Kernel Bounds for Path and Cycle Problems

Bart M. P. Jansen

Joint work with
Hans L. Bodlaender & Stefan Kratsch

Universiteit Utrecht

September 8th 2011, Saarbrucken

Path and Cycle problems

Path and Cycle problems

Long Path

- Given G and an integer ℓ, does G contain a path on at least ℓ vertices?

Long Cycle

- Given G and an integer ℓ, does G contain a cycle on at least ℓ vertices?

Disjoint Paths

- Given G and pairs of vertices $(s_1, t_1), \dots, (s_\ell, t_\ell)$, are there vertex-disjoint paths connecting each s_i to t_i?

Disjoint Cycles

- Given G and an integer ℓ, are there ℓ vertex-disjoint simple cycles in G?

Background

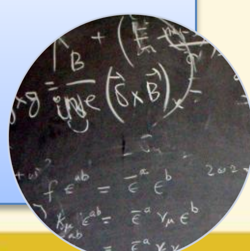
- Various path and cycle problems have been important to the development of parameterized complexity

Background

- Various path and cycle problems have been important to the development of parameterized complexity

- **Disjoint Paths** lies at the heart of the Graph Minors algorithm
- **Long Path** was one of the first problems known to be fixed-parameter tractable
- **Long Path** was one of the main motivations for the kernel lower-bound framework
- **Disjoint Cycles** inspired one of the first non-trivial compositions

Theoretical



- **Long Path** has applications in computational biology
- …

Practical

Previous results

Previous results

- Many recent developments in FPT algorithms

Previous results

- Many recent developments in FPT algorithms
  - Disjoint Paths: improvements to the Unique Linkage Theorem for planar graphs [AdlerKKLSThilikos@ICALP'11]

Previous results

- Many recent developments in FPT algorithms
  - Disjoint Paths: improvements to the Unique Linkage Theorem for planar graphs [AdlerKKLSThilikos@ICALP'11]
  - k-Path continues to inspire new algorithmic techniques [BjörklundHKK'10]

Previous results

- Many recent developments in FPT algorithms
  - Disjoint Paths: improvements to the Unique Linkage Theorem for planar graphs [AdlerKKLSThilikos@ICALP'11]
  - k-Path continues to inspire new algorithmic techniques [BjörklundHKK'10]

Table 1. k-path in time $O^*(f(k))$

$k!$		Monien [29]	12.6^k		Chen *et al.* [6]
$k!2^k$		Bodlaender [5]	4^k	r	Chen *et al.* [6]
5.44^k	r	Alon *et al.* [1]	2.83^k	r	Koutis (2008) [21]
c^k		$c > 8000$, Alon *et al.* [1]	2^k	r	Williams [37]
16^k		Kneis *et al.* [25]	1.66^k	r	*this paper*

Previous results

- Many recent developments in FPT algorithms
  - Disjoint Paths: improvements to the Unique Linkage Theorem for planar graphs [AdlerKKLSThilikos@ICALP'11]
  - k-Path continues to inspire new algorithmic techniques [BjörklundHKK'10]

Previous results

- Many recent developments in FPT algorithms
  - Disjoint Paths: improvements to the Unique Linkage Theorem for planar graphs [AdlerKKLSThilikos@ICALP'11]
  - k-Path continues to inspire new algorithmic techniques [BjörklundHKK'10]
- Natural parameterizations k-Path, k-Disjoint Paths, k-Disjoint Cycles are fixed-parameter tractable but do not admit polynomial kernels unless NP ⊆ coNP/poly [BodlaenderDFH@ICALP'08, BodlaenderTY@ESA'09, Robertson&Seymour]

Previous results

- Many recent developments in FPT algorithms
  - Disjoint Paths: improvements to the Unique Linkage Theorem for planar graphs [AdlerKKLSThilikos@ICALP'11]
  - k-Path continues to inspire new algorithmic techniques [BjörklundHKK'10]
- Natural parameterizations k-Path, k-Disjoint Paths, k-Disjoint Cycles are fixed-parameter tractable but do not admit polynomial kernels unless NP ⊆ coNP/poly [BodlaenderDFH@ICALP'08, BodlaenderTY@ESA'09, Robertson&Seymour]
  - For k-Path: not even a polynomial kernel on connected planar graphs [ChenFM@CiE'09]

Preprocessing for path & cycle problems

- Even though natural parameterizations do not admit polynomial kernels, we might still benefit from preprocessing

Preprocessing for path & cycle problems

- Even though natural parameterizations do not admit polynomial kernels, we might still benefit from preprocessing
- How to guide the search for good reduction rules?

Preprocessing for path & cycle problems

- Even though natural parameterizations do not admit polynomial kernels, we might still benefit from preprocessing
- How to guide the search for good reduction rules?
  - Non-standard parameters!

Preprocessing for path & cycle problems

- Even though natural parameterizations do not admit polynomial kernels, we might still benefit from preprocessing
- How to guide the search for good reduction rules?
  - Non-standard parameters!

- One example known:

Hamiltonian Cycle parameterized by Max Leaf Number has a kernel with 5.75k vertices [FellowsLMMRS@CiE'07]

Our results

Our results

**Long Path, Long Cycle,
Disjoint Paths, Disjoint Cycles**

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- Admit polynomial kernels parameterized by vertex-deletion distance to a Cluster graph

Hamiltonian Path & Hamiltonian Cycle

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

Our results

Long Path, Long Cycle,
Disjoint Paths, Disjoint Cycles

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- ance to a Cluster graph

Generalizes kernel for Hamiltonian Cycle by [FellowsLMMRS@CIE'07]

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

Our results

**Long Path, Long Cycle,
Disjoint Paths, Disjoint Cycles**

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- Admit polynomial kernels parameterized by vertex-deletion distance to a Cluster graph

Hamiltonian Path & Hamiltonian Cycle

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

Our results

Long Path, Long Cycle, Disjoint Paths, Disjoint Cycles

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- Admit polynomial kernels parameterized by vertex-deletion distance to a Cluster graph

Hamiltonian Path & Hamiltonian Cycle

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

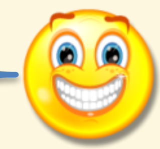
Our results

Long Path, Long Cycle, Disjoint Paths, Disjoint Cycles

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- Admit polynomial kernels parameterized by vertex-deletion distance to a Cluster graph



Hamiltonian Path & Hamiltonian Cycle

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

Our results

Long Path, Long Cycle, Disjoint Paths, Disjoint Cycles

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- Admit polynomial kernels parameterized by vertex-deletion distance to a Cluster graph

Hamiltonian Path & Hamiltonian Cycle

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

Our results

Long Path, Long Cycle, Disjoint Paths, Disjoint Cycles

- Admit $O(k^2)$-vertex kernels parameterized by Vertex Cover Number
- Admit polynomial kernels parameterized by Max Leaf Number

Long Path & Long Cycle

- Admit polynomial kernels parameterized by vertex-deletion distance to a Cluster graph

Hamiltonian Path & Hamiltonian Cycle

- Do not admit polynomial kernels parameterized by vertex-deletion distance to an outerplanar graph

Path problems with Forbidden Pairs

- First study of parameterized complexity: para-NP-completeness, FPT, W[1]-hardness and kernel lower-bounds

	vc(G)	vc(H)	tw(H)	tw(G ∪ H)	vc(G ∪ H)
$s-t$ Path F.P.	W[1]-hard	FPT	Para-NP-c	FPT	No poly
Shortest $s-t$ Path F.P.	W[1]-hard	FPT	Para-NP-c	FPT	No poly
Longest $s-t$ Path F.P.	W[1]-hard	Para-NP-c	Para-NP-c	FPT	No poly
Longest Path F.P.	W[1]-hard	Para-NP-c	Para-NP-c	FPT	No poly

Quadratic-vertex kernel parameterized by Vertex Cover #

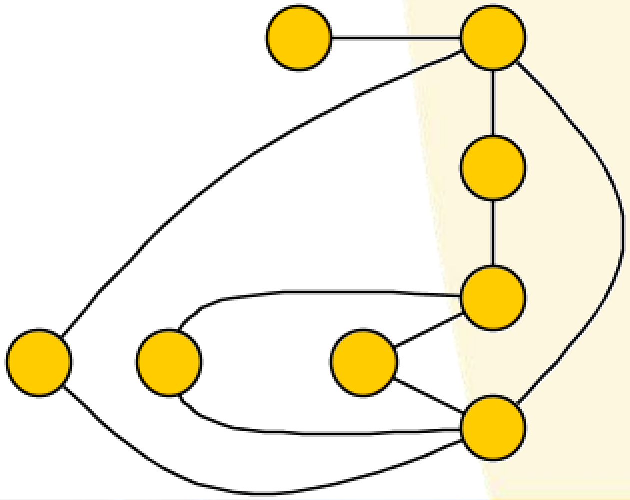
LONG CYCLE

Quadratic-vertex kernel for Long Cycle by Vertex Cover

- Input: Graph G, vertex cover X of G, integer ℓ
- Question: Does G have a cycle on at least ℓ vertices?

Quadratic-vertex kernel for Long Cycle by Vertex Cover

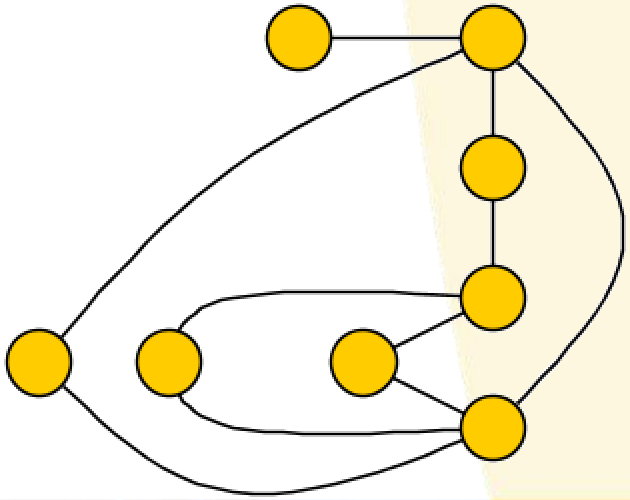
- Input: Graph G, vertex cover X of G, integer ℓ
- Question: Does G have a cycle on at least ℓ vertices?



Quadratic-vertex kernel for Long Cycle by Vertex Cover

- Input: Graph G, vertex cover X of G, integer ℓ
- Question: Does G have a cycle on at least ℓ vertices?

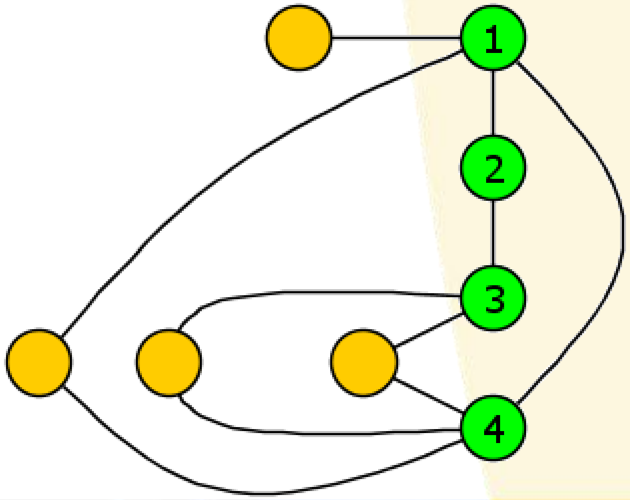
- Example for $\ell = 6$



Quadratic-vertex kernel for Long Cycle by Vertex Cover

- Input: Graph G, vertex cover X of G, integer ℓ
- Question: Does G have a cycle on at least ℓ vertices?

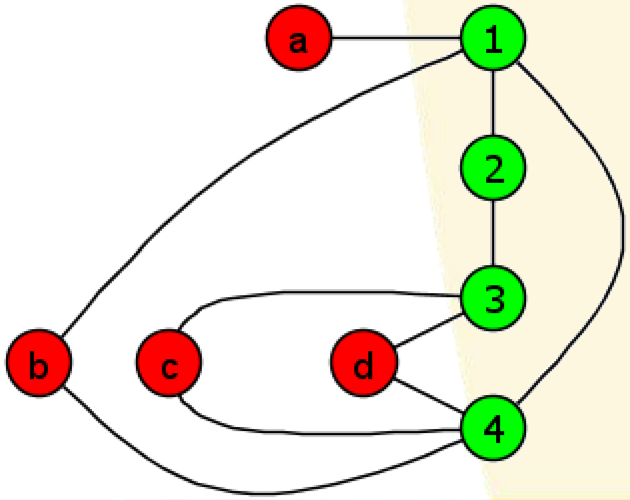
- Example for $\ell = 6$



Quadratic-vertex kernel for Long Cycle by Vertex Cover

- Input: Graph G, vertex cover X of G, integer ℓ
- Question: Does G have a cycle on at least ℓ vertices?

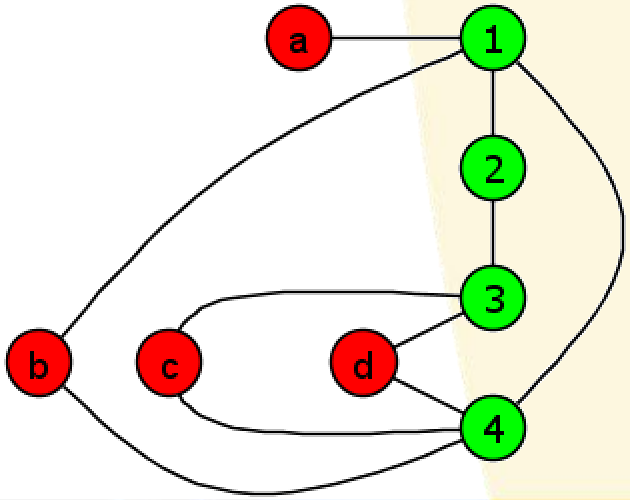
- Example for $\ell = 6$



Quadratic-vertex kernel for Long Cycle by Vertex Cover

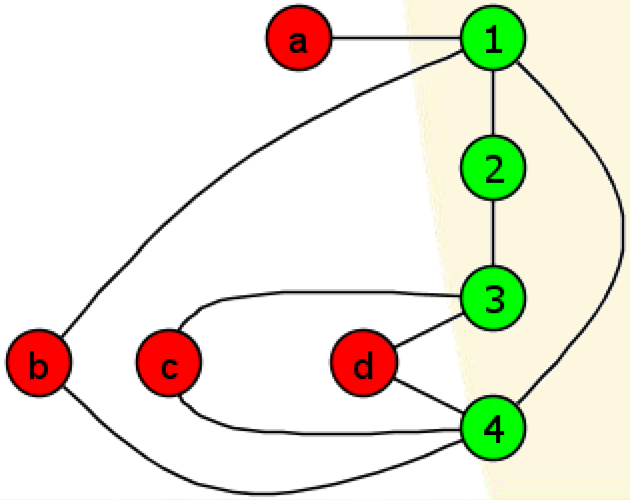
- Input: Graph G, vertex cover X of G, integer ℓ
- Question: Does G have a cycle on at least ℓ vertices?
  - Assume $\ell > 4$ (otherwise, solve by brute force)

