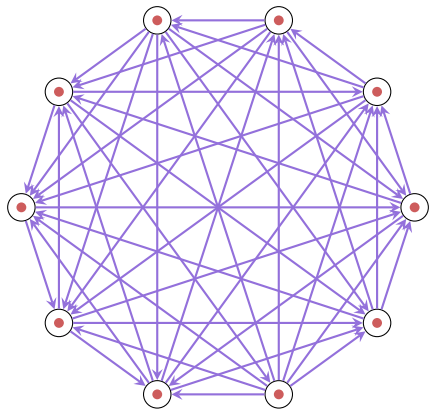
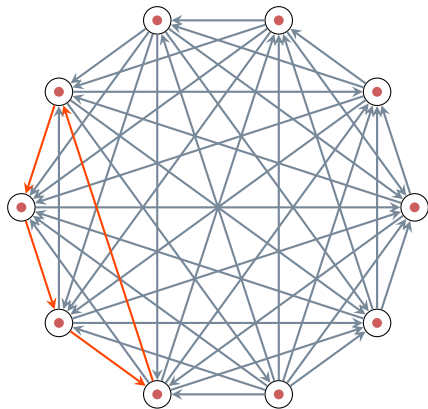


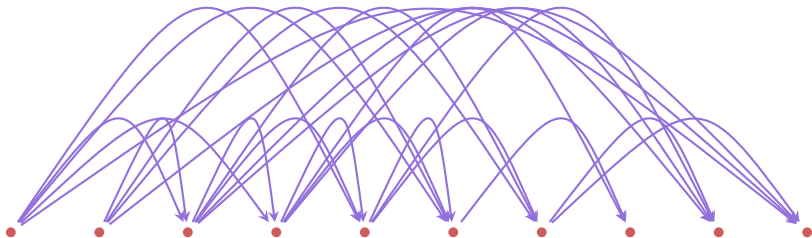
Random¹ Explorations in Algorithms for “Feedback Set” problems on Tournaments

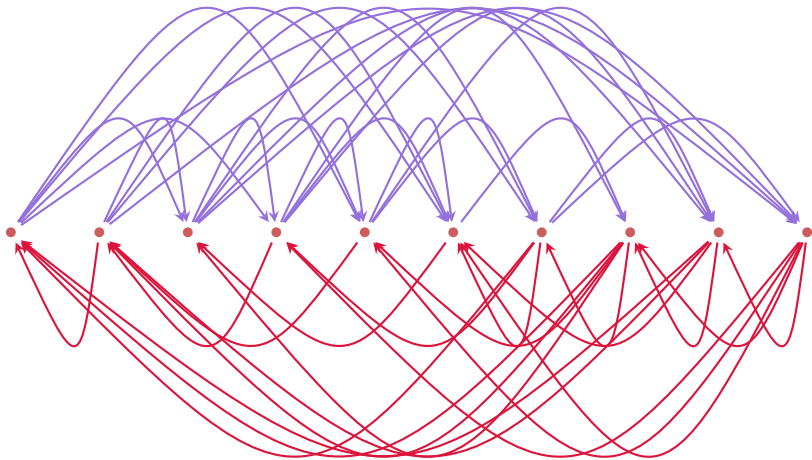
¹Pun unintended!











Feedback Vertex Set

Given: A tournament T and an integer k .

Find a subset S of at most k vertices so that $T \setminus S$ is acyclic.

Feedback Arc Set [By Deletions]

Given: A tournament T and an integer k .

Find a subset S of at most k arcs so that $T \setminus S$ is acyclic.

Feedback Arc Set [By Reversals]

Given: A tournament T and an integer k .

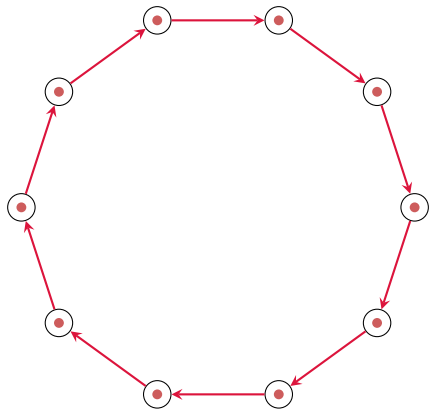
Find a subset S of at most k arcs so that $T \circ S$ is acyclic.

