Kernelization

Every block of stone has a statue inside it and it is the task of the sculptor to discover it.

— Michelangelo Buonarroti

Lecture #5

Roohani Sharma

November 16, 2021

Kernelization

Every block of stone has a statue inside it and it is the task of the sculptor to discover it.

— Michelangelo Buonarroti

Lecture #5

Roohani Sharma

November 16, 2021

The lost island of pre-processing…

Pre-processing for NP-hard problems?

*With guarantees

Can we preprocess a given instance of an NP-hard problem and guarantee a 1% decrease in its size
in polynomial time?

Pre-processing for NP-hard problems?

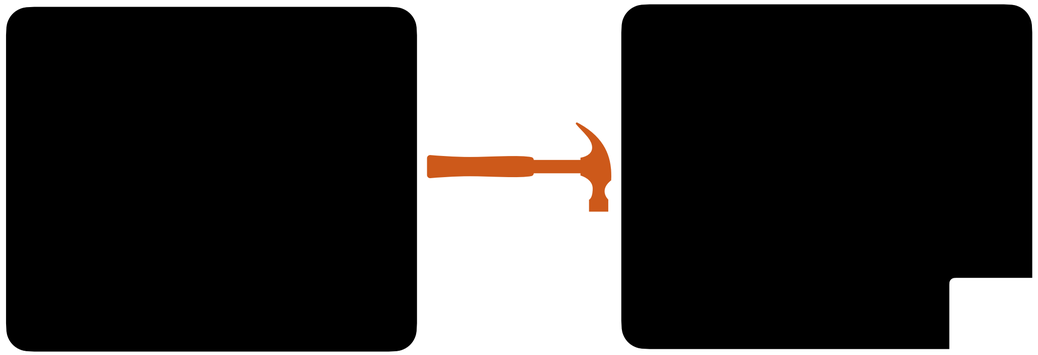
*With guarantees

Can we preprocess a given instance of an NP-hard problem and guarantee a 1% decrease in its size
in polynomial time?

Pre-processing for NP-hard problems?

*With guarantees

Can we preprocess a given instance of an NP-hard problem and guarantee a 1% decrease in its size
in polynomial time?



Pre-processing for NP-hard problems?

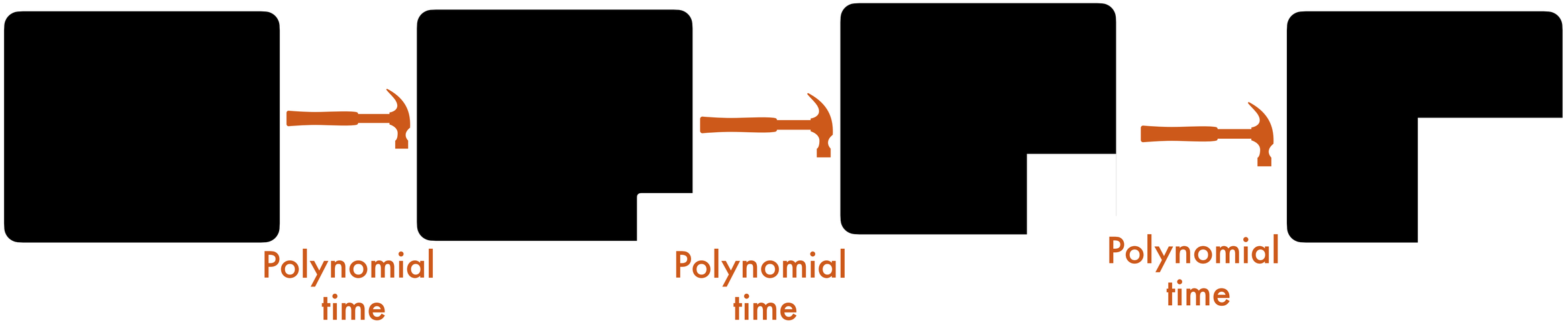
*With guarantees

Can we preprocess a given instance of an NP-hard problem and guarantee a 1% decrease in its size
in polynomial time?

Pre-processing for NP-hard problems?

*With guarantees

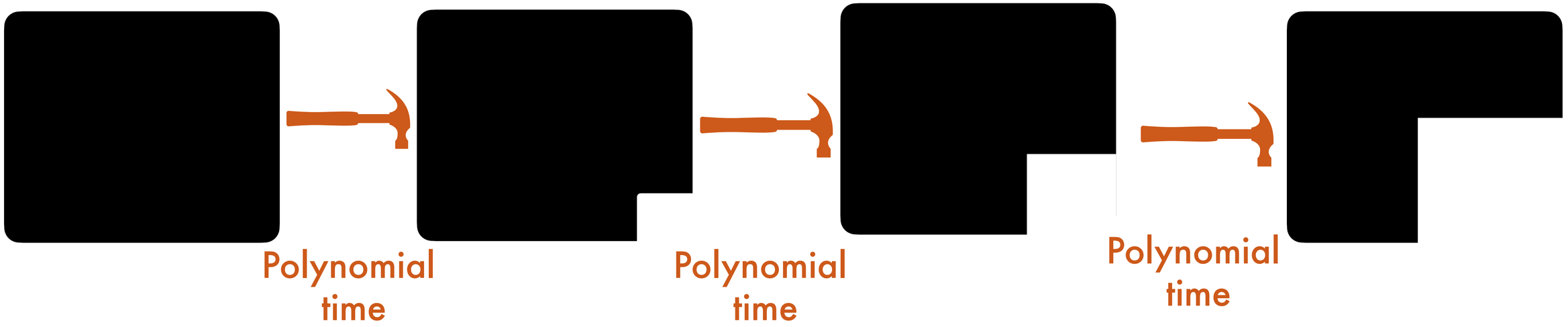
Can we preprocess a given instance of an NP-hard problem and guarantee a 1% decrease in its size
in polynomial time?



Pre-processing for NP-hard problems?

*With guarantees

Can we preprocess a given instance of an NP-hard problem and guarantee a 1% decrease in its size in polynomial time?



Kernelization

- Efficient pre-processing with guarantees for parameterized problems

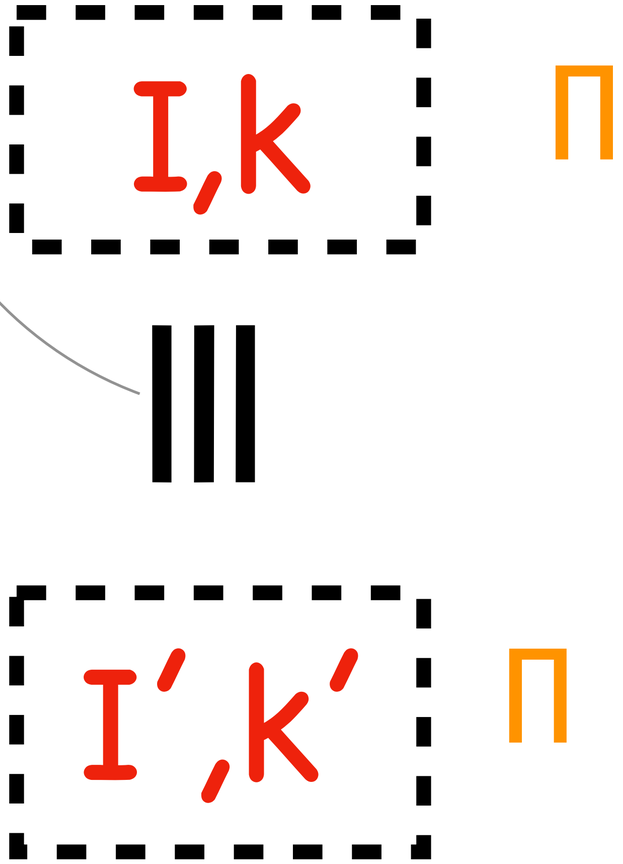
Kernelization

- Efficient pre-processing with guarantees for parameterized problems

Kernelization

- Efficient pre-processing with guarantees for parameterized problems

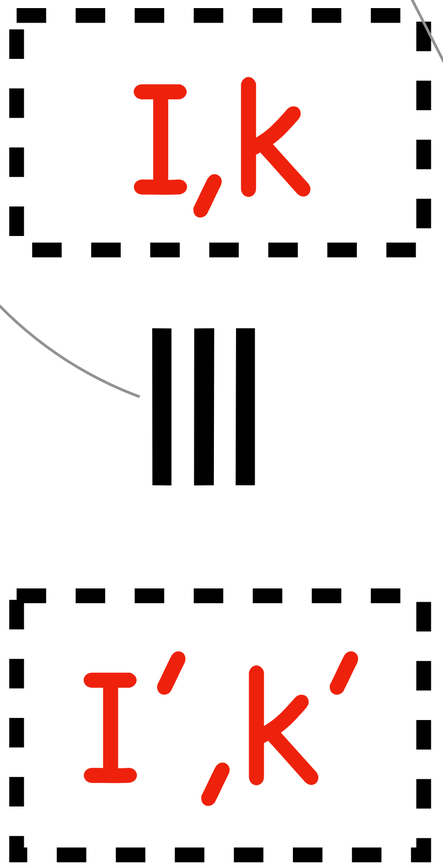
Runs in polynomial time



Kernelization

- Efficient pre-processing with guarantees for parameterized problems

Runs in polynomial time

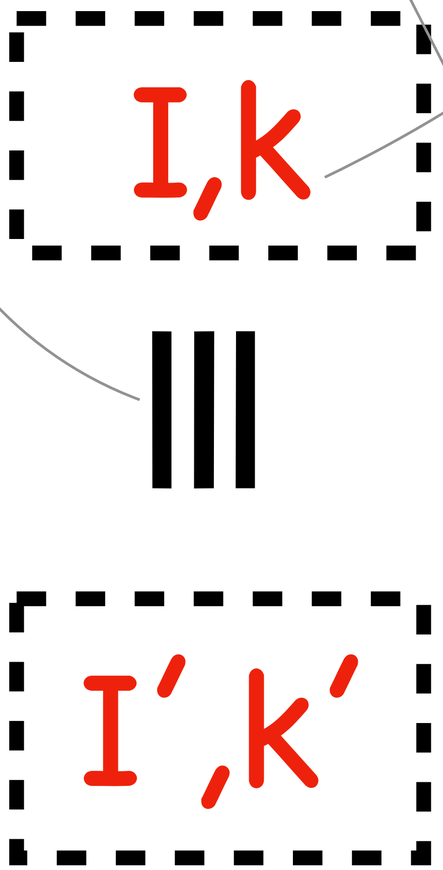


$|I'|, k' \leq g(k)$

Kernelization

- Efficient pre-processing with guarantees for parameterized problems

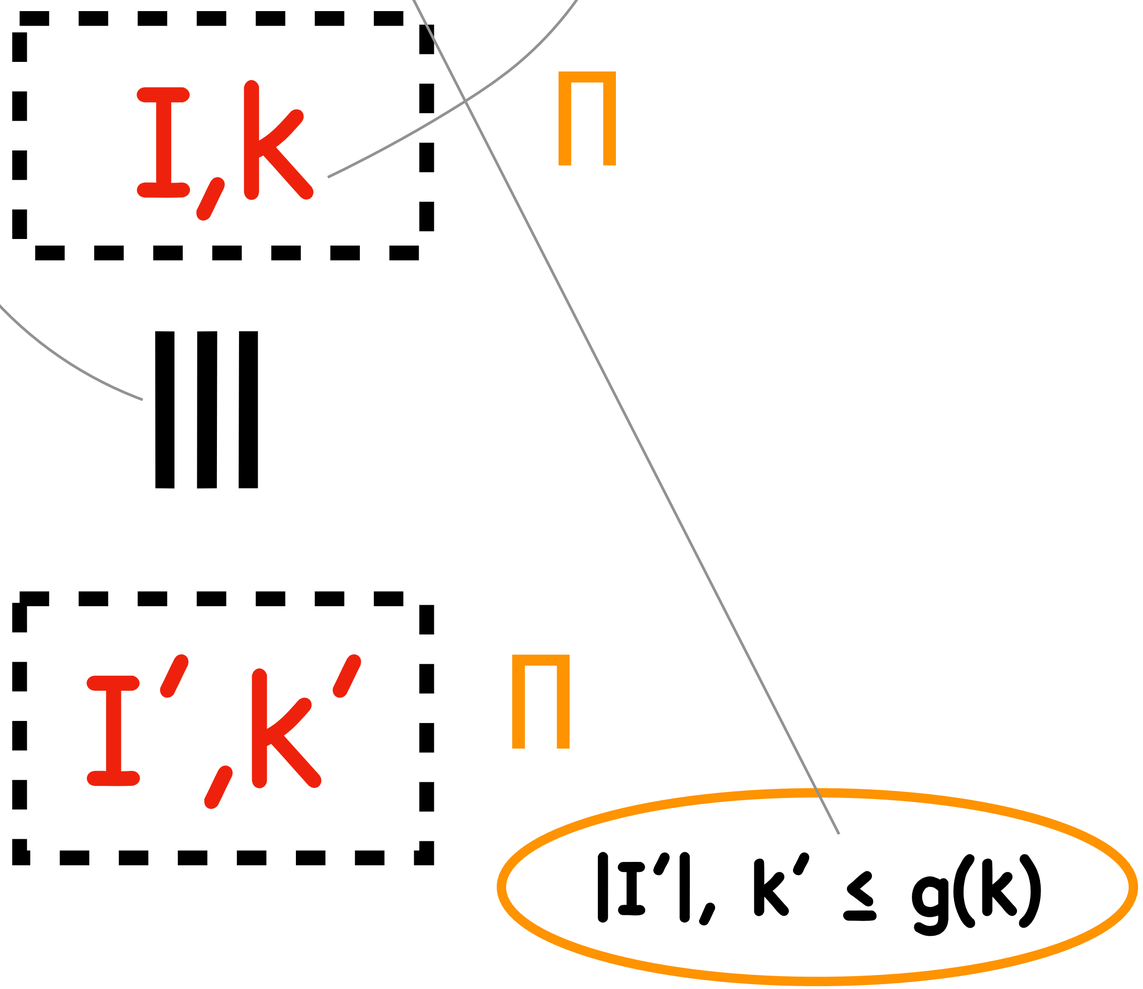
Runs in polynomial time



$|I'|, k' \leq g(k)$

Kernelization

- Efficient pre-processing with guarantees for parameterized problems

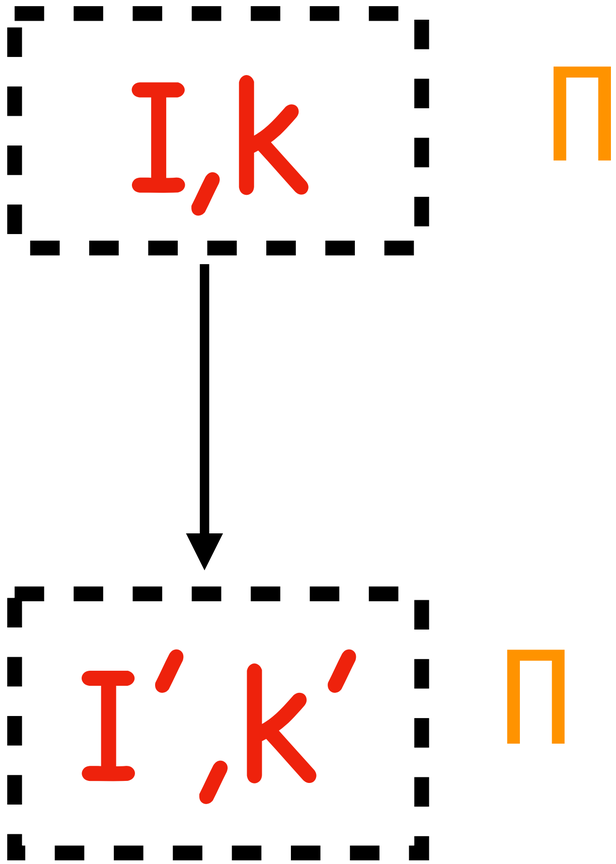


Π admits a kernel of size g(k).

