The Supergraph Method and a Polynomial Delay Algorithm for Generating Connected Induced Subgraphs of a Given Cardinality

Khaled Elbassioni

Masdar Institute of Science and Technology
Abu Dhabi, UAE

Enumeration Algorithms Using Structure, August 24, 2015
Generating $k$-connected Induced Subgraphs

- Let $G = (V, E)$ be an undirected graph of $|V| = n$ vertices, $m = |E|$ edges, and maximum degree $\Delta$
Let $G = (V, E)$ be an undirected graph of $|V| = n$ vertices, $m = |E|$ edges, and maximum degree $\Delta$.

Given an integer $k \in \mathbb{Z}_+$.
Generating \( k \)-connected Induced Subgraphs

- Let \( G = (V, E) \) be an undirected graph of \( |V| = n \) vertices, \( m = |E| \) edges, and maximum degree \( \Delta \)
- Given an integer \( k \in \mathbb{Z}_+ \)

**Connected induced subgraphs of size \( k \)**

Let \( C(G; k) \) be the family of all subsets \( X \subset V \) of vertices such that \( |X| = k \) and the subgraph \( G[X] \) induced on \( X \) is connected
 Generating $k$-connected Induced Subgraphs

Let $G = (V, E)$ be an undirected graph of $|V| = n$ vertices, $m = |E|$ edges, and maximum degree $\Delta$

Given an integer $k \in \mathbb{Z}_+$

**Connected induced subgraphs of size $k$**

Let $\mathcal{C}(G; k)$ be the family of all subsets $X \subset V$ of vertices such that $|X| = k$ and the subgraph $G[X]$ induced on $X$ is connected

**GEN($G; k$)**

Enumerate $\mathcal{C}(G; k)$
Let $G = (V, E)$ be an undirected graph of $|V| = n$ vertices, $m = |E|$ edges, and maximum degree $\Delta$

- Given an integer $k \in \mathbb{Z}_+$

**Connected induced subgraphs of size $k$**

Let $C(G; k)$ be the family of all subsets $X \subseteq V$ of vertices such that $|X| = k$ and the subgraph $G[X]$ induced on $X$ is connected

**GEN($G; k$)**

Enumerate $C(G; k)$

- Has applications in Constraint Satisfaction, Information retrieval, ...
Example: 4-connected Induced Subgraphs
An algorithm for GEN($G; k$)

- is output (or total) polynomial if it outputs all the elements of $\mathcal{C}(G; k)$ in time poly($n, |\mathcal{C}(G; k)|$)
An algorithm for GEN($G; k$)

- is output (or total) polynomial if it outputs all the elements of $\mathcal{C}(G; k)$ in time $\text{poly}(n, |\mathcal{C}(G; k)|)$
- has a polynomial delay of $\text{poly}(n, k)$ if it outputs all the elements of $\mathcal{C}(G; k)$, such that the running time between any two successive outputs is at most $\text{poly}(n, k)$
An algorithm for GEN($G; k$)

- is output (or total) polynomial if it outputs all the elements of $C(G; k)$ in time $\text{poly}(n, |C(G; k)|)$
- has a polynomial delay of $\text{poly}(n, k)$ if it outputs all the elements of $C(G; k)$, such that the running time between any two successive outputs is at most $\text{poly}(n, k)$
- uses linear space, if the total space required by the algorithm (excluding the space for writing the output) is $O(n + m)$
Some Previous Work

- GEN(G; k) trivial to solve in total time $n^k + O(1)$
Some Previous Work

- GEN($G; k$) trivial to solve in total time $n^k + O(1)$
- Uehara (200?):

$$|C(G; k)| \leq n \cdot \frac{(e\Delta)^k}{(\Delta - 1)^k}$$
Some Previous Work

- GEN($G; k$) trivial to solve in total time $n^k + O(1)$
- Uehara (200?):
  \[ |C(G; k)| \leq n \cdot \frac{(e\Delta)^k}{(\Delta - 1)^k} \]
  there exists a family of connected graphs $G$ with maximum degree $\Delta < \frac{2n}{k}$, for which
  \[ |C(G; k)| \geq n\left(\frac{\Delta}{2}\right)^{k-1} \]
Some Previous Work

