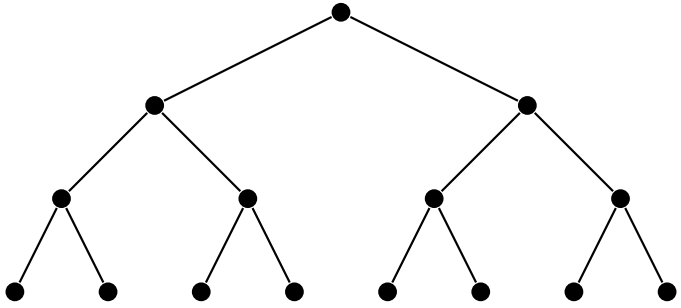
Advanced Treewidth DPs

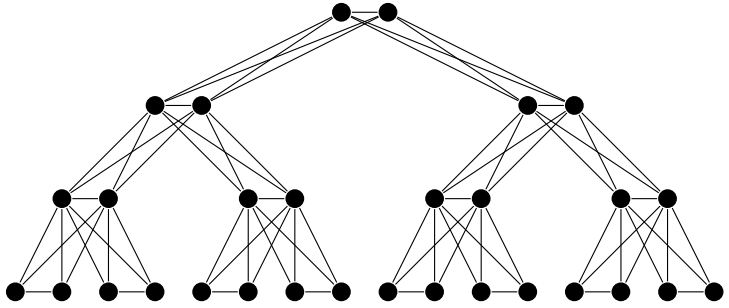
Marcin Pilipczuk

University of Bergen
University of Warsaw
University of Warwick

21st August 2014, Bedlewo

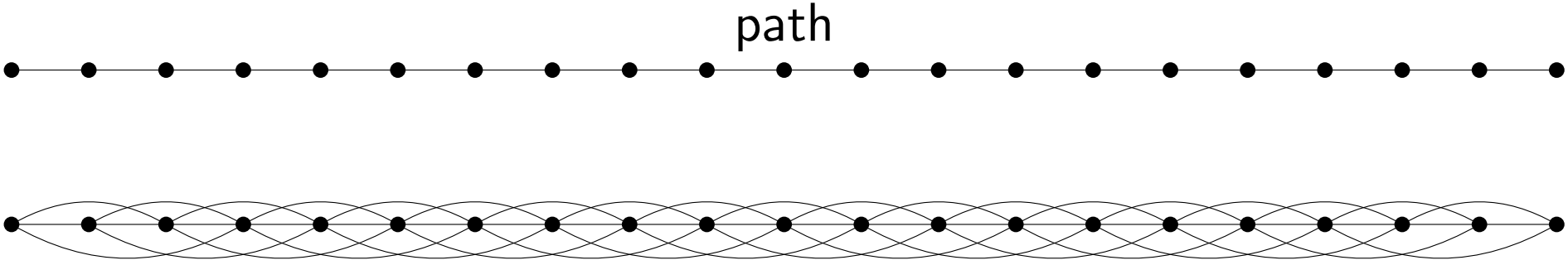


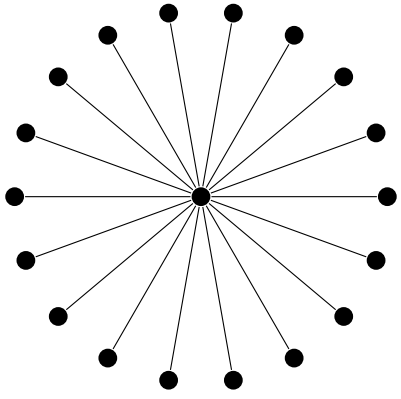
tree



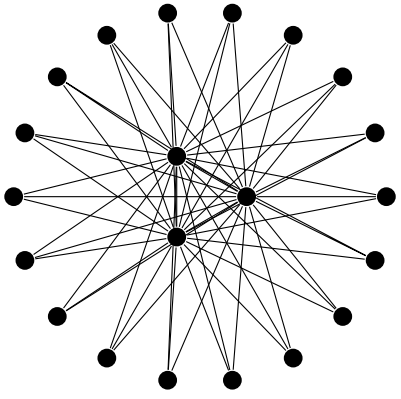
treelike

Treewidth/pathwidth


path
pathlike



tree

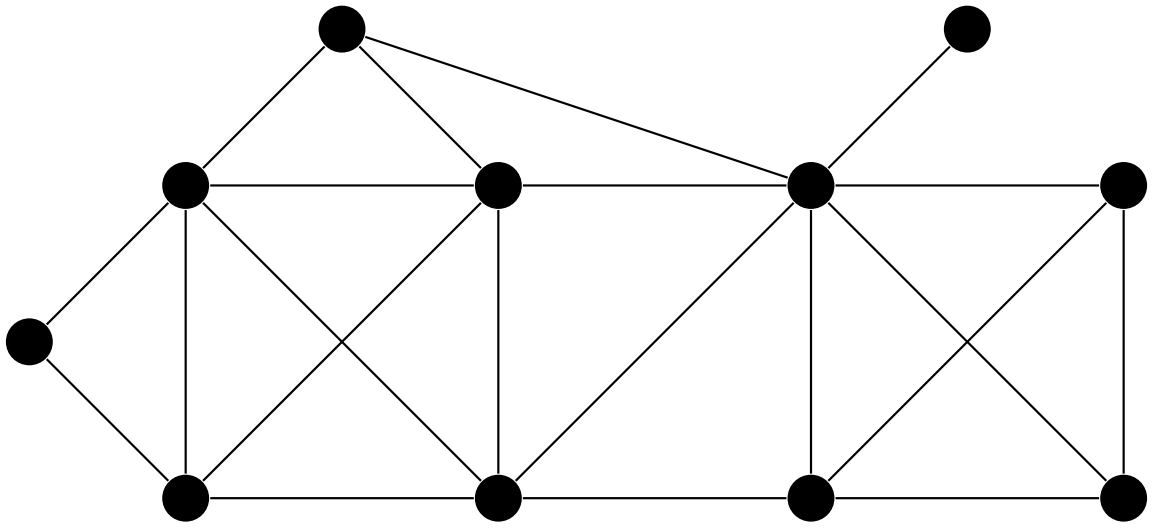


treelike

1. Join nodes for Dominating Set
2. Connectivity problems
    1. Naive algorithm for Hamiltonian Cycle
    2. Cut and count
    3. Rank based approach

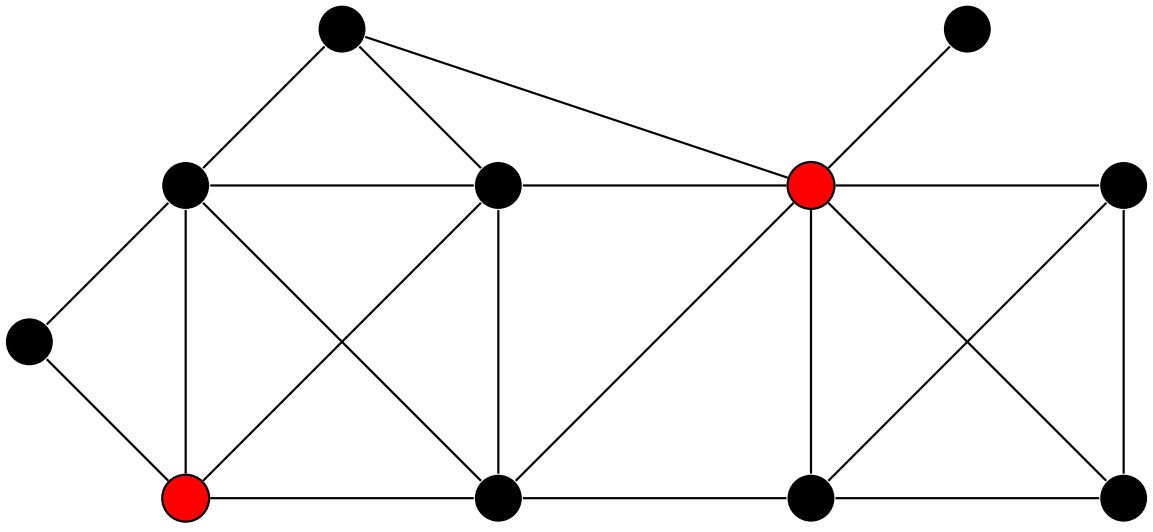
Outline

Dominating Set



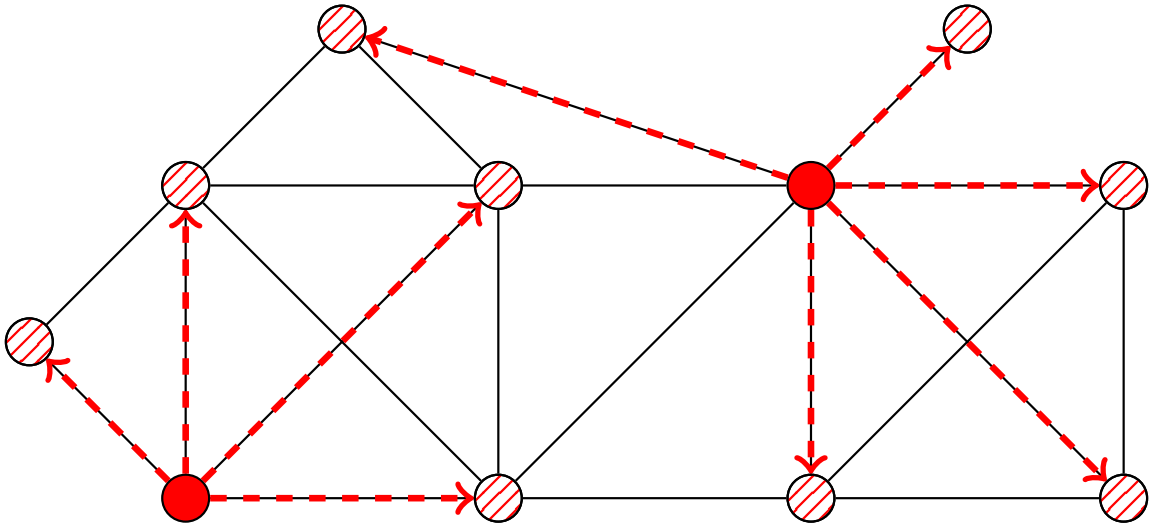
Given a graph G, find minimum $X \subseteq V(G)$ such that $N[X] = V(G)$.

