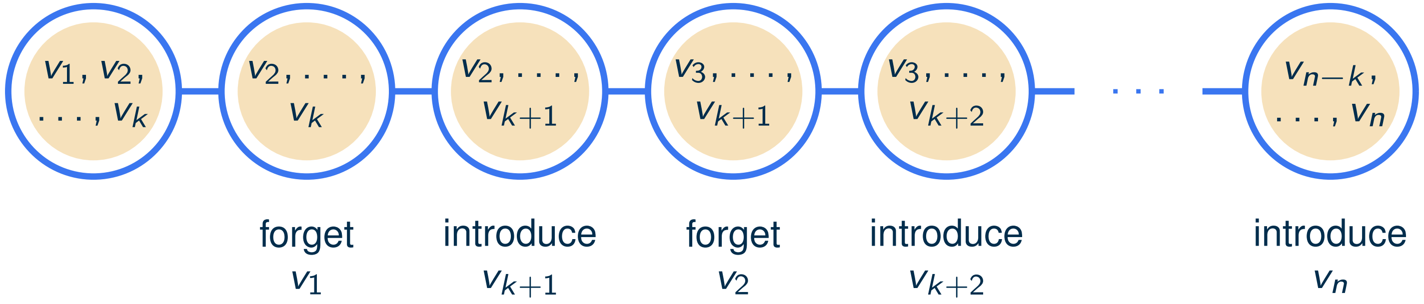


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

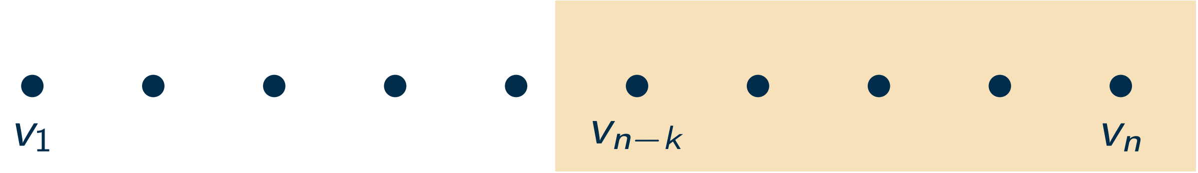


- Nice path decomposition of width $k - 1$:



Sheet 6 - Dominating Set on Special Graphs

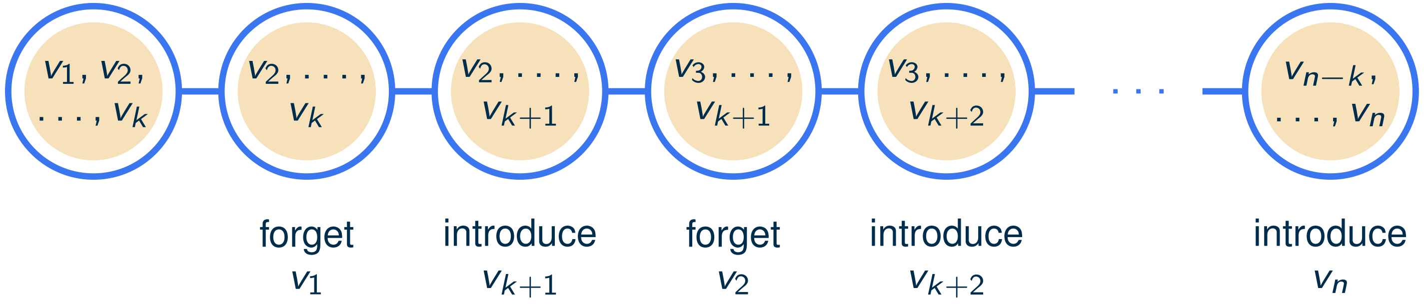
Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.



DP algorithm on path decomposition:

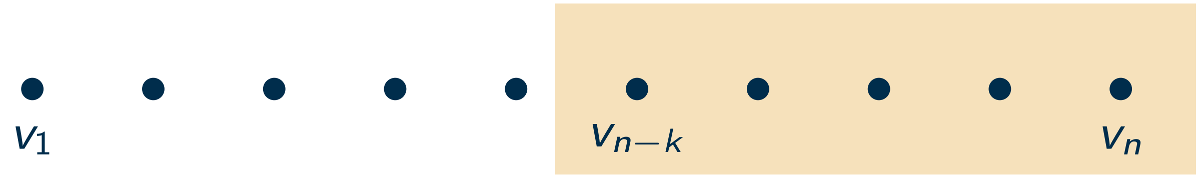
- how does a partial solution look like?

- Nice path decomposition of width $k - 1$:

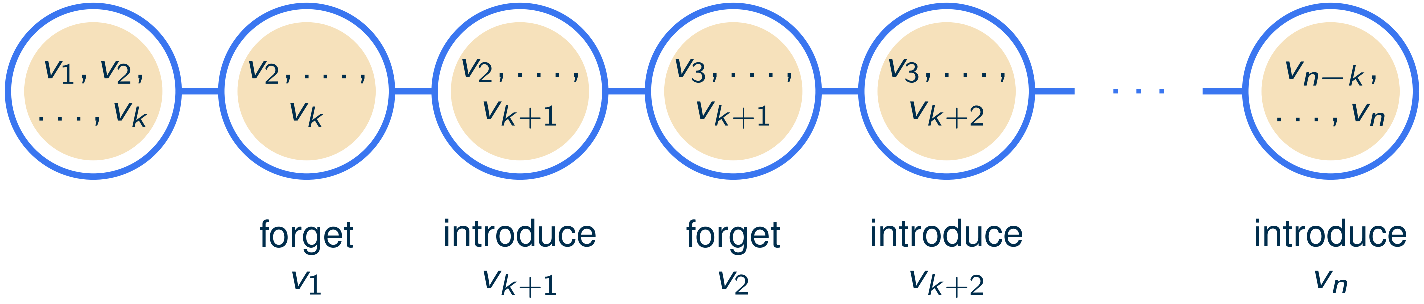


Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

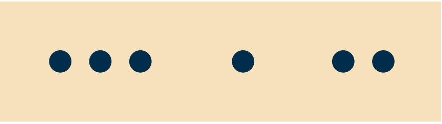


- Nice path decomposition of width $k-1$:



DP algorithm on path decomposition:

- how does a partial solution look like?

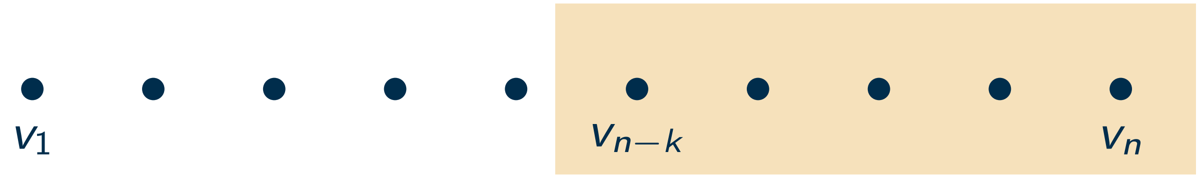


$+$ current size of DomSet

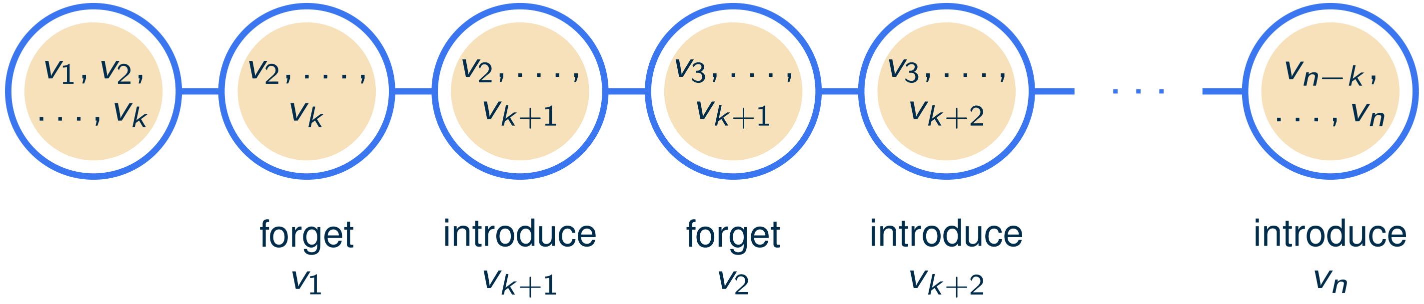
vertices of current bag

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

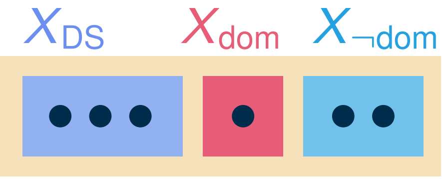


- Nice path decomposition of width $k - 1$:



DP algorithm on path decomposition:

- how does a partial solution look like?



$+$ current size of DomSet

vertices of current bag

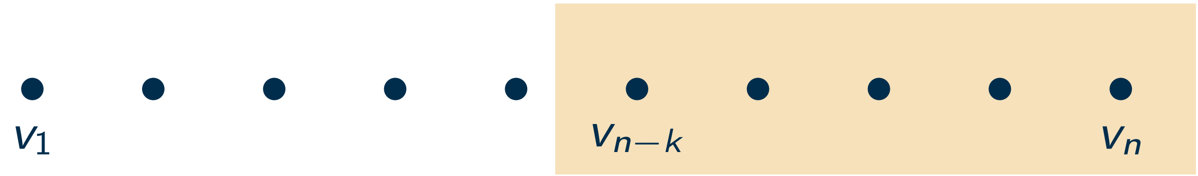
X_{DS}: contained in the DomSet

X_{dom}: not in DomSet, but dominated

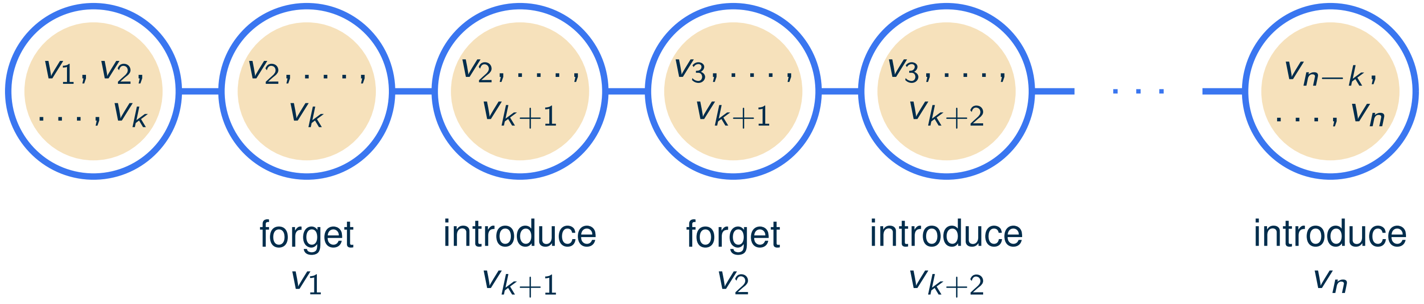
$X_{\neg\text{dom}}$: not in DomSet, not dominated

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

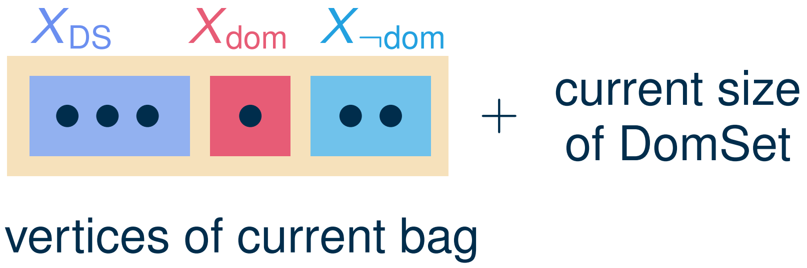


- Nice path decomposition of width $k - 1$:



DP algorithm on path decomposition:

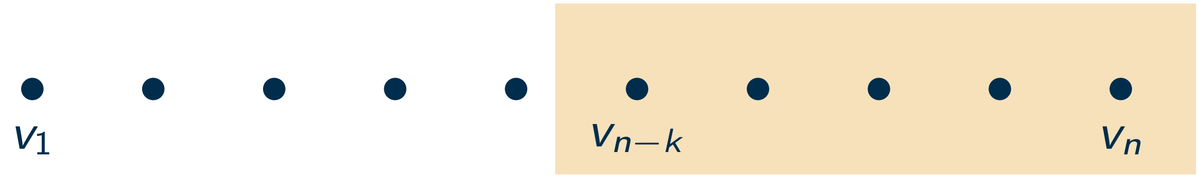
- how does a partial solution look like?



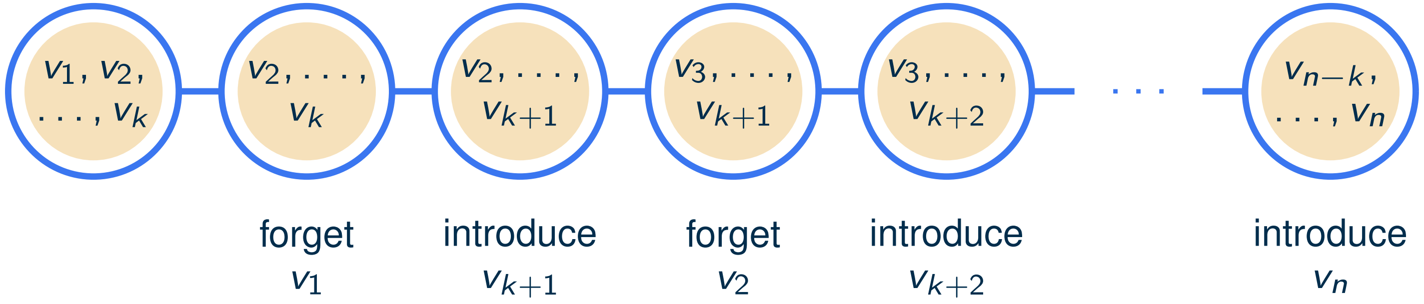
- what do we have to do for introduce nodes and forget nodes?

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \ldots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

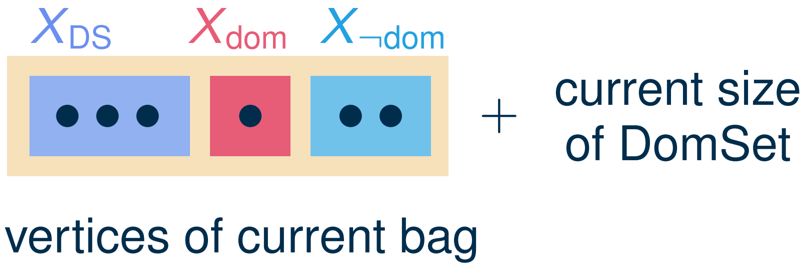


- Nice path decomposition of width $k - 1$:



DP algorithm on path decomposition:

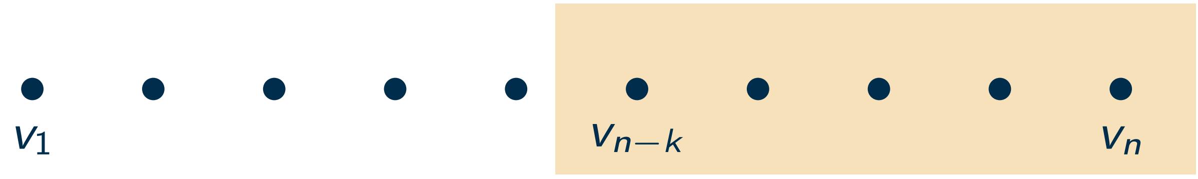
- how does a partial solution look like?



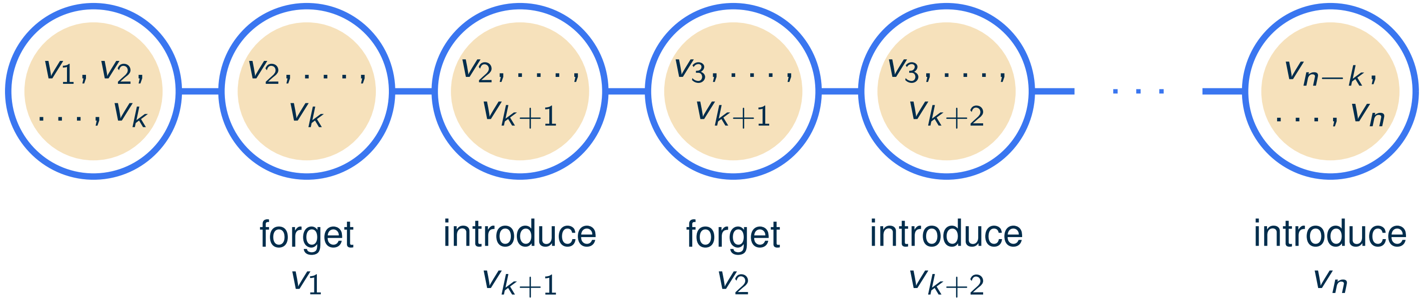
- what do we have to do for introduce nodes and forget nodes?
- how do we obtain overall solution?

Sheet 6 - Dominating Set on Special Graphs

Given a graph G and an ordering of the vertices $V = \{v_1, \dots, v_n\}$. The *length* of an edge $\{v_i, v_j\}$ is $|i - j|$. The parameter k is the maximum edge length. Provide an FPT algorithm to find a minimum dominating set.

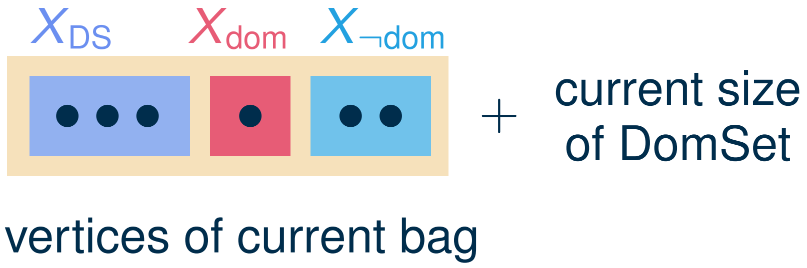


- Nice path decomposition of width $k - 1$:



DP algorithm on path decomposition:

- how does a partial solution look like?



- what do we have to do for introduce nodes and forget nodes?
- how do we obtain overall solution?
- runtime?

Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

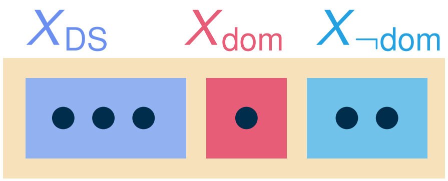
- how does a partial solution look like?

Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



$+$ current size of DomSet

vertices of current bag

X_{DS}: contained in the DomSet

X_{dom}: not in DomSet, but dominated

$X_{\neg dom}$: not in DomSet, not dominated

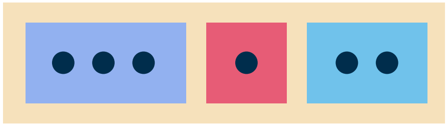
Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?

X_{DS} X_{dom} $X_{\neg\text{dom}}$



$+$ current size of DomSet

vertices of current bag

X_{DS}: contained in the DomSet

X_{dom}: not in DomSet, but dominated

$X_{\neg\text{dom}}$: not in DomSet, not dominated

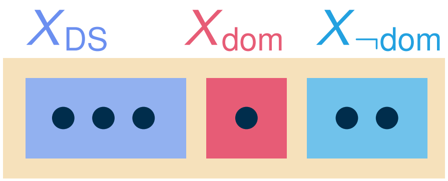
what else do we need to store?

Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?

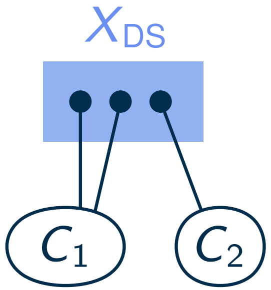


$+$ current size of DomSet

vertices of current bag

$+$

partition of X_{DS} into connected components

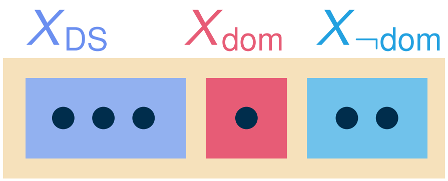


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



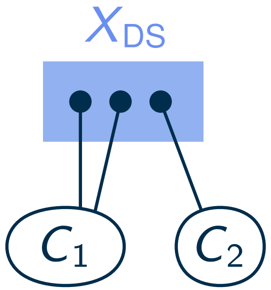
$+$ current size of DomSet

vertices of current bag

$+$

partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



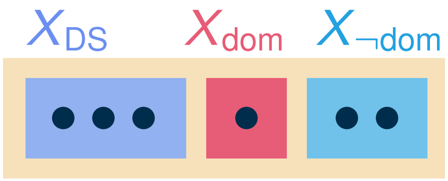
Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?

How do we keep track of connected components?



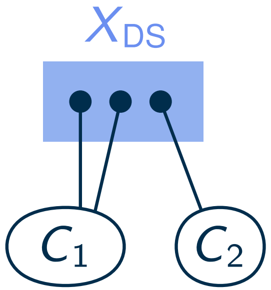
$+$ current size of DomSet

vertices of current bag

$+$

partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices

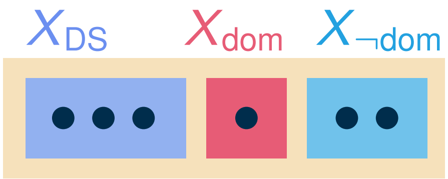


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



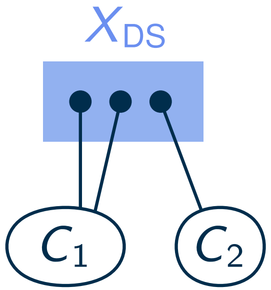
$+$ current size of DomSet

vertices of current bag

$+$

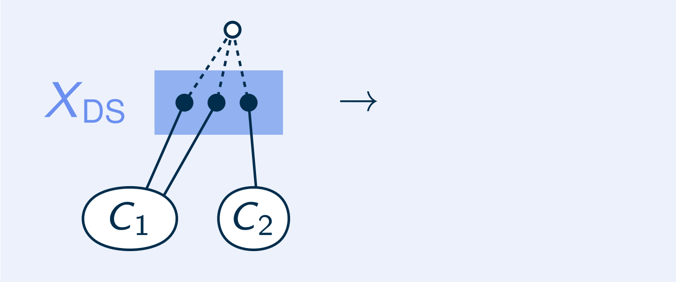
partition of X_{DS} into connected components

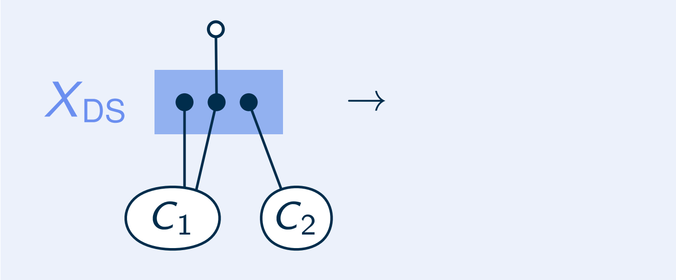
connection either in current X_{DS} or over already forgotten DS-vertices

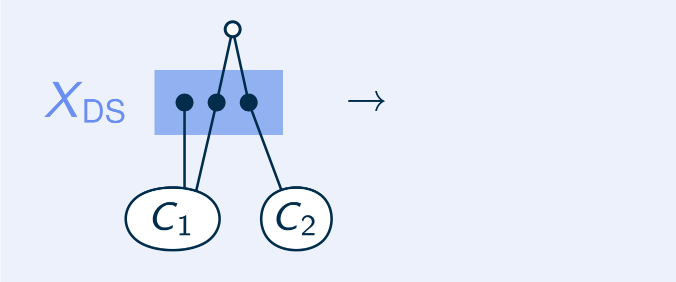


How do we keep track of connected components?

- Introduce node: update partition if added to X_{DS}





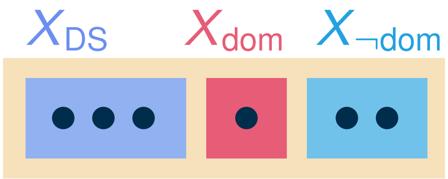


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



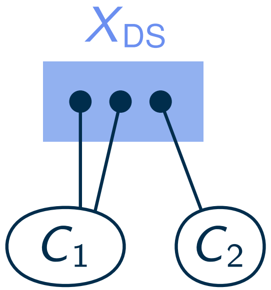
$+$ current size of DomSet

vertices of current bag

$+$

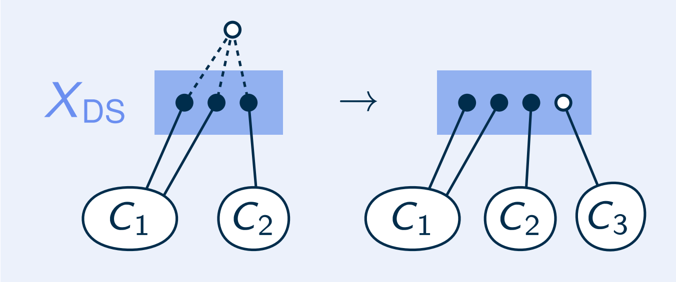
partition of X_{DS} into connected components

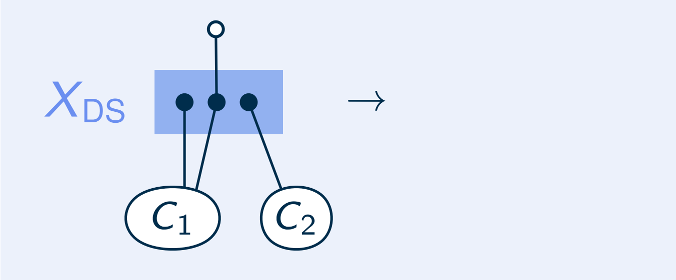
connection either in current X_{DS} or over already forgotten DS-vertices

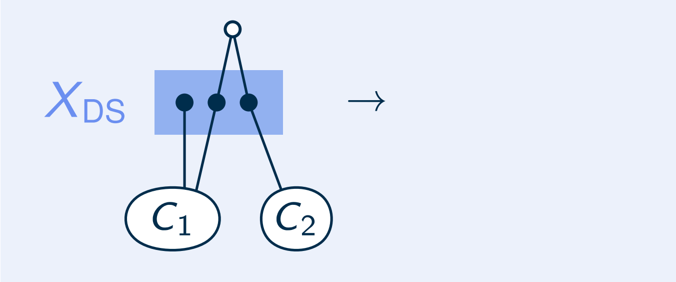


How do we keep track of connected components?

- Introduce node: update partition if added to X_{DS}





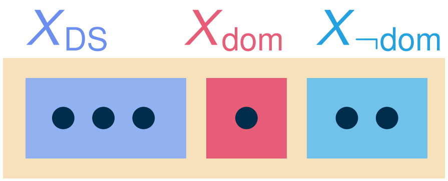


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



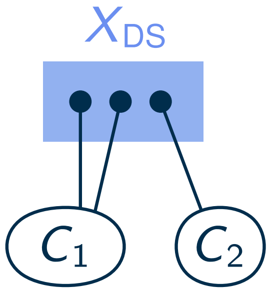
$+$ current size of DomSet

vertices of current bag

$+$

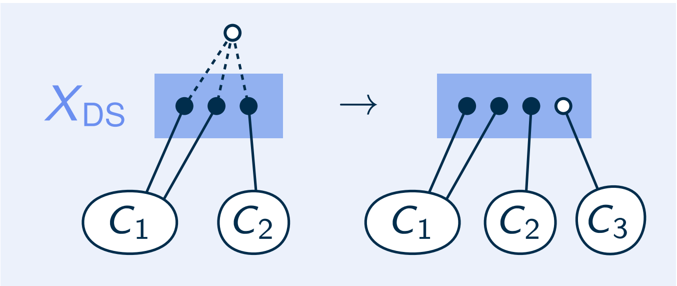
partition of X_{DS} into connected components

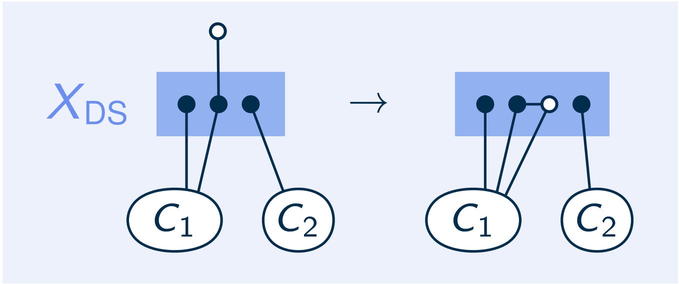
connection either in current X_{DS} or over already forgotten DS-vertices

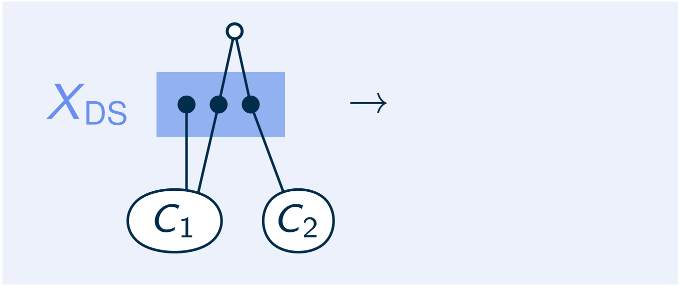


How do we keep track of connected components?

- Introduce node: update partition if added to X_{DS}





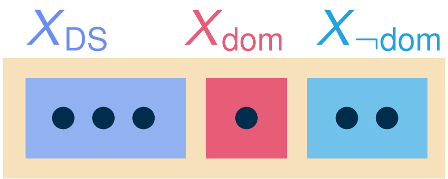


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



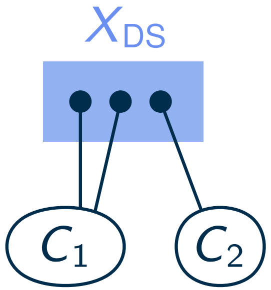
$+$ current size of DomSet

vertices of current bag

$+$

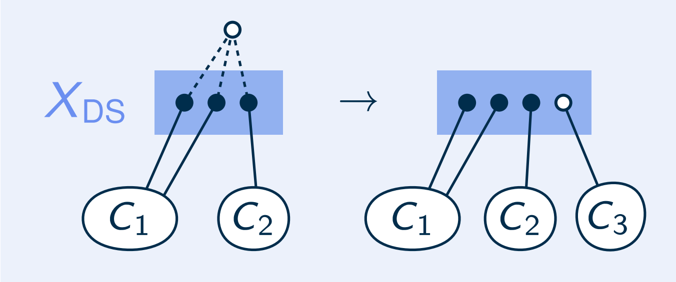
partition of X_{DS} into <u>connected</u> components

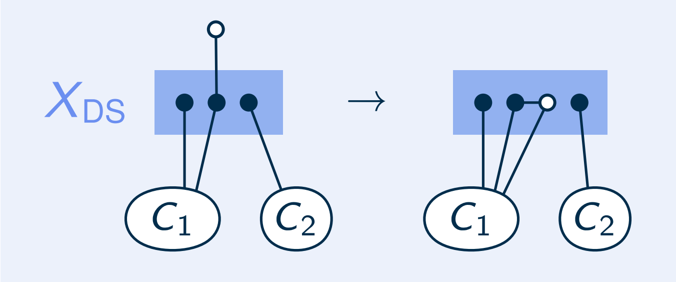
connection either in current X_{DS} or over already forgotten DS-vertices

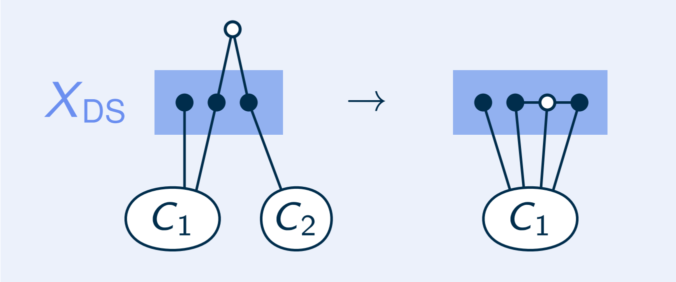


How do we keep track of connected components?