- Example for $\ell = 6$



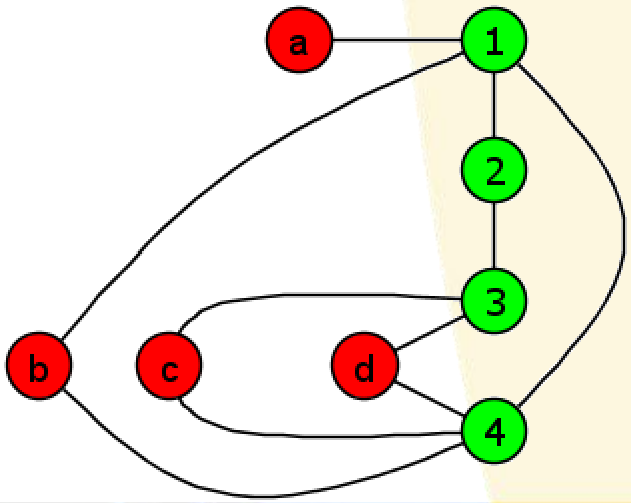
Reduction algorithm

- Bipartite auxiliary graph H = (R ∪ B, E)



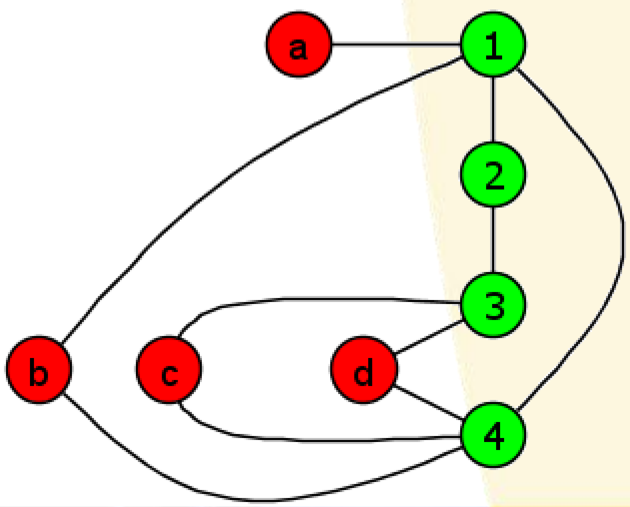
Reduction algorithm

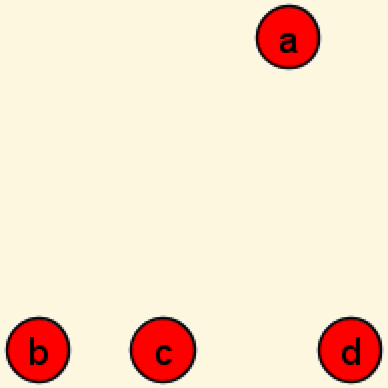
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X



Reduction algorithm

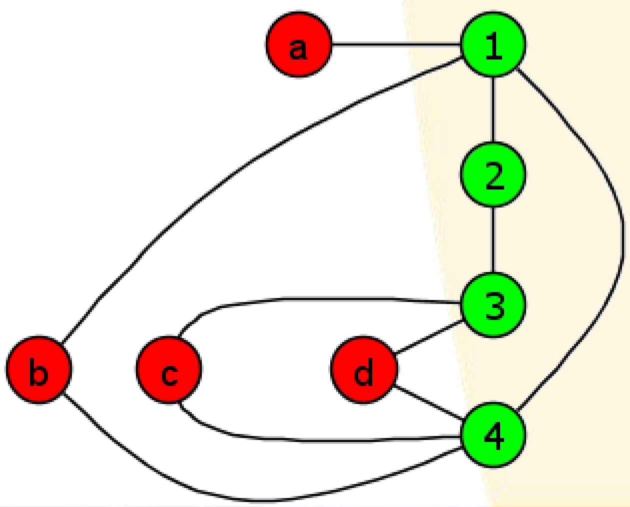
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X

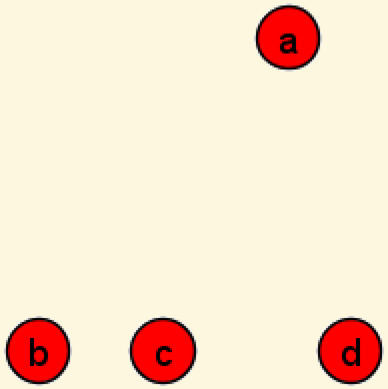




Reduction algorithm

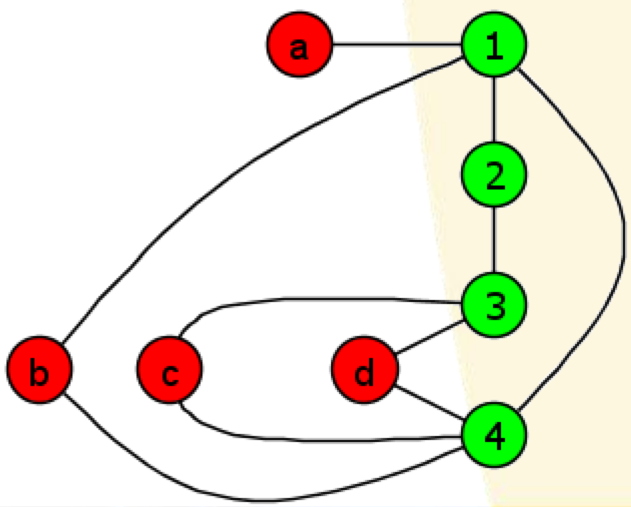
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X

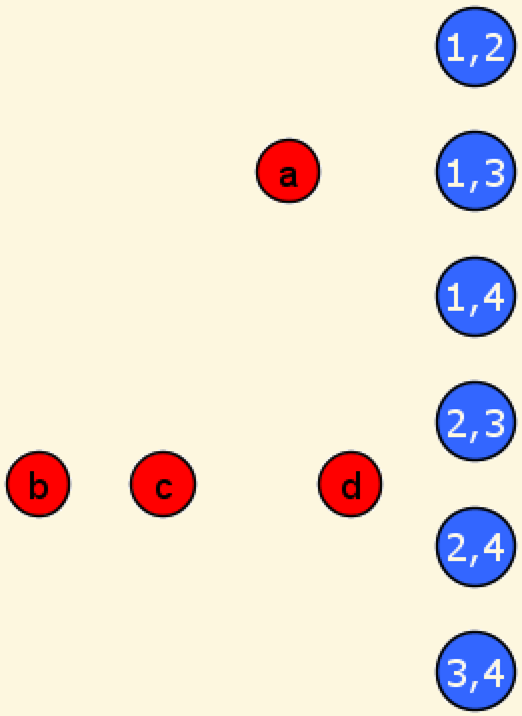




Reduction algorithm

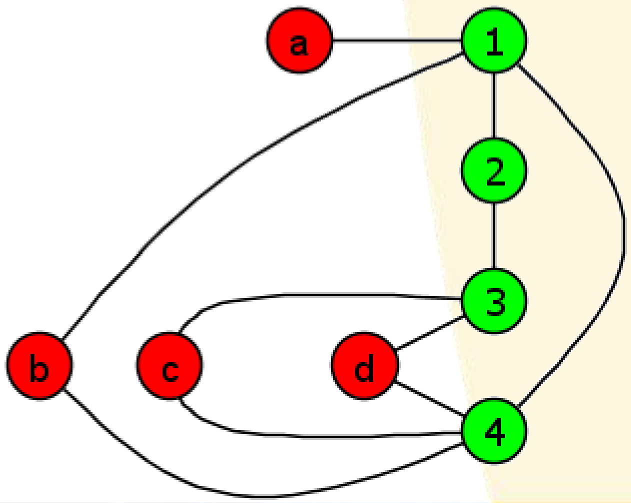
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X

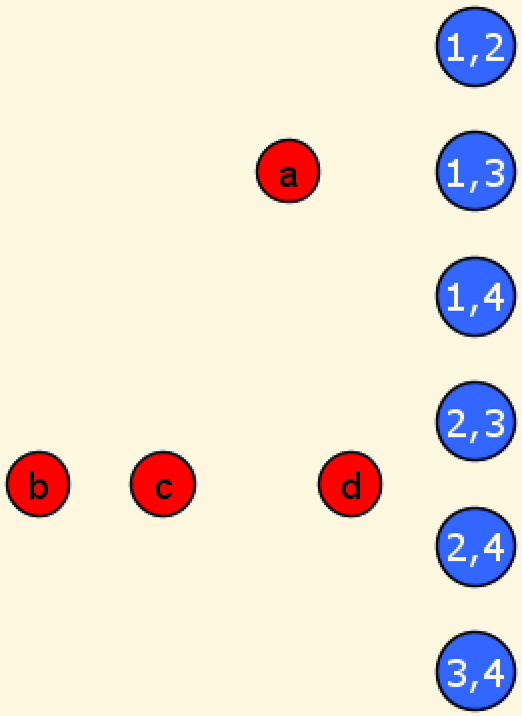




Reduction algorithm

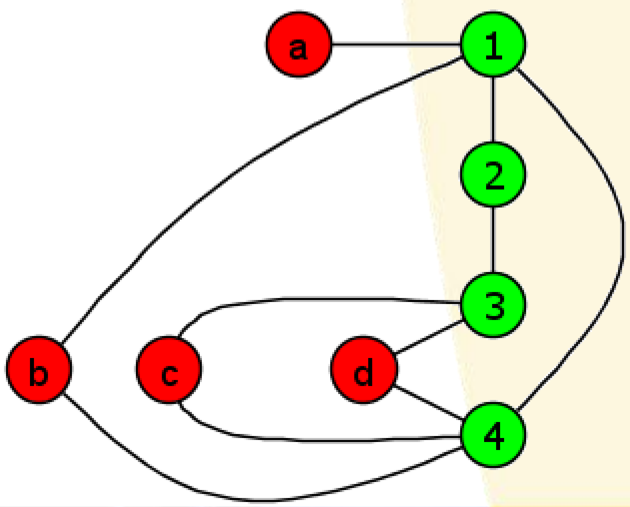
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X

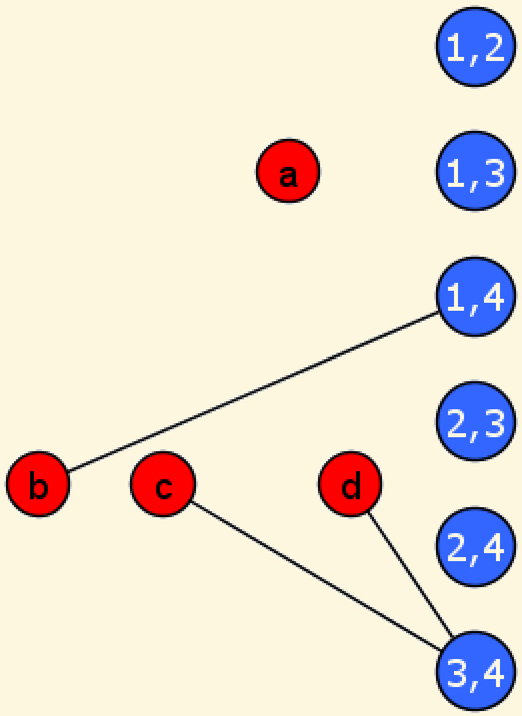




Reduction algorithm

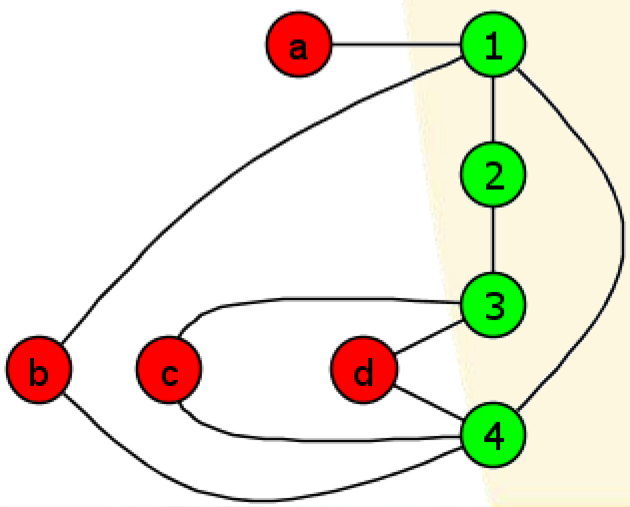
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X

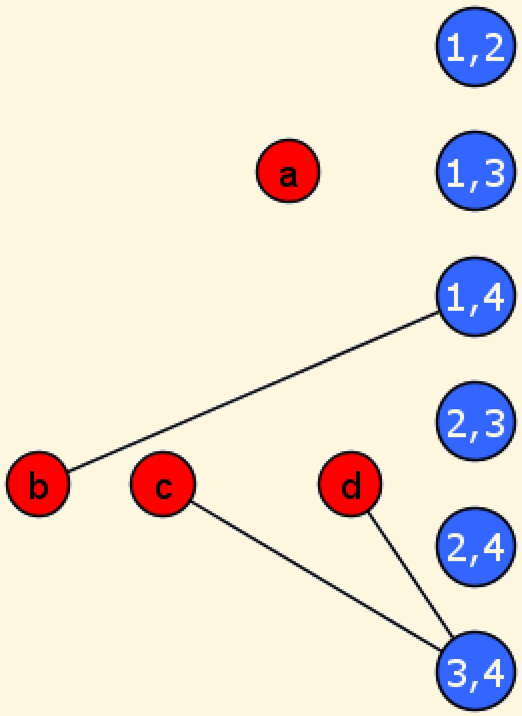




Reduction algorithm

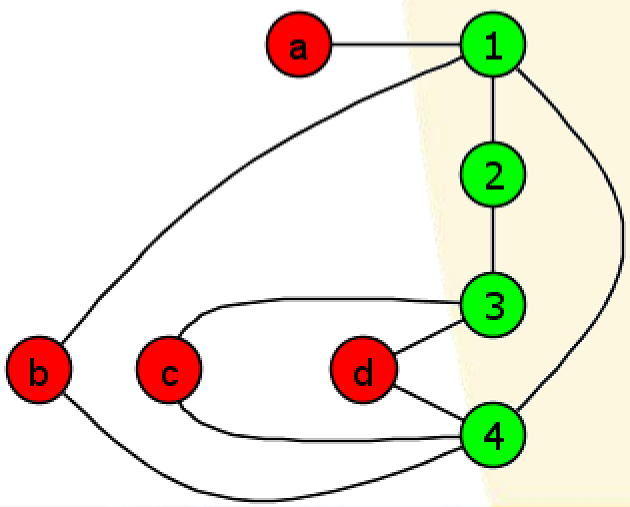
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X
- Compute maximum matching in H