Dominating Set



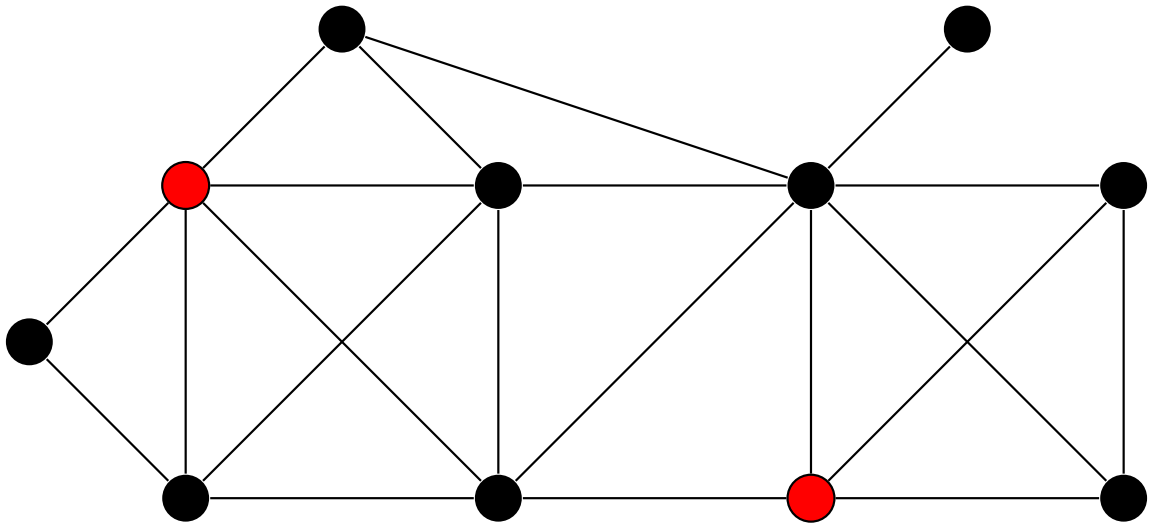
Given a graph G, find minimum $X \subseteq V(G)$ such that $N[X] = V(G)$.

Dominating Set



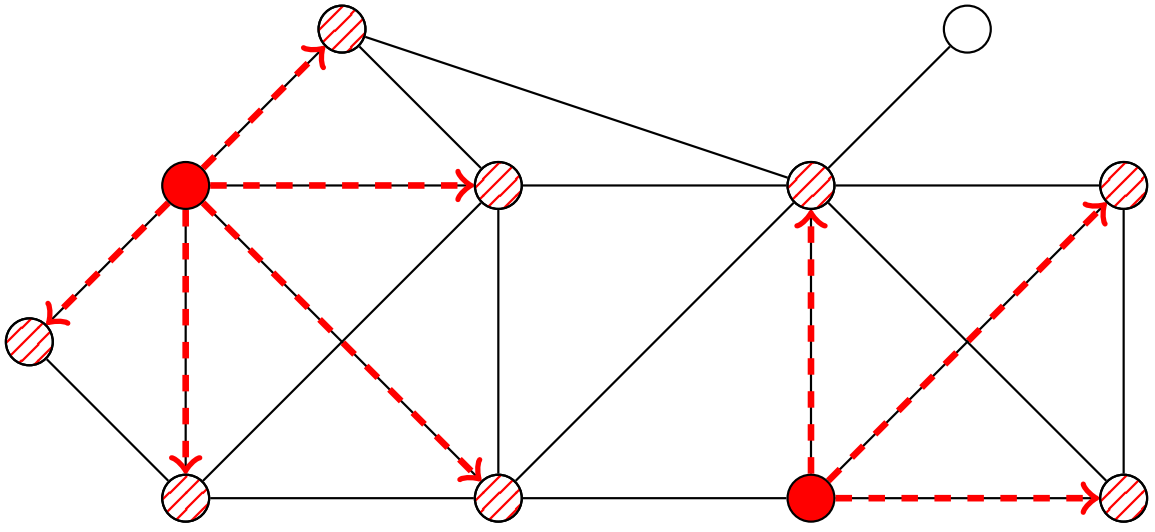
Given a graph G, find minimum $X \subseteq V(G)$ such that $N[X] = V(G)$.

Dominating Set



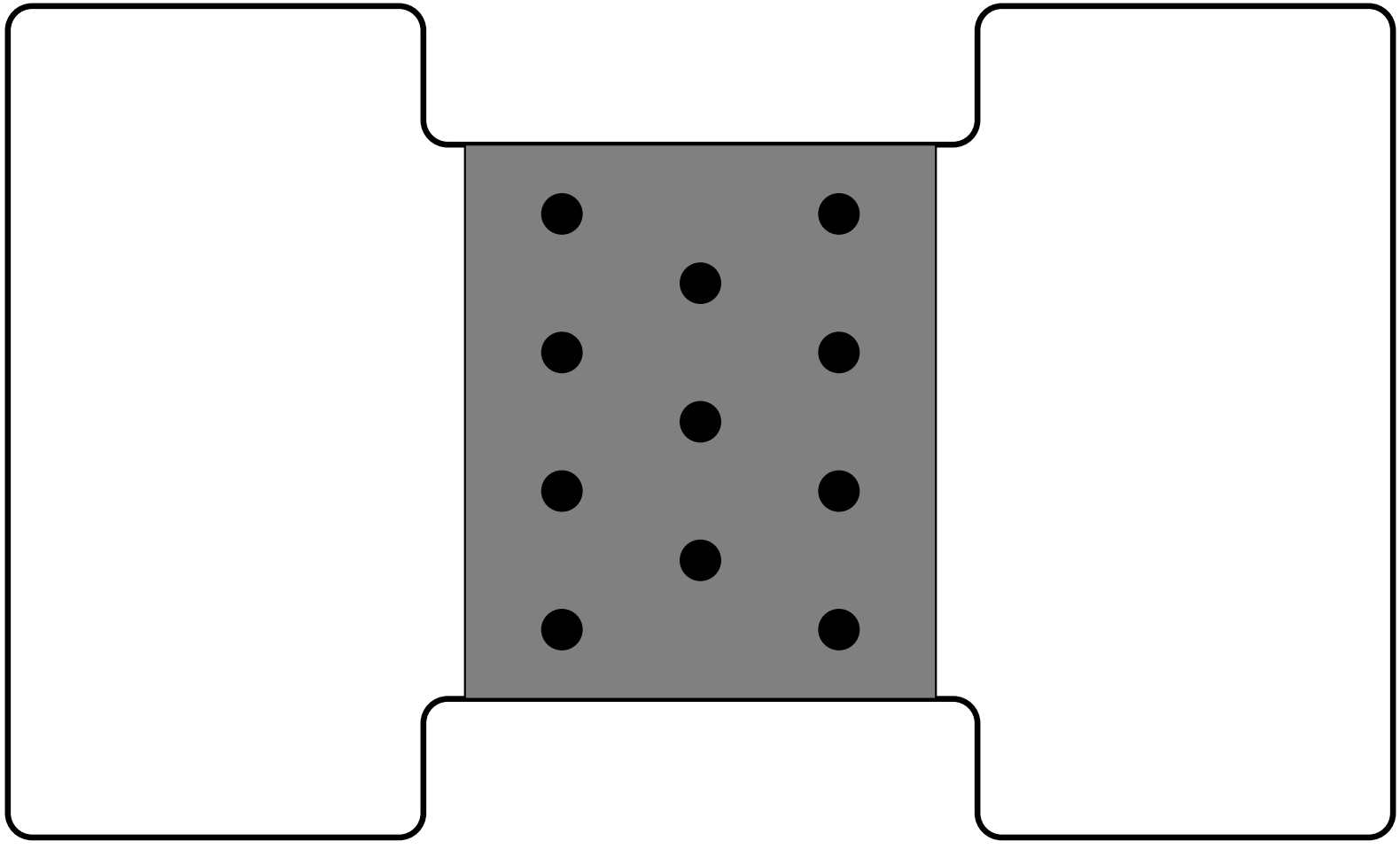
Given a graph G, find minimum $X \subseteq V(G)$ such that $N[X] = V(G)$.