- Introduce node: update partition if added to X_{DS}





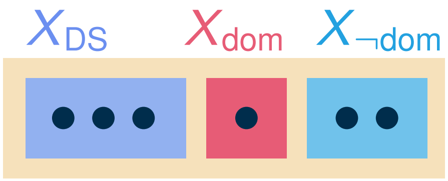


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



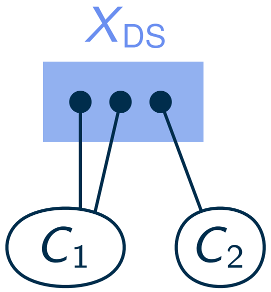
$+$ current size of DomSet

vertices of current bag

$+$

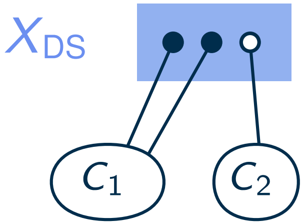
partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



How do we keep track of connected components?

- forget node: throw away partial solution if forgotten vertex is only the one in its component

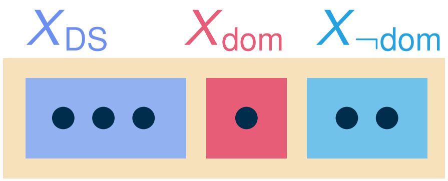


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



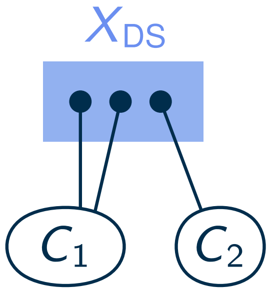
$+$ current size of DomSet

vertices of current bag

$+$

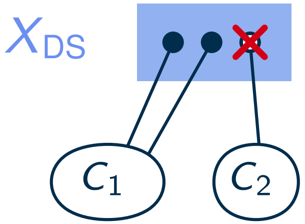
partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



How do we keep track of connected components?

- forget node: throw away partial solution if forgotten vertex is only the one in its component



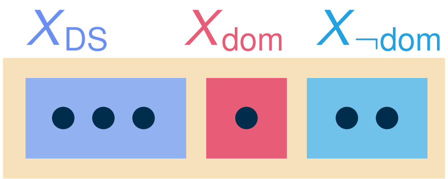
C_2 can not be connected to the rest of the DomSet anymore!

Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



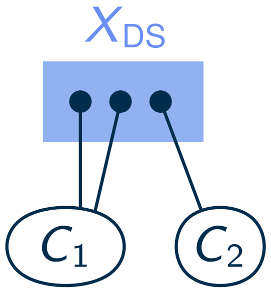
$+$ current size of DomSet

vertices of current bag

$+$

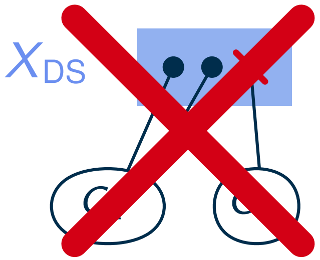
partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



How do we keep track of connected components?

- forget node: throw away partial solution if forgotten vertex is only the one in its component



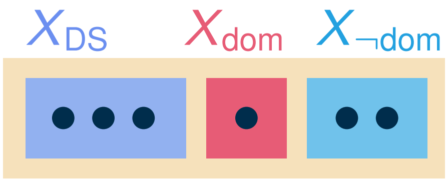
C_2 can not be connected to the rest of the DomSet anymore!

Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



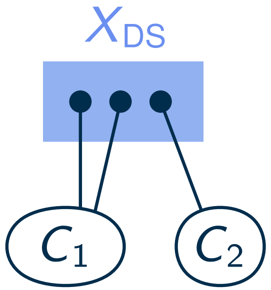
$+$ current size of DomSet

vertices of current bag

$+$

partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



How do we keep track of connected components?

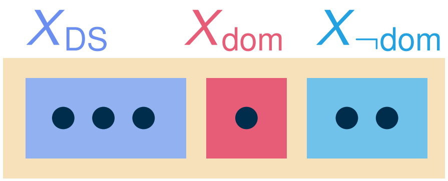
- join node: combine two partial solutions if X_{DS} is identical

Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



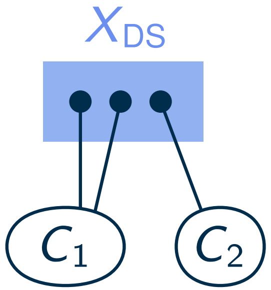
$+$ current size of DomSet

vertices of current bag

$+$

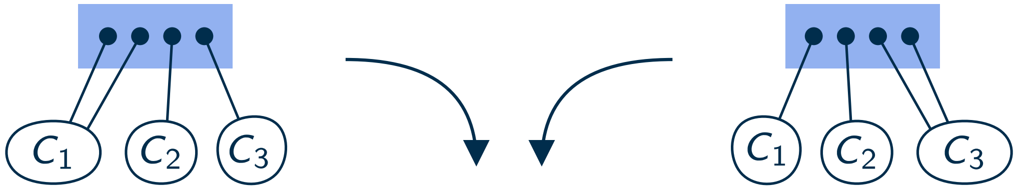
partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



How do we keep track of connected components?

- join node: combine two partial solutions if X_{DS} is identical
- merge connected components accordingly:

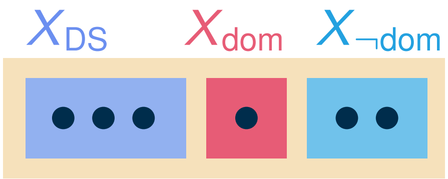


Sheet 6 - Connected Dominating Set

Given a graph G, a parameter k, and a tree decomposition of width k, provide an FPT algorithm to find a minimum connected dominating set. (use dynamic programming on the tree decomposition)

DP algorithm on tree decomposition:

- how does a partial solution look like?



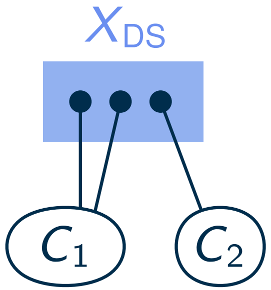
$+$ current size of DomSet

vertices of current bag

$+$

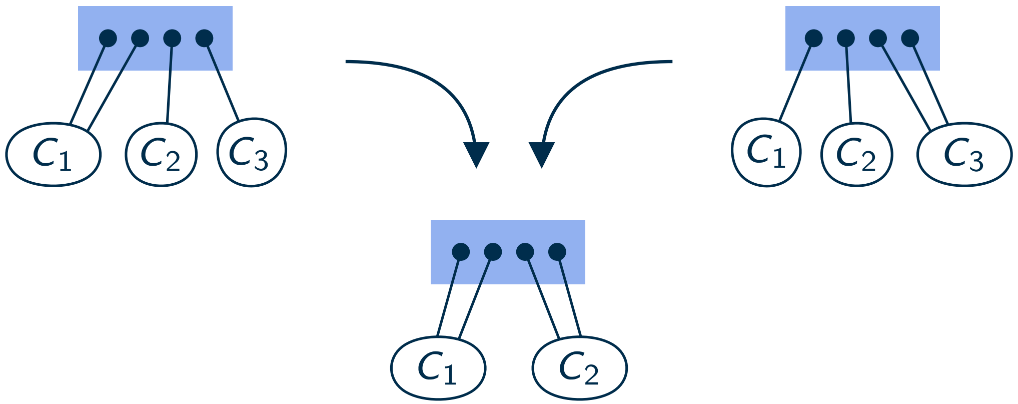
partition of X_{DS} into connected components

connection either in current X_{DS} or over already forgotten DS-vertices



How do we keep track of connected components?

- join node: combine two partial solutions if X_{DS} is identical
- merge connected components accordingly:



Recap: Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

instance (I, k)
of problem $\mathcal{L}$

$\longrightarrow$

instance (I', k')
of problem $\mathcal{L}'$

with $k' \leq g(k)$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$

Recap: Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

instance (I, k) of problem $\mathcal{L}$ $\Rightarrow$ equivalent instance (I', k') of problem $\mathcal{L}'$ with $k' \leq g(k)$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$
- (I, k) yes-instance $\Leftrightarrow$ (I', k') yes-instance

Recap: Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

instance (I, k) of problem $\mathcal{L}$

FPT time $f(k) \cdot |I|^{O(1)}$

equivalent instance (I', k') of problem $\mathcal{L}'$

with $k' \leq g(k)$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$
- (I, k) yes-instance $\Leftrightarrow$ (I', k') yes-instance
- the map must be computable in FPT-time $f(k) \cdot |I|^{O(1)}$

(f and g computable)

Recap: Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

instance (I, k) of problem $\mathcal{L}$	FPT time $f(k) \cdot \lvert I \rvert^{O(1)}$ $\Rightarrow$	equivalent instance (I', k') of problem $\mathcal{L}'$	with $k' \leq g(k)$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$
- (I, k) yes-instance $\Leftrightarrow$ (I', k') yes-instance
- the map must be computable in FPT-time $f(k) \cdot \lvert I \rvert^{O(1)}$

(f and g computable)

Implications

- FPT algorithm for $\mathcal{L}'$ $\Rightarrow$ FPT algorithm for $\mathcal{L}$

Recap: Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

instance (I, k) of problem $\mathcal{L}$ → FPT time $f(k) \cdot |I|^{O(1)}$ → equivalent instance (I', k') of problem $\mathcal{L}'$ with $k' \leq g(k)$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$
- (I, k) yes-instance $\Leftrightarrow$ (I', k') yes-instance
- the map must be computable in FPT-time $f(k) \cdot |I|^{O(1)}$

(f and g computable)

Implications

- FPT algorithm for $\mathcal{L}'$ $\Rightarrow$ FPT algorithm for $\mathcal{L}$
- problem $\mathcal{L}$ known to be $W[t]$-hard $\Rightarrow$ problem $\mathcal{L}'$ is $W[t]$-hard

Recap: Parameterized reduction

Reduction from problem $\mathcal{L}$ to problem $\mathcal{L}'$

instance (I, k) of problem $\mathcal{L}$ → FPT time $f(k) \cdot |I|^{O(1)}$ → equivalent instance (I', k') of problem $\mathcal{L}'$ with $k' \leq g(k)$

- map each instance (I, k) of $\mathcal{L}$ to instance (I', k') of $\mathcal{L}'$ with $k' \leq g(k)$
- (I, k) yes-instance $\Leftrightarrow$ (I', k') yes-instance
- the map must be computable in FPT-time $f(k) \cdot |I|^{O(1)}$

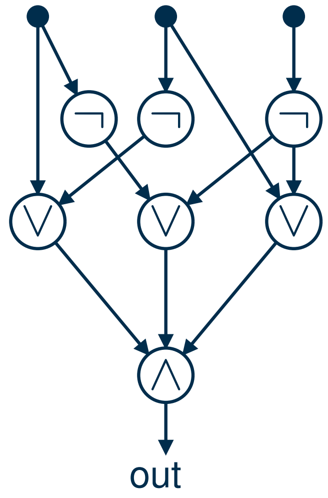
(f and g computable)

Implications

- FPT algorithm for $\mathcal{L}'$ $\Rightarrow$ FPT algorithm for $\mathcal{L}$
- problem $\mathcal{L}$ known to be $W[t]$-hard $\Rightarrow$ problem $\mathcal{L}'$ is $W[t]$-hard
- problem $\mathcal{L}'$ known to be $\in W[t]$ $\Rightarrow$ problem $\mathcal{L} \in W[t]$

Recap: W-Hierarchy

Problem: WEIGHTED CIRCUIT SATISFIABILITY (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

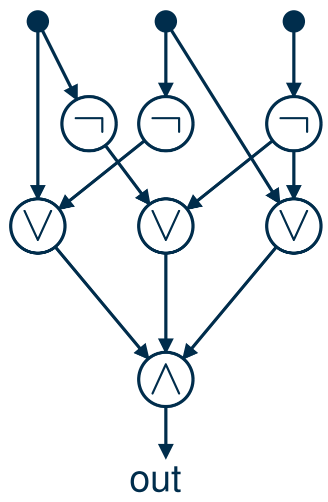


Recap: W-Hierarchy

Problem: WEIGHTED CIRCUIT SATISFIABILITY (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path



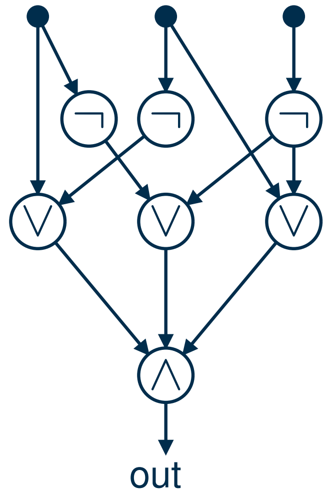
Recap: W-Hierarchy

Problem: WEIGHTED CIRCUIT SATISFIABILITY (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path

***W*[t]** is the class of problems that have a parameterized reduction to WCS[t].



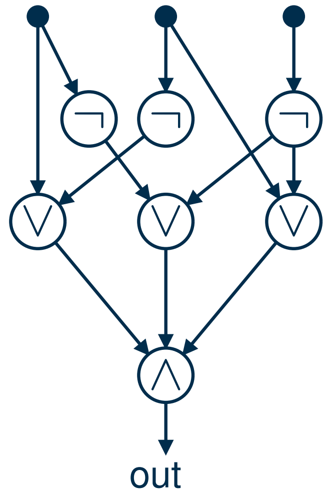
Recap: W-Hierarchy

Problem: Weighted Circuit Satisfiability (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path

$W[t]$ is the class of problems that have a parameterized reduction to WCS[t].



FPT

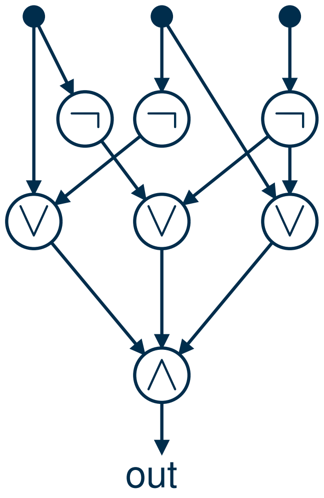
Recap: W-Hierarchy

Problem: Weighted Circuit Satisfiability (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path

$\boldsymbol{W[t]}$ is the class of problems that have a parameterized reduction to WCS[t].



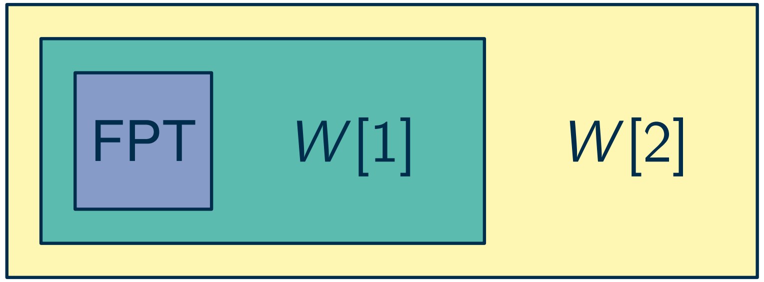
Recap: W-Hierarchy

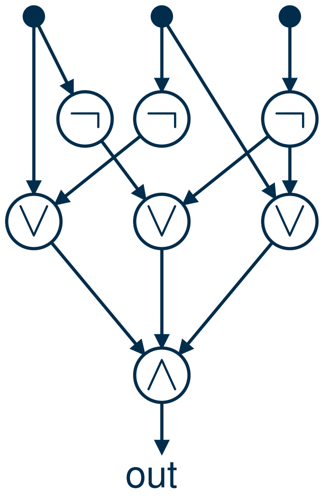
Problem: Weighted Circuit Satisfiability (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path

$W[t]$ is the class of problems that have a parameterized reduction to WCS[t].





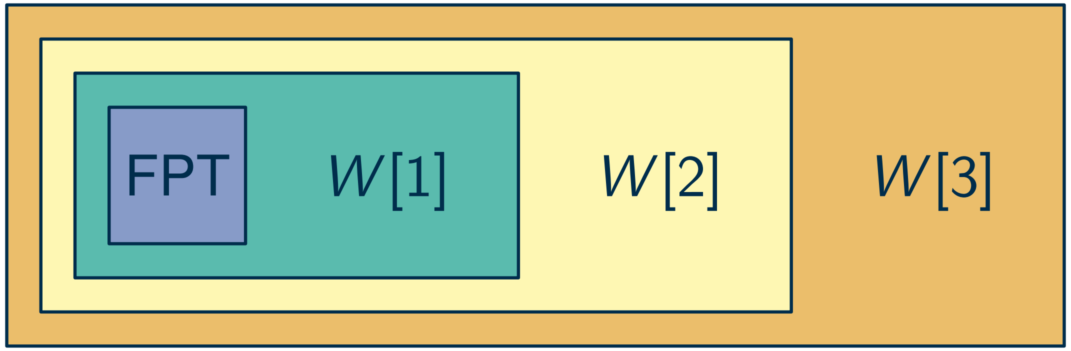
Recap: W-Hierarchy

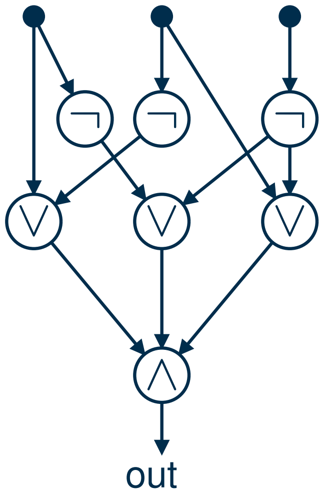
Problem: Weighted Circuit Satisfiability (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path

$W[t]$ is the class of problems that have a parameterized reduction to WCS[t].





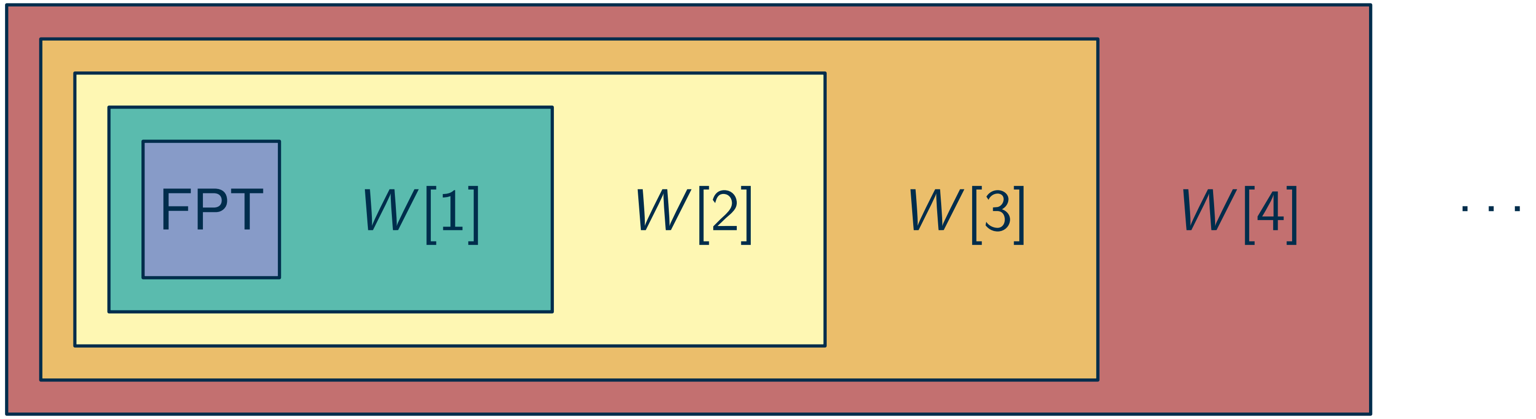
Recap: W-Hierarchy

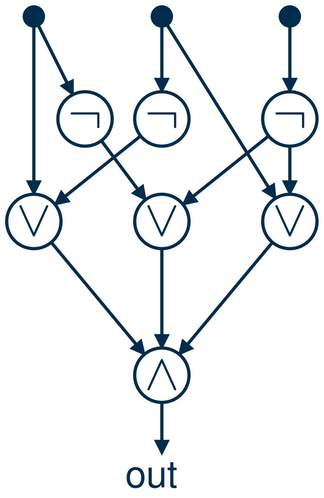
Problem: WEIGHTED CIRCUIT SATISFIABILITY (WCS)
Given a Boolean circuit and a parameter k, is there a satisfying assignment of weight exactly k?

Problem WCS[t]: restricts WCS to circuits of constant height and weft $\leq t$.

weft: largest number of nodes with in-degree > 2 on a directed path

$W[t]$ is the class of problems that have a parameterized reduction to WCS[t].





Recap: Showing $W[t]$-hardness

problem for which
I want to show
$W[t]$-hardness

Recap: Showing $W[t]$-hardness

problem known to be $W[t]$-hard

parameterized reduction

problem for which I want to show $W[t]$-hardness

Recap: Showing $W[t]$-hardness

problem known to be $W[t]$-hard → parameterized reduction → problem for which I want to show $W[t]$-hardness

which problem should I use?

Recap: Showing $W[t]$-hardness

problem known to be $W[t]$-hard

parameterized reduction

problem for which I want to show $W[t]$-hardness

which problem should I use?

$W[1]$-hard problems:

- Indpendent Set
- Clique
- Multicolored Independent Set
- Multicolored Clique
- List Coloring
- Strongly Connected Steiner Subgraph
- ...
- WCS[1] (often not a good choice)

Recap: Showing $W[t]$-hardness

problem known to be $W[t]$-hard

parameterized reduction

problem for which I want to show $W[t]$-hardness

which problem should I use?

often good idea to try one of these first

$W[1]$-hard problems:

- INDPENDENT SET
- CLIQUE
- MULTICOLORED INDEPENDENT SET
- MULTICOLORED CLIQUE
- LIST COLORING
- STRONGLY CONNECTED STEINER SUBGRAPH
- . . .
- WCS[1] (often not a good choice)

Recap: Showing $W[t]$-hardness

problem known to be $W[t]$-hard

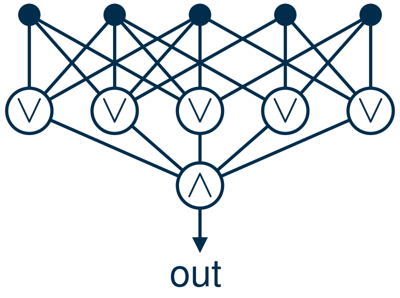
parameterized reduction

problem for which I want to show $W[t]$-hardness

which problem should I use?

$W[2]$-hard problems:

- Dominating Set
- Hitting Set
- WCS[2] (often not a good choice)
- Weighted Monotone 2-Normalized Satisfiability



Recap: Showing $W[t]$-hardness

problem known to be $W[t]$-hard

parameterized reduction

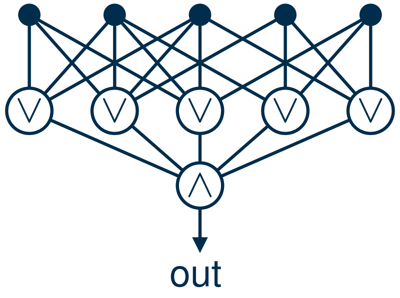
problem for which I want to show $W[t]$-hardness

which problem should I use?

often good idea to try this first

$W[2]$-hard problems:

- DOMINATING SET
- HITTING SET
- WCS[2] (often not a good choice)
- WEIGHTED MONOTONE 2-NORMALIZED SATISFIABILITY



Hierarchical Problems

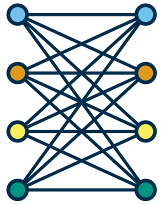
Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ⋆

Hierarchical Problems

Is there a parameterized reduction from Vertex Cover to Independent Set? If so, provide one. If not, argue why not. ⋆

- Show $W[1]$-hardness for:

Multicolored Biclique Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful? ⋆⋆

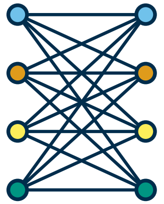

$K_{4,4}$

Hierarchical Problems

Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

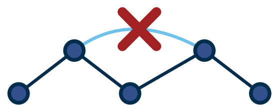
- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful? ★★



$K_{4,4}$

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k? ★★★

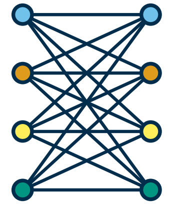


Hierarchical Problems

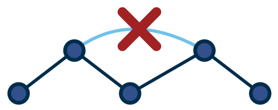
Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful? ★★

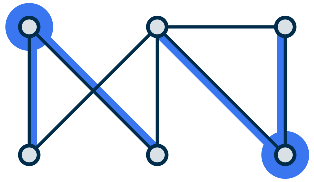


LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k? ★★★



- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S? ★★★

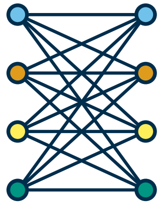


Hierarchical Problems

Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

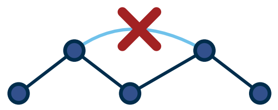
- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful? ★★



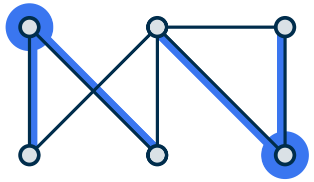
$K_{4,4}$

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k? ★★★



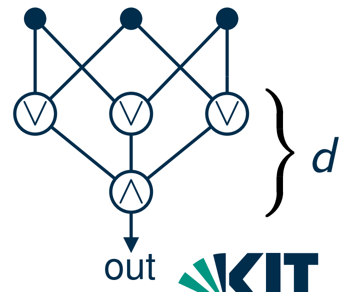
- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S? ★★★



- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$? ★★★★

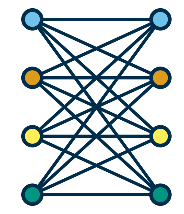


Hierarchical Problems

Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

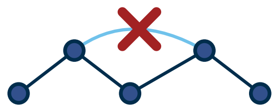
- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful? ★★



$K_{4,4}$

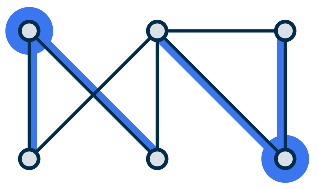
LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k? ★★★



hints on the next slide

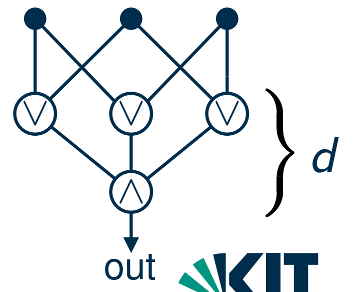
- Determine the parameterize

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S? ★★★



- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$? ★★★★



Hierarchical Problems

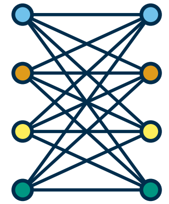
Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful?

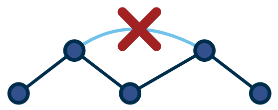
reduce from MULTICOLORED CLIQUE

★★



LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

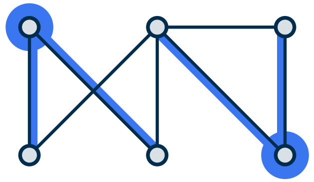
★★★



- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S?

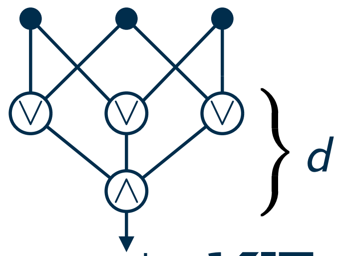
★★★



- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

★★★★



Hierarchical Problems

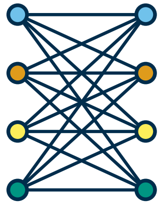
Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful?

reduce from MULTICOLORED CLIQUE

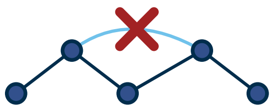
★★


$K_{4,4}$

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

★★★

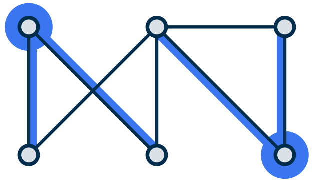
reduce from MULTICOLORED INDEPENDENT SET



- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S?

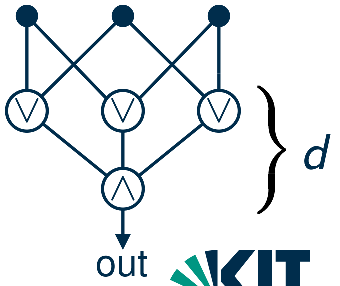
★★★



- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

★★★★



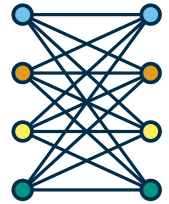
Hierarchical Problems

Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

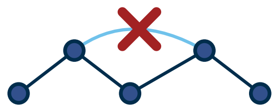
MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful?

reduce from MULTICOLORED CLIQUE ★★



LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

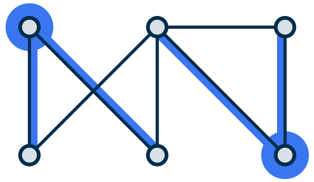
★★★ reduce from MULTICOLORED INDEPENDENT SET



- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

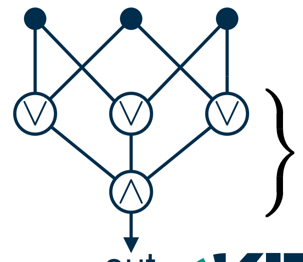
INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S? ★★★

reduce from DOMINATING SET and to WCS[2]



- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$? ★★★★



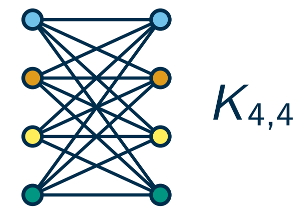
Hierarchical Problems

Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

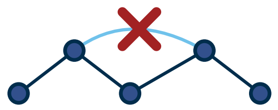
MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful?

reduce from MULTICOLORED CLIQUE ★★



LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

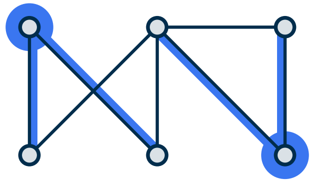
reduce from MULTICOLORED INDEPENDENT SET ★★★



- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S? ★★★

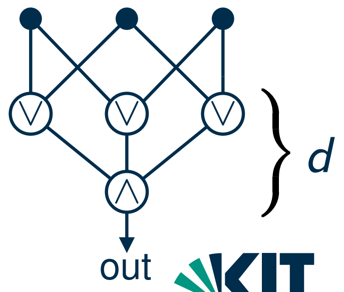
reduce from DOMINATING SET and to WCS[2]



- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$? ★★★★

how many high in-degree gates are there?



Hierarchical Problems

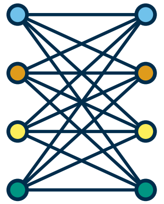
Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful?

reduce from MULTICOLORED CLIQUE

★★

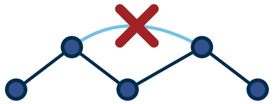


$K_{4,4}$

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

reduce from MULTICOLORED INDEPENDENT SET

★★★

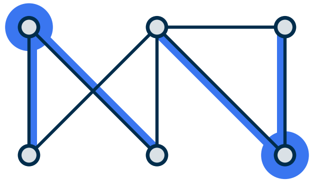


- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S?

★★★

reduce from DOMINATING SET and to WCS[2]



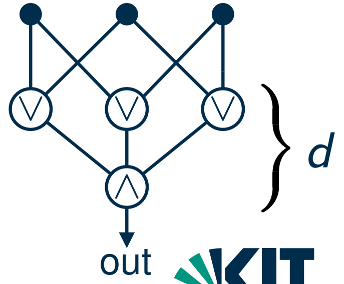
- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

★★★★

how many high in-degree gates are there?

on how many circuit inputs do the parents of the high in-degree gates depend?