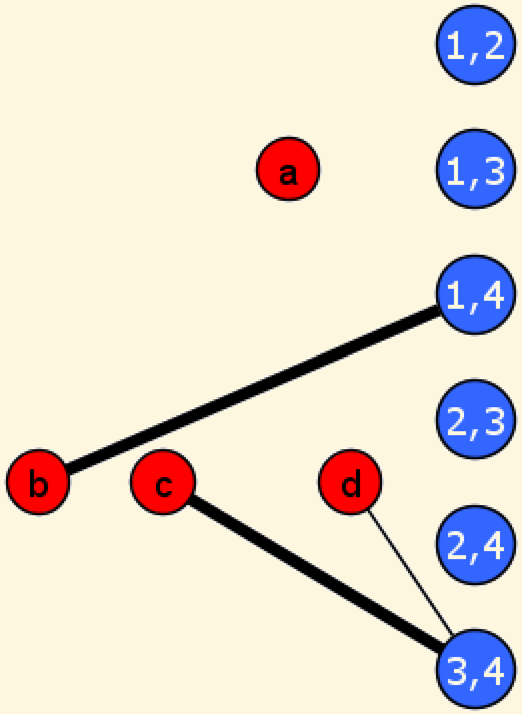




Reduction algorithm

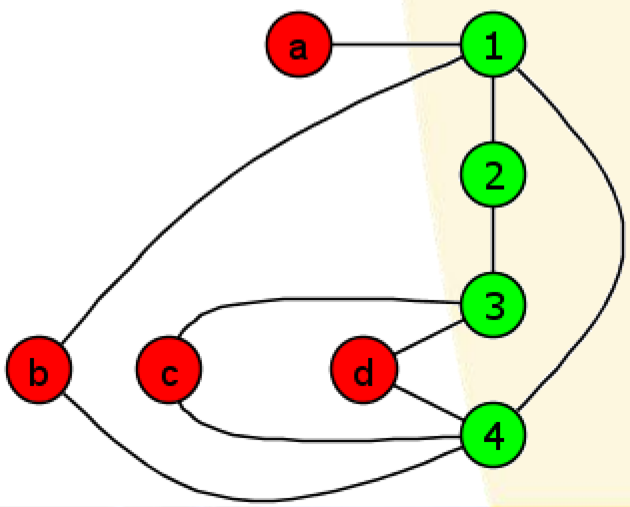
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X
- Compute maximum matching in H

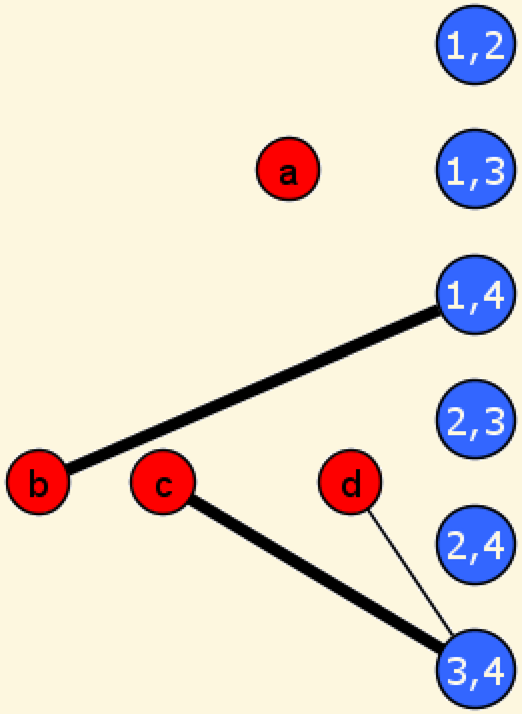




Reduction algorithm

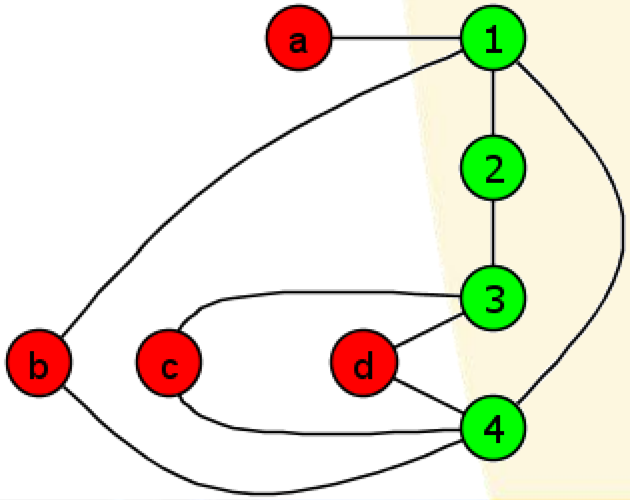
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X
- Compute maximum matching in H
  - Let R_U be the unsaturated red vertices

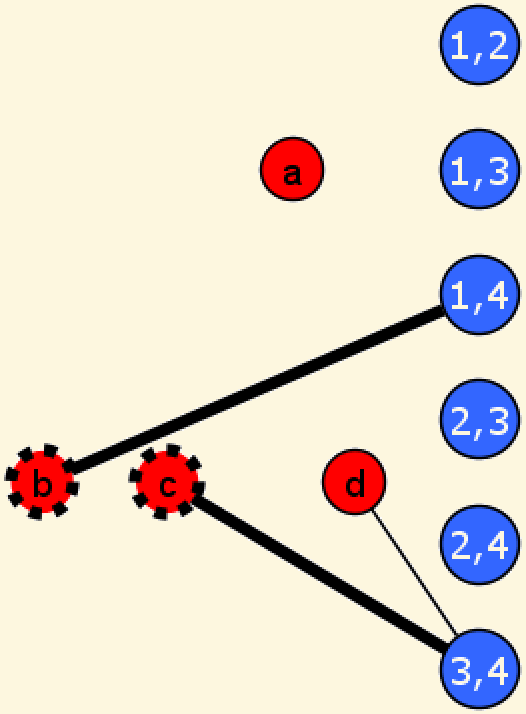




Reduction algorithm

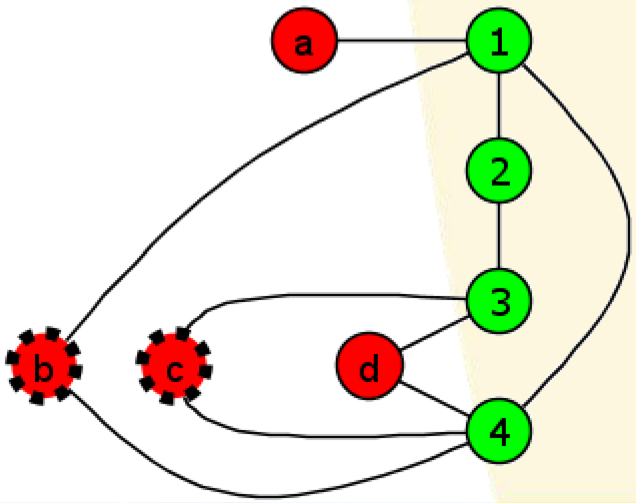
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X
- Compute maximum matching in H
  - Let R_U be the unsaturated red vertices

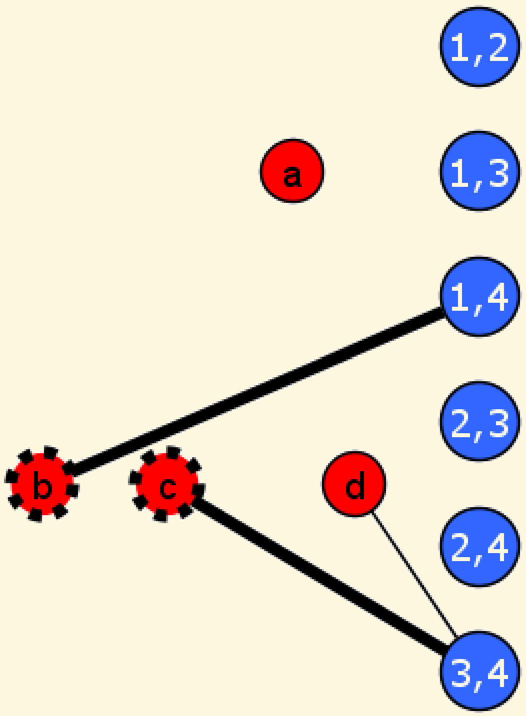




Reduction algorithm

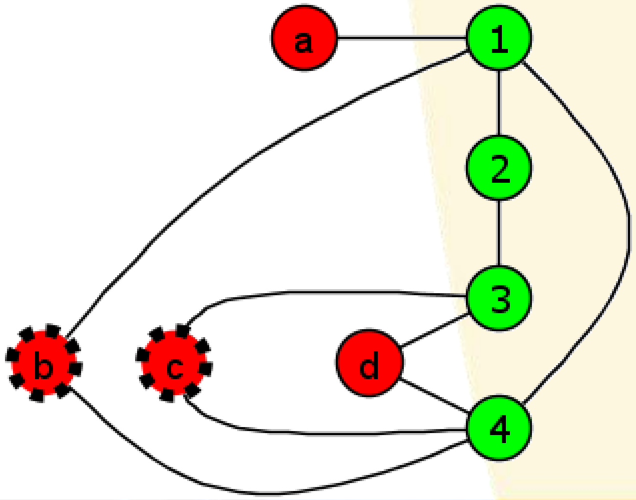
- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X
- Compute maximum matching in H
  - Let R_U be the unsaturated red vertices

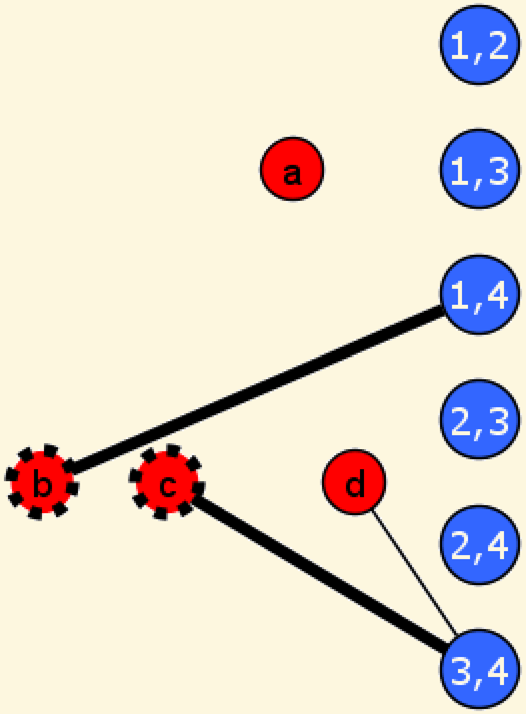




Reduction algorithm

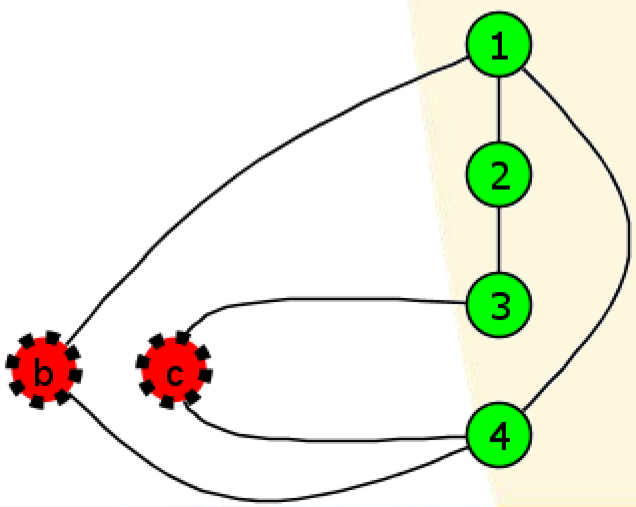
- Bipartite auxiliary graph $H = (R \cup B, E)$
  - Red vertices are $V(G) \setminus X$
  - Blue vertex $v(p,q)$ for each pair $p,q \in X$
    - $v(p,q)$ adjacent to $N(p) \cap N(q) \setminus X$
- Compute maximum matching in H
  - Let R_U be the unsaturated red vertices
- Output $G - R_U$ with $\leq |X| + |X|^2$ vertices

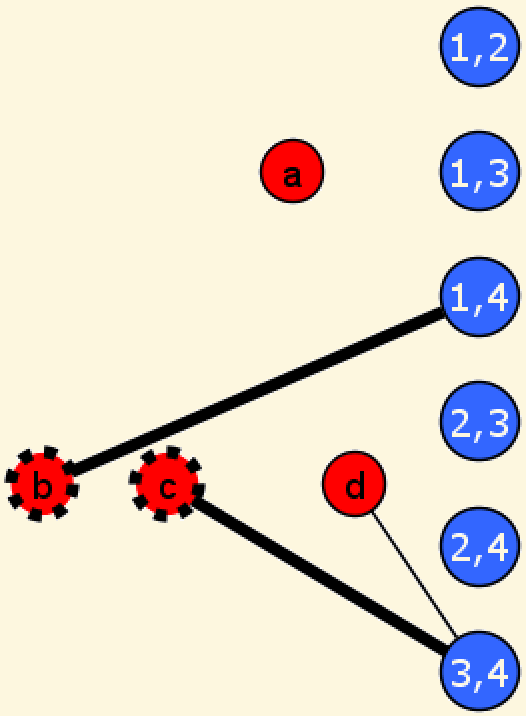




Reduction algorithm

- Bipartite auxiliary graph H = (R ∪ B, E)
  - Red vertices are V(G) \ X
  - Blue vertex v(p,q) for each pair p,q ∈ X
    - v(p,q) adjacent to N(p)∩N(q) \ X
- Compute maximum matching in H
  - Let R_U be the unsaturated red vertices
- Output G – R_U with ≤ |X| + |X|² vertices





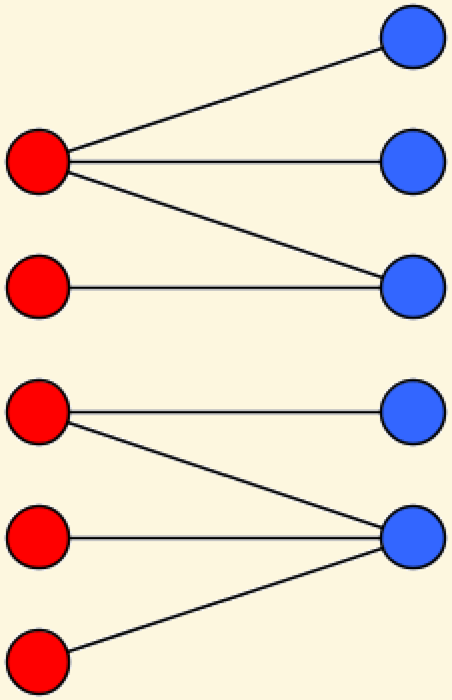
Property of maximum matchings

Property of maximum matchings

- Let H = (R ∪ B, E) be a bipartite graph

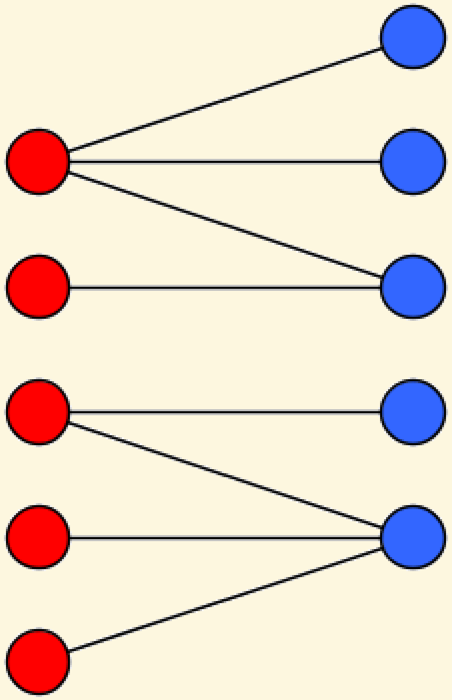
Property of maximum matchings

- Let H = (R ∪ B, E) be a bipartite graph



Property of maximum matchings

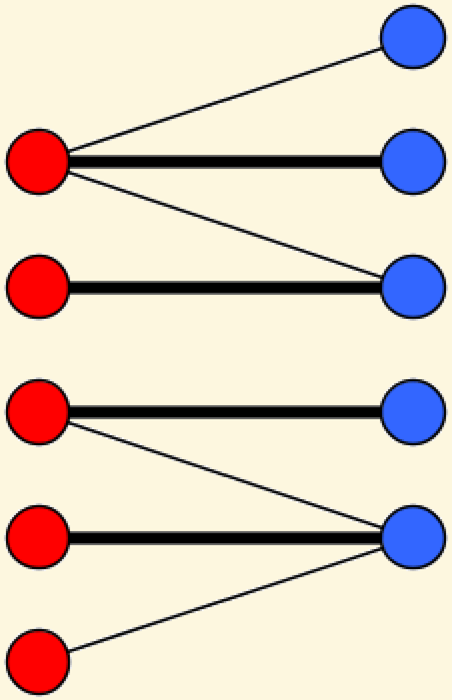
- Let H = (R ∪ B, E) be a bipartite graph
- Let M be a maximum matching in H



