Hamiltonian Cycle in EPT time for a non-sparse parameter

By Sigve Hortemo Sæther

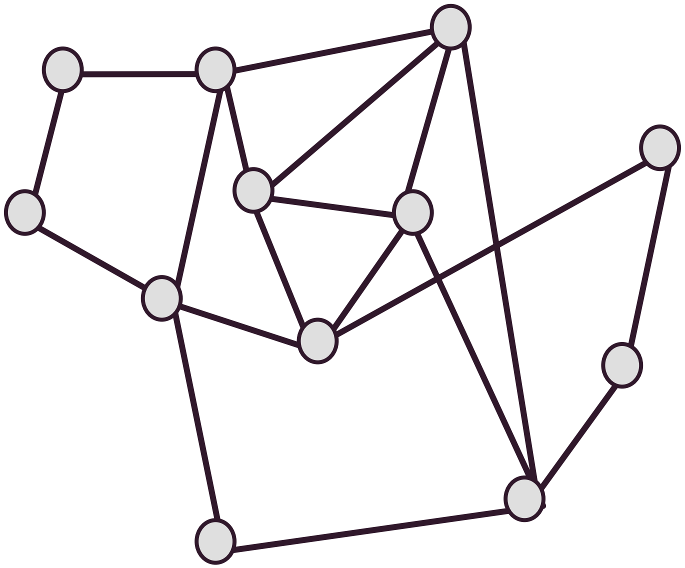
University of Bergen

for CALDAM 2015

Hamilonian Cycle (HC)

Question:

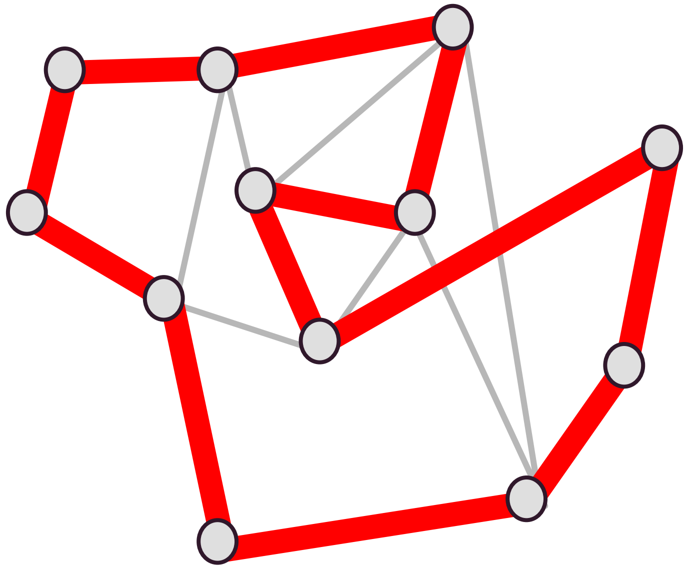
Does there exist a cycle of length n?



Hamilonian Cycle (HC)

Question:

Does there exist a cycle of length n?



Lower bound on HC

- Unlikely to be solvable in poly-time (NP-hard)
- No "natural" parameter (unlike e.g., Vertex Cover)
- Solvable in $n^{O(1)}2^{O(tw(G))}$-time (optimal by ETH)

Lower bound on HC

- Unlikely to be solvable in poly-time (NP-hard)
- No “natural” parameter (unlike e.g., Vertex Cover)
- Solvable in $n^{O(1)}2^{O(tw(G))}$-time (optimal by ETH)
  - Implies unlikely for $n^{O(1)}2^{o(smw(G))}$-time

little-o

Goal (EPT algorithm)

Show $n^{O(1)}2^{O(smw(G))}$-time algo for HC

Goal (EPT algorithm)

Show $n^{O(1)} 2^{O(smw(G))}$-time algo for HC

I will hide the $n^{O(1)}$ part in runtimes from here on out

Goal (EPT algorithm)

Show $2^{O(smw(G))}$-time algo for HC

Goal (EPT algorithm)

Show $2^{O(smw(G))}$-time algo for HC

1) Find a decomposition **D** of sm-width ***k*** = O(smw(G)) (in $2^{O(k)}$-time)

2) Solve HC in $2^{O(k)}$-time using **D**

Goal (EPT algorithm)

Show $2^{O(smw(G))}$-time algo for HC

1) Find a decomposition **D** of sm-width ***k*** = O(smw(G)) (in $2^{O(k)}$-time)

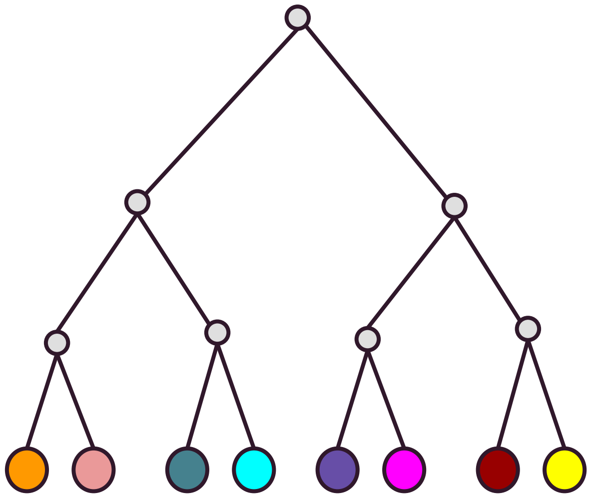
2) Solve HC in $2^{O(k)}$-time using **D**

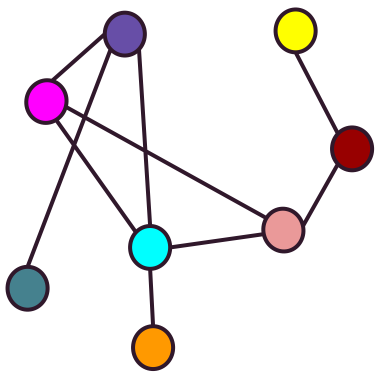
Branch decompositions (over V(G))

Branch decompositions (over V(G))

For dynamic programming.

D = (T, δ)

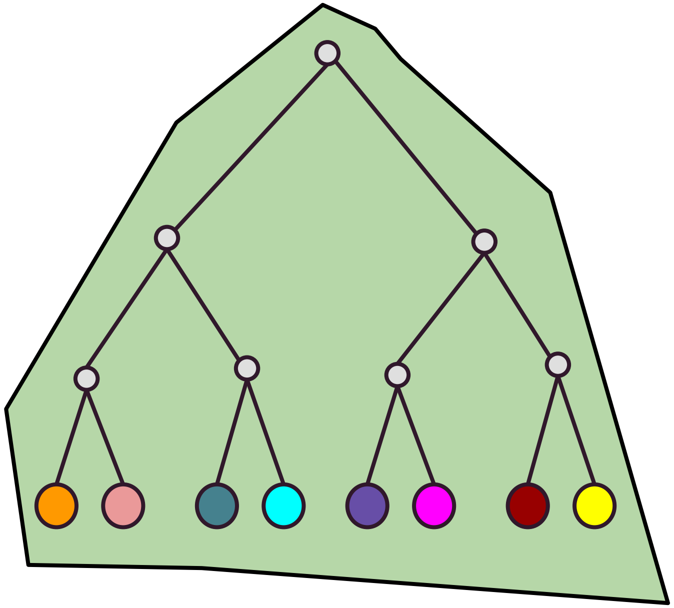


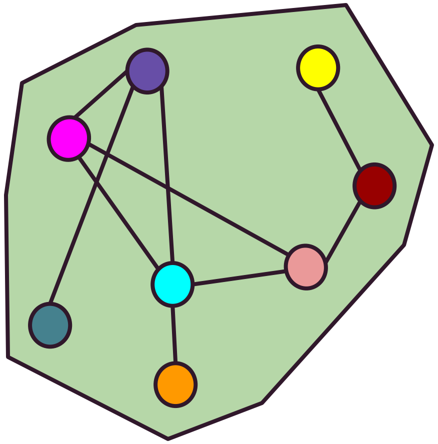


Branch decompositions (over V(G))

For dynamic programming.

D = (T, δ)

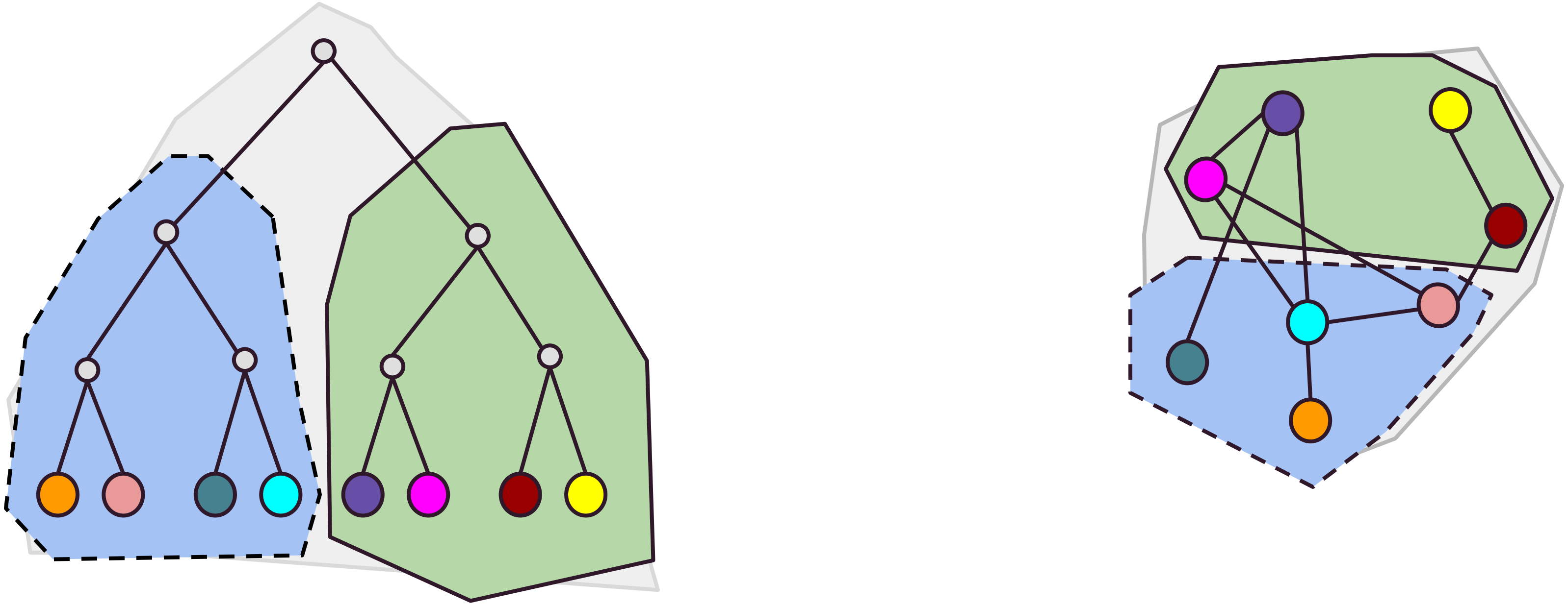




Branch decompositions (over V(G))

For dynamic programming.