Hierarchical Problems

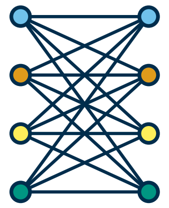
Is there a parameterized reduction from VERTEX COVER to INDEPENDENT SET? If so, provide one. If not, argue why not. ★

- Show $W[1]$-hardness for:

MULTICOLORED BICLIQUE Given a colored bipartite graph $G = (V, E)$ with k colors, does G contain the complete bipartite graph $K_{k,k}$ as an induced subgraph, where every side of the bipartition is colorful?

reduce from MULTICOLORED CLIQUE

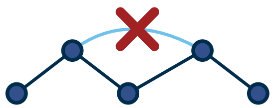
★★



$K_{4,4}$

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

★★★

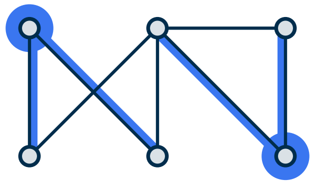


reduce from MULTICOLORED INDEPENDENT SET

- Determine the parameterized complexity ($W[t]$-completeness for some t) of:

INDEPENDENT DOMINATING SET: Given a graph $G = (V, E)$ and a parameter k, does G contain an independent set S of size k s.t. each vertex in $V \setminus S$ is adjacent to a vertex in S?

★★★

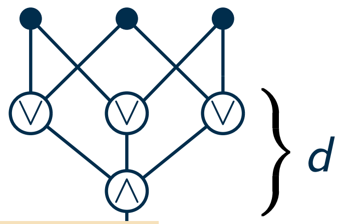


reduce from DOMINATING SET and to WCS[2]

- Find an FPT algorithm for:

Monotone Weft-1 Circuits: Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

★★★★



how many high in-degree gates are there?

on how many circuit inputs do the parents of the high in-degree gates depend?

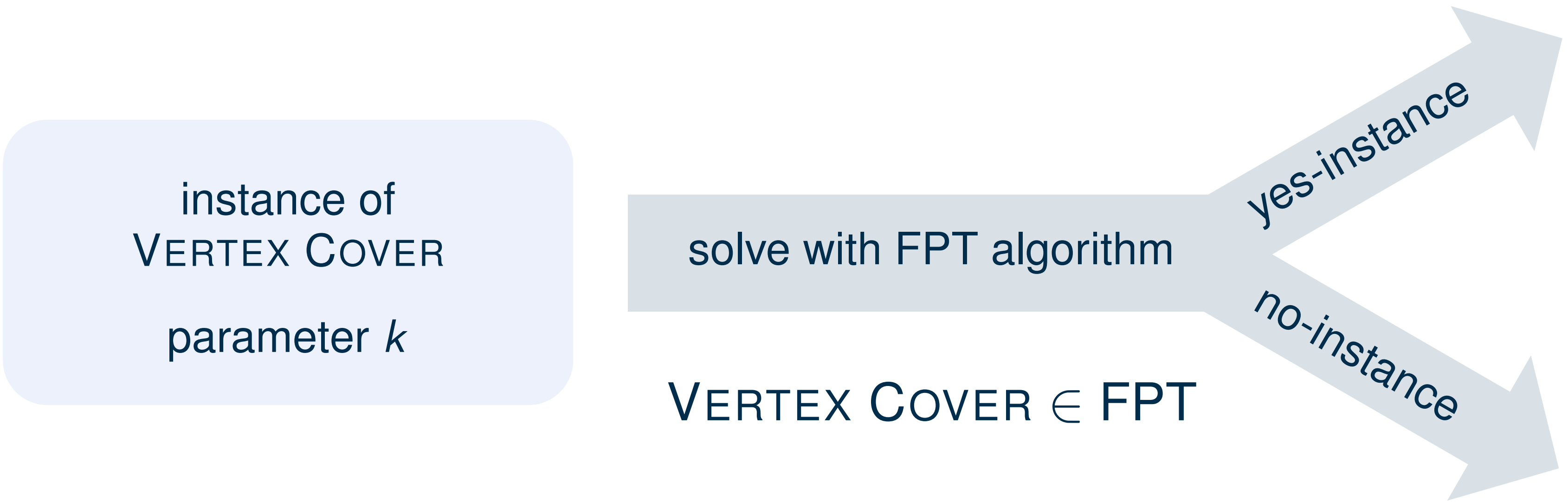
use branching algorithm for AND gates

Vertex Cover → Independent Set

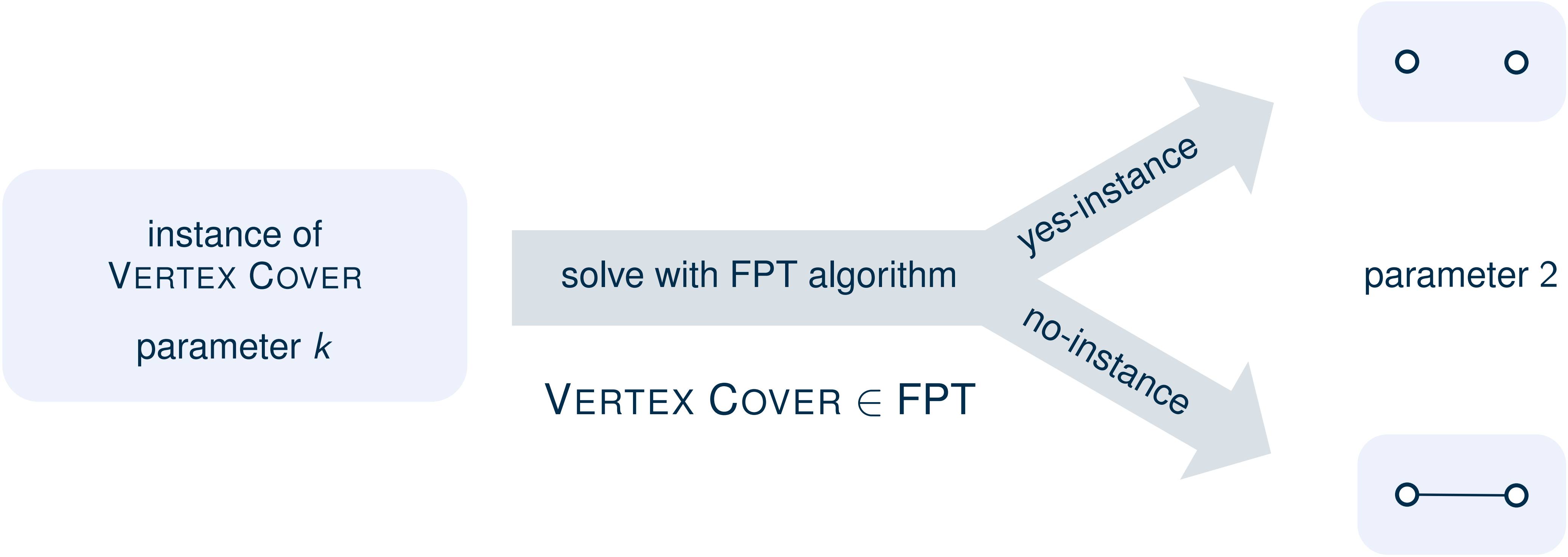
instance of
Vertex Cover

parameter k

VERTEX COVER $\to$ INDEPENDENT SET



VERTEX COVER → INDEPENDENT SET



VERTEX COVER → INDEPENDENT SET

parameterized reduction

instance of
VERTEX COVER

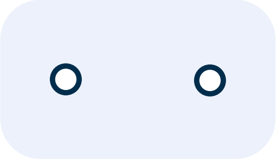
parameter k

solve with FPT algorithm

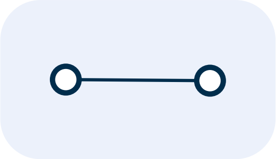
yes-instance

no-instance

VERTEX COVER ∈ FPT



parameter 2



VERTEX COVER → INDEPENDENT SET

instance of
VERTEX COVER

parameter k

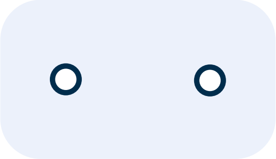
parameterized reduction

solve with FPT algorithm

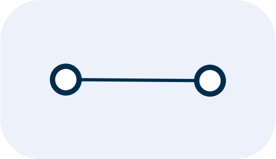
VERTEX COVER $\in$ FPT

yes-instance

no-instance

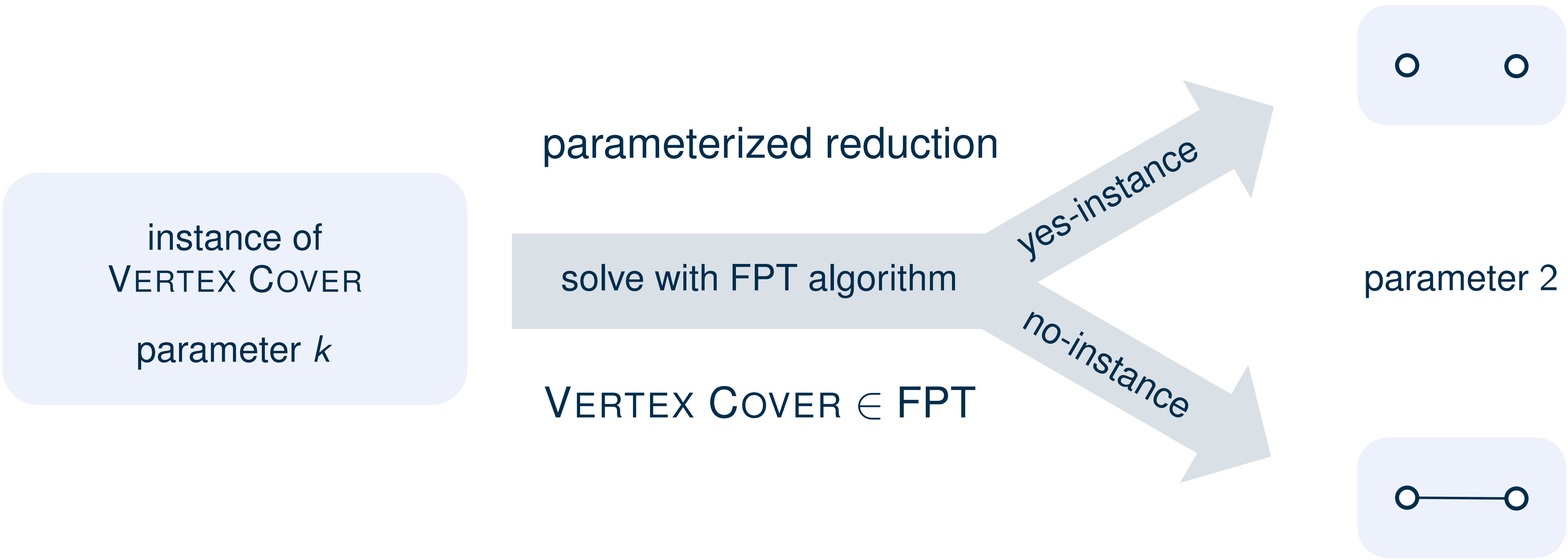


parameter 2



INDPENDENT SET $\in W[1] \implies$ VERTEX COVER $\in W[1]$

Vertex Cover $\rightarrow$ Independent Set

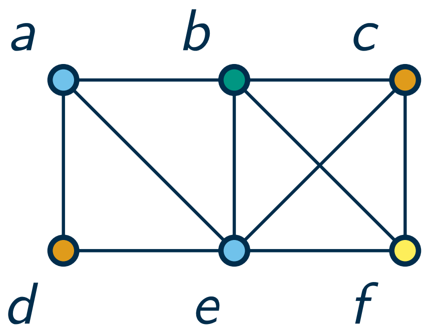


Indpendent Set $\in W[1] \implies$ Vertex Cover $\in W[1]$

repeat analogously for every FPT-problem $\rightarrow$ FPT $\subseteq W[1]$

Multicolored Biclique is $W[1]$-hard

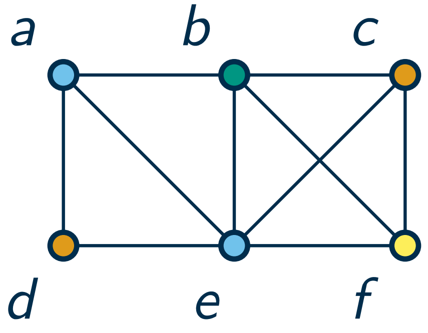
instance of
MULTICOLORED CLIQUE



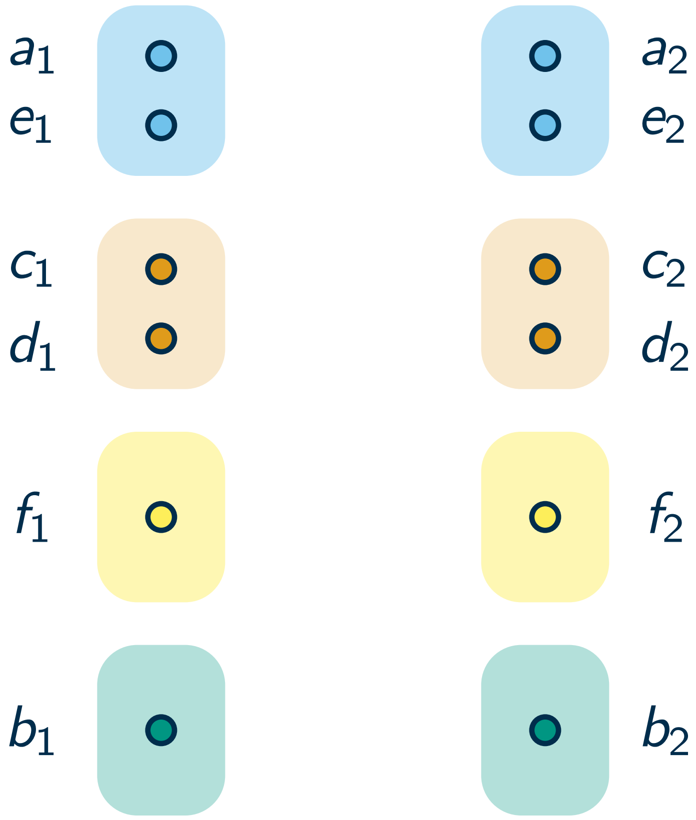
parameter k

Multicolored Biclique is $W[1]$-hard

instance of
MULTICOLORED CLIQUE



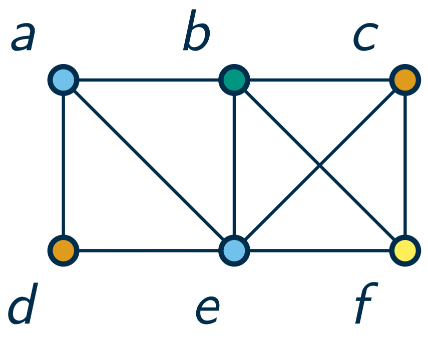
parameter k



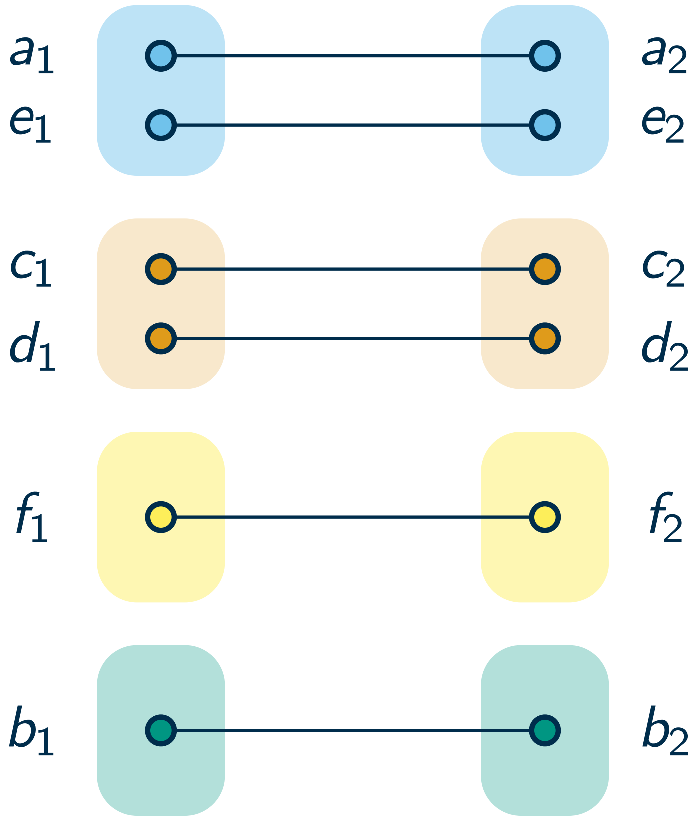
- make two copies for each vertex (left and right)

Multicolored Biclique is $W[1]$-hard

instance of
MULTICOLORED CLIQUE

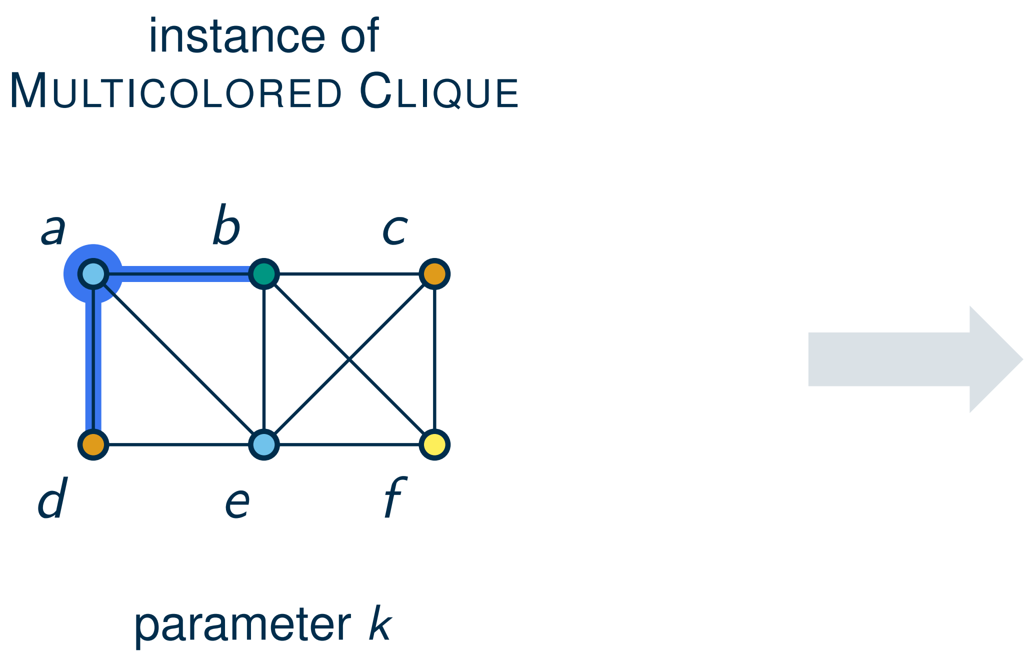


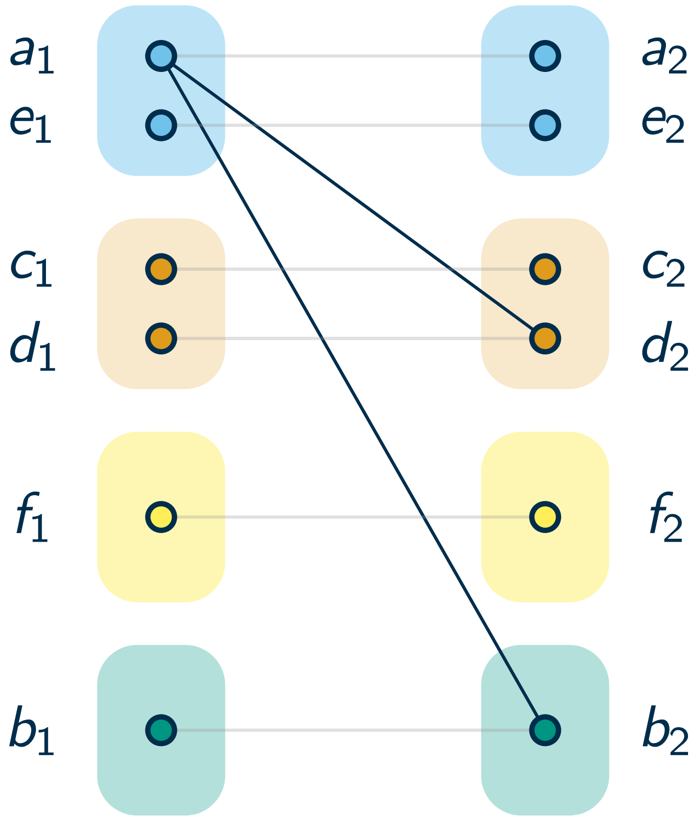
parameter k



- make two copies for each vertex (left and right)
- connect copies that correspond to same input vertex

Multicolored Biclique is $W[1]$-hard

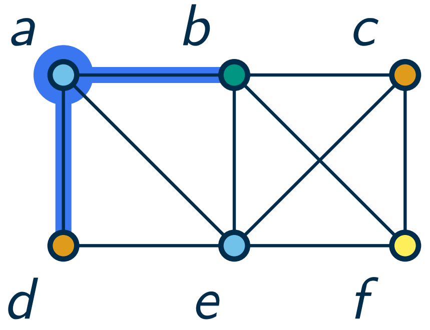




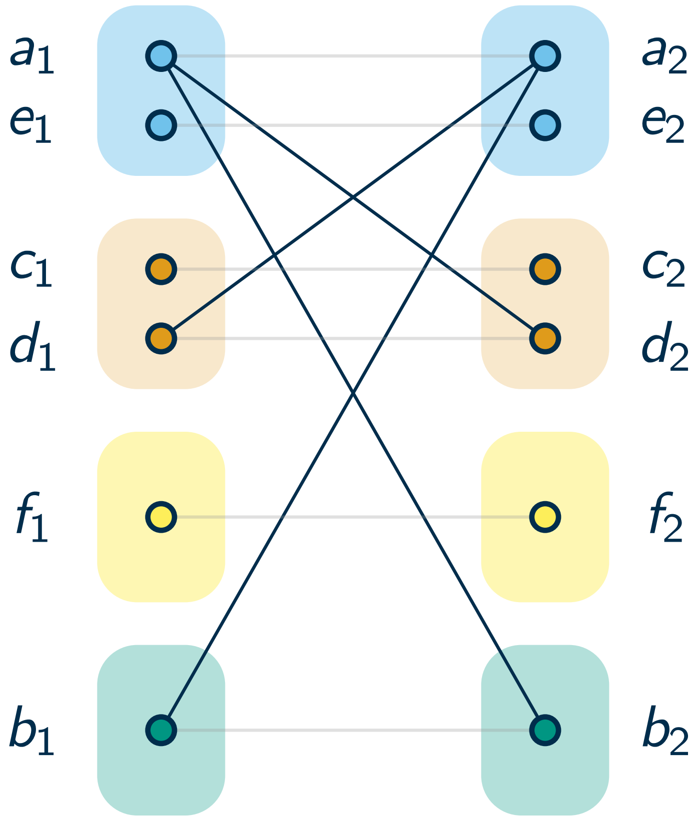
- make two copies for each vertex (left and right)
- connect copies that correspond to same input vertex
- add edge between v_i and w_j if v and w adjacent in input graph and $i \neq j$, unless same color

Multicolored Biclique is $W[1]$-hard

instance of
Multicolored Clique



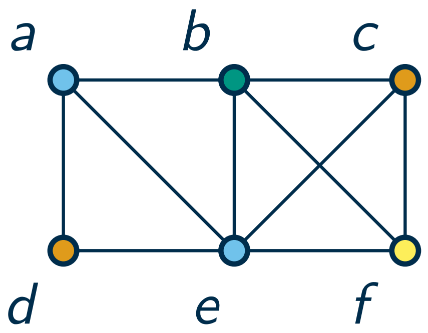
parameter k



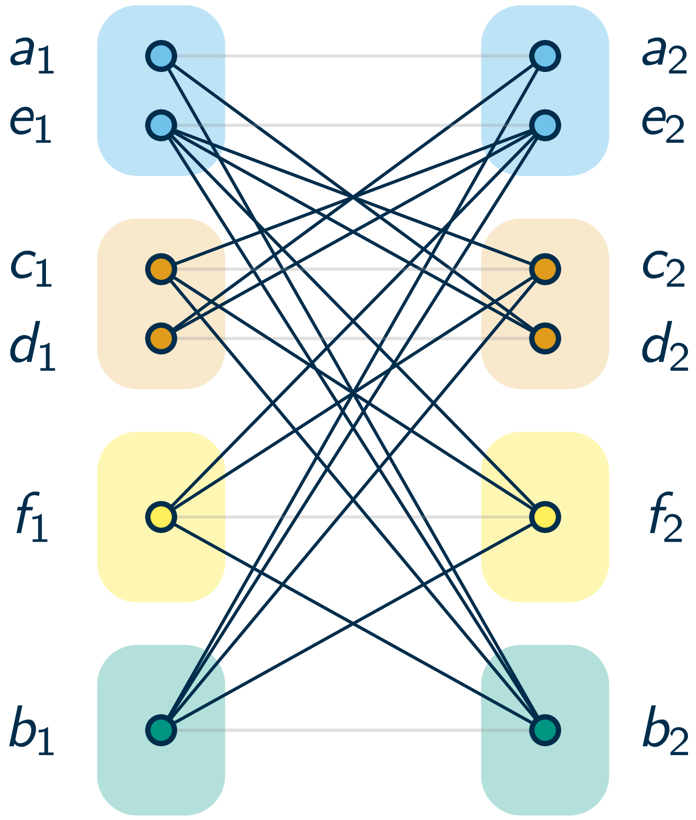
- make two copies for each vertex (left and right)
- connect copies that correspond to same input vertex
- add edge between v_i and w_j if v and w adjacent in input graph and $i \neq j$, unless same color

Multicolored Biclique is $W[1]$-hard

instance of
MULTICOLORED CLIQUE



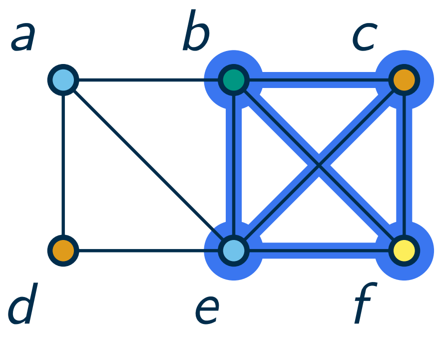
parameter k



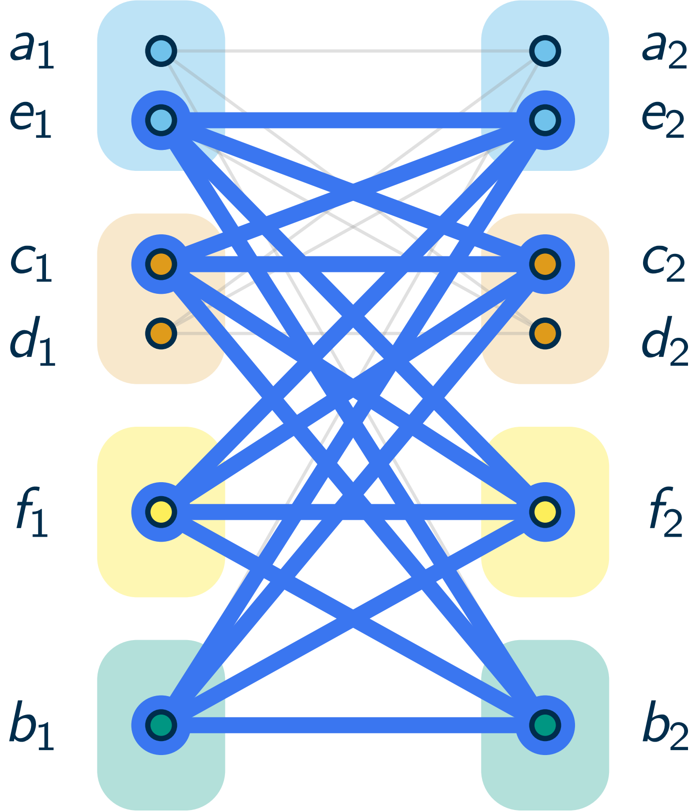
- make two copies for each vertex (left and right)
- connect copies that correspond to same input vertex
- add edge between v_i and w_j if v and w adjacent in input graph and $i \neq j$, unless same color

Multicolored Biclique is $W[1]$-hard

instance of
MULTICOLORED CLIQUE



parameter k

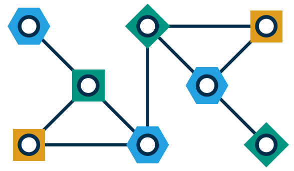


- make two copies for each vertex (left and right)
- connect copies that correspond to same input vertex
- add edge between v_i and w_j if v and w adjacent in input graph and $i \neq j$, unless same color

LONG INDUCED PATH IS $W[1]$-HARD

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

MULTICOLORED INDEPENDENT SET

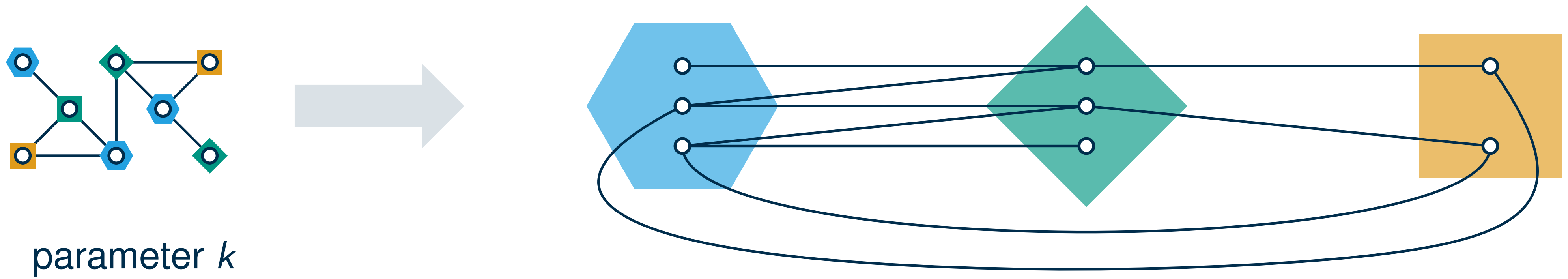


parameter k

Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



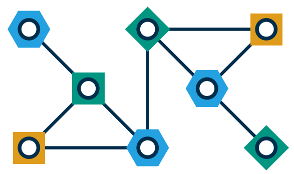
parameter k

- copy over all vertices and edges

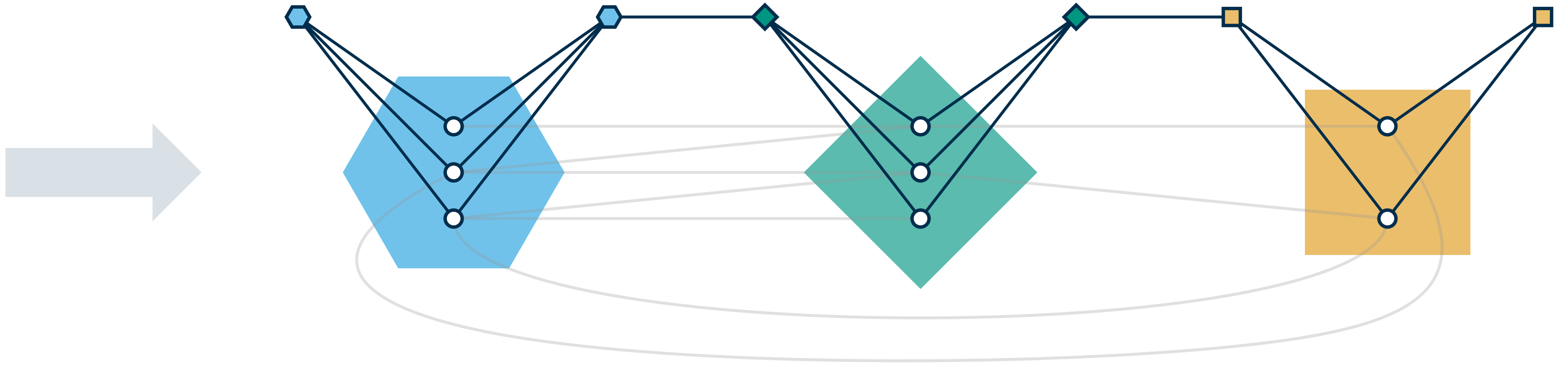
Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



parameter k

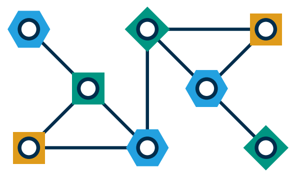


- copy over all vertices and edges
- add two universal vertices for each color class ("entry and exit"), connect exit to next entry