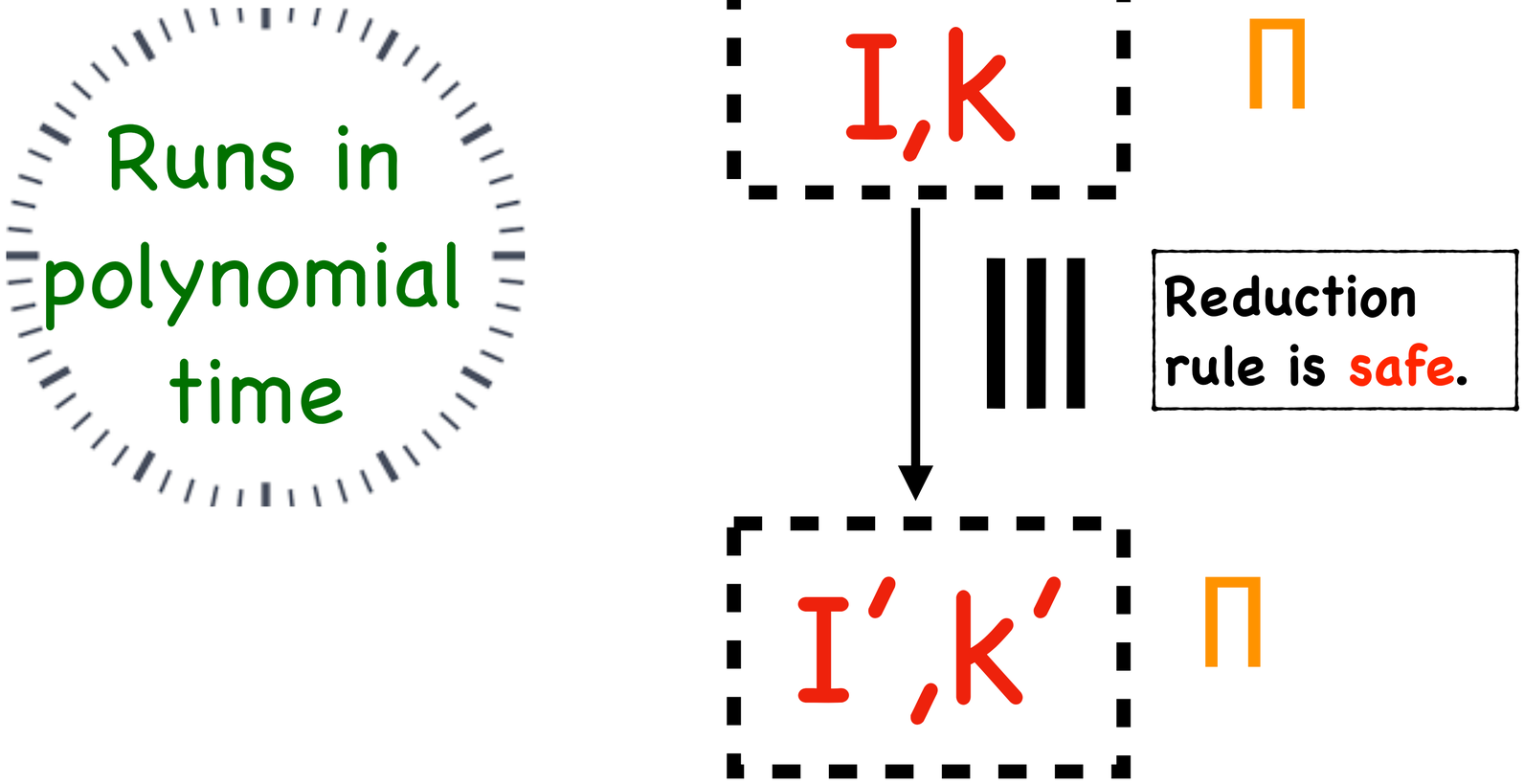
If g(k) is a polynomial/exponential function, then Π admits a polynomial/exponential kernel.

Reduction rules (RR)

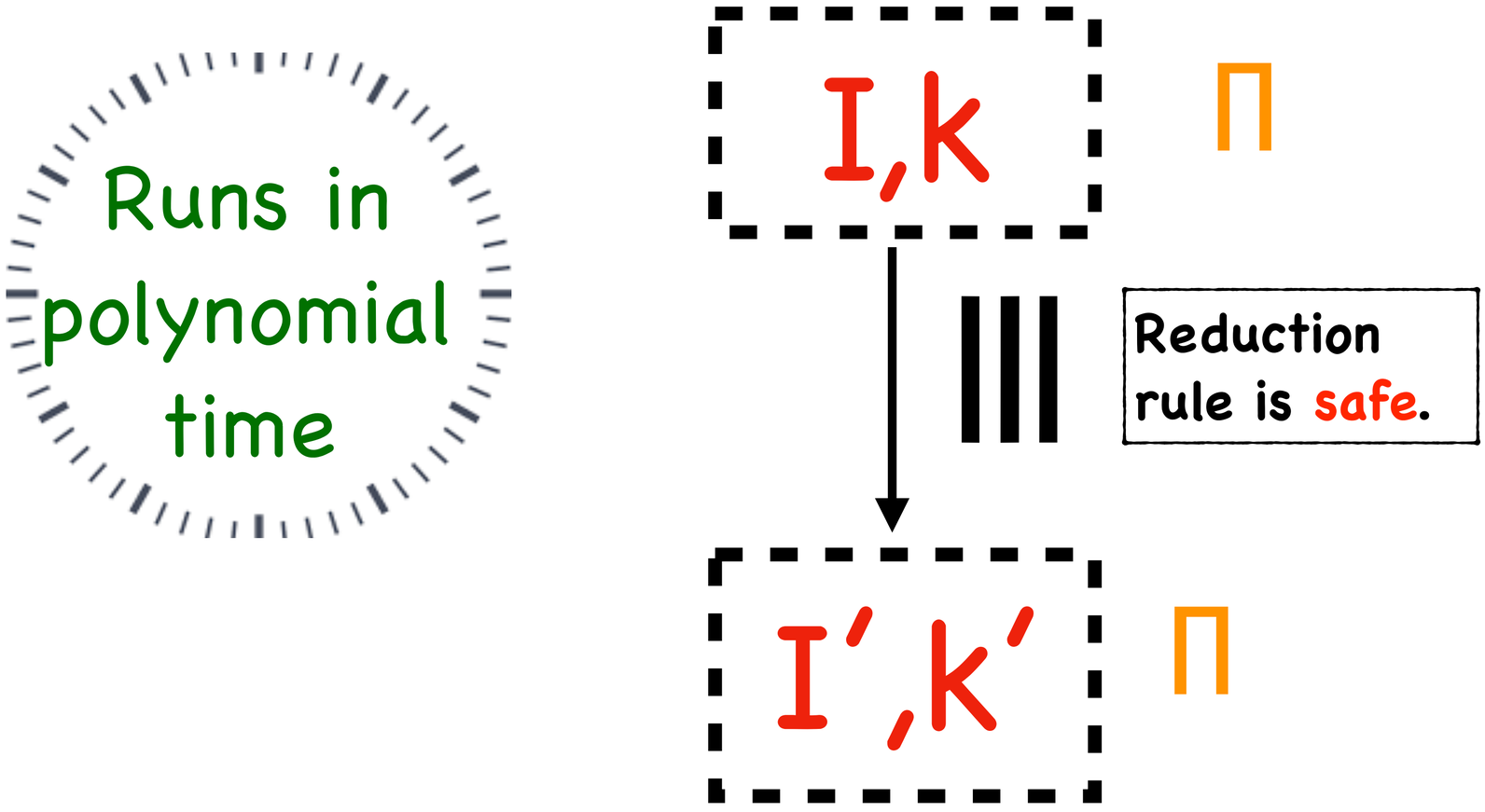
Runs in polynomial time



Reduction rules (RR)

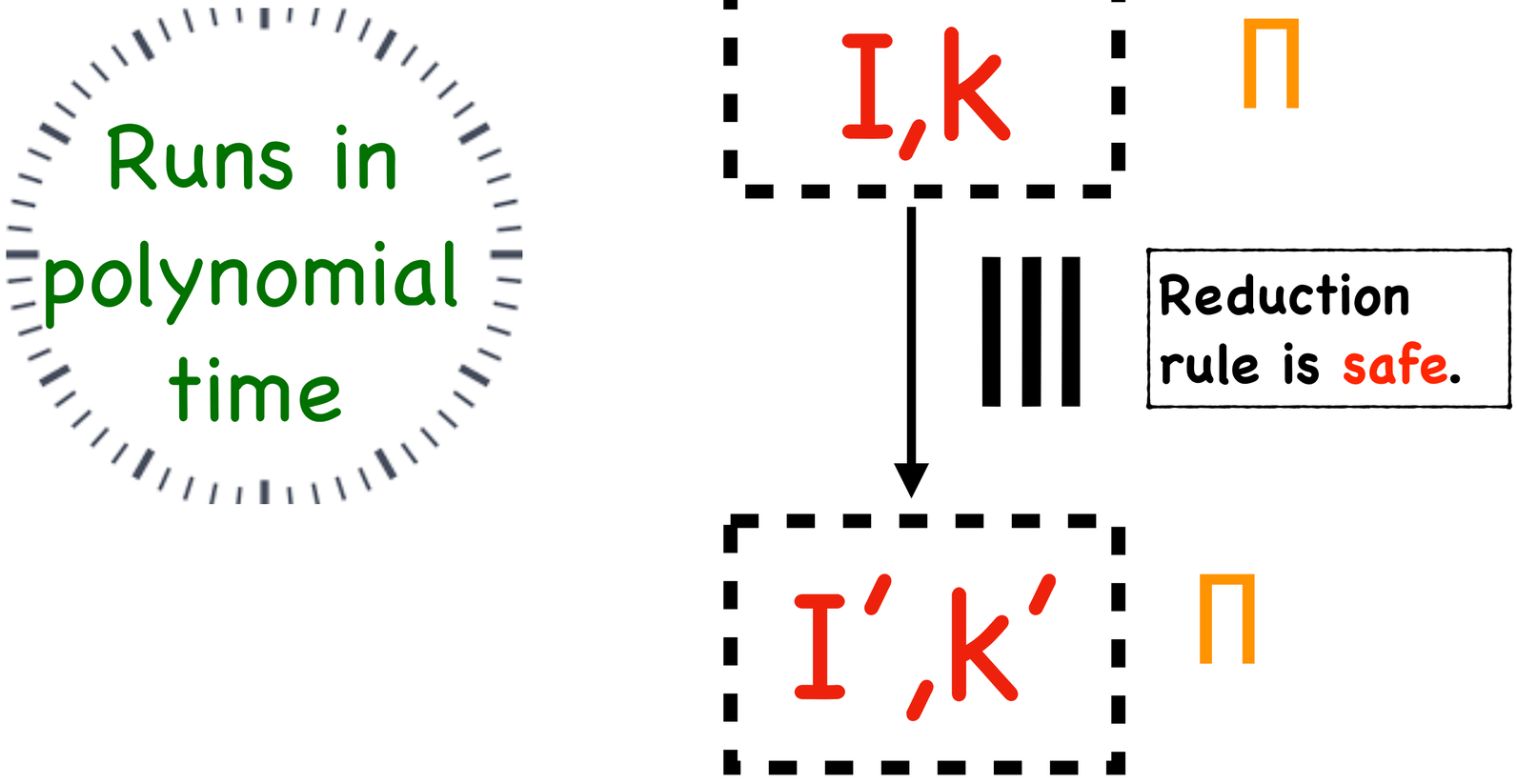


Reduction rules (RR)



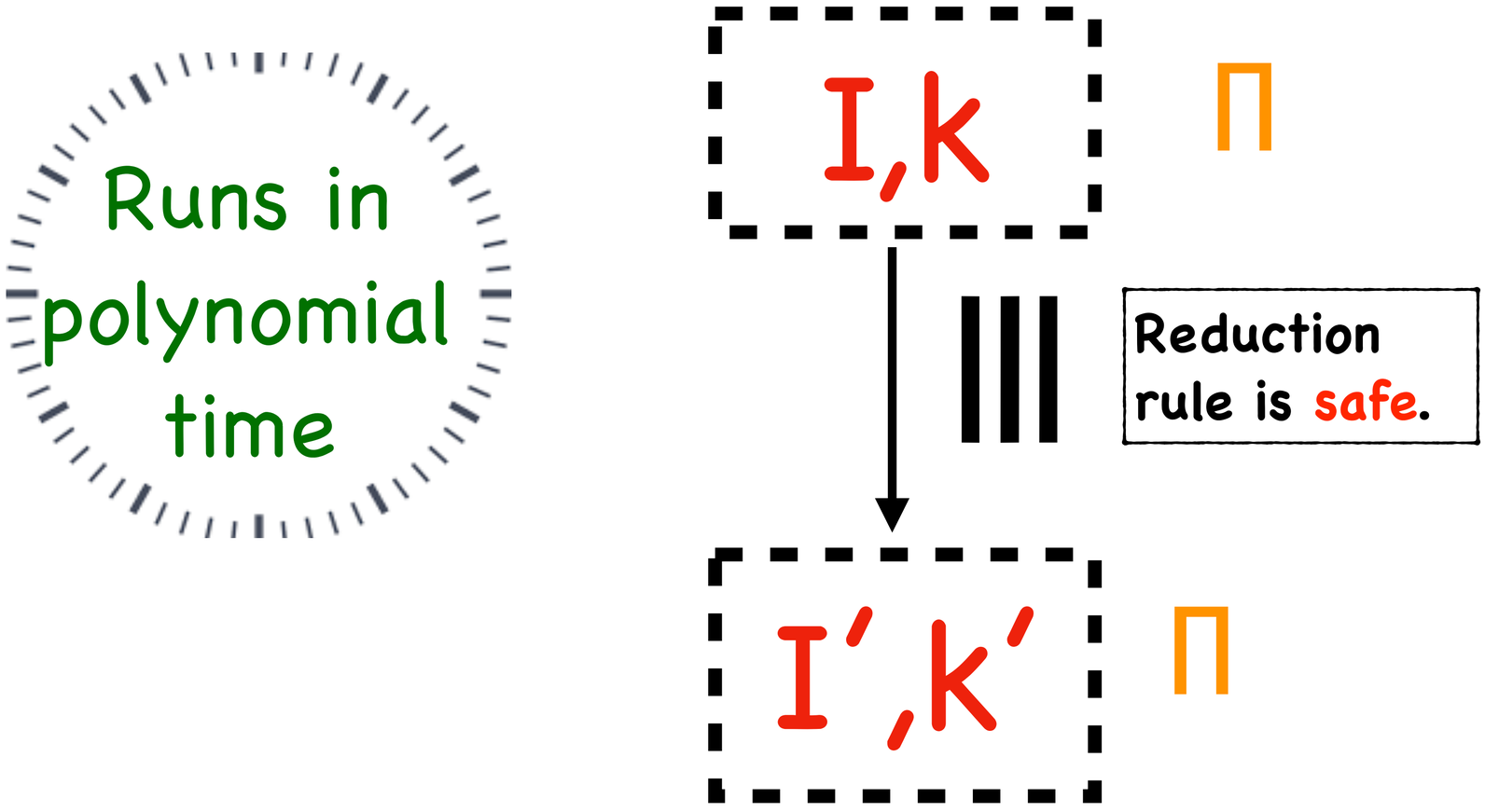
- A kernelization algorithm comprises of possibly several (polynomially many) safe reduction rules.