Dominating Set

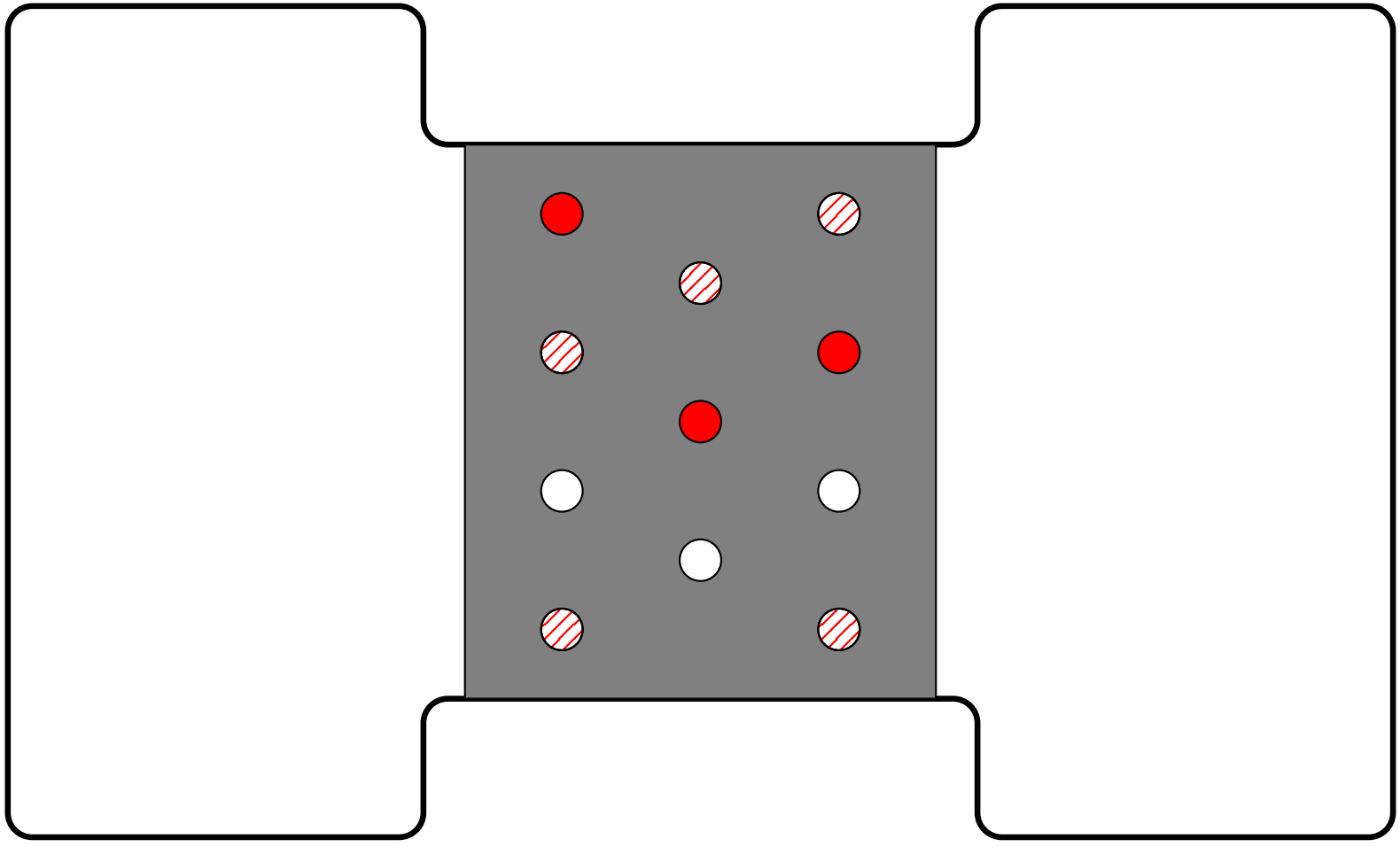


Given a graph G, find minimum $X \subseteq V(G)$ such that $N[X] = V(G)$.

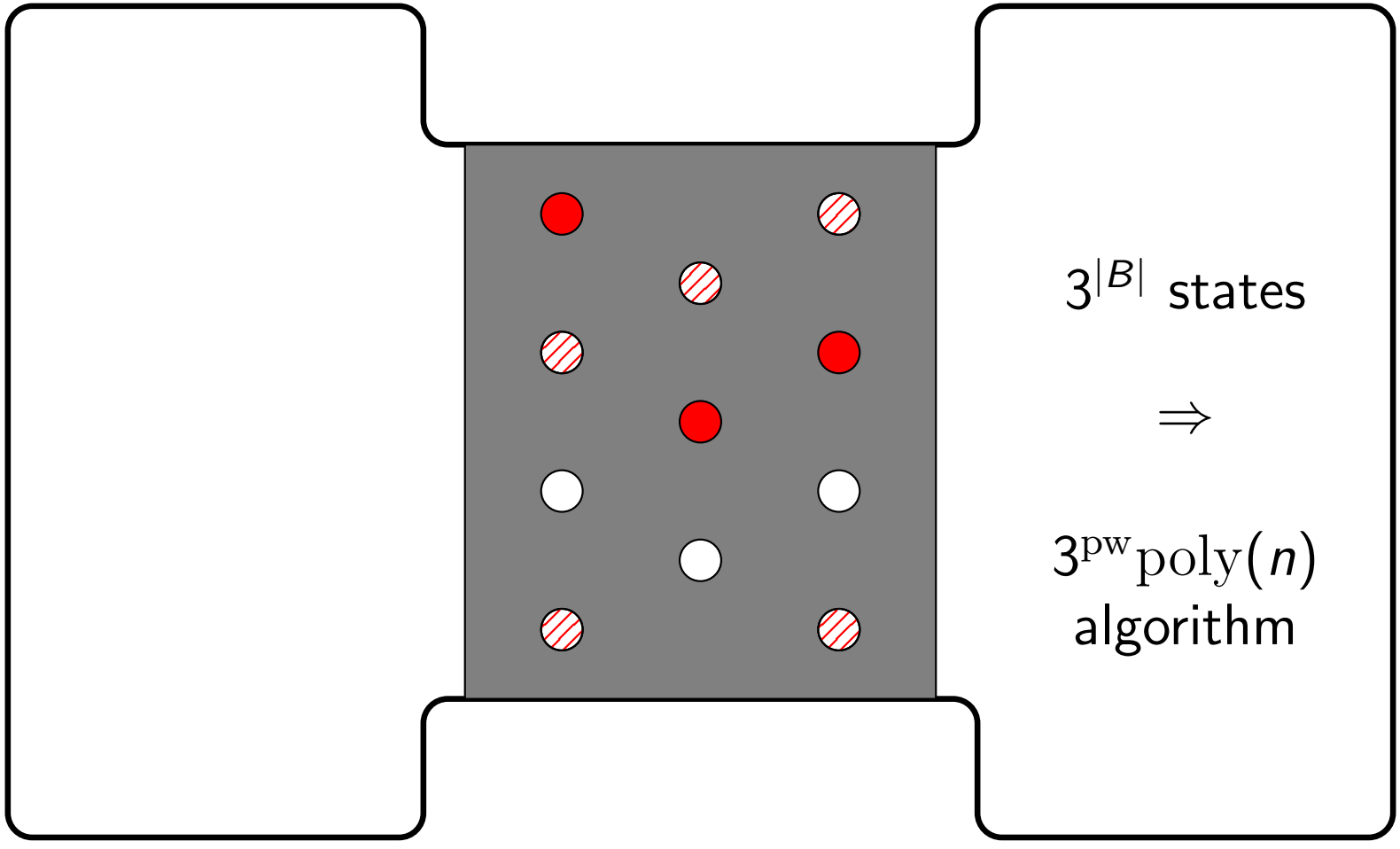
Dominating Set



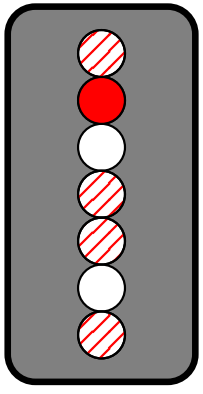
Dominating Set on a separator



Dominating Set on a separator

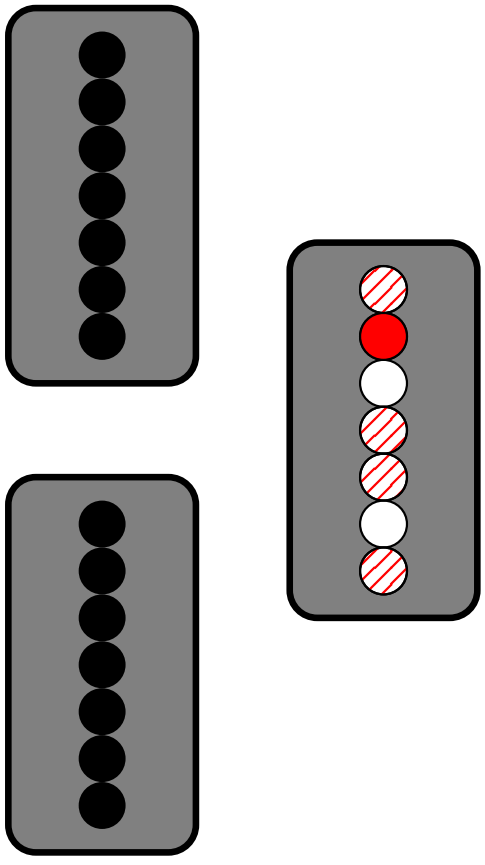


Dominating Set on a separator



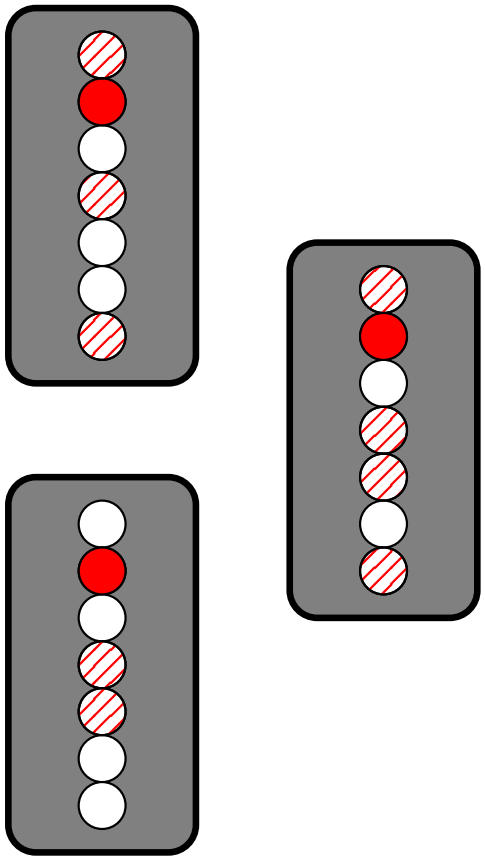
- $B = B_X \uplus B_+ \uplus B_-$;
  $T[B_X, B_+] = \min\{|X| : X \cap B = B_X,$
  $N_G[X] = B_X \cup B_+ \cup \text{forgotten vrts}\}$

