Kernelizations for Parameterized Counting Problems

Zachary Frenette

University of Waterloo

March 11, 2014

Overview

1. Counting Kernelizations - Definitions and Concepts

2. Counting Kernelization Examples
   - The Vertex Cover Problem
   - The Hitting Set Problem
   - The Unique Hitting Set Problem

3. Conclusions and Future Work

1. Counting Kernelizations - Definitions and Concepts

2. Counting Kernelization Examples
   - The Vertex Cover Problem
   - The Hitting Set Problem
   - The Unique Hitting Set Problem

3. Conclusions and Future Work

Motivation

Recall the concepts of kernelization and fixed parameter tractability for regular decision problems.

Motivation

Recall the concepts of kernelization and fixed parameter tractability for regular decision problems.

Definition

For a decision problem $Q \subseteq \Sigma^*$ and parameterization $k \in \mathbb{N}$, a kernelization is a mapping $\Psi : \Sigma^* \to \Sigma^*$ satisfying the following three conditions.

- Ψ *is a function computable in polynomial time.*
- *There exists a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $|\Psi(x)| \leq g(k)$ for all $x \in \Sigma^*$.*
- $x \in Q \iff \Psi(x) \in Q$ *for all $x \in \Sigma^*$.*

We say that Q is kernelizable if there exists such a mapping Ψ for Q.

Motivation

Furthermore, we have seen the following theorem relating kernelization and fixed parameter tractability together.

Motivation

Furthermore, we have seen the following theorem relating kernelization and fixed parameter tractability together.

Theorem

Let FPT denote the class of all fixed parameter tractable decision problems and let (Q, k) be a parameterized decision problem. Then $(Q, k) \in FPT$ if and only if (Q, k) is kernelizable.

Motivation

Furthermore, we have seen the following theorem relating kernelization and fixed parameter tractability together.

Theorem

Let FPT denote the class of all fixed parameter tractable decision problems and let (Q, k) be a parameterized decision problem. Then $(Q, k) \in FPT$ if and only if (Q, k) is kernelizable.

With the success of kernelization in the design of fixed parameter tractable algorithms for decision problems, we would like to generalize and expand this notion to counting problems as well.

Counting Kernelizations

We begin with some preliminary definitions, some of which may seem arbitrary for the time being.

Counting Kernelizations

We begin with some preliminary definitions, some of which may seem arbitrary for the time being.

Definition

Let $F : \Sigma^* \to \mathbb{N}$ be a counting problem and let $k \in \mathbb{N}$ be a parameterization of F.

1. *A mapping $\Psi : \Sigma^* \to \Sigma^*$ is k-bounded if there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $|\Psi(x)| \leq g(k)$ for all $x \in \Sigma^*$.*
2. *A relation $Y \subseteq \Sigma^* \times \{0,1\}^*$ is Ψ-aware for a k-bounded mapping Ψ if there are computable functions $f : \mathbb{N} \to \mathbb{N}$ and $h : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \Sigma^*$ and every $z \in \{0,1\}^*$:*
   - *Given $\Psi(x)$, it can be decided in time $\mathcal{O}(f(k))$ whether or not $(\Psi(x), z) \in Y$ holds.*
   - *If $(\Psi(x), z) \in Y$, then $|z| \leq h(k)$.*

Counting Kernelizations

As it stands, we are still missing a characterization of the solutions of the reduced kernel instance $\Psi(x)$. The notion of Ψ-awareness, in conjunction with the notion of witnesses, will be used to create this desired characterization. Informally speaking, we can think of a witness as an encoding of a solution for a particular problem instance.

Counting Kernelizations

As it stands, we are still missing a characterization of the solutions of the reduced kernel instance $\Psi(x)$. The notion of Ψ-awareness, in conjunction with the notion of witnesses, will be used to create this desired characterization. Informally speaking, we can think of a witness as an encoding of a solution for a particular problem instance.

Definition

Let $Y \subseteq \Sigma^* \times \{0,1\}^*$ be a relation. For any $x \in \Sigma^*$, we define $w_Y(x) := \{z : (x,z) \in Y\}$ to be the set of witnesses of x in Y.

Counting Kernelizations

Furthermore, it is possible for the reduced kernel instance $\Psi(x)$ to have fewer solutions than x itself! In other words, there may not necessarily be a bijective correspondence between the solutions of x and $\Psi(x)$.

Counting Kernelizations

Furthermore, it is possible for the reduced kernel instance $\Psi(x)$ to have fewer solutions than x itself! In other words, there may not necessarily be a bijective correspondence between the solutions of x and $\Psi(x)$.
Hence, we need a function μ such that, for each solution z of $\Psi(x)$, μ computes the number of solutions of x that are represented by z.

Counting Kernelizations

Furthermore, it is possible for the reduced kernel instance $\Psi(x)$ to have fewer solutions than x itself! In other words, there may not necessarily be a bijective correspondence between the solutions of x and $\Psi(x)$.
Hence, we need a function μ such that, for each solution z of $\Psi(x)$, μ computes the number of solutions of x that are represented by z.
Putting everything together gives the following definition of counting kernelization for a given parameterized counting problem.

Counting Kernelizations

Definition

Let $F : \Sigma^* \to \mathbb{N}$ be a counting problem and let $k \in \mathbb{N}$ be a parameterization of F. A counting kernelization of (F, k) is a pair (μ, Ψ) of computable functions $\mu : \Sigma^* \times \Sigma^* \times \{0,1\}^* \to \mathbb{N}$ and $\Psi : \Sigma^* \to \Sigma^*$ such that the following three conditions are satisfied.

1. *μ and Ψ are computable in polynomial time.*
2. *Ψ is a k-bounded mapping.*
3. *There is a Ψ-aware relation $Y \subseteq \Sigma^* \times \{0,1\}^*$ such that:*

$$F(x) = \sum_{z \in w_Y(\Psi(x))} \mu(x, \Psi(x), z)$$

We call $|\Psi(x)|$ the size of the kernel.

Counting Kernelizations

Just as in the case of kernelizations for decision problems, there is a theorem that characterizes the relationship between counting kernelizations and fixed parameter tractability.

Counting Kernelizations

Just as in the case of kernelizations for decision problems, there is a theorem that characterizes the relationship between counting kernelizations and fixed parameter tractability.

Theorem

Let FFPT denote the class of all fixed parameter tractable counting problems and let (F, k) be a parameterized counting problem. Then $(F, k) \in$ FFPT if and only if (F, k) has a counting kernelization.

Counting Kernelizations

Just as in the case of kernelizations for decision problems, there is a theorem that characterizes the relationship between counting kernelizations and fixed parameter tractability.

Theorem

Let FFPT denote the class of all fixed parameter tractable counting problems and let (F, k) be a parameterized counting problem. Then $(F, k) \in$ FFPT if and only if (F, k) has a counting kernelization.

Though this theorem may not provide any practical purpose, it does however provide an interesting theoretical result relating fixed parameter tractability and counting kernelizations.

1. Counting Kernelizations - Definitions and Concepts

2. Counting Kernelization Examples
   - The Vertex Cover Problem
   - The Hitting Set Problem
   - The Unique Hitting Set Problem

3. Conclusions and Future Work

Vertex Cover - Problem Definition

In this section, we will see how to adapt the Crown Rule Reduction to counting problems. In particular, we will see how to apply this kernelization technique to count vertex covers.