Universiteit Utrecht

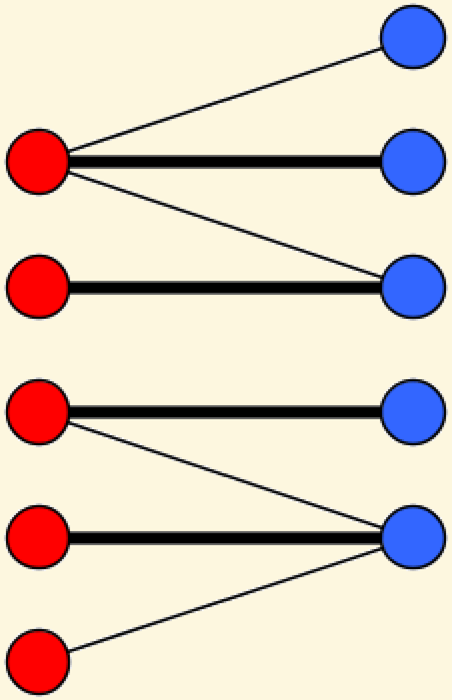
Property of maximum matchings

- Let H = (R ∪ B, E) be a bipartite graph
- Let M be a maximum matching in H



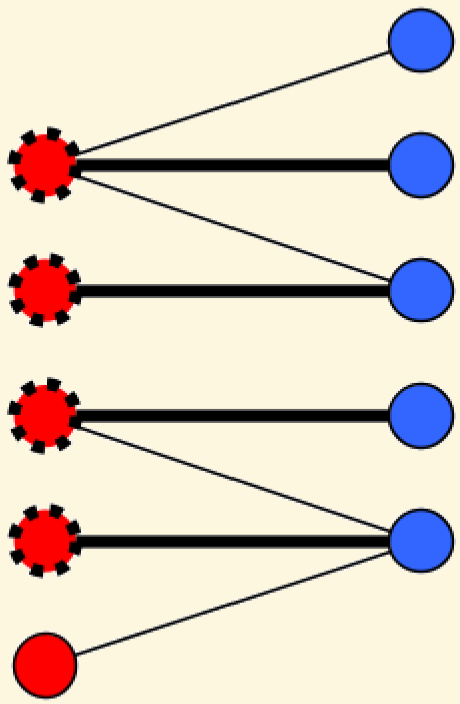
Property of maximum matchings

- Let H = (R ∪ B, E) be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M



Property of maximum matchings

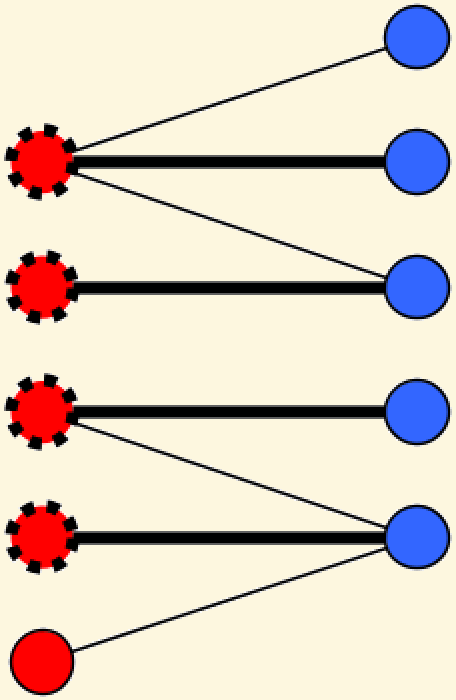
- Let H = (R ∪ B, E) be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M



Property of maximum matchings

- Let $H = (R \cup B, E)$ be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M

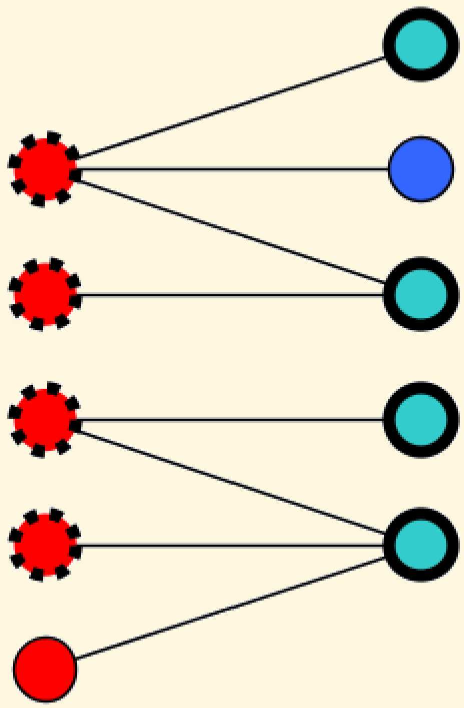
Theorem. For all $B' \subseteq B$:
if H has a matching saturating B',
then $H - R_U$ has a matching saturating B'.



Property of maximum matchings

- Let H = (R ∪ B, E) be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M

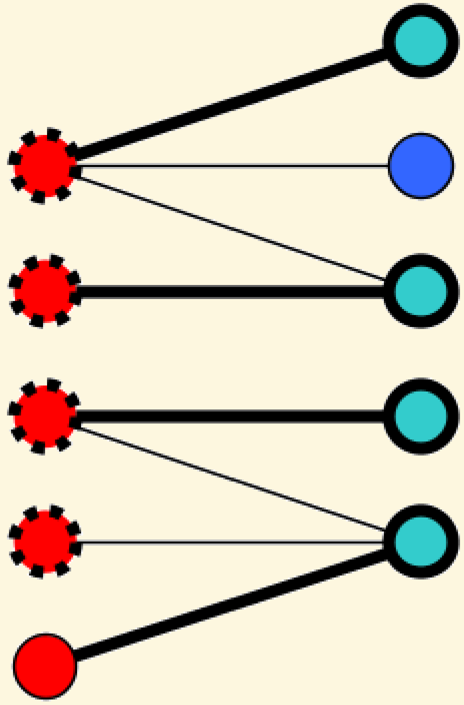
Theorem. For all B’ ⊆ B:
if H has a matching saturating B’,
then H – R_U has a matching saturating B’.



Property of maximum matchings

- Let $H = (R \cup B, E)$ be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M

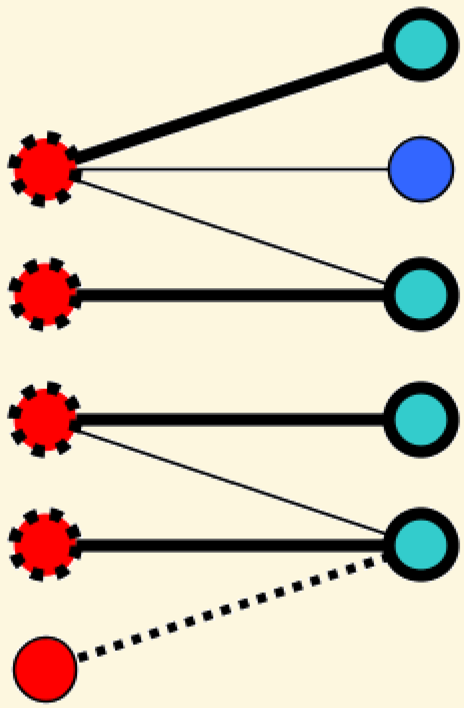
Theorem. For all $B' \subseteq B$:
if H has a matching saturating B',
then $H - R_U$ has a matching saturating B'.



Property of maximum matchings

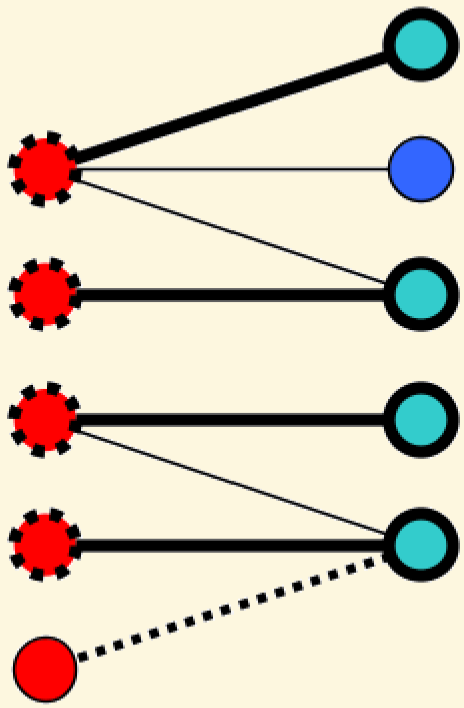
- Let $H = (R \cup B, E)$ be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M

Theorem. For all $B' \subseteq B$:
if H has a matching saturating B',
then $H - R_U$ has a matching saturating B'.



Property of maximum matchings

- Let H = (R ∪ B, E) be a bipartite graph
- Let M be a maximum matching in H
- Let R_U be vertices of R **not** saturated by M



Theorem. For all B' ⊆ B:
if H has a matching saturating B',
then H – R_U has a matching saturating B'.

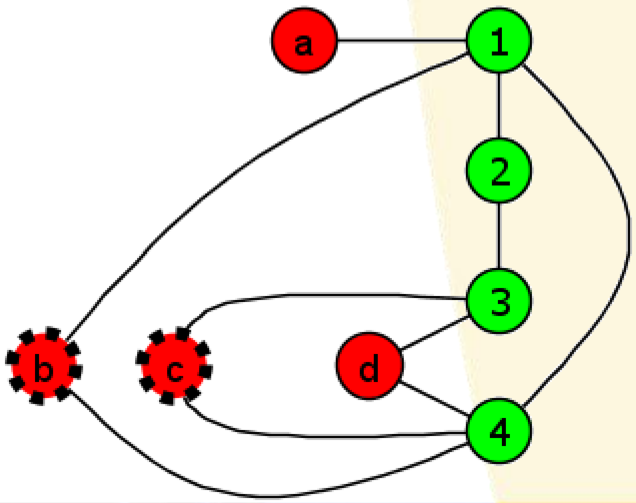
- Proof using augmenting paths

Correctness (I)

- G has a cycle of length ℓ ⇔ G – R_U has a cycle of length ℓ

Correctness (I)

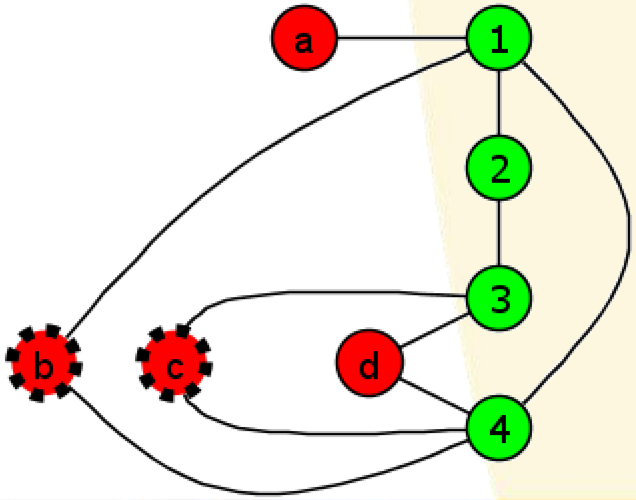
- G has a cycle of length ℓ ⇔ G – R_U has a cycle of length ℓ
- (🡐) Trivial since cycle in subgraph gives cycle in G



Correctness (I)

- G has a cycle of length ℓ ⇔ G – R_U has a cycle of length ℓ

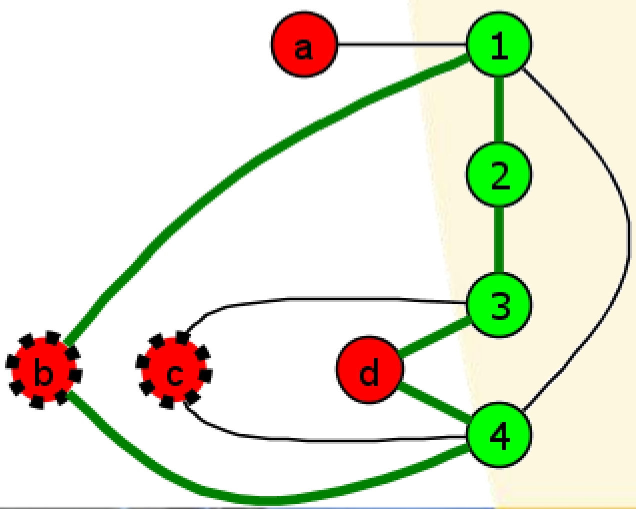
- (←) Trivial since cycle in subgraph gives cycle in G
- (→) Proof using a matching property
  - Suppose G has a cycle C of length $\ell > 4$



Correctness (I)

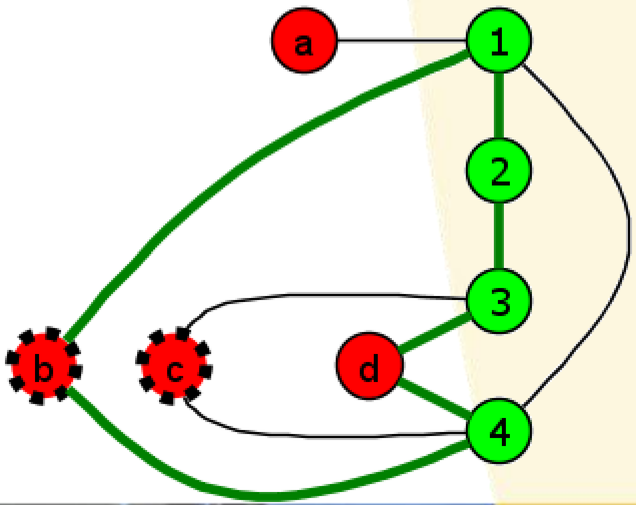
- G has a cycle of length $\ell \Leftrightarrow$ G – R_U has a cycle of length ℓ

- (←) Trivial since cycle in subgraph gives cycle in G
- (→) Proof using a matching property
  - Suppose G has a cycle C of length $\ell > 4$