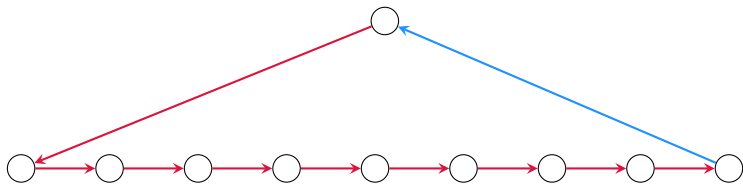
FPT Algorithms for FVST.

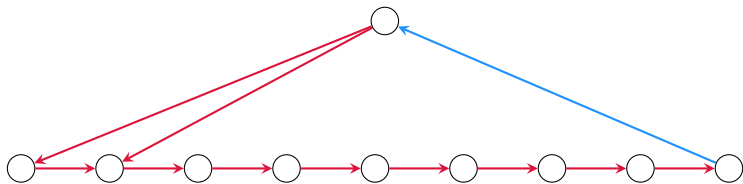
A tournament T has a directed cycle

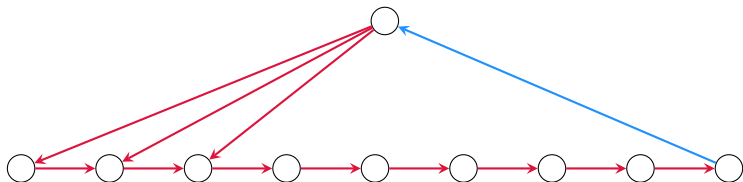


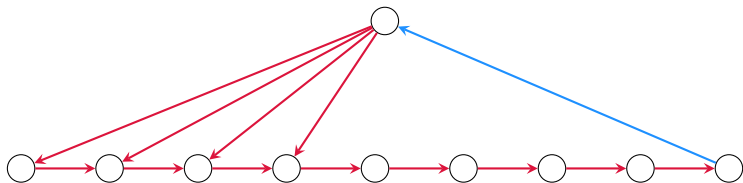
T has a directed triangle.

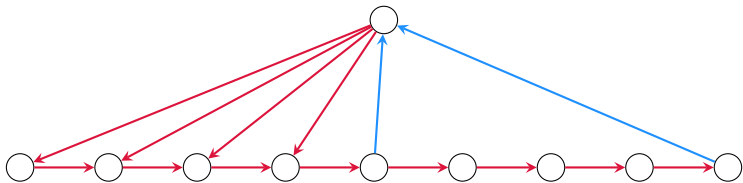


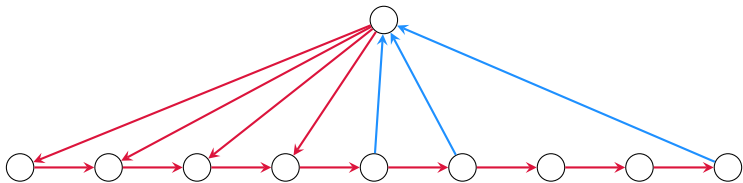


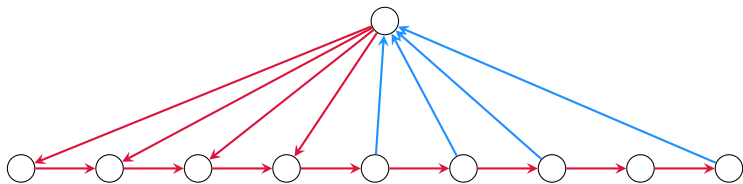


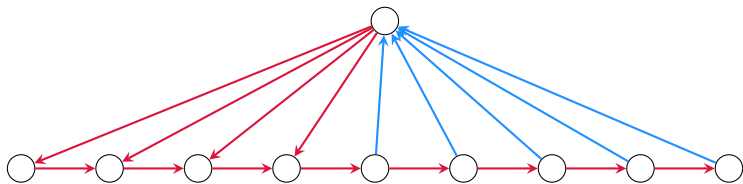


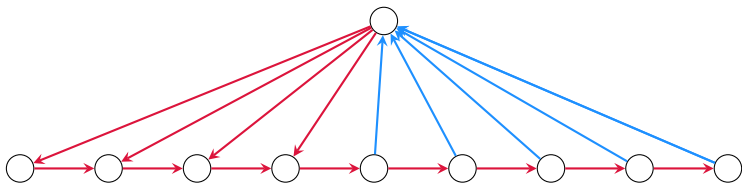


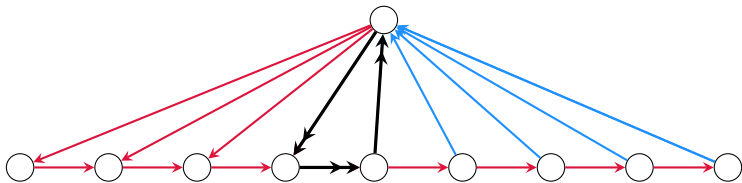












Immediately suggests a $\mathcal{O}(3^k)$ algorithm for Feedback Vertex Set.