Vertex Cover - Problem Definition

In this section, we will see how to adapt the Crown Rule Reduction to counting problems. In particular, we will see how to apply this kernelization technique to count vertex covers.

Problem

p-#VertexCover
Input: A graph $\mathcal{G} := (V, E)$ *and* $k \in \mathbb{N}$
Parameter: k
Question: Compute the number of vertex covers of cardinality k *in* $\mathcal{G}$

Vertex Cover - Problem Definition

In this section, we will see how to adapt the Crown Rule Reduction to counting problems. In particular, we will see how to apply this kernelization technique to count vertex covers.

Problem

p-#VertexCover
Input: A graph $\mathcal{G} := (V, E)$ *and* $k \in \mathbb{N}$
Parameter: k
Question: Compute the number of vertex covers of cardinality k *in* $\mathcal{G}$

We will use the notation $vc(\mathcal{G})$ to denote the size of a minimum vertex cover in $\mathcal{G}$, and we will also use $\#vc(\mathcal{G}, k)$ to denote the number of vertex covers of size k in $\mathcal{G}$.

The Crown Reduction Rule

We begin by recalling the definition of a crown in $\mathcal{G}$.

The Crown Reduction Rule

We begin by recalling the definition of a crown in $\mathcal{G}$.

Definition

Let $\mathcal{G} := (V, E)$ be a graph. A crown in $\mathcal{G}$ is a bipartite subgraph $\mathcal{C} := (I, N(I), F)$ of $\mathcal{G}$ satisfying the following three conditions.

1. *I is an independent set in $\mathcal{G}$ and $N(I)$ is the set of all neighbours of vertices from I in $\mathcal{G}$.*
2. *F contains all of the edges of E that connect vertices from I to $N(I)$.*
3. *$\mathcal{C}$ has a matching of cardinality $|N(I)|$.*

The Crown Reduction Rule

We begin by recalling the definition of a crown in $\mathcal{G}$.

Definition

Let $\mathcal{G} := (V, E)$ be a graph. A crown in $\mathcal{G}$ is a bipartite subgraph $\mathcal{C} := (I, N(I), F)$ of $\mathcal{G}$ satisfying the following three conditions.

1. *I is an independent set in $\mathcal{G}$ and $N(I)$ is the set of all neighbours of vertices from I in $\mathcal{G}$.*
2. *F contains all of the edges of E that connect vertices from I to $N(I)$.*
3. *$\mathcal{C}$ has a matching of cardinality $|N(I)|$.*

For simplicity, we will asume that $\mathcal{G}$ has no isolated vertices - The paper shows how to handle this case.

The Crown Reduction Rule - Recap

Recall from the Crown Reduction Rule seminar that we have a crown consutruction algorithm for the vertex cover problem that can produce one of three different results.

- Determine if $vc(\mathcal{G}) > k$ (a no-instance).
- Check whether $|V| \leq 3k$.
- Construct a crown $\mathcal{C} = (I, N(I), F)$ on at least $|V| - 3k$ vertices.

The Crown Reduction Rule - Recap

Recall from the Crown Reduction Rule seminar that we have a crown consutruction algorithm for the vertex cover problem that can produce one of three different results.

- Determine if $vc(\mathcal{G}) > k$ (a no-instance).
- Check whether $|V| \leq 3k$.
- Construct a crown $\mathcal{C} = (I, N(I), F)$ on at least $|V| - 3k$ vertices.

Recall also that we had shown that $|N(I)| < |I|$. We will use this fact again later on.

Counting Vertex Covers Using The Crown Reduction Rule

Let $X_{\mathcal{C}}$ be a vertex cover of $\mathcal{C}$ and let $\mathcal{G}' := (V', E')$ be the graph obtained from $\mathcal{G}$ by removing the vertices and edges that are covered by $X_{\mathcal{C}}$. In particular, we have that $\#vc(\mathcal{G}', k - |X_{\mathcal{C}}|)$ is equal to the number of vertex covers X of size k in $\mathcal{G}$ with $X \supseteq X_{\mathcal{C}}$.

Counting Vertex Covers Using The Crown Reduction Rule

Let $X_{\mathcal{C}}$ be a vertex cover of $\mathcal{C}$ and let $\mathcal{G}' := (V', E')$ be the graph obtained from $\mathcal{G}$ by removing the vertices and edges that are covered by $X_{\mathcal{C}}$. In particular, we have that $\#vc(\mathcal{G}', k - |X_{\mathcal{C}}|)$ is equal to the number of vertex covers X of size k in $\mathcal{G}$ with $X \supseteq X_{\mathcal{C}}$.
Since $\mathcal{C}$ has a matching of cardinality $|N(I)|$, we can assume that $|N(I)| \leq k$, as we would automatically have an answer of 0 otherwise.

Counting Vertex Covers Using The Crown Reduction Rule

Let $X_{\mathcal{C}}$ be a vertex cover of $\mathcal{C}$ and let $\mathcal{G}' := (V', E')$ be the graph obtained from $\mathcal{G}$ by removing the vertices and edges that are covered by $X_{\mathcal{C}}$. In particular, we have that $\#vc(\mathcal{G}', k - |X_{\mathcal{C}}|)$ is equal to the number of vertex covers X of size k in $\mathcal{G}$ with $X \supseteq X_{\mathcal{C}}$.
Since $\mathcal{C}$ has a matching of cardinality $|N(I)|$, we can assume that $|N(I)| \leq k$, as we would automatically have an answer of 0 otherwise.
Since $|V'| \leq |\mathcal{G} \backslash \mathcal{C}| + |N(I)| \leq 4k$, we have that $\#vc(\mathcal{G}', k - |X_{\mathcal{C}}|)$ can be computed in time depending on k only. Hence it suffices to show how enumerate the vertex covers of $\mathcal{C}$.

Counting Vertex Covers Using The Crown Reduction Rule

Since $|N(I)| \leq k$, we can define an equivalency relation with at most 2^k equivalence classes as follows.

Counting Vertex Covers Using The Crown Reduction Rule

Since $|N(I)| \leq k$, we can define an equivalency relation with at most 2^k equivalence classes as follows.
For all $v, w \in I$, we define $v \sim w \iff N(v) = N(w)$. Moreover, we define, for every $v \in I$, $[v] := \{w \in I : v \sim w\}$.

Counting Vertex Covers Using The Crown Reduction Rule

Since $|N(I)| \leq k$, we can define an equivalency relation with at most 2^k equivalence classes as follows.
For all $v, w \in I$, we define $v \sim w \iff N(v) = N(w)$. Moreover, we define, for every $v \in I$, $[v] := \{w \in I : v \sim w\}$.

Lemma

Let $X_{\mathcal{C}}$ be a vertex cover of $\mathcal{C}$ such that there exists a vertex $y \in N(I) \backslash X_{\mathcal{C}}$. If $v \in I$ is a vertex with $y \in N(v)$, then $[v] \subseteq X_{\mathcal{C}}$.