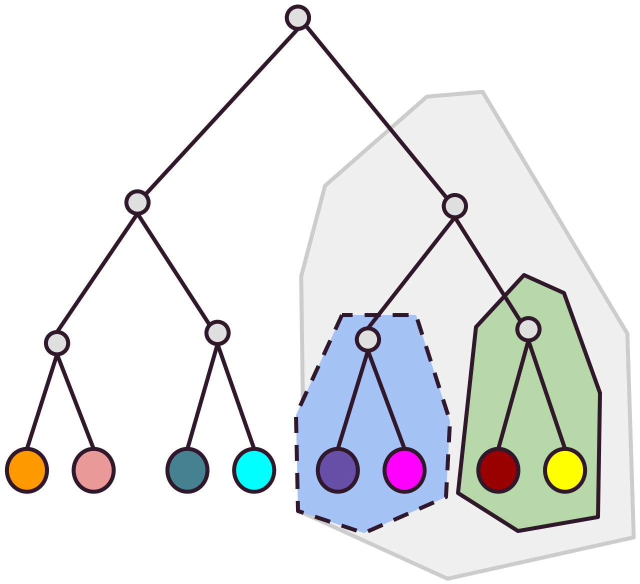
D = (T, δ)

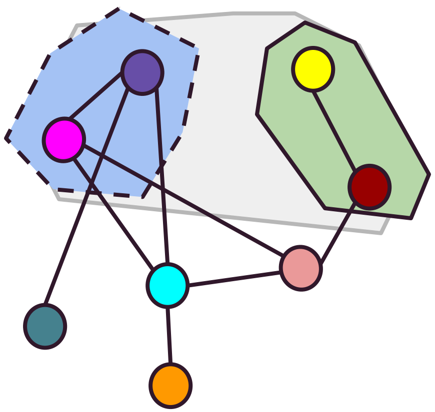


Branch decompositions (over V(G))

For dynamic programming.

D = (T, δ)

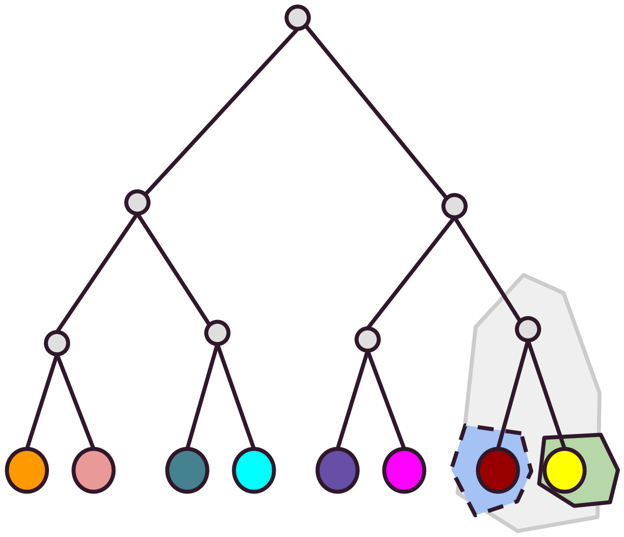


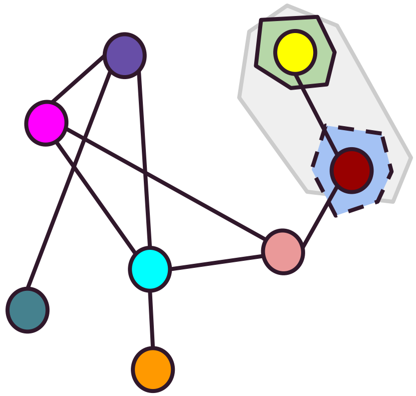


Branch decompositions (over V(G))

For dynamic programming.

$D = (T, \delta)$





Branch decompositions (over V(G))

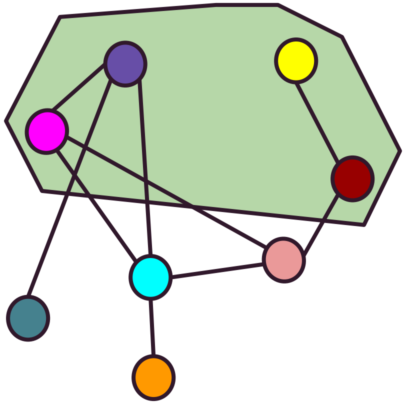
Cut function

$\boldsymbol{x} : 2^{V} \rightarrow \mathbb{N}$

Branch decompositions (over V(G))

Cut function

$\boldsymbol{x} : 2^{V} \rightarrow \mathbb{N}$

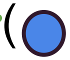


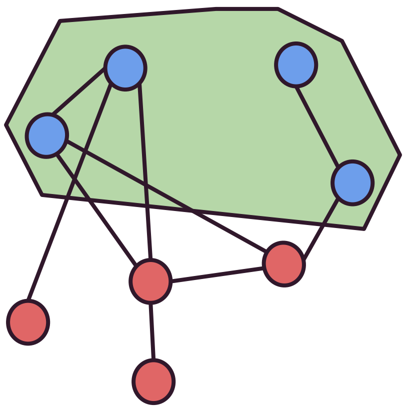
Branch decompositions (over V(G))

Cut function

$\boldsymbol{x} : 2^V \longrightarrow \mathbb{N}$

Example 1: $\boldsymbol{x}(S) = \delta(S)$ (# of edges crossing S, V\S)

$\Rightarrow$ $\boldsymbol{x}$() = $\boldsymbol{x}$() = **5**



Example 2: $\boldsymbol{x}$(S) = size of max. matching in G[S, V\S]

$\Rightarrow$ $\boldsymbol{x}$() = $\boldsymbol{x}$() = **3**

Branch decompositions (over V(G))

Cut function

$\boldsymbol{x} : 2^V \longrightarrow \mathbb{N}$

Carving-width

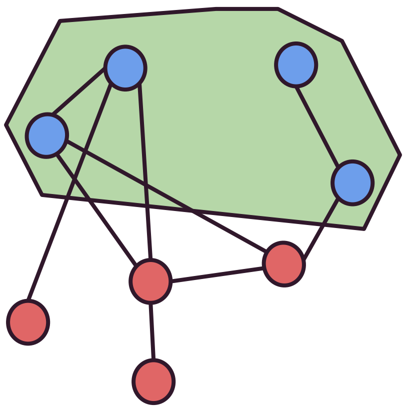
Example 1: $\boldsymbol{x}(S) = \delta(S)$ (# of edges crossing S, V\S)

$\Rightarrow \boldsymbol{x}($ blue $) = \boldsymbol{x}($ red $) = \mathbf{5}$

Example 2: $\boldsymbol{x}(S)$ = size of max. matching in G[S, V\S]

mm-width

$\Rightarrow \boldsymbol{x}($ blue $) = \boldsymbol{x}($ red $) = \mathbf{3}$



Branch decompositions (over V(G))

Cut function

$x : 2^V \to \mathbb{N}$

- x-width (dec. **D**) = $\max_{\text{cut S in } \mathbf{D}}(x(S))$
- x-width (graph **G**) = $\min_{\text{dec. } \mathbf{D} \text{ on } \mathbf{G}}(x\text{-width}(\mathbf{D}))$

mm-width

Split-matching-width (smw)

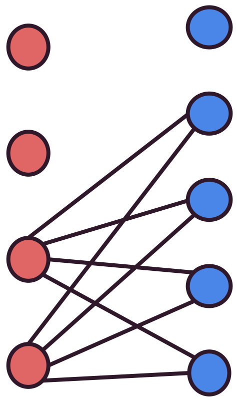
sm-width: ***x***-width where ***x***(S) = ***sm***(S)

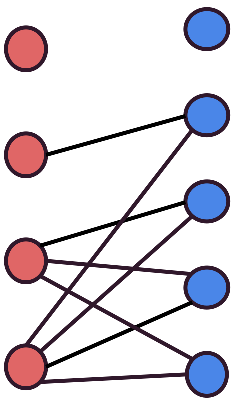
1) ***sm***(S) = 1 If (S, V\S) is a *split*,
2) ***sm***(S) = mm(S) Otherwise

Split-matching-width (smw)

sm-width: ***x***-width where ***x***(S) = ***sm***(S)

1) ***sm***(S) = 1 If (S, V\S) is a *split*,
2) ***sm***(S) = mm(S) Otherwise

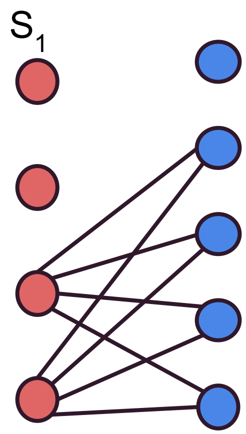




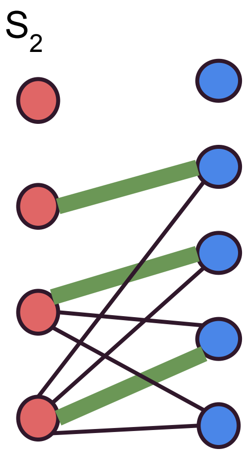
Split-matching-width (smw)

sm-width: ***x***-width where ***x***(S) = ***sm***(S)

1) ***sm***(S) = 1 If (S, V\S) is a *split*,
2) ***sm***(S) = mm(S) Otherwise



sm(S_1) = 1



sm(S_2) = mm(S_2) = 3

Branch decompositions (over V(G))

Thm (Oum, Seymour '06):

If cut-func. is symmetric and submodular, then we can 3-approximate optimal dec. in $2^{O(k)}$-time.

Branch decompositions (over V(G))

Thm (Oum, Seymour '06):

If cut-func. is symmetric and submodular, then we can 3-approximate optimal dec. in $2^{O(k)}$-time.

