Parameterized Algorithms via Set Systems, Polynomials etc.

Some slides from Dr. Pranabendu Misra

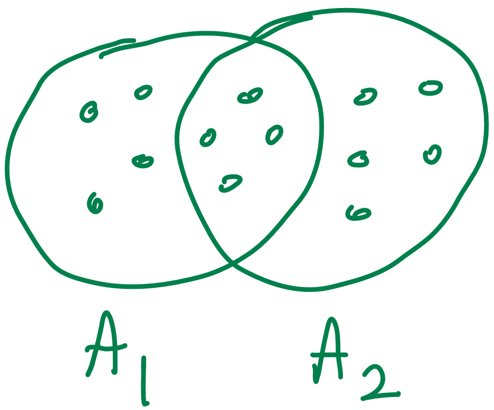
Inclusion-Exclusion Principle

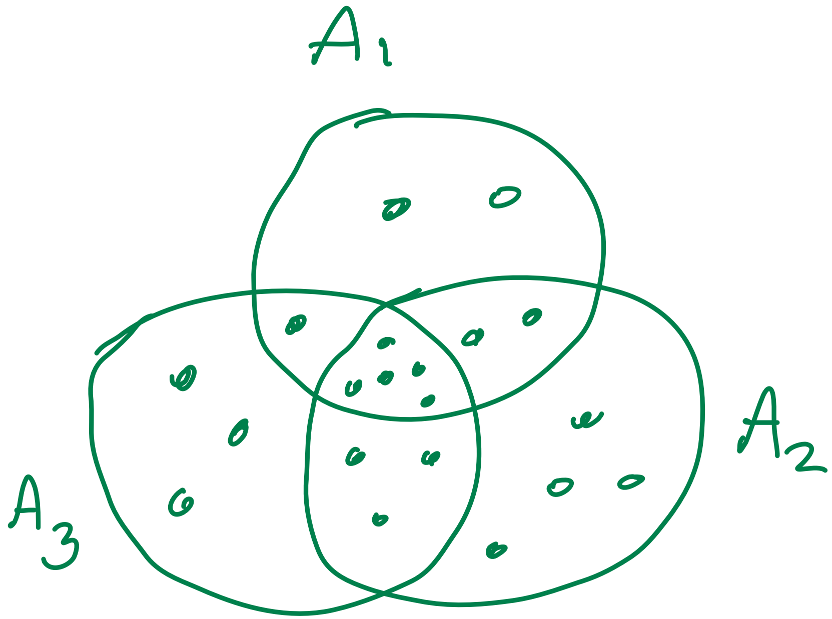
Inclusion-Exclusion

Theorem

Let $A_1, A_2, \ldots, A_k$ *be subsets of a universe* U*, and let* $B_i = U \setminus A_i$*. Then*

$$|\bigcap_{i\in[k]} A_i| = \sum_{X\subseteq[k]} (-1)^{|X|} |\cap_{j\in X} B_j|$$





Unweighted Steiner Tree

Unweighted STEINER TREE: Given a graph G in n vertices, a subset K of k terminals, find a subgraph(tree) on at most ℓ edges that connects all the terminals.[1]

Theorem

Unweighted STEINER TREE *can be solved in* $2^k \cdot poly(n)$ *time.*

Using Inclusion-Exclusion

[1] $\ell \geq k$, and we can always guess the smallest value of ℓ for which a Steiner Tree exists.

Unweighted Steiner Tree

Intuition:

- We solve the Counting Problem.

  If the number of ℓ-edge subtrees of G containing K is non-zero, then a Steiner Tree on ℓ edges exists.

- Counting trees is hard, so we count an easier object called Branching Walks.
- We count Branching Walks via Inclusion-Exclusion.