- GEN($G; k$) trivial to solve in total time $n^k + O(1)$
- Uehara (200?):
  $$|C(G; k)| \leq n \cdot \frac{(e\Delta)^k}{(\Delta - 1)^k}$$
  there exists a family of connected graphs $G$ with maximum degree $\Delta < \frac{2n}{k}$, for which
  $$|C(G; k)| \geq n(\frac{\Delta}{2})^{k-1}$$
- Avis and Fukuda (1996): Reverse Search generates all connected induced subgraphs of size at most $k$ with delay $O(nm)$ and space $O(n + m)$
Some Previous Work

- GEN($G; k$) trivial to solve in total time $n^k + O(1)$
- Uehara (200?):
  \[ |C(G; k)| \leq n \cdot \frac{(e\Delta)^k}{(\Delta - 1)^k} \]
  - there exists a family of connected graphs $G$ with maximum degree $\Delta < \frac{2n}{k}$, for which
    \[ |C(G; k)| \geq n\left(\frac{\Delta}{2}\right)^{k-1} \]
- Avis and Fukuda (1996): Reverse Search generates all connected induced subgraphs of size at most $k$ with delay $O(nm)$ and space $O(n + m)$
- Uehara (200?) noted that such an algorithm is total polynomial for enumerating $C(G; k)$ when $k$ is a fixed constant:
  \[ T := O\left(n + m + \sum_{i=1}^{k-1} |C(G; i)| + k^2 |C(G; k)|\right) = O\left(m + n \cdot \frac{(e\Delta)^k}{(\Delta - 1)^k} k^2\right) \]
Some Previous Work

- $T$ is not total polynomial, when $k$ is part of the input
Some Previous Work

- $T$ is not total polynomial, when $k$ is part of the input
- Trivially, when $k = n - \Delta$:

\[
|C(G; k) \leq \binom{n}{k} < n^\Delta,
\]
Some Previous Work

- $T$ is \textit{not} total polynomial, when $k$ is part of the input
- Trivially, when $k = n - \Delta$:

\[
|C(G; k)| \leq \binom{n}{k} < n^\Delta,
\]

- By the L.B. example above, for all $i < \frac{2n}{\Delta}$:

\[
|C(G; i)| \geq n\left(\frac{\Delta}{2}\right)^{i-1}
\]
Some Previous Work

- $T$ is not total polynomial, when $k$ is part of the input
- Trivially, when $k = n - \Delta$:
  \[
  |C(G; k)| \leq \binom{n}{k} < n^\Delta,
  \]
- By the L.B. example above, for all $i < \frac{2n}{\Delta}$:
  \[
  |C(G; i)| \geq n\left(\frac{\Delta}{2}\right)^{i-1}
  \]
- for $i = \frac{2n}{\Delta} - 1$, $\Delta = n^\epsilon$:
  \[
  T \geq n^{(2n^{1-\epsilon}-2)(\epsilon-\frac{1}{\log n})+1} > |C(G; k)|^{(2n^{1-2\epsilon}-2n^{-\epsilon})(\epsilon-\frac{1}{\log n})} = |C(G; k)|^{n^{\Omega(1)}}
  \]
Some Previous Work

- $T$ is not total polynomial, when $k$ is part of the input
- Trivially, when $k = n - \Delta$:
  \[|C(G; k)\leq \binom{n}{k} < n^{\Delta},\]
- By the L.B. example above, for all $i < \frac{2n}{\Delta}$:
  \[|C(G; i)| \geq n\left(\frac{\Delta}{2}\right)^{i-1}\]
- for $i = \frac{2n}{\Delta} - 1$, $\Delta = n^\epsilon$:
  \[T \geq n^{(2n^{1-\epsilon}-2)(\epsilon-\frac{1}{\log n})+1} > |C(G; k)|^{(2n^{1-2\epsilon}-2n^{-\epsilon})(\epsilon-\frac{1}{\log n})} = |C(G; k)|^{\Omega(1)}\]
- Karakashian, Choueiry and Hartke (2013) gave an algorithm with delay $O(\Delta^k)$
A Polynomial delay Algorithm