Counting Vertex Covers Using The Crown Reduction Rule

Since $|N(I)| \leq k$, we can define an equivalency relation with at most 2^k equivalence classes as follows.
For all $v, w \in I$, we define $v \sim w \iff N(v) = N(w)$. Moreover, we define, for every $v \in I$, $[v] := \{w \in I : v \sim w\}$.

Lemma

Let $X_{\mathcal{C}}$ be a vertex cover of $\mathcal{C}$ such that there exists a vertex $y \in N(I) \backslash X_{\mathcal{C}}$. If $v \in I$ is a vertex with $y \in N(v)$, then $[v] \subseteq X_{\mathcal{C}}$.

Proof.

Assume that there is a vertex $w \in [v]$ such that $w \notin X_{\mathcal{C}}$. Then there exists an uncovered edge $\{y, w\}$ in $X_{\mathcal{C}}$, contradicting the fact that $X_{\mathcal{C}}$ is a vertex cover. □

Counting Vertex Covers Using The Crown Reduction Rule

Consider a vertex cover $X_{\mathcal{C}}$ of $\mathcal{C}$ and let $Y = N(I) \cap X_{\mathcal{C}}$ and $Z = I \cap X_{\mathcal{C}}$.

Counting Vertex Covers Using The Crown Reduction Rule

Consider a vertex cover $X_{\mathcal{C}}$ of $\mathcal{C}$ and let $Y = N(I) \cap X_{\mathcal{C}}$ and $Z = I \cap X_{\mathcal{C}}$. Let $\mathcal{B}(Y) := \{[v] : v \in I, \exists w \in N(I) \setminus Y \text{ such that } w \in N(v)\}$ and let

$$\mathcal{L}(Y) := \bigcup_{[v] \in \mathcal{B}(Y)} [v].$$

Counting Vertex Covers Using The Crown Reduction Rule

Consider a vertex cover $X_{\mathcal{C}}$ of $\mathcal{C}$ and let $Y = N(I) \cap X_{\mathcal{C}}$ and $Z = I \cap X_{\mathcal{C}}$. Let $\mathcal{B}(Y) := \{[v] : v \in I, \exists w \in N(I) \backslash Y \text{ such that } w \in N(v)\}$ and let

$$\mathcal{L}(Y) := \bigcup_{[v] \in \mathcal{B}(Y)} [v].$$

Note that the previous lemma implies that $\mathcal{L}(Y) \subseteq X_{\mathcal{C}}$, which also implies that $\mathcal{L}(Y) \subseteq Z$.

Counting Vertex Covers Using The Crown Reduction Rule

Consider a vertex cover $X_{\mathcal{C}}$ of $\mathcal{C}$ and let $Y = N(I) \cap X_{\mathcal{C}}$ and $Z = I \cap X_{\mathcal{C}}$.
Let $\mathcal{B}(Y) := \{[v] : v \in I, \exists w \in N(I) \backslash Y \text{ such that } w \in N(v)\}$ and let

$$\mathcal{L}(Y) := \bigcup_{[v] \in \mathcal{B}(Y)} [v].$$

Note that the previous lemma implies that $\mathcal{L}(Y) \subseteq X_{\mathcal{C}}$, which also implies that $\mathcal{L}(Y) \subseteq Z$.
Furthermore, we can observe that $Y \cup \mathcal{L}(Y)$ forms a minimal vertex cover, since a vertex is not in $\mathcal{L}(Y)$ if all of its neighbours are already in the vertex cover (hence all of the adjacent edges are already covered).

Counting Vertex Covers Using The Crown Reduction Rule

We now define our kernelization procedure $\Psi(\mathcal{G}, \mathcal{C}, k) := (\mathcal{G}', \mathcal{C}', k)$, where $\mathcal{C}' := (I', N(I'), F')$ is obtained from $\mathcal{C}$ by keeping one vertex from I per equivalency class and then deleting the rest - $\mathcal{G}' := (V', E')$ is obtained from $\mathcal{G}$ in the same manner.

Counting Vertex Covers Using The Crown Reduction Rule

We now define our kernelization procedure $\Psi(\mathcal{G}, \mathcal{C}, k) := (\mathcal{G}', \mathcal{C}', k)$, where $\mathcal{C}' := (I', N(I'), F')$ is obtained from $\mathcal{C}$ by keeping one vertex from I per equivalency class and then deleting the rest - $\mathcal{G}' := (V', E')$ is obtained from $\mathcal{G}$ in the same manner.
Hence we have that $\|\mathcal{C}'\| \leq 2^k + k$ and $\|\mathcal{G}'\| \leq 2^k + 4k$, and so Ψ is a k-bounded mapping computable in polynomial time.

Counting Vertex Covers Using The Crown Reduction Rule

We now define our kernelization procedure $\Psi(\mathcal{G}, \mathcal{C}, k) := (\mathcal{G}', \mathcal{C}', k)$, where $\mathcal{C}' := (I', N(I'), F')$ is obtained from $\mathcal{C}$ by keeping one vertex from I per equivalency class and then deleting the rest - $\mathcal{G}' := (V', E')$ is obtained from $\mathcal{G}$ in the same manner.
Hence we have that $\|\mathcal{C}'\| \leq 2^k + k$ and $\|\mathcal{G}'\| \leq 2^k + 4k$, and so Ψ is a k-bounded mapping computable in polynomial time.
We can define μ as follows. Let X' be a vertex cover of $\mathcal{G}'$ such that $|X'| \leq k$ and $X' \cap (I' \cup N(I'))$ is minimal with respect to the vertices in $\mathcal{C}'$. Furthermore, we let $Y' := N(I') \cap X'$. Defining μ as follows gives us the desired result.

$$\mu((\mathcal{G}, \mathcal{C}, k), (\mathcal{G}', \mathcal{C}', k), X') := \binom{|I \backslash \mathcal{L}(Y')|}{k - |Y'| - |\mathcal{L}(Y')|}$$

1. Counting Kernelizations - Definitions and Concepts

2. Counting Kernelization Examples
   - The Vertex Cover Problem
   - The Hitting Set Problem
   - The Unique Hitting Set Problem

3. Conclusions and Future Work

Hitting Set - Problem Definition

Before discussing the hitting set problem on hypergraphs with edges of bounded cardinality, we require several preliminary definitions.

Hitting Set - Problem Definition

Before discussing the hitting set problem on hypergraphs with edges of bounded cardinality, we require several preliminary definitions.