Immediately suggests a $\mathcal{O}(3^k)$ algorithm for Feedback Vertex Set.

Also suggests:

FEEDBACK VERTEX SET $\implies$ 3-HITTING SET

Immediately suggests a $\mathcal{O}(3^k)$ algorithm for Feedback Vertex Set.

Also suggests:

FEEDBACK VERTEX SET $\implies$ 3-HITTING SET

Does this work for Feedback **Arc** Set in Tournaments (by deletions)?

Feedback Arc Set in Tournaments (by Deletions)



Feedback Arc Set in Tournaments (by Reversals)

T has a **minimal** feedback arc set of size k (by deletions).

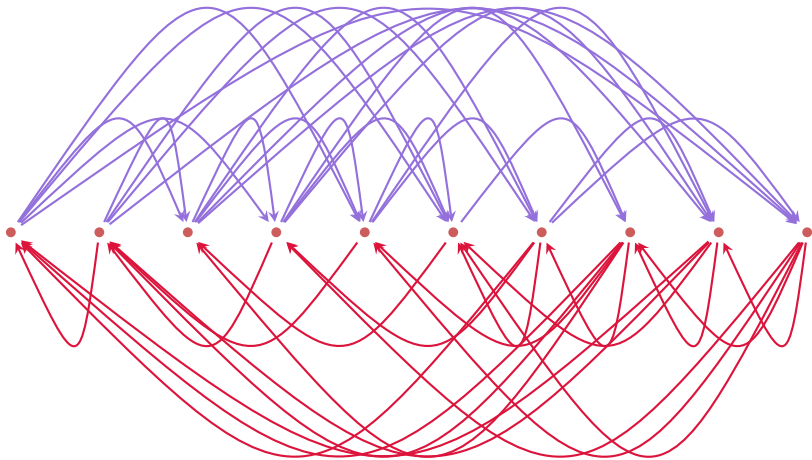


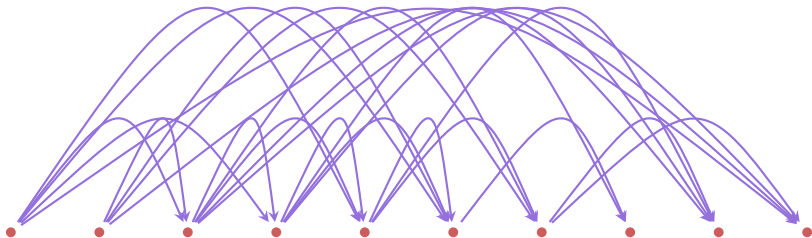
T has a **minimal** feedback arc set of size k (by reversals).

T has a minimal feedback arc set of size k (by deletions).



T has a minimal feedback arc set of size k (by reversals).





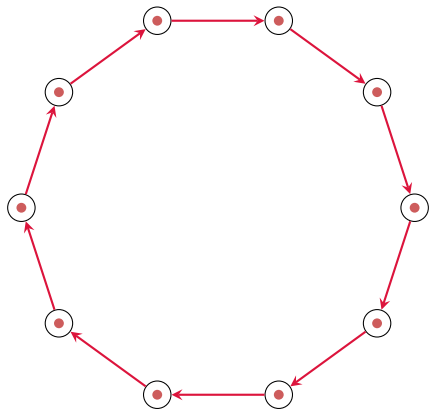
T has a minimal feedback arc set, F , of size k (by deletions).

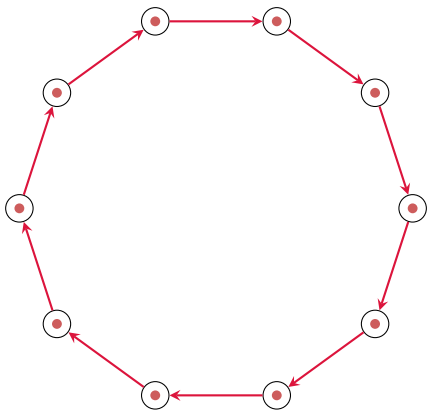


T has a minimal feedback arc set of size k (by reversals).