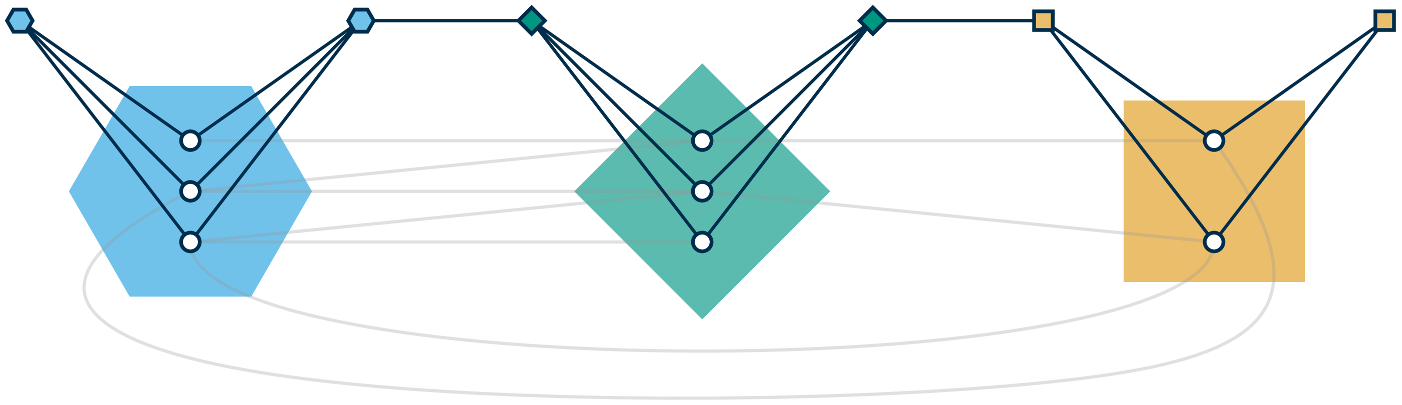
Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



parameter k



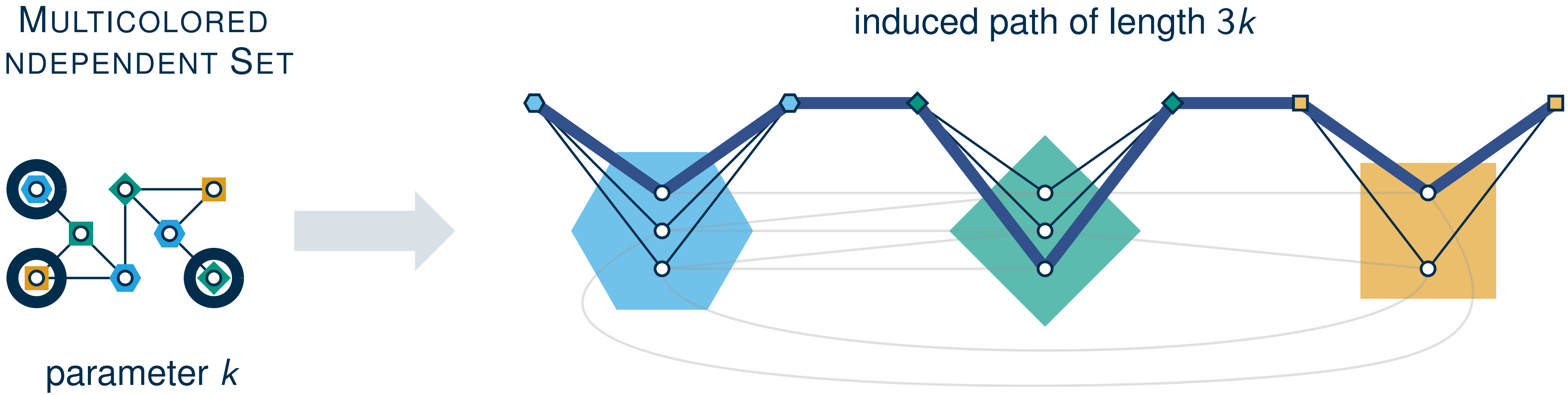
parameter $k' = 3k$

- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry

Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



parameter k

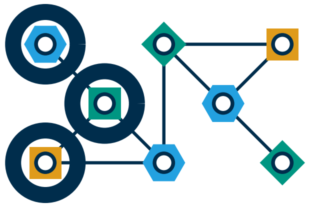
parameter $k' = 3k$

- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry

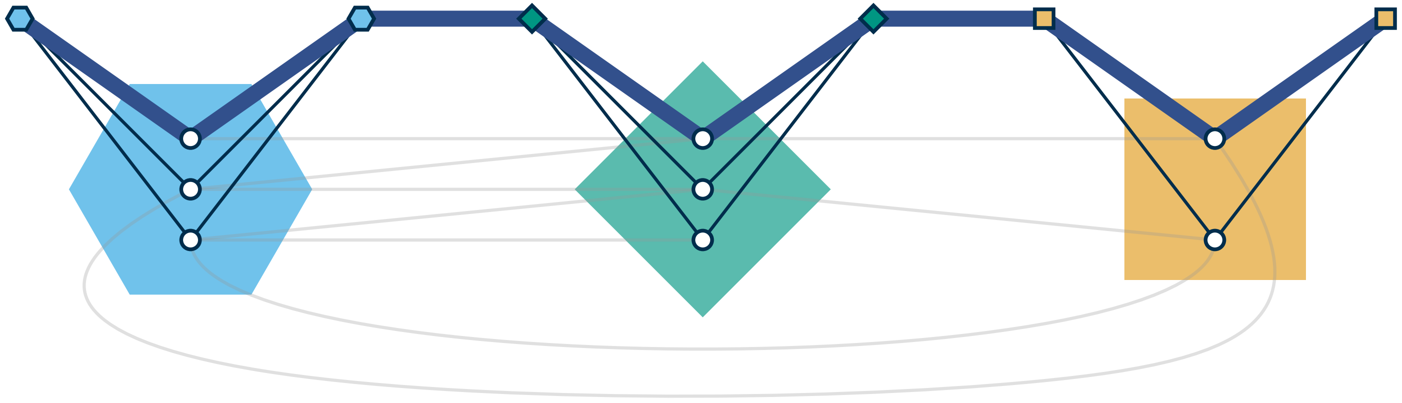
Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



parameter k



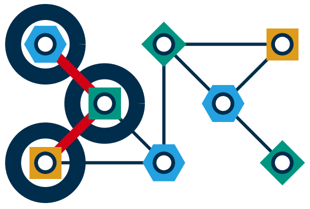
parameter $k' = 3k$

- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry

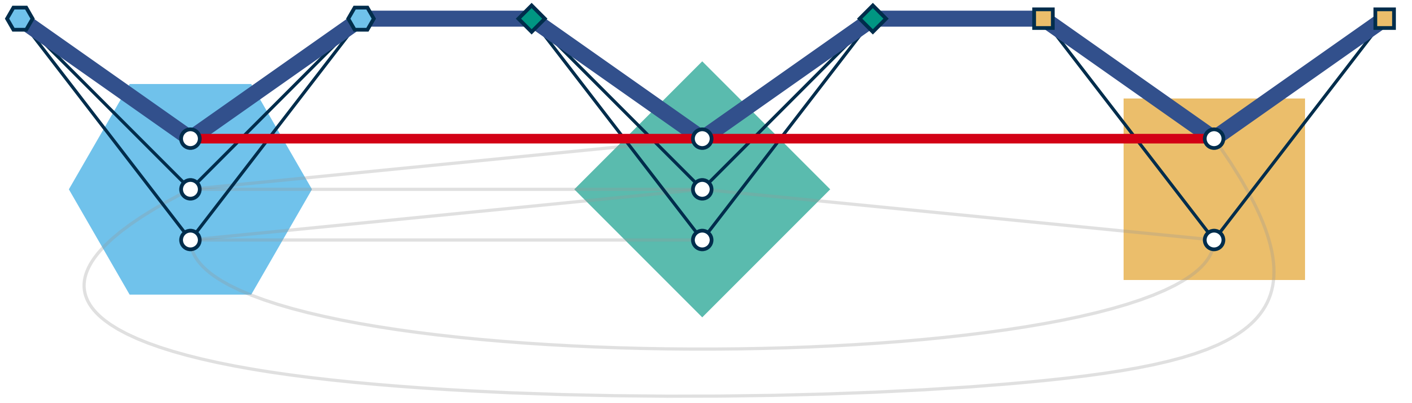
Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



parameter k



parameter $k' = 3k$

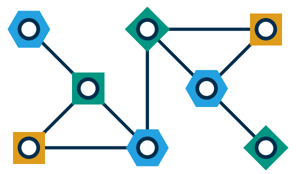
- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry

LONG INDUCED PATH IS $W[1]$-HARD

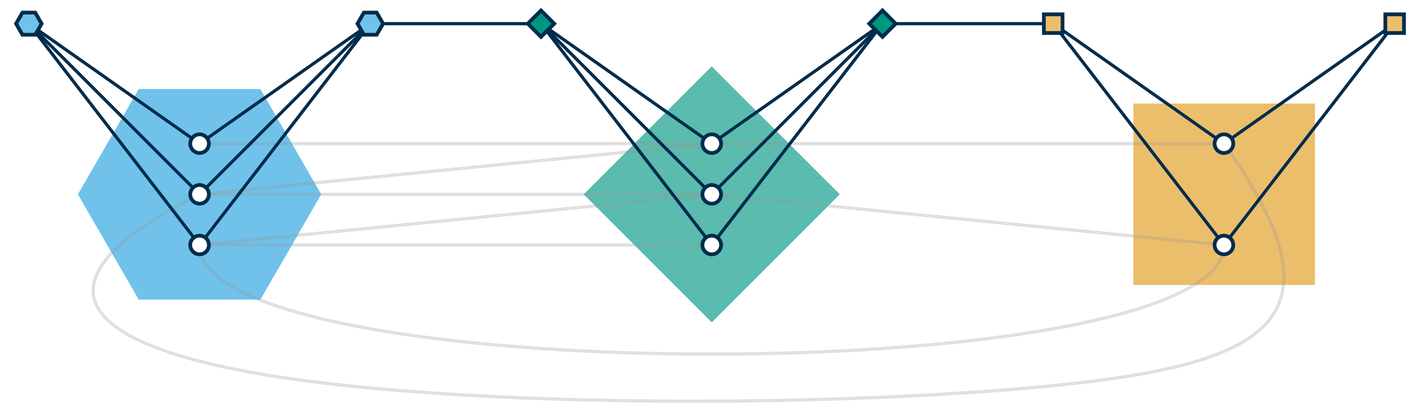
LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

correct?

MULTICOLORED INDEPENDENT SET



parameter k



parameter $k' = 3k$

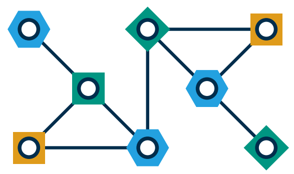
- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry

LONG INDUCED PATH IS $W[1]$-HARD

LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

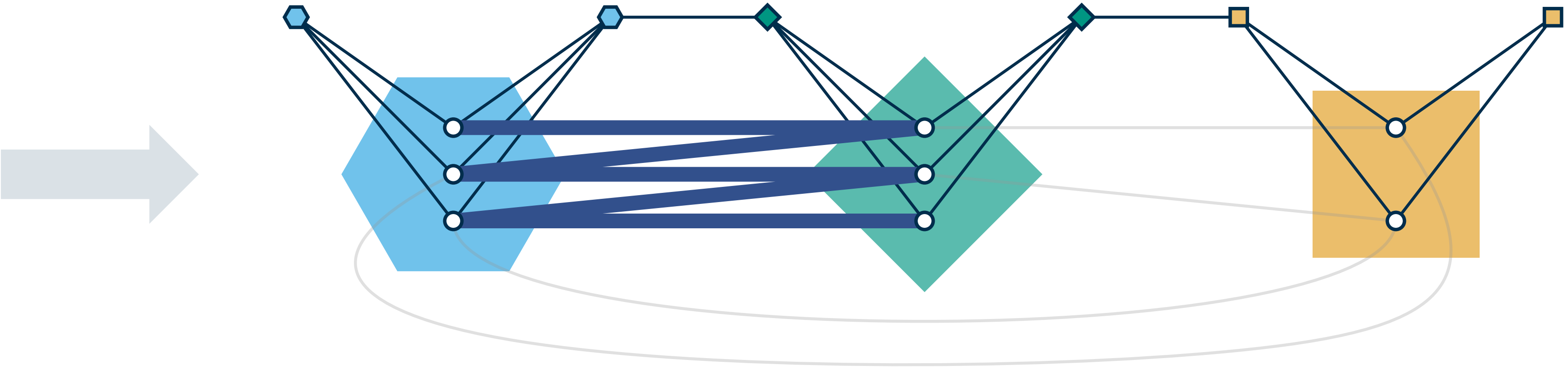
correct?

MULTICOLORED INDEPENDENT SET



parameter k

there could be long induced path that doesn't use all colors



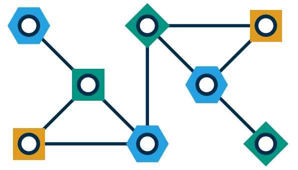
parameter $k' = 3k$

- copy over all vertices and edges
- add two universal vertices for each color class ("entry and exit"), connect exit to next entry

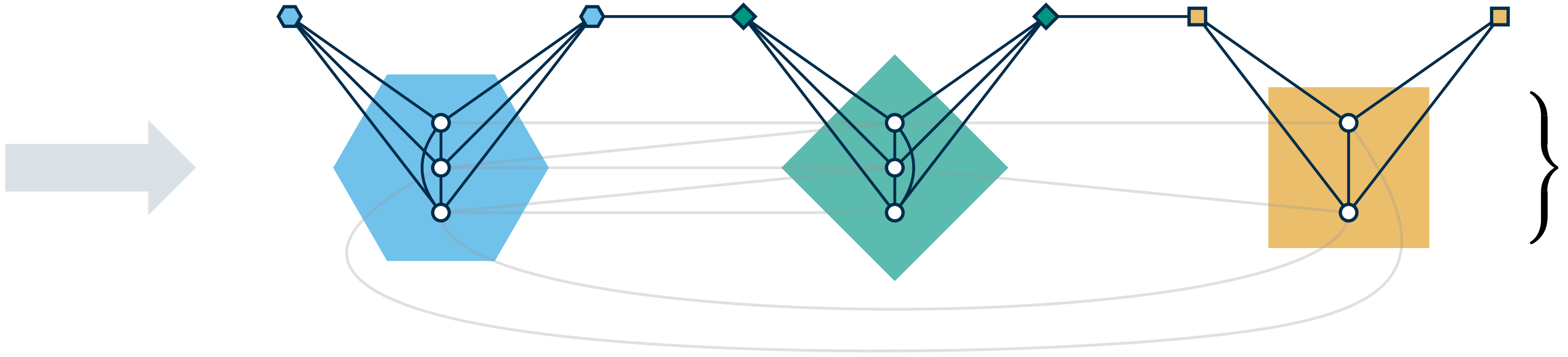
Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

Multicolored Independent Set



parameter k



parameter $k' = 3k$

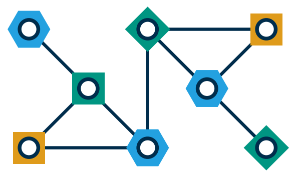
- copy over all vertices and edges
- add two universal vertices for each color class ("entry and exit"), connect exit to next entry
- make color classes form cliques

LONG INDUCED PATH IS $W[1]$-HARD

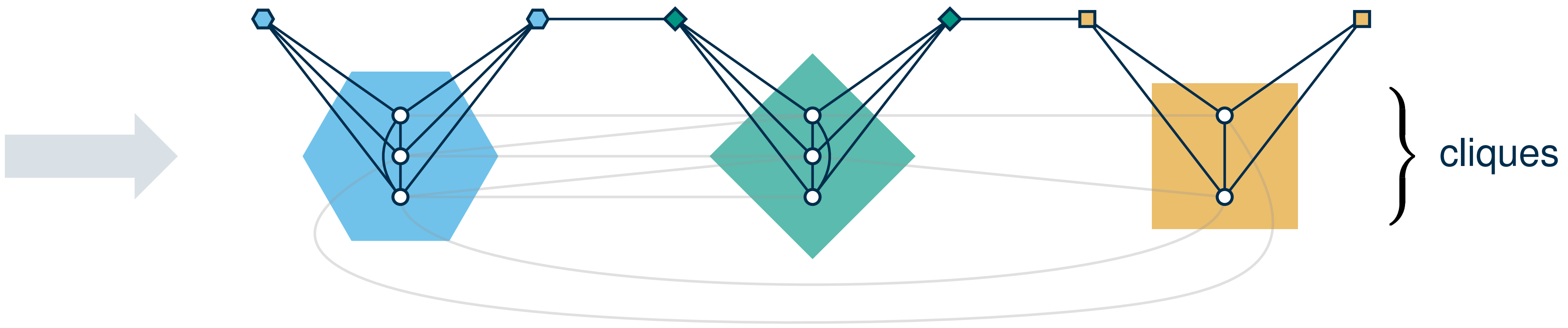
LONG INDUCED PATH Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

correct?

MULTICOLORED INDEPENDENT SET



parameter k



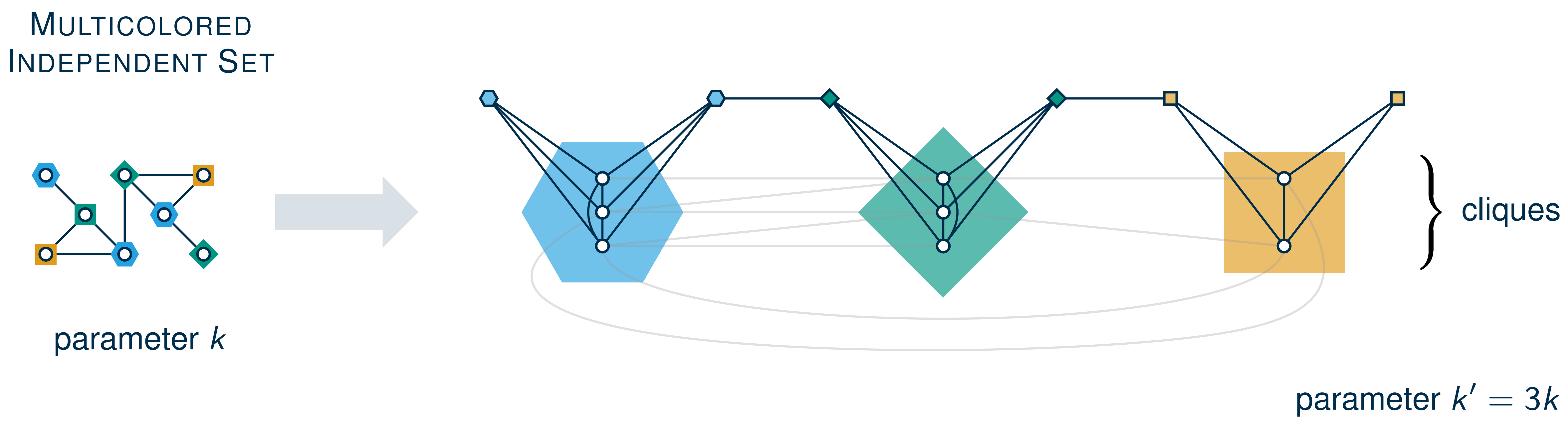
parameter $k' = 3k$

- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry
- make color classes form cliques

Long Induced Path is $W[1]$-hard

Long Induced Path Given a graph $G = (V, E)$ and a parameter k, does G contain an induced path of length k?

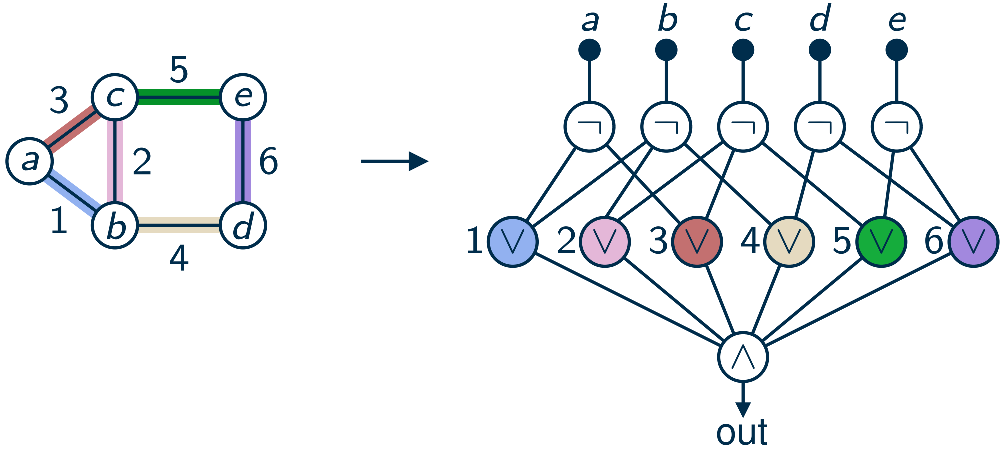
correct? ✓



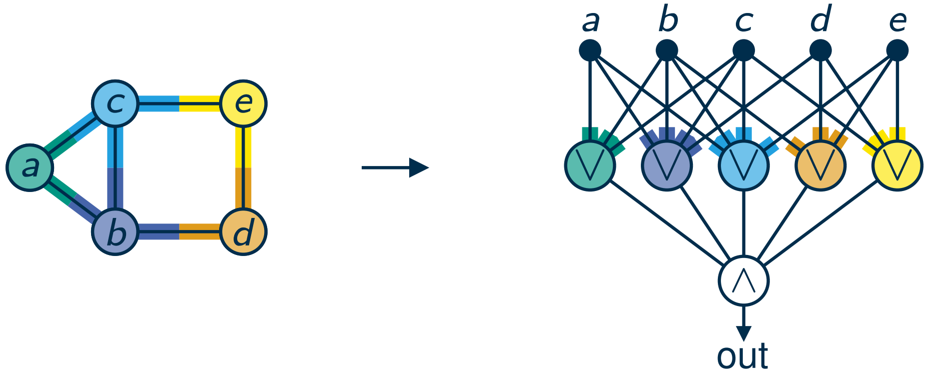
- copy over all vertices and edges
- add two universal vertices for each color class (“entry and exit”), connect exit to next entry
- make color classes form cliques

INDEPENDENT DOMINATING SET $\in W[2]$

INDEPENDENT SET → WCS



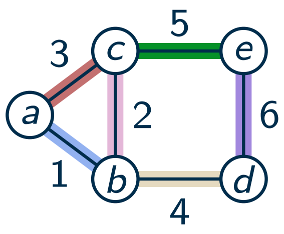
DOMINATING SET → WCS



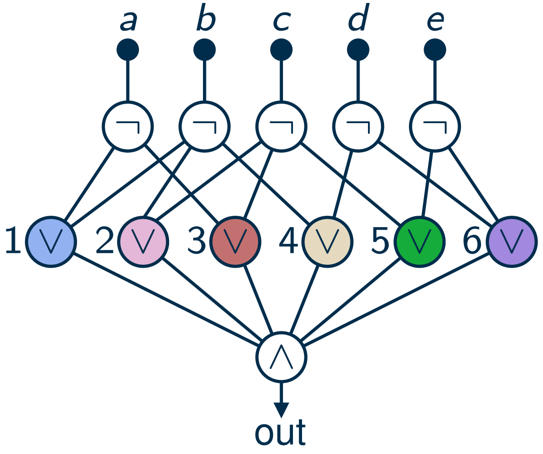
INDEPENDENT DOMINATING SET → WCS

Independent Dominating Set $\in W[2]$

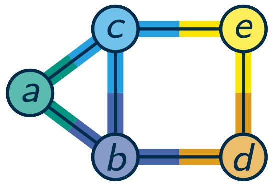
Independent Set → WCS



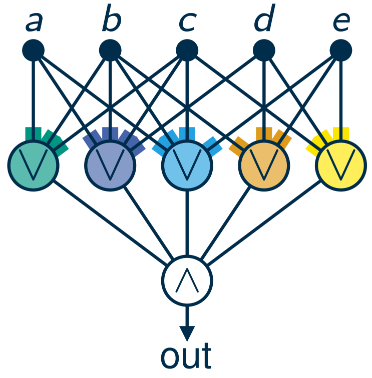
→



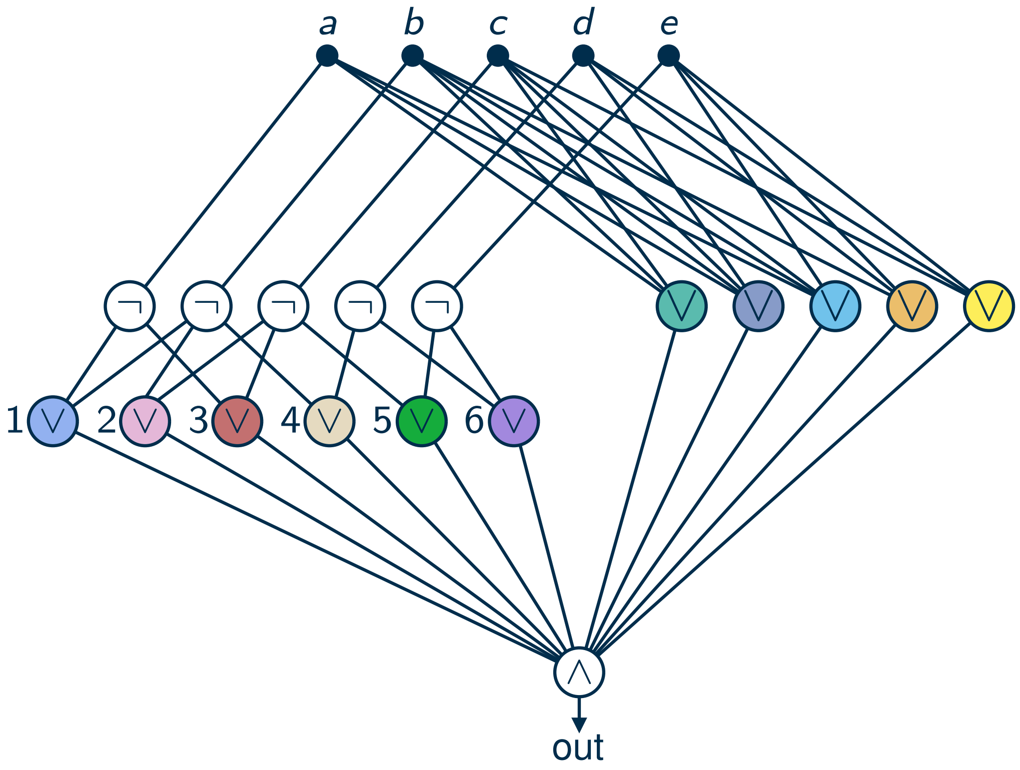
Dominating Set → WCS



→

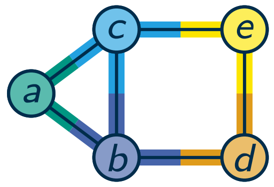


Independent Dominating Set → WCS



INDEPENDENT DOMINATING SET is $W[2]$-hard

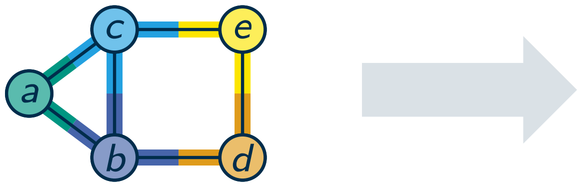
instance of
DOMINATING SET



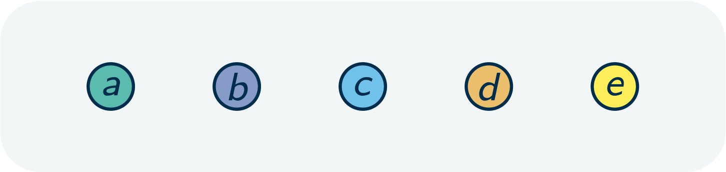
parameter k

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET

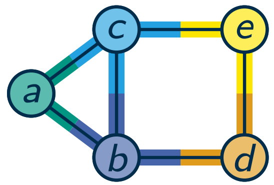


parameter k



Independent Dominating Set is $W[2]$-hard

instance of
Dominating Set



parameter k

possible choices
for vertex 1

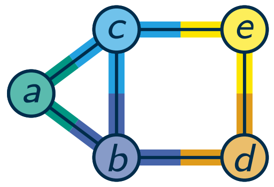
$\dots$

possible choices
for vertex k

parameter $k' = k$

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k

selection gadget

selection gadget

possible choices
for vertex 1

a b c d e

. . .