Definition

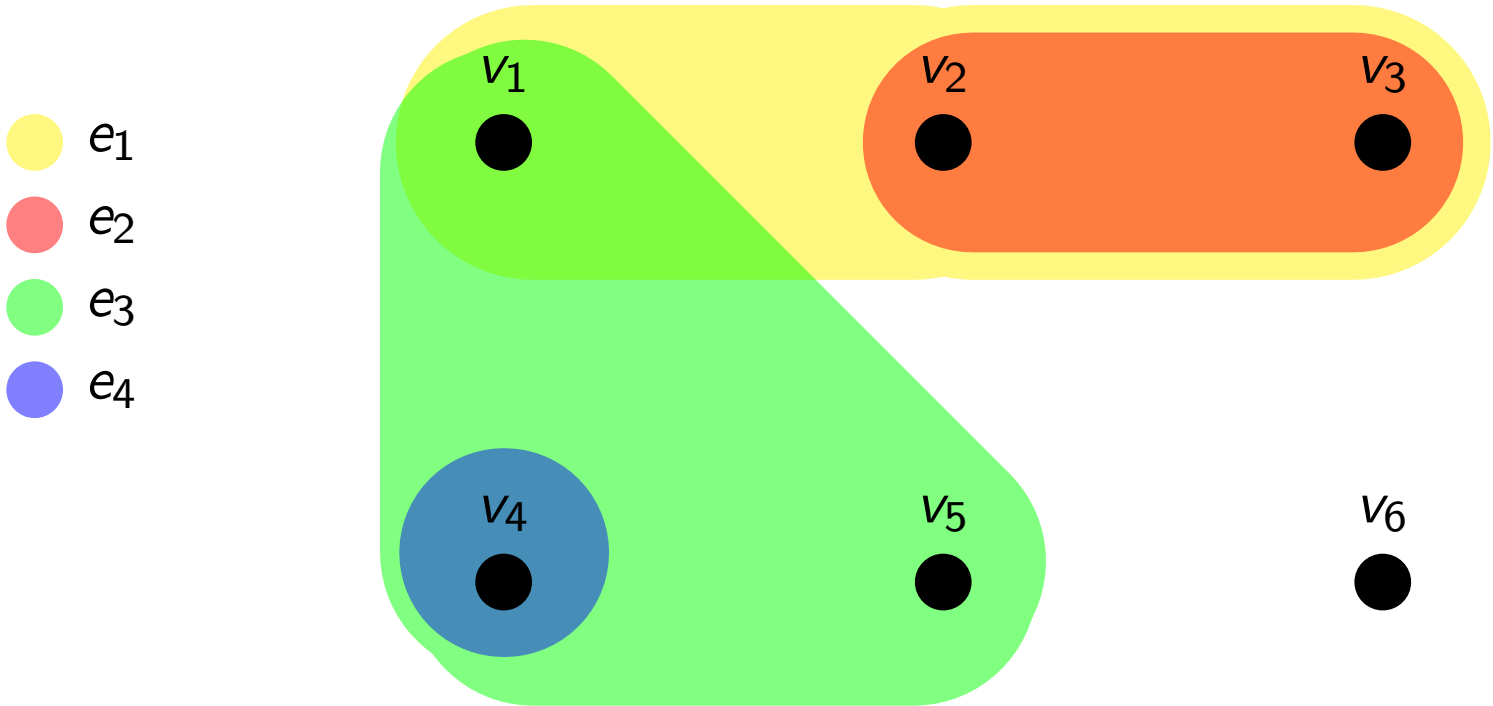
A hypergraph $\mathcal{H}$ is a pair $\mathcal{H} := (V, E)$, where V is a set of vertices and $E \subseteq \mathcal{P}(V) \backslash \{\emptyset\}$ is a set of hyperedges.

Hitting Set - Problem Definition

Before discussing the hitting set problem on hypergraphs with edges of bounded cardinality, we require several preliminary definitions.

Definition

A hypergraph $\mathcal{H}$ is a pair $\mathcal{H} := (V, E)$, where V is a set of vertices and $E \subseteq \mathcal{P}(V) \backslash \{\emptyset\}$ is a set of hyperedges.



Hitting Set - Problem Definition

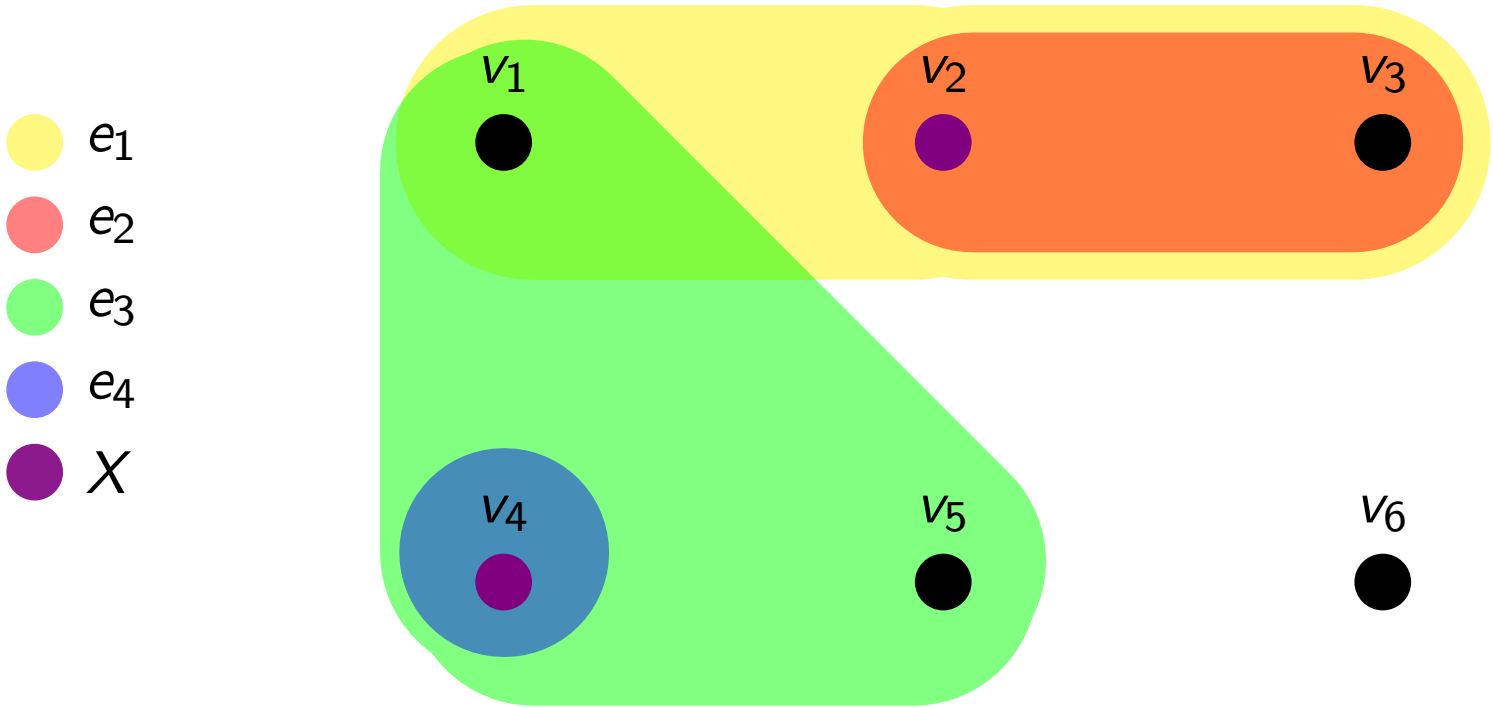
Definition

Given a hypergraph $\mathcal{H} := (V, E)$, a hitting set in $\mathcal{H}$ is a set $X \subseteq V$ such that for all $e \in E$, $X \cap e \neq \emptyset$.

Hitting Set - Problem Definition

Definition

Given a hypergraph $\mathcal{H} := (V, E)$, a hitting set in $\mathcal{H}$ is a set $X \subseteq V$ such that for all $e \in E$, $X \cap e \neq \emptyset$.



Hitting Set - Problem Definition

We can now define the hitting set problem on hypergraphs with edges of of bounded cardinality as follows.