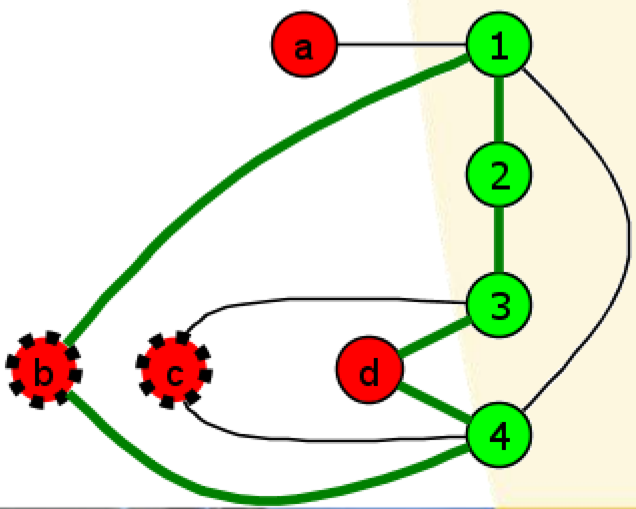
Correctness (II)

- All (green) vertices and edges of G[X] are still present



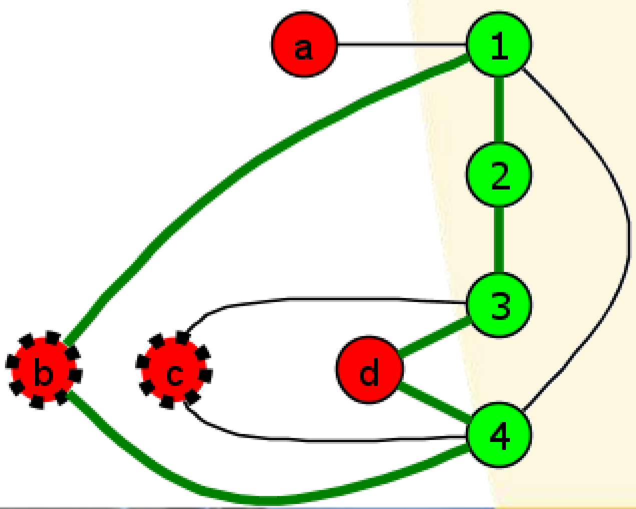
Correctness (II)

- All (green) vertices and edges of G[X] are still present
- Red vertices in G-X are used to connect two green vertices in X



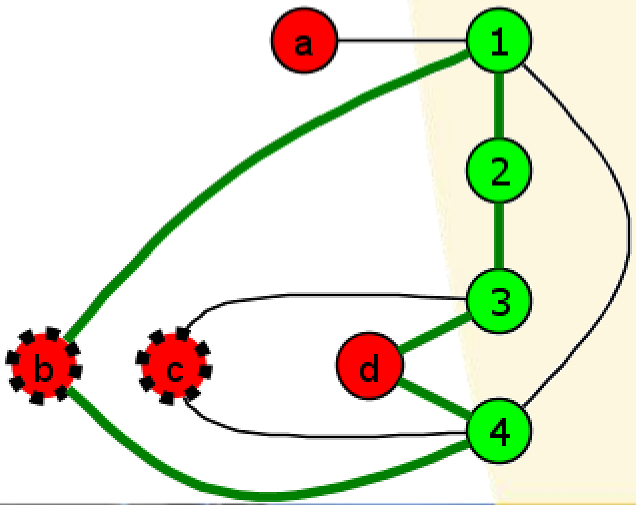
Correctness (II)

- All (green) vertices and edges of G[X] are still present
- Red vertices in G-X are used to connect two green vertices in X
- Subpath (g_1, r, g_2) of C is an **indirect connection**



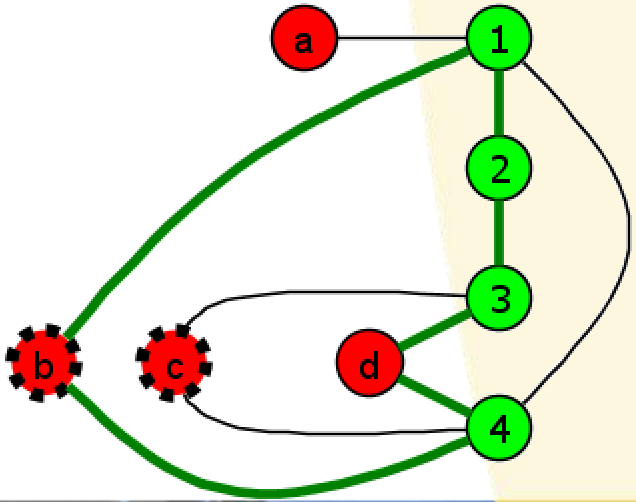
Correctness (II)

- All (green) vertices and edges of G[X] are still present
- Red vertices in G-X are used to connect two green vertices in X
- Subpath (g_1, r, g_2) of C is an **indirect connection**
  - $r \in N(g_1) \cap N(g_2) \setminus X$



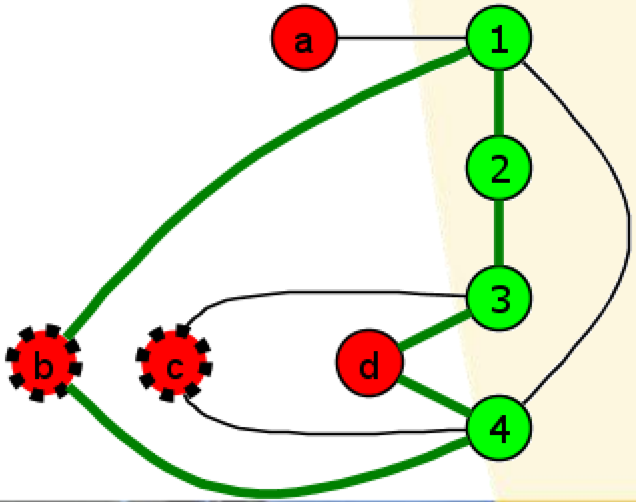
Correctness (II)

- All (green) vertices and edges of G[X] are still present
- Red vertices in G-X are used to connect two green vertices in X
- Subpath (g_1, r, g_2) of C is an **indirect connection**
  - $r \in N(g_1) \cap N(g_2) \setminus X$
- Find red vertices in $R \setminus R_U$ to replace all indirect connections



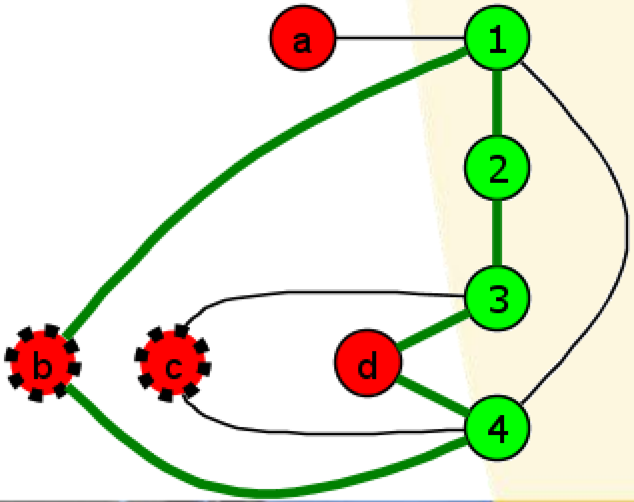
Correctness (III)

- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$



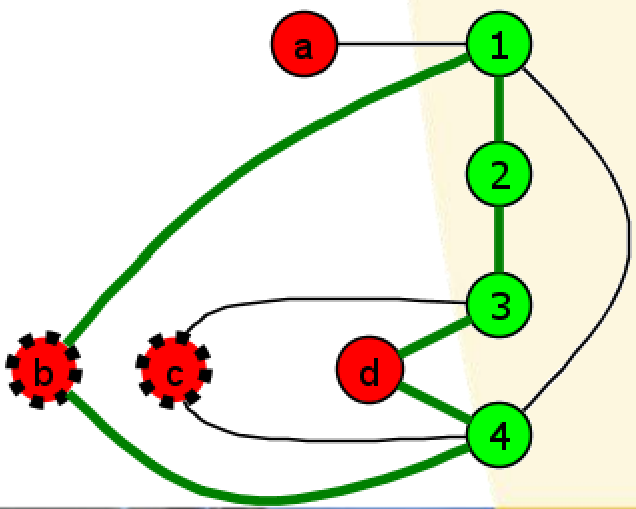
Correctness (III)

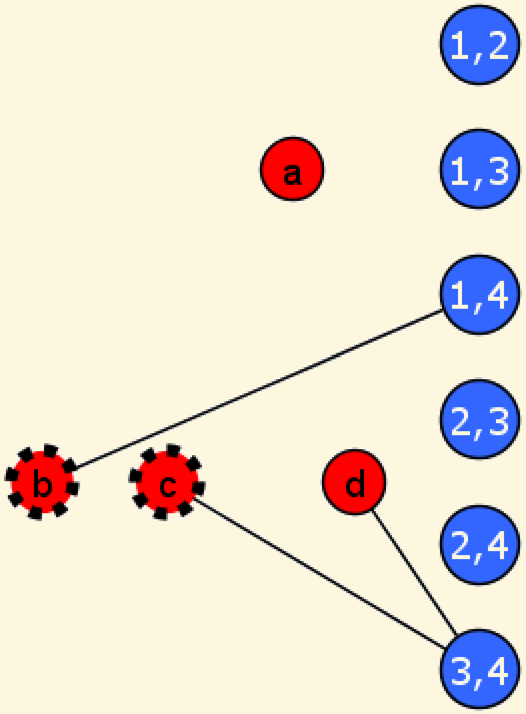
- No two connections (g_1, r, g_2) and (g_1, r′, g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):



Correctness (III)

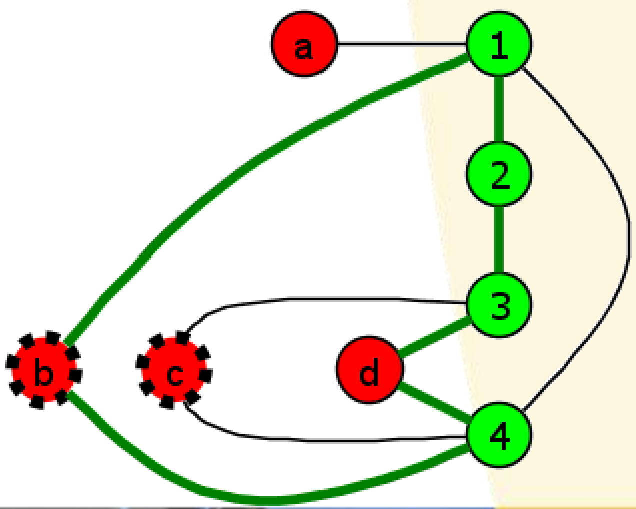
- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):

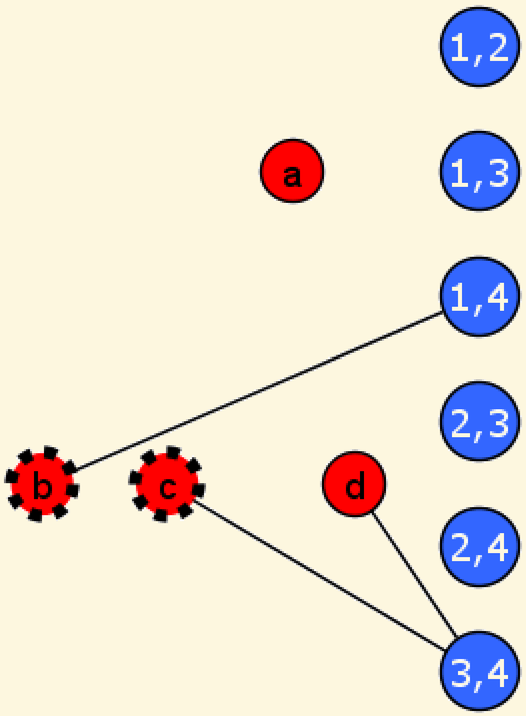




Correctness (III)

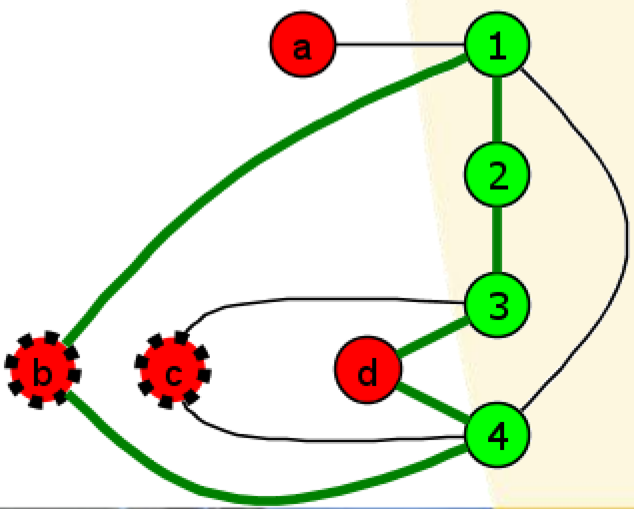
- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match $v(g_1,g_2)$ to r in H

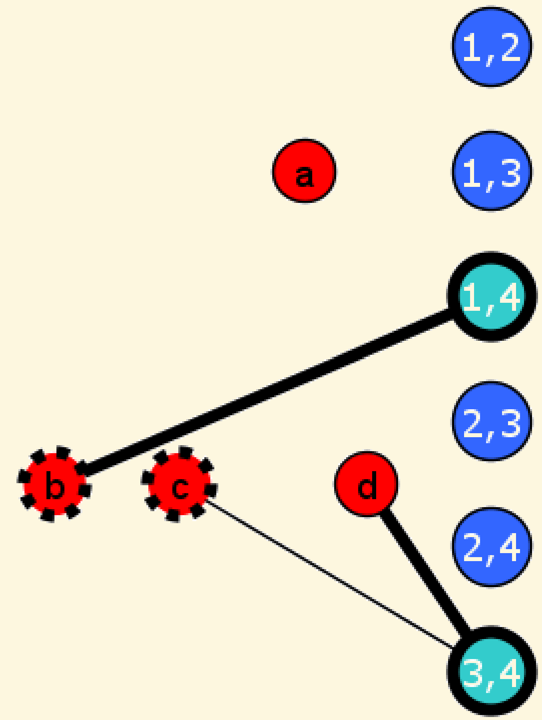




Correctness (III)

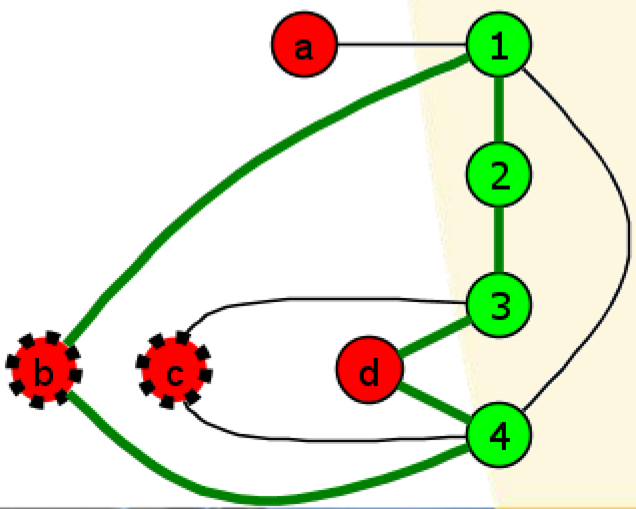
- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match $v(g_1,g_2)$ to r in H