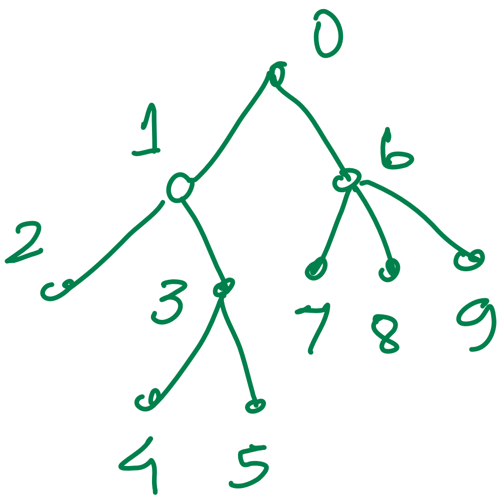
Unweighted Steiner Tree

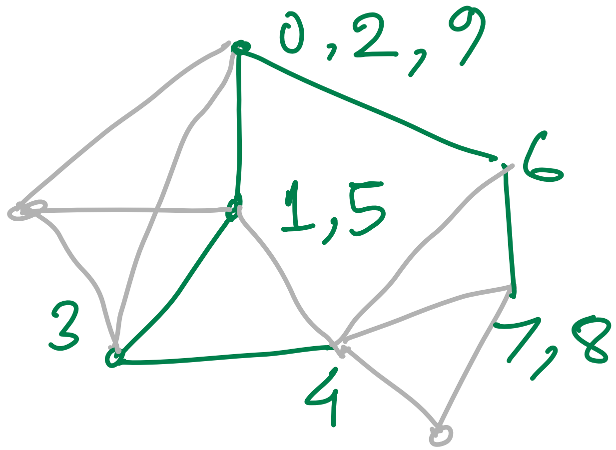
Ordered Rooted Tree : A tree H where vertices have been labeled by $\{0, 2, 3 \ldots, |V(H)| - 1\}$ via a DFS. Alternatively, every internal node of H has an ordering among it's children.

Let $r \in V(H)$ denote the <u>root</u> of H.

Branching Walk : A Homomorphic Image of an ordered rooted tree in G. It is a pair $B = (H, h)$ where H is an ordered rooted tree, and $h : V(H) \to V(G)$ is a map such that if $(x, y) \in E(H)$ then $(h(x), h(y) \in E(G)$.

Let $V(B) = \{h(x) \mid x \in V(H)\}$, and $s = h(r)$ be the <u>root</u> of B.





Unweighted Steiner Tree

Ordered Rooted Tree : A tree H where vertices have been labeled by $\{0, 2, 3 \ldots, |V(H)| - 1\}$ via a DFS. Alternatively, every internal node of H has an ordering among it's children.

Let $r \in V(H)$ denote the <u>root</u> of H.

Branching Walk : A Homomorphic Image of an ordered rooted tree in G. It is a pair $B = (H, h)$ where H is an ordered rooted tree, and $h : V(H) \to V(G)$ is a map such that if $(x, y) \in E(H)$ then $(h(x), h(y)) \in E(G)$.

Let $V(B) = \{h(x) \mid x \in V(H)\}$, and $s = h(r)$ be the <u>root</u> of B.

Lemma

Fix a terminal $s \in K$ as the <u>root</u>. G contains a Steiner Tree on $\leq \ell$ edges if and only if there is a Branching Walk $B = (H, h)$ from s such that $K \subseteq V(B)$, and $|E(H)| = \ell$.

Call $\ell = |E(H)|$ the <u>length</u> of the Branching Walk.

Unweighted Steiner Tree

Counting Branching Walks from s.

- Universe U = all Branching walks of length ℓ from s
- For each $v \in K$, $A_v = \{B \in U \mid v \in V(B)\}$
- Clearly $|\bigcap_{v \in K} A_v| \neq 0$ if and only if there is a Steiner Tree.
- Sufficient: Given $X \subseteq K$ compute $|\bigcap_{v \in X} B_v|$ where $B_v = U \setminus A_v$.

  $\bigcap_{v \in X} B_v$ is the set of all Branching Walks that avoid X

- They lie in the graph $G - X$,

  so enough to count all Branching Walks from s in the graph $G - X$ of length ℓ.

Lemma

$|\bigcap_{v \in X} B_v|$ can be computed in polynomial time.