Join node



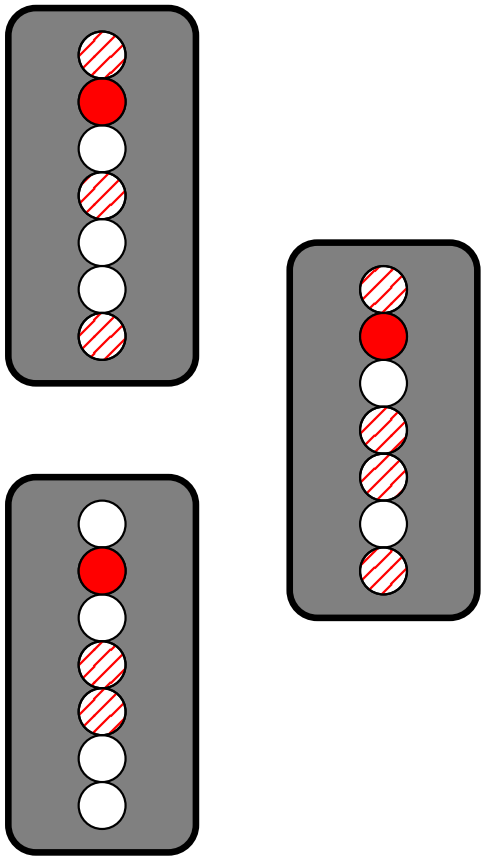
- $B = B_X \uplus B_+ \uplus B_-$;
  $T[B_X, B_+] = \min\{|X| : X \cap B = B_X,$
  $N_G[X] = B_X \cup B_+ \cup \text{forgotten vrts}\}$
- $B = B^1 = B^2$

Join node



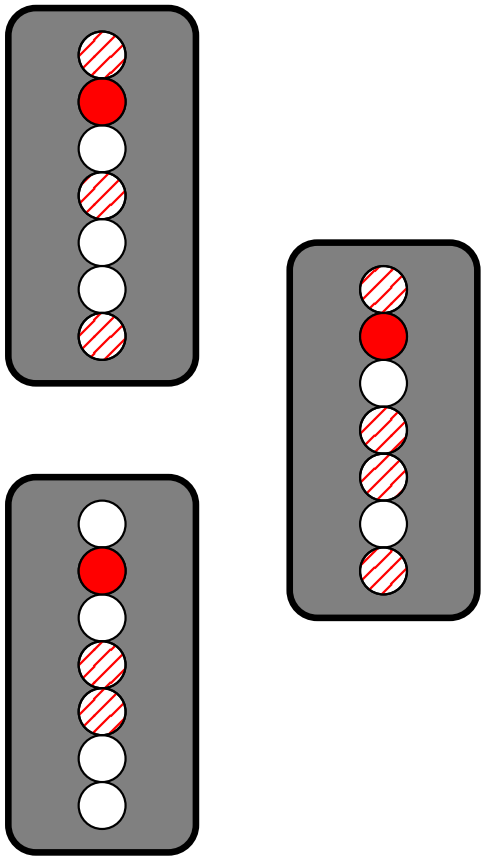
- $B = B_X \uplus B_+ \uplus B_-$;
  $T[B_X, B_+] = \min\{|X| : X \cap B = B_X,$
  $N_G[X] = B_X \cup B_+ \cup \text{forgotten vrts}\}$
- $B = B^1 = B^2$
- $B_X = B^1_X = B^2_X$ and $B_+ = B^1_+ \cup B^2_+$

Join node



- $B = B_X \uplus B_+ \uplus B_-$;
  $T[B_X, B_+] = \min\{|X| : X \cap B = B_X,$
  $N_G[X] = B_X \cup B_+ \cup \text{forgotten vrts}\}$
- $B = B^1 = B^2$
- $B_X = B^1_X = B^2_X$ and $B_+ = B^1_+ \cup B^2_+$
- $T[B_X, B_+] =$
  $\min\{ T^1[B_X, P] + T^2[B_X, Q] - |B_X| :$
  $P, Q \subseteq B_+, P \cup Q = B_+\}$

Join node



- $B = B_X \uplus B_+ \uplus B_-$;
  $T[B_X, B_+] = \min\{|X| : X \cap B = B_X,$
  $N_G[X] = B_X \cup B_+ \cup \text{forgotten vrts}\}$
- $B = B^1 = B^2$
- $B_X = B^1_X = B^2_X$ and $B_+ = B^1_+ \cup B^2_+$
- $T[B_X, B_+] =$
  $\min\{T^1[B_X, P] + T^2[B_X, Q] - |B_X| :$
  $P, Q \subseteq B_+, P \cup Q = B_+\}$
- naive implementation: $5^{|B|}$ time

Join node

- Need to compute in $2^n \mathrm{poly}(n)$ time convolution

$$f *_{\min} g(A) = \min\{f(B) + g(C) : B \cup C = A\}.$$

- Need to compute in $2^n \mathrm{poly}(n)$ time convolution

$$f *_{\min} g(A) = \min\{f(B) + g(C) : B \cup C = A\}.$$

- In our case values in $\{0, 1, \ldots, n\}$.

- Need to compute in $2^n \mathrm{poly}(n)$ time convolution

$$f *_{\min} g(A) = \min\{f(B) + g(C) : B \cup C = A\}.$$

- In our case values in $\{0, 1, \ldots, n\}$.
- Suffices to solve: for fixed $0 \leqslant b, c \leqslant n$, find all A for which there exist $A = B \cup C$ with $f(B) = b$ and $g(C) = c$.

- Need to compute in $2^n \mathrm{poly}(n)$ time convolution

$$f *_{\min} g(A) = \min\{f(B) + g(C) : B \cup C = A\}.$$

- In our case values in $\{0, 1, \ldots, n\}$.
- Suffices to solve: for fixed $0 \leqslant b, c \leqslant n$, find all A for which there exist $A = B \cup C$ with $f(B) = b$ and $g(C) = c$.
- Define $f_b(B) = [f(B) = b]$, $g_c(C) = [g(C) = c]$.

- Need to compute in $2^n \mathrm{poly}(n)$ time convolution

$$f *_{\min} g(A) = \min\{f(B) + g(C) : B \cup C = A\}.$$

- In our case values in $\{0, 1, \ldots, n\}$.
- Suffices to solve: for fixed $0 \leqslant b, c \leqslant n$, find all A for which there exist $A = B \cup C$ with $f(B) = b$ and $g(C) = c$.
- Define $f_b(B) = [f(B) = b]$, $g_c(C) = [g(C) = c]$.
- Compute covering product and check if $f_b *_c g_c(A) > 0$.

$$f_b *_c g_c(A) = \sum_{B,C \subseteq A : B \cup C = A} f_b(B) g_c(C).$$

For fixed B_X, define $f(P) = T^1[B_X, P]$ and $g(Q) = T^2[B_X, Q]$ over universe $B \setminus B_X$.

Theorem (BRR, ESA'09)

We can solve Dominating Set *in* $3^{\mathrm{tw}}\mathrm{poly}(n)$ *time.*

For fixed B_X, define $f(P) = T^1[B_X, P]$ and $g(Q) = T^2[B_X, Q]$ over universe $B \setminus B_X$.

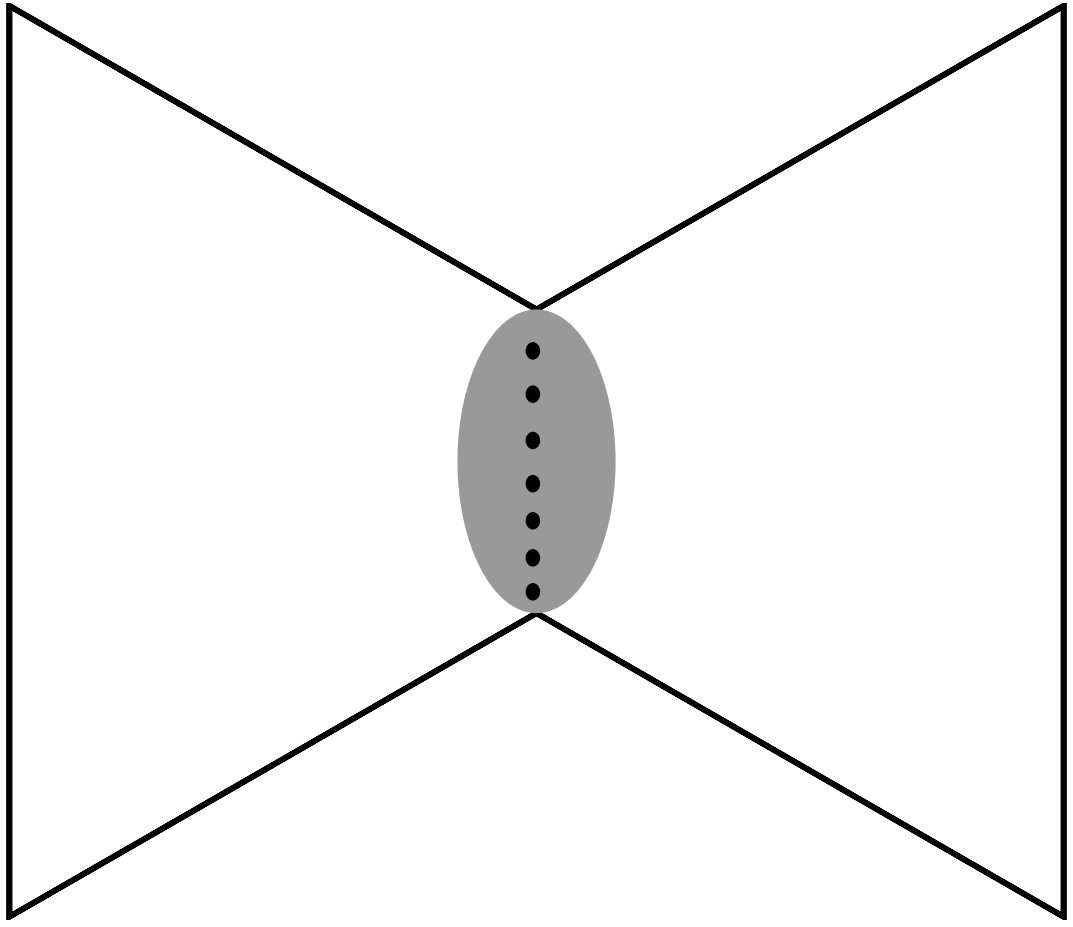
Theorem (BRR, ESA'09)