Theorem

(i) There is an algorithm for solving \( \text{GEN}(G; k) \), for any graph \( G = (V, E) \) and integer \( k \in \mathbb{Z}_+ \), with polynomial delay \( O(k^2 \cdot \min\{(n - k), k\Delta\} \cdot (\Delta + \log k + \log n)) \)
Theorem

(i) There is an algorithm for solving $GEN(G; k)$, for any graph $G = (V, E)$ and integer $k \in \mathbb{Z}_+$, with polynomial delay $O(k^2 \cdot \min\{(n - k), k\Delta\} \cdot (\Delta + \log k + \log n))$

(ii) There is an algorithm for solving $GEN(G; k)$, for any graph $G = (V, E)$ and integer $k \in \mathbb{Z}_+$, with polynomial delay $O((k \min\{(n - k), k\Delta\})^2(\Delta + \log k))$ and space $O(n + m)$
Define a directed (super)graph $G = (\mathcal{C}(G; k), \mathcal{E})$ on the family $\mathcal{C}(G; k)$.
Define a directed (super)graph $G = (\mathcal{C}(G; k), \mathcal{E})$ on the family $\mathcal{C}(G; k)$.

The arcs of $G := G_N$ are defined by a neighborhood function $\mathcal{N} : \mathcal{C}(G; k) \rightarrow 2^{\mathcal{C}(G; k)}$:

$$\mathcal{N}(X) := \text{set of its successors in } G, \text{ for } X \in \mathcal{C}(G; k)$$
The Supergraph Method

- Define a directed \textit{(super)graph} $G = (\mathcal{C}(G; k), \mathcal{E})$ on the family $\mathcal{C}(G; k)$.
- The arcs of $G := G_\mathcal{N}$ are defined by a \textit{neighborhood function} $\mathcal{N}: \mathcal{C}(G; k) \to 2^{\mathcal{C}(G; k)}$:
  \[
  \mathcal{N}(X) := \text{set of its successors in } G, \text{ for } X \in \mathcal{C}(G; k)
  \]
- A root node: $X_0 \in \mathcal{C}(G; k)$ from which all other nodes of $G$ are reachable.
Define a directed \((super)\)graph \(G = (\mathcal{C}(G; k), \mathcal{E})\) on the family \(\mathcal{C}(G; k)\).

The arcs of \(G := G_N\) are defined by a \textit{neighborhood function} \(N : \mathcal{C}(G; k) \to 2^{\mathcal{C}(G; k)}:\)

\[ N(X) := \text{set of its successors in } G, \text{ for } X \in \mathcal{C}(G; k) \]

A \textit{root node}: \(X_0 \in \mathcal{C}(G; k)\) from which all other nodes of \(G\) are reachable

Algorithm: traverse, either, in \textit{depth-first} or \textit{breadth-first} search order, the nodes of \(G\), starting from \(X_0\)
Define a directed *(super)graph* \( \mathcal{G} = (\mathcal{C}(G; k), \mathcal{E}) \) on the family \( \mathcal{C}(G; k) \)

The arcs of \( \mathcal{G} := \mathcal{G}_N \) are defined by a *neighborhood function* 
\[ \mathcal{N} : \mathcal{C}(G; k) \to 2^{\mathcal{C}(G; k)} : \]

\[ \mathcal{N}(X) := \text{set of its successors in } \mathcal{G}, \text{ for } X \in \mathcal{C}(G; k) \]

A root node: \( X_0 \in \mathcal{C}(G; k) \) from which all other nodes of \( \mathcal{G} \) are reachable

Algorithm: traverse, either, in *depth-first* or *breadth-first* search order, the nodes of \( \mathcal{G} \), starting from \( X_0 \)

If \( \mathcal{G} \) is *strongly connected* then \( X_0 \) can be any node in \( \mathcal{C}(G; k) \)
Lemma

Suppose that

(i) a node $X_0$ in $G$ can be found in time $t_0(n, m, k)$
(ii) for any node $X$ in $G$, $|N(X)| \leq N(n, m, k)$ and we can generate $N(X)$ with delay $t(n, m, k)$
(iii) $G$ is strongly connected

then $C(G; k)$ can be generated with delay

$O\left(\max\{t_0(n, m, k), (t(n, m, k) + \log |C(G; k)|) \cdot N(n, m, k)\}\right)$ and space

$O(n + m + k|C(G; k)|)$
BFS($G_N$) $\implies$ Poly Delay Alg.