possible choices
for vertex k

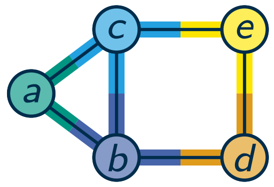
a b c d e

a b c d e

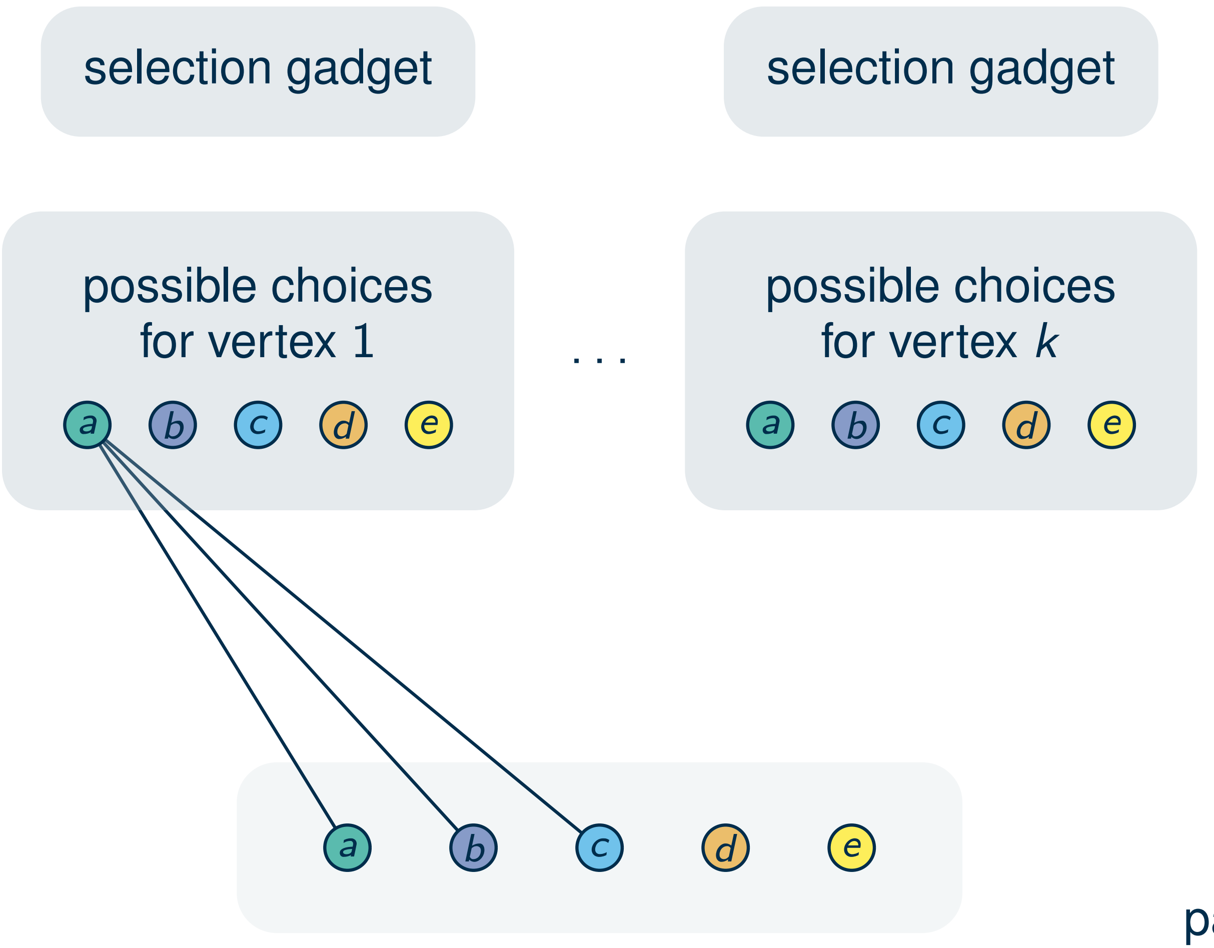
parameter $k' = k$

Independent Dominating Set is $W[2]$-hard

instance of
Dominating Set



parameter k

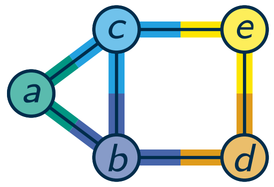


edge if
adjacent or
same vertex

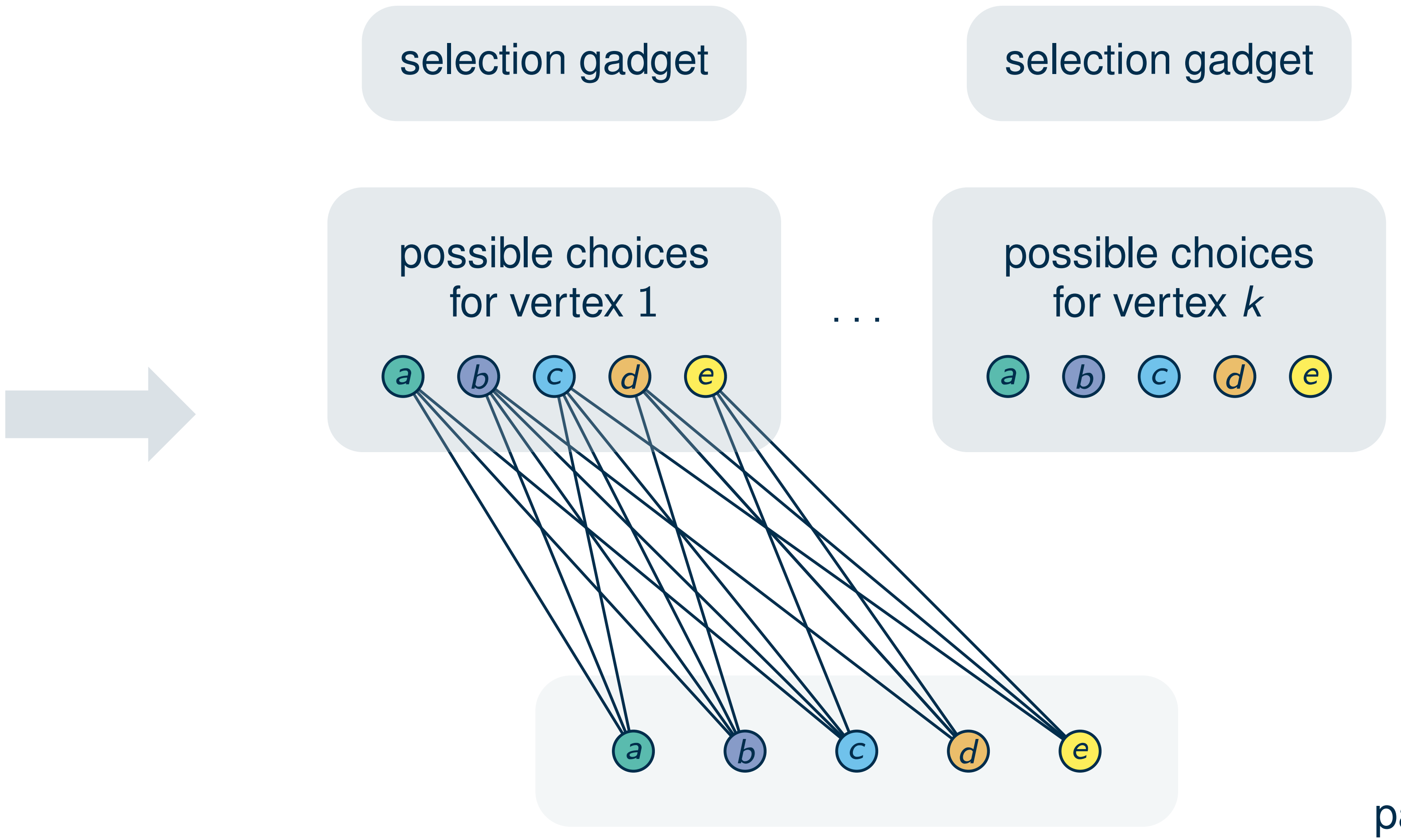
parameter $k' = k$

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k

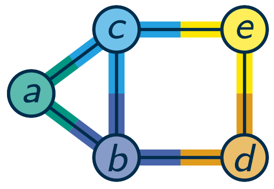


edge if adjacent or same vertex

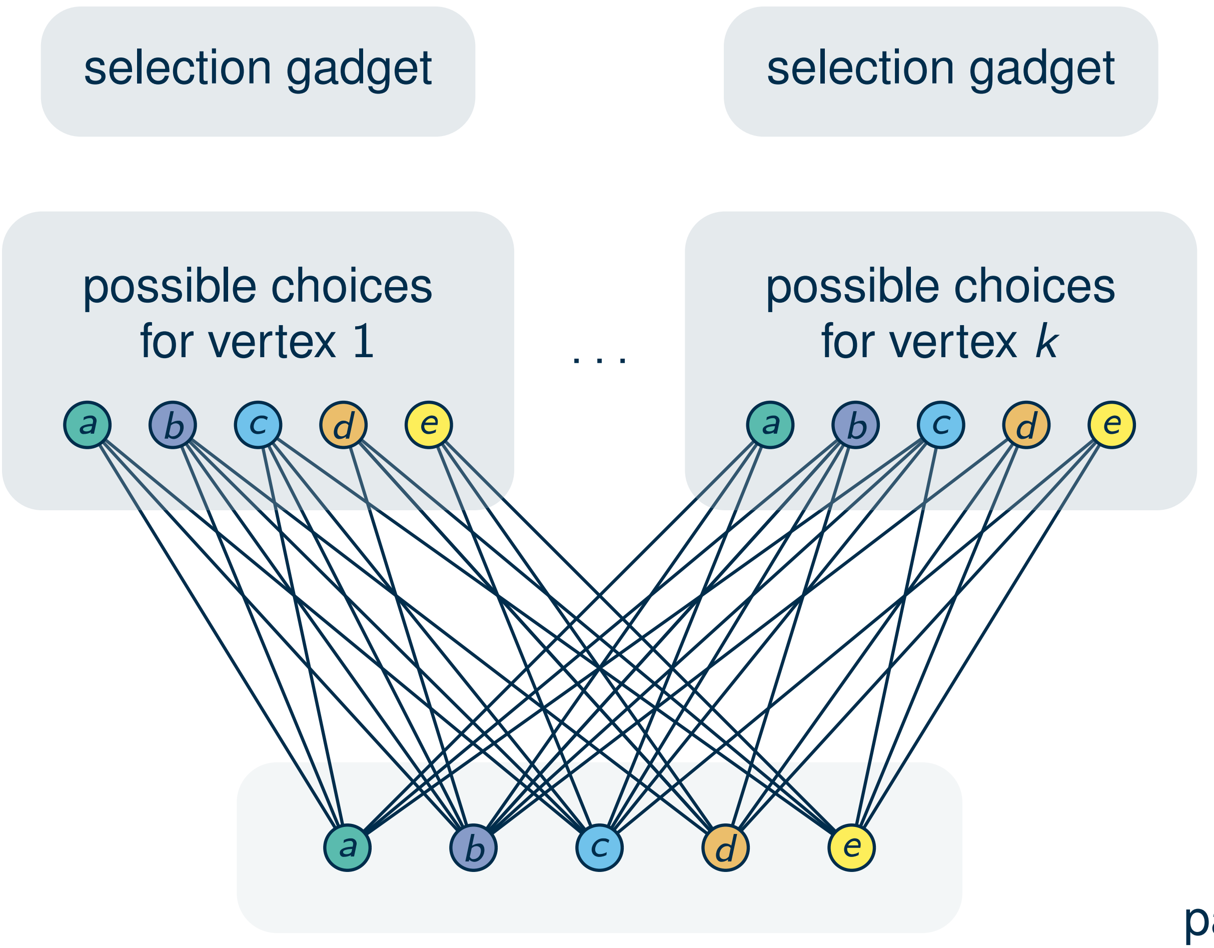
parameter $k' = k$

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k

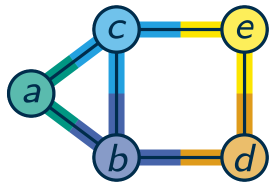


edge if
adjacent or
same vertex

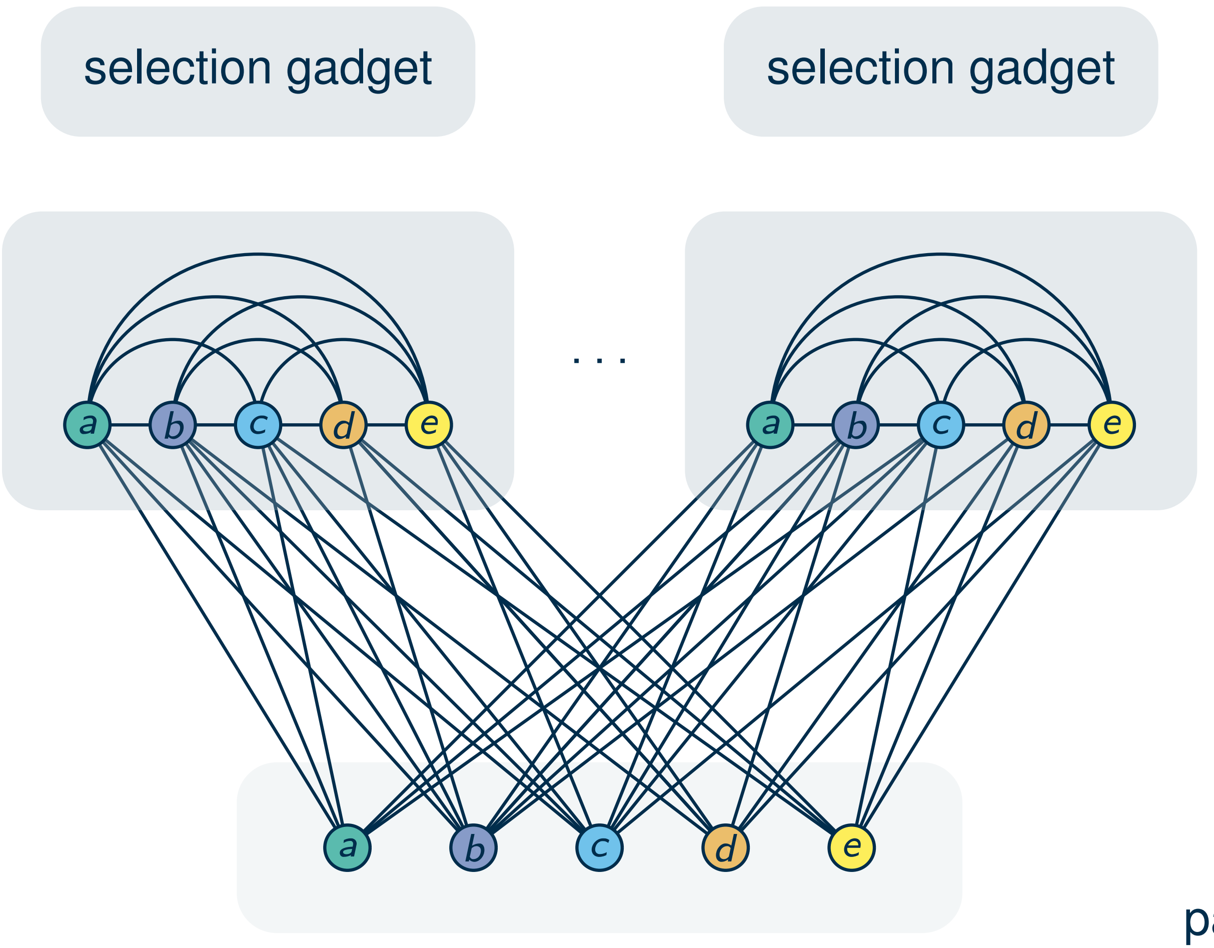
parameter $k' = k$

Independent Dominating Set is $W[2]$-hard

instance of
Dominating Set



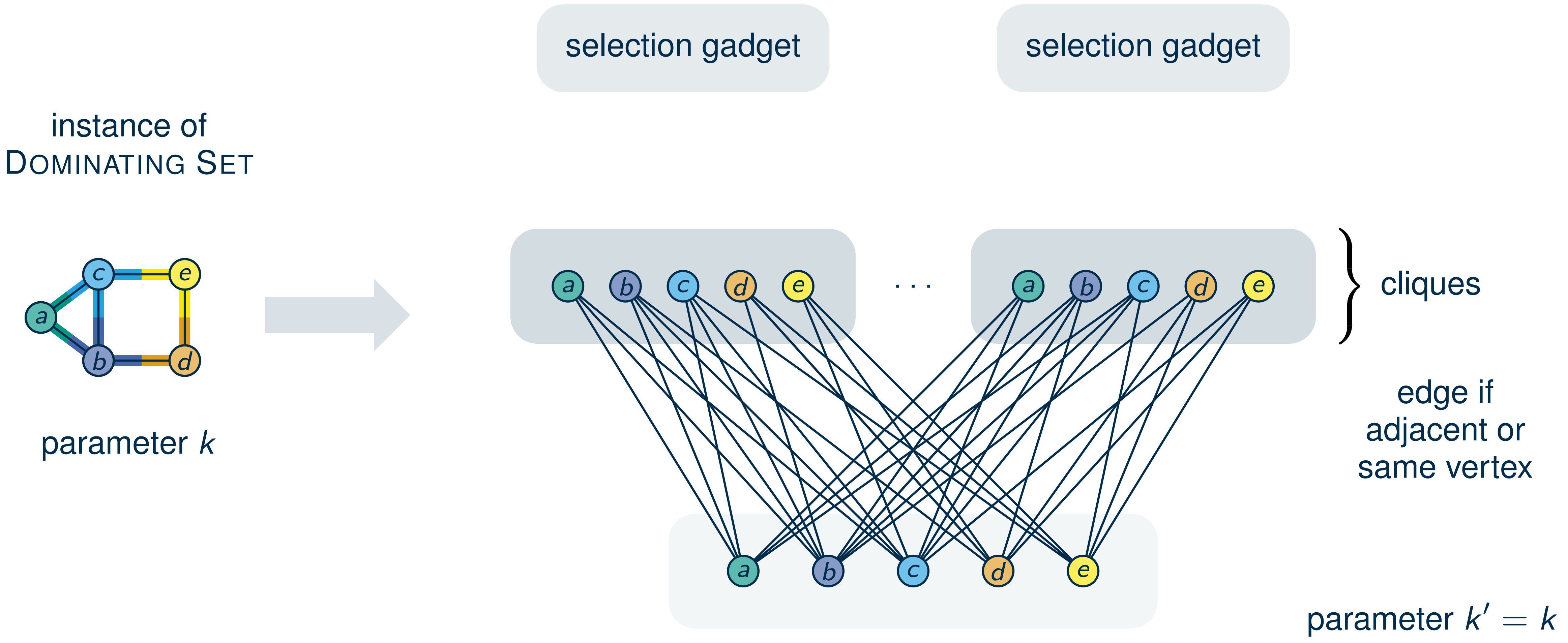
parameter k



edge if
adjacent or
same vertex

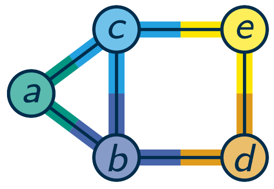
parameter $k' = k$

Independent Dominating Set is $W[2]$-hard

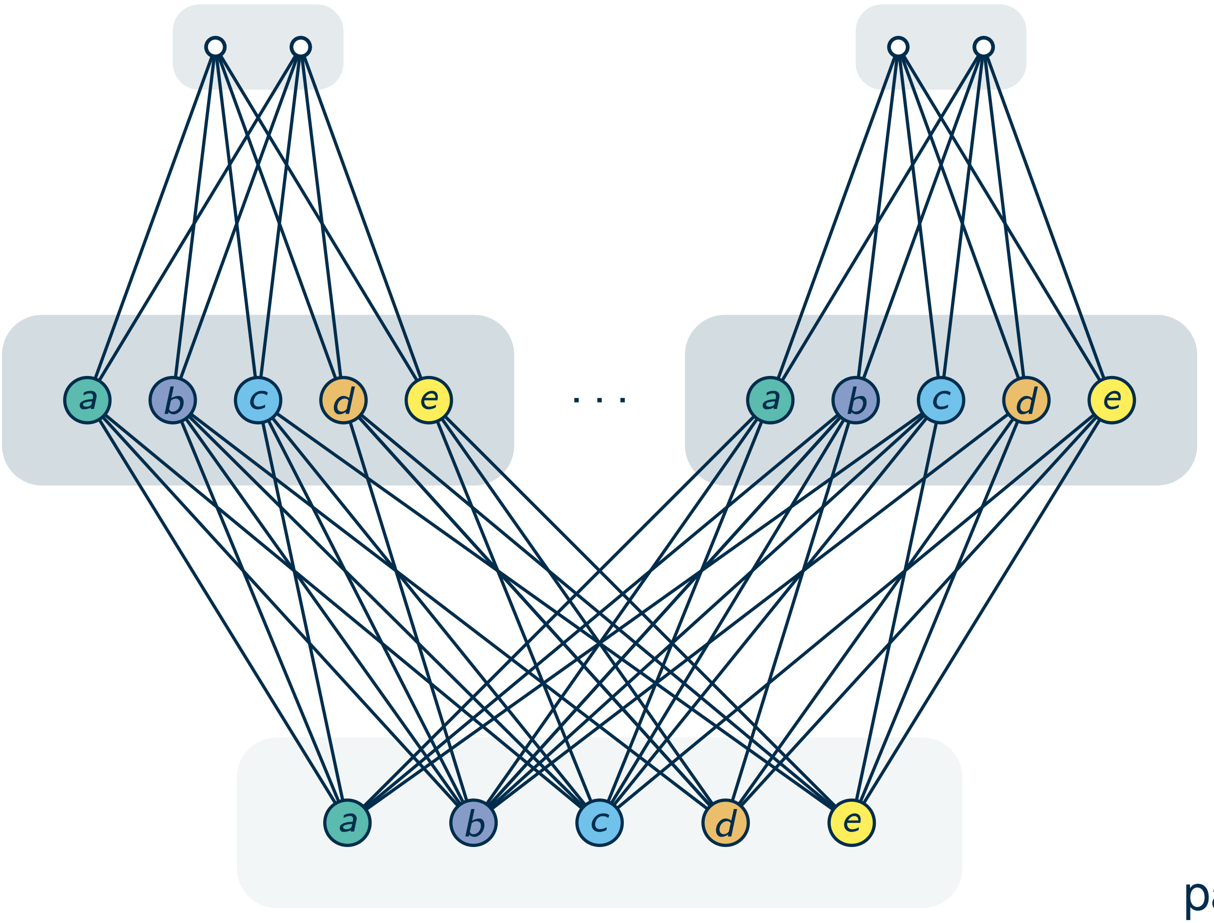


INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k



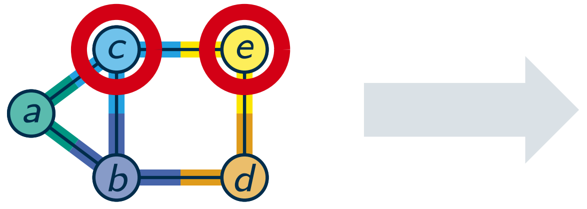
cliques

edge if
adjacent or
same vertex

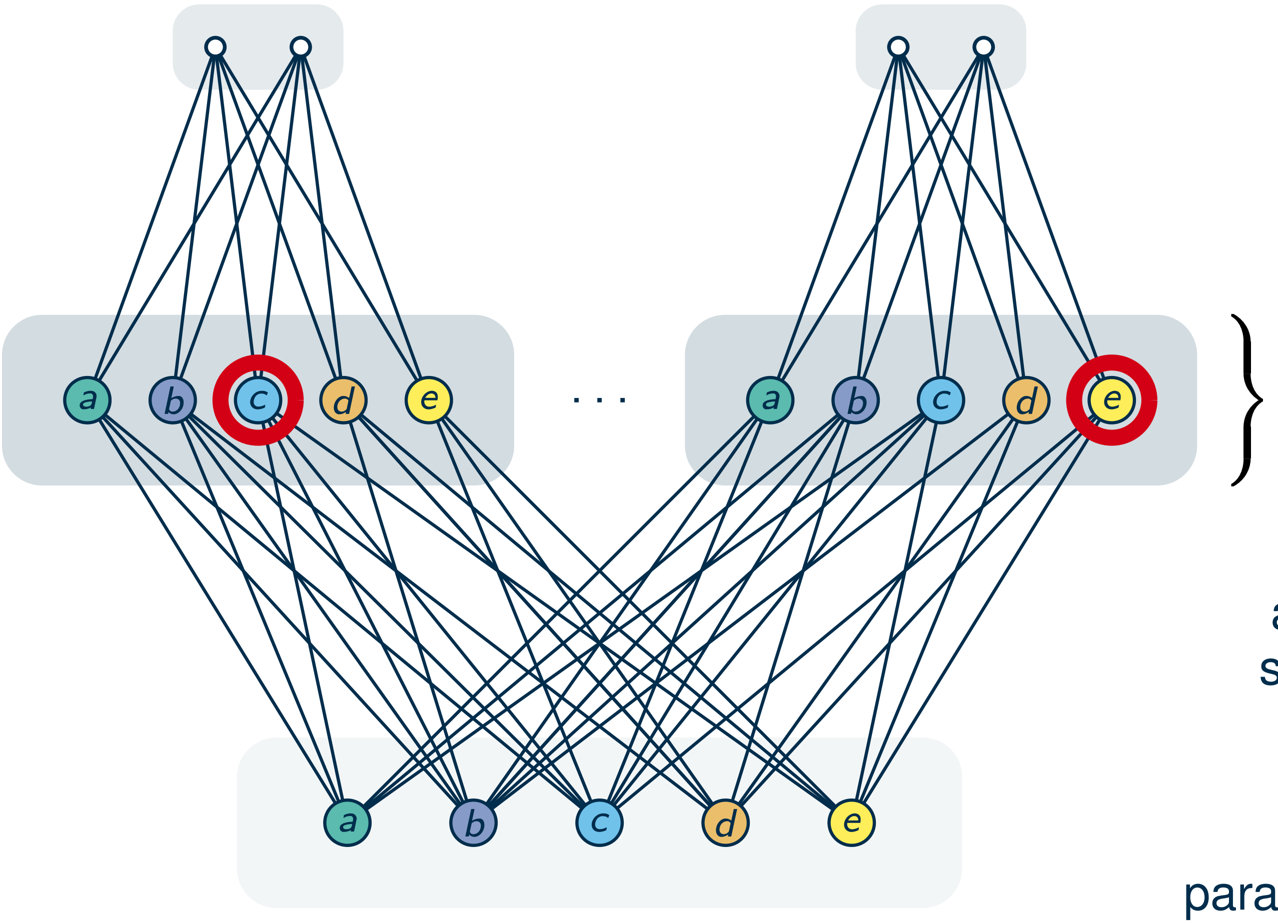
parameter $k' = k$

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k



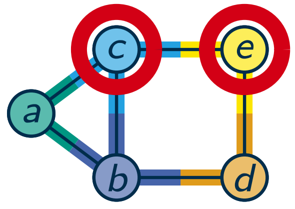
cliques

edge if
adjacent or
same vertex

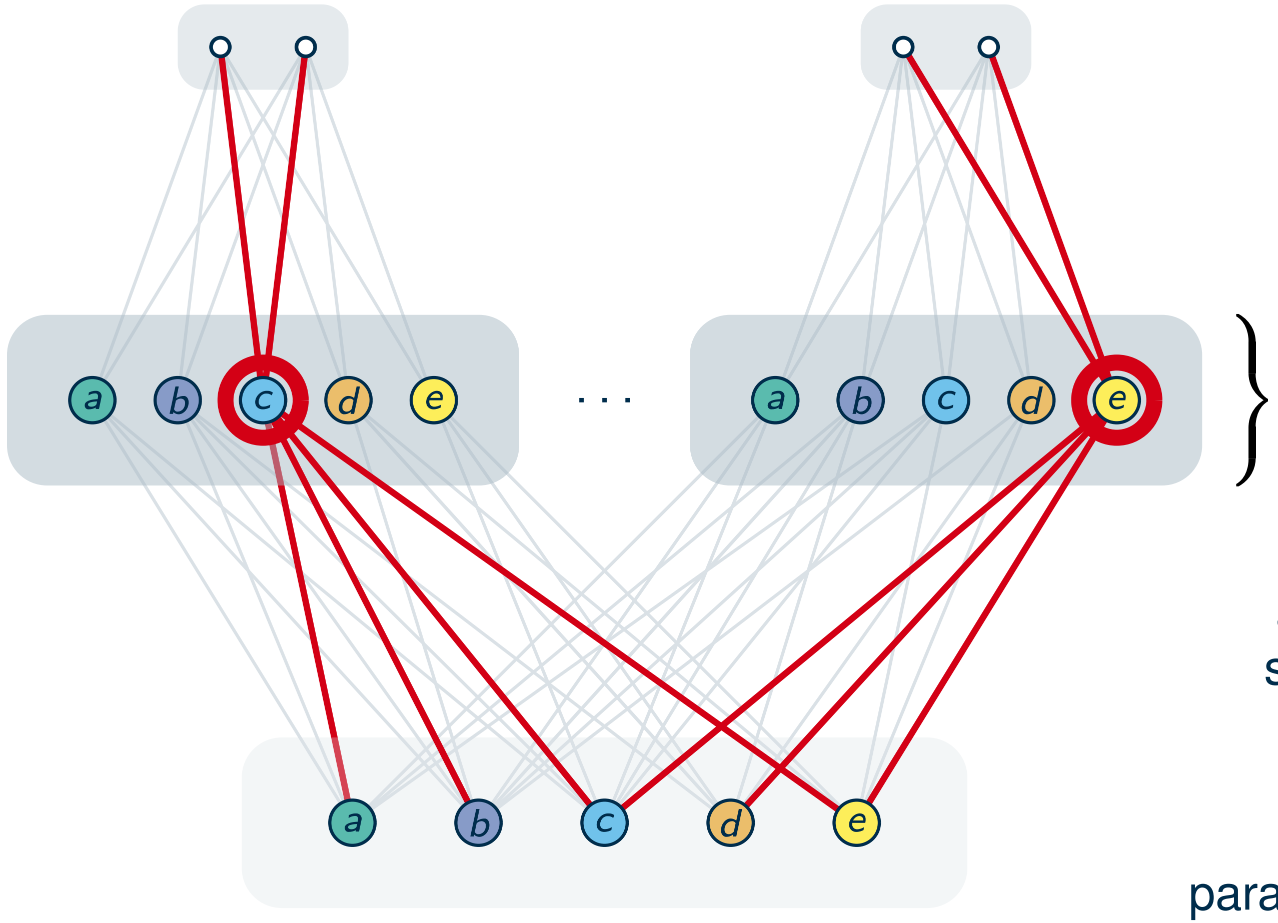
parameter $k' = k$

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k



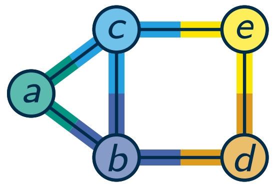
cliques

edge if
adjacent or
same vertex

parameter $k' = k$

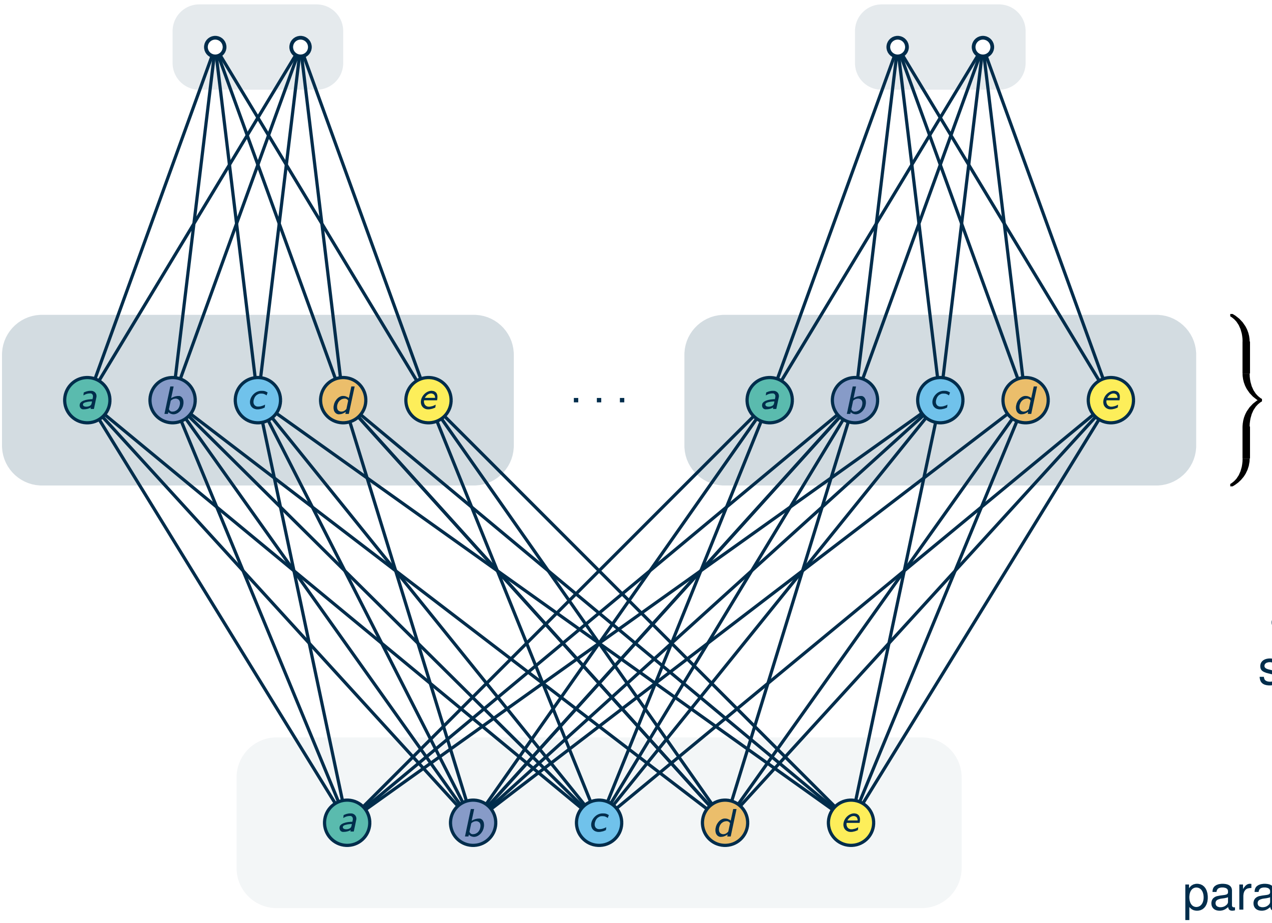
INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k

correct?