Reduction rules (RR)



- A kernelization algorithm comprises of possibly several (polynomially many) safe reduction rules.
- Each reduction rules should be applicable only a polynomial number of times.

Reduction rules (RR)



- A kernelization algorithm comprises of possibly several (polynomially many) safe reduction rules.
- Each reduction rules should be applicable only a polynomial number of times.
- This is followed by an analysis showing that if none of the designed reduction rules are applicable, then the size of the instance is bounded by some g(k).

Vertex Cover: Refresher

Vertex Cover

Input: A graph G, an integer k
Question: Does there at most k vertices, say S, such that each edge of G is incident on S?

Vertex Cover

Input: A graph G, an integer k
Question: Does there at most k vertices, say S, such that each edge of G is incident on S?

RR 1: Delete isolated vertices.

Vertex Cover

Input: A graph G, an integer k
Question: Does there at most k vertices, say S, such that each edge of G is incident on S?

RR 1: Delete isolated vertices.

RR 2 (Buss Rule): If there exists a vertex of degree at least k+1, then delete this vertex and decrease the budget k by 1.

Vertex Cover

Input: A graph G, an integer k
Question: Does there at most k vertices, say S, such that each edge of G is incident on S?

RR 1: Delete isolated vertices.

RR 2 (Buss Rule): If there exists a vertex of degree at least k+1, then delete this vertex and decrease the budget k by 1.

Let the reduced instance (with respect to RR 1 and RR 2) have n′ vertices, m′ edges, and k′ budget.

Vertex Cover

Input: A graph G, an integer k
Question: Does there at most k vertices, say S, such that each edge of G is incident on S?

RR 1: Delete isolated vertices.

RR 2 (Buss Rule): If there exists a vertex of degree at least k+1, then delete this vertex and decrease the budget k by 1.

Let the reduced instance (with respect to RR 1 and RR 2) have n' vertices, m' edges, and k' budget.

RR 3: If $n' > k'^2 + k'$ or $m' > k'^2$, then say No.

Vertex Cover

RR 3: If $n' > k'^2 + k'$ or $m' > k'^2$, then say No.

Safeness of RR 3: Suppose for the sake of contradiction that the reduced instance is a Yes-instance.
Let S be a vertex cover of the reduced instance of size at most k'.

Vertex Cover

RR 3: If $n' > k'^2 + k'$ or $m' > k'^2$, then say No.

Safeness of RR 3: Suppose for the sake of contradiction that the reduced instance is a Yes-instance.
Let S be a vertex cover of the reduced instance of size at most k′.

$$m' \leq \sum_{v \in S} |N(v)| \leq k'k' = k'^2$$ (because Buss rule is not applicable)

Vertex Cover

RR 3: If $n' > k'^2 + k'$ or $m' > k'^2$, then say No.

Safeness of RR 3: Suppose for the sake of contradiction that the reduced instance is a Yes-instance.
Let S be a vertex cover of the reduced instance of size at most k′.

$$m' \leq \sum_{v \in S} |N(v)| \leq k'k' = k'^2$$ (because Buss rule is not applicable)

$$n' \leq |S| \cup_{v \in S} |N(v)| \leq k' + k'^2$$ (because there are no isolated vertices)

Vertex Cover

RR 3: If $n' > k'^2 + k'$ or $m' > k'^2$, then say No.

Safeness of RR 3: Suppose for the sake of contradiction that the reduced instance is a Yes-instance.
Let S be a vertex cover of the reduced instance of size at most k′.

$$m' \leq \sum_{v \in S} |N(v)| \leq k'k' = k'^2$$ (because Buss rule is not applicable)

$$n' \leq |S| \cup_{v \in S} |N(v)| \leq k' + k'^2$$ (because there are no isolated vertices)

VC admits a kernel with $k^2 + k$ vertices and k^2 edges.

Feedback Arc Set in Tournaments

Feedback Arc Set in Tournaments

Input: A tournament G, an integer k
Question: Does there at most k edges, say F, such that G-F is acyclic?

Feedback Arc Set in Tournaments

Input: A tournament G, an integer k
Question: Does there at most k edges, say F, such that G-F is acyclic?

Observation: A tournament has a directed cycle if and only is it has a directed triangle.

Feedback Arc Set in Tournaments

Input: A tournament G, an integer k
Question: Does there at most k edges, say F, such that G-F is acyclic?

Observation: A tournament has a directed cycle if and only is it has a directed triangle.

RR 0: If an edge is contained in at least k+1 triangles, delete it??

Feedback Arc Set in Tournaments

Feedback Arc Set in Tournaments

Lemma: Let G be a directed graph. Then F is an inclusion minimal feedback arc set of G if and only if F is an inclusion minimal set of arcs such that G◉F is acyclic (G◉F is the graph G where the arcs of F have been reversed).

Feedback Arc Set in Tournaments

RR 1: If an edge is contained in at least k+1 triangles, then reverse it and decrease k by1.

Feedback Arc Set in Tournaments

RR 1: If an edge is contained in at least k+1 triangles, then reverse it and decrease k by1.

RR 2: If a vertex does not participate in a triangle, delete it.

Feedback Arc Set in Tournaments

Let the reduced instance (with respect to RR 1 and RR 2) have n' vertices and k' budget.

RR 3: If $n' > k' (k' + 2)$ then say No.

Feedback Arc Set in Tournaments

Let the reduced instance (with respect to RR 1 and RR 2) have n' vertices and k' budget.

RR 3: If $n' > k'(k'+2)$ then say No.

FAST admits a kernel with $k(k+2)$ vertices.

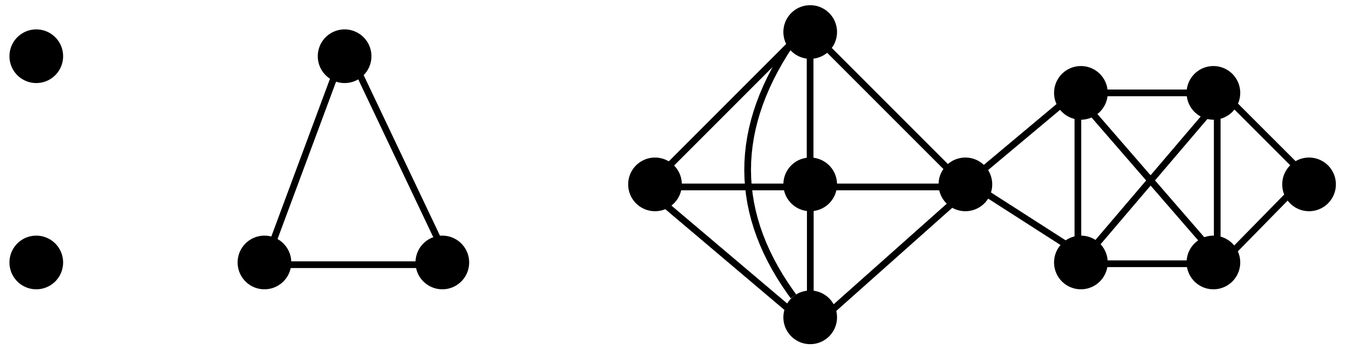
Edge Clique Cover

Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, $C_1, \ldots, C_k$, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?

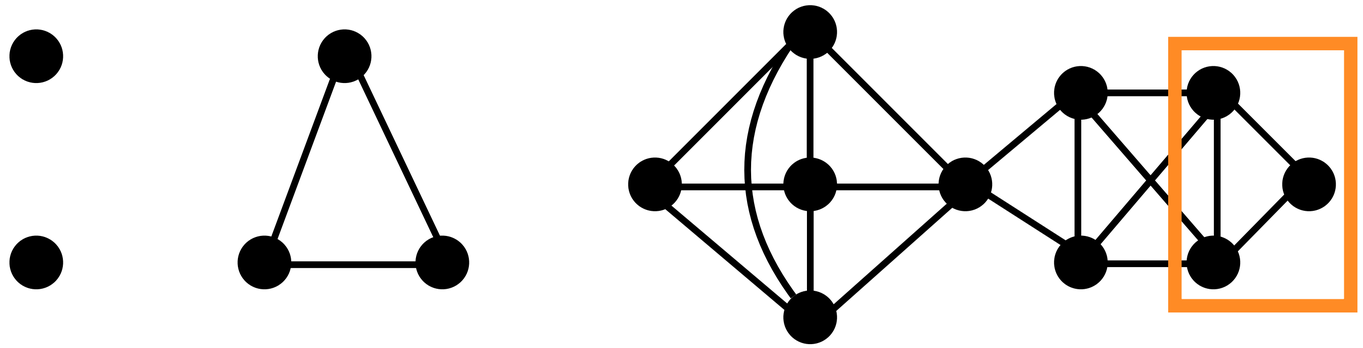
Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?



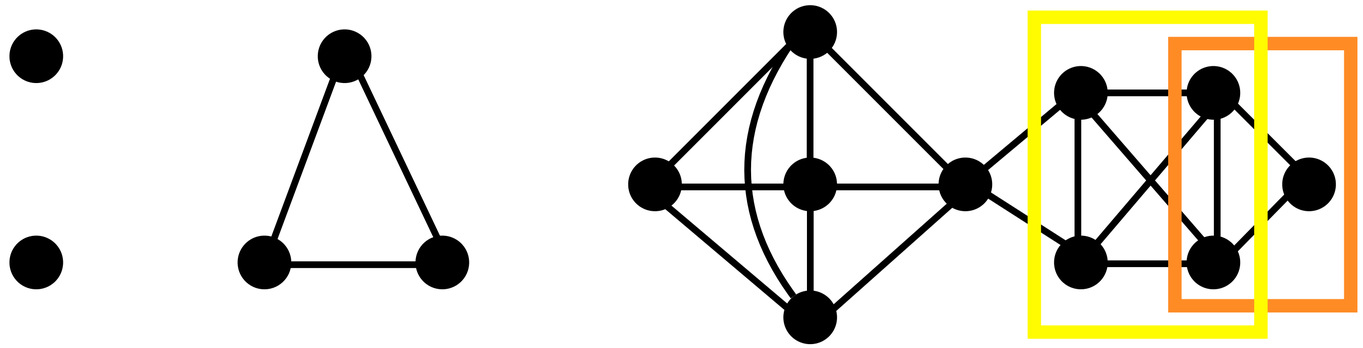
Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?



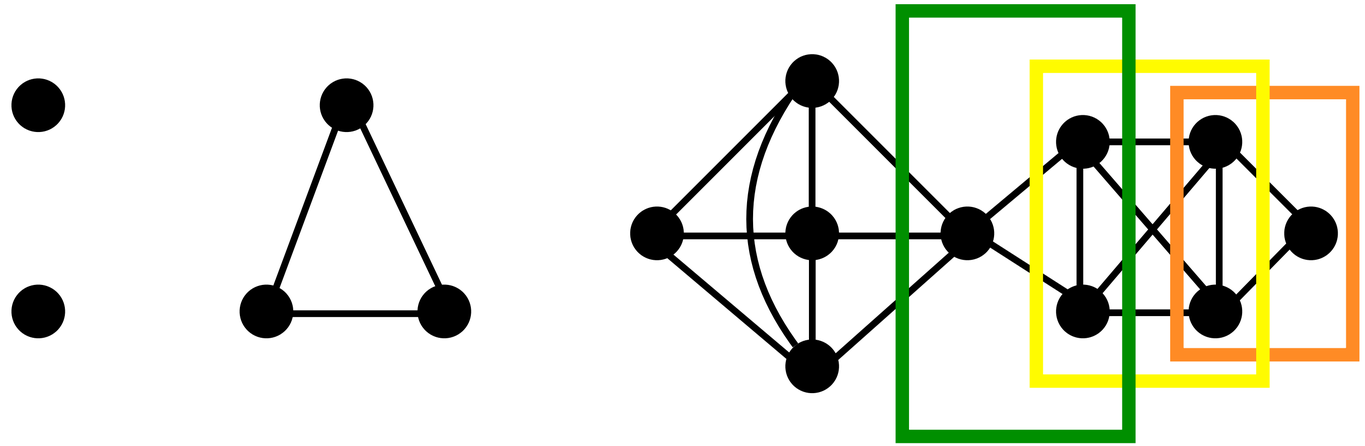
Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?



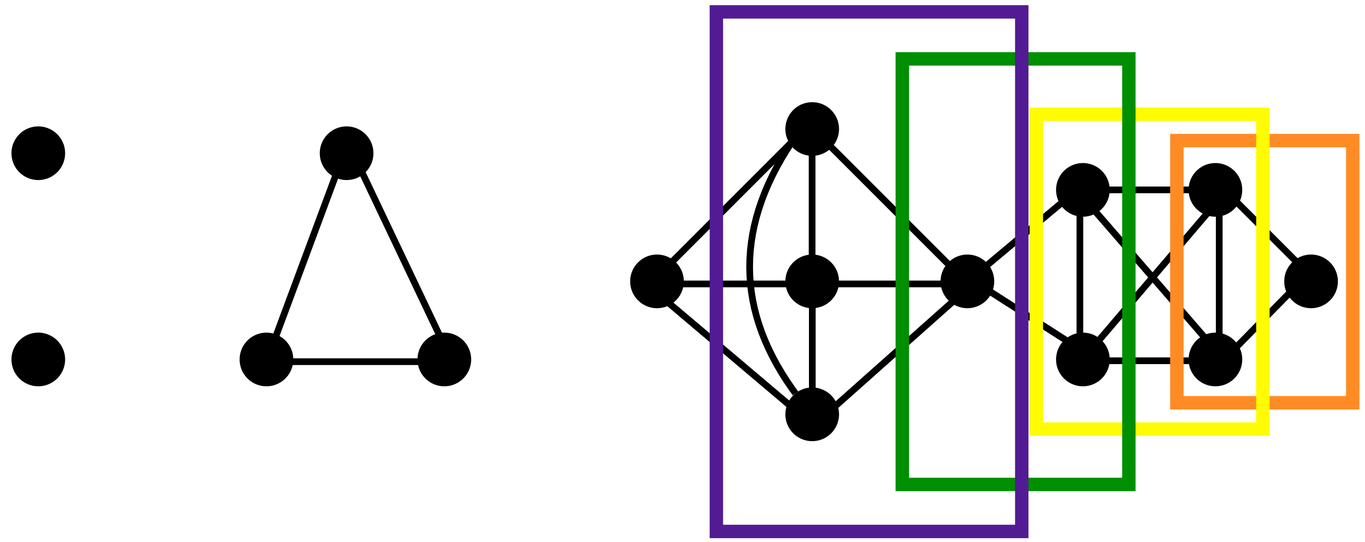
Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?