Auxiliary data structure:
$BFS(G_N) \implies \text{Poly Delay Alg.}$

Auxiliary data structure:
- **BST** on $C(G; k)$ sorted lexicographically
Auxiliary data structure:

- **BST** on \( C(G; k) \) sorted lexicographically
- **Queue** containing at any point in time all elements of \( C(G; k) \) that have been generated but whose neighborhoods have not been yet explored
Lemma

Suppose that

1. a node $X_0$ in $G$ can be found in time $t_0(n, m, k)$
2. for any node $X$ in $G$, $|\mathcal{N}(X)| \leq N(n, m, k)$ and we can generate $\mathcal{N}(X)$ with delay $t(n, m, k)$
3. there is a function $f : \mathcal{C}(G; k) \setminus \{X_0\} \rightarrow \mathcal{C}(G; k)$, that for every node $X \neq X_0$ in $G$, identifies (in a unique way), in time $t_1(n, m, k)$, a node $X' = f(X)$ in $G$ such that $X \in \mathcal{N}(X')$, and $f$ satisfies the following acyclicity property: there exist no $X_{\ell+1} = X_1, X_2, \ldots, X_\ell \in \mathcal{C}(G; k)$ such that $X_i = f(X_{i+1})$ for $i = 1, \ldots, \ell$,

then $\mathcal{C}(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\})$ and space $O(n + m)$.
DFS(G_N) $\Rightarrow$ Poly Delay, Poly Space Alg.

Trick: output a node X just after the first visit to it, if the depth of X in the tree is odd (assume the root X_0 has depth 1), or X is a leaf; otherwise, X is output just before coming back to the parent.
**Trick**: output a node \( X \) just after the first visit to it, if the depth of \( X \) in the tree is *odd* (assume the root \( X_0 \) has depth 1), or \( X \) is a leaf; otherwise, \( X \) is output *just before coming back to the parent*.
DFS($G_N$) $\Rightarrow$ Poly Delay, Poly Space Alg.
DFS(G_N) $\Rightarrow$ Poly Delay, Poly Space Alg.

OUTPUT: $X_0, X_4$

depth

1
2
3
4
5
6
DFS($G_N$) $\implies$ Poly Delay, Poly Space Alg.

OUTPUT: $X_0, X_4, X_6$

$X_0$

$X_1$

$X_2$

$X_3$

$X_4$

$X_5$

$X_6$

$X_7$

$X_8$

$X_9$

depth

1

2

3

4

5

6
$\text{DFS}(G_N) \implies \text{Poly Delay, Poly Space Alg.}$
$DFS(G_N) \implies Poly\ Delay,\ Poly\ Space\ Alg.$

OUTPUT: $X_0, X_4, X_6, X_7$

depth
1
2
3
4
5
6
DFS($G_N$) $\implies$ Poly Delay, Poly Space Alg.
DFS$(G_N)$ $\Rightarrow$ Poly Delay, Poly Space Alg.
DFS($G_N$) $\implies$ Poly Delay, Poly Space Alg.
$\text{DFS}(G_N) \implies \text{Poly Delay, Poly Space Alg.}$
Note: the delay between two successive outputs is not more than the time to generate the neighborhoods of at most two nodes.
DFS$(G_N) \implies$ Poly Delay, Poly Space Alg.