Hitting Set - Problem Definition

We can now define the hitting set problem on hypergraphs with edges of of bounded cardinality as follows.

Problem

p-Card-#Hittingset

Input: A hypergraph $\mathcal{H} := (V, E)$ *and* $k \in \mathbb{N}$

Parameter: $k + d$ *where* $d := \max_{e \in E} |e|$

Question: Compute the number of hitting sets of cardinality k *in* $\mathcal{H}$

Hitting Set - Problem Definition

We can now define the hitting set problem on hypergraphs with edges of of bounded cardinality as follows.

Problem

p-Card-#Hittingset

Input: A hypergraph $\mathcal{H} := (V, E)$ *and* $k \in \mathbb{N}$

Parameter: $k + d$ *where* $d := \max_{e \in E} |e|$

Question: Compute the number of hitting sets of cardinality k *in* $\mathcal{H}$

We will also use the notation $\#hs(\mathcal{H}, k)$ to denote the number of hitting sets of size k in the hypergraph $\mathcal{H}$.

Hitting Set - Data Reduction Rule

Definition

Given a hypergraph $\mathcal{H} := (V, E)$, a sunflower in $\mathcal{H}$ is a family of hyperedges $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq E$ such that there exists a set $\mathcal{C} \subseteq V$ satisfying

$$S_i \cap S_j = \mathcal{C} \qquad \text{for all } 1 \leq i < j \leq k$$

We call $\mathcal{C}$ the core of the sunflower $\mathcal{S}$, while $S_i \backslash \mathcal{C}$ is called a petal of $\mathcal{S}$ if it is nonempty.

Hitting Set - Data Reduction Rule

Definition

Given a hypergraph $\mathcal{H} := (V, E)$, a sunflower in $\mathcal{H}$ is a family of hyperedges $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq E$ such that there exists a set $\mathcal{C} \subseteq V$ satisfying

$$S_i \cap S_j = \mathcal{C} \qquad \text{for all } 1 \leq i < j \leq k$$

We call $\mathcal{C}$ the core of the sunflower $\mathcal{S}$, while $S_i \backslash \mathcal{C}$ is called a petal of $\mathcal{S}$ if it is nonempty.

The following lemma is known as the Sunflower Lemma. It will be the primary tool used in the kernelization process of our problem.

Hitting Set - Data Reduction Rule

Definition

Given a hypergraph $\mathcal{H} := (V, E)$, a sunflower in $\mathcal{H}$ is a family of hyperedges $\mathcal{S} = \{S_1, \ldots, S_k\} \subseteq E$ such that there exists a set $\mathcal{C} \subseteq V$ satisfying

$$S_i \cap S_j = \mathcal{C} \qquad \text{for all } 1 \leq i < j \leq k$$

We call $\mathcal{C}$ the core of the sunflower $\mathcal{S}$, while $S_i \backslash \mathcal{C}$ is called a petal of $\mathcal{S}$ if it is nonempty.

The following lemma is known as the Sunflower Lemma. It will be the primary tool used in the kernelization process of our problem.

Lemma

Let $\mathcal{H} := (V, E)$ be a d-uniform hypergraph with more than $(k-1)^d \cdot d!$ hyperedges, for some $k, d \in \mathbb{N}$. Then there exists a sunflower $\mathcal{S}$ of cardinality k in $\mathcal{H}$.

Hitting Set - Data Reduction Rule

The proof of the Sunflower Lemma provides us with a recursive framework for finding a sunflower. In particular, we have the following lemma.

Hitting Set - Data Reduction Rule

The proof of the Sunflower Lemma provides us with a recursive framework for finding a sunflower. In particular, we have the following lemma.

Lemma

There is an algorithm $\mathrm{FindSunflower}$ *that computes a sunflower* $\mathcal{S}$ *according to the Sunflower Lemma in time* $\mathcal{O}(d\|\mathcal{H}\|)$.

Hitting Set - Data Reduction Rule

The proof of the Sunflower Lemma provides us with a recursive framework for finding a sunflower. In particular, we have the following lemma.

Lemma

There is an algorithm $\mathrm{FindSunflower}$ *that computes a sunflower* $\mathcal{S}$ *according to the Sunflower Lemma in time* $\mathcal{O}(d\|\mathcal{H}\|)$.

Furthermore, the Sunflower Lemma can be used for counting hitting sets in the following manner.

Hitting Set - Data Reduction Rule

The proof of the Sunflower Lemma provides us with a recursive framework for finding a sunflower. In particular, we have the following lemma.

Lemma

There is an algorithm $\mathrm{FindSunflower}$ *that computes a sunflower* $\mathcal{S}$ *according to the Sunflower Lemma in time* $\mathcal{O}(d\|\mathcal{H}\|)$.

Furthermore, the Sunflower Lemma can be used for counting hitting sets in the following manner.

Lemma

Let $\mathcal{S}$ *be a sunflower in* $\mathcal{H}$ *with* $k+1$ *petals and core* $\mathcal{C}$. *Also, let* $\mathcal{H}' := (V, E')$ *be a hypergraph with* $E' := E \backslash \mathcal{S} \cup \{\mathcal{C}\}$. *Then we have that* $\#hs(\mathcal{H}, k) = \#hs(\mathcal{H}', k)$.

Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

- Given a sunflower $\mathcal{S}$ with $k+1$ petals, every hitting set of size k in $\mathcal{H}$ must have a non-empty intersection with the core of the sunflower. Therefore, every hitting set of size k in $\mathcal{H}$ is a hitting set of size k in $\mathcal{H}'$ as well.

Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

- Given a sunflower $\mathcal{S}$ with $k+1$ petals, every hitting set of size k in $\mathcal{H}$ must have a non-empty intersection with the core of the sunflower. Therefore, every hitting set of size k in $\mathcal{H}$ is a hitting set of size k in $\mathcal{H}'$ as well.
- Every hitting set of size k in $\mathcal{H}'$ has a nonempty intersection with the core of $\mathcal{S}$, since the core of $\mathcal{S}$ is a hyperedge in $\mathcal{H}'$. Therefore, every hitting set of size k in $\mathcal{H}'$ is also a hitting set of size k in $\mathcal{H}$.