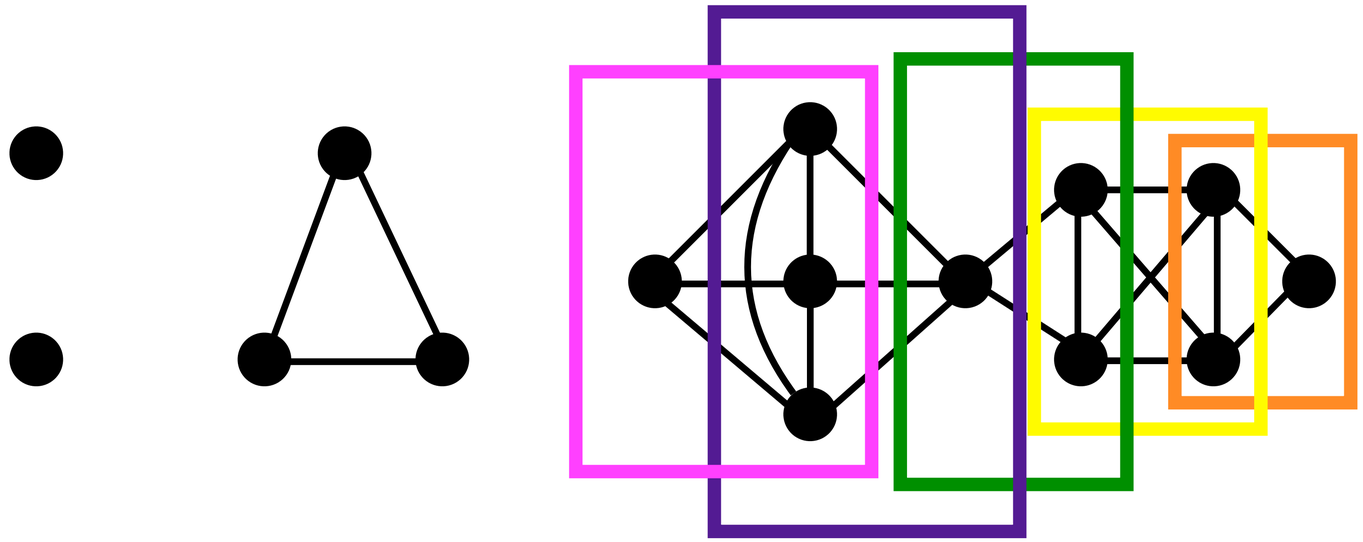
Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?



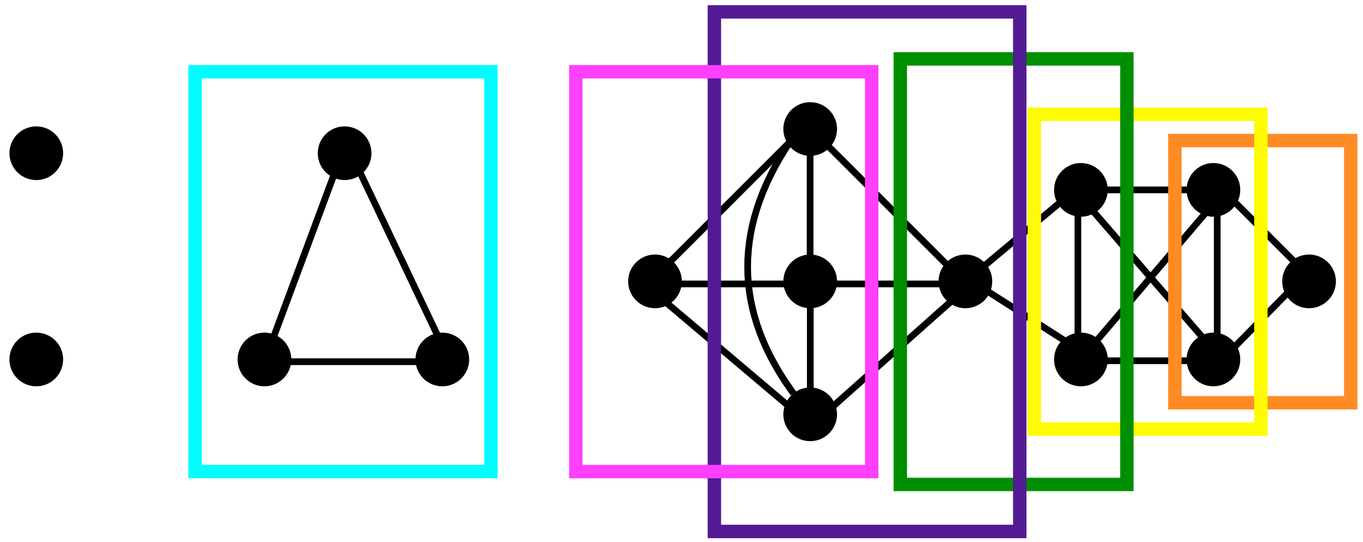
Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?

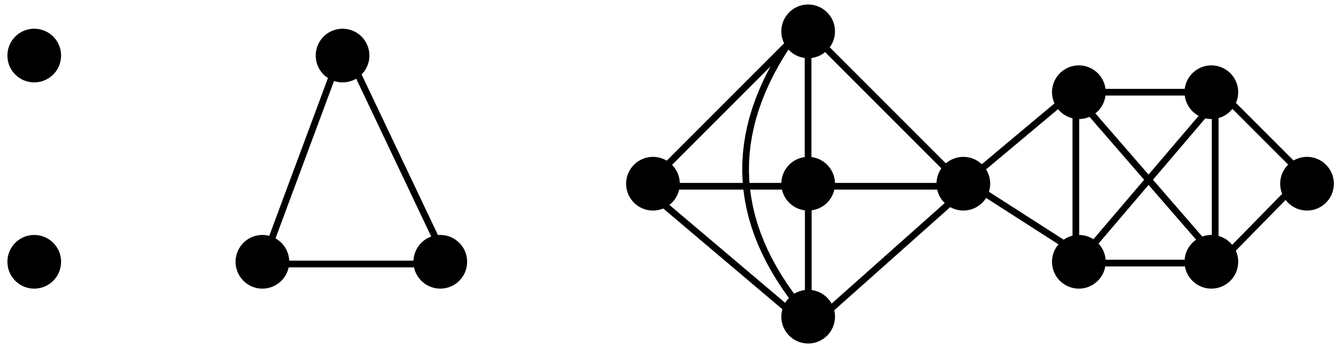


Edge Clique Cover

Input: A graph G, an integer k
Question: Does there exist k cliques in G, C_1, …, C_k, such that for each edge (u,v) there exists an $i \in \{1,\ldots,k\}$ such that $(u,v) \in C_i$?

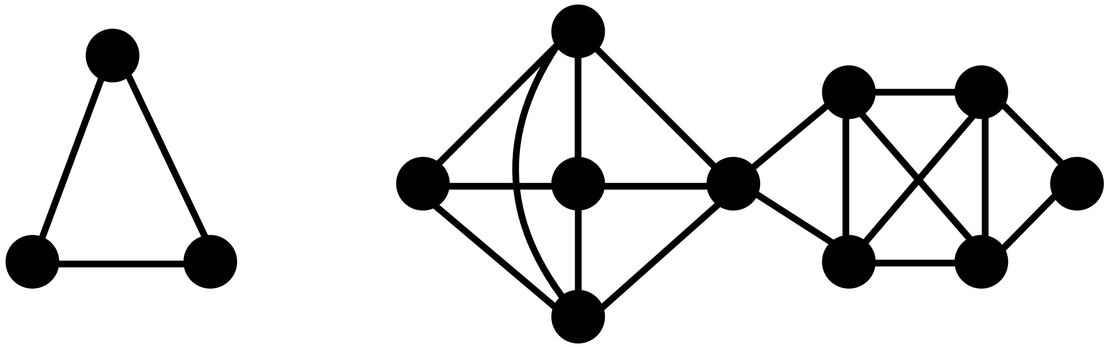


Edge Clique Cover



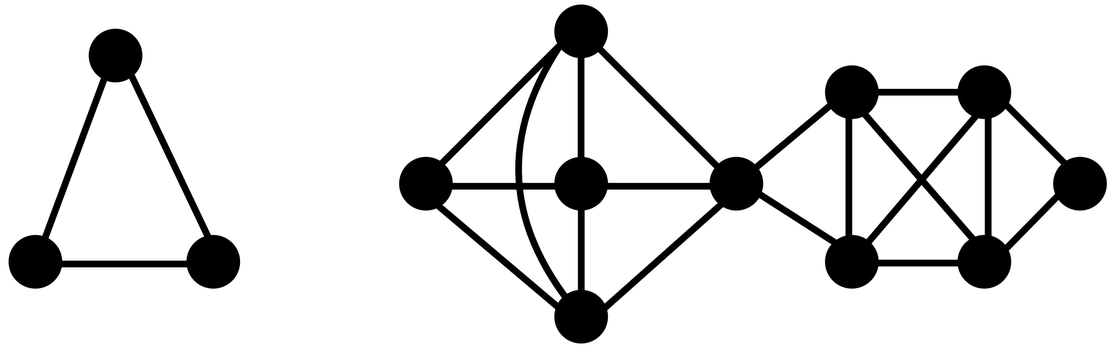
RR 1: Delete isolated vertices.

Edge Clique Cover



RR 1: Delete isolated vertices.

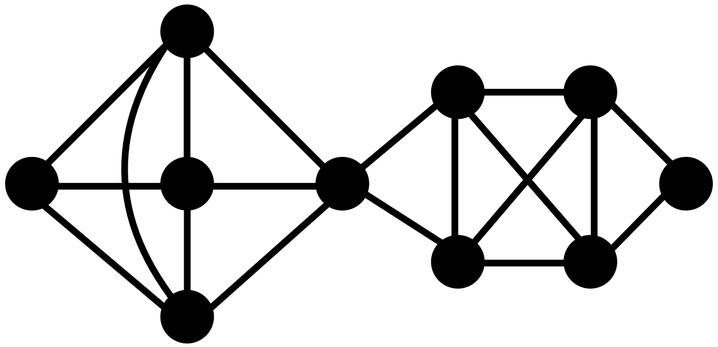
Edge Clique Cover



RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

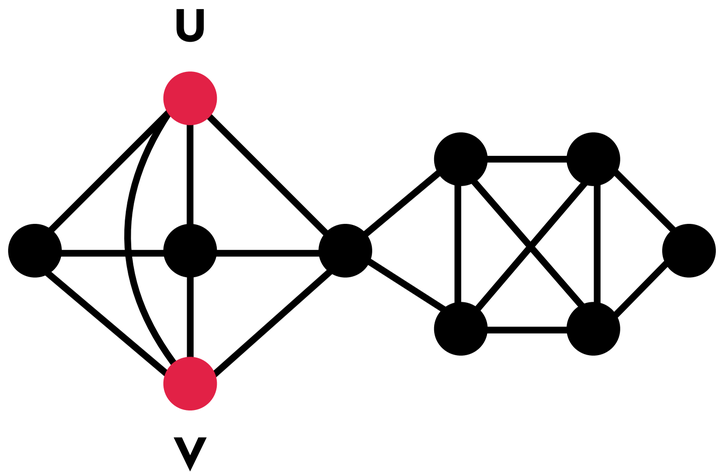
Edge Clique Cover



RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

Edge Clique Cover

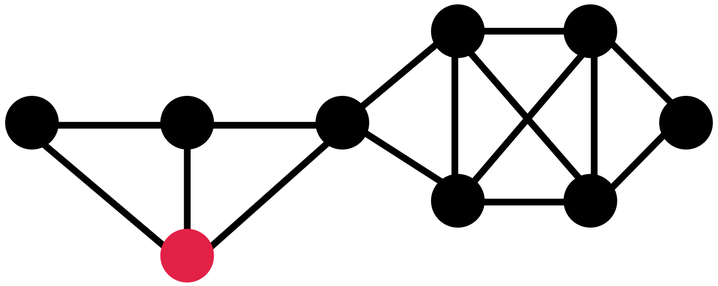


RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.