We can solve Dominating Set *in* $3^{\mathrm{tw}}\mathrm{poly}(n)$ *time.*

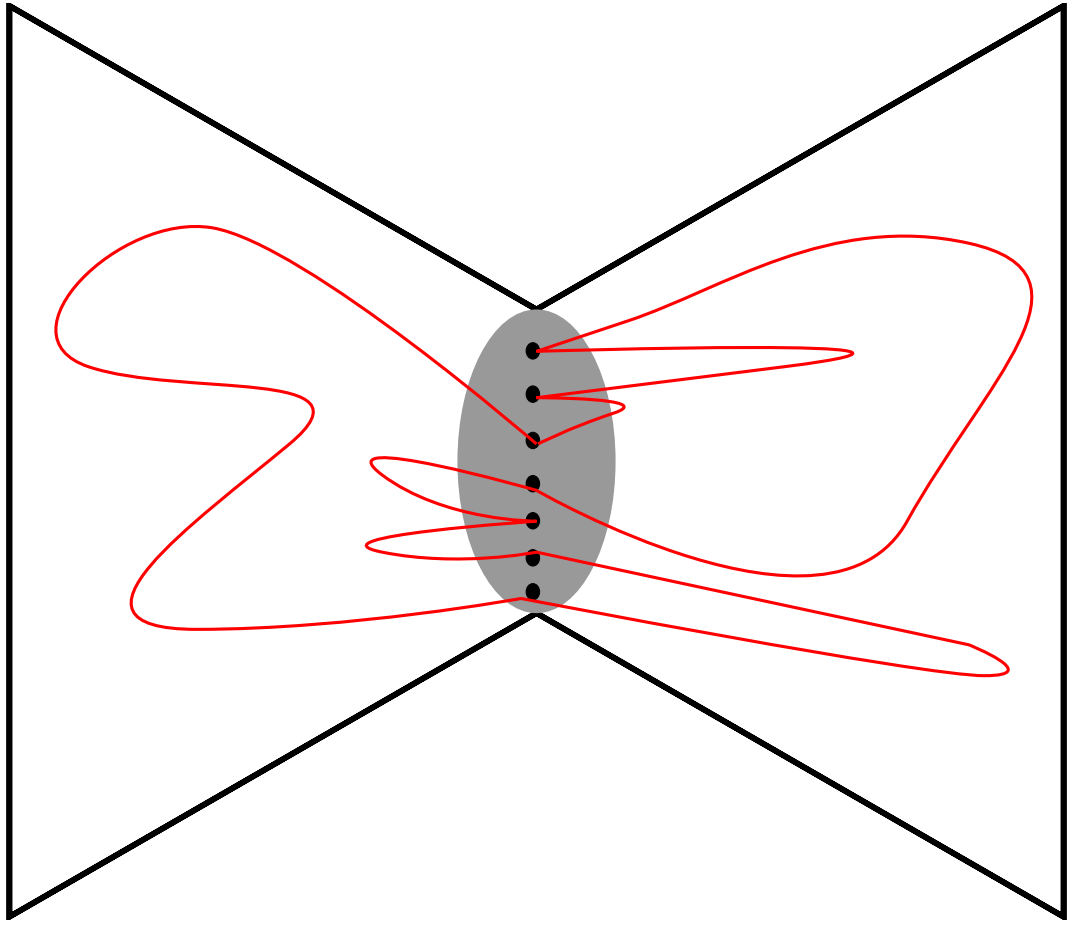
In many cases we can handle the join node efficiently using some convolution.

Connectivity problems

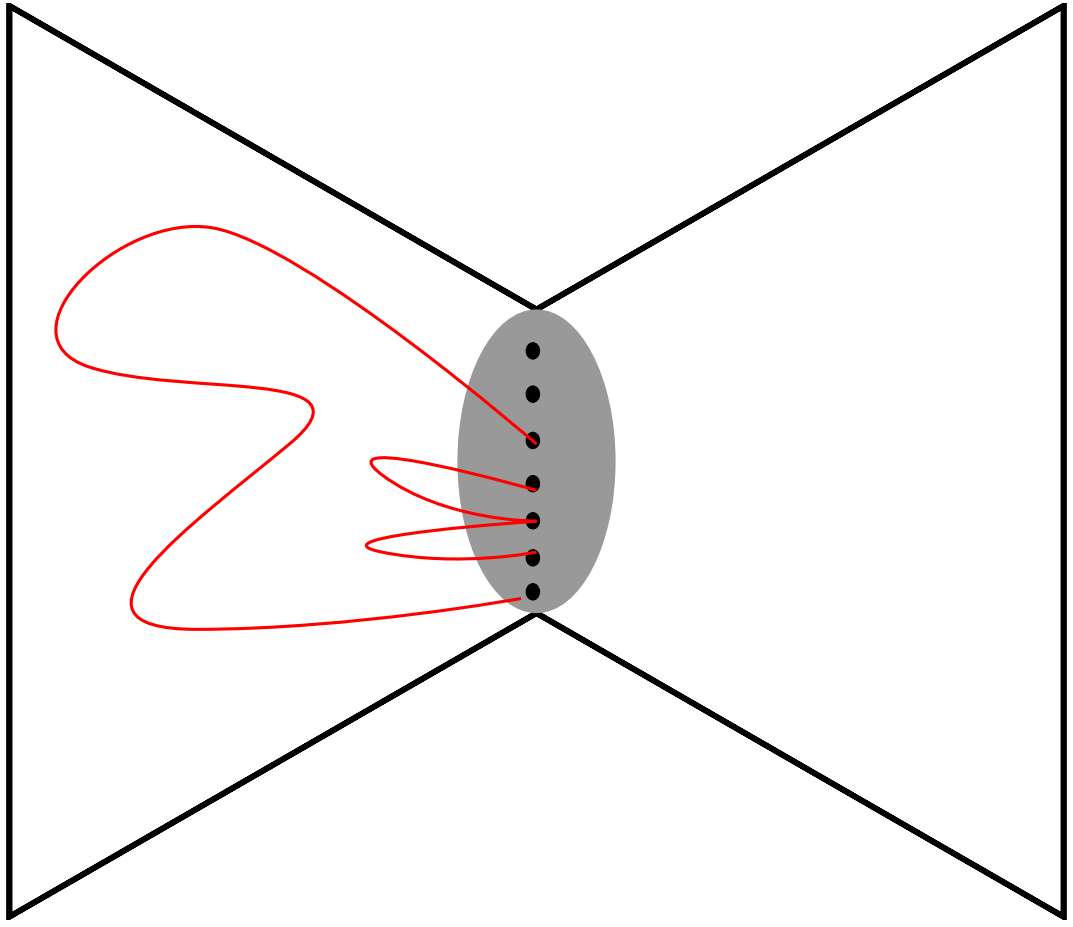
Consider the Hamiltonian cycle problem:



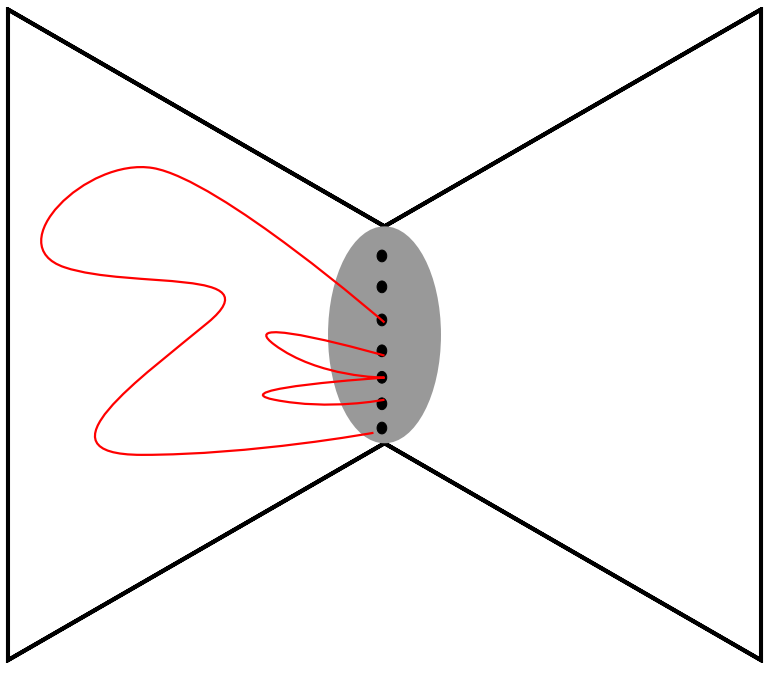
Consider the Hamiltonian cycle problem:



Consider the Hamiltonian cycle problem:

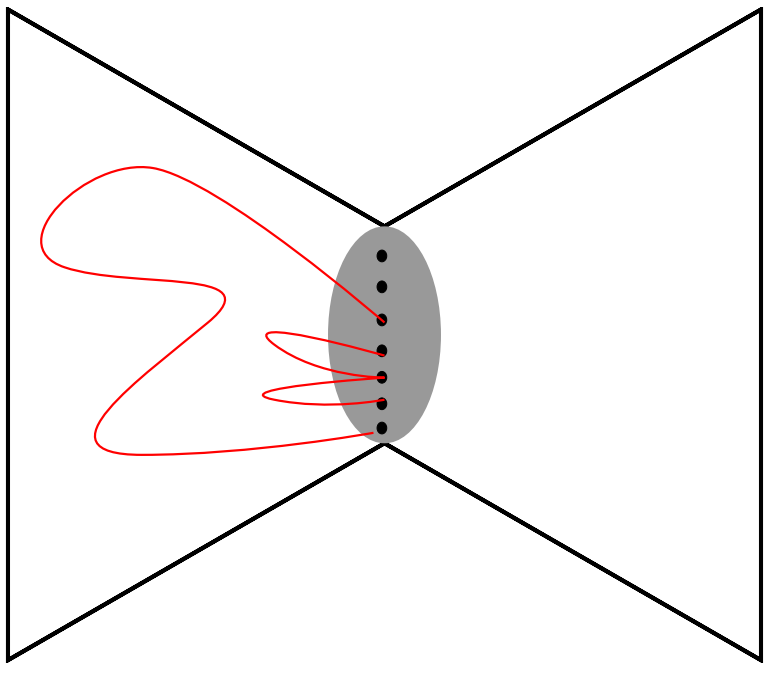


In a state of the standard DP for HC we store:



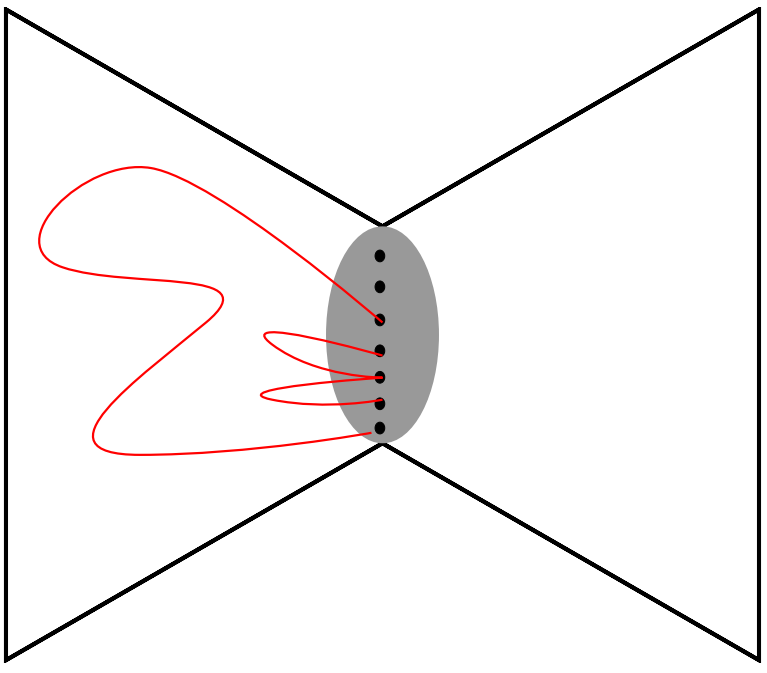
In a state of the standard DP for HC we store:

1. a partition according to degrees $B = B_0 \uplus B_1 \uplus B_2$,



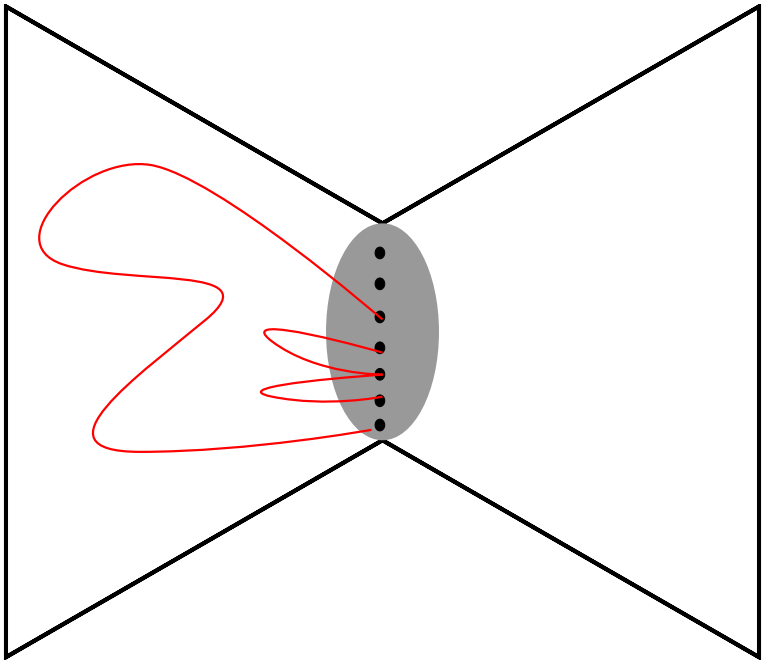
In a state of the standard DP for HC we store:

1. a partition according to degrees $B = B_0 \uplus B_1 \uplus B_2$,
2. a perfect matching on B_1 (path's endpoints).



In a state of the standard DP for HC we store:

1. a partition according to degrees $B = B_0 \uplus B_1 \uplus B_2$, (cheap: $3^{|B|}$)
2. a perfect matching on B_1 (path's endpoints).
   (expensive: $|B_1|!! \geqslant (|B_1|/2)!$)



For several problems with "connectivity" flavour standard approach leads to $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}\mathrm{poly}(n) = c^{\mathrm{tw}\log\mathrm{tw}}\mathrm{poly}(n)$ time algorithms:

- Hamiltonicity,
- Steiner Tree,
- Connected Vertex Cover,
- Connected Dominating Set,
- Feedback Vertex Set,
- Cycle Packing,
- ...

For several problems with "connectivity" flavour standard approach leads to $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})}\mathrm{poly}(n) = c^{\mathrm{tw}\log\mathrm{tw}}\mathrm{poly}(n)$ time algorithms:

- Hamiltonicity,
- Steiner Tree,
- Connected Vertex Cover,
- Connected Dominating Set,
- Feedback Vertex Set,
- Cycle Packing,
- ...

Better algorithm for special graph classes (planar, H-minor free) using Sphere-Cut Decompositions and Catalan Structures [DPBF, ESA'05 + SODA'08]

Cut&Count technique introduced in [CNPPRW, FOCS 2011] gives Monte Carlo algorithms:

Problem name	algorithm	l.bound (SETH)
Steiner Tree	3^{tw}	$(3-\epsilon)^{\mathrm{tw}}$
Feedback Vertex Set	3^{tw}	$(3-\epsilon)^{\mathrm{tw}}$
Connected Vertex Cover	3^{tw}	$(3-\epsilon)^{\mathrm{tw}}$
Connected Dominating Set	4^{tw}	$(4-\epsilon)^{\mathrm{tw}}$
Hamiltonian Cycle (directed)	4^{tw} (6^{tw})	
...		
Cycle Packing		$2^{\Omega(p \log p)}$

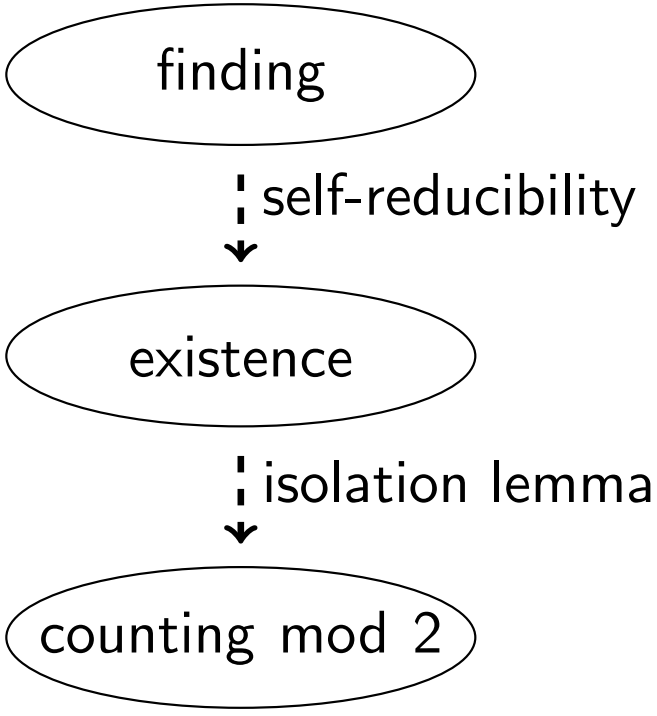
- What is a Hamiltonian cycle?

- What is a Hamiltonian cycle?
- It is a cycle that visits each vertex exactly once - correct but not useful.

- What is a Hamiltonian cycle?
- It is a cycle that visits each vertex exactly once - correct but not useful.
- But we can define the problem in a different way, where we split the constraints such that the connectivity requirement is isolated.

- What is a Hamiltonian cycle?
- It is a cycle that visits each vertex exactly once - correct but not useful.
- But we can define the problem in a different way, where we split the constraints such that the connectivity requirement is isolated.
- Lets try once again: a Hamiltonian cycle is a cycle cover that is connected.

- What is a Hamiltonian cycle?
- It is a cycle that visits each vertex exactly once - correct but not useful.
- But we can define the problem in a different way, where we split the constraints such that the connectivity requirement is isolated.
- Lets try once again: a Hamiltonian cycle is a cycle cover that is connected.
- Cycle cover is a subset of edges such that each vertex is of degree exactly two.