Hitting Set - Kernelization Procedure

Input: $\mathcal{H} := (V, E)$ and $k \in \mathbb{N}$
Output: Kernelized instance of $\mathcal{H}$
$d \leftarrow \max_{e \in E} |e|$;
for $i \leftarrow 1$ **to** d **do**
  $E_i \leftarrow \{e \in E : |e| = i\}$;
  $V_i \leftarrow \bigcup_{e \in E_i} e$;
  while $\mathrm{FindSunflower}((V_i, E_i), k+1) \neq (\emptyset, \emptyset)$ **do**
    Let $(\mathcal{S}, \mathcal{C})$ be the sunflower returned by $\mathrm{FindSunflower}$;
    $E_i \leftarrow (E_i \backslash \mathcal{S}) \cup \{\mathcal{C}\}$;
    $V_i \leftarrow (V_i \backslash \bigcup_{X \in \mathcal{S}} X) \cup \mathcal{C}$;
  end
end
$V' \leftarrow V_1 \cup \ldots \cup V_d$;
$E' \leftarrow E_1 \cup \ldots \cup E_d$;
return (V', E')

Hitting Set - Putting It All Together

Lemma

Let $\mathcal{H}$ be a hypergraph, let $k \in \mathbb{N}$, and let $\mathcal{H}' := (V', E')$ be the kernelized instance of $\mathcal{H}$. If $I = V \backslash V'$, then

$$\#hs(\mathcal{H}, k) = \sum_{j=0}^{k} \#hs(\mathcal{H}', j) \cdot \binom{|I|}{k-j}.$$

Hitting Set - Putting It All Together

Lemma

Let $\mathcal{H}$ be a hypergraph, let $k \in \mathbb{N}$, and let $\mathcal{H}' := (V', E')$ be the kernelized instance of $\mathcal{H}$. If $I = V \backslash V'$, then

$$\#hs(\mathcal{H}, k) = \sum_{j=0}^{k} \#hs(\mathcal{H}', j) \cdot \binom{|I|}{k-j}.$$

- For any hitting set X of size k in $\mathcal{H}$, we have that $|X \cap V'| = j$ and $|X \cap I| = k - j$, for some suitable value of $j \leq k$.

Hitting Set - Putting It All Together

Lemma

Let $\mathcal{H}$ be a hypergraph, let $k \in \mathbb{N}$, and let $\mathcal{H}' := (V', E')$ be the kernelized instance of $\mathcal{H}$. If $I = V \backslash V'$, then

$$\#hs(\mathcal{H}, k) = \sum_{j=0}^{k} \#hs(\mathcal{H}', j) \cdot \binom{|I|}{k-j}.$$

- For any hitting set X of size k in $\mathcal{H}$, we have that $|X \cap V'| = j$ and $|X \cap I| = k - j$, for some suitable value of $j \leq k$.
- For any hitting set X' of size j in $\mathcal{H}'$ and for every set $I' \subseteq I$ of size $k - j$, the union $X' \cup I'$ is a hitting set of size k in $\mathcal{H}$.

Hitting Set - Putting It All Together

It can be shown that the kernelization procedure takes $\mathcal{O}(d^2 \cdot |E| \cdot \|\mathcal{H}\|)$ time. Moreover, for the hypergraph $\mathcal{H}' := (V', E')$, the Sunflower Lemma implies that $|E'| \leq k^d \cdot d! \cdot d$ and that $|V'| \leq k^d \cdot d! \cdot d^2$.

Hitting Set - Putting It All Together

It can be shown that the kernelization procedure takes $\mathcal{O}(d^2 \cdot |E| \cdot \|\mathcal{H}\|)$ time. Moreover, for the hypergraph $\mathcal{H}' := (V', E')$, the Sunflower Lemma implies that $|E'| \leq k^d \cdot d! \cdot d$ and that $|V'| \leq k^d \cdot d! \cdot d^2$.

- In particular, this implies a kernel of size $\|\mathcal{H}'\| \leq 2 \cdot k^d \cdot d! \cdot d^2$.

Hitting Set - Putting It All Together

It can be shown that the kernelization procedure takes $\mathcal{O}(d^2 \cdot |E| \cdot \|\mathcal{H}\|)$ time. Moreover, for the hypergraph $\mathcal{H}' := (V', E')$, the Sunflower Lemma implies that $|E'| \leq k^d \cdot d! \cdot d$ and that $|V'| \leq k^d \cdot d! \cdot d^2$.

- In particular, this implies a kernel of size $\|\mathcal{H}'\| \leq 2 \cdot k^d \cdot d! \cdot d^2$.

Following our framework for counting kernelizations, we can define functions $\Psi(\mathcal{H}) := \mathcal{H}'$ and $\mu(\mathcal{H}, \mathcal{H}', X) := \binom{|I|}{k-|X|}$ that are computable in polynomial time.

Unique Hitting Set - Problem Definition

The last example we will consider is the unique hitting set problem, which can be defined as follows.

Unique Hitting Set - Problem Definition

The last example we will consider is the unique hitting set problem, which can be defined as follows.

Problem

p-#UniqueHittingset

Input: A hypergraph $\mathcal{H} := (V, E)$

Parameter: $k := |E|$

Question: Compute the number of unique hitting sets in $\mathcal{H}$*; that is all sets* $X \subseteq V$ *satisfying the following condition.*

- *For each* $e \in E$, $|e \cap X| = 1$

Unique Hitting Set - Problem Definition

The last example we will consider is the unique hitting set problem, which can be defined as follows.

Problem

p-#UniqueHittingset

Input: A hypergraph $\mathcal{H} := (V, E)$

Parameter: $k := |E|$

Question: Compute the number of unique hitting sets in $\mathcal{H}$*; that is all sets* $X \subseteq V$ *satisfying the following condition.*

- *For each* $e \in E$, $|e \cap X| = 1$

We will develop the kernelization technique and reduction rule in the upcoming slides.

Unique Hitting Set - Kernelization

Let $\mathcal{H} := (V, E)$ be a hypergraph and let $k := |E|$. For all $x, y \in V$ we will define an equivalency relation by $x \sim y =_{def} \forall e \in E : x \in e \iff y \in e$.

Unique Hitting Set - Kernelization

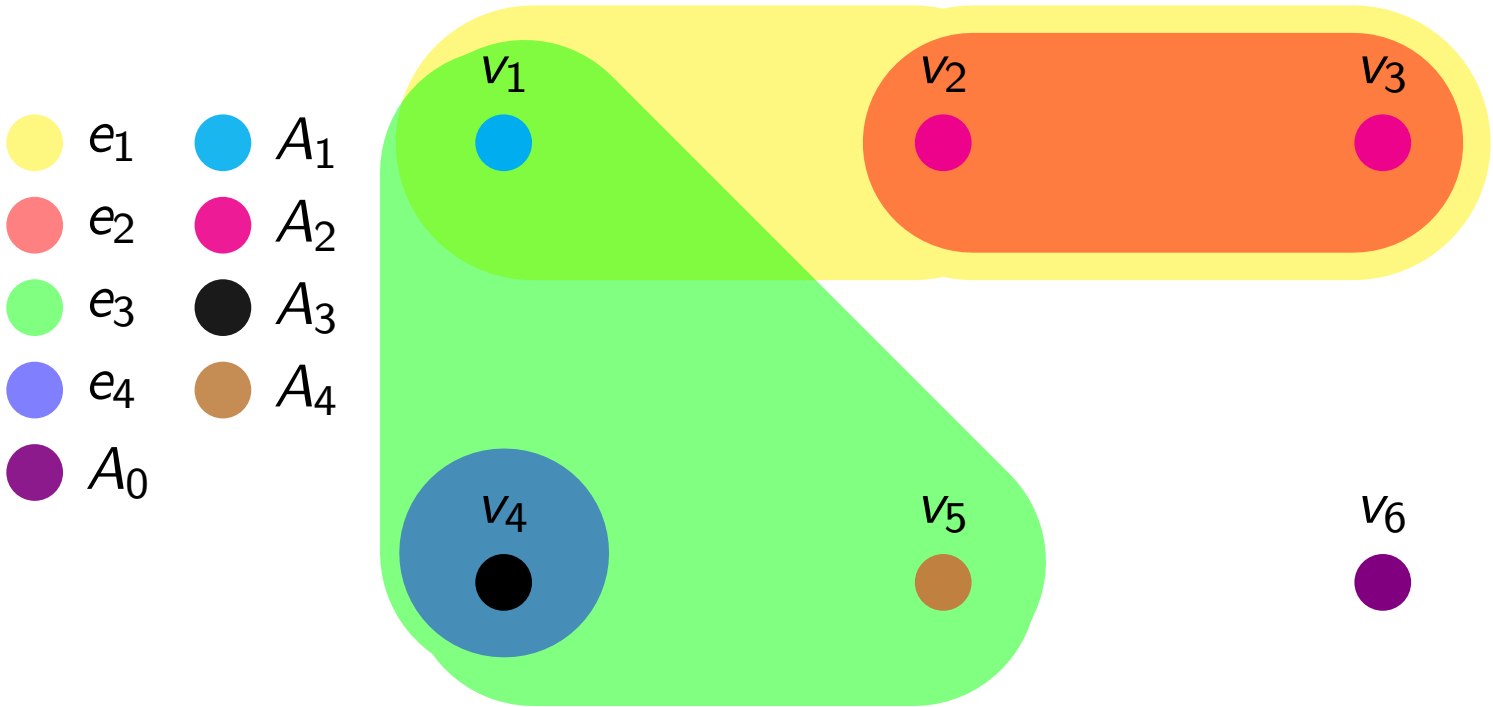
Let $\mathcal{H} := (V, E)$ be a hypergraph and let $k := |E|$. For all $x, y \in V$ we will define an equivalency relation by $x \sim y =_{def} \forall e \in E : x \in e \iff y \in e$. This relation defines L nonempty equivalency classes, for some $L \leq 2^k$.