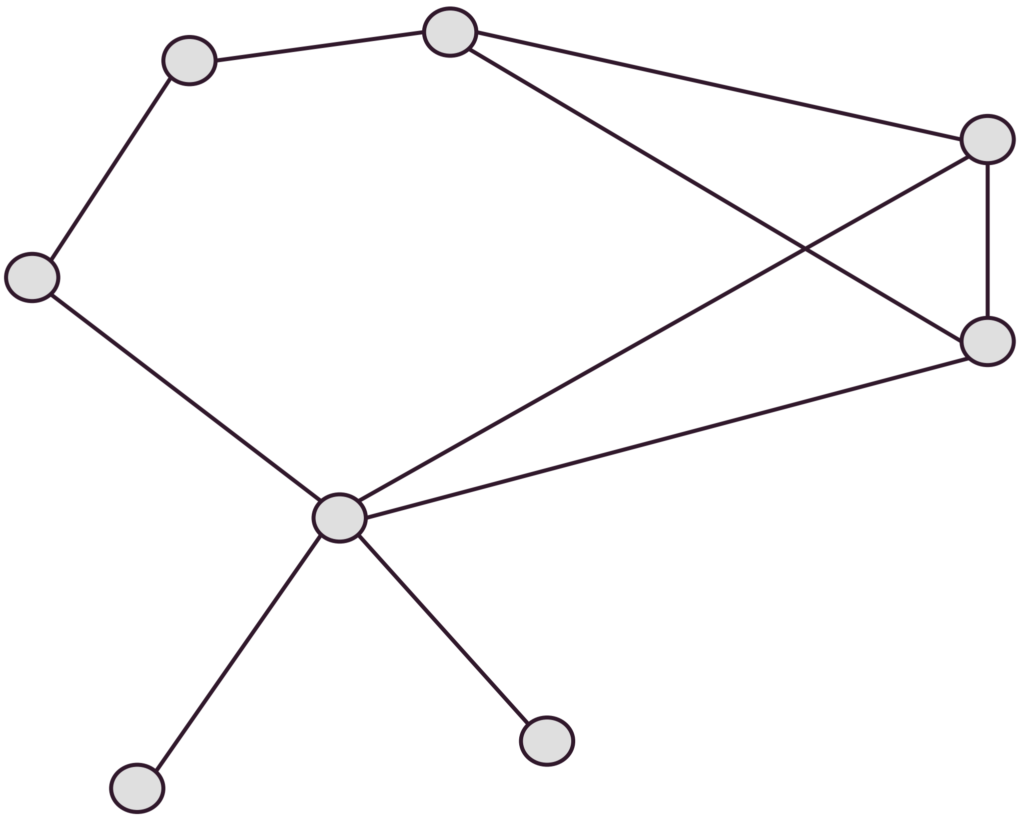
mm is symmetric and submodular

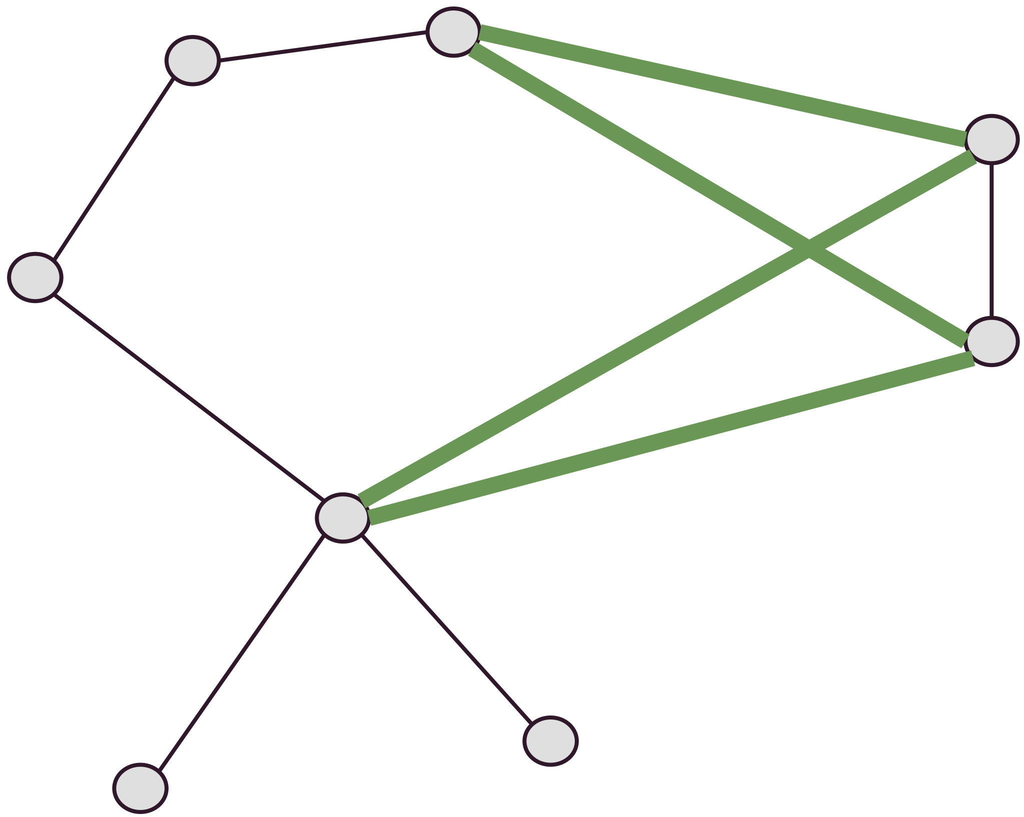
sm is not submodular

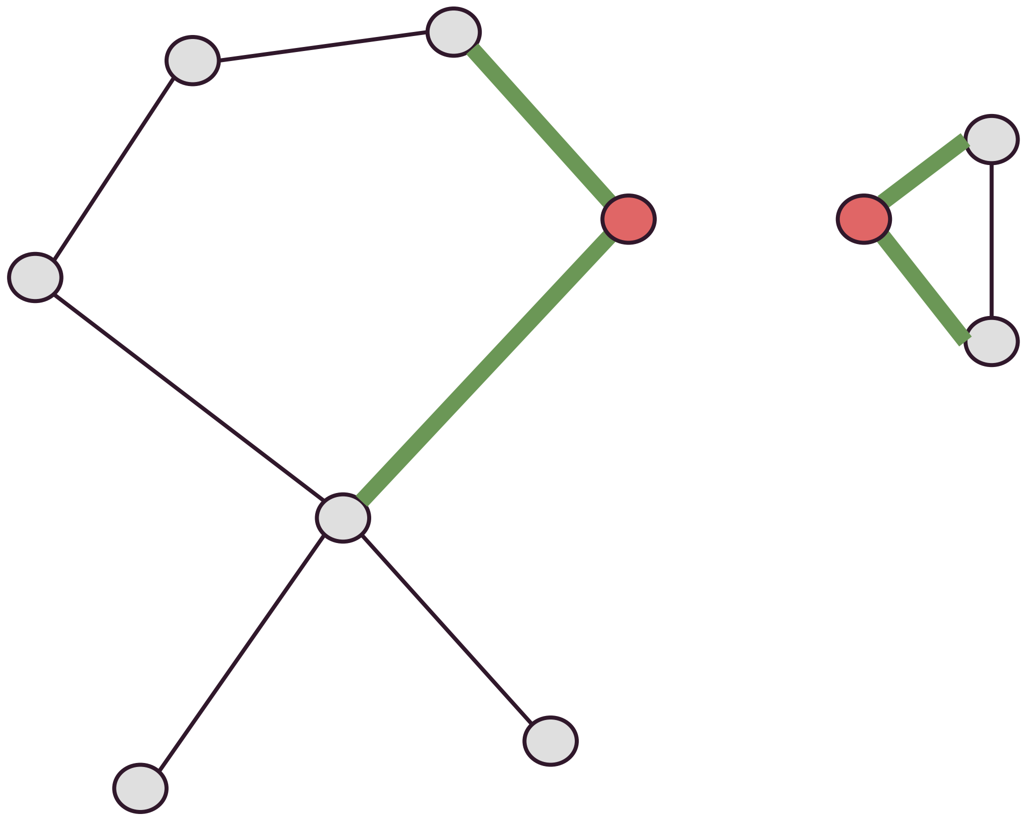
Previous approximation

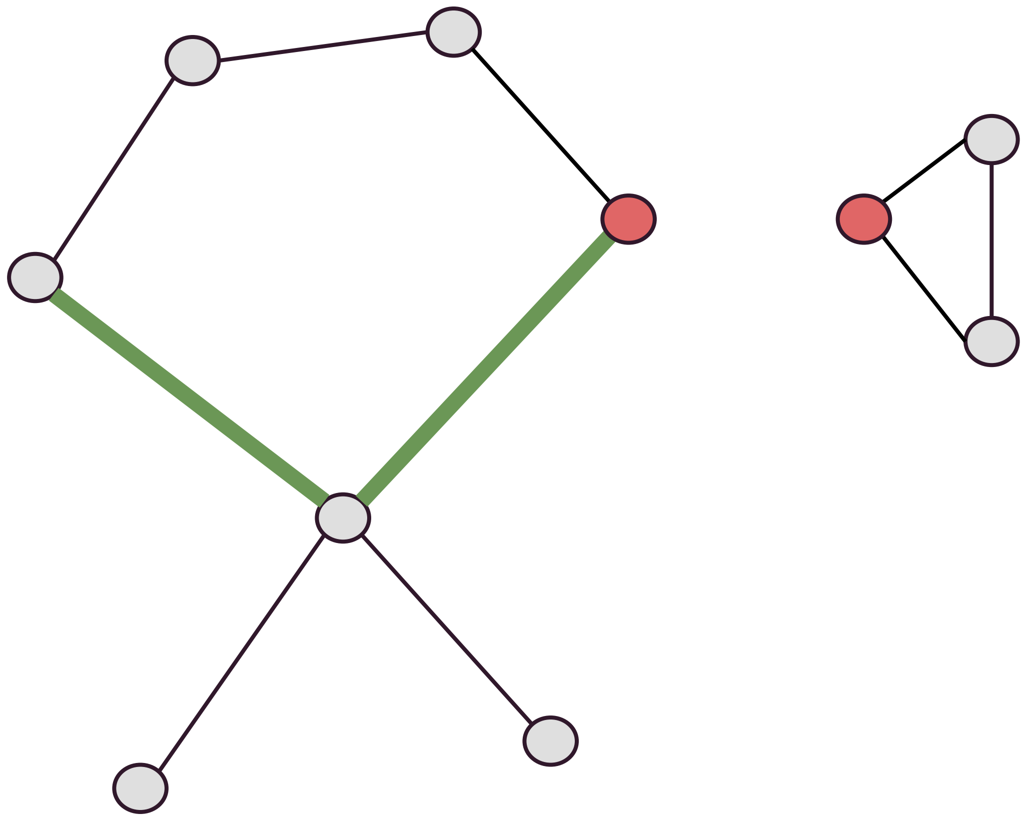
Previous approximation for ***smw***, using approx. of ***mm-w*** and **split decomposition tree**

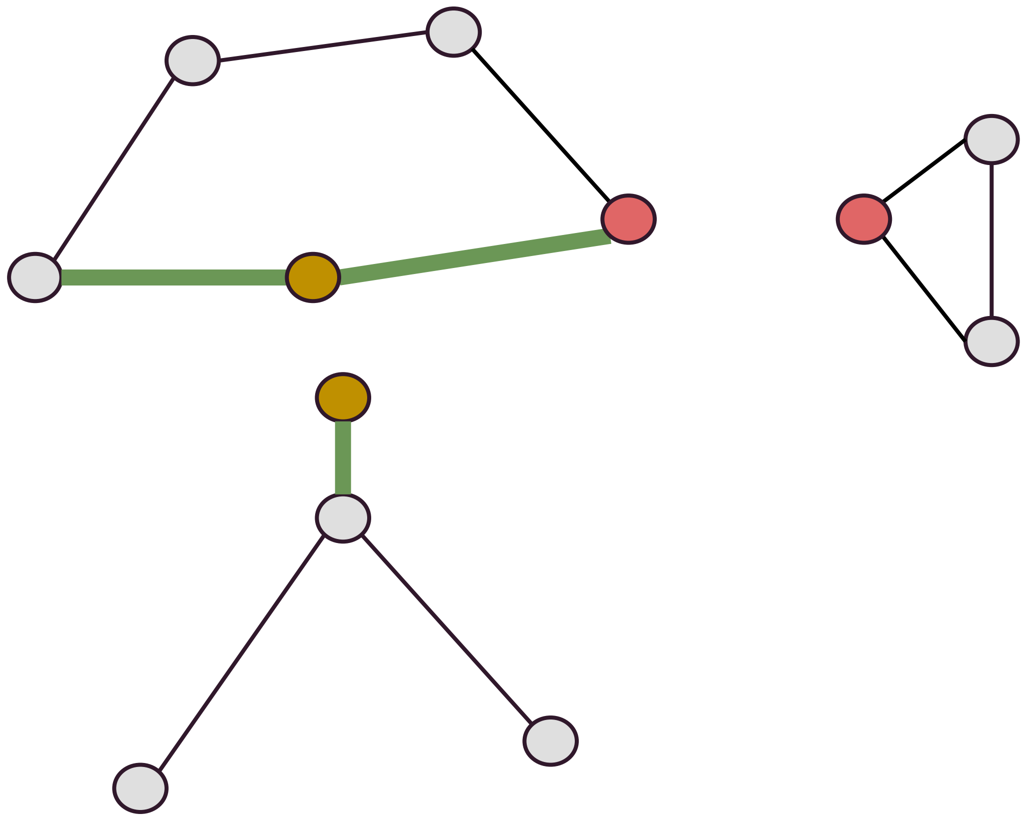
(resulted in decomp. of sm-width $O(smw(G)^2)$)