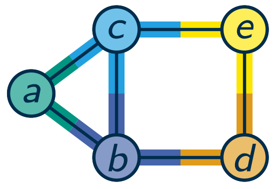
cliques

edge if
adjacent or
same vertex

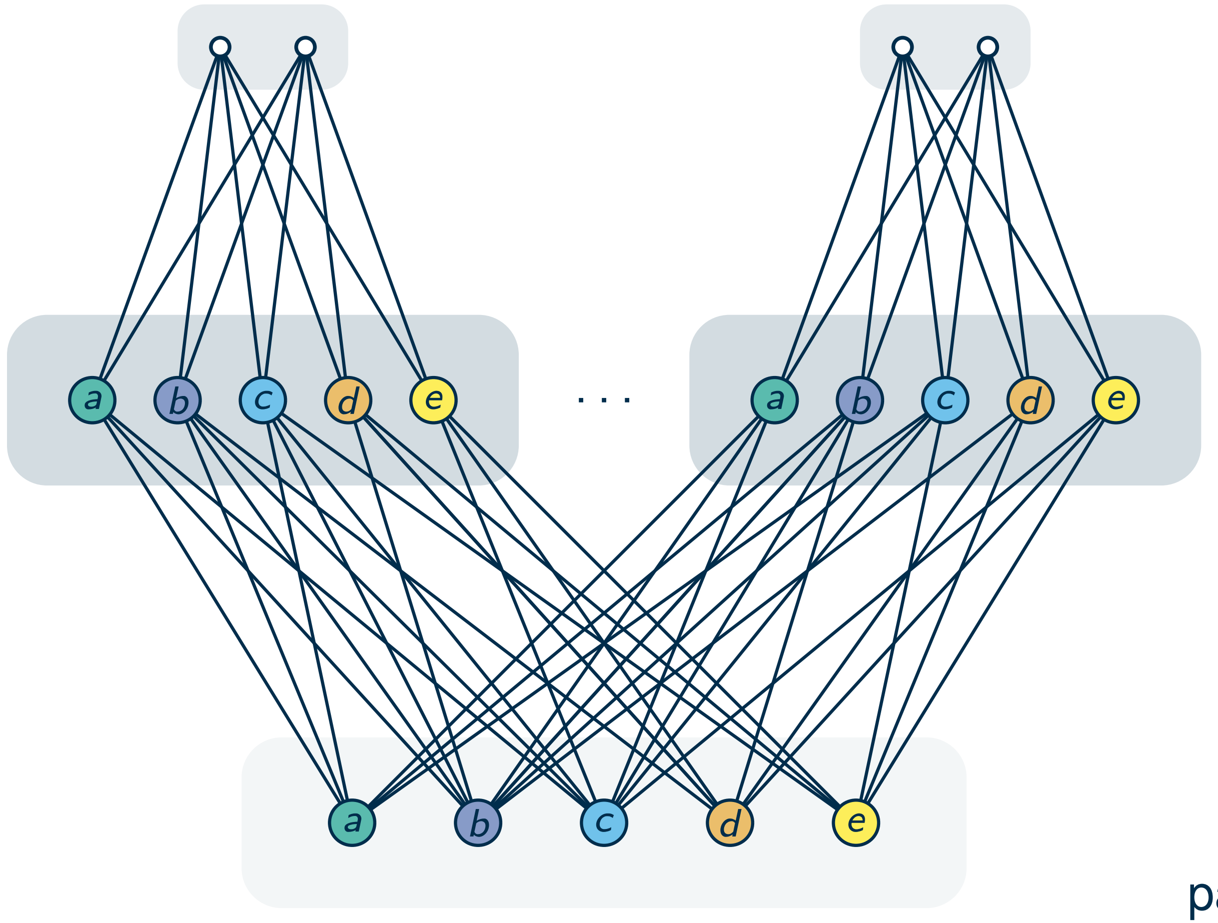
parameter $k' = k$

INDEPENDENT DOMINATING SET is $W[2]$-hard

instance of
DOMINATING SET



parameter k



cliques

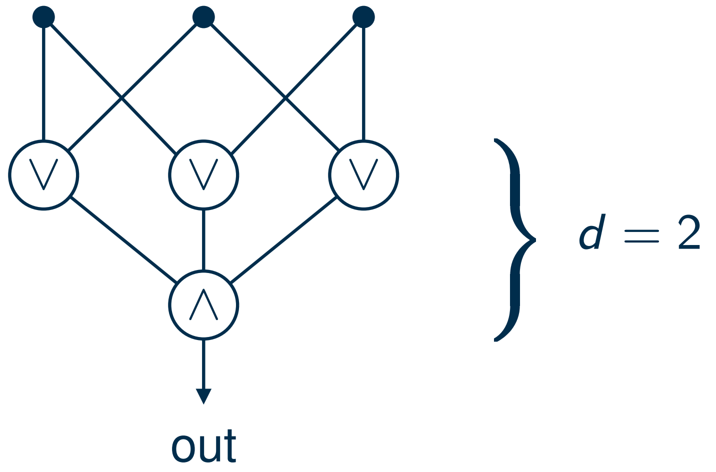
edge if
adjacent or
same vertex

parameter $k' = k$

correct? ✓ (if $n \geq k$ in input graph)

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

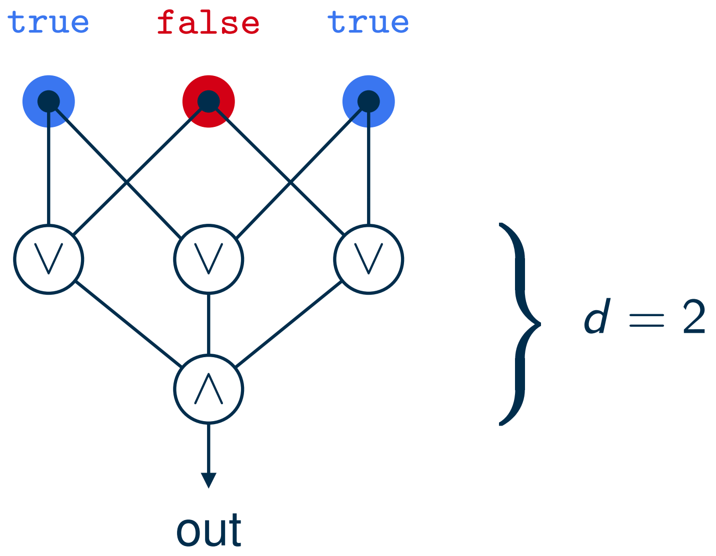


weft (of circuit): largest number of nodes with in-degree > 2 on a directed path

weight (of assignment): number of 1s for input nodes

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

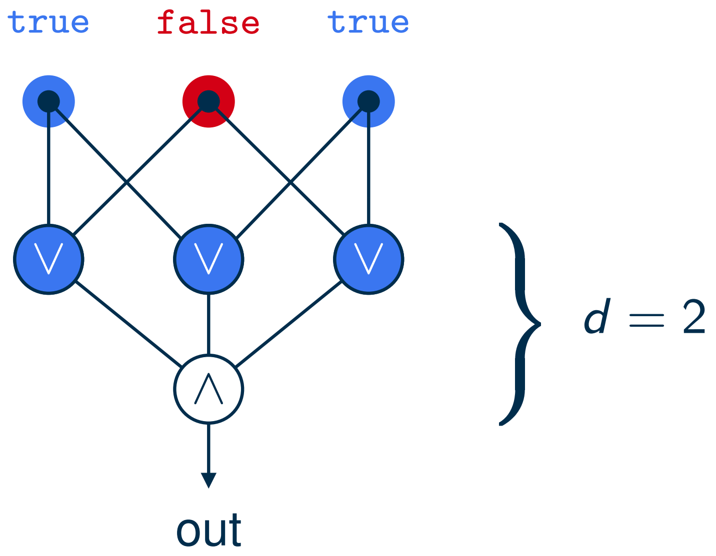


weft (of circuit): largest number of nodes with in-degree > 2 on a directed path

weight (of assignment): number of 1s for input nodes

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

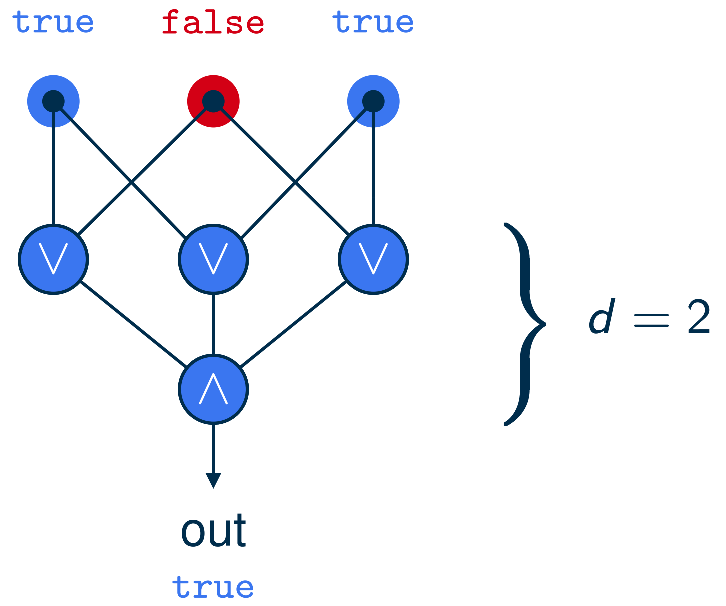


weft (of circuit): largest number of nodes with in-degree > 2 on a directed path

weight (of assignment): number of 1s for input nodes

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?



weft (of circuit): largest number of nodes with in-degree > 2 on a directed path

weight (of assignment): number of 1s for input nodes

Monotone Weft-1 Circuits

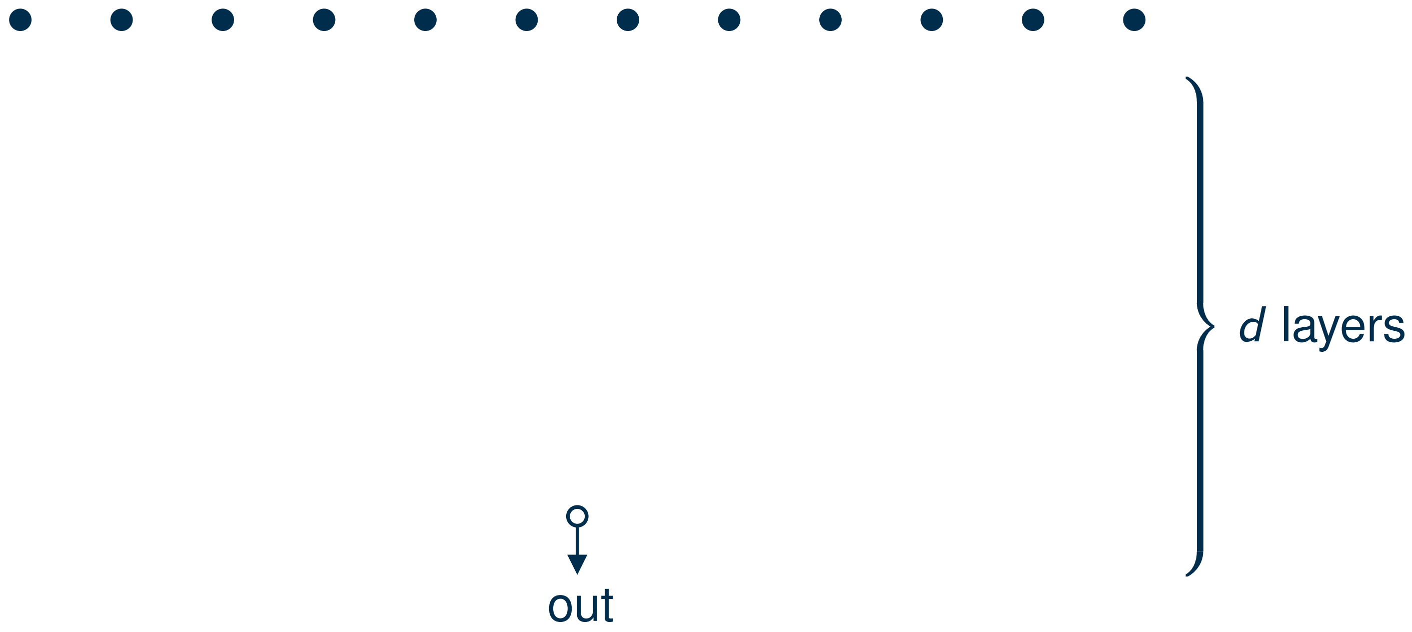
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

how many nodes can there be with in-degree > 2?

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

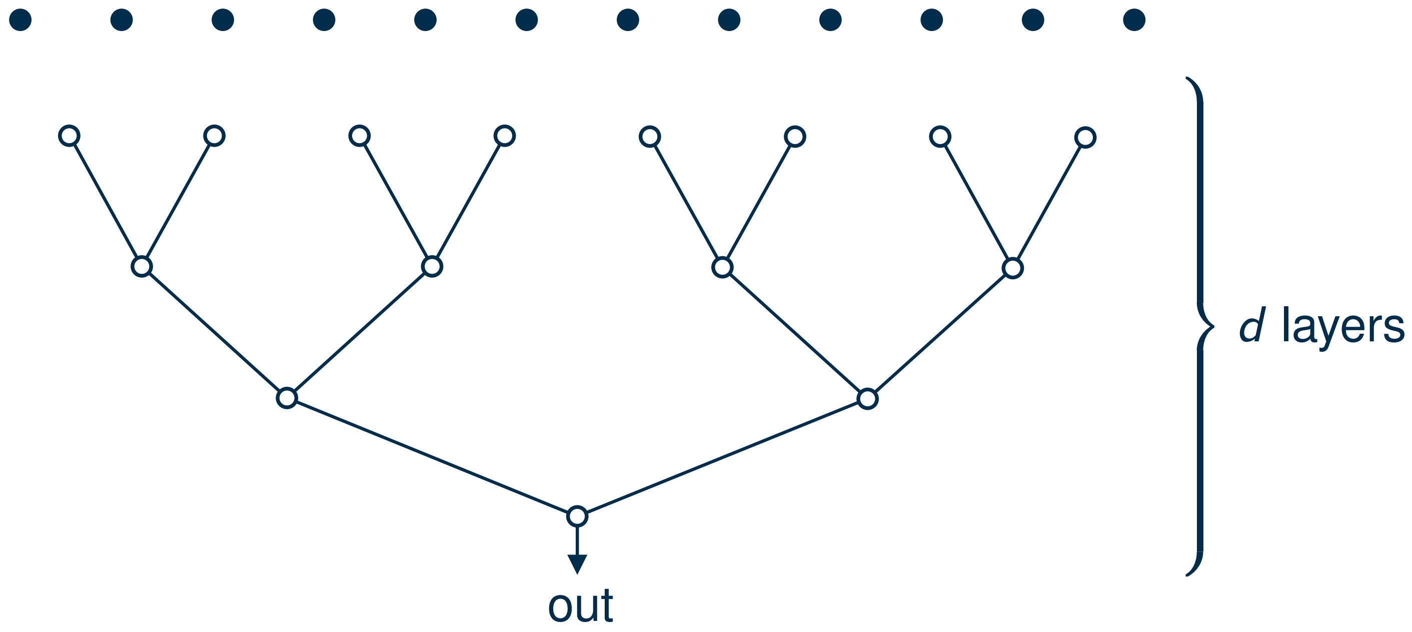
how many nodes can there be with in-degree > 2?



Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

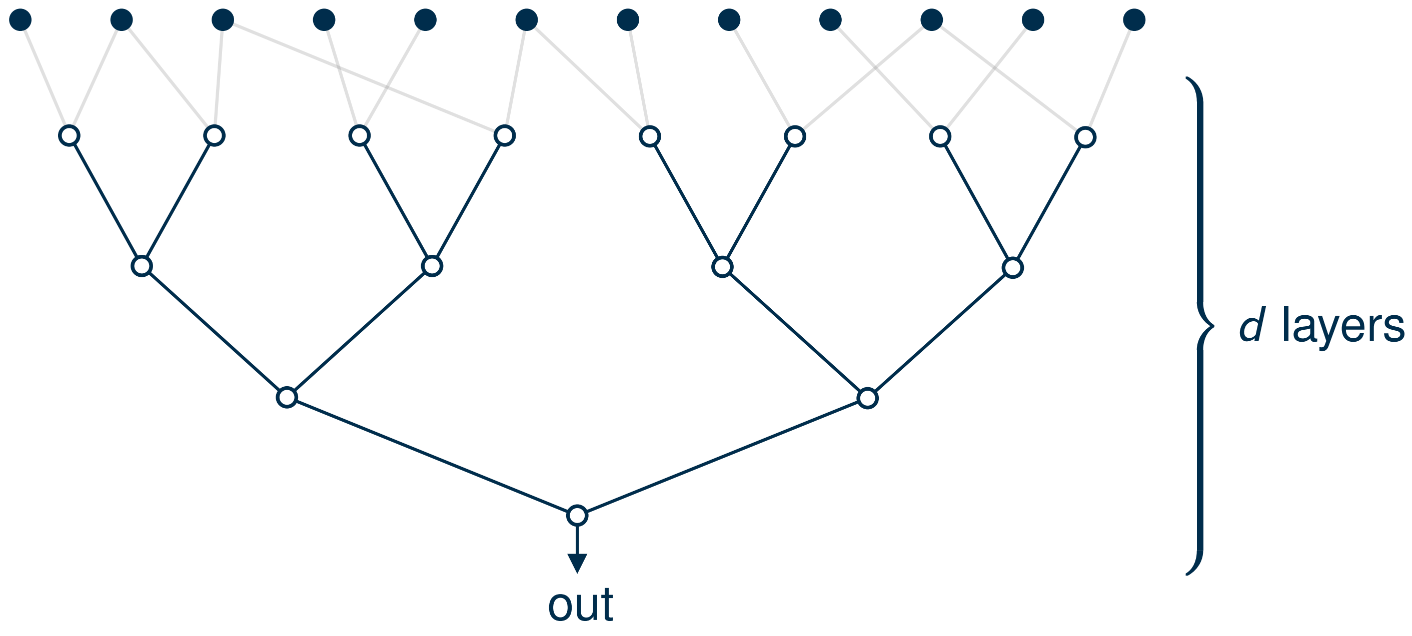
how many nodes can there be with in-degree > 2?



Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

how many nodes can there be with in-degree > 2?

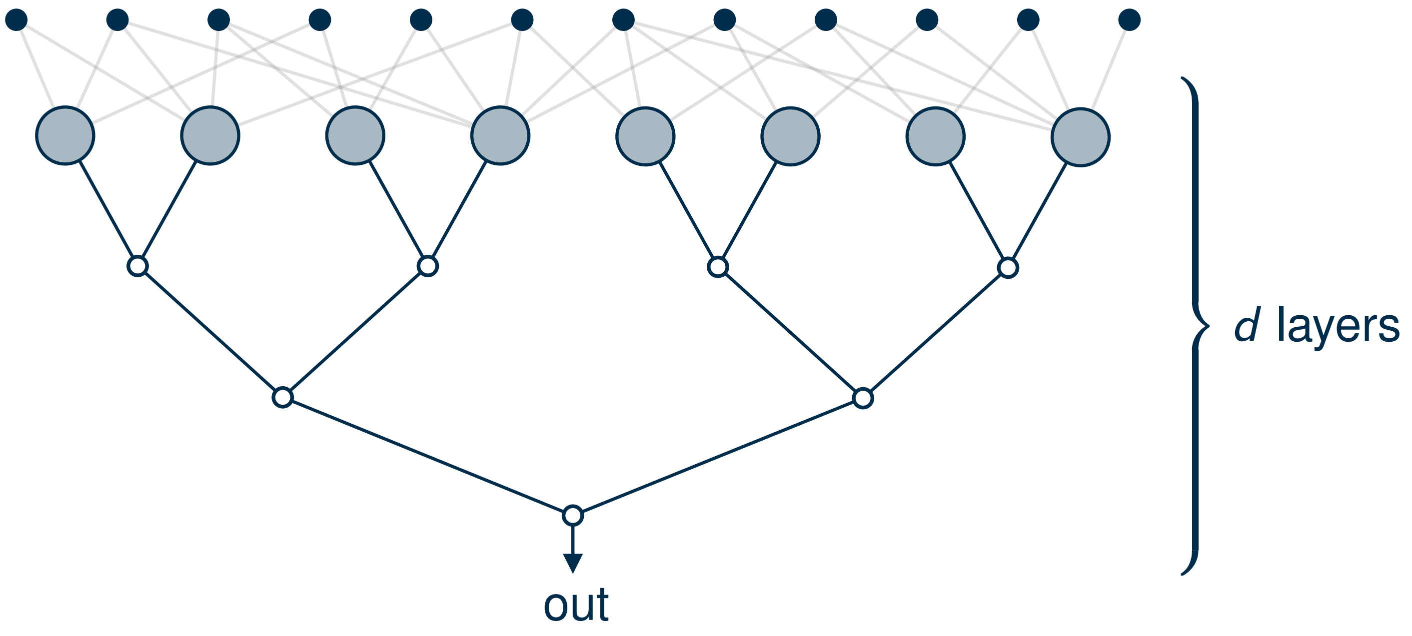


Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

how many nodes can there be with in-degree > 2?

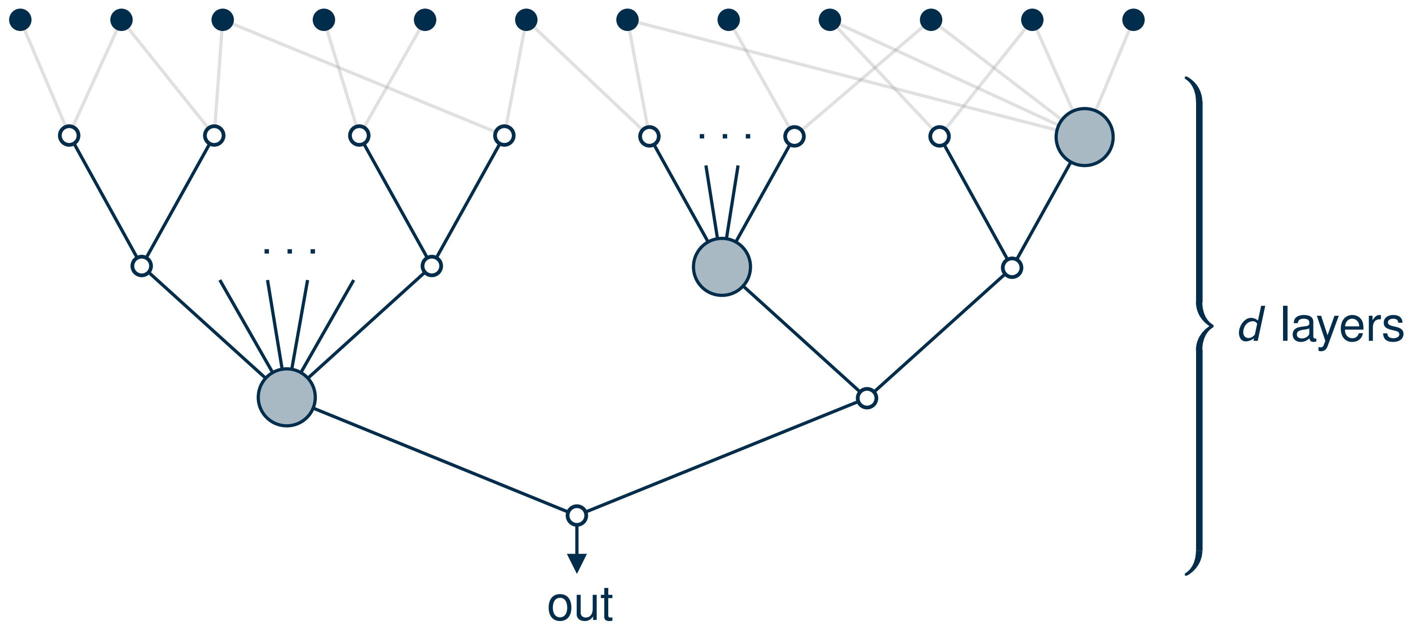
2^{d-1} large gates



Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

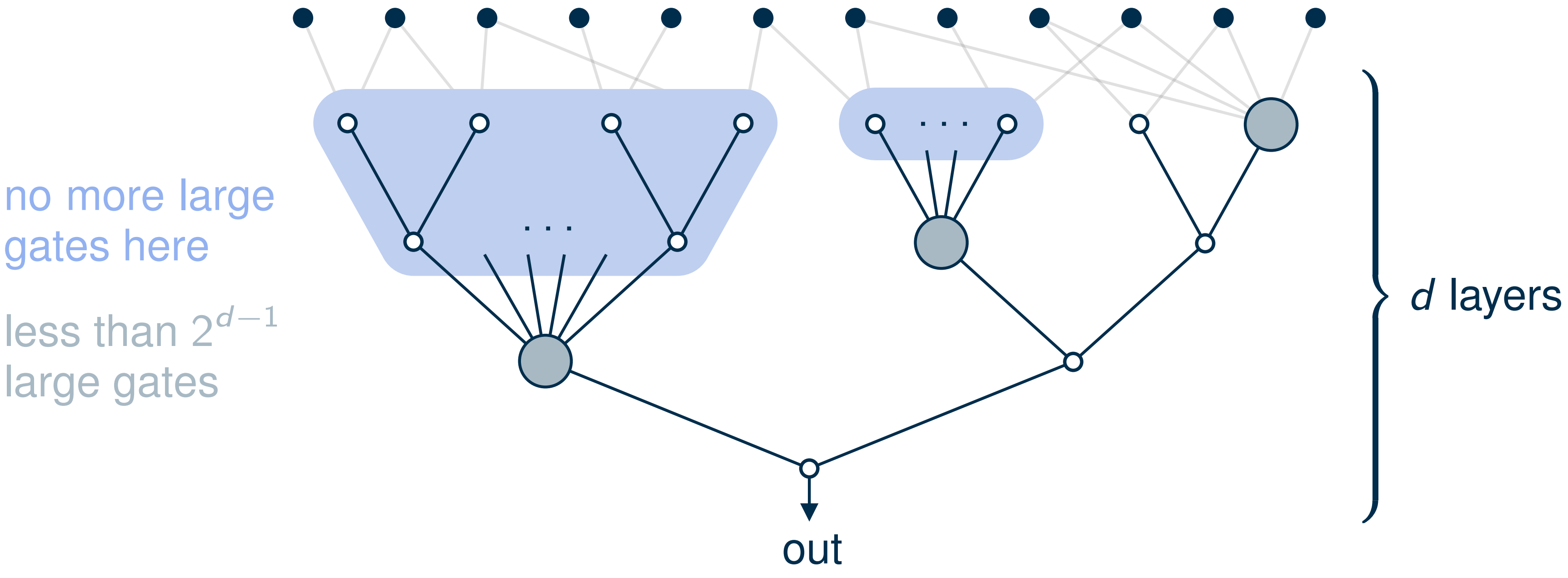
how many nodes can there be with in-degree > 2?



Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

how many nodes can there be with in-degree > 2?

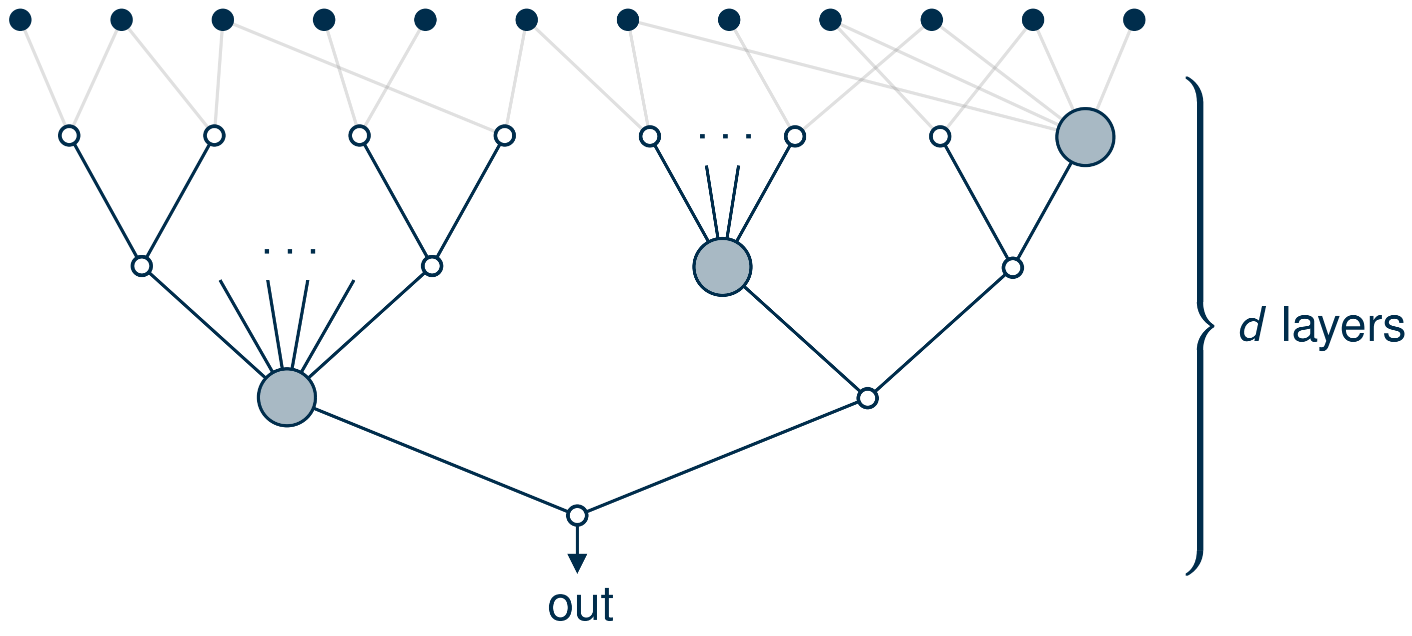


Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist

$\leq 2^{d-1}$ large gates

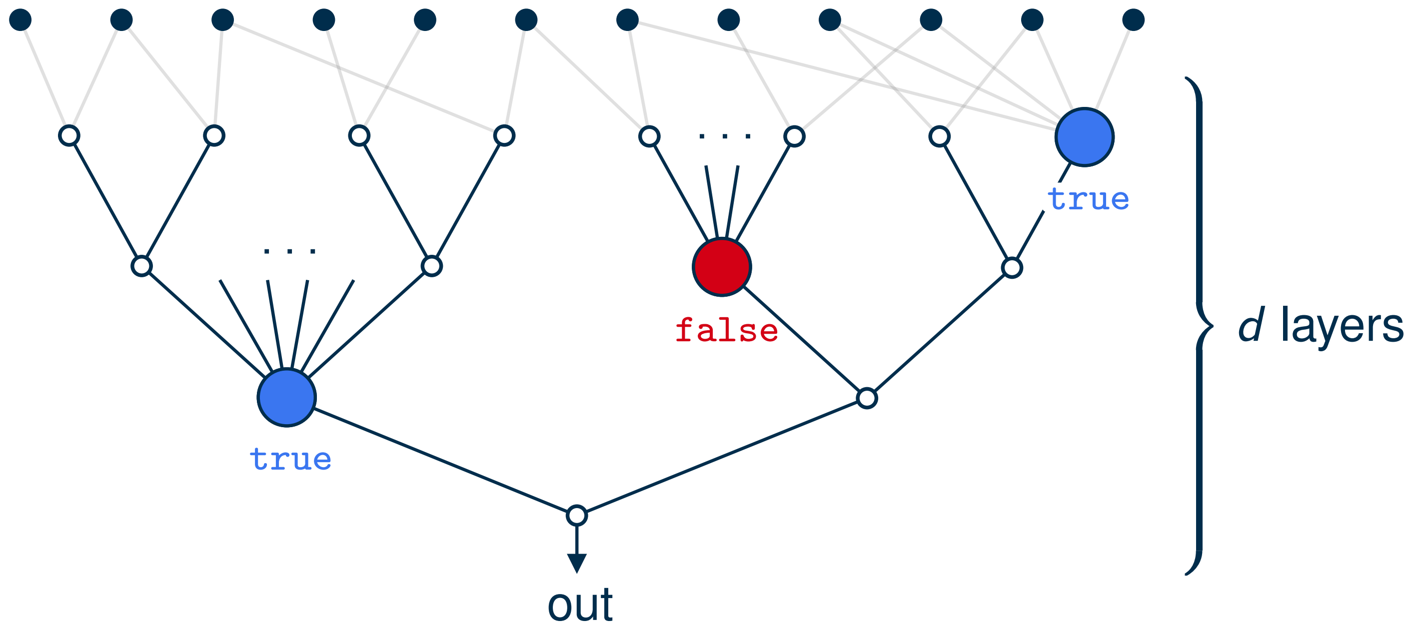


Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates



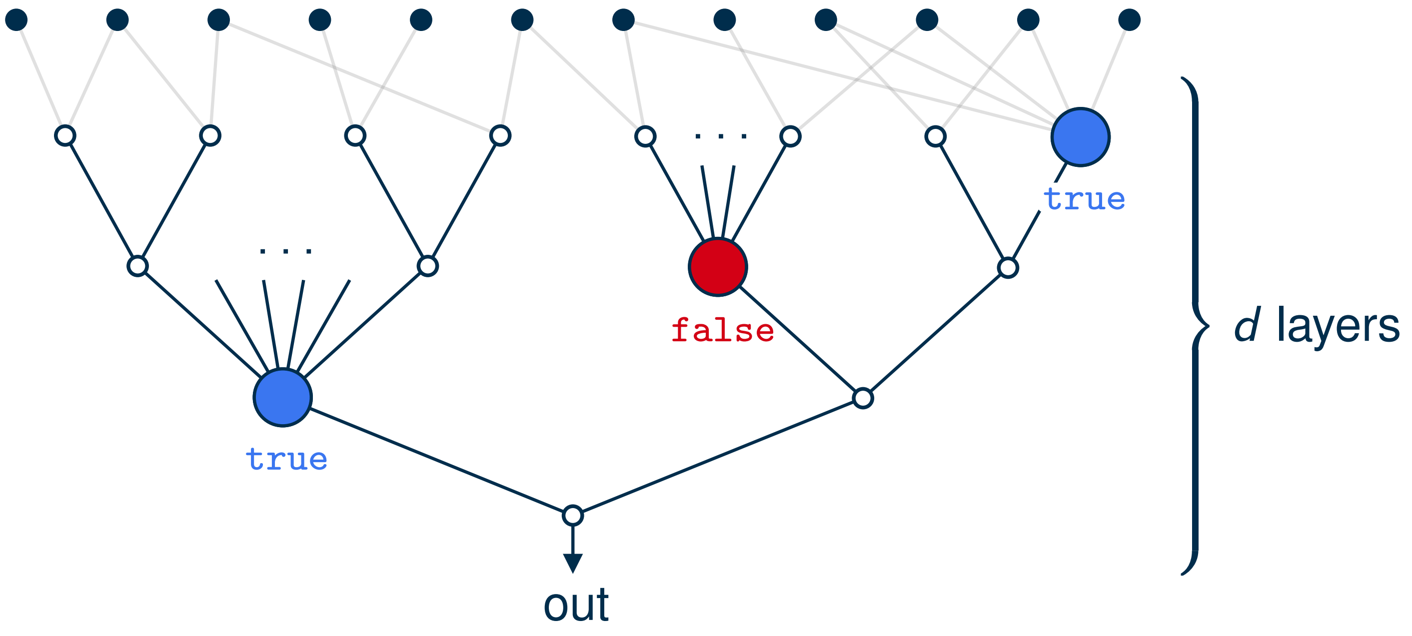
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