Unweighted Steiner Tree

Computing $|\bigcap_{v \in X} B_v|$:

- Let $G' = G - X$. It contains all Branching Walks avoiding X.
- For $u \in V(G')$ and $j \leq \ell$ let $b_j(u)$ denote the number of Branching Walks from u of length j in G'.
- We want the value $b_\ell(s) = |\bigcap_{v \in X} B_v|$, assuming that $s \in V(G')$.
- Dynamic Programming:

$$b_j(u) = \begin{cases} 1 & \text{if } j = 0 \\ \sum_{w \in N_{G'}(a)} \sum_{j_1 + j_2 = j-1} \quad b_{j_1}(u) b_{j_2}(w) & \text{otherwise} \end{cases}$$

*Importance of the ordering of the leaves in the branching walk definition

Unweighted Steiner Tree

Counting Steiner Trees

- Once we have the numbers $|\bigcap_{v\in X} B_v|$ for every $X \subseteq K$, we can compute the number of Steiner Trees via the Inclusion-Exclusion formula

$$|\bigcap_{v\in K} A_v| = \sum_{X\subseteq K} (-1)^{|X|} |\cap_{u\in X} B_u|$$

- Running Time: $2^k \cdot poly(n)$.

This approach can be applied to many other problems such as Hamiltonian Path, Chromatic Number etc.

Chromatic Number

A k-colouring of a graph G is a function $c : V(G) \to [k]$, such that $c(u) \neq c(v)$ if $uv \in E(G)$.

CHROMATIC NUMBER **Input:** A graph G and an integer k **Question:** Is there a k-colouring of G?

$O^*(2^n)$ time algorithm by applying Inclusion-Exclusion.

Properties of k-colourings

Given a k-colouring:

- Each colour class must be an independent set.
- Every subset of an independent set is also an independent set.
- G has a k-colouring if and only if there is a *cover* of $V(G)$ by k independent sets, i.e., there are k independent sets $I_1, \ldots, I_k$ such that $\bigcup_{j=1}^{k} I_j = V(G)$.

k-colouring and Counting

- Enough to find a cover of $V(G)$ by k independent sets.
- Enough to compute the number of covers of $V(G)$ by k independent sets.
- If the number is non-zero then G has a k-colouring.
- Try to design an Inclusion-Exclusion algorithm.

Setting up Inclusion-Exclusion

- Universe U: set of tuples $(I_1, \ldots, I_k)$ where each I_j is an independent set (need not be disjoint).
- For each $v \in V(G)$, define $A_v = \{(I_1, \ldots, I_k) \in U | v \in \bigcup_{j=1}^{k} I_j\}$.
- Number of covers of size k: $|\bigcap_{v \in V(G)} A_v|$.

Computing $|\bigcap_{v \in V(G)} A_v|$

- Need to compute $\Sigma_{X \subseteq V(G)} (-1)^{|X|} |\bigcap_{v \in X} B_v|$, where $B_v = U \setminus A_v$.
- Need to compute for each $X \subseteq V(G)$, $|\bigcap_{v \in X} B_v|$ $= |\{(I_1, \ldots, I_k) \in U | I_1, \ldots, I_k \subseteq V(G) \setminus X\}|$.
- for $Y \subseteq V(G)$, $s(Y)$ is the number of independent sets in $G[Y]$.
- $|\{(I_1, \ldots, I_k) \in U | I_1, \ldots, I_k \subseteq V(G) \setminus X\}| = s(V(G) \setminus X)^k$.

Computing $s(V(G) \setminus X)^k$, $X \subseteq V(G)$

- Compute $s(Y)$ for all $Y \subseteq V(G)$ through dynamic programming with an algorithm using $O^*(2^n)$ time and space.
- Recursion: $s(Y) = s(Y \setminus \{y\}) + s(Y \setminus N[y])$.
- Finally, $s(V(G) \setminus X)^k$ can be computed from $s(V(G) \setminus X)$ by $\log k$ multiplications of $O(nk)$-bit numbers.

Inclusion-Exclusion algorithm

- $O^*(2^n)$ time algorithm for Chromatic Number. Space complexity is also $O^*(2^n)$.
- Can be decrease space complexity even if time complexity goes up a little?
- If $s(Y)$ is computed recursively instead of storing values in a table, then for each Y time taken is $O^*(2^{|Y|})$, but space complexity becomes polynomial!
- Total time complexity for Chromatic Number using polynomial space: $(\Sigma_{X \subseteq V(G)} 2^{|V(G) \setminus X|}) n^{O(1)} = (\Sigma_{k=0}^{n} \binom{n}{k} 2^{n-k}) n^{O(1)} = O^*(3^n)$.