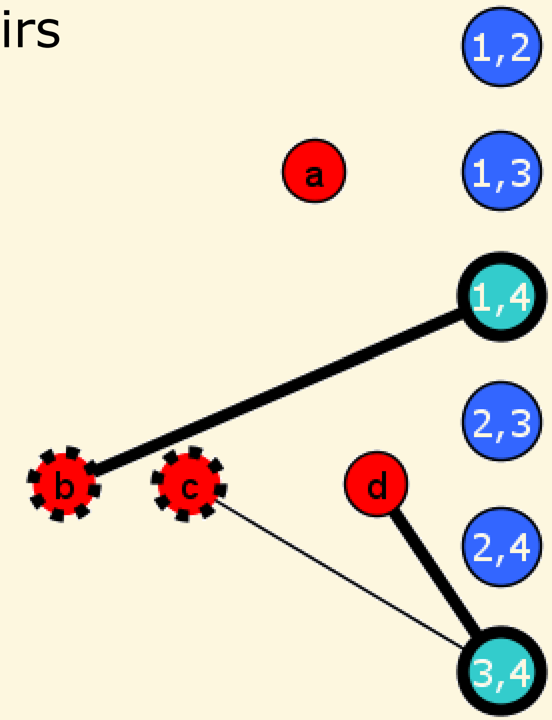




Correctness (III)

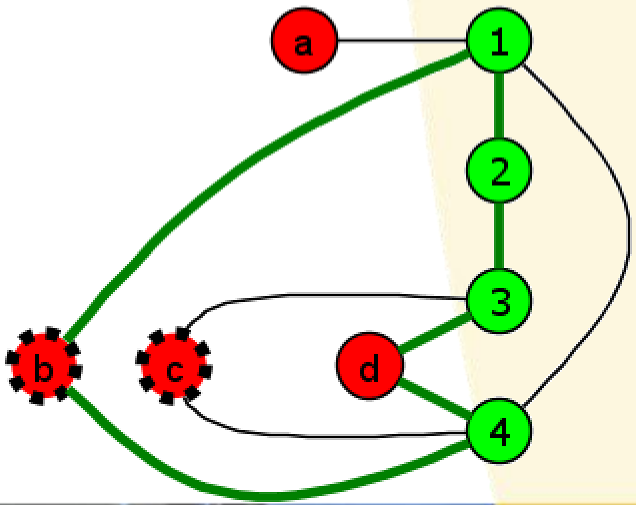
- No two connections (g_1, r, g_2) and (g_1, r′, g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match v(g_1,g_2) to r in H
  - matching in H saturating all connected pairs

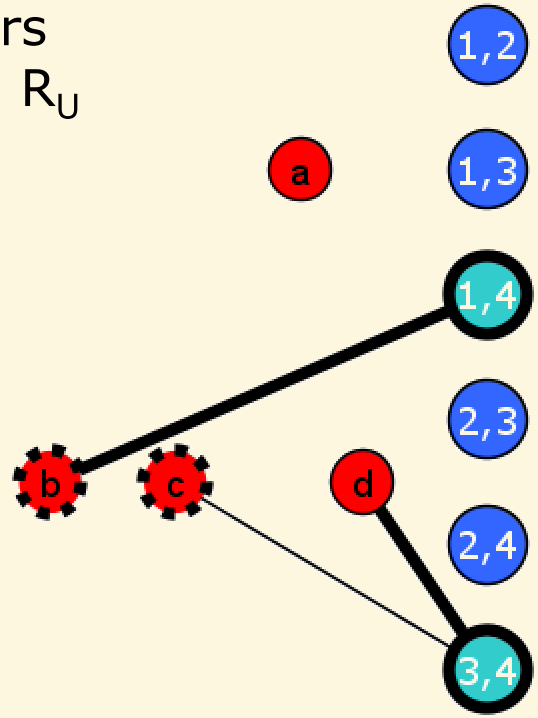




Correctness (III)

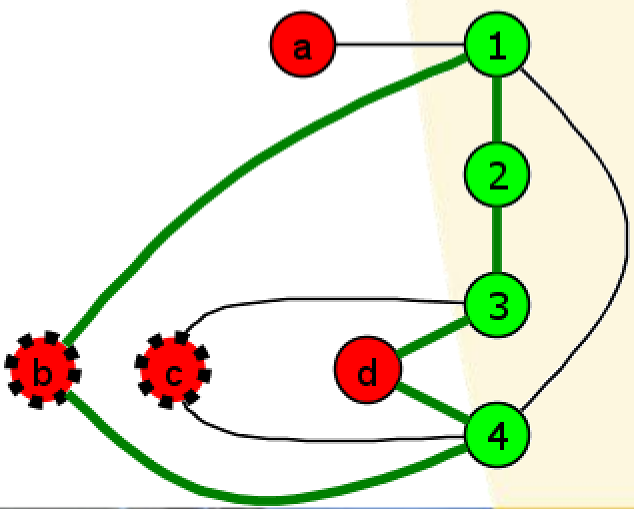
- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match $v(g_1,g_2)$ to r in H
  - matching in H saturating all connected pairs
- By matching property: exists matching in $H - R_U$ saturating all connected pairs

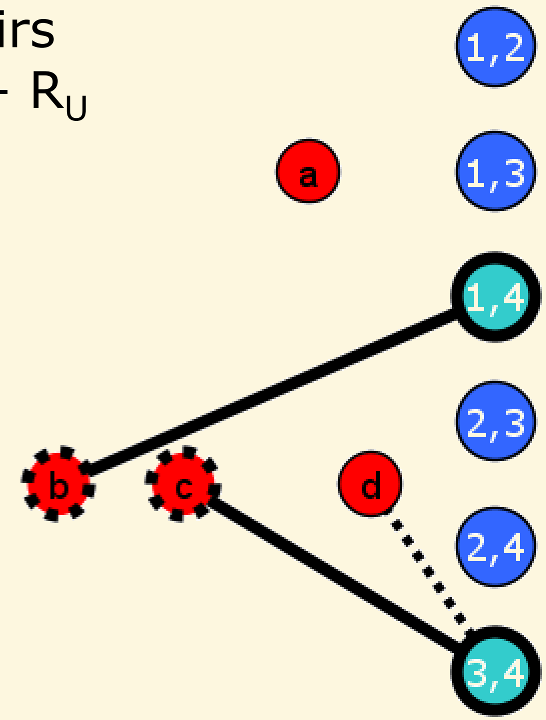




Correctness (III)

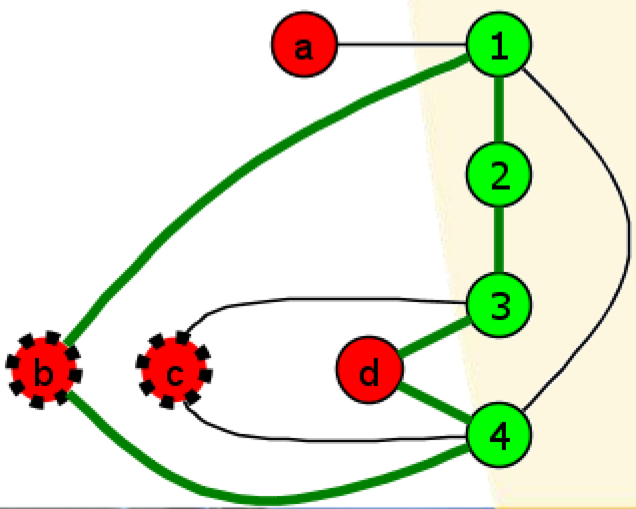
- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match $v(g_1,g_2)$ to r in H
  - matching in H saturating all connected pairs
- By matching property: exists matching in $H - R_U$ saturating all connected pairs

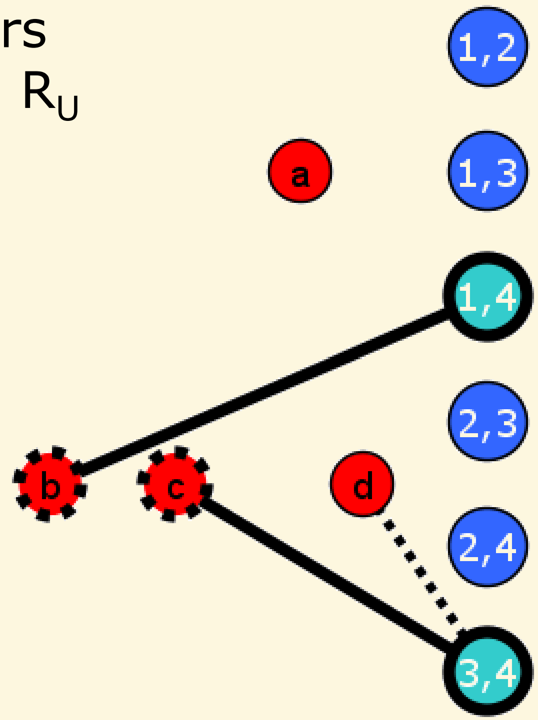




Correctness (III)

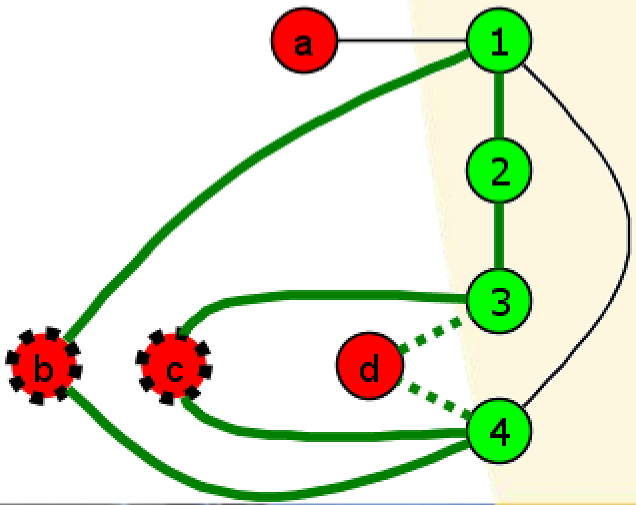
- No two connections (g_1, r, g_2) and (g_1, r′, g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match v(g_1,g_2) to r in H
  - matching in H saturating all connected pairs
- By matching property: exists matching in H – R_U saturating all connected pairs
- Update cycle accordingly

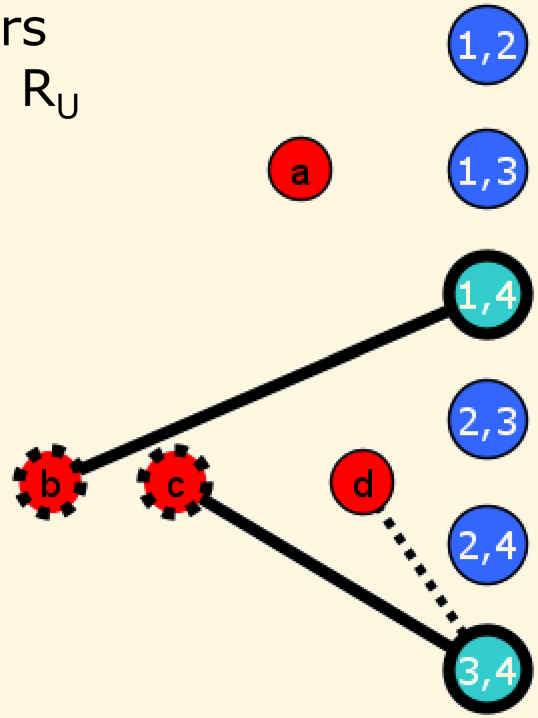




Correctness (III)

- No two connections (g_1, r, g_2) and (g_1, r', g_2) since $\ell > 4$
- For each connection (g_1, r, g_2):
  - match $v(g_1,g_2)$ to r in H
  - matching in H saturating all connected pairs
- By matching property: exists matching in $H - R_U$ saturating all connected pairs
- Update cycle accordingly





The kernel

- For the decision problem with a vertex cover in the input:

The kernel

- For the decision problem with a vertex cover in the input:

Long Cycle parameterized by a vertex cover X has a kernel with $|X| + |X|^2$ vertices.

The kernel

- For the decision problem with a vertex cover in the input:

Long Cycle parameterized by a vertex cover X has a kernel with $|X| + |X|^2$ vertices.

- Kernel does not depend on desired length of the cycle
  - Works for optimization problem as well

The kernel

- For the decision problem with a vertex cover in the input:

> Long Cycle parameterized by a vertex cover X has a kernel with $|X| + |X|^2$ vertices.

- Kernel does not depend on desired length of the cycle
  – Works for optimization problem as well

- If X is not given:
  – Compute a 2-approximate vertex cover, use it as X

The kernel

- For the decision problem with a vertex cover in the input:

> Long Cycle parameterized by a vertex cover X has a kernel with $|X| + |X|^2$ vertices.

- Kernel does not depend on desired length of the cycle
  - Works for optimization problem as well
- If X is not given:
  - Compute a 2-approximate vertex cover, use it as X
- Also applies to Long Path, Disjoint Paths, Disjoint Cycles

Polynomial kernel by Max Leaf Number

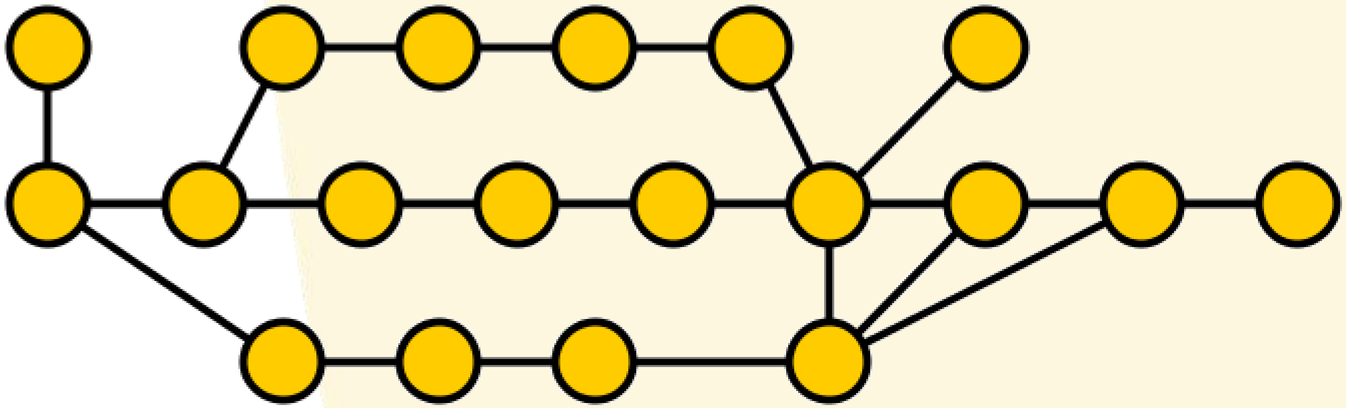
LONG CYCLE

Long Cycle parameterized by Max Leaf Number

- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?