$\leq 2^{2^{d-1}}$ possible assignments



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

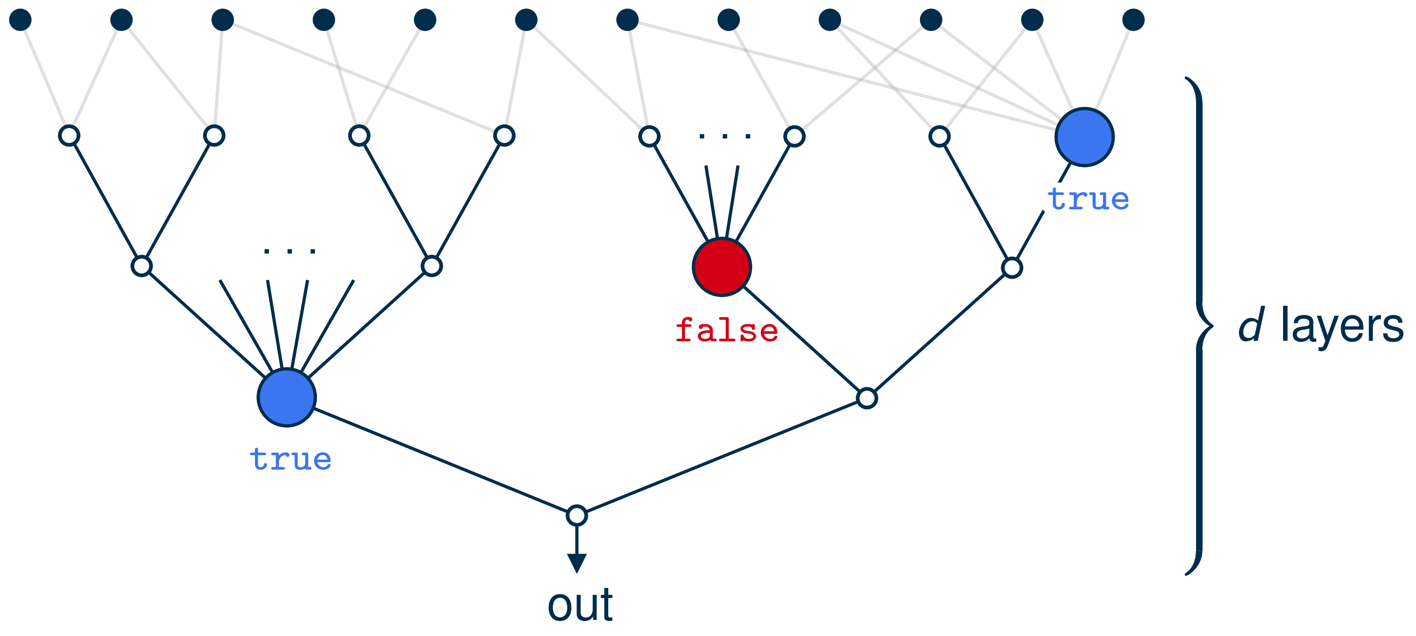
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

$\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

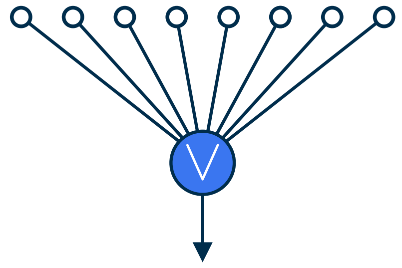
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

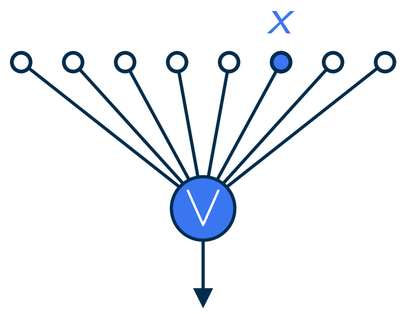
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many “large” gates (in-deg > 2) exist $\leq 2^{d-1}$ large gates
- “guess” truth values of outputs of large gates $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - “guess” which one $\leq$ *n* options per gate



“guess” = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

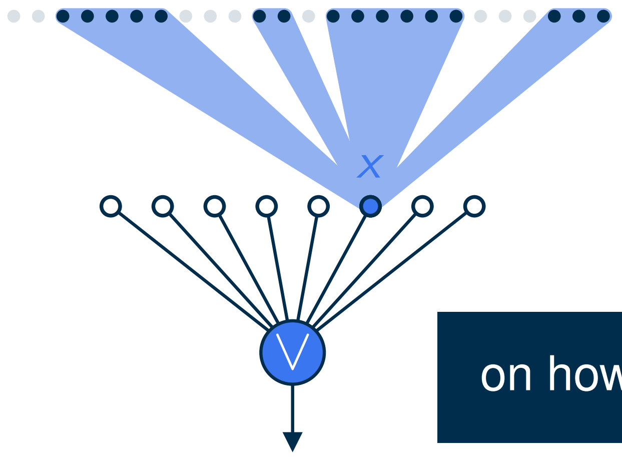
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate



on how many inputs does x depend?

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

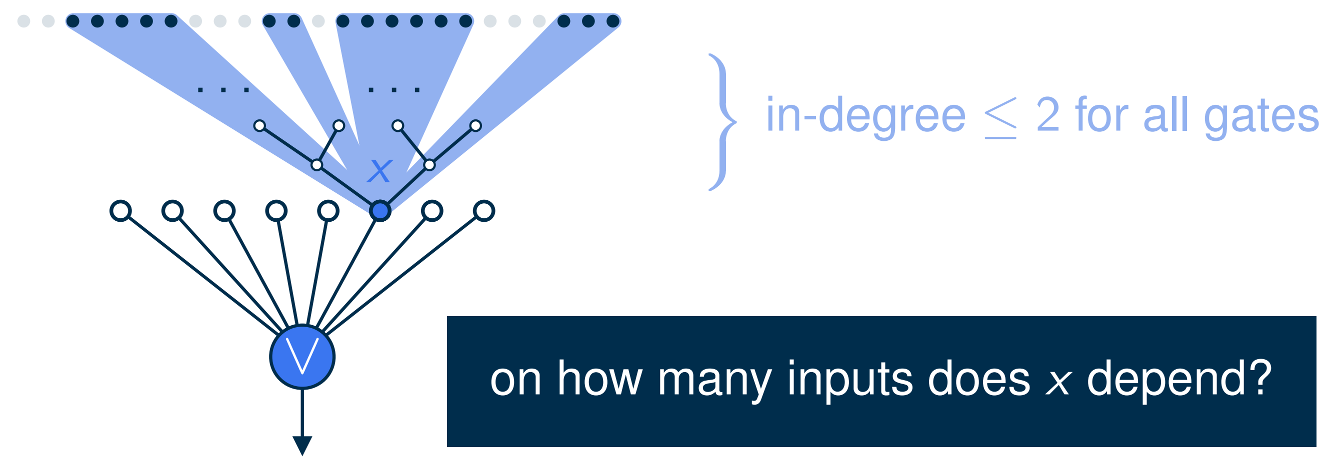
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

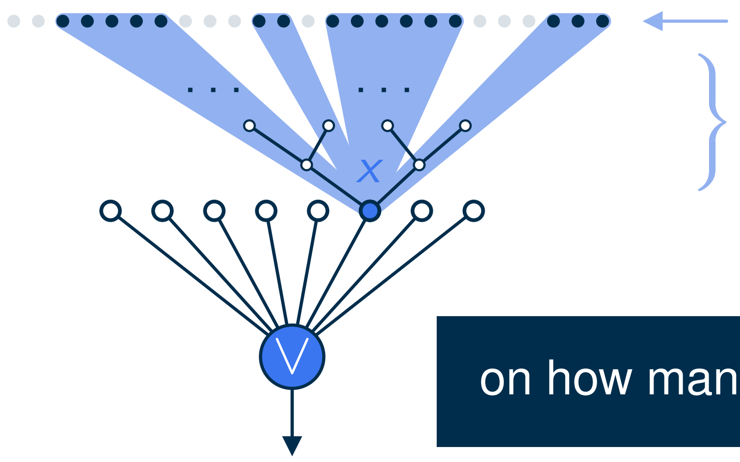
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

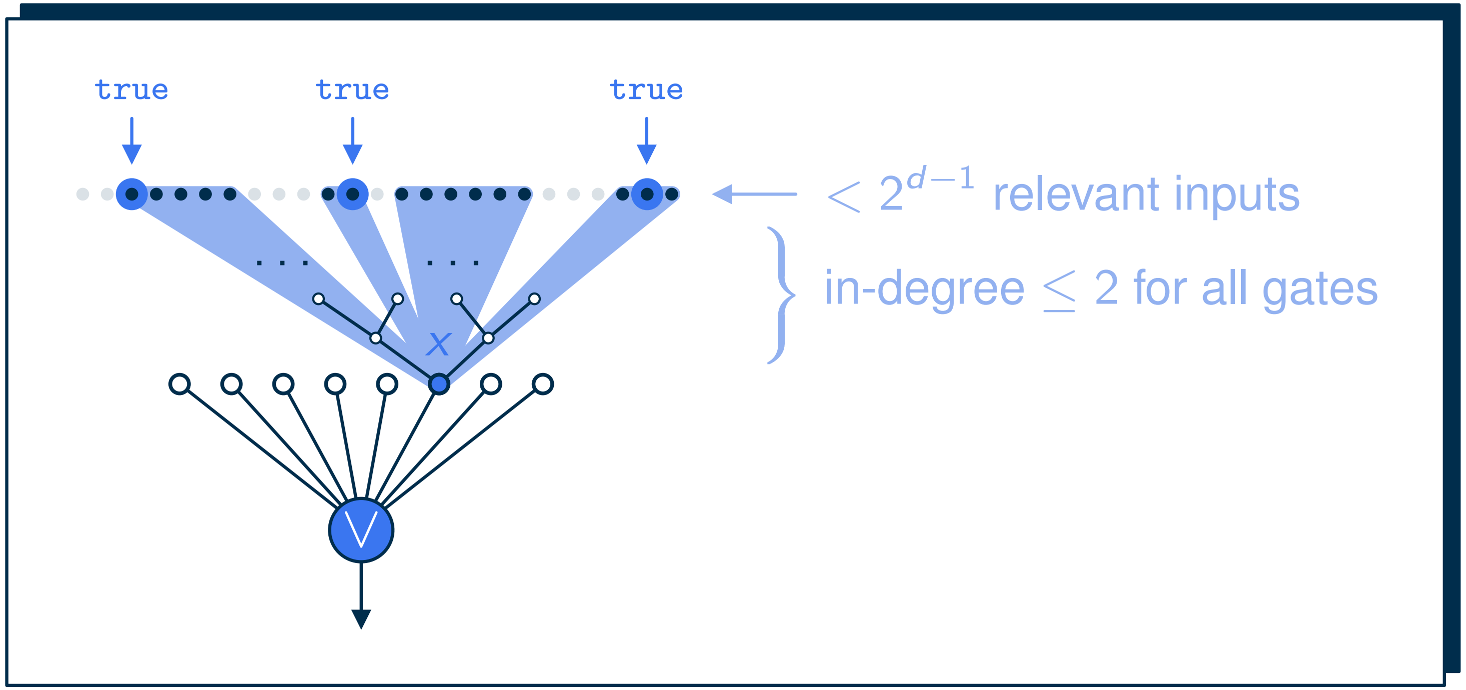
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

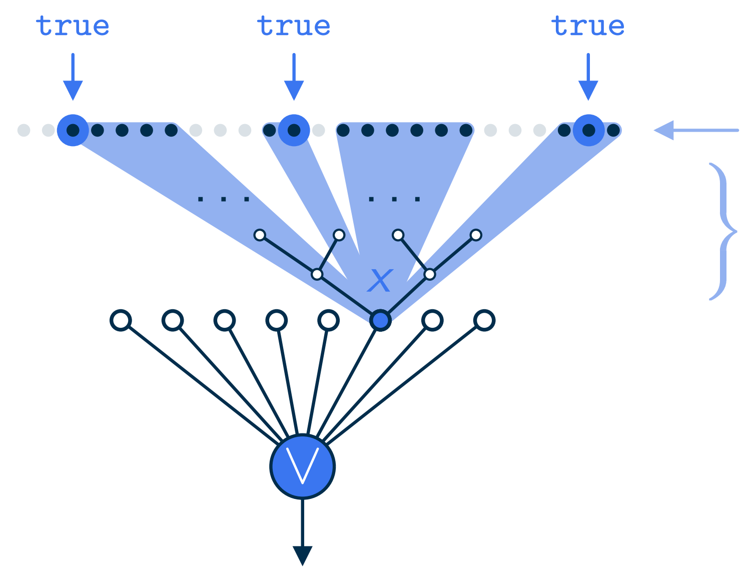
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs



$< 2^{2^{d-1}}$ assignments

$< 2^{d-1}$ relevant inputs

in-degree ≤ 2 for all gates

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

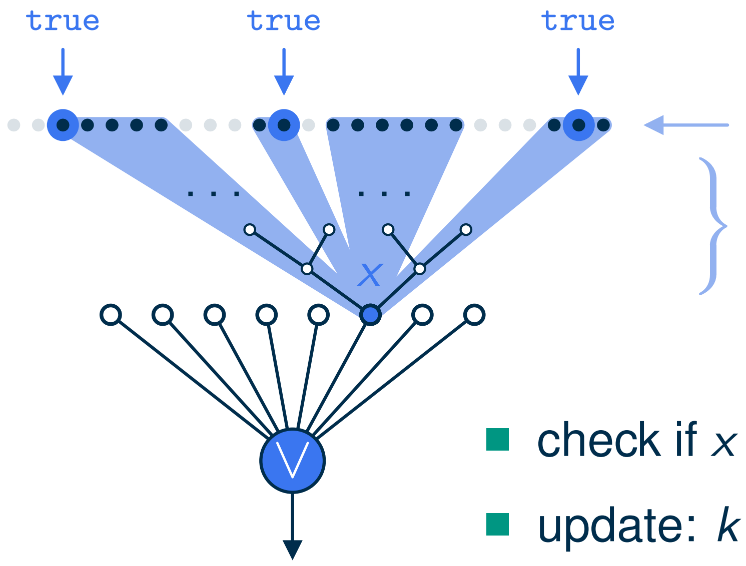
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs



$< 2^{2^{d-1}}$ assignments

$< 2^{d-1}$ relevant inputs

in-degree ≤ 2 for all gates

- check if x satisfied (if not: next guess)
- update: $k \leftarrow k - \#$ true inputs

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

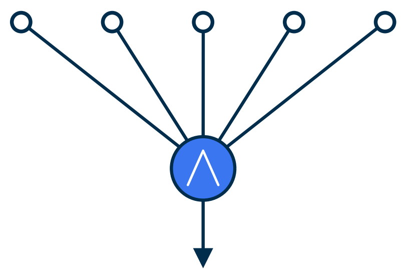
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

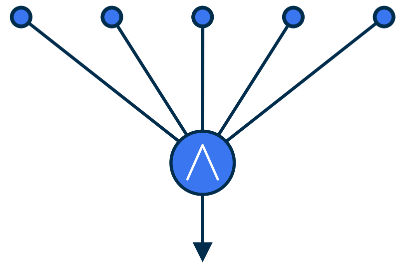
- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

$\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

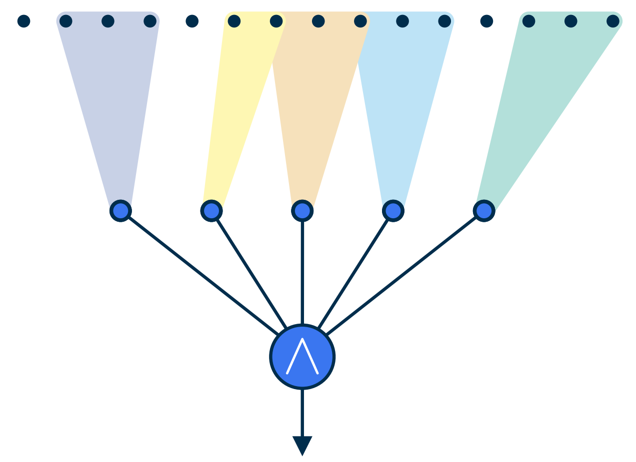
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

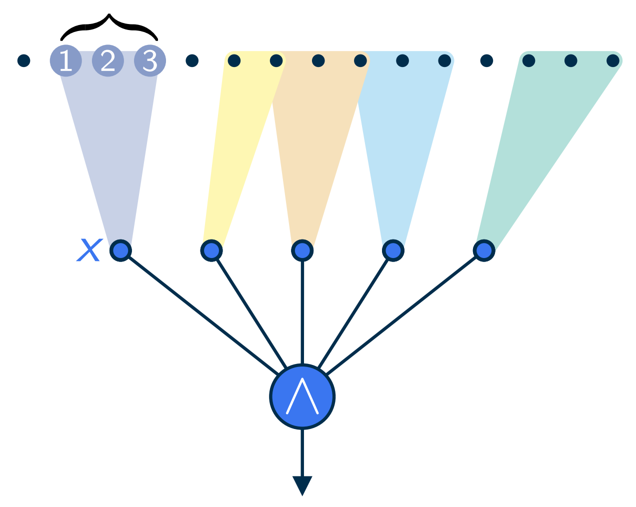
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

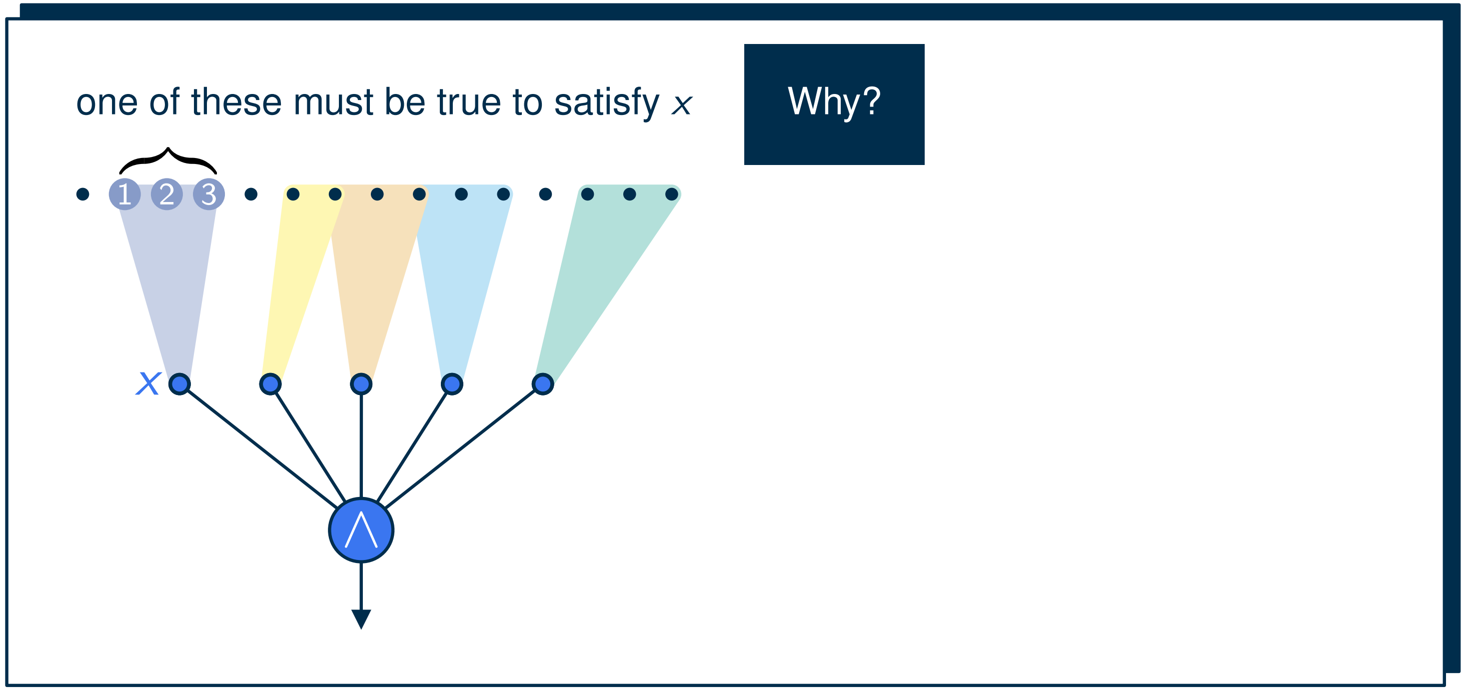
- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

$\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

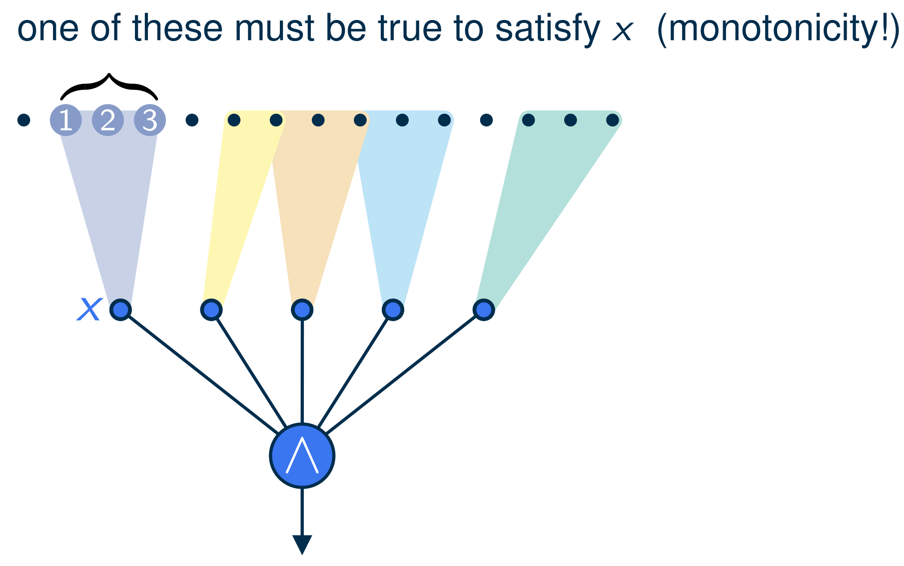
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

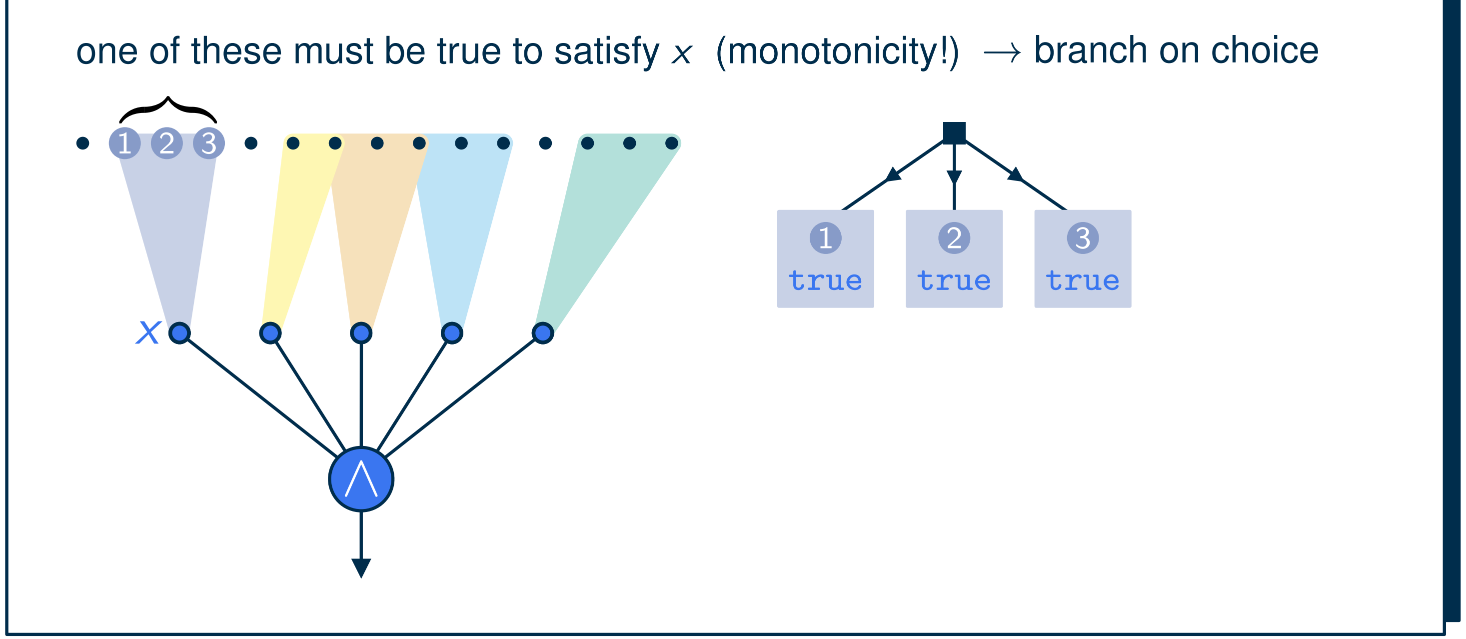
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

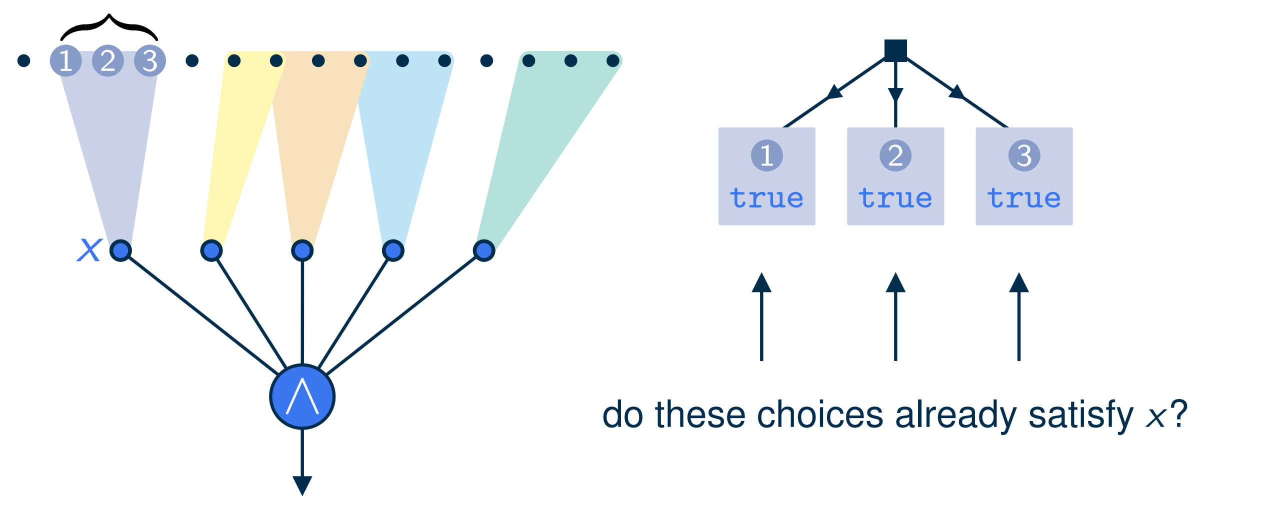
- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

$\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

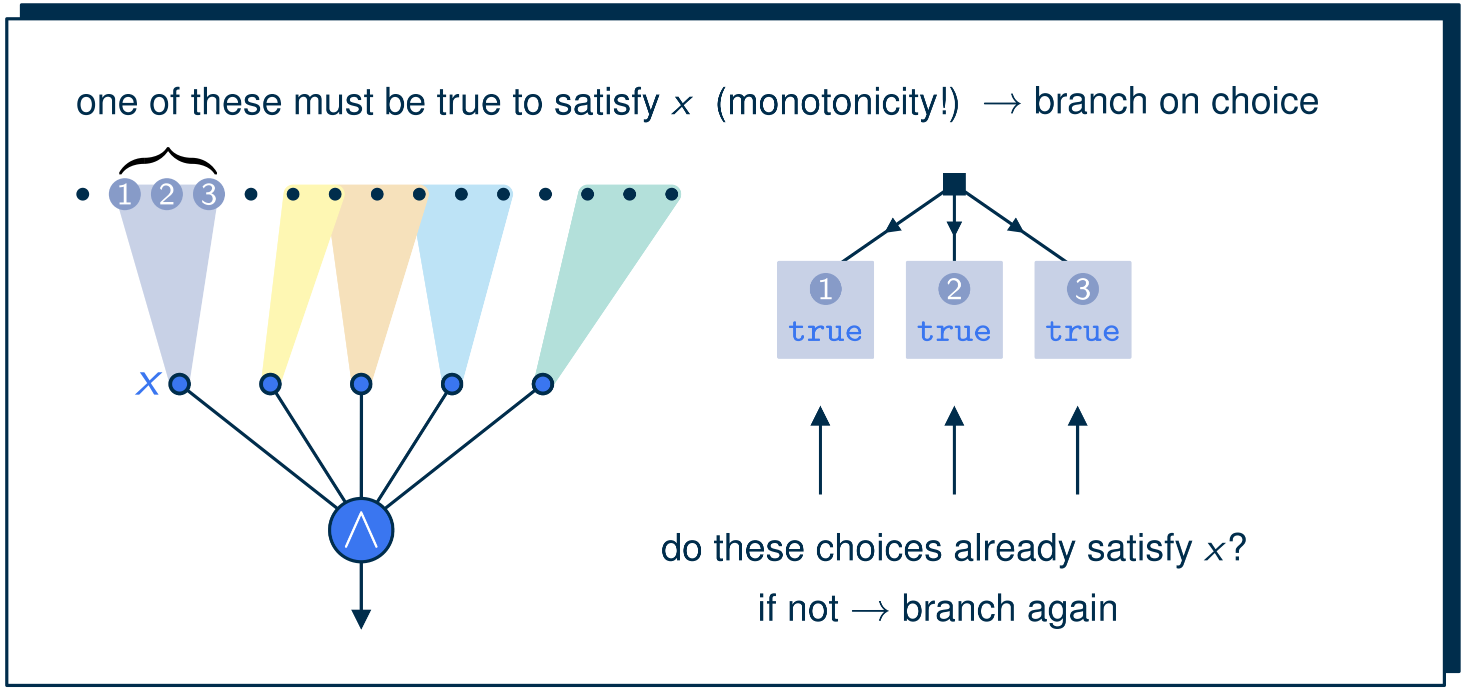
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it