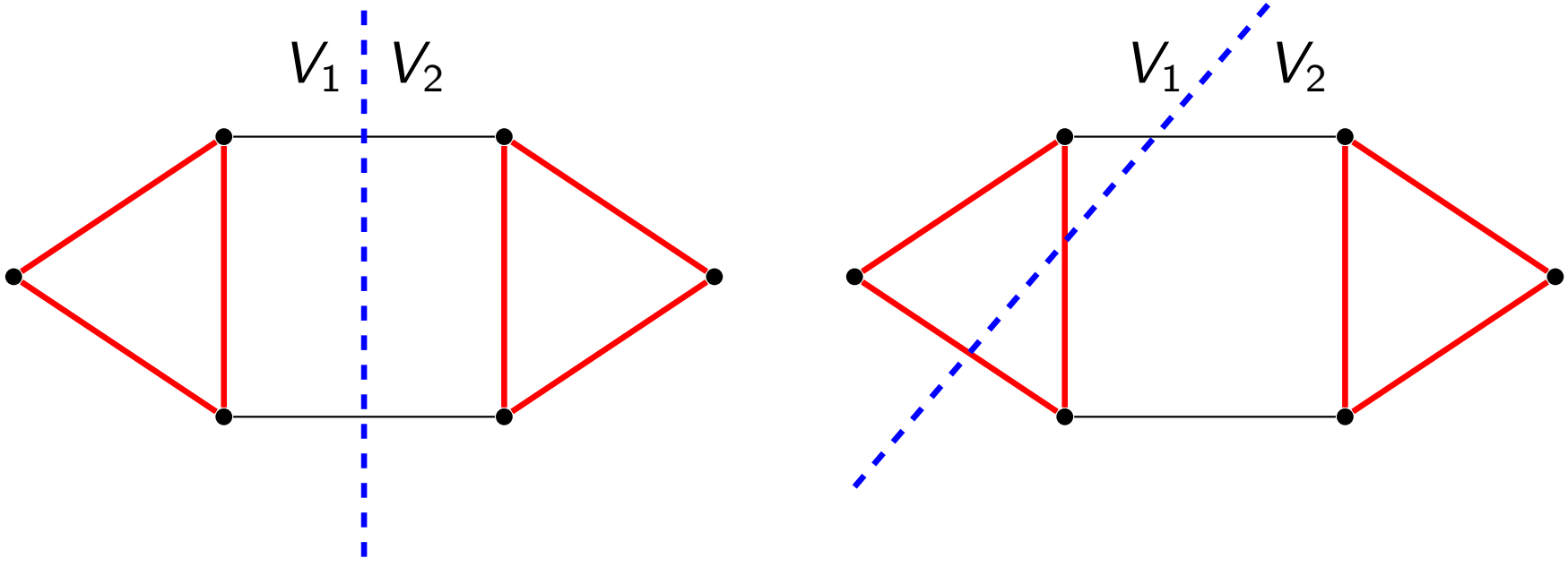

finding
self-reducibility
existence
isolation lemma
counting mod 2

Isolation Lemma[Mulmuley, Vazirani, Vazirani'87]

Let S be the set of Hamiltonian cycles ($S \subseteq 2^E$) with $|\mathcal{S}| > 0$. For each $e \in E$, choose a weight $\omega(e) \in \{1, 2, \ldots, 2|E|\}$ uniformly and independently at random. Then with probability at least $1/2$ there exists exactly one Hamiltonian cycle $X_0 \in \mathcal{S}$ of minimum weight w.r.t. ω.

New goal: find the number of (weighted) Hamiltonian cycles mod 2.

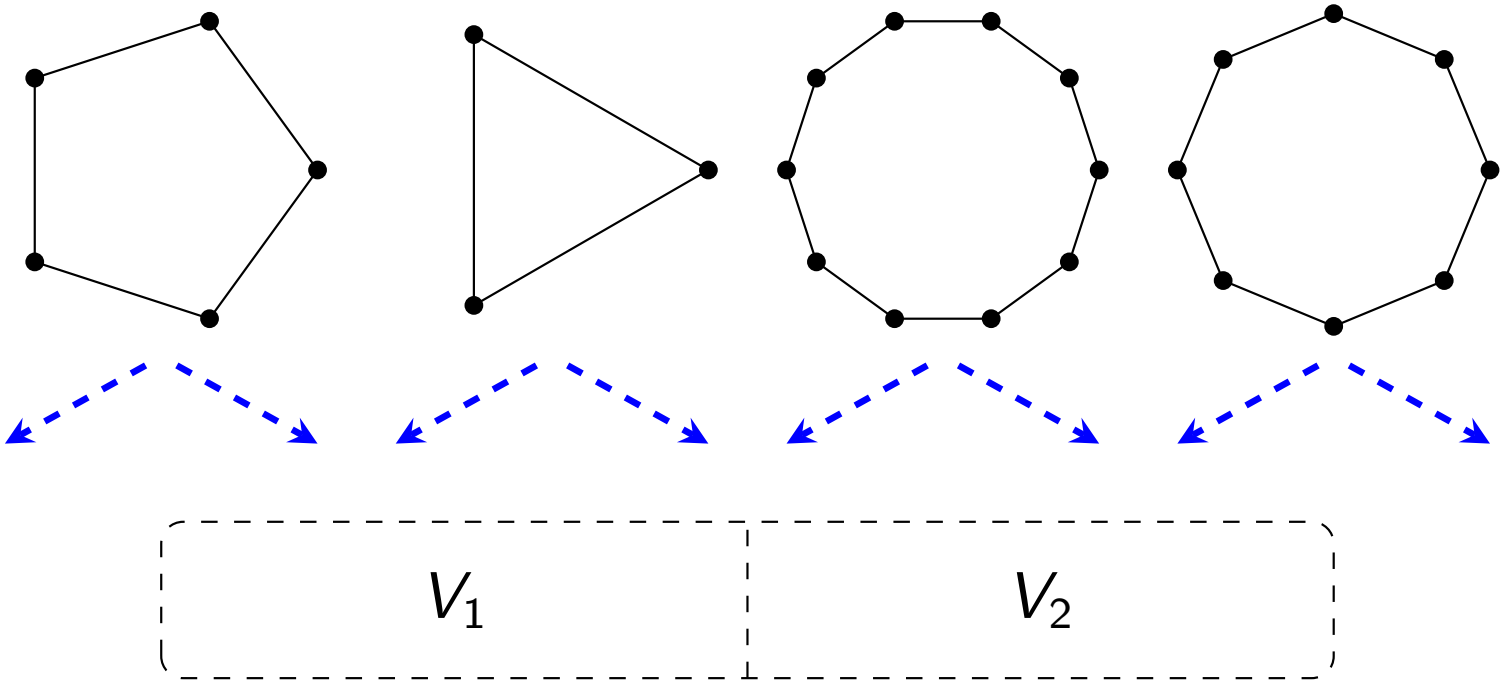
- By a cut (V_1, V_2) we denote a partition $V = V_1 \uplus V_2$.
- We say that a cycle cover $X \subseteq E$ is *consistent* with a cut (V_1, V_2) iff $X \cap E(V_1, V_2) = \emptyset$.



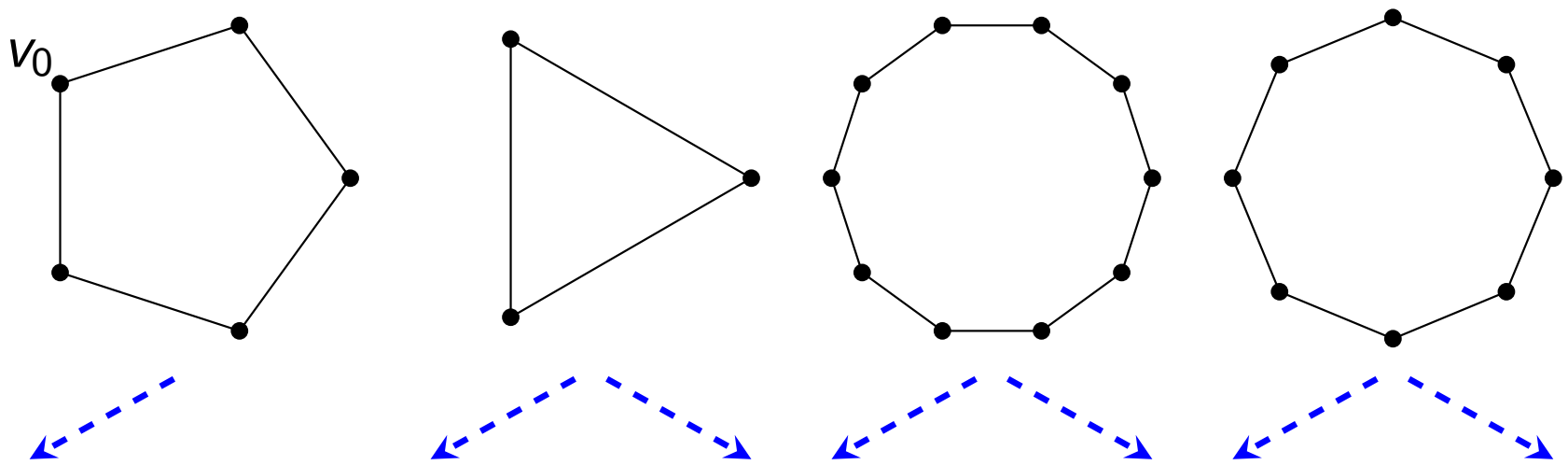
Main idea

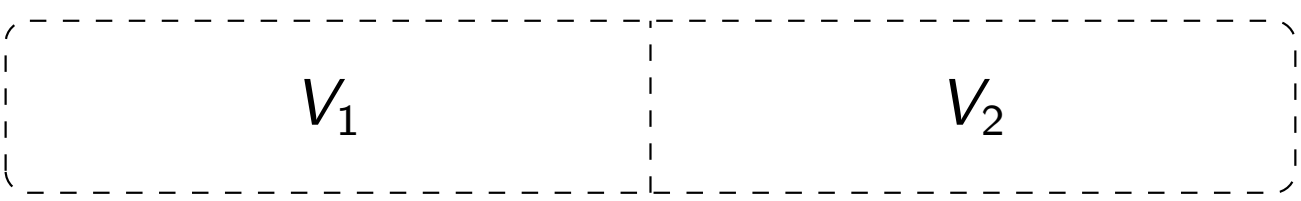
Instead of counting Hamiltonian cycles we will count pairs (cycle cover, consistent cut).