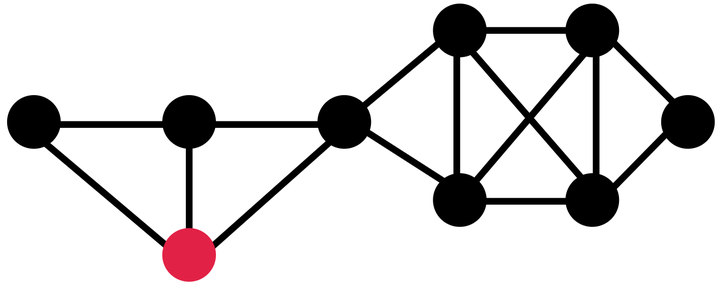
Edge Clique Cover



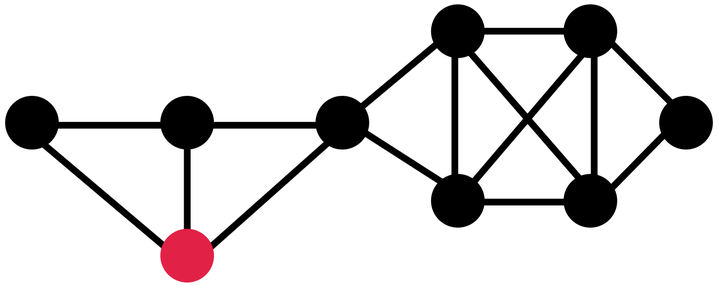
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



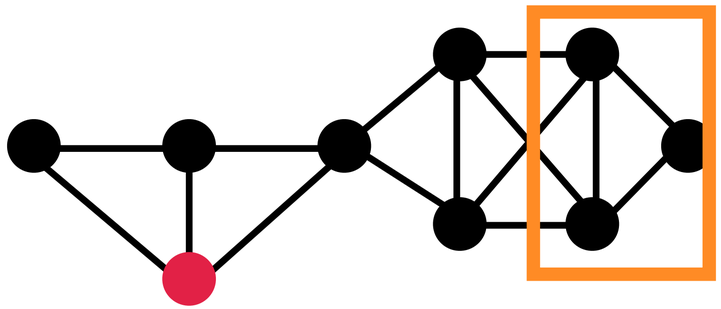
Edge Clique Cover



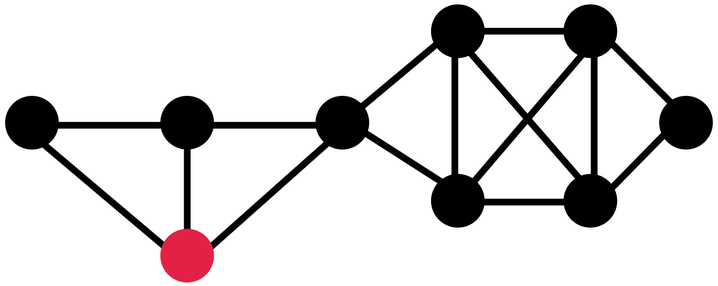
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



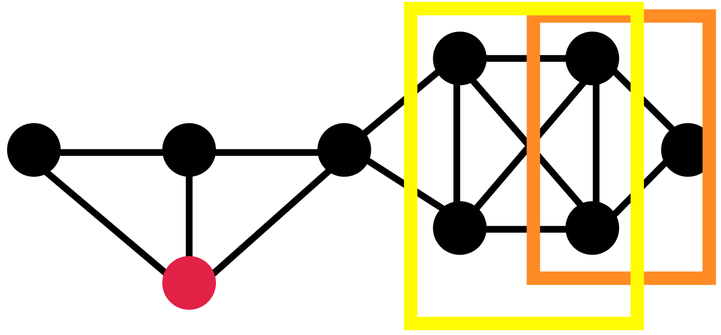
Edge Clique Cover



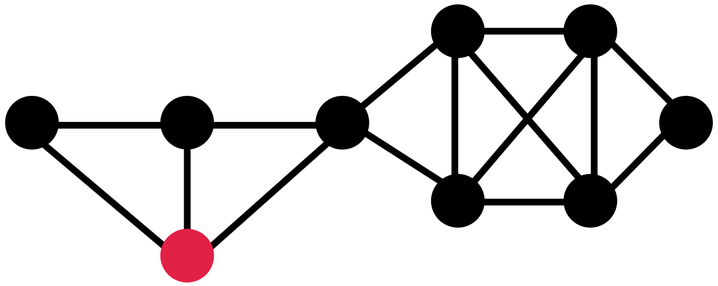
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



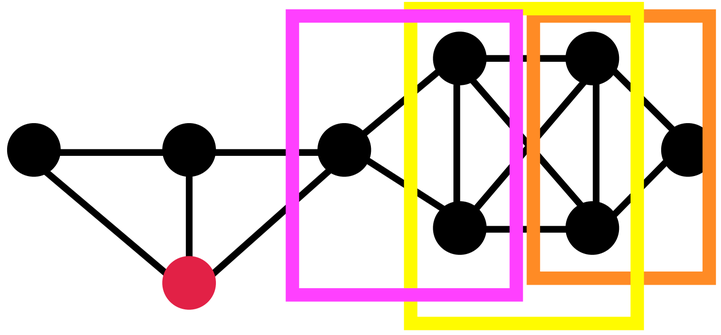
Edge Clique Cover



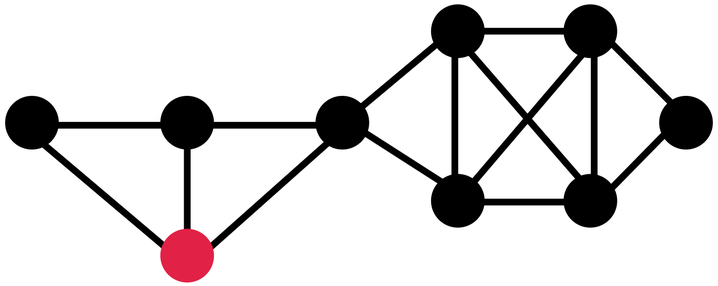
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



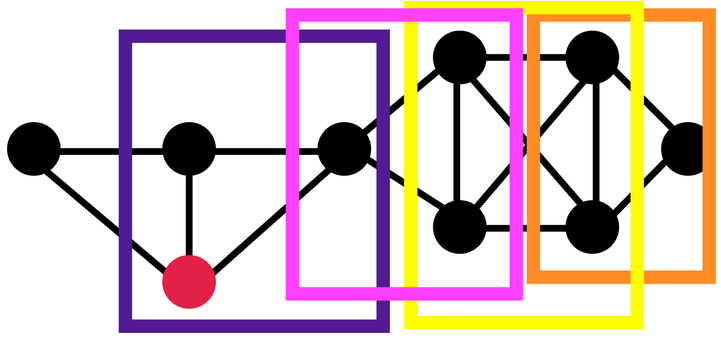
Edge Clique Cover



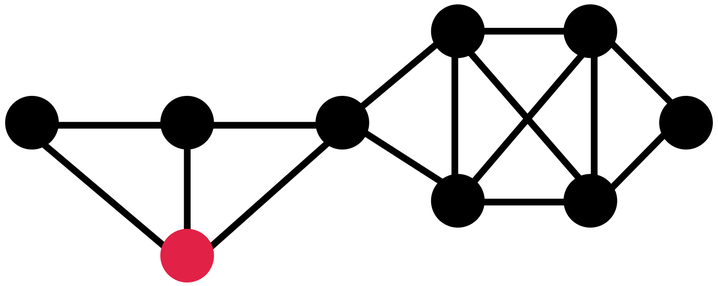
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



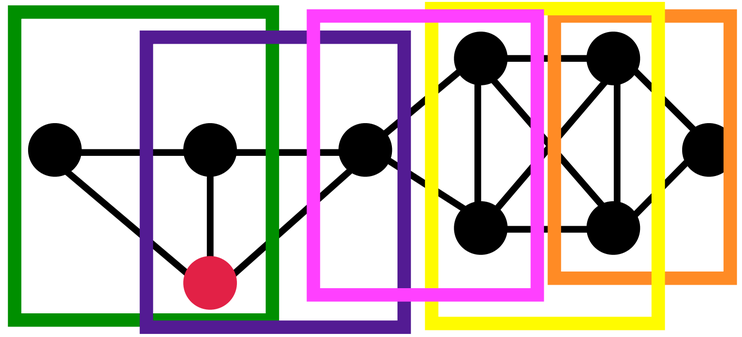
Edge Clique Cover



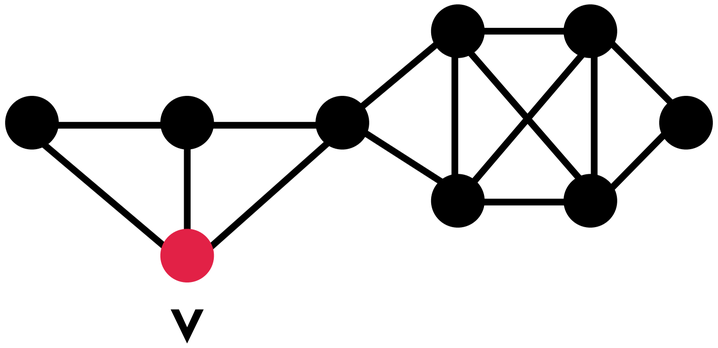
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



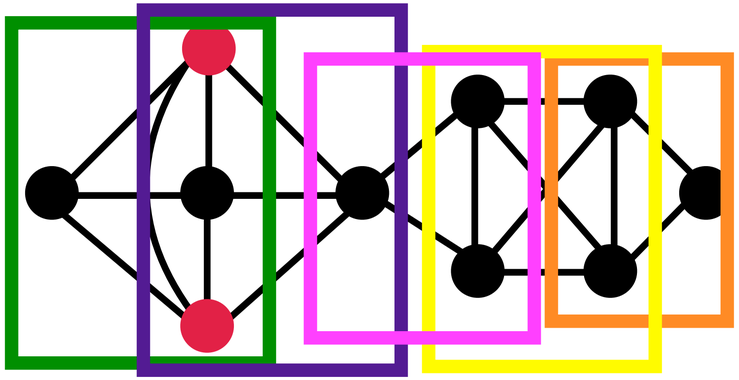
Edge Clique Cover



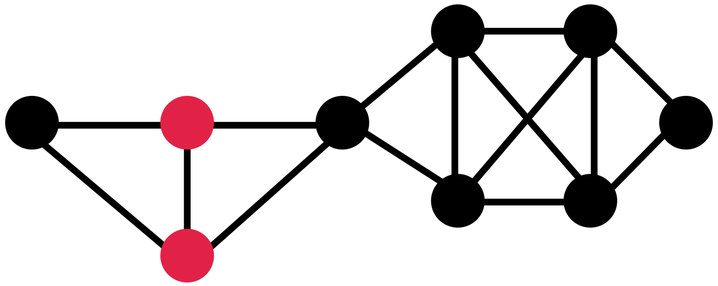
RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.



Edge Clique Cover

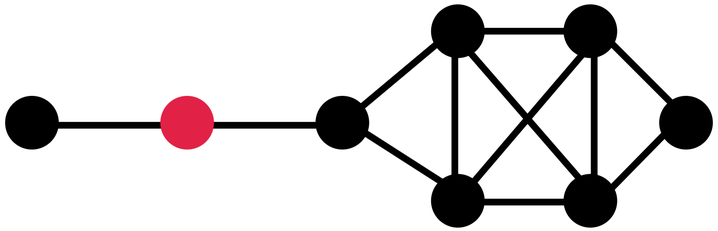


RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.

Edge Clique Cover

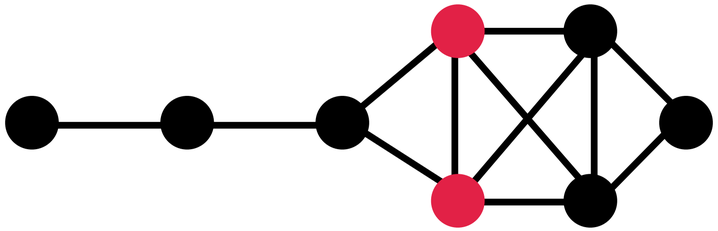


RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.

Edge Clique Cover

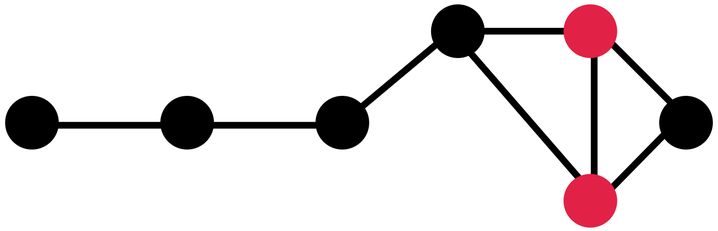


RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.

Edge Clique Cover

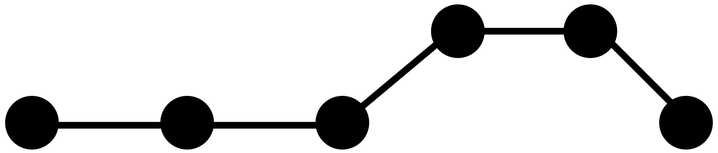


RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.

Edge Clique Cover



RR 1: Delete isolated vertices.

RR 2: Delete a connected component that is a clique and decrease the budget k by1.

RR 3: If there exists vertices u,v such that N[u]=N[v], delete u.

Let the reduced instance (with respect to RR 1, RR 2 and RR 3) have
n′ vertices and k′ budget.

RR 4: If $n' \geq 2^{k'} - 1$ then say No.

Edge Clique Cover

RR 4: If $n' \geq 2^{k'} - 1$ then say No.

Safeness of RR 4:

Suppose the reduced instance is a Yes-instance.
Let $C_1, \ldots, C_{k'}$ be the solution cliques.

Assign a k′ length bit vector to each vertex v.

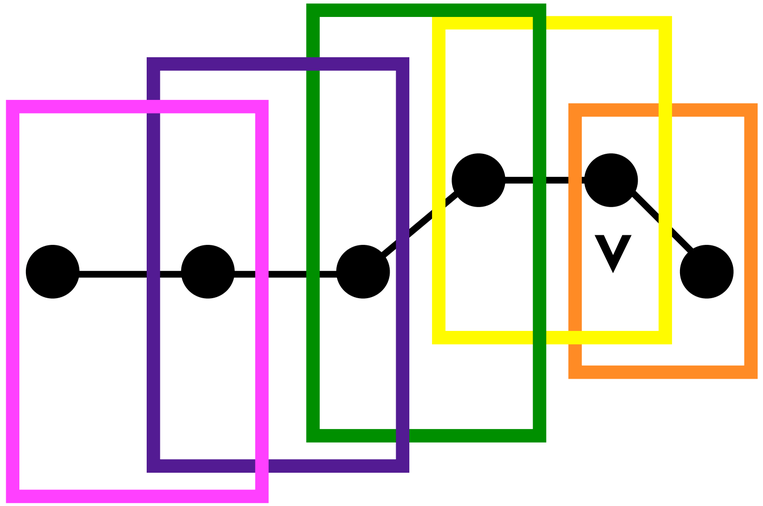
$vec(v) = (v \stackrel{?}{\in} C_1, v \stackrel{?}{\in} C_2, \ldots, v \stackrel{?}{\in} C_{k'})$

vec(v)=(1,1,0,0,0)

Total number of bit vectors of length $k' = 2^{k'}$

If v is isolated, vec(v)=(0,0,0,0,0).

If vec(u)=vec(v), then N[u]=N[v].



Edge Clique Cover

RR 4: If $n' \geq 2^{k'} - 1$ then say No.

Safeness of RR 4:

Suppose the reduced instance is a Yes-instance.
Let $C_1, \ldots, C_{k'}$ be the solution cliques.

Assign a k′ length bit vector to each vertex v.

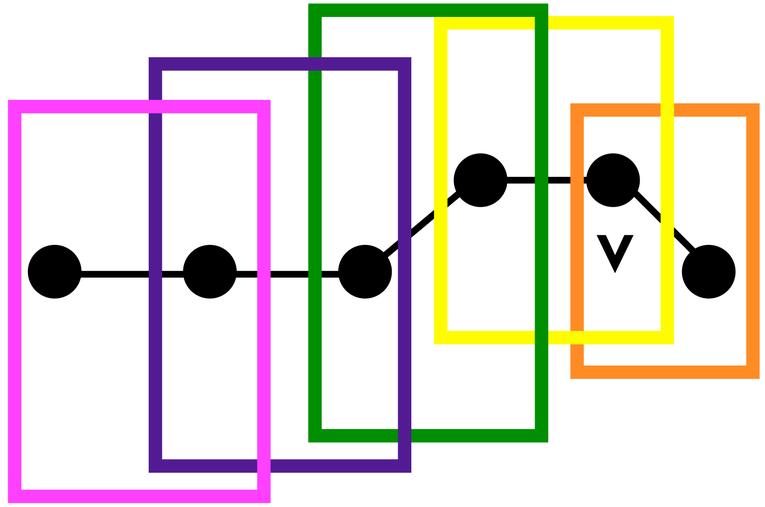
$vec(v) = (v \stackrel{?}{\in} C_1, v \stackrel{?}{\in} C_2, \ldots, v \stackrel{?}{\in} C_{k'})$

vec(v)=(1,1,0,0,0)

Total number of bit vectors of length $k' = 2^{k'}$

If v is isolated, vec(v)=(0,0,0,0,0).