Note: Best time complexity for Chromatic Number using polynomial space $= 2.238^n n^{O(1)}$.

Multivariate Polynomials: FPT Algorithms

Multivariate Polynomials

- **Finite Field:** A tuple $(\mathbb{F}, +, \star)$ capturing arithmetic in a finite set. Subtraction, division well defined; Field axioms: Associativity, Commutativity, additive and multiplicative identity and inverse, Distributivity.
- **Characteristic 2:** For any $a \in \mathbb{F}, a + a = 0$.
  Note that $|\mathbb{F}| >> 2$ is possible.
- **Polynomials over $\mathbb{F}$:** coefficients $a_{\ldots} \in \mathbb{F}$

$$P(x_1, x_2, \ldots, x_n) = \sum_{(c_1, c_2, \ldots, c_n) \in (\mathbb{N} \cup \{0\})^n} a_{c_1, c_2, \ldots, c_n} x_1^{c_1} x_2^{c_2} \ldots x_n^{c_n}$$

  degree of $P = \max_{(c_1, c_2, \ldots, c_n) | a_{c_1, c_2, \ldots, c_n} \neq 0} \sum c_i$ where
- **Identically Zero Polynomial:** $P \equiv 0$ means $P(x_1 = b_1, x_2 = b_2, \ldots, x_n = b_n) = 0$ for all choices in $\mathbb{F}^n$

Lemma (Schwartz-Zippel)

Let P be a polynomial over a field $\mathbb{F}$ of degree d, and let $S \subseteq \mathbb{F}$. Pick $b_1, b_2, \ldots, b_n$ randomly from S. If $P \not\equiv 0$, then $P(b_1, b_2, \ldots, b_n) = 0$ with probability at most $d/|S|$.

k-Path

k-Path: Given a graph G and an integer k, decide if G contains a path of length k.

Theorem

There is a randomized FPT algorithm for k-Path running in time $2^k \cdot poly(n)$.

* Schwartz-Zippel lemma used as a subroutine

k-Path

k-PATH: Given a graph G and an integer k, decide if G contains a path of length k.

Theorem

There is a randomized FPT algorithm for k-PATH running in time $2^k \cdot poly(n)$.

Intuition

- Encode k-walks as monomials of a polynomial
- Ensure the walks "cancel out" (using characteristic 2), hence the polynomial encodes only k-paths
- The polynomial is non-zero means there is a k-path. Test using Schwartz-Zippel Lemma.

Path to Polynomials

- variables $x =< x_1, \ldots, x_m >$ for edges,
  $y =< y_1, \ldots, y_n >$ for vertices.
- Path polynomial (hard to eval)

$$P(x,y) = \sum_{k\text{-Path } R \in G} \Big(\prod_{(v_i, v_{i+1}) \in R} x_{v_i, v_{i+1}} \Big) \cdot \Big(\prod_{v_i \in R} y_{v_i} \Big)$$

- Walk polynomial (easy to eval, but not very useful)

$$P(x,y) = \sum_{k\text{-Walk } W \in G} \Big(\prod_{(v_i, v_{i+1}) \in W} x_{v_i, v_{i+1}} \Big) \cdot \Big(\prod_{v_i \in W} y_{v_i} \Big)$$

- **Labeled Walk Polynomial.**
  - vertex variable set $y = \{y_{v,i} \mid v \in V(G), i \in [k]\}$
  - For a bijective function $\ell : [k] \to [k]$ and a k-Walk W we have the monomial
    $mon(W, \ell) = (\prod_{(v_i, v_{i+1}) \in W} x_{v_i, v_{i+1}}) \cdot (\prod_{v_i \in W} y_{v_i, \ell(i)})$

$$P(x,y) = \sum_{\text{Walks } W} \sum_{\text{bijection } \ell} mon(W, \ell)$$

Path to polynomials

Lemma

Over a field of characteristic 2,

$$P(x,y) \equiv \sum_{Paths\ R} \sum_{bijection\ \ell} mon(R,\ell)$$

- Any k-Walk W corresponds to a number of labeled walks, one for each bijection $\ell : [k] \to [k]$.
- For a k-Path R, every bijection ℓ gives a distinct monomial.
- However for a walk W, for every bijection ℓ there is another bijection ℓ' that produces the same monomial, and they cancel out.
  - For a walk W where a vertex v repeats at pos a and b
  - Given $\ell : [k] \to [k]$ define

$$\ell'(i) = \begin{cases} \ell(b) & i = a \\ \ell(a) & i = b \\ \ell(i) & \text{otherwise} \end{cases}$$