**Lemma**

Suppose that

(i) a poly$(n, m, k)$ set of nodes $X_0$ in $G$ can be found in time $t_0(n, m, k)$

(ii) for any node $X$ in $G$, $|\mathcal{N}(X)| \leq N(n, m, k)$ and we can generate $\mathcal{N}(X)$ with delay $t(n, m, k)$

(iii') there is a function $f : \mathcal{C}(G; k) \setminus \{X_0\} \rightarrow \mathcal{C}(G; k)$, that for every node $X \notin X_0$ in $G$, identifies (in a unique way), in time $t_1(n, m, k)$, a node $X' = f(X)$ in $G$ such that $X \in \mathcal{N}(X')$, and $f$ satisfies the following acyclicity property: there exist no $X_{\ell+1} = X_1, X_2, \ldots, X_\ell \in \mathcal{C}(G; k)$ such that $X_i = f(X_{i+1})$ for $i = 1, \ldots \ell$,

then $\mathcal{C}(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\})$ and space $O(n + m)$
A Warm Up: Generating connected induced subgraphs of size at most $k$

For a set $X \in \mathcal{C}(G; k)$, define the neighbors of $X$ as those which are obtained from $X$ by adding/deleting one vertex:

$$\mathcal{N}(X) = \{X' \in \bigcup_{i \leq k} \mathcal{C}(G; i) : |X \setminus X'| = 1 \text{ or } |X' \setminus X| = 1\}.$$  

That is, $X, X' \subseteq V$ are neighbors if they differ in exactly one vertex.
A Warm Up: Generating connected induced subgraphs of size at most $k$

For a set $X \in \mathcal{C}(G; k)$, define the neighbors of $X$ as those which are obtained from $X$ by adding/deleting one vertex:

$$\mathcal{N}(X) = \{ X' \in \bigcup_{i \leq k} \mathcal{C}(G; i) : |X \setminus X'| = 1 \text{ or } |X' \setminus X| = 1 \}.$$ 

That is, $X, X' \subseteq V$ are neighbors if they differ in exactly one vertex.

**Lemma**

$\mathcal{G}_\mathcal{N}$ is strongly connected.
For a set $X \in \mathcal{C}(G; k)$, define the neighbors of $X$ as those which are obtained from $X$ by exchanging exactly one vertex:

$$\mathcal{N}(X) = \{X' \in \mathcal{C}(G; k) : X \cap X' = k - 1\}.$$
A Neighborhood Operator for \( C(G; k) \)

For a set \( X \in C(G; k) \), define the neighbors of \( X \) as those which are obtained from \( X \) by exchanging exactly one vertex:

\[
N(X) = \{ X' \in C(G; k) : X \cap X' = k - 1 \}.
\]

Lemma

Let \( X, Y \) be two distinct elements of \( C(G; k) \). Then there exist vertex sets \( X_1, X_2, \ldots, X_\ell \in C(G; k) \) such that \( X_1 = X, \ X_\ell = Y, \ \ell \leq n - k + 1 \), and for \( i = 1, \ldots, \ell - 1 \), \( X_{i+1} \in N(X_i) \) Thus, \( G_N \) is strongly connected.
$G_N$ is Strongly Connected
Recall

**Lemma**

Suppose that (i), (ii) and (iii’) hold, then \( C(G; k) \) can be generated with delay \( O(\max\{t_0(n, m, k), (t(n, m, k) + \log |C(G; k)|) \cdot N(n, m, k)\}) \) and space \( O(n + m + k|C(G; k)|) \).
Recall

**Lemma**

Suppose that (i), (ii) and (iii’) hold, then $C(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + \log |C(G; k)|) \cdot N(n, m, k)\}$ and space $O(n + m + k|C(G; k)|)$

- $|\mathcal{N}(X)| \leq k \cdot \min\{(n - k), k \Delta\}$
A Poly Delay Alg. for GEN(G; k)

Recall

Lemma

Suppose that (i), (ii) and (iii’) hold, then \( C(G; k) \) can be generated with delay \( O(\max\{t_0(n, m, k), (t(n, m, k) + \log |C(G; k)|) \cdot N(n, m, k)\}) \) and space \( O(n + m + k|C(G; k)|) \)

- \( |N(X)| \leq k \cdot \min\{(n - k), k\Delta\} \)
- \( t_0(n, m, k) = O(k\Delta); \ t(n, m, k) = O(k(\Delta + \log k)) \)
A Poly Delay Alg. for GEN(G; k)

Recall

**Lemma**

Suppose that (i), (ii) and (iii’) hold, then $C(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + \log |C(G; k)|) \cdot N(n, m, k)\}$ and space $O(n + m + k|C(G; k)|)$