If vec(u)=vec(v), then N[u]=N[v].



Thus, when none of the previous reduction rules are applicable, then each vertex gets a unique not-all-zero vector associated to it.

ECC admits a kernel with $2^k - 1$ vertices.

FPT and Kernelization

A parameterized problem is FPT if and only if it admits a kernel.

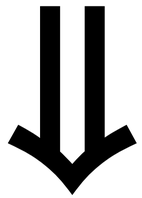
f(k) size kernel

Find a kernel
+
Brute force

$n^{O(1)} + g(k)$

A parameterized problem is FPT if and only if it admits a kernel.

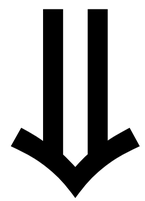
$f(k)\ n^{O(1)}$



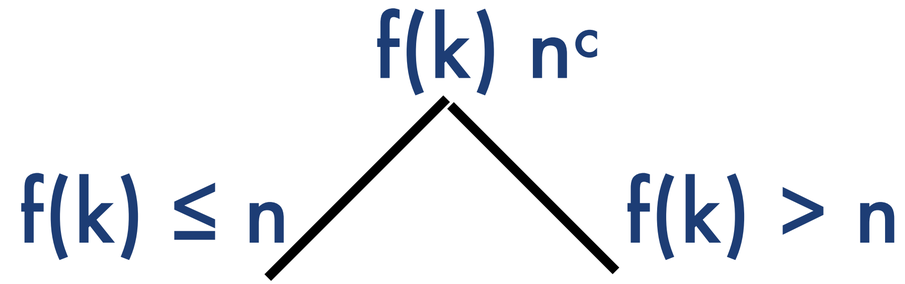
f(k) size kernel

A parameterized problem is FPT if and only if it admits a kernel.

$f(k)\, n^{O(1)}$



f(k) size kernel

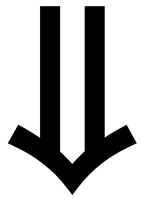


FPT algorithm is the kernelization algorithm. (constant size kernel)

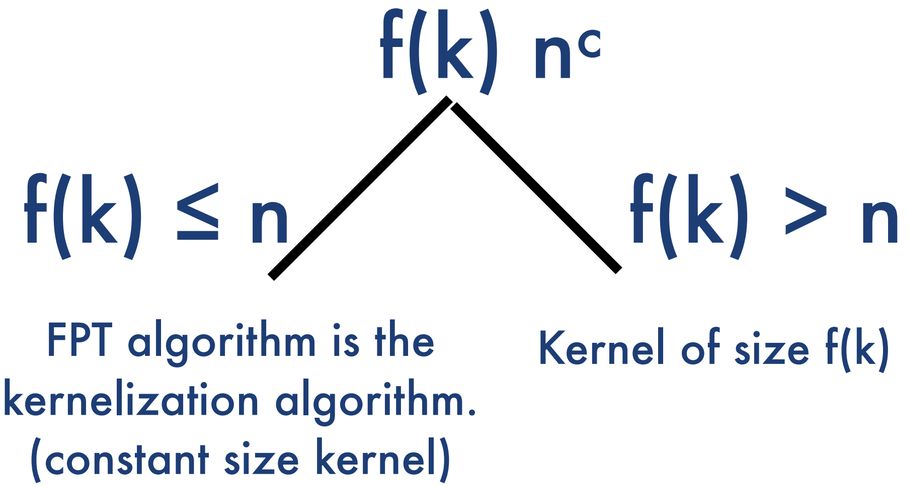
Kernel of size f(k)

A parameterized problem is FPT if and only if it admits a kernel.

$f(k)\, n^{O(1)}$



f(k) size kernel



Hence a search for polynomial kernels for FPT problems!

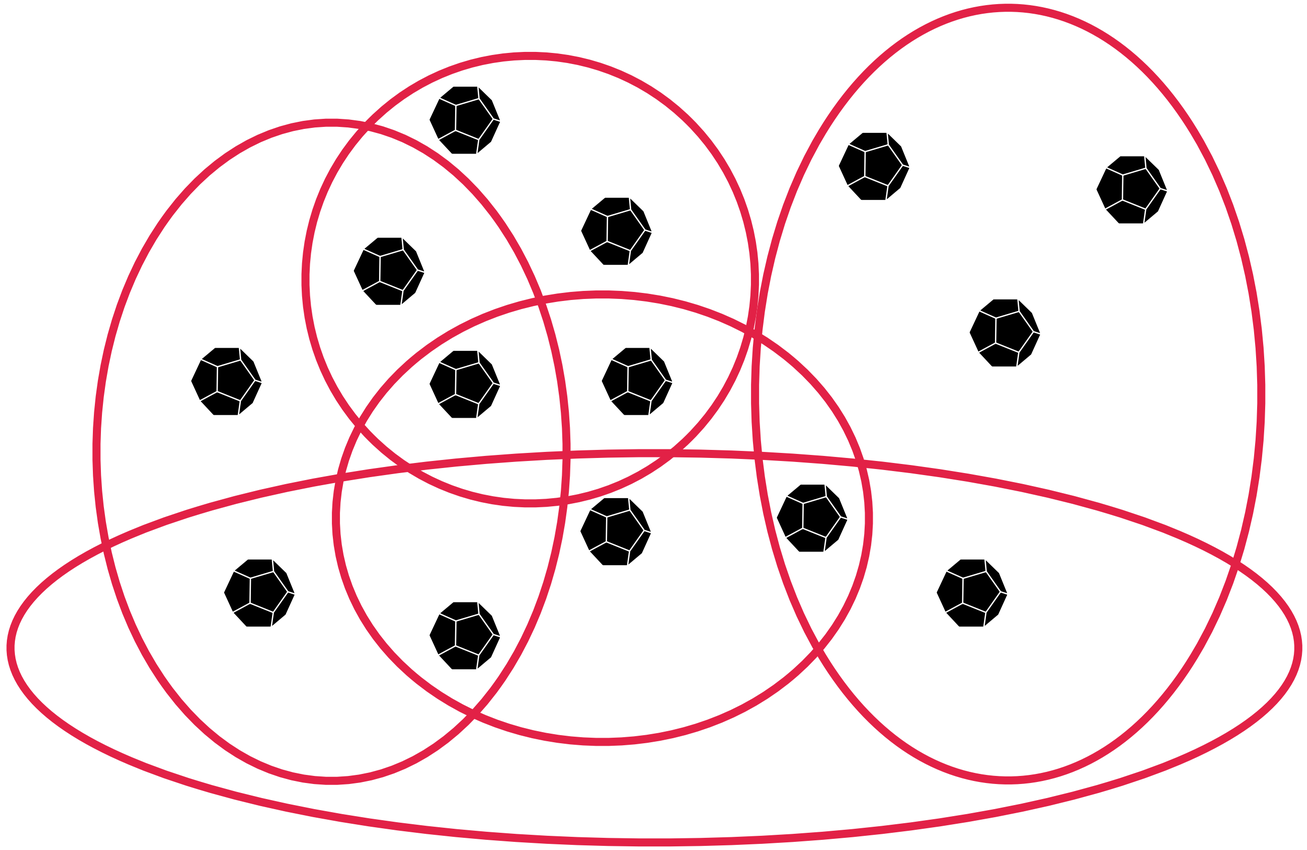
d-Hitting Set

d-Hitting Set

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist $X \subseteq U$ of size at most k such that for each $S \in \mathcal{F}$, $S \cap X \neq \Phi$?

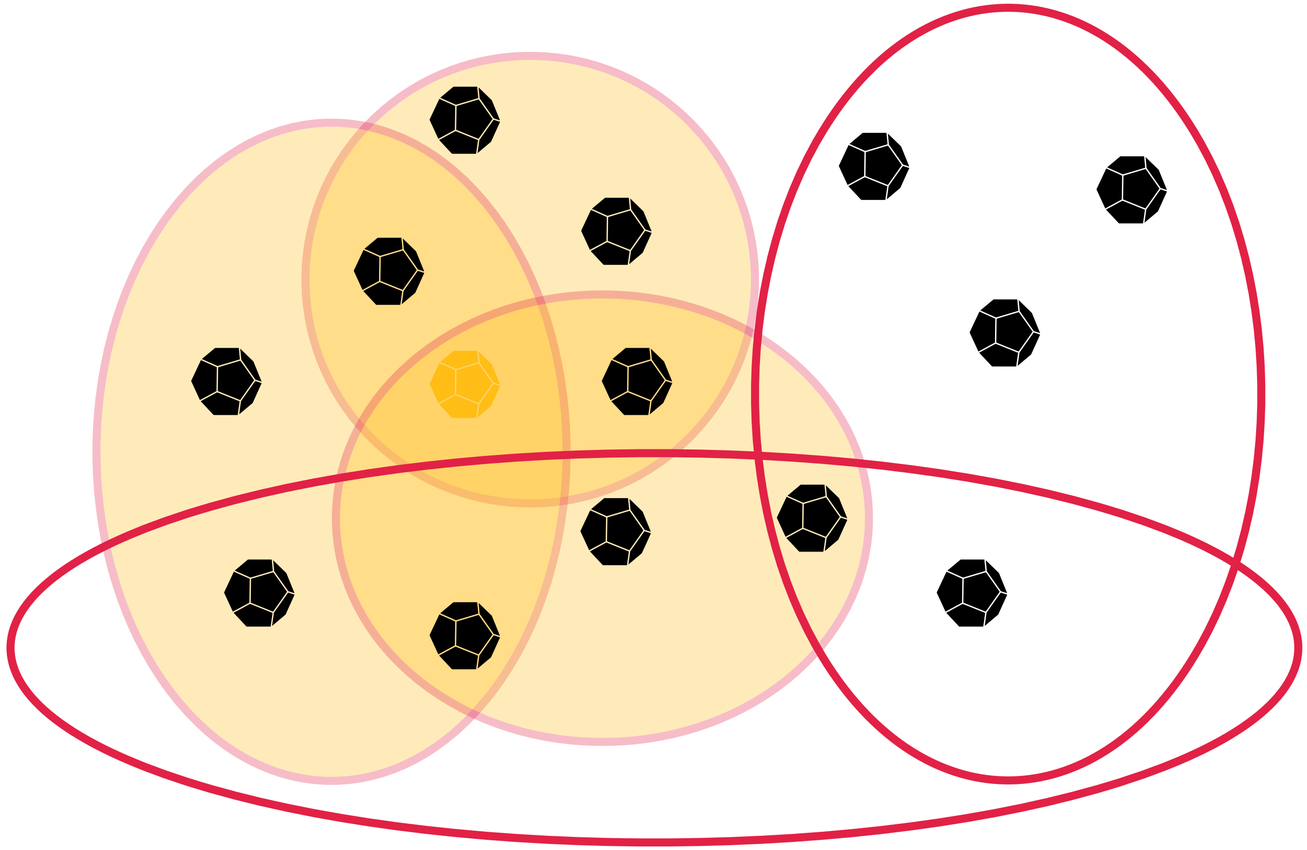
d-Hitting Set

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist $X \subseteq U$ of size at most k such that for each $S \in \mathcal{F}$, $S \cap X \neq \Phi$?



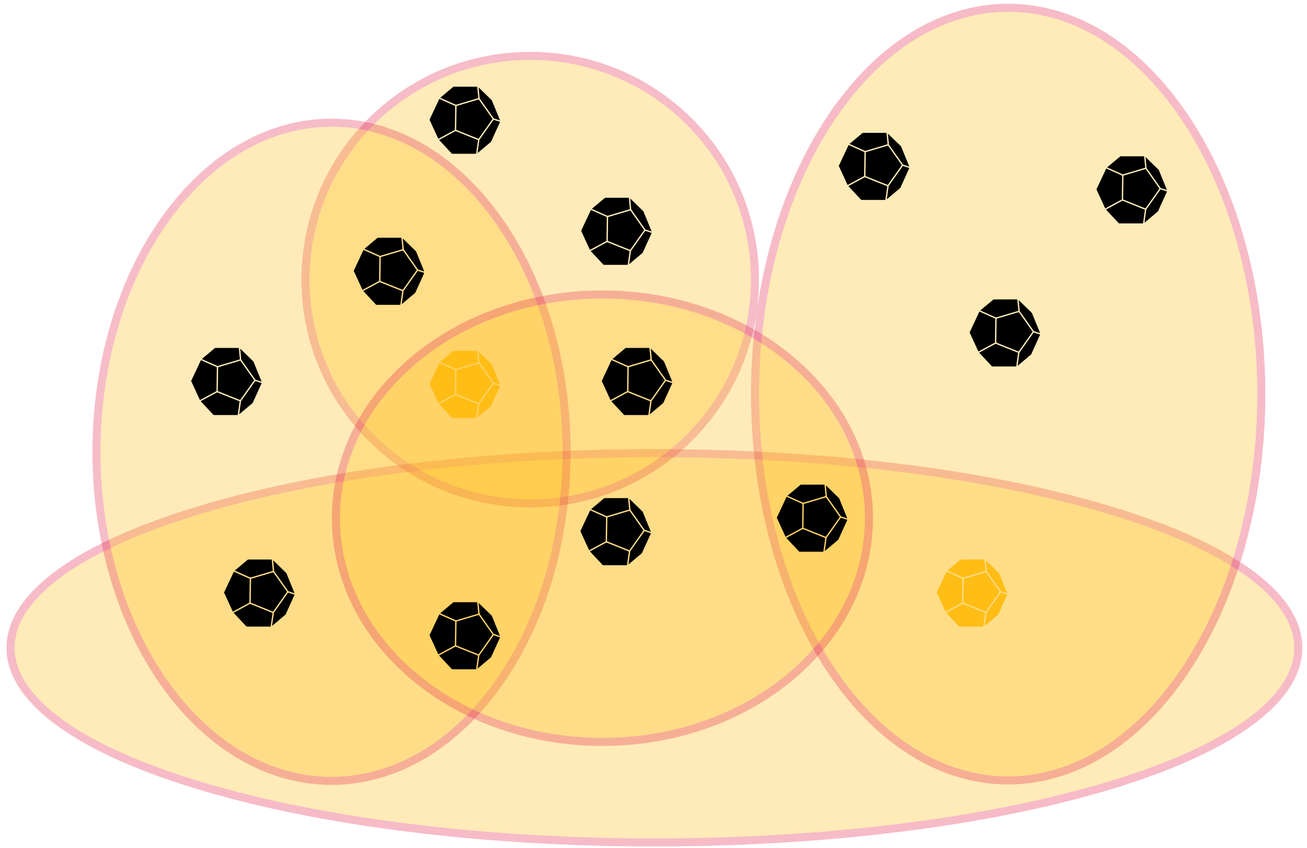
d-Hitting Set

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist $X \subseteq U$ of size at most k such that for each $S \in \mathcal{F}$, $S \cap X \neq \Phi$?