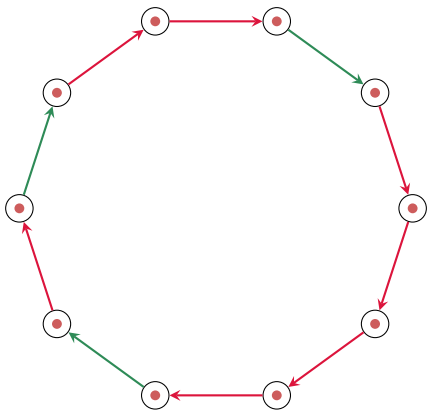
Reverse the set of arcs in the deletion set

Reverse the set of arcs in the deletion set,
and suppose we still have a cycle.

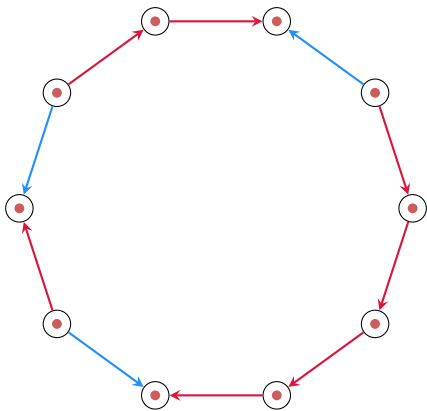




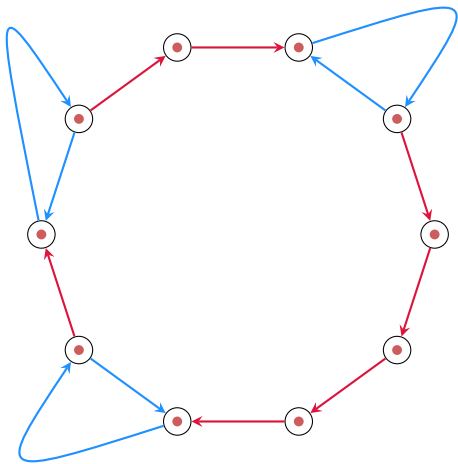
Clearly, the cycle must indulge in some of the reversed arcs.



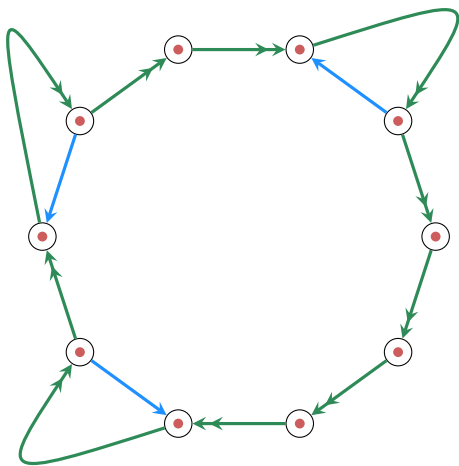
Clearly, the cycle must indulge in some of the reversed arcs.



Clearly, the cycle must indulge in some of the reversed arcs.



Clearly, the cycle must indulge in some of the reversed arcs.



Clearly, the cycle must indulge in some of the reversed arcs.

FVST: 3-HITTING SET-based algorithms (2006)

FVST: 3-HITTING SET-based algorithms (2006)
→ Iterative compression $\mathcal{O}(2^k)$ (2008).

FVST: 3-HITTING SET-based algorithms (2006)
→ Iterative compression $\mathcal{O}(2^k)$ (2008).

FAST: Efficient Branching (2006)

FVST: 3-HITTING SET-based algorithms (2006)
→ Iterative compression $\mathcal{O}(2^k)$ (2008).

FAST: Efficient Branching (2006)
→ $\mathcal{O}(2^k)$ [Open]

FVST: 3-HITTING SET-based algorithms (2006)

→ Iterative compression $\mathcal{O}(2^k)$ (2008).

FAST: Efficient Branching (2006)

→ $\mathcal{O}(2^k)$ [Open]

→ $\mathcal{O}(2^{o(k)})$. (2009)

Kernelization for FAST

An edge that participates in more than k triangles?