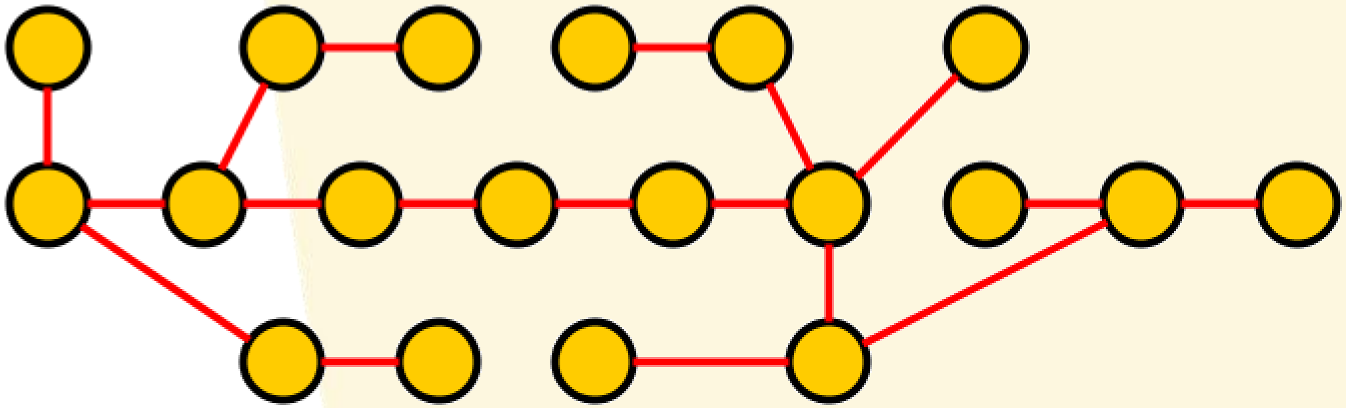
Long Cycle parameterized by Max Leaf Number

- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?



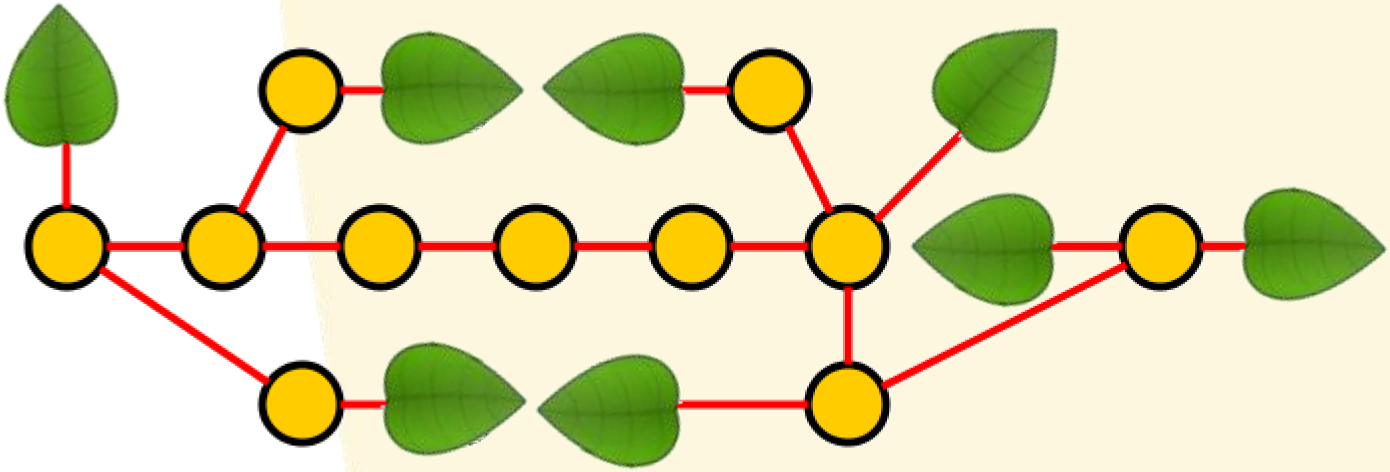
Long Cycle parameterized by Max Leaf Number

- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?



Long Cycle parameterized by Max Leaf Number

- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?



Long Cycle parameterized by Max Leaf Number

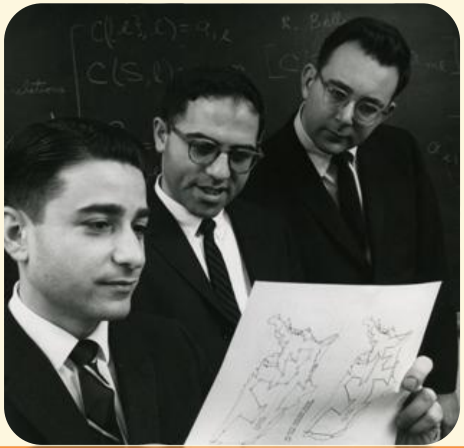
- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?

1. Kleitman-West Theorem

Long Cycle parameterized by Max Leaf Number

- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?

1. Kleitman-West Theorem



2. Held-Karp Dynamic Programming

Long Cycle parameterized by Max Leaf Number

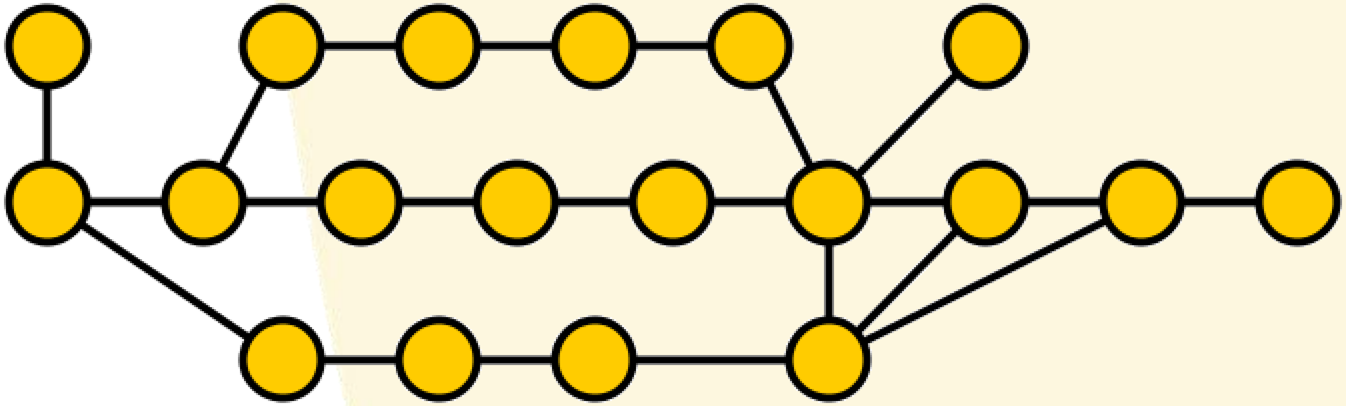
- Input: Graph G, integer ℓ, integer k.
- Parameter: k, promised to be the max leaf number of G.
- Question: Does G contain a simple cycle of length $\geq \ell$?

1. Kleitman-West Theorem

2. Held-Karp Dynamic Programming

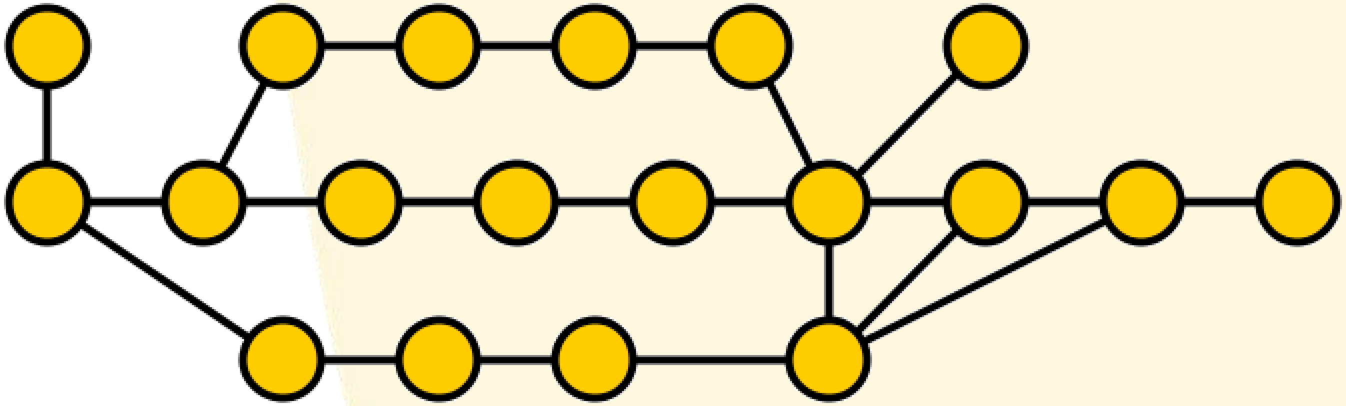
3. Karp Reduction

The kernelization algorithm



The kernelization algorithm

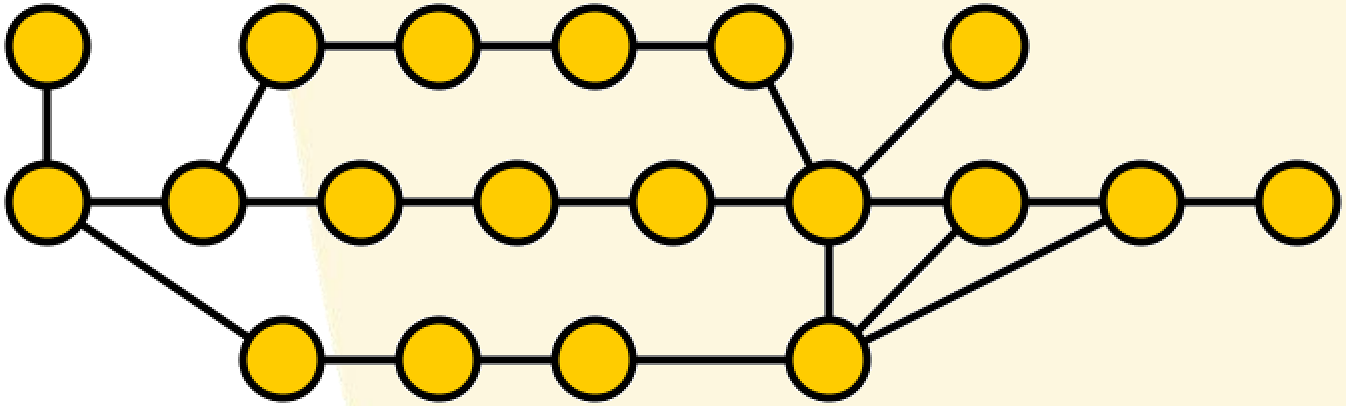
Kleitman-West Theorem



The kernelization algorithm

Kleitman-West Theorem

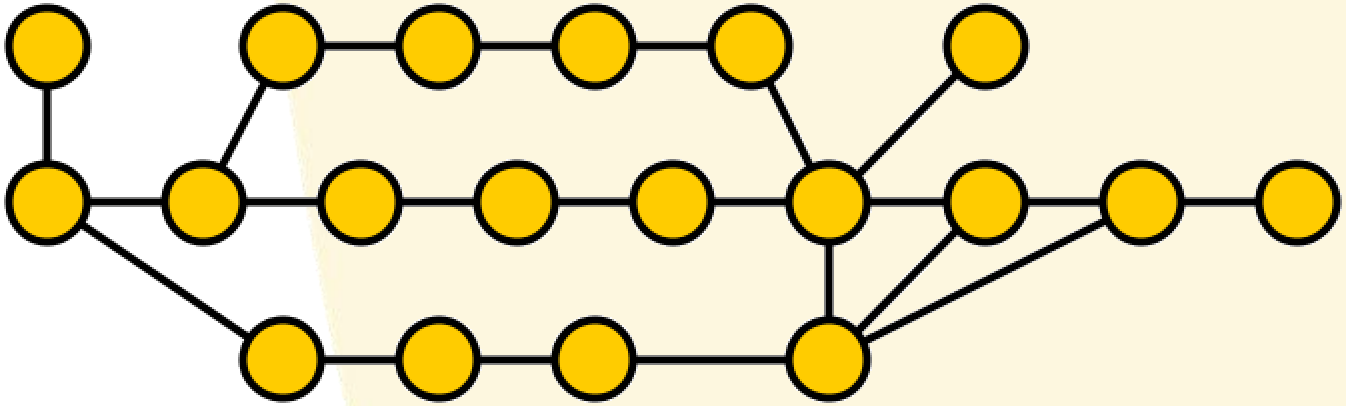
- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$



The kernelization algorithm

Kleitman-West Theorem

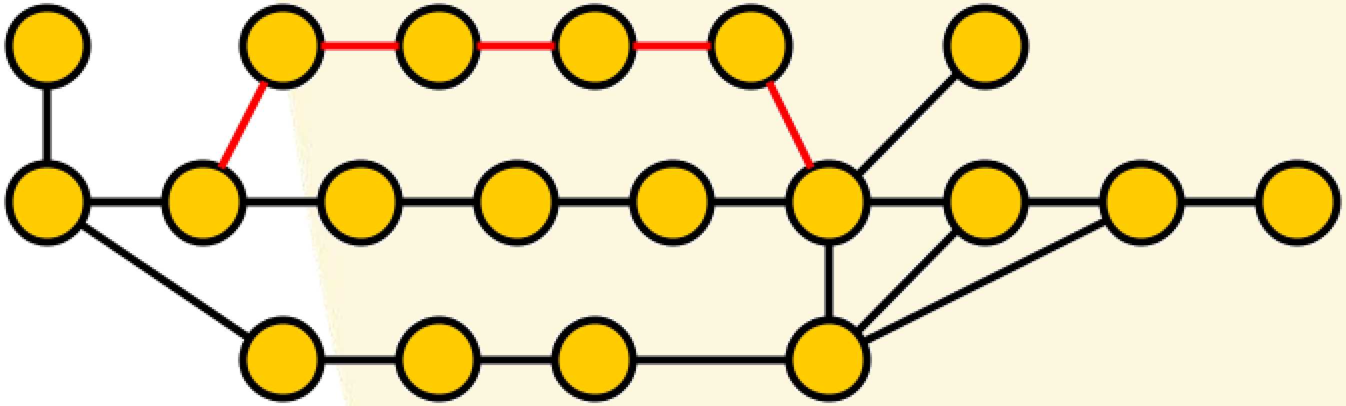
- Let X be vertices of degree ≠ 2: |X| ≤ c·k
- Transform paths of degree-2 vertices into weighted edges