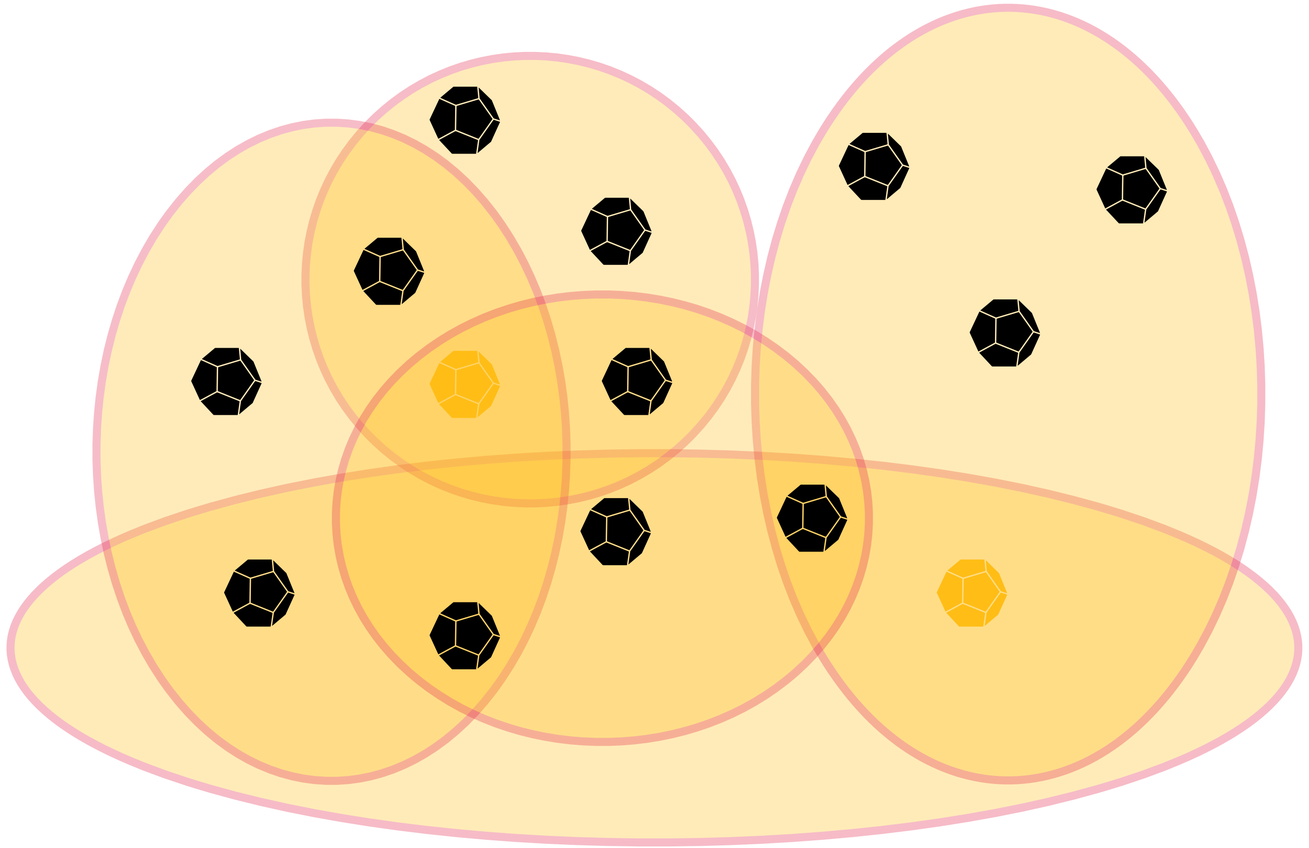
d-Hitting Set

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist $X \subseteq U$ of size at most k such that for each $S \in \mathcal{F}$, $S \cap X \neq \Phi$?



d-Hitting Set

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist $X \subseteq U$ of size at most k such that for each $S \in \mathcal{F}$, $S \cap X \neq \Phi$?



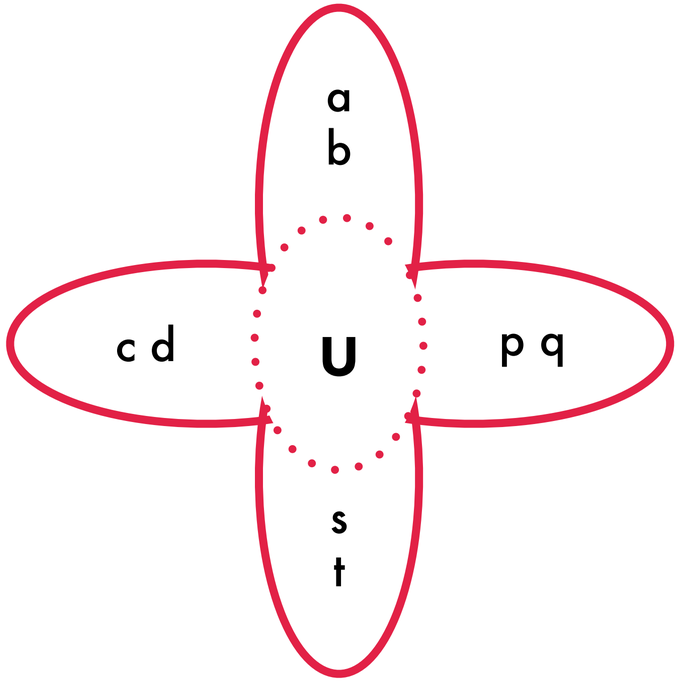
d=2 is the Vertex Cover problem.

Reduction rules for d-HS

RR 0: If there exists an element u and sets $S_1,\ldots, S_{k+1}$ such that $S_i \cap S_j = \{u\}$, for each $i,j \in \{1,\ldots,k+1\}$, then u must be in the solution.

Reduction rules for d-HS

RR 0: If there exists an element u and sets $S_1,\ldots, S_{k+1}$ such that $S_i \cap S_j = \{u\}$, for each $i,j \in \{1,\ldots,k+1\}$, then u must be in the solution.

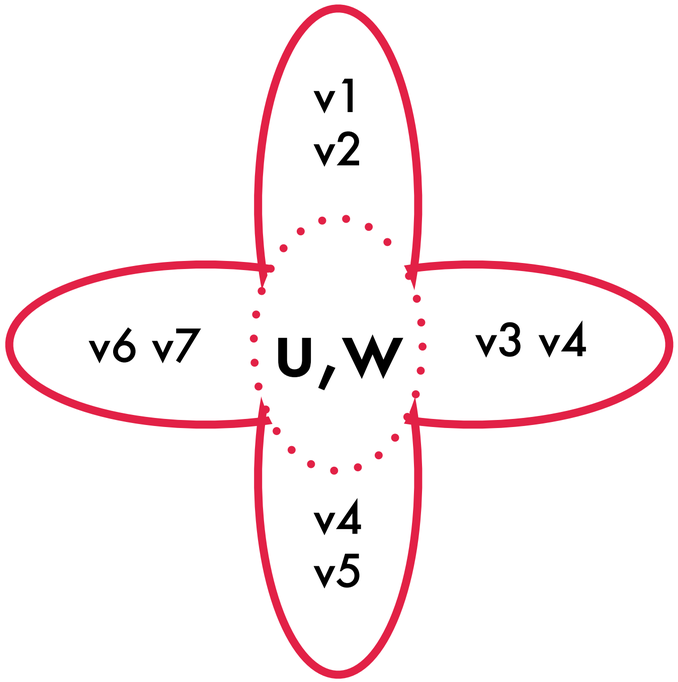


Reduction rules for d-HS

RR 1: If there exist sets $S_1,\ldots, S_{k+1}$ such that $S_i \cap S_j = \{C\}$, for each $i,j \in \{1,\ldots,k+1\}$, delete $S_1,\ldots,S_{k+1}$ from F and add C to F.
If $C = \varnothing$, say No.

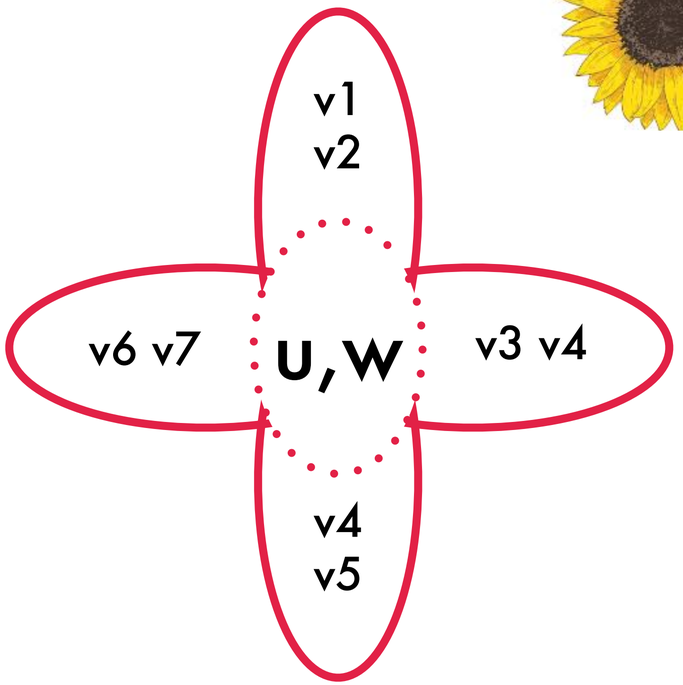
Reduction rules for d-HS

RR 1: If there exist sets $S_1,\ldots, S_{k+1}$ such that $S_i \cap S_j = \{C\}$, for each $i,j \in \{1,\ldots,k+1\}$, delete $S_1,\ldots,S_{k+1}$ from F and add C to F.
If C =∅, say No.



Reduction rules for d-HS

RR 1: If there exist sets $S_1,\ldots, S_{k+1}$ such that $S_i \cap S_j = \{C\}$, for each $i,j \in \{1,\ldots,k+1\}$, delete $S_1,\ldots,S_{k+1}$ from F and add C to F.
If $C = \emptyset$, say No.



Sunflower: A collection of sets $S_1,\ldots,S_r$ such that for each $i,j \in \{1,..,r\}$, $S_i \cap S_j = C$.

C is called the core of the sunflower. (could be empty)

S_i/C are called the petals of the sunflower.

Sunflower lemma

Every family with > $d!\, k^d$ sets of size exactly d contains a sunflower with k+1 petals.
Such a sunflower can be computed in polynomial time.

Sunflower lemma

Every family with > $d!\, k^d$ sets of size exactly d contains a sunflower with k+1 petals.
Such a sunflower can be computed in polynomial time.

Sunflower lemma

Every family with > $d!\, k^d$ sets of size exactly d contains a sunflower with $k+1$ petals.
Such a sunflower can be computed in polynomial time.

d-HS admits a kernel with $d!\, k^d\, d$ sets and $d!\, k^d\, d^2$ elements ($O(k^d)$ kernel).

Proof of the Sunflower Lemma

Every family with > $d!\, k^d$ sets of size exactly d contains a sunflower with k+1 petals.
Such a sunflower can be computed in polynomial time.

Proof of the Sunflower Lemma

Every family with > $d!\, k^d$ sets of size exactly d contains a sunflower with k+1 petals.
Such a sunflower can be computed in polynomial time.

Every family $\mathcal{F}$ with > $d!\, k^d$ sets of size d contains a sunflower with $k+1$ petals.

Induction on d

Every family $\mathcal{F}$ with > $d!\, k^d$ sets of size d contains a sunflower with k+1 petals.

Induction on d

Base case (d=1): $\mathcal{F}$ has > k singleton sets =>
Sunflower with k+1 petals and empty core.

Every family $\mathcal{F}$ with > $d!\, k^d$ sets of size d contains a sunflower with $k+1$ petals.

Induction on d

Base case (d=1): $\mathcal{F}$ has > k singleton sets =>
Sunflower with k+1 petals and empty core.

Induction step: SUPPOSE there exists an element v that is present
$> (d-1)!\, k^{d-1}$ sets of $\mathcal{F}$.

Every family $\mathcal{F}$ with > $d!\, k^d$ sets of size d contains a sunflower with $k+1$ petals.

Induction on d

Base case (d=1): $\mathcal{F}$ has > k singleton sets =>
Sunflower with k+1 petals and empty core.

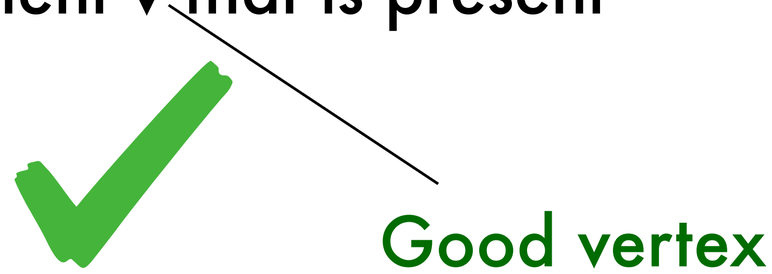
Induction step: SUPPOSE there exists an element v that is present
> $(d-1)!\, k^{d-1}$ sets of $\mathcal{F}$.

Every family $\mathcal{F}$ with > $d!\ k^d$ sets of size d contains a sunflower with $k+1$ petals.

Induction on d

Base case (d=1): $\mathcal{F}$ has > k singleton sets =>
Sunflower with k+1 petals and empty core.

Induction step: SUPPOSE there exists an element v that is present
$> (d-1)!\ k^{d-1}$ sets of $\mathcal{F}$.



Good vertex

Every family $\mathcal{F}$ with > $d!\, k^d$ sets of size d
either contains a sunflower with k+1 petals and empty core
or there exists a good vertex.

Greedy

$S_1, \ldots, S_r$ be pairwise disjoint sets of $\mathcal{F}$ (maximal)

$r \geq k+1$ ✓

$r \leq k$

Every family $\mathcal{F}$ with > d! k^d sets of size d
either contains a sunflower with k+1 petals and empty core
or there exists a good vertex.