An edge that participates in more than k triangles?

Reverse it!

$$k \leftarrow k - 1.$$

A vertex that doesn't participate in any triangles?

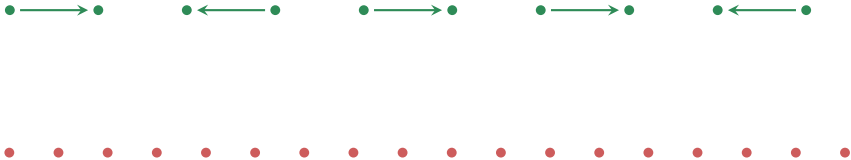
A vertex that doesn't participate in any triangles?

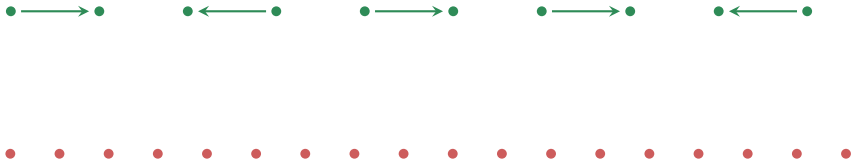
Delete it...

$k \leftarrow k.$

Structure of a YES-Instance...

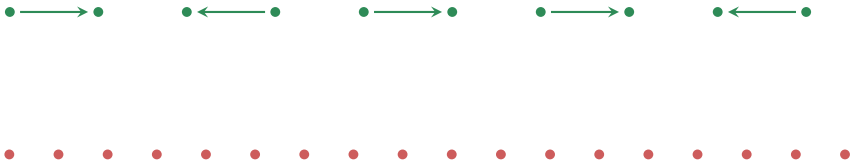






Each vertex on the bottom participates in
a triangle with an edge on top.

Each edge on top can “handle” at most k triangles.



Each vertex on the bottom participates in
a triangle with an edge on top.

Each edge on top can “handle” at most k triangles.



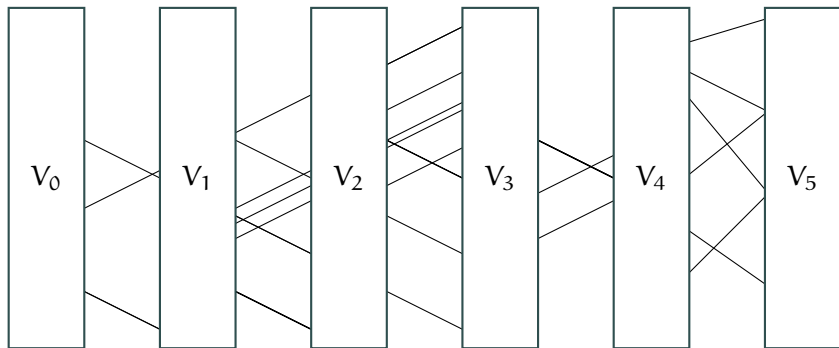
The total size of a valid instance is $\mathcal{O}(k^2)$.

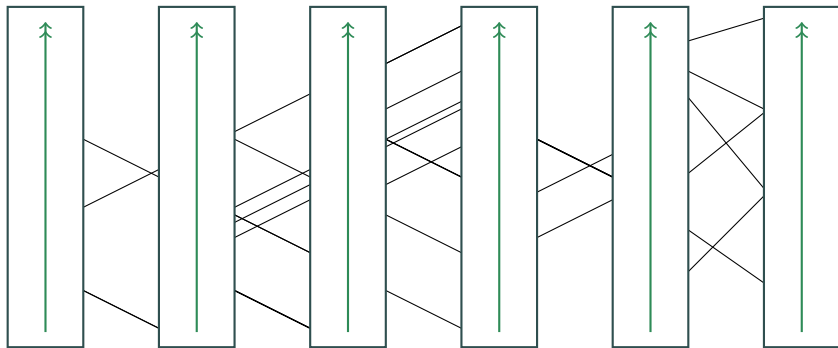


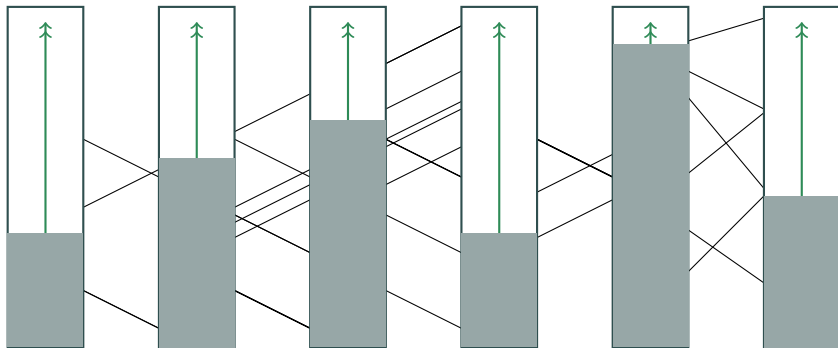
Each vertex on the bottom participates in
a triangle with an edge on top.