Path to polynomials

Lemma

Over a field of characteristic 2*,*

$$P(x,y) \equiv \sum_{Paths\ R} \sum_{bijection\ \ell} mon(R,\ell)$$

Corollary

The polynomial $P(x,y)$ *is non-zero over fields of characteristic* 2 *if and only if* G *contains a* k*-path.*

- We test if $P \equiv 0$ using the Schwartz-Zippel Lemma
- We randomly pick an assignment of the variables from $\mathbb{F}$ and then evaluate P.
- Evaluating P will require an algorithm based on Inclusion-Exclusion.

Evaluating $P(x, y)$

Theorem (Weighted Inclusion Exclusion)

Let $A_1, A_2, \ldots, A_k$ *be subsets of a universe* U*, and let* $B_i = U \setminus A_i$*. Let* $w : U \to \mathbb{R}$ *be a weight function Then*

$$w(\bigcap_{i \in [k]} A_i) = \sum_{X \subseteq [k]} (-1)^{|X|} w(\cap_{j \in X} B_j)$$

Evaluating $P(x, y)$

Fix a walk W

- Universe U = all functions $[k] \to [k]$
- for $\ell \in U$, define $w(\ell) = mon(W, \ell)$
- For each $i \in [k]$, $A_i = \{\ell \in U \mid \ell^{-1}(i) \neq \emptyset\}$
- Then $w(\cap_{i\in[k]} A_i) = \sum_{\text{bijection } \ell} \; mon(W, \ell)$
- $w(\bigcap_{i\in[k]} A_i) = \sum_{X\subseteq[k]} w(\cap_{j\in X} B_j)$,
- and $\sum_{X\subseteq[k]} w(\cap_{j\in X} B_j) = \sum_{X\subseteq[k]} \sum_{\ell:[k]\to[k]\setminus X} mon(W, \ell)$,

Therefore,

$$\begin{aligned} P(x, y) &= \sum_{\text{Walks } W} \; \sum_{\text{bijection } \ell} mon(W, \ell) \\ &= \sum_{\text{Walks } W} \; \sum_{X\subseteq[k]} \; \sum_{\ell:[k]\to[k]\setminus X} mon(W, \ell) \end{aligned}$$

Evaluating $P(x, y)$

$$P(x,y) = \sum_{X \subseteq [k]} \sum_{\text{Walks } W} \sum_{\ell:[k] \to [k] \setminus X} mon(W, \ell)$$

- fixing $X \subseteq [k]$ and let $Y = [k] \setminus X$ we obtain a polynomial

$$P_Y(x,y) = \sum_{\text{Walks } W} \sum_{\ell:[k] \to Y} mon(W, \ell)$$

- To evaluate $P_Y(x, y)$ we use Dynamic Programming.
- For $d \leq k$, and vertex v

$$T[v,d] = \sum_{\text{Walk } W: v = v_1 v_2 \ldots v_d} \sum_{\ell:[d] \to Y} \Big(\prod_{e \in W} x_e \Big) \Big(\prod [v_i \in W] y_{v_i, \ell(i)} \Big)$$

We want the value $T[v, k]$ for all vertices $v \in V(G)$.

Evaluating $P(x, y)$

$$T[v,d] = \begin{cases} \sum_{i \in Y} y_{v,i} & d = 1 \\ \sum_{i \in Y} y_{v,i} \sum_{(v,w) \in E(G)} x_{v,w} \cdot T[w, d-1] & \text{otherwise} \end{cases}$$

Once we have computed this table,

$$P_Y(x,y) = \sum_{v \in V(G)} T[v,k]$$

Then over all $Y \subseteq [k]$

$$P(x,y) = \sum_{Y \subseteq [k]} P_Y(x,y)$$

Summary: k-Path via Polynomials

Theorem

There is a randomized FPT algorithm for k-Path running in time $2^k \cdot poly(n)$.

Mainly time for evaluating the polynomial P(x,y).