- A cycle cover with a cycles is consistent with exactly 2^a cuts.



- We would like a Hamiltonian cycle to be counted once, while a cycle cover with $\geqslant 2$ cycles an even number of times.
- Fix $v_0 \in V$ and consider cuts with $v_0 \in V_1$.





Cut & count

$$
\begin{aligned}
S = \quad & \text{Hamiltonian cycles} \\
R = \quad & \text{cycle covers} \\
C = \quad & \{(X, (V_1, V_2)) \ : \ X \in R \ \wedge \ v_0 \in V_1 \\
& \qquad \wedge \ X \text{ is consistent with } (V_1, V_2)\}
\end{aligned}
$$

Lemma

$$|C| \equiv |S| \pmod 2$$

$$
\begin{aligned}
S = &\quad \text{Hamiltonian cycles} \\
R = &\quad \text{cycle covers} \\
C = &\quad \{(X,(V_1,V_2)) \ : \ X \in R \ \wedge \ v_0 \in V_1 \\
&\qquad \wedge \ X \text{ is consistent with } (V_1,V_2)\}
\end{aligned}
$$

Lemma

$$|C| \equiv |S| \pmod 2$$

One can compute $|C|$ in $6^{\mathrm{tw}}\mathrm{poly}(n)$ time by standard methods!

Cut&Count technique introduced in [CNPPRW, FOCS 2011] gives Monte Carlo algorithms:

Problem name	algorithm	l.bound (SETH)
Steiner Tree	3^{tw}	$(3-\epsilon)^{\mathrm{tw}}$
Feedback Vertex Set	3^{tw}	$(3-\epsilon)^{\mathrm{tw}}$
Connected Vertex Cover	3^{tw}	$(3-\epsilon)^{\mathrm{tw}}$
Connected Dominating Set	4^{tw}	$(4-\epsilon)^{\mathrm{tw}}$
Hamiltonian Cycle (directed)	4^{tw} (6^{tw})	
...		
Cycle Packing		$2^{\Omega(p \log p)}$

Rank based approach

Although Cut&Count allowed to break the $c^{\mathrm{tw}\log\mathrm{tw}}$ barrier and in some cases even gave (probably) optimal constants c, it has drawbacks:

1. it is randomized,
2. it does not solve weighted nor counting versions,

Although Cut&Count allowed to break the $c^{\mathrm{tw}\log\mathrm{tw}}$ barrier and in some cases even gave (probably) optimal constants c, it has drawbacks:

1. it is randomized,
2. it does not solve weighted nor counting versions,

[BCKN, ICALP'13][CKN, STOC'13]

There is a way to partially resolve the above issues.

[FLS, SODA'14]

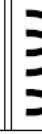
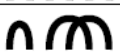
Alternative explanation using matroids.

In a state of the standard DP for HC we store:

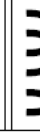
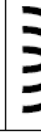
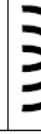
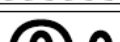
1. a partition according to degrees $B = B_0 \uplus B_1 \uplus B_2$, (cheap)
2. a perfect matching on B_1 (path's endpoints). (expensive)

Main objective

We want to shrink the set of reachable states, by keeping for a fixed partition $B = B_0 \uplus B_1 \uplus B_2$ at most $2^{|B_1|}$ perfect matchings on B_1.

Nr		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
																
1		0	0	0	0	1	1	0	1	1	1	1	0	1	1	0
2		0	0	0	1	0	1	1	1	0	0	1	1	1	0	1
3		0	0	0	1	1	0	1	0	1	1	0	1	0	1	1
4		0	1	1	0	0	0	0	1	1	1	0	1	1	0	1
5		1	0	1	0	0	0	1	0	1	0	1	1	1	1	0
6		1	1	0	0	0	0	1	1	0	1	1	0	0	1	1
7		0	1	1	0	1	1	0	0	0	0	1	1	0	1	1
8		1	1	0	1	0	1	0	0	0	1	0	1	1	1	0
9		1	0	1	1	1	0	0	0	0	1	1	0	1	0	1
10		1	0	1	1	0	1	0	1	1	0	0	0	0	1	1
11		1	1	0	0	1	1	1	0	1	0	0	0	1	0	1
12		0	1	1	1	1	0	1	1	0	0	0	0	1	1	0
13		1	1	0	1	1	0	0	1	1	0	1	1	0	0	0
14		1	0	1	0	1	1	1	1	0	1	0	1	0	0	0
15		0	1	1	1	0	1	1	0	1	1	1	0	0	0	0

From DP to rank - big table

Nr		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
																
1		0	0	0	0	1	1	0	1	1	1	1	0	1	1	0
2		0	0	0	1	0	1	1	1	0	0	1	1	1	0	1
3		0	0	0	1	1	0	1	0	1	1	0	1	0	1	1
4		0	1	1	0	0	0	0	1	1	1	0	1	1	0	1
5		1	0	1	0	0	0	1	0	1	0	1	1	1	1	0
6		1	1	0	0	0	0	1	1	0	1	1	0	0	1	1
7		0	1	1	0	1	1	0	0	0	0	1	1	0	1	1
8		1	1	0	1	0	1	0	0	0	1	0	1	1	1	0
9		1	0	1	1	1	0	0	0	0	1	1	0	1	0	1
10		1	0	1	1	0	1	0	1	1	0	0	0	0	1	1
11		1	1	0	0	1	1	1	0	1	0	0	0	1	0	1
12		0	1	1	1	1	0	1	1	0	0	0	0	1	1	0
13		1	1	0	1	1	0	0	1	1	0	1	1	0	0	0
14		1	0	1	0	1	1	1	1	0	1	0	1	0	0	0
15		0	1	1	1	0	1	1	0	1	1	1	0	0	0	0

From DP to rank - big table

Theorem

The rank of this matrix over $\mathbb{Z}_2$ is exactly $2^{|B|/2-1}$.

- Consider a matrix $\mathbf{T}$:
  - rows indexed by matchings M,
  - colums indexed by cuts (V_1, V_2) with $v_0 \in V_1$,
  - 1 if M is consistent with the cut, 0 otherwise.

Theorem

The rank of this matrix over $\mathbb{Z}_2$ is exactly $2^{|B|/2-1}$.

- Consider a matrix $\mathbf{T}$:
  - rows indexed by matchings M,
  - colums indexed by cuts (V_1, V_2) with $v_0 \in V_1$,
  - 1 if M is consistent with the cut, 0 otherwise.
- Crucial observation: $\mathbf{M} = \mathbf{T}\mathbf{T}^\top$ over $\mathbb{Z}_2$.
  - In a cell (M_1, M_2), the product on the left counts the number of cuts consistent with both M_1 and M_2.
  - Two matchings sum up to Hamiltonian cycle if there is only one such cut, and an even number of such cuts if they do not.

- We can keep only realizable matchings that are independent in $\mathbf{M} \Rightarrow$ only $2^{|B|/2-1}$ of them.

- We can keep only realizable matchings that are independent in $\mathbf{M} \Rightarrow$ only $2^{|B|/2-1}$ of them.
- Using $\mathbf{T}$, we can efficiently discard redundant (dependent) matchings.

- We can keep only realizable matchings that are independent in $\mathbf{M} \Rightarrow$ only $2^{|B|/2-1}$ of them.
- Using $\mathbf{T}$, we can efficiently discard redundant (dependent) matchings.
- Discard heaviest dependent matching $\Rightarrow$ can solve weighted variants.

- We can keep only realizable matchings that are independent in $\mathbf{M} \Rightarrow$ only $2^{|B|/2-1}$ of them.
- Using $\mathbf{T}$, we can efficiently discard redundant (dependent) matchings.
- Discard heaviest dependent matching $\Rightarrow$ can solve weighted variants.
- Generalizes to all Cut & Count algorithms.

Weighted Steiner Tree	$n(1+2^{\omega+1})^{\mathrm{tw}}\mathrm{tw}^{\mathcal{O}(1)}$
Traveling Salesman	$n(5+2^{(\omega+2)/2})^{\mathrm{tw}}\mathrm{tw}^{\mathcal{O}(1)}$
k-Path	$n(5+2^{(\omega+2)/2})^{\mathrm{tw}}(k+\mathrm{tw})^{\mathcal{O}(1)}$
Feedback Vertex Set	$n(1+2^{\omega+1})^{\mathrm{tw}}\mathrm{tw}^{\mathcal{O}(1)}$

Conclusions and open problems

1. **Message 1**: in many cases, convolution tools speed up computations at join nodes.

1. **Message 1**: in many cases, convolution tools speed up computations at join nodes.
2. **Message 2**: keeping stuff connected can be done significantly faster than the naive approach.

1. **Message 1**: in many cases, convolution tools speed up computations at join nodes.
2. **Message 2**: keeping stuff connected can be done significantly faster than the naive approach.
3. **Open problem 1**: Understand optimal constants in the base of the exponent (mostly for deterministic algorithms).

1. **Message 1**: in many cases, convolution tools speed up computations at join nodes.
2. **Message 2**: keeping stuff connected can be done significantly faster than the naive approach.
3. **Open problem 1**: Understand optimal constants in the base of the exponent (mostly for deterministic algorithms).
4. **Open problem 2**: Can we use the deterministic results to design $3^k \mathrm{poly}(n)$ time algorithm for FVS? (parameterized by the solution size)

1. **Message 1**: in many cases, convolution tools speed up computations at join nodes.
2. **Message 2**: keeping stuff connected can be done significantly faster than the naive approach.
3. **Open problem 1**: Understand optimal constants in the base of the exponent (mostly for deterministic algorithms).
4. **Open problem 2**: Can we use the deterministic results to design $3^k \mathrm{poly}(n)$ time algorithm for FVS? (parameterized by the solution size)
5. Exercises: all from Chapter 11.