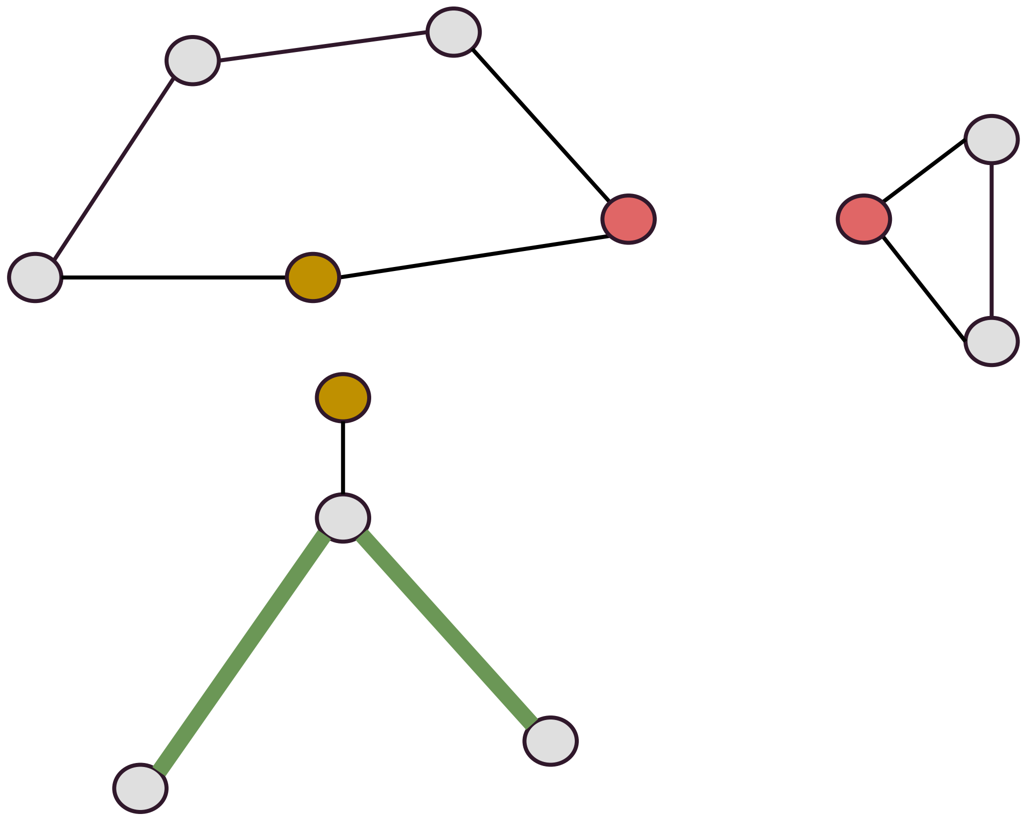


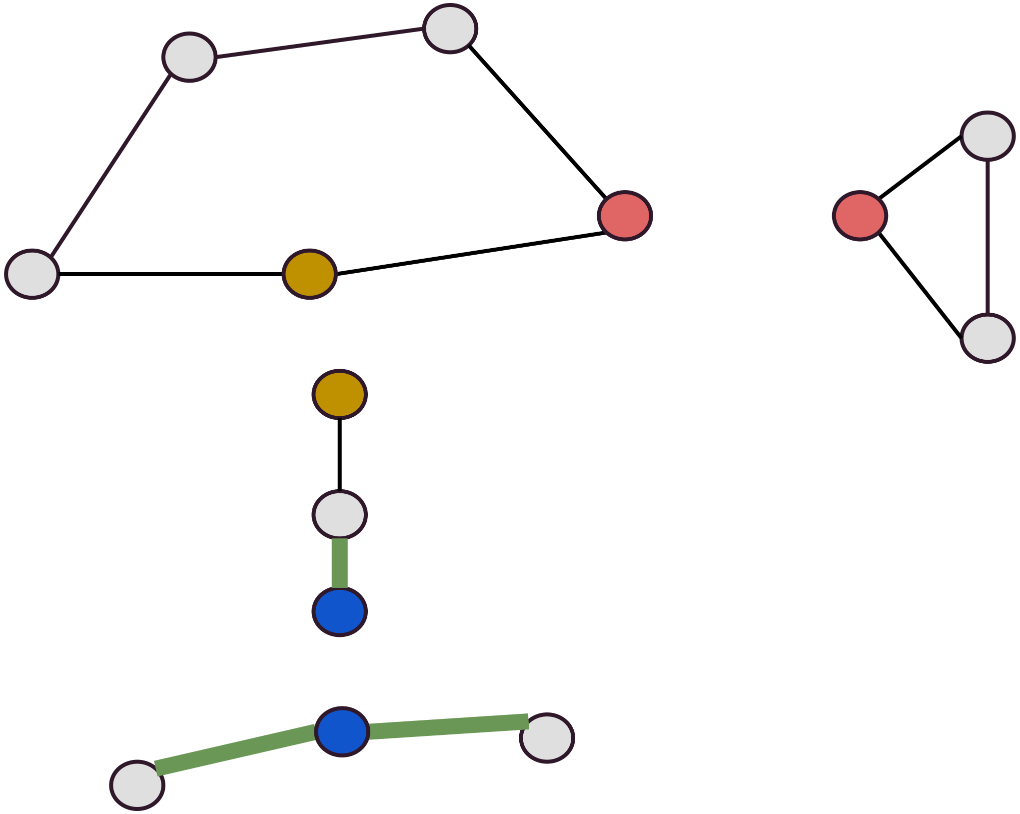


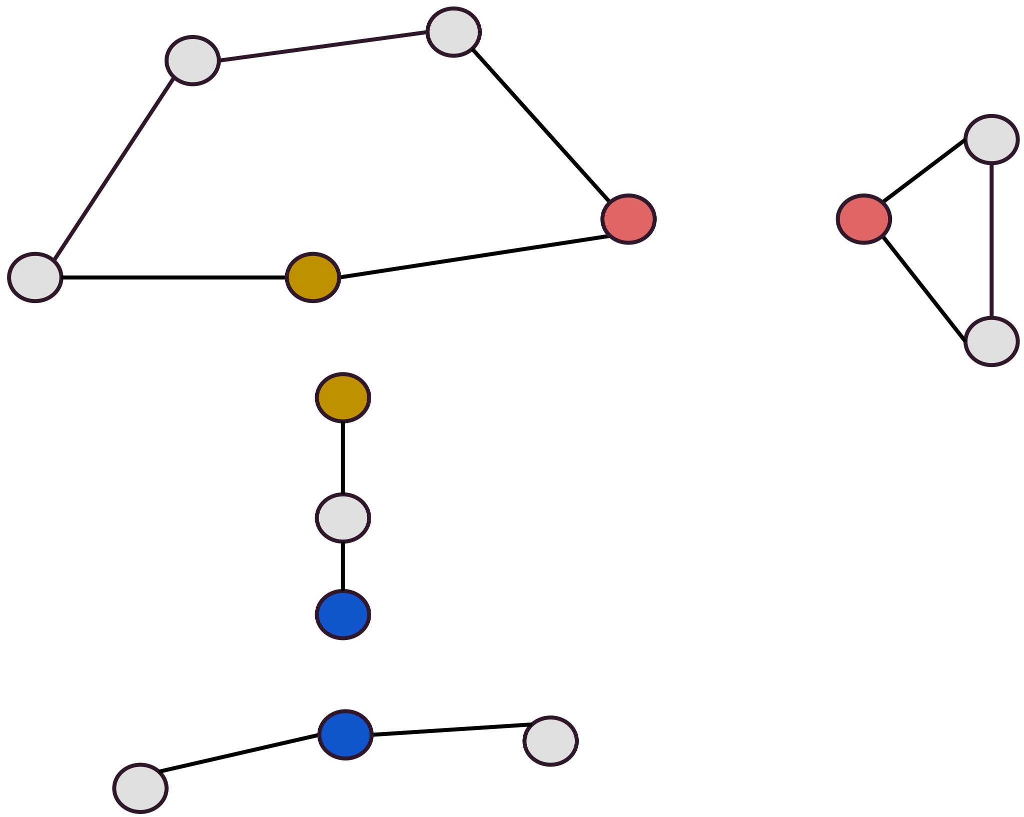


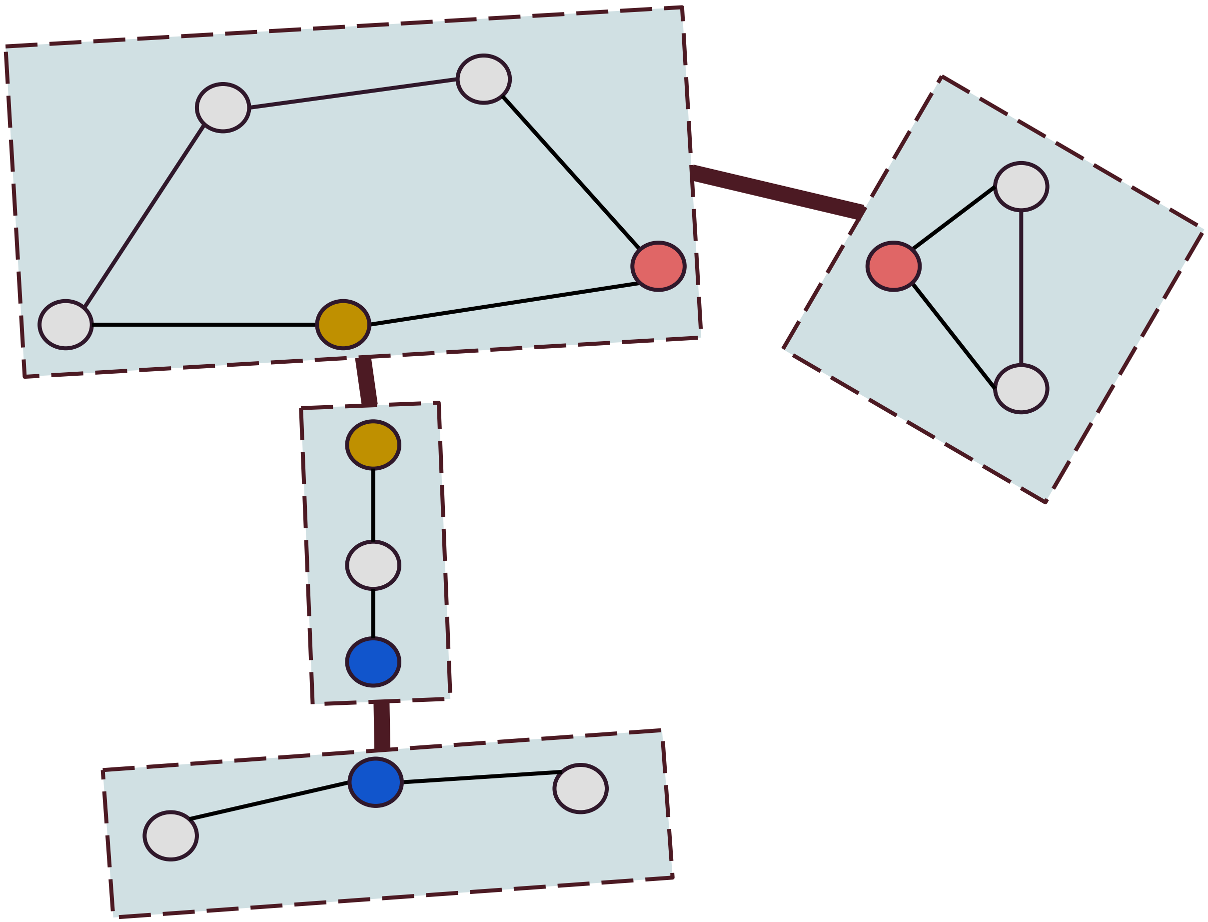


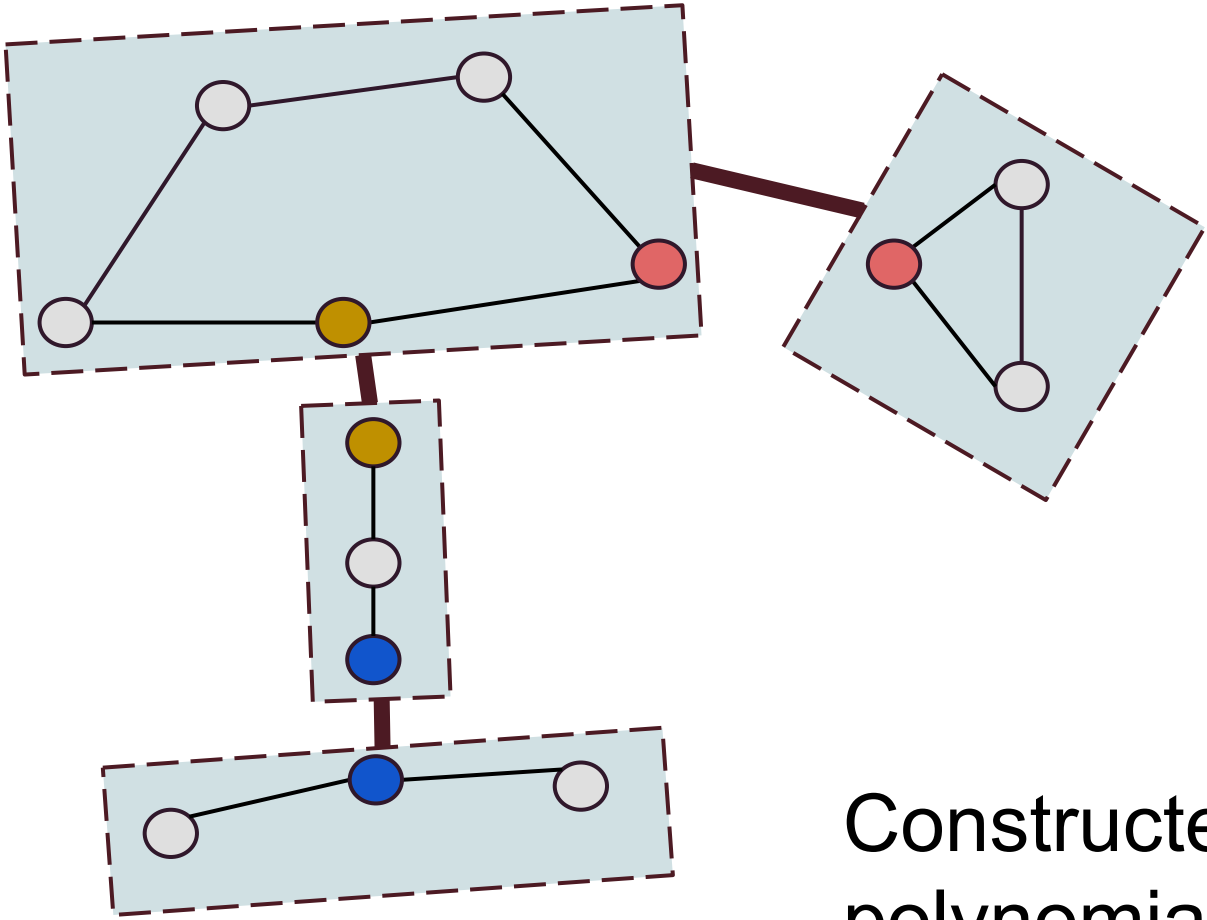