Greedy

$S_1,\ldots,S_r$ be pairwise disjoint sets of $\mathcal{F}$ (maximal)

r≥ k+1 ✓

r≤ k

$S_1,\ldots,S_r$, r≤ k

Every set of $\mathcal{F}$ intersects some S_i

$$X = \cup_{i\in\{1,\ldots,r\}} S_i$$

$$|X| \leq dk$$

Pigeonhole Principle

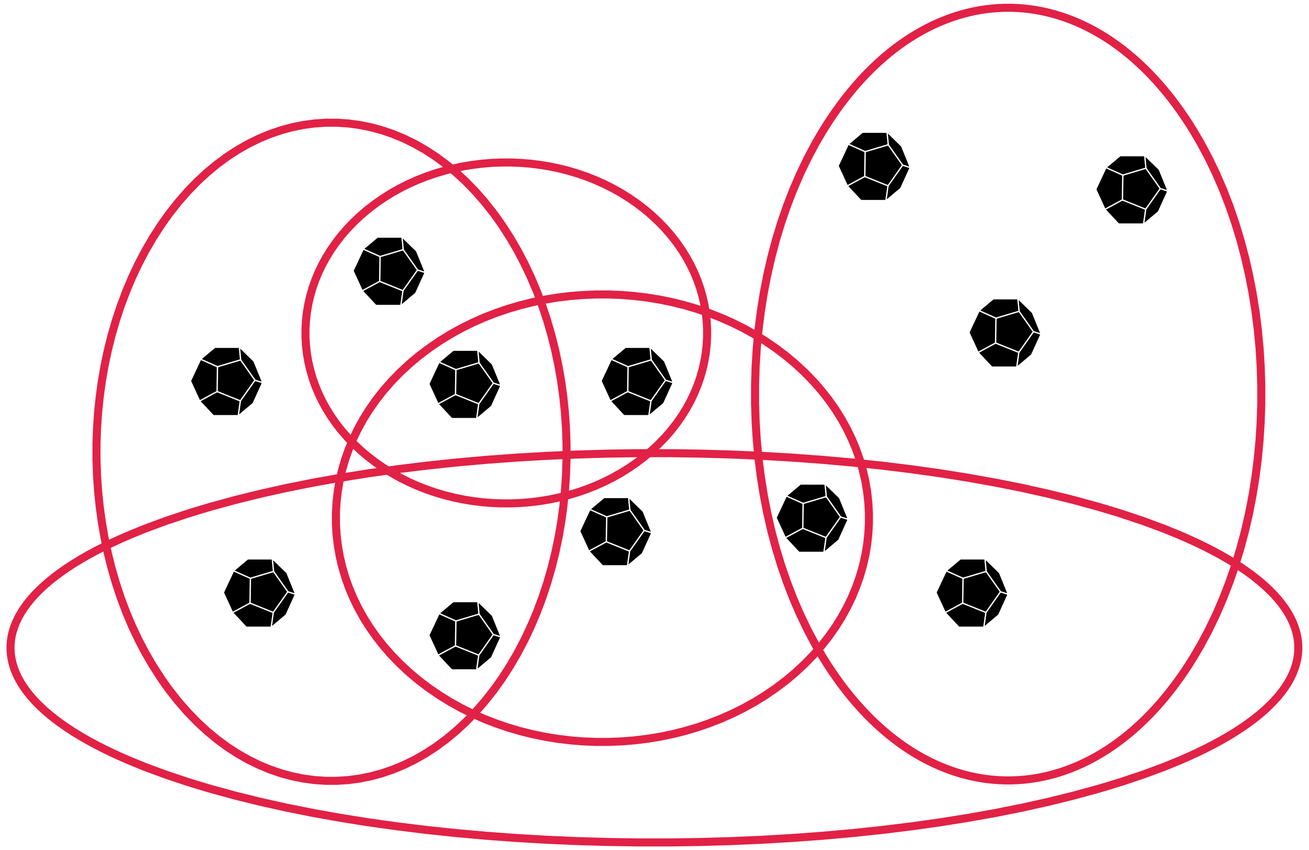
∃ x ∈ X such that x is present in > $|\mathcal{F}|/|X|$ sets

$|\mathcal{F}|/|X|$ > d! k^d/dk = (d-1)! k^{d-1}

d-Set Packing

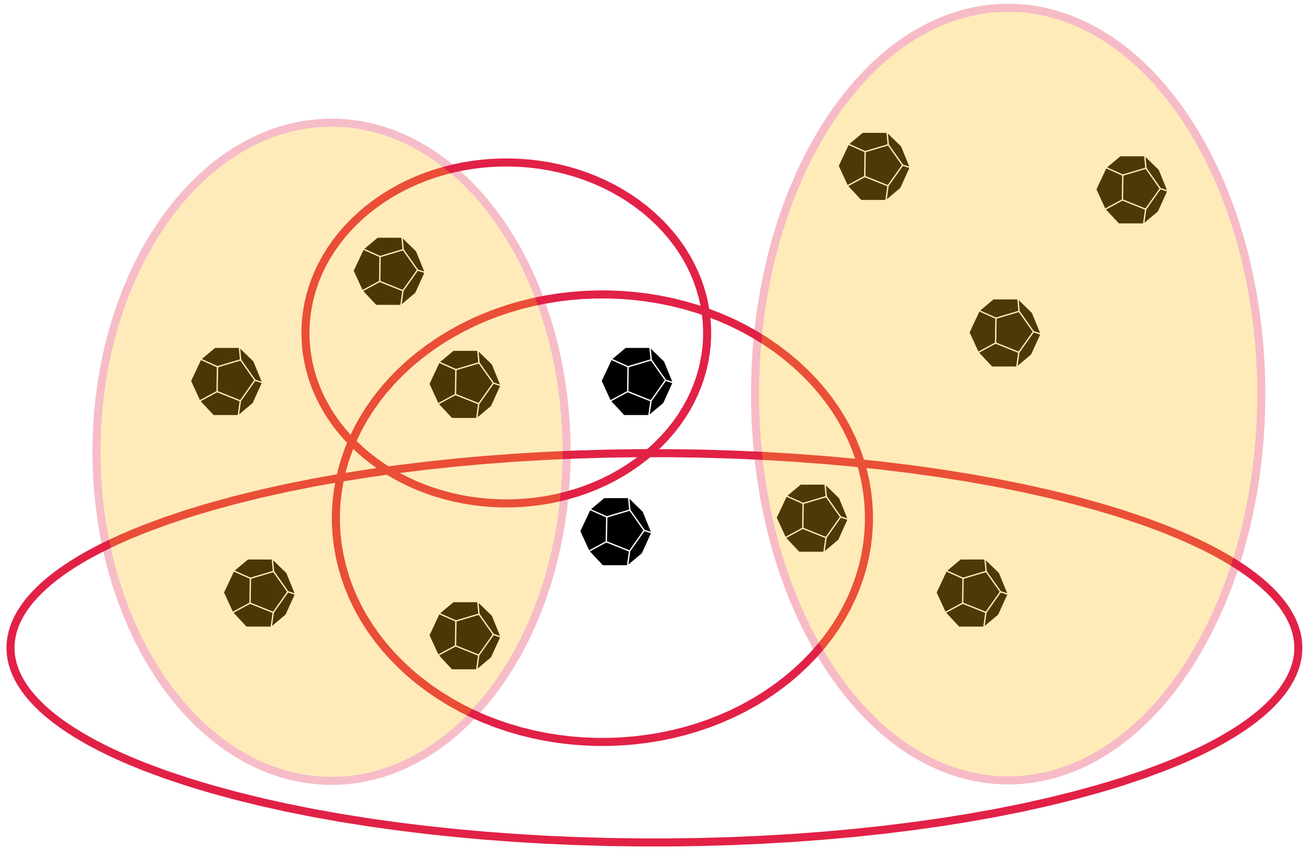
d-Set Packing

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist at least k pairwise disjoint sets in $\mathcal{F}$?



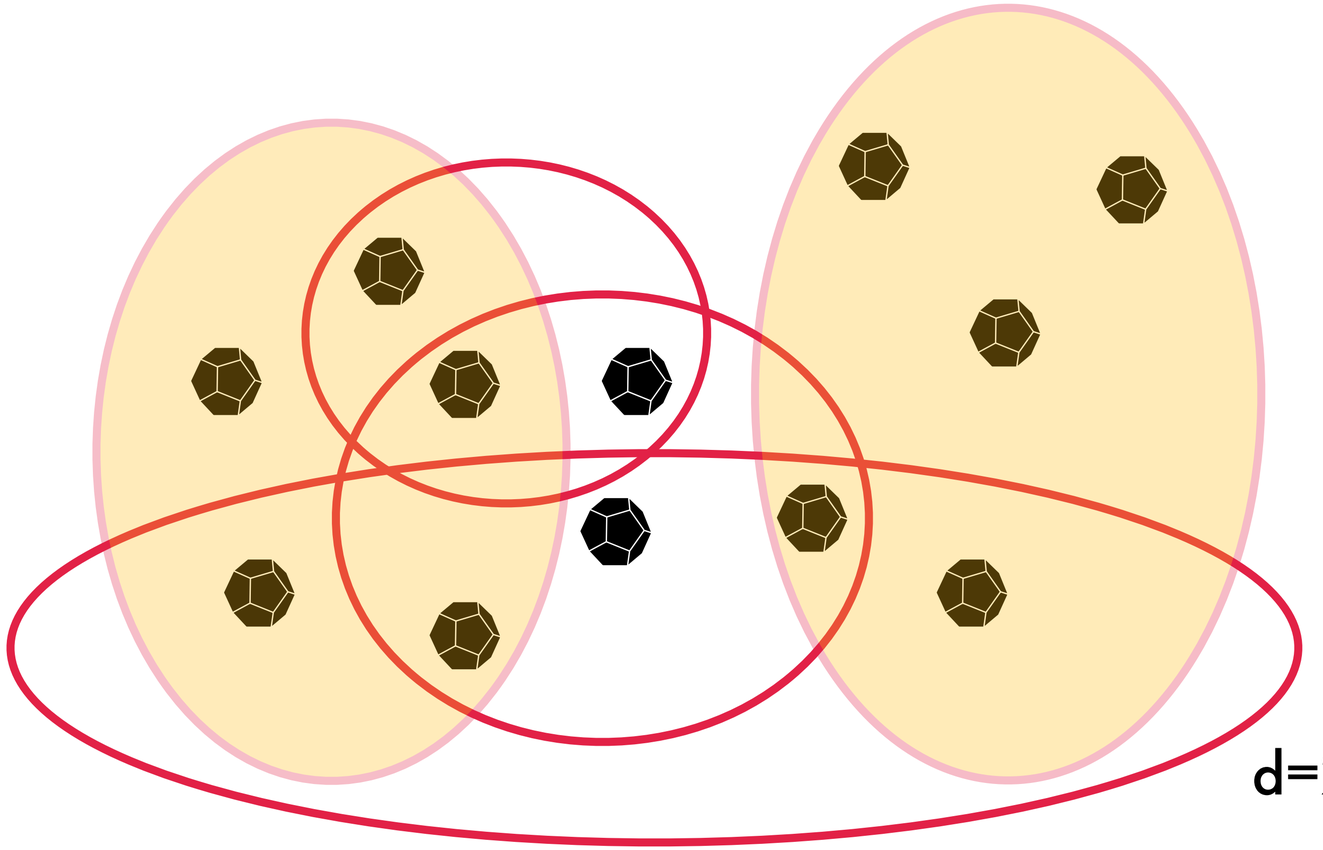
d-Set Packing

Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist at least k pairwise disjoint sets in $\mathcal{F}$?



d-Set Packing

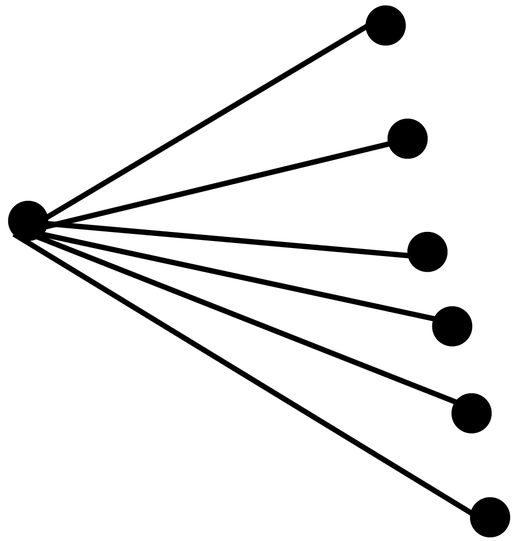
Input: Universe U, family $\mathcal{F}$ of subsets of U of size at most d, integer k
Question: Does there exist at least k pairwise disjoint sets in $\mathcal{F}$?



d=2 is the k-Matching problem.

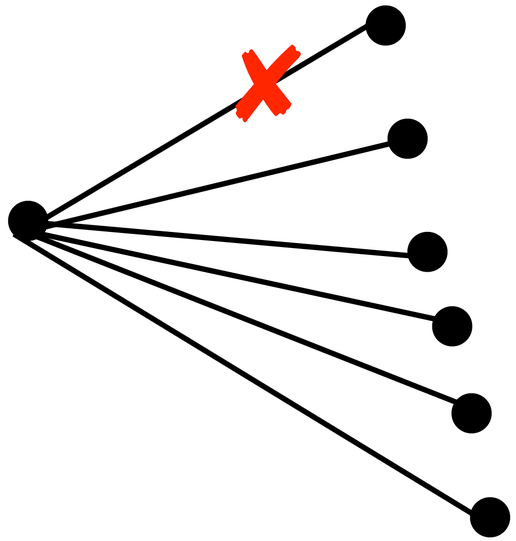
“Buss rule” for 2-Set Packing

RR: If there exists a vertex with degree at least 2k, then delete an edge incident on it.



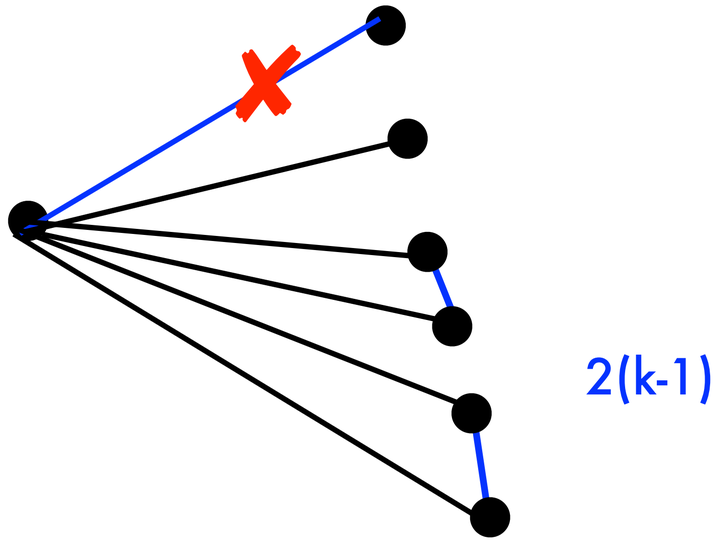
"Buss rule" for 2-Set Packing

RR: If there exists a vertex with degree at least 2k, then delete an edge incident on it.



"Buss rule" for 2-Set Packing

RR: If there exists a vertex with degree at least 2k, then delete an edge incident on it.



2(k-1)

d-Set Packing

RR 1: If a family contains a sunflower with > d(k-1)+2 petals, then one of these sets can be deleted.

Exercise: Show that RR 1 is safe.

d-Set Packing

RR 1: If a family contains a sunflower with > d(k-1)+2 petals, then one of these sets can be deleted.

Exercise: Show that RR 1 is safe.

d-SP admits a kernel $d!\,(dk-d+1)^d\,d$ sets and $d!\,(dk-d+1)^d\,d^2$ elements ($O(k^{O(d)})$ kernel).