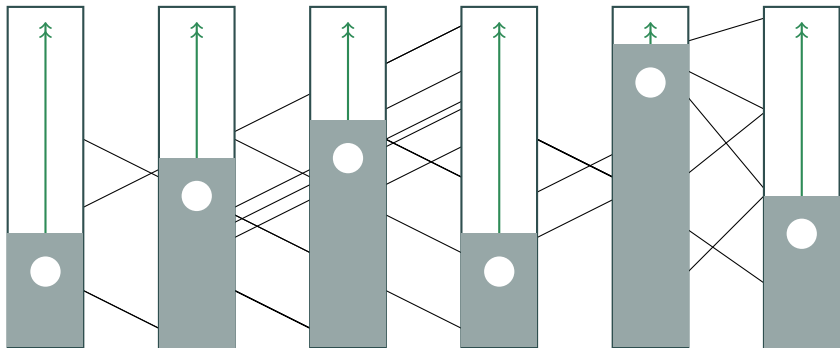
Chromatic Coding

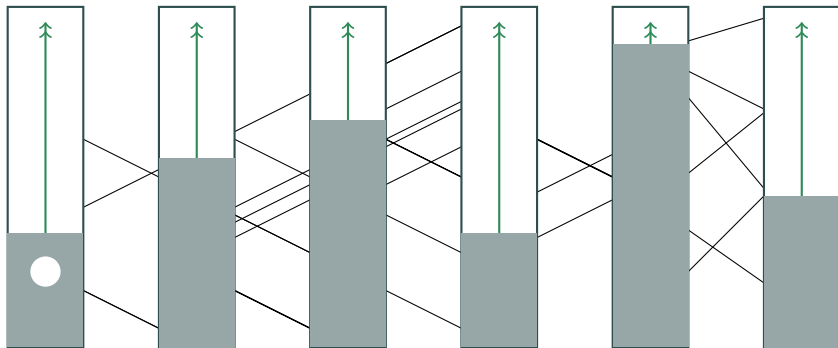
A “Promise” Version

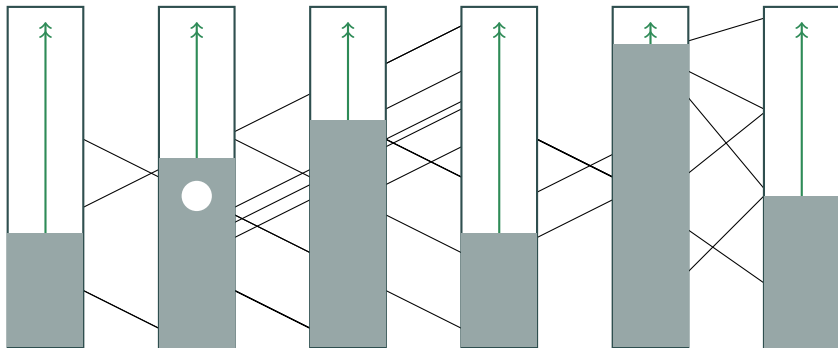


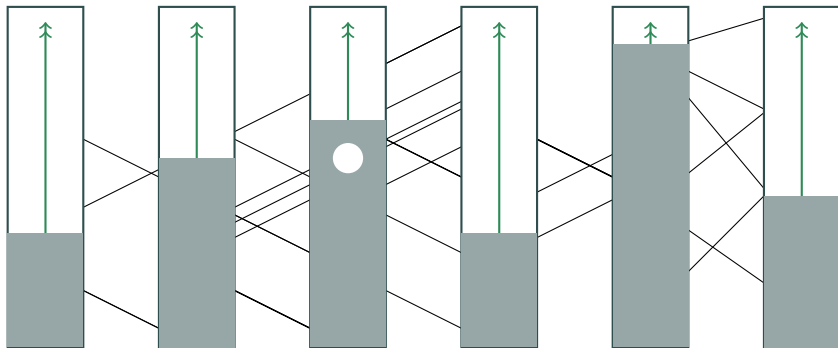


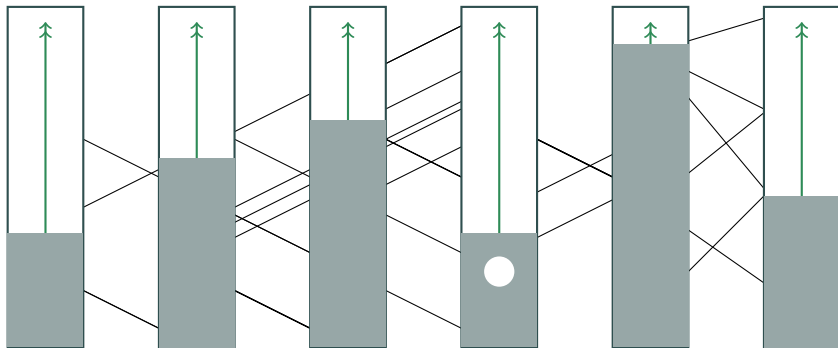


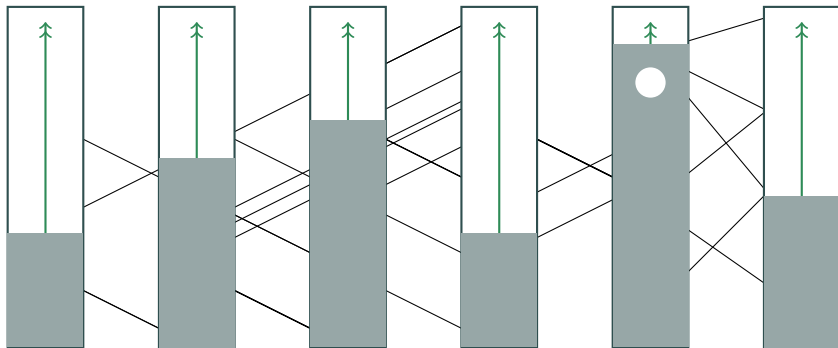


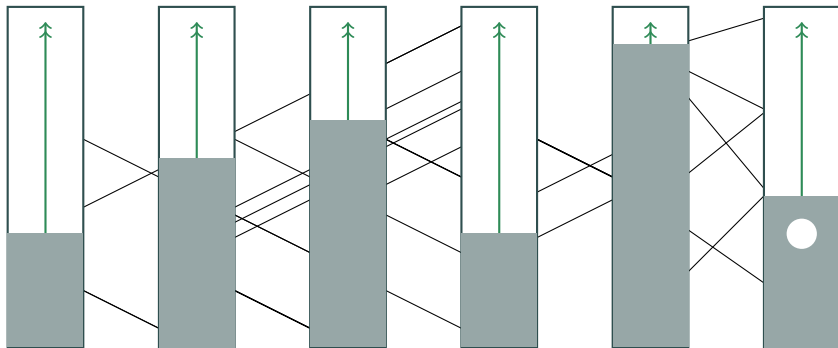


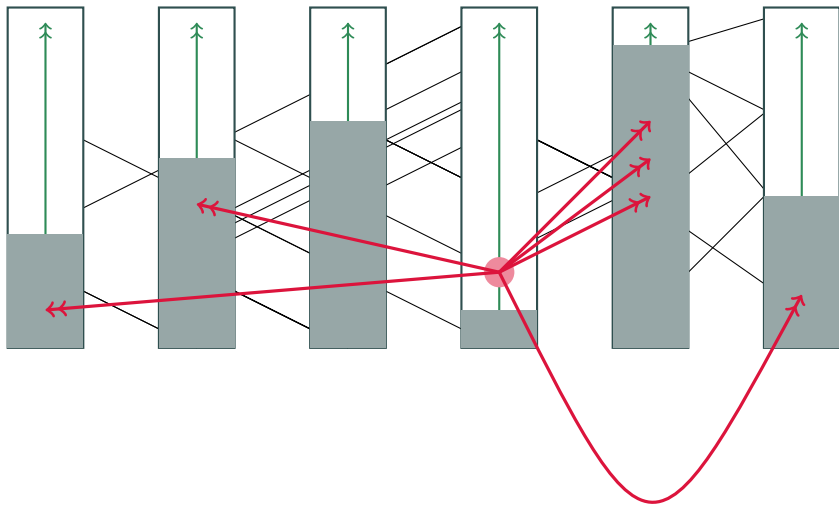






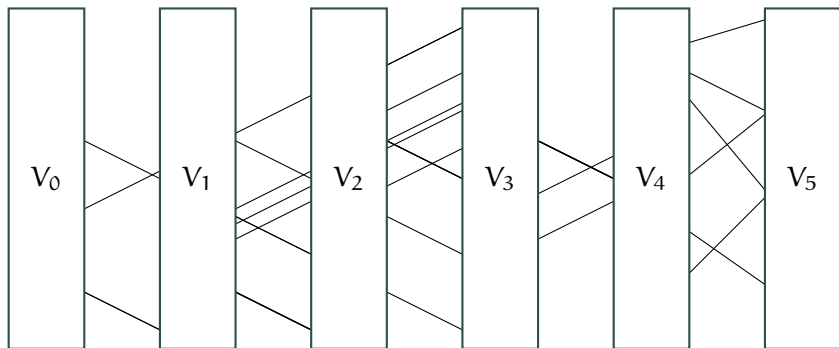


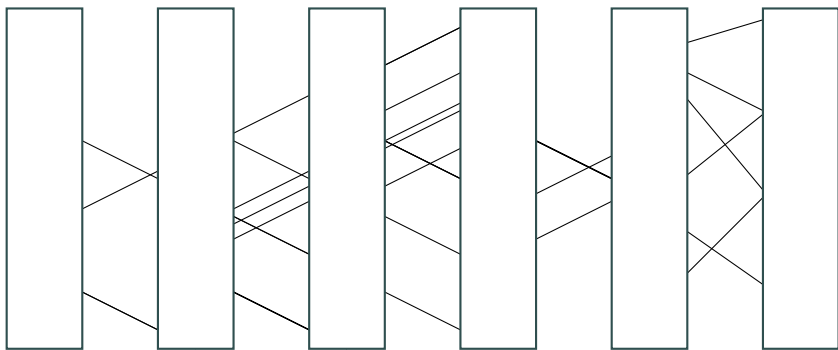




$$\text{cost} = \min_w \{ \text{cost}(w) + \text{cost}(\text{"old solution"}) \}.$$

Without the promise...





A random partition is good with high probability.

If the vertices of a simple graph G on k edges are colored independently and uniformly at random with

$\sqrt{8k}$ colors,