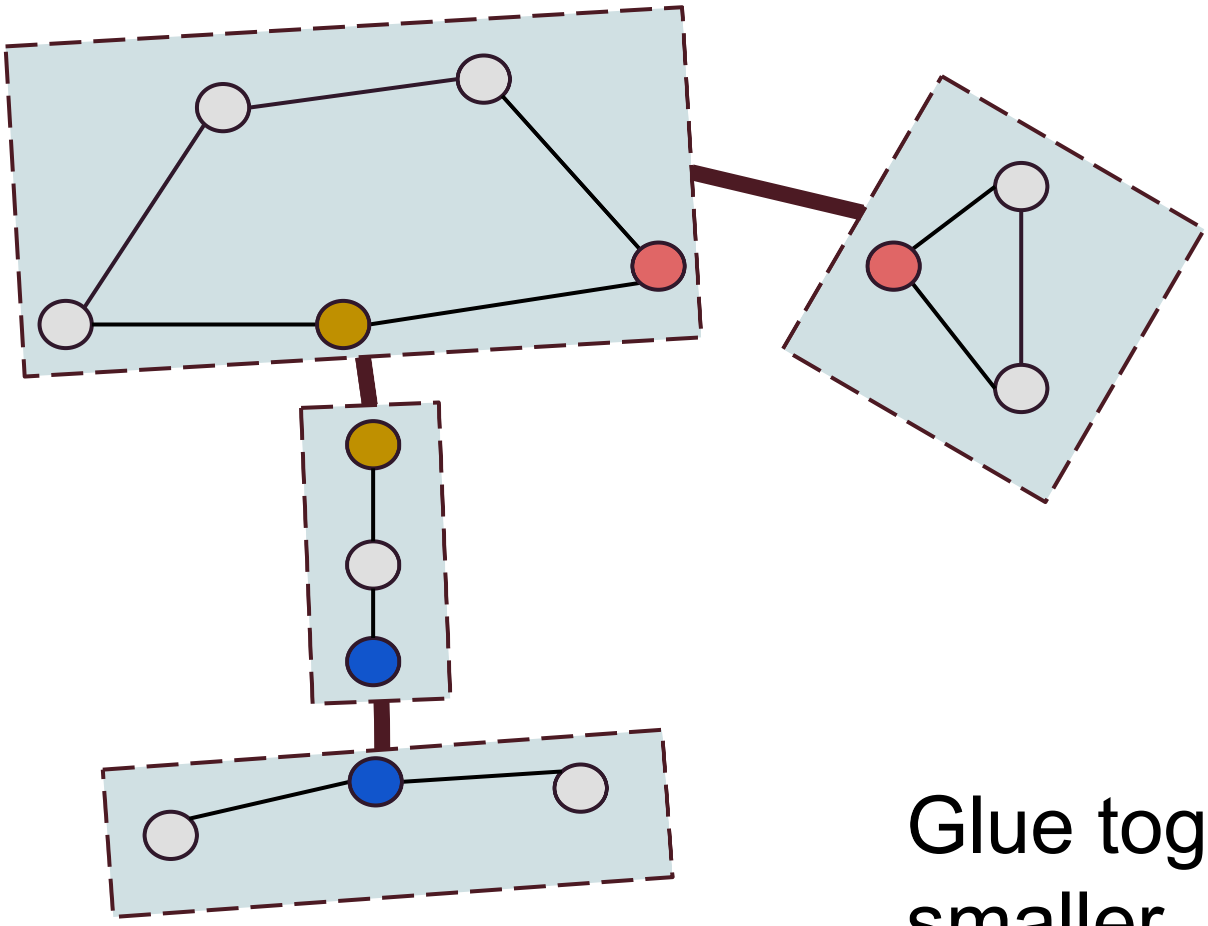




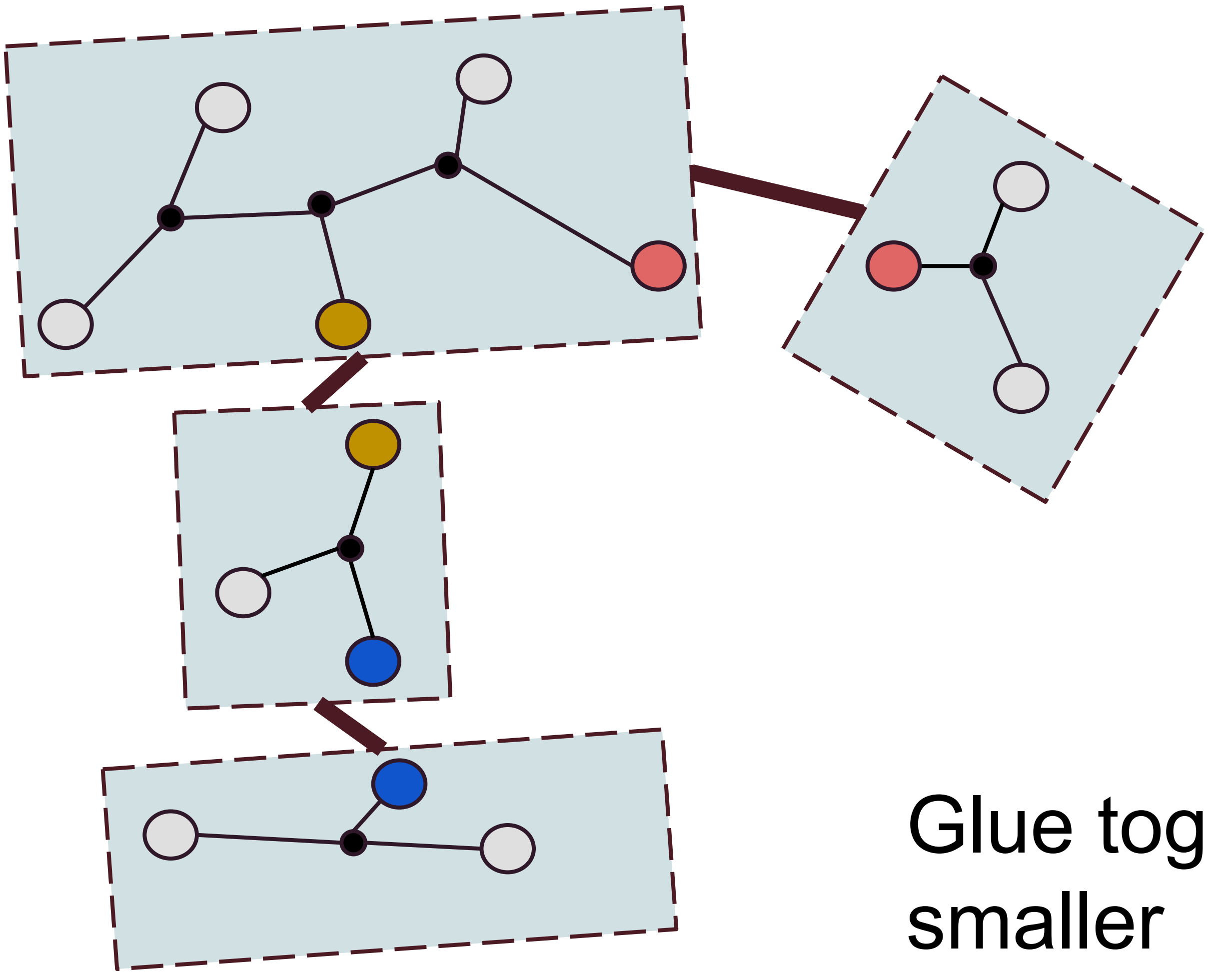




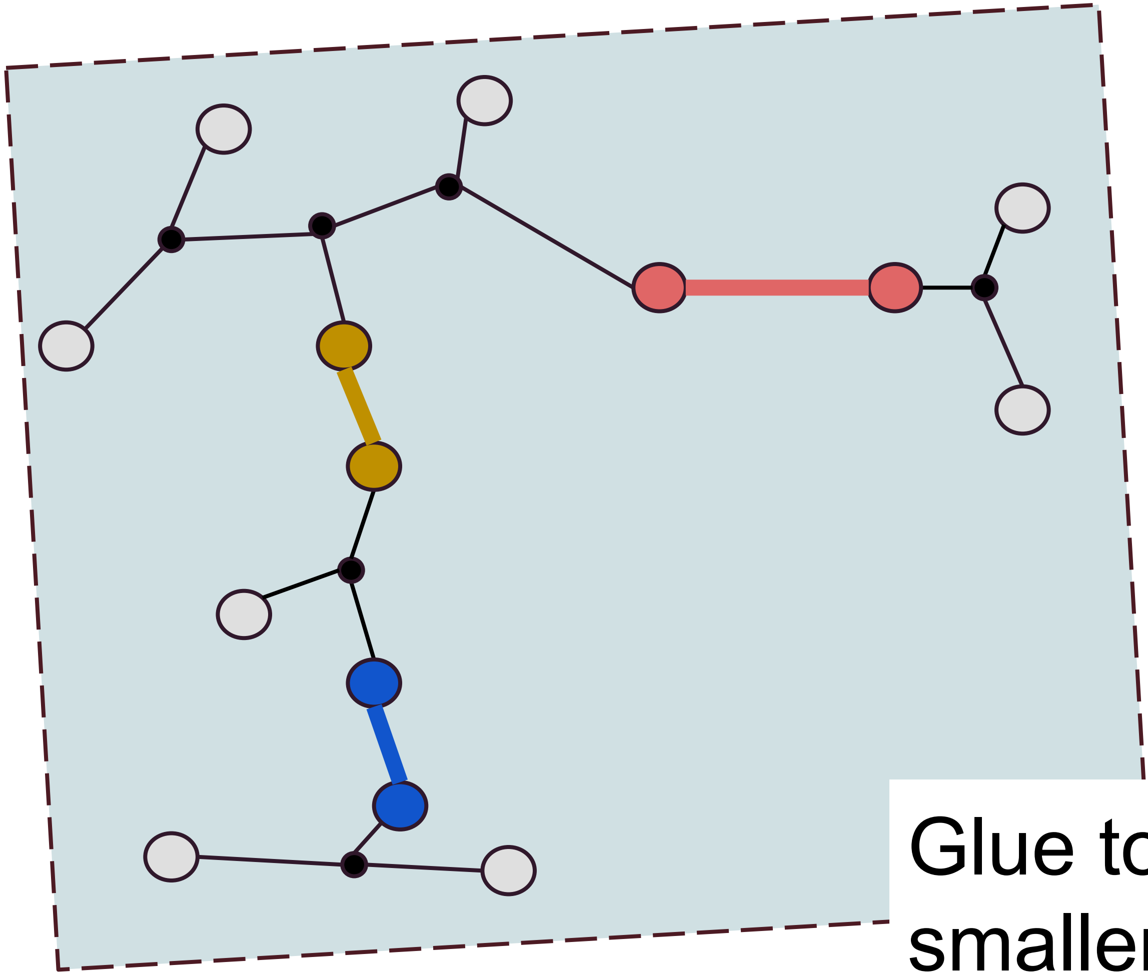
Constructed in
polynomial time
Cunningham'82