- $|\mathcal{N}(X)| \leq k \cdot \min\{(n - k), k\Delta\}$
- $t_0(n, m, k) = O(k\Delta)$; $t(n, m, k) = O(k(\Delta + \log k))$
- $|C(G; k)| \leq n \cdot \frac{(e\Delta)^k}{(\Delta-1)k} \quad \text{[Uehara (200?)]}$
A Poly Delay Alg. for GEN\((G; k)\)

Recall

**Lemma**

Suppose that (i), (ii) and (iii’) hold, then \(C(G; k)\) can be generated with delay \(O(\max\{t_0(n, m, k), (t(n, m, k) + \log |C(G; k)|) \cdot N(n, m, k)\})\) and space \(O(n + m + k|C(G; k)|)\)

- \(|\mathcal{N}(X)| \leq k \cdot \min\{(n - k), k\Delta\}\)
- \(t_0(n, m, k) = O(k\Delta)\); \(t(n, m, k) = O(k(\Delta + \log k))\)
- \(|C(G; k)| \leq n \cdot \frac{(e\Delta)^k}{(\Delta-1)^k} [\text{Uehara (200?)})\]

**Corollary**

There is an algorithm for solving GEN\((G; k)\), for any graph \(G = (V, E)\) and integer \(k \in \mathbb{Z}^+\), with polynomial delay \(O(k^2 \cdot \min\{(n - k), k\Delta\} \cdot (\Delta + \log k + \log n))\)
Lemma

Suppose that

(i) a node $X_0$ in $G$ can be found in time $t_0(n, m, k)$

(ii) ...

(iii') there is a function $f : \mathcal{C}(G; k) \setminus \{X_0\} \rightarrow \mathcal{C}(G; k)$, that for every node $X \neq X_0$ in $G$, identifies (in a unique way), in time $t_1(n, m, k)$, a node $X' = f(X)$ in $G$ such that $X \in \mathcal{N}(X')$, and $f$ satisfies the following acyclicity property: there exist no $X_{\ell+1} = X_1, X_2, \ldots, X_\ell \in \mathcal{C}(G; k)$ such that $X_i = f(X_{i+1})$ for $i = 1, \ldots, \ell$,

then $\mathcal{C}(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\}$ and space $O(n + m)$.
Suppose that

(i) a node $X_0$ in $G$ can be found in time $t_0(n, m, k)$

(ii) ...

(iii') there is a function $f : C(G; k) \setminus \{X_0\} \rightarrow C(G; k)$, that for every node $X \neq X_0$ in $G$, identifies (in a unique way), in time $t_1(n, m, k)$, a node $X' = f(X)$ in $G$ such that $X \in N(X')$, and $f$ satisfies the following acyclicity property: there exist no $X_{\ell+1} = X_1, X_2, \ldots, X_\ell \in C(G; k)$ such that $X_i = f(X_{i+1})$ for $i = 1, \ldots, \ell$,

then $C(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\})$ and space $O(n + m)$

To achieve (iii'), we show that every node $X \in C(G; k) \setminus \{X_0\}$ is reachable from $X_0$ by a monotonically increasing lexicographically ordered sequence of nodes.
Lexicographic Ordering

We would like:

**Lemma**

Fix an order on the vertices of $G$ and consider the lexicographic ordering $\preceq$ on $C(G; k)$. Let $X$ be any element in $C(G; k)$ that is not lexicographically smallest. Then there is an $X' \in C(G; k)$ such that $X' \in N(X)$ and $X' \prec X$

- Unfortunately, not true!
We would like:

**Lemma**

Fix an order on the vertices of $G$ and consider the lexicographic ordering $\preceq$ on $\mathcal{C}(G; k)$. Let $X$ be any element in $\mathcal{C}(G; k)$ that is not lexicographically smallest. Then there is an $X' \in \mathcal{C}(G; k)$ such that $X' \in \mathcal{N}(X)$ and $X' \prec X$.

- Unfortunately, not true!
We would like:

**Lemma**

Fix an order on the vertices of $G$ and consider the lexicographic ordering $\preceq$ on $\mathcal{C}(G; k)$. Let $X$ be any element in $\mathcal{C}(G; k)$ that is not lexicographically smallest. Then there is an $X' \in \mathcal{C}(G; k)$ such that $X' \in \mathcal{N}(X)$ and $X' \prec X$.