Monotone Weft-1 Circuits

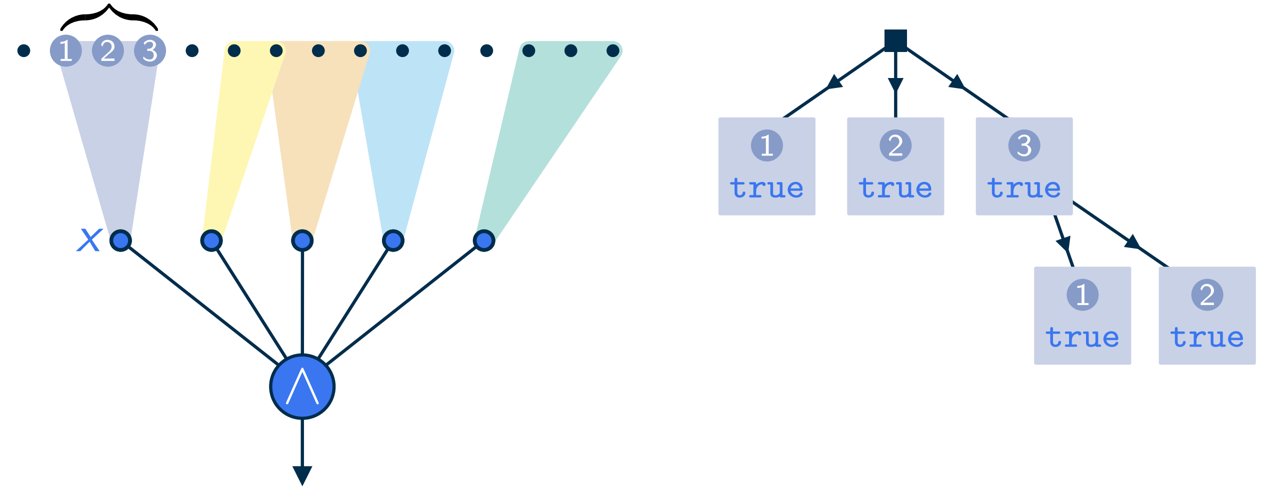
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true

one of these must be true to satisfy x (monotonicity!) $\rightarrow$ branch on choice



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

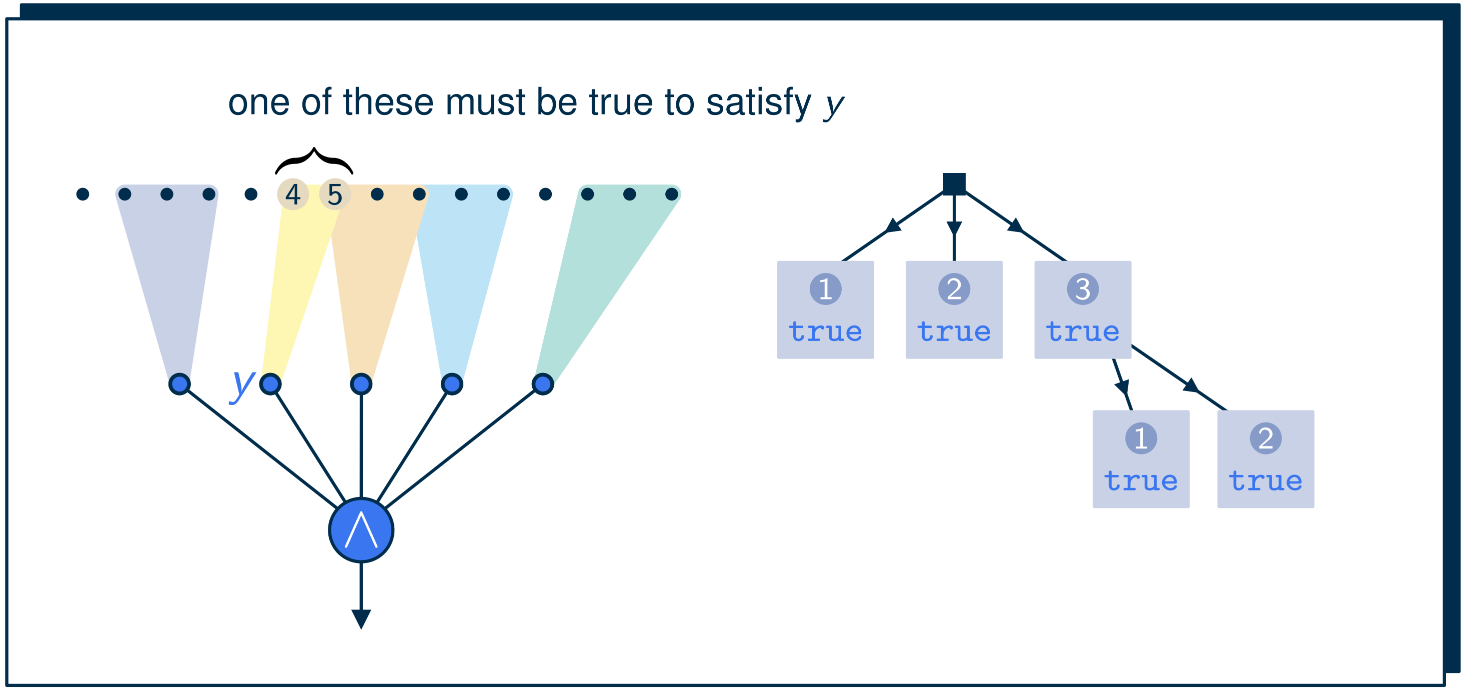
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

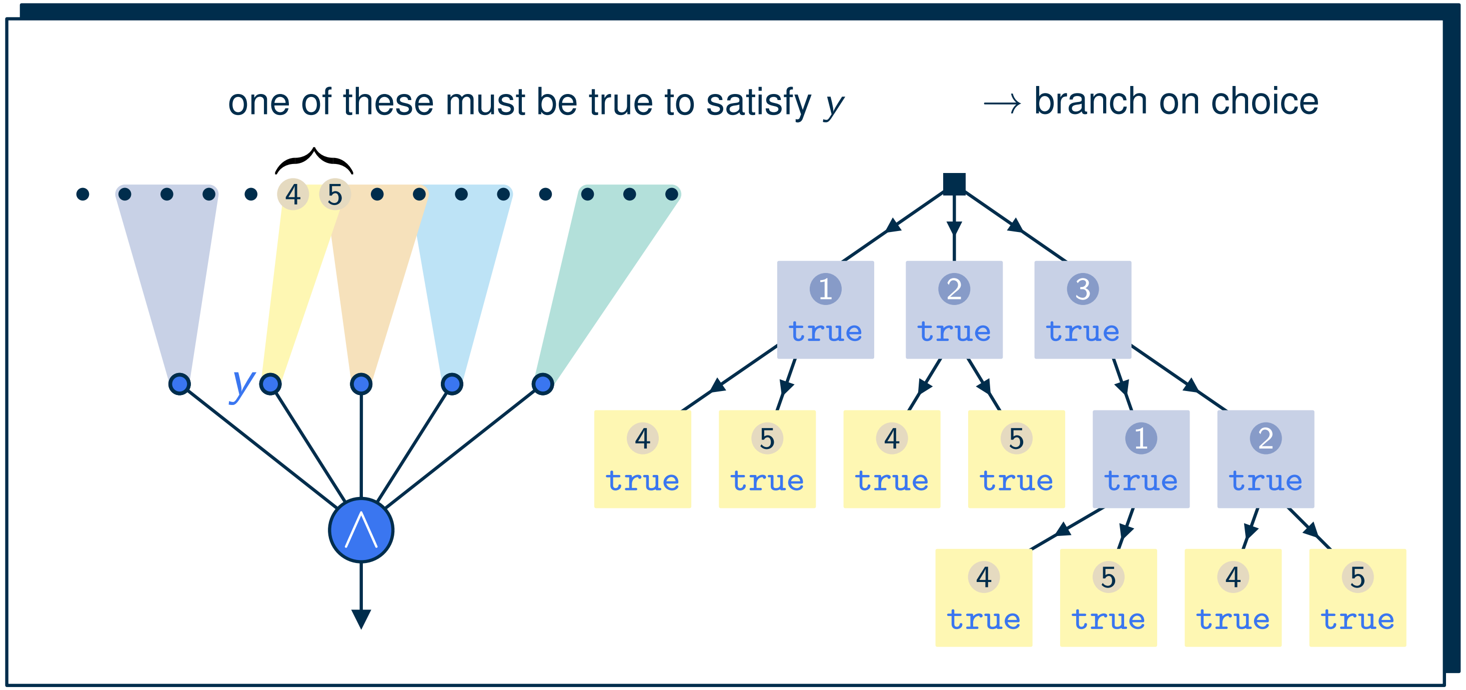
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

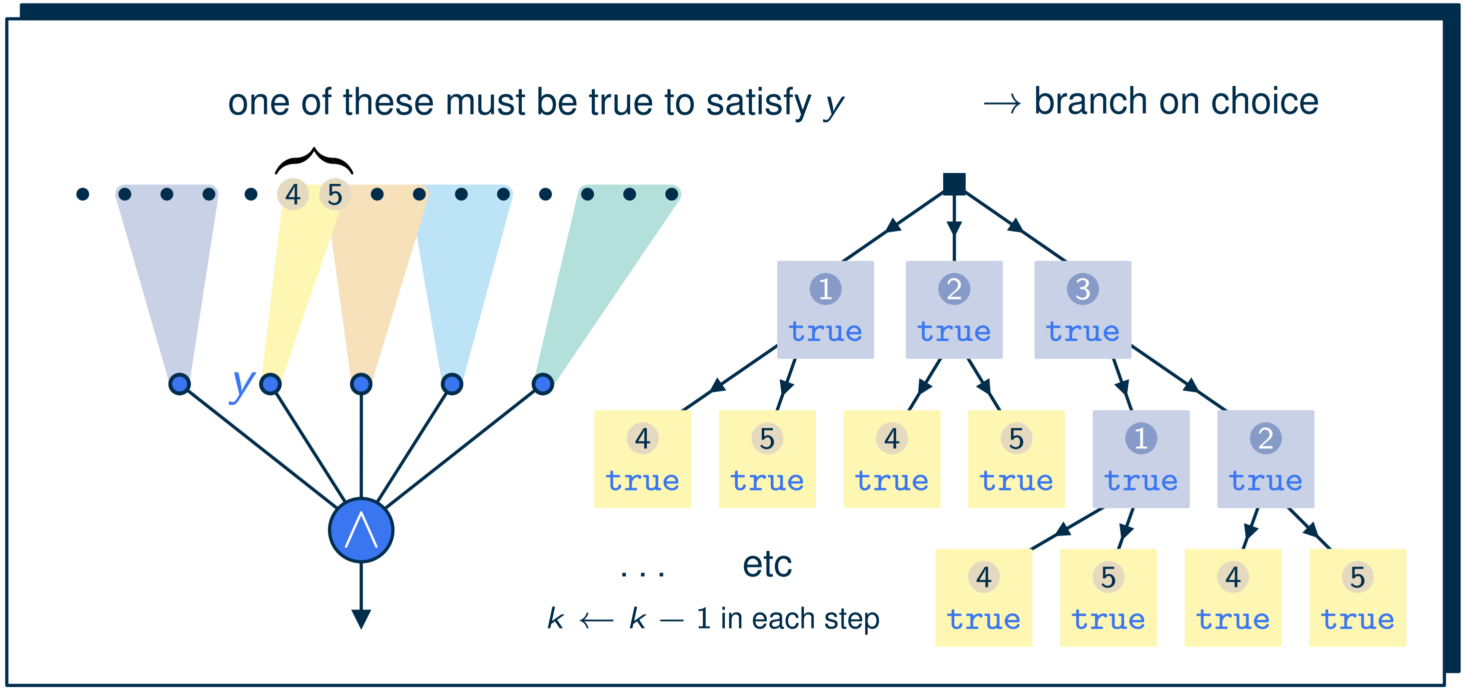
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

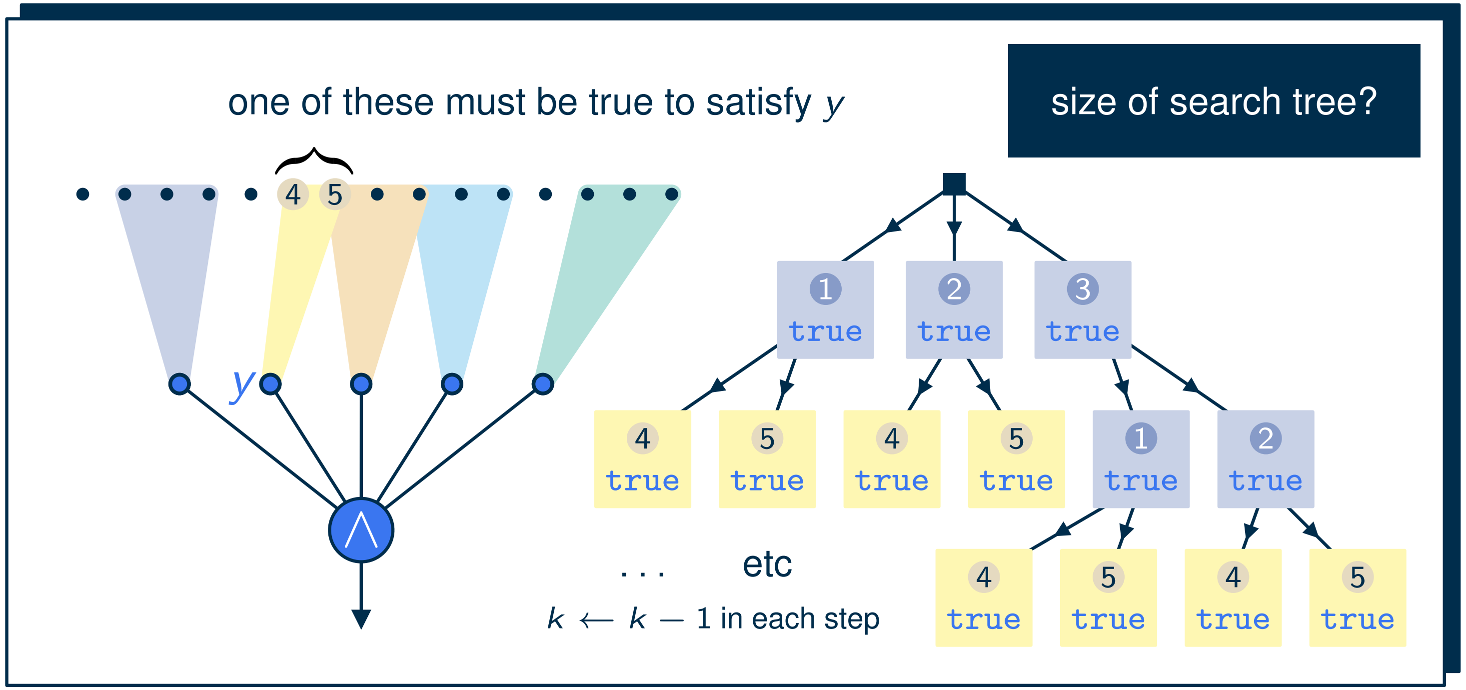
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many “large” gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- “guess” truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - “guess” which one — $\leq n$ options per gate
  - “guess” truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)

“guess” = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it



Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

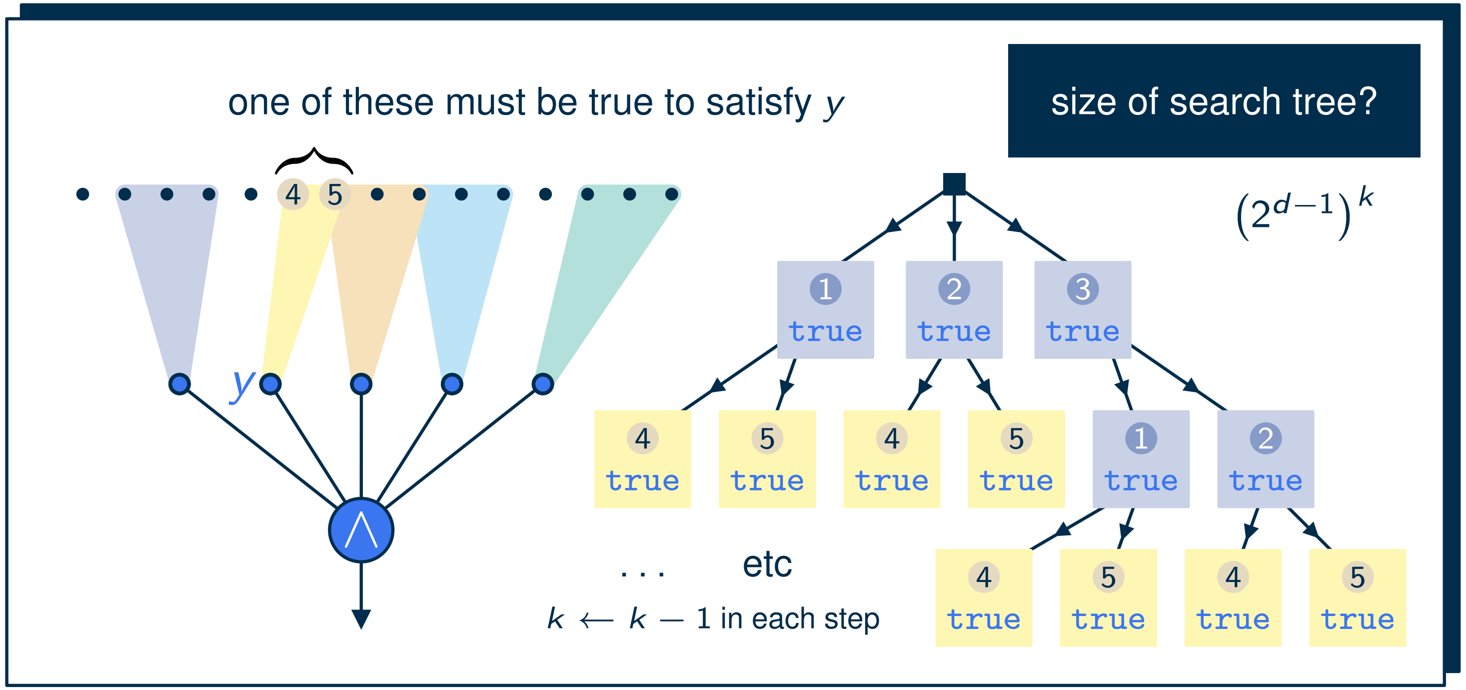
- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true

    (common search tree for all AND-gates)

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it



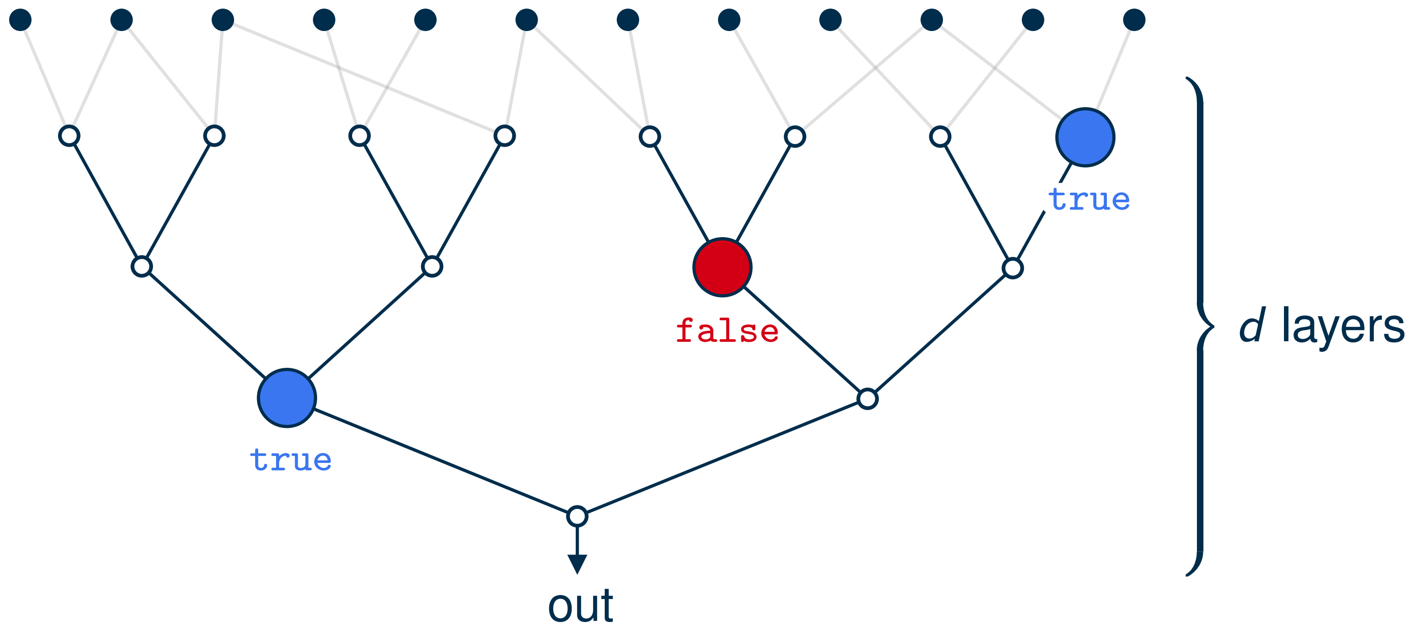
Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)



"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

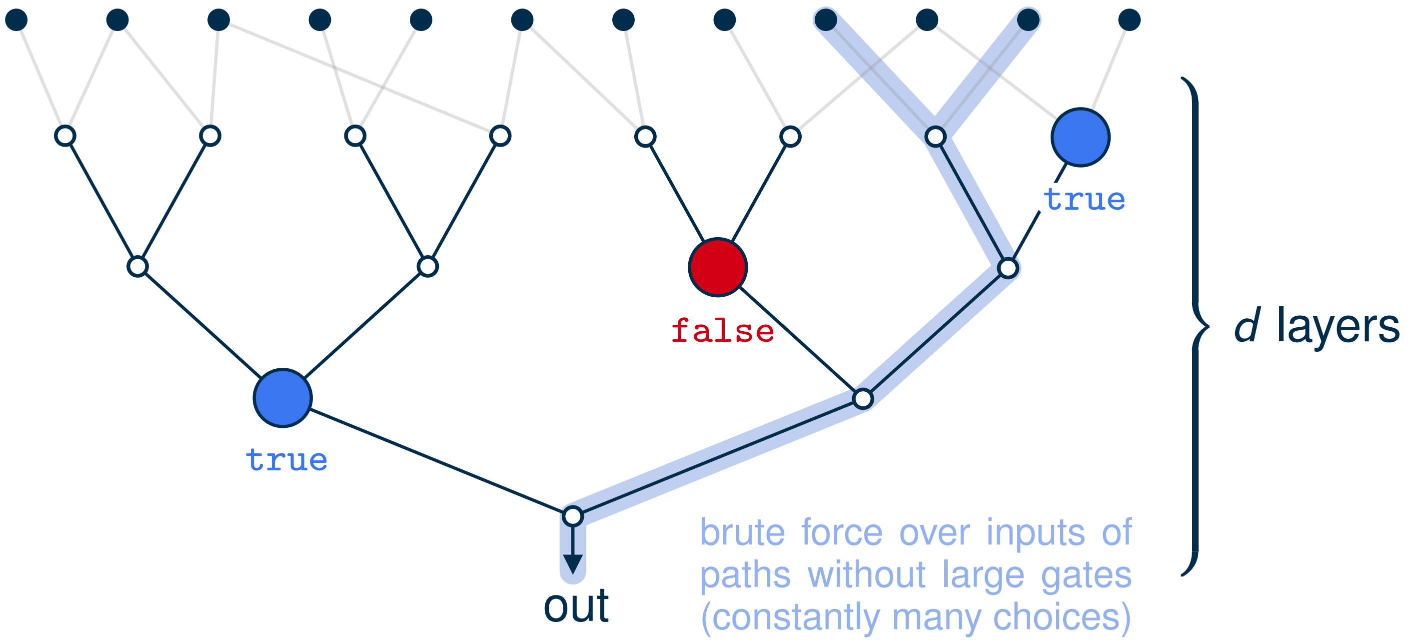
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- "guess" truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one — $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it



Monotone Weft-1 Circuits

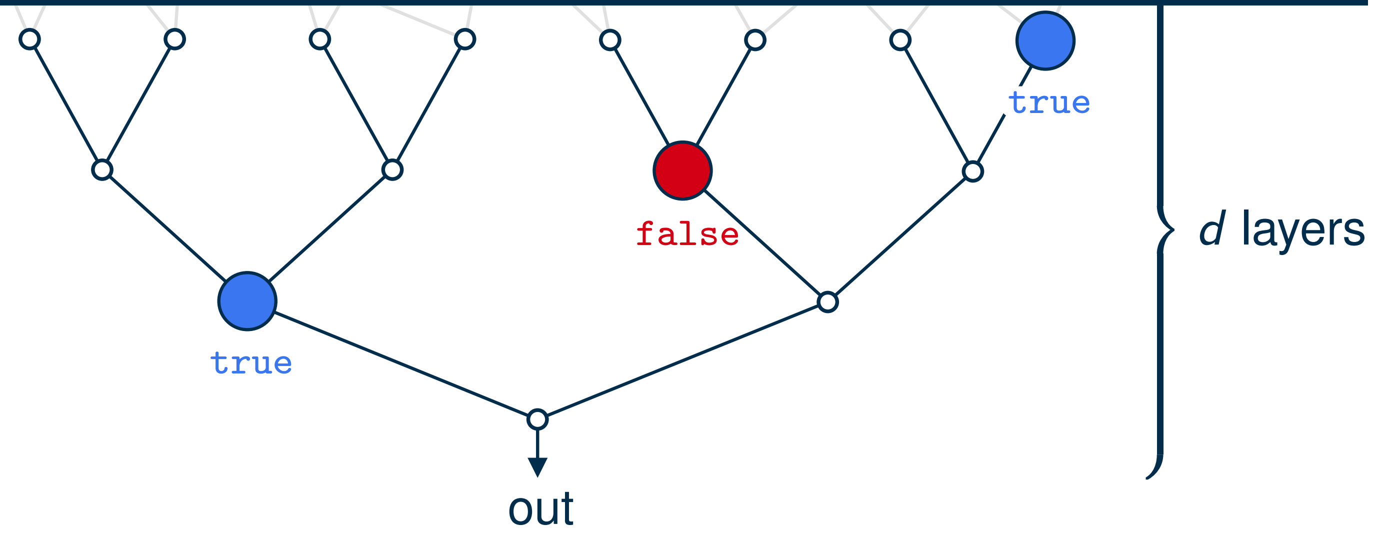
Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many “large” gates (in-deg > 2) exist — $\leq 2^{d-1}$ large gates
- “guess” truth values of outputs of large gates — $\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - “guess” which one — $\leq n$ options per gate
  - “guess” truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)

what about large gates that are false in the guessed assignment?



“guess” = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

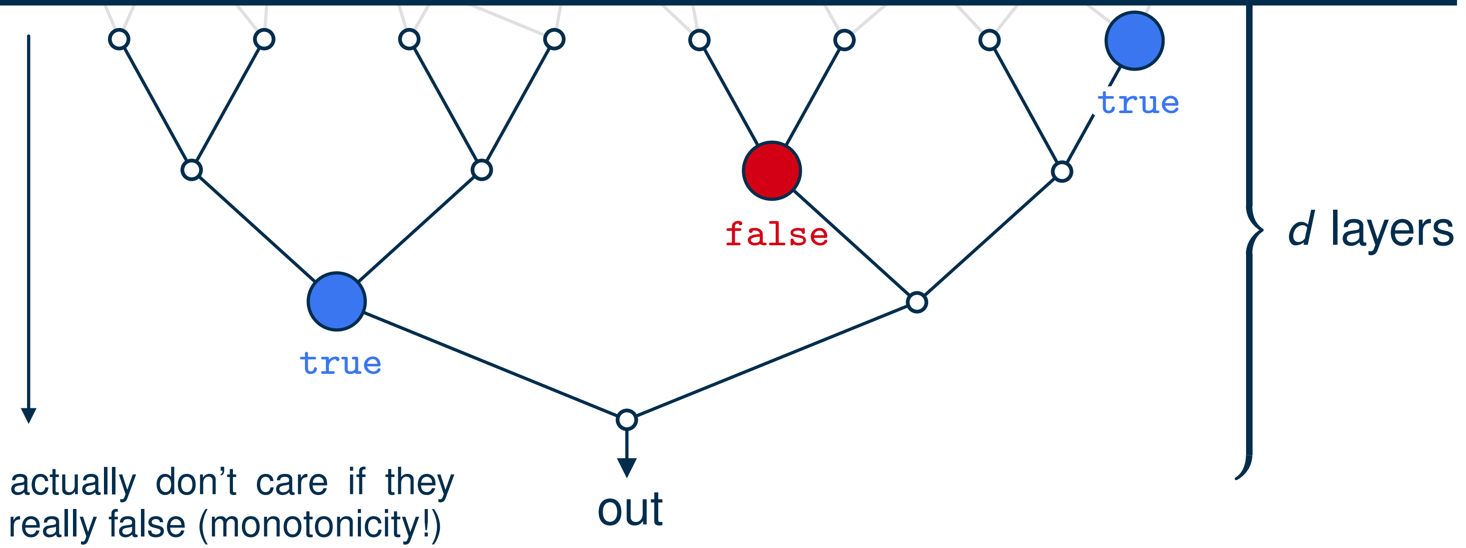
$\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq n$ options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)

"guess" = actually, we iterate over all possibilities, but for now we fix one and describe how to solve it

what about large gates that are false in the guessed assignment?



we actually don't care if they are really false (monotonicity!)

Monotone Weft-1 Circuits

Given an instance of WEIGHTED CIRCUIT SATISFIABILITY with weft 1, constant depth d, and without negation nodes, together with a parameter k. Is there a satisfying assignment of weight $\leq k$?

- only constantly many "large" gates (in-deg > 2) exist
- "guess" truth values of outputs of large gates

$\leq 2^{d-1}$ large gates

$\leq 2^{2^{d-1}}$ possible assignments

New question: Can the guessed truth values of large nodes be achieved by setting $\leq k$ inputs to true?

- large true OR-gates: ≥ 1 parent true
  - "guess" which one $\leq$ *n* options per gate
  - "guess" truth values of relevant inputs
- large true AND-gates: all parents true
  - repeatedly branch over which input values are set to true
    (common search tree for all AND-gates)
- check if global output is true with chosen assignment
  (if not: next iteration)