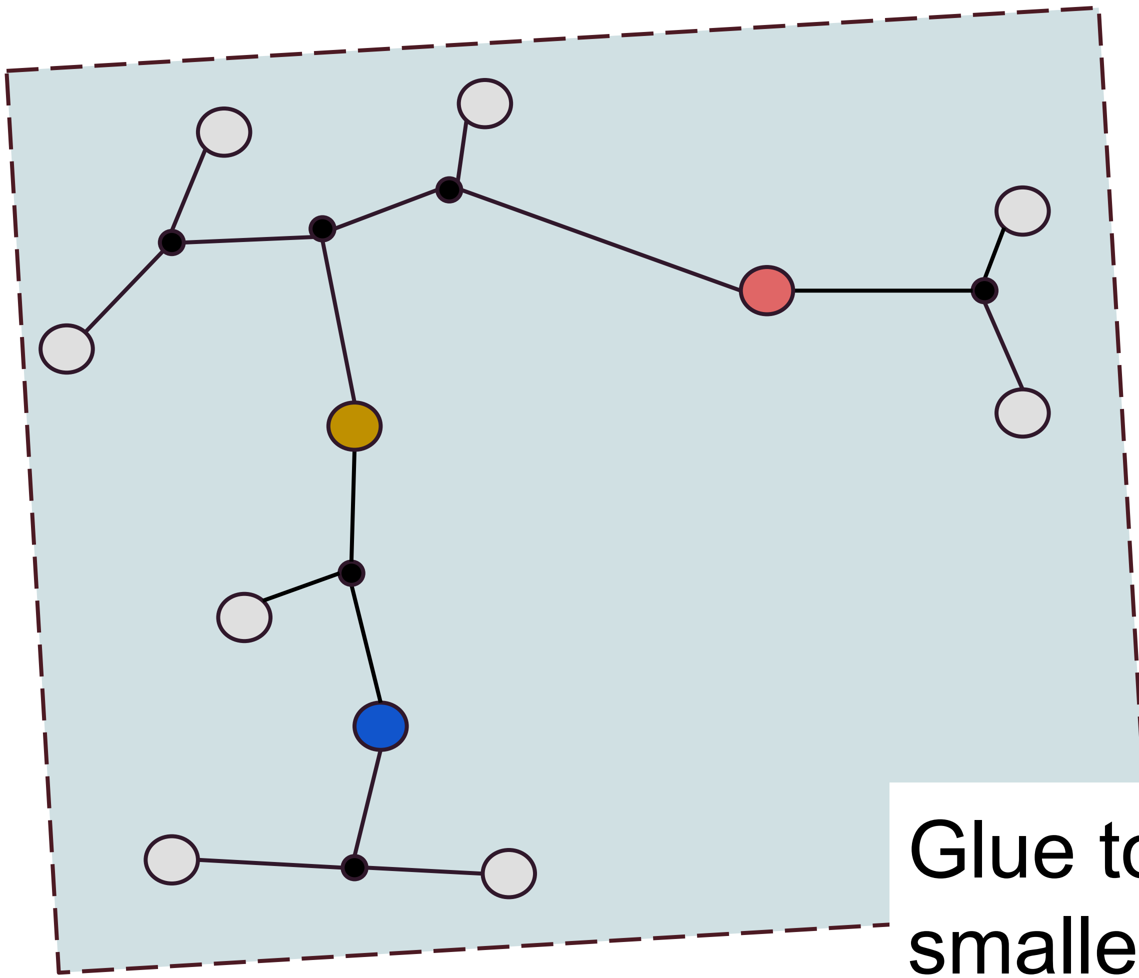
Glue together smaller decompositions



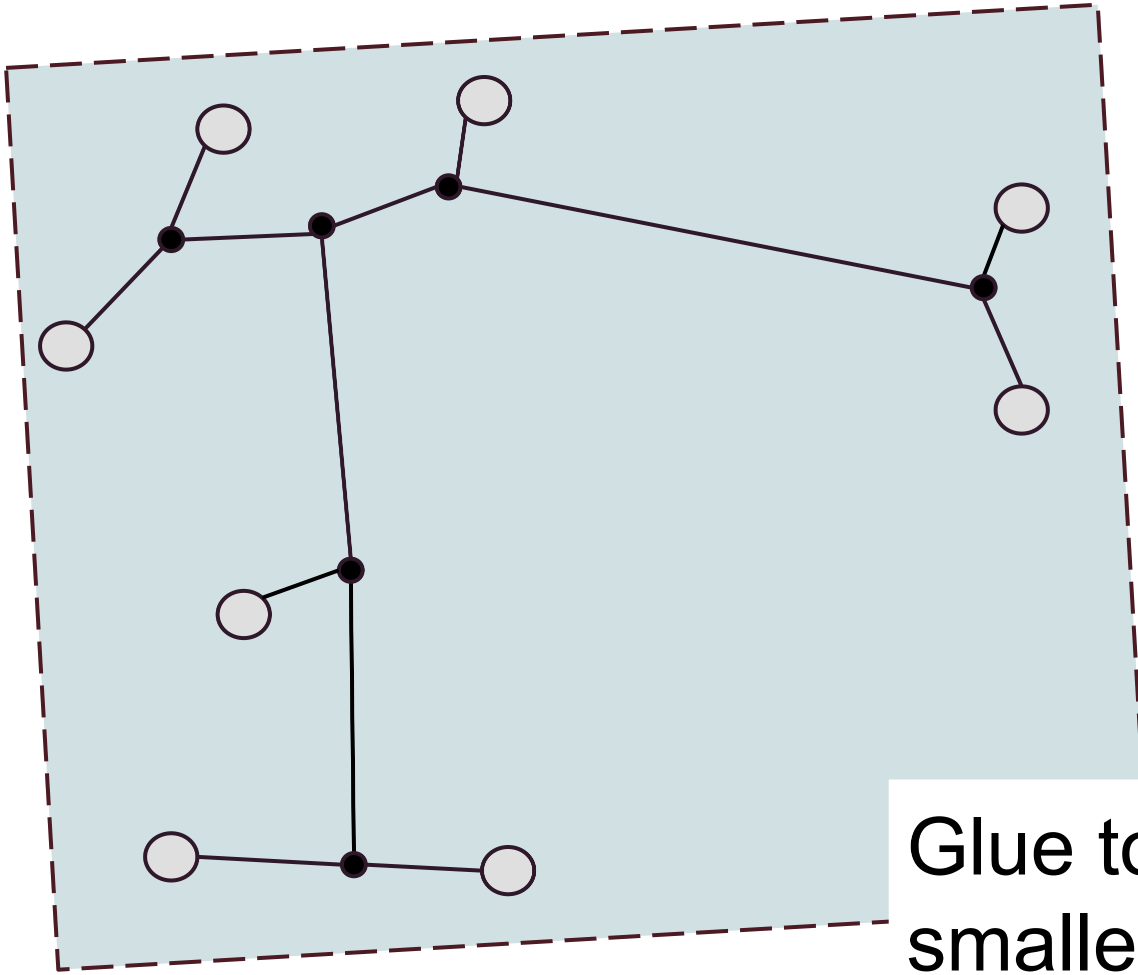
Glue together
smaller
decompositions



Glue together smaller decompositions



Glue together smaller decompositions

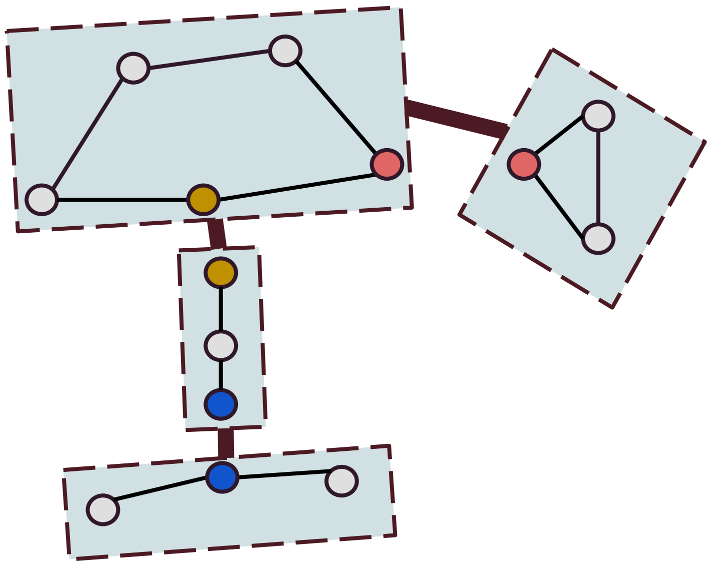


Glue together smaller decompositions

Lifted *x-width*

How to find good decompositions of prime graphs?

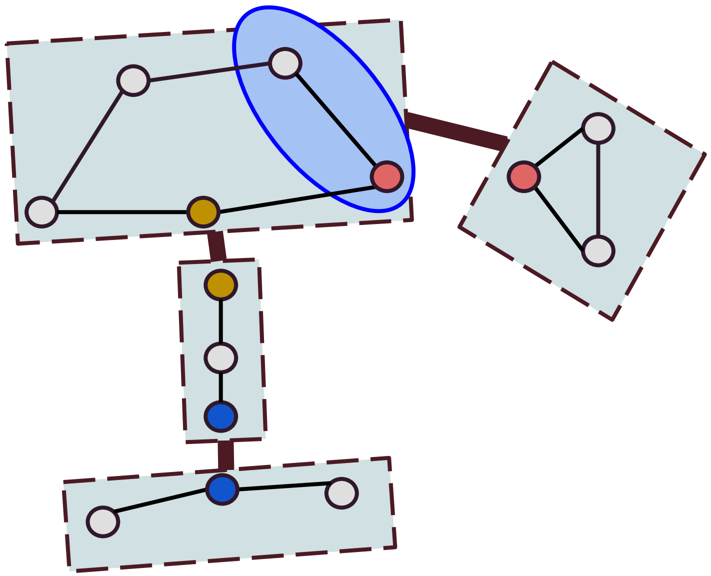
Lifted x-width



Lifted *x-width*

How to find good decompositions of prime graphs?