- Unfortunately, not true!
Use a DFS order!

**Lemma**

Consider the lexicographic ordering \( \preceq \) on \( \mathcal{C}(G; k) \) defined by a DFS order on the vertices of \( G \). Let \( X \) be any element in \( \mathcal{C}(G; k) \) that is not lexicographically smallest. Then there is an \( X' \in \mathcal{C}(G; k) \) such that \( X' \in \mathcal{N}(X) \) and \( X' \prec X \).
$G_N$ Satisfies $(iii')$
Observation: for $x < y$, the DFS-tree walk (that is, the sequence of vertices visited on the way in the DFS order) from $x$ to $y$ contain only nodes $z \leq y$
Recall

**Lemma**

Suppose that (i), (ii) and (iii’) hold, then \( C(G; k) \) can be generated with delay \( O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\}) \) and space \( O(n + m) \)
Recall

**Lemma**

Suppose that (i), (ii) and (iii') hold, then \( C(G; k) \) can be generated with delay \( O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\} \) and space \( O(n + m) \)

- \( |\mathcal{N}(X)| \leq k \cdot \min\{(n - k), k\Delta\} \)
Recall

**Lemma**

Suppose that (i), (ii) and (iii') hold, then $C(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\}$ and space $O(n + m)$

- $|N(X)| \leq k \cdot \min\{(n - k), k\Delta\}$
- $t_0(n, m, k) = O(k\Delta)$; $t(n, m, k) = O(k(\Delta + \log k))$
Recall

Lemma

Suppose that (i), (ii) and (iii') hold, then $C(G; k)$ can be generated with delay $O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\})$ and space $O(n + m)$

- $|\mathcal{N}(X)| \leq k \cdot \min\{(n - k), k\Delta\}$
- $t_0(n, m, k) = O(k\Delta)$; $t(n, m, k) = O(k(\Delta + \log k))$
- $t_1(n, m, k) \leq t(n, m, k) \cdot \min\{(n - k), k\Delta\}$
Recall

**Lemma**

Suppose that (i), (ii) and (iii') hold, then \( C(G; k) \) can be generated with delay \( O(\max\{t_0(n, m, k), (t(n, m, k) + t_1(n, m, k)) \cdot N(n, m, k)\}) \) and space \( O(n + m) \)

- \(|\mathcal{N}(X)| \leq k \cdot \min\{(n - k), k\Delta\}\)
- \(t_0(n, m, k) = O(k\Delta); \ t(n, m, k) = O(k(\Delta + \log k))\)
- \(t_1(n, m, k) \leq t(n, m, k) \cdot \min\{(n - k), k\Delta\}\)

**Corollary**

There is an algorithm for solving \( \text{GEN}(G; k) \), for any graph \( G = (V, E) \) and integer \( k \in \mathbb{Z}_+ \), with polynomial delay \( O((k \min\{(n - k), k\Delta\})^2(\Delta + \log k)) \) and space \( O(n + m) \)
More on the Supergraph Approach

Can be used in many enumeration problems:

1. Feedback edge/vertex sets in directed graphs [Schwikowski and Speckenmeyer (2002)]
2. Spanning trees
3. Perfect matchings in bipartite graphs
4. Minimal blockers to perfect matchings in bipartite graphs [Boros, E. and Gurvich (2004)]
5. Minima of submodular functions
6. Minimal transversals of hypergraphs with bounded intersections
7. Vertices of simple polytopes
8. ....
Generating spanning trees

Given a spanning tree $X$, define a neighbour $X'$ by adding an edge to get a cycle, then removing an edge from the cycle to get a spanning tree $X'$. $G_N$ is strongly connected:
Generating Minimal blockers to Bipartite Perfect matchings

Minimal blocker

A minimal blocker in a bipartite graph $G$ is a minimal set of edges the removal of which leaves no perfect matching in $G$.
Generating Minimal blockers to Bipartite Perfect matchings

**Minimal blocker**

A minimal blocker in a bipartite graph $G$ is a minimal set of edges the removal of which leaves no perfect matching in $G$.