then the probability that $E(G)$ is properly colored is at least

$$2^{-\sqrt{2k}}.$$

Kernelize first!

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Perform Dynamic Programming.

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Perform Dynamic Programming.

$n_1 \cdots n_p$

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Perform Dynamic Programming.

$$n_1 \cdots n_p \leq n^{\sqrt{8k}}$$

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Perform Dynamic Programming.

$$n_1 \cdots n_p \leq n^{\sqrt{8k}} \leq k^{\sqrt{8k}},$$

Kernelize first!

Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.

Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Perform Dynamic Programming.

$n_1 \cdots n_p \leq n^{\sqrt{8k}} \leq k^{\sqrt{8k}}$, we're done!.

Kernelize first!
Polynomial time, $n = \mathcal{O}(k^2)$.

Randomly color with $\sqrt{8k}$ colors.
Polynomial time, good with probability at least $2^{-\sqrt{2k}}$.

Perform Dynamic Programming.
 $n_1 \cdots n_p \leq n^{\sqrt{8k}} \leq k^{\sqrt{8k}}$, we're done!.

Repeat $2^{\sqrt{2k}}$ times.
Probability of error is at most:

$$\left(1 - \frac{1}{2^{\sqrt{2k}}}\right)^{2^{\sqrt{2k}}}$$

Repeat $2^{\sqrt{2k}}$ times.
Probability of error is at most:

$$\left(1 - \frac{1}{2^{\sqrt{2k}}}\right)^{2^{\sqrt{2k}}} \leq 1/e$$

Overall running time:

$$2^{\sqrt{2k}} \cdot 2^{\log k \sqrt{8k}} \cdot n^{\mathcal{O}(1)}$$

Other Variants

Feedback Vertex Set Problems on general (di)graphs.
Dominating Set and other problems on (bipartite) tournaments.