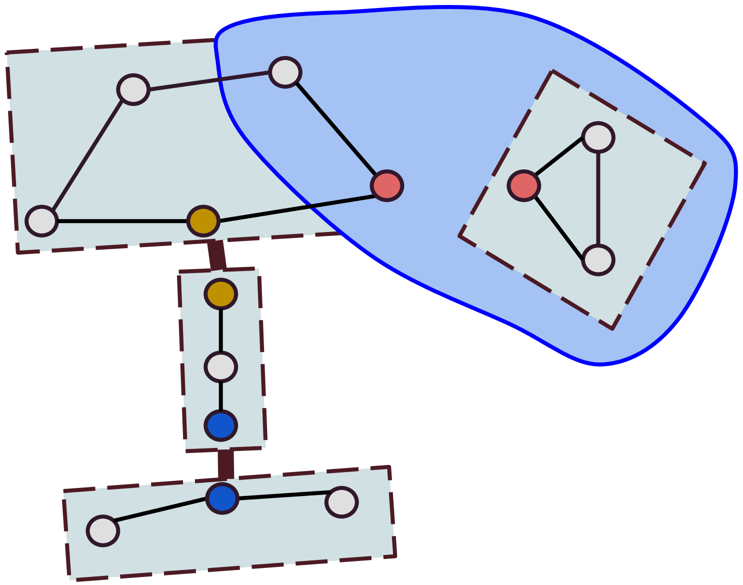
Lifted x-width



Lifted *x-width*

How to find good decompositions of prime graphs?

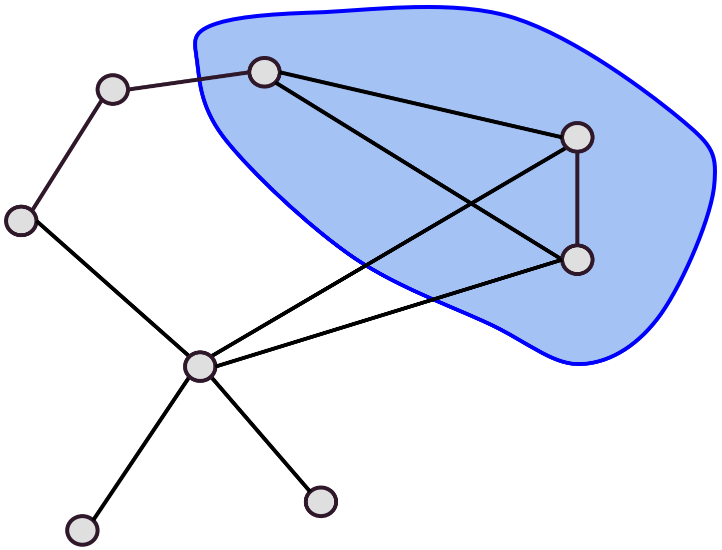
Lifted x-width



Lifted *x-width*

How to find good decompositions of prime graphs?

Lifted x-width



Lifted *x-width*

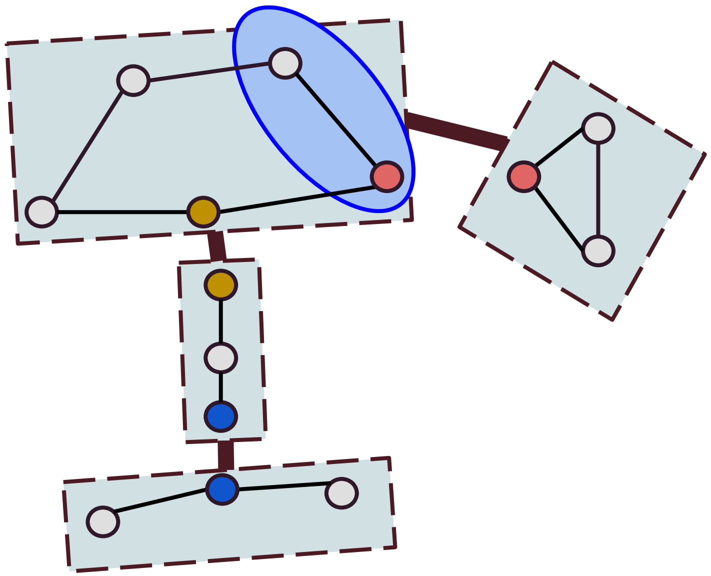
How to find good decompositions of prime graphs?

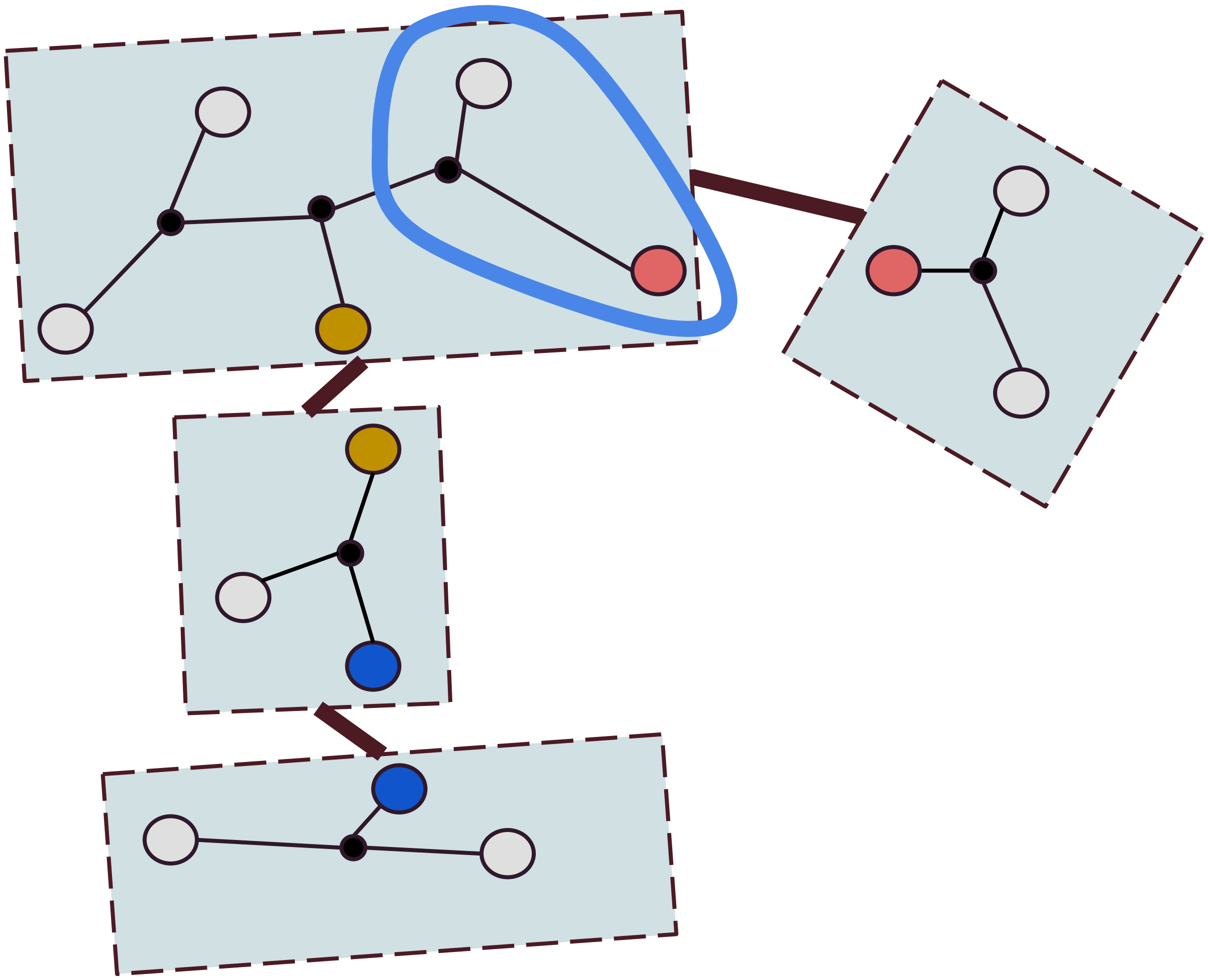
Lifted x-width

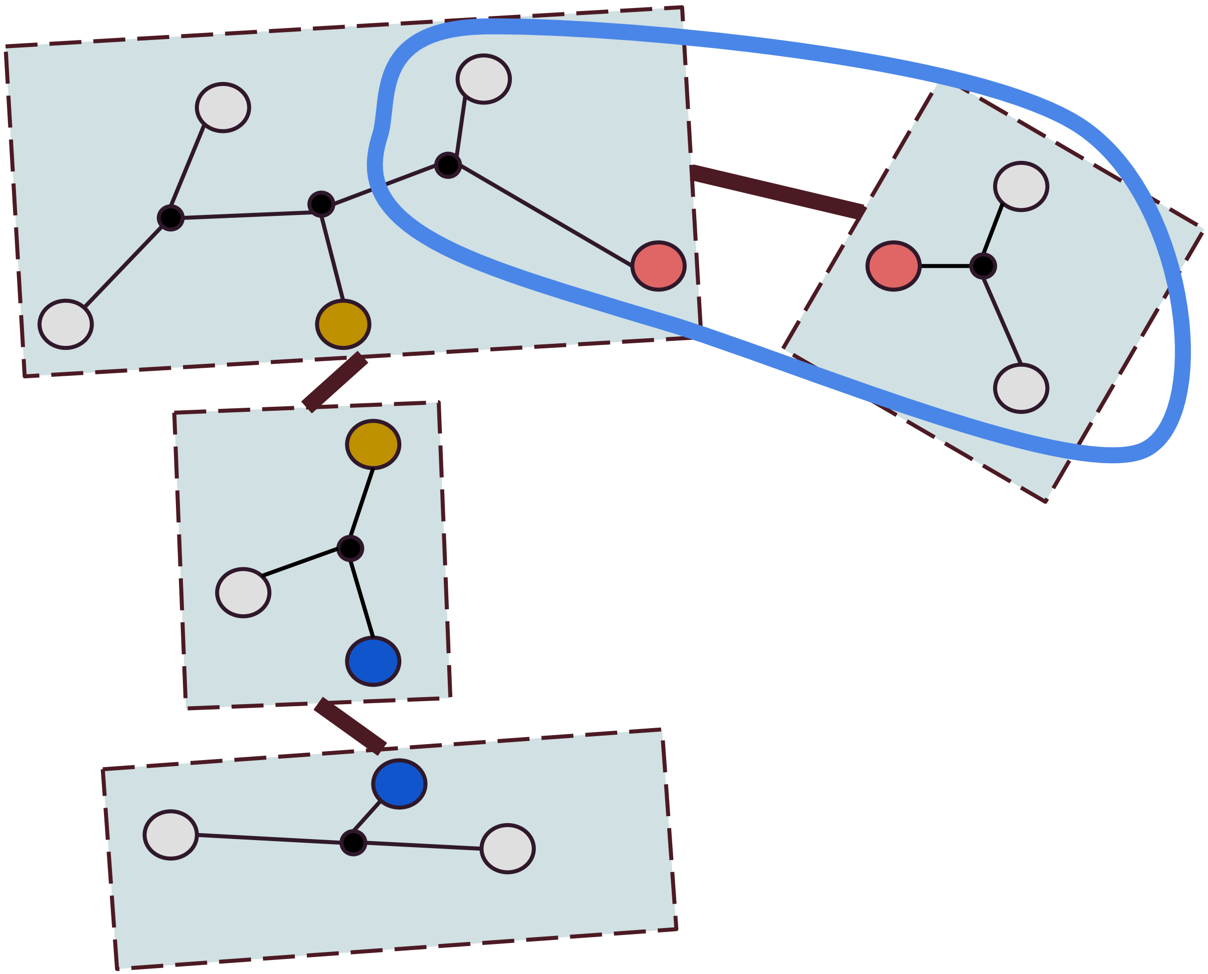
This example:

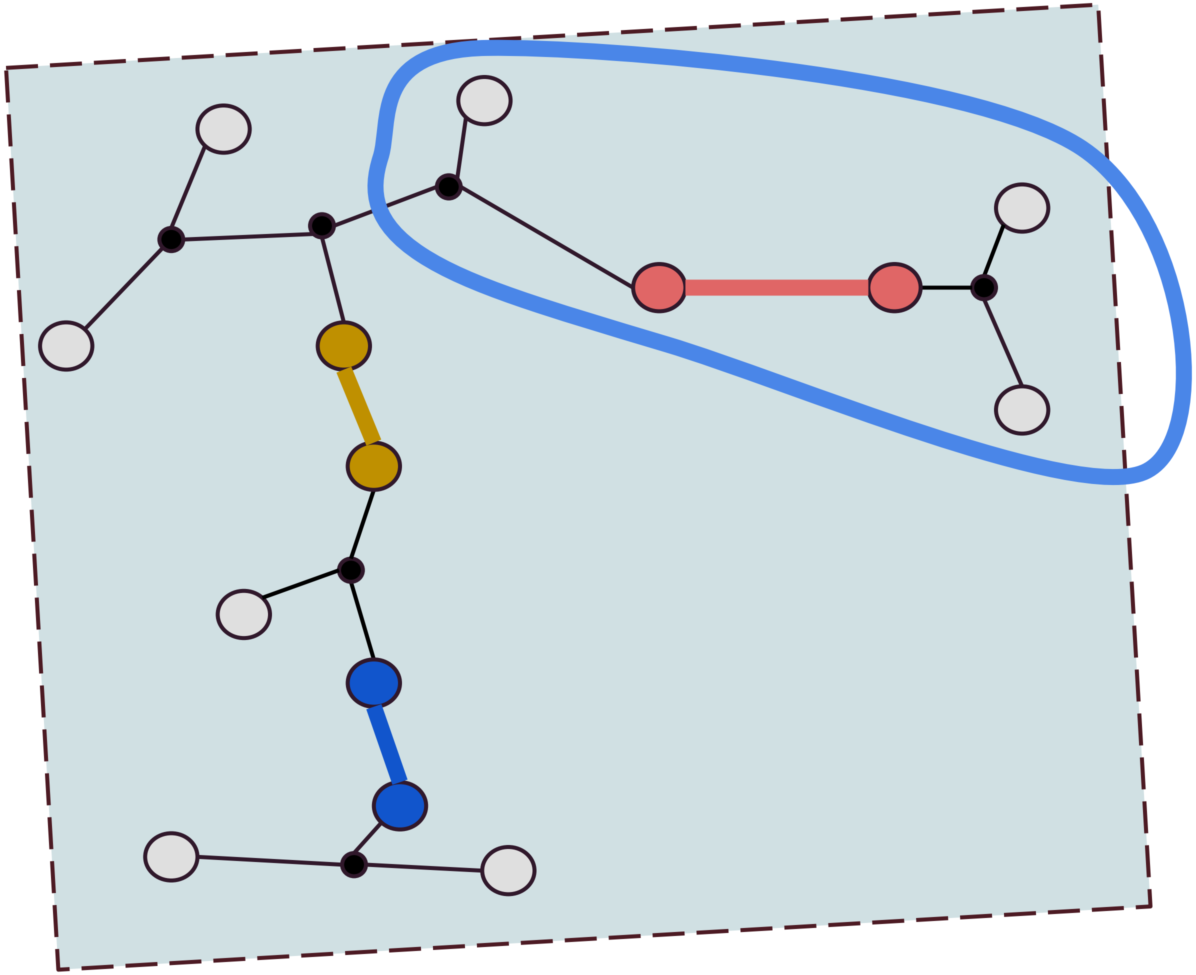
lifted carving width = 3

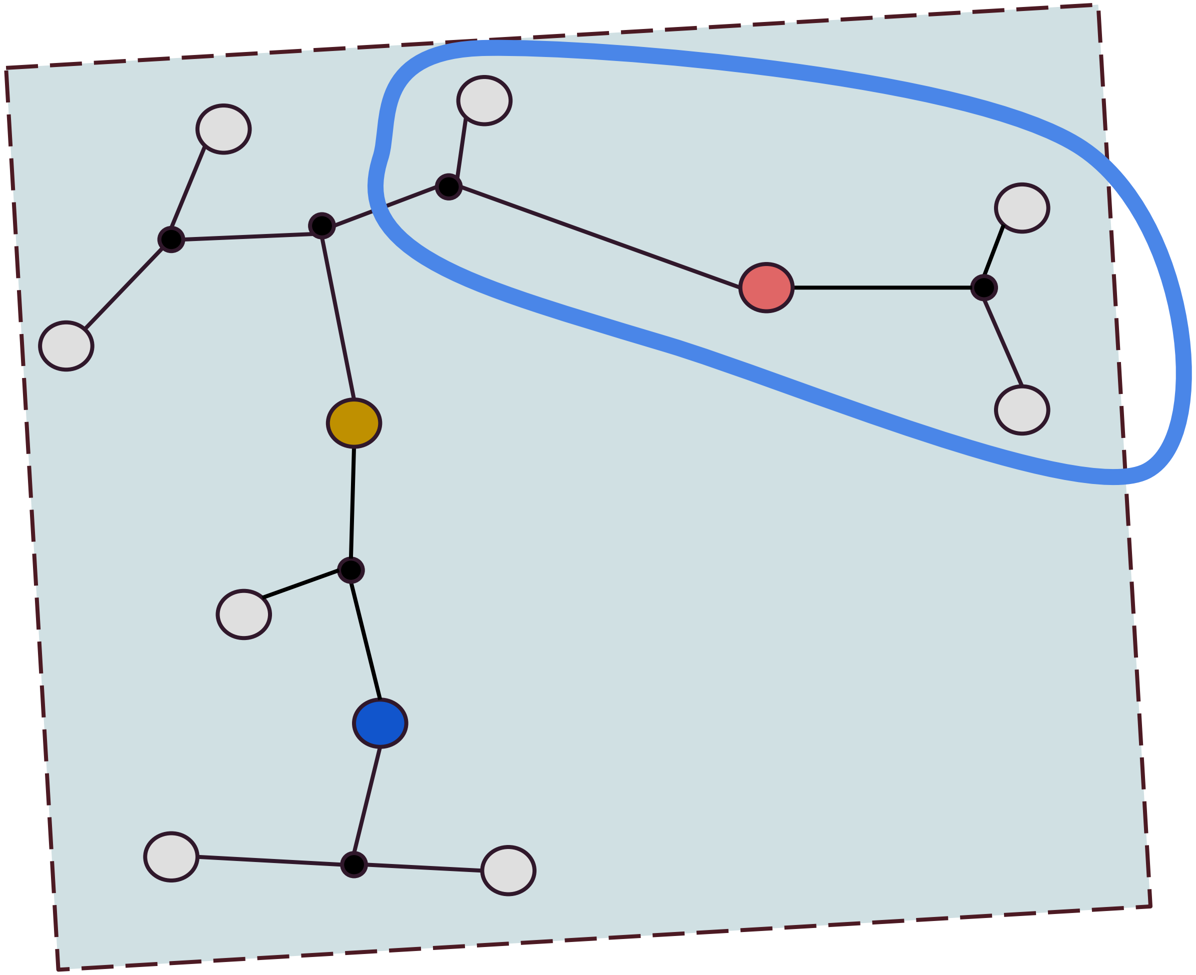
carving width = 2

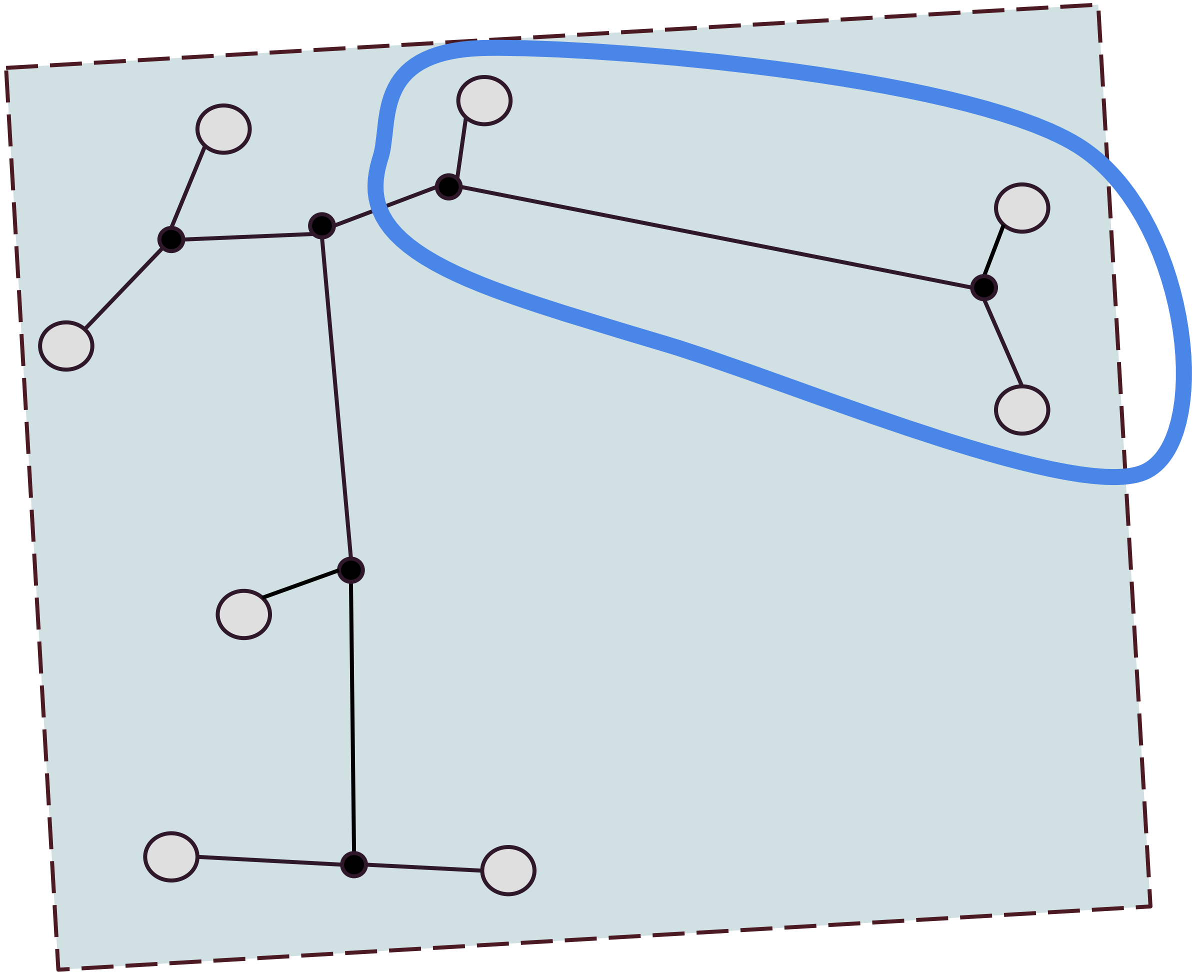












Lemma: For any primal graph G' in split decomposition of G, lifted-*smw*(G') ≤ 3**smw*(G)

Lemma: For any primal graph G' in split decomposition of G, lifted-*smw*(G') $\leq 3 * smw$(G)

Lemma: For primal graphs lifted-*smw* can be 6-approximated in $2^{O(\text{lifted-}smw)}$-time

Lemma: For any primal graph G’ in split decomposition of G, lifted-*smw*(G’) $\leq$ 3**smw*(G)

Lemma: For primal graphs lifted-*smw* can be 6-approximated in $2^{O(\text{lifted-}smw)}$-time

Theorem: *smw*(G) can be 18-approximated in $2^{O(smw(G))}$-time

Conclusion

- Using lifted-smw we can 18-approximate smw
- Combined with improved DP algo gives $2^{O(smw(G))}$-time Hamiltonian Cycle algorithm
- Combined with previous DP algorithms of S.,Telle'14
  - Edge Dominating Set in $2^{O(smw(G))}$-time
  - Chromatic Number in $smw(G)^{O(smw(G))}$-time
  - Max-Cut in $2^{O(smw(G))}$-time

All optimal under Exponential Time Hypothesis.

Thank you