Unique Hitting Set - Kernelization

Let $\mathcal{H} := (V, E)$ be a hypergraph and let $k := |E|$. For all $x, y \in V$ we will define an equivalency relation by $x \sim y =_{def} \forall e \in E : x \in e \iff y \in e$. This relation defines L nonempty equivalency classes, for some $L \leq 2^k$. Let A_0 to be the equivalency class for all isolated vertices and let $\mathcal{A}' := \{A_1, \ldots, A_{L-1}\}$ be the family of all nonempty equivalency classes besides for A_0.

Unique Hitting Set - Kernelization

Let $\mathcal{H} := (V, E)$ be a hypergraph and let $k := |E|$. For all $x, y \in V$ we will define an equivalency relation by $x \sim y =_{def} \forall e \in E : x \in e \iff y \in e$. This relation defines L nonempty equivalency classes, for some $L \leq 2^k$. Let A_0 to be the equivalency class for all isolated vertices and let $\mathcal{A}' := \{A_1, \ldots, A_{L-1}\}$ be the family of all nonempty equivalency classes besides for A_0.



Unique Hitting Set - Kernelization

We will now define the two mappings that will be used in the construction of our kernel.

Unique Hitting Set - Kernelization

We will now define the two mappings that will be used in the construction of our kernel.

1. Let $g : V \to \mathcal{A}'$ be a function that maps a vertex in $\mathcal{H}$ to its corresponding equivalency class. In particular, we have that for every $u \in V$, $g(u) := A \in \mathcal{A}'$ such that $u \in A$.

Unique Hitting Set - Kernelization

We will now define the two mappings that will be used in the construction of our kernel.

1. Let $g : V \to \mathcal{A}'$ be a function that maps a vertex in $\mathcal{H}$ to its corresponding equivalency class. In particular, we have that for every $u \in V$, $g(u) := A \in \mathcal{A}'$ such that $u \in A$.
2. Let $f : 2^V \to 2^{\mathcal{A}'}$ be an analogous function defined for sets of vertices, where $f(\{u_1, \ldots, u_b\}) := \{g(u_1), \ldots, g(u_b)\}$.

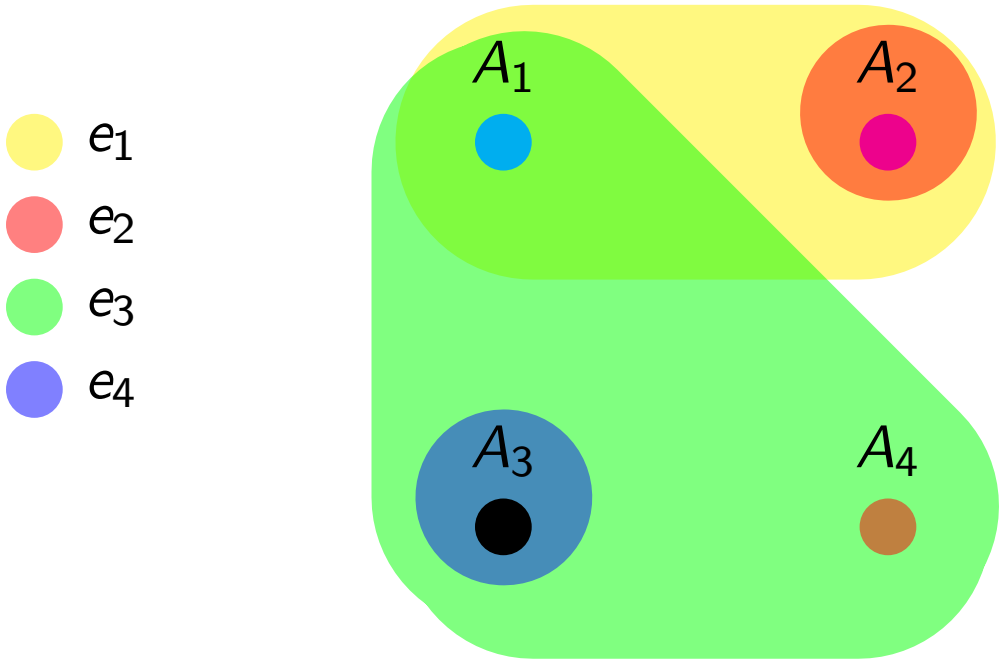
Unique Hitting Set - Kernelization

We will now define the two mappings that will be used in the construction of our kernel.