The kernelization algorithm

Kleitman-West Theorem

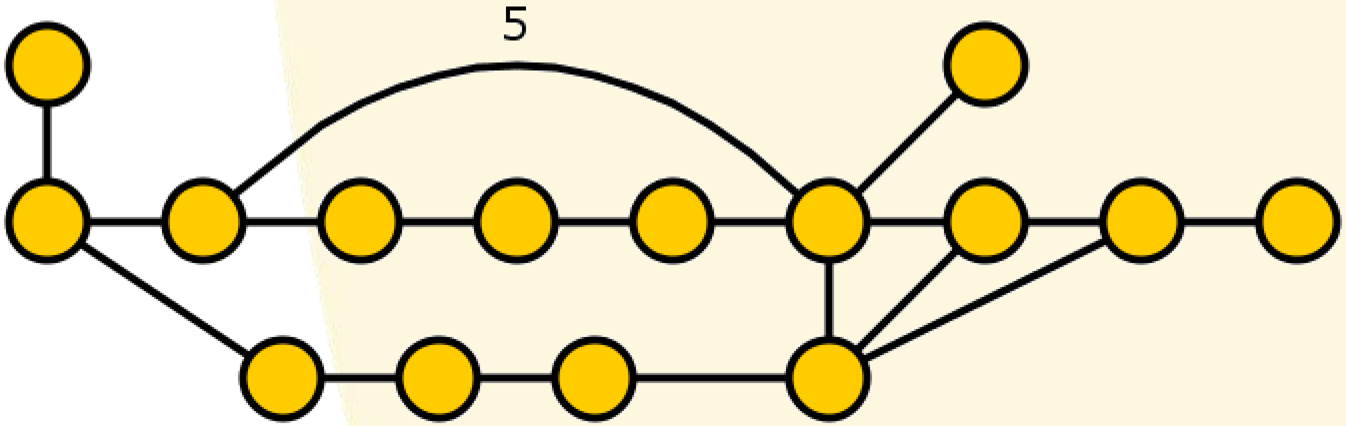
- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges



The kernelization algorithm

Kleitman-West Theorem

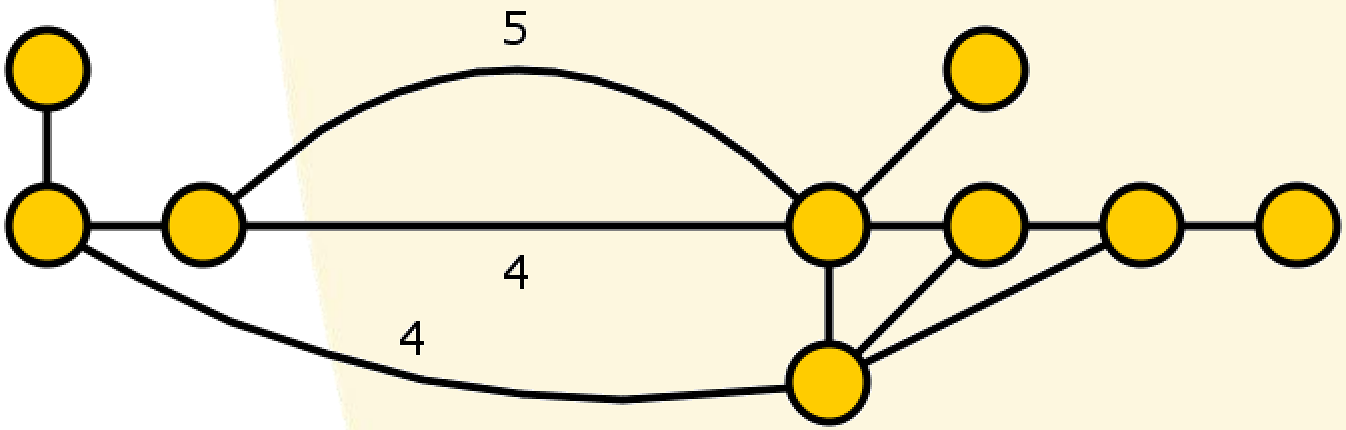
- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges



The kernelization algorithm

Kleitman-West Theorem

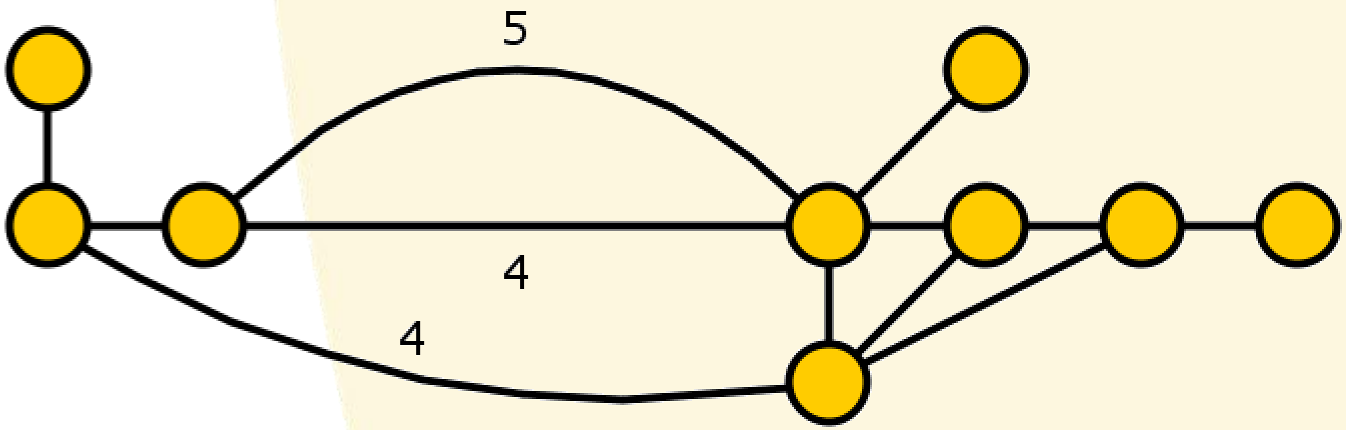
- Let X be vertices of degree ≠ 2: |X| ≤ c·k
- Transform paths of degree-2 vertices into weighted edges



The kernelization algorithm

Kleitman-West Theorem

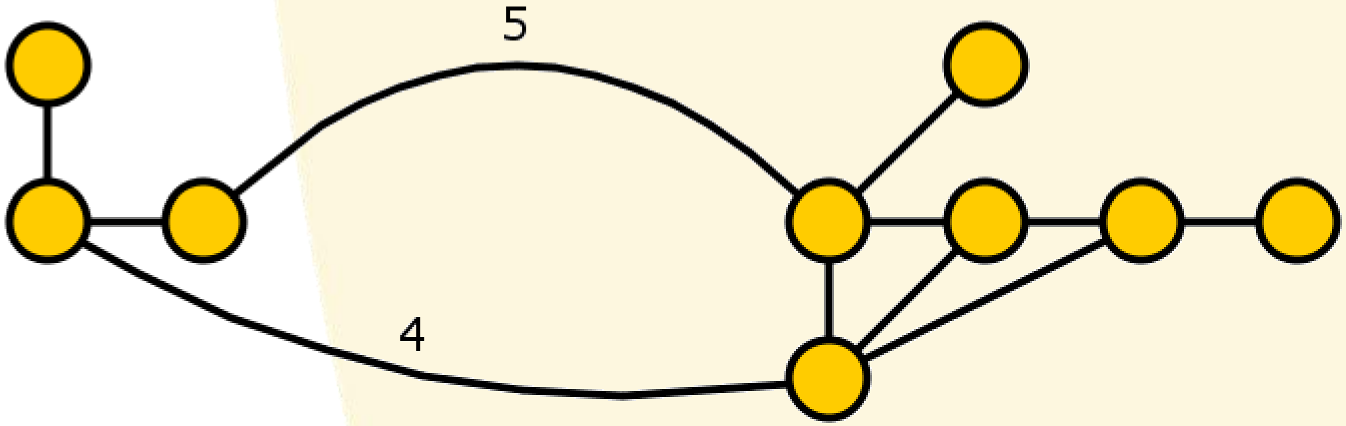
- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges
- Reduce to weighted simple graph (G', w') with $|V(G')| = |X| \leq c \cdot k$



The kernelization algorithm

Kleitman-West Theorem

- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges
- Reduce to weighted simple graph (G’, w’) with $|V(G')| = |X| \leq c \cdot k$



The kernelization algorithm

Kleitman-West Theorem

- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges
- Reduce to weighted simple graph (G', w') with $|V(G')| = |X| \leq c \cdot k$

The kernelization algorithm

Kleitman-West Theorem

- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges
- Reduce to weighted simple graph (G', w') with $|V(G')| = |X| \leq c \cdot k$

Held-Karp Dynamic Programming

- If binary encoding of a weight uses > $c \cdot k$ bits:
  - There were > $2^{c \cdot k}$ degree-2 vertices so $n > 2^{c \cdot k}$
  - Solve weighted instance: $O(2^{|X|} |X|^3)$ is $O(n^4)$ time

The kernelization algorithm

Kleitman-West Theorem

- Let X be vertices of degree ≠ 2: $|X| \leq c \cdot k$
- Transform paths of degree-2 vertices into weighted edges
- Reduce to weighted simple graph (G', w') with $|V(G')| = |X| \leq c \cdot k$

Held-Karp Dynamic Programming

- If binary encoding of a weight uses > $c \cdot k$ bits:
  - There were > $2^{c \cdot k}$ degree-2 vertices so $n > 2^{c \cdot k}$
  - Solve weighted instance: $O(2^{|X|}\ |X|^3)$ is $O(n^4)$ time

Karp Reduction

- If binary encoding is small: (G', w', ℓ') has bitsize poly(k)
  - Weighted Long Cycle is in NP
  - Reduce to back to unweighted problem
  - Polynomial-time transformation, output has size poly(k)

DISCUSSION & CONCLUSION

Structural parameterizations of Hamiltonian Cycle (& related)

Structural parameterizations of Hamiltonian Cycle (& related)

Vertex Cover Number

- Deletion distance to treewidth 0

Structural parameterizations of Hamiltonian Cycle (& related)

Vertex Cover Number

- Deletion distance to treewidth 0

Structural parameterizations of Hamiltonian Cycle (& related)

Vertex Cover Number

- Deletion distance to treewidth 0

Deletion distance to Outerplanar

- Deletion distance to treewidth 2

Structural parameterizations of Hamiltonian Cycle (& related)

Vertex Cover Number

- Deletion distance to treewidth 0

Deletion distance to Outerplanar

- Deletion distance to treewidth 2

Structural parameterizations of Hamiltonian Cycle (& related)

Vertex Cover Number

- Deletion distance to treewidth 0

Feedback Vertex Number

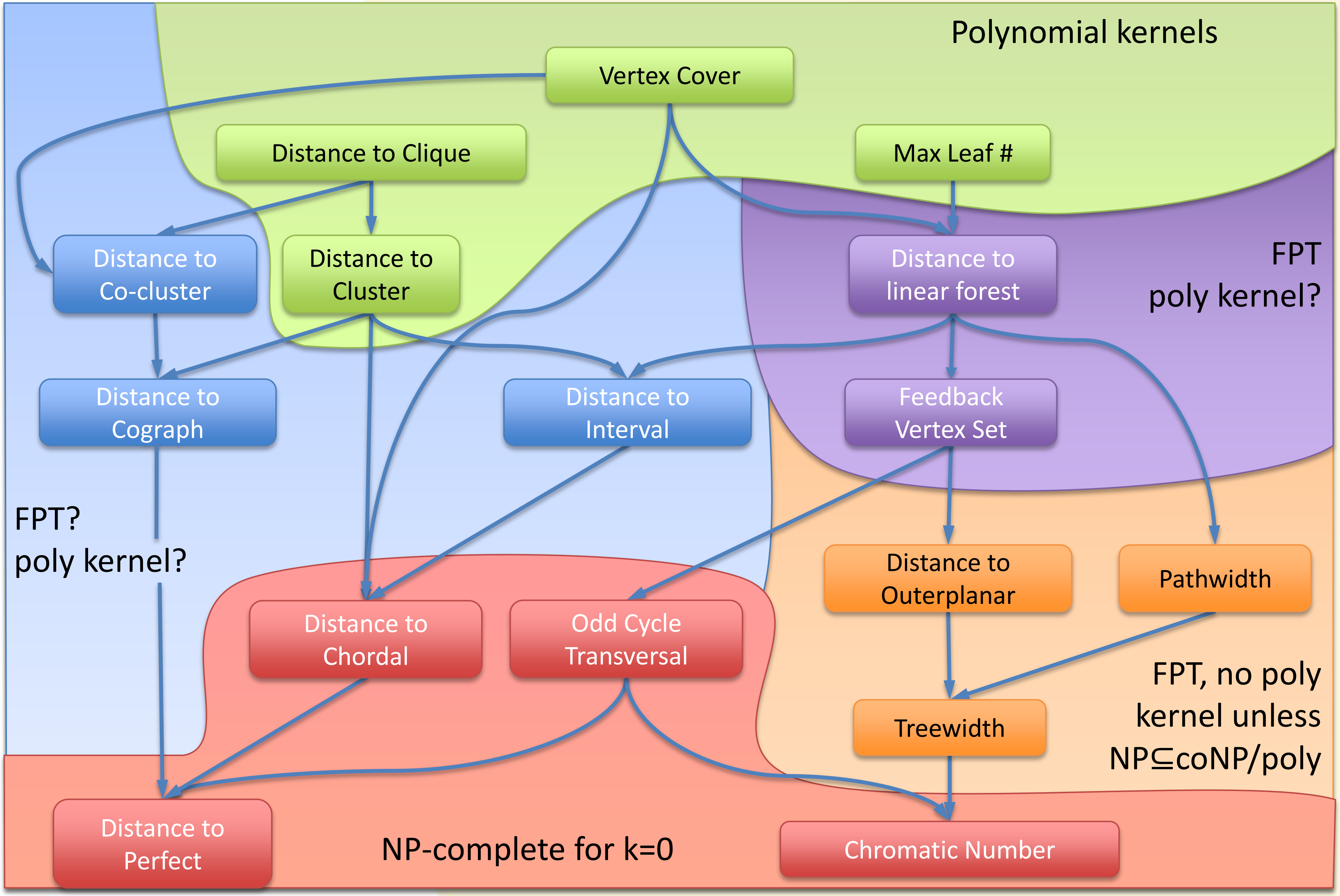
- Deletion distance to treewidth 1

Deletion distance to Outerplanar

- Deletion distance to treewidth 2

Structural parameterizations of Hamiltonian Cycle (& related)

Vertex Cover Number	Feedback Vertex Number	Deletion distance to Outerplanar
• Deletion distance to treewidth 0	• Deletion distance to treewidth 1	• Deletion distance to treewidth 2



Complexity overview for Long Cycle parameterized by…

Conclusion

- Structural parameterizations of Path and Cycle problems admit polynomial kernels
- Various upper and lower-bound results

Conclusion

- Structural parameterizations of Path and Cycle problems admit polynomial kernels
- Various upper and lower-bound results

Poly kernels for Long Path parameterized by:

- feedback vertex number
- vertex-deletion distance to a cograph

Poly kernels for Long Path parameterized by:

- Max Leaf Number, *without* using binary encoding?

Is Longest Path in FPT …

- parameterized by a (given) deletion set to an Interval graph?

Conclusion

- Structural parameterizations of Path and Cycle problems admit polynomial kernels
- Various upper and lower-bound results

Poly kernels for Long Path parameterized by:

encou...

Is Longest Path in FPT …

- parameterized by a (given) deletion set to an Interval graph?

THANK YOU!