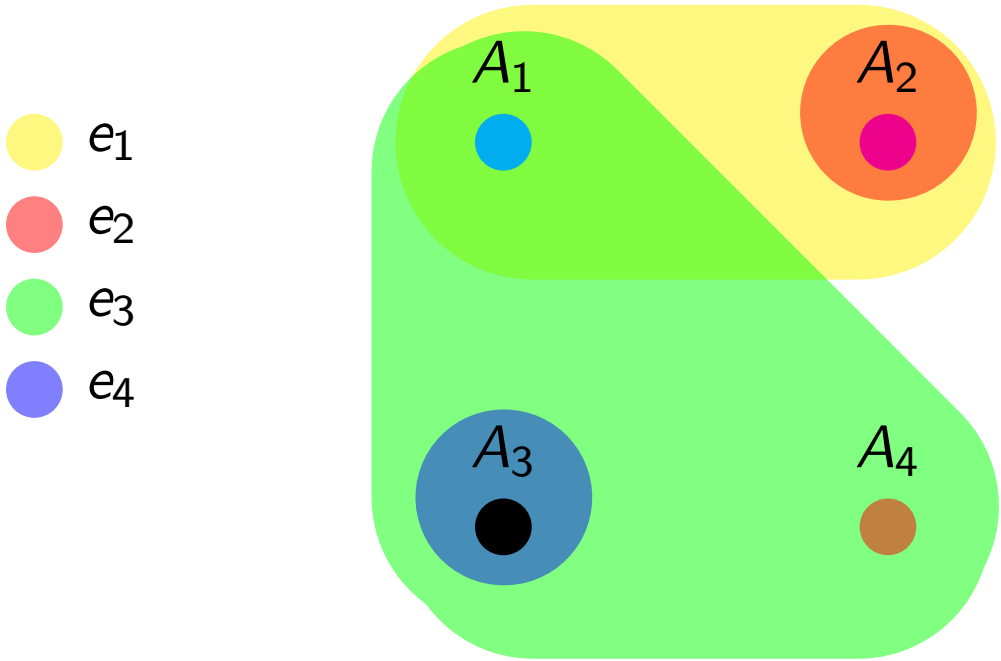
1. Let $g : V \to \mathcal{A}'$ be a function that maps a vertex in $\mathcal{H}$ to its corresponding equivalency class. In particular, we have that for every $u \in V$, $g(u) := A \in \mathcal{A}'$ such that $u \in A$.
2. Let $f : 2^V \to 2^{\mathcal{A}'}$ be an analogous function defined for sets of vertices, where $f(\{u_1, \ldots, u_b\}) := \{g(u_1), \ldots, g(u_b)\}$.

We now claim that the hypergraph $\mathcal{H}' := (\mathcal{A}', \mathcal{E}')$ will be the kernel in our algorithm, where $\mathcal{E}' := \{f(e) : e \in E\}$.

Unique Hitting Set - Kernelization



Unique Hitting Set - Kernelization



Lemma

Let $\mathcal{H}$, $\mathcal{H}'$ and $\mathcal{A}'$ be defined as above, and let $\mathcal{U}'$ be the set of all unique hitting sets in $\mathcal{H}'$. Then the number $\#uhs(\mathcal{H})$ of unique hitting sets in $\mathcal{H}$ satisfies:

$$\#uhs(\mathcal{H}) = \sum_{S' \in \mathcal{U}'} 2^{|A_0|} \prod_{A \in S'} |A|.$$

Unique Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

Unique Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

- If S' is a unique hitting set of $\mathcal{H}'$, then we can construct a unique hitting set for $\mathcal{H} \backslash A_0$ by selecting any single vertex from each equivalency class $A \in S'$, since each vertex in A covers the same set of hyperedges.

Unique Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

- If S' is a unique hitting set of $\mathcal{H}'$, then we can construct a unique hitting set for $\mathcal{H} \backslash A_0$ by selecting any single vertex from each equivalency class $A \in S'$, since each vertex in A covers the same set of hyperedges.
- For any unique hitting set S of $\mathcal{H} \backslash A_0$ and for any subset of vertices $X \subseteq A_0$, both S and $S \cup X$ are unique hitting sets of $\mathcal{H}$, since X is a set of isolated vertices.

Unique Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

- If S' is a unique hitting set of $\mathcal{H}'$, then we can construct a unique hitting set for $\mathcal{H} \backslash A_0$ by selecting any single vertex from each equivalency class $A \in S'$, since each vertex in A covers the same set of hyperedges.
- For any unique hitting set S of $\mathcal{H} \backslash A_0$ and for any subset of vertices $X \subseteq A_0$, both S and $S \cup X$ are unique hitting sets of $\mathcal{H}$, since X is a set of isolated vertices.
- Following our framework of counting kernelizations, we can define $\Psi(\mathcal{H}) := \mathcal{H}'$ and $\mu(\mathcal{H}, \mathcal{H}', S') := 2^{|A_0|} \prod_{A \in S'} |A|$ for any unique hitting set S' of $\mathcal{H}'$.

Unique Hitting Set - A Few Comments

We would like to make a few comments regarding the previous lemma.

- If S' is a unique hitting set of $\mathcal{H}'$, then we can construct a unique hitting set for $\mathcal{H} \backslash A_0$ by selecting any single vertex from each equivalency class $A \in S'$, since each vertex in A covers the same set of hyperedges.
- For any unique hitting set S of $\mathcal{H} \backslash A_0$ and for any subset of vertices $X \subseteq A_0$, both S and $S \cup X$ are unique hitting sets of $\mathcal{H}$, since X is a set of isolated vertices.
- Following our framework of counting kernelizations, we can define $\Psi(\mathcal{H}) := \mathcal{H}'$ and $\mu(\mathcal{H}, \mathcal{H}', S') := 2^{|A_0|} \prod_{A \in S'} |A|$ for any unique hitting set S' of $\mathcal{H}'$.
- We have that $|\mathcal{E}'| = |E| = k$ and $|\mathcal{A}'| \leq 2^k$, and so the size of $\mathcal{H}'$ is a function of k. Furthermore, we have that $\mathcal{H}'$ can be constructed in polynomial time, while μ can be computed in polynomial time.

1. Counting Kernelizations - Definitions and Concepts

2. Counting Kernelization Examples
   - The Vertex Cover Problem
   - The Hitting Set Problem
   - The Unique Hitting Set Problem

3. Conclusions and Future Work

Concluding Remarks

We have developed a formal framework for the kernelization of counting problems, and have applied this notion to develop parameterized algorithms for three different counting problems on graphs.

Concluding Remarks

We have developed a formal framework for the kernelization of counting problems, and have applied this notion to develop parameterized algorithms for three different counting problems on graphs.
We have also related the concept of counting kernelization to the class of fixed parameter tractable counting problems ($FFPT$).

Concluding Remarks

We have developed a formal framework for the kernelization of counting problems, and have applied this notion to develop parameterized algorithms for three different counting problems on graphs.
We have also related the concept of counting kernelization to the class of fixed parameter tractable counting problems ($FFPT$).
Lastly, we have shown that several well known kernelization techniques for decision problems can be adapted to solve their counting counterpart, such as the Crown Rule Decomposition and the Sunflower Kernel, all of which provide a solid structural framework for the development of other parameterized algorithms.

Future Work

In terms of future work, it could be worth investigating if other well known kernelization techniques can be adapted to counting problems as well, such as the Buss Kernelization for vertex cover.

Future Work

In terms of future work, it could be worth investigating if other well known kernelization techniques can be adapted to counting problems as well, such as the Buss Kernelization for vertex cover.
How can kernelization be combined with other techniques seen in class to solve counting problems? For example, does it make sense to use color-coding or branching with kernelization to help obtain even smaller kernels?

Future Work

In terms of future work, it could be worth investigating if other well known kernelization techniques can be adapted to counting problems as well, such as the Buss Kernelization for vertex cover.
How can kernelization be combined with other techniques seen in class to solve counting problems? For example, does it make sense to use color-coding or branching with kernelization to help obtain even smaller kernels?
There are still many problems whose counting counterparts are not known to be in $FFPT$ or in $\#W[1]$-hard.

- p-DisjointTriangles
- p-(n-k)-Coloring

References

Marc Thurley (2007)

Kernelizations for Parameterized Counting Problems

TAMC, volume 4484 of Lecture Notes in Computer Science, 703 – 714.

Thank You!