- The neighborhood definition requires a ”good” characterization of a minimal blocker.
Generating Minimal blockers to Bipartite Perfect matchings

**Minimal blocker**

A minimal blocker in a bipartite graph $G$ is a minimal set of edges the removal of which leaves no perfect matching in $G$

- The neighborhood definition requires a "good" characterization of a minimal blocker.
- Strong connectivity proof is obtained by showing a path between every pair of minimal blockers that goes through one of the "simple" blockers (the stars).
Given a polytope \( P = \{ x \in \mathbb{R}^n : Ax = b, \ x \geq 0 \} \), \( A \in \mathbb{R}^{m \times n} \), find the set of vertices (extreme points) \( V(P) \) of \( P \).
Consider a supergraph $G$ on $\mathcal{V}(P)$:
- Two vertices of $G$ are neighbors iff there is a (polyhedral) edge in $P$ connecting them.
Consider a supergraph $G$ on $\mathcal{V}(P)$:

- Two vertices of $G$ are neighbors iff there is a (polyhedral) edge in $P$ connecting them.
- $G$ is strongly connected
Consider a supergraph $G$ on $\mathcal{V}(P)$:

- Two vertices of $G$ are neighbors iff there is a (polyhedral) edge in $P$ connecting them.
- $G$ is strongly connected.
- However, finding $\mathcal{N}(x)$ for $x \in \mathcal{V}(P)$ is equivalent to the original problem [Provan (1994)].
Example I: Simple Polytopes

- Given $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^m$
- $P = \{x \in \mathbb{R}^n : Ax = b, \ x \geq 0\}$
- Assume $P$ is bounded and $\text{rank}(A) = m$
- For $x \in P$, $\text{Supp}(x) = \{i \in [n] : x_i > 0\}$
- $P$ is simple if $\text{Supp}(x) = m$ for all $x \in \mathcal{V}(P)$
Example I: Simple Polytopes

- Given $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^m$
- $P = \{x \in \mathbb{R}^n : Ax = b, \ x \geq 0\}$
- Assume $P$ is bounded and $\text{rank}(A) = m$
- For $x \in P$, $\text{Supp}(x) = \{i \in [n] : x_i > 0\}$
- $P$ is simple if $\text{Supp}(x) = m$ for all $x \in \mathcal{V}(P)$
- For every vertex $x \in \mathcal{V}(P)$ there is a feasible basis $B(x)$ of rank $m$, and vice versa
Example I: Simple Polytopes

- Given $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^m$
- $P = \{ x \in \mathbb{R}^n : Ax = b, x \geq 0 \}$
- Assume $P$ is bounded and rank($A$) = $m$
- For $x \in P$, $\text{Supp}(x) = \{ i \in [n] : x_i > 0 \}$
- $P$ is simple if $\text{Supp}(x) = m$ for all $x \in \mathcal{V}(P)$
- For every vertex $x \in \mathcal{V}(P)$ there is a feasible basis $B(x)$ of rank $m$, and vice versa
- $x'$ is a neighbor of $x$ if $B(x')$ is obtained from $B(x)$ by exchanging a column (pivoting)
Example I: Simple Polytopes

- Given $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^m$
- $P = \{x \in \mathbb{R}^n : Ax = b, \ x \geq 0\}$
- Assume $P$ is bounded and $\text{rank}(A) = m$
- For $x \in P$, $\text{Supp}(x) = \{i \in [n] : x_i > 0\}$
- $P$ is simple if $\text{Supp}(x) = m$ for all $x \in \mathcal{V}(P)$
- For every vertex $x \in \mathcal{V}(P)$ there is a feasible basis $B(x)$ of rank $m$, and vice versa
- $x'$ is a neighbor of $x$ if $B(x')$ is obtained from $B(x)$ by exchanging a column (pivoting)

Theorem (Avis and Fukuda (1991))

If $P$ is simple then $\mathcal{V}(P)$ can be enumerated with polynomial delay.
Example II: Flow Polytope

- Given a directed graph $G = (V, E)$
- Let $A$ be the node–arc incidence matrix of $G$: for $v \in V$, $e \in E$, let $a_{v,e} = \begin{cases} 1 & \text{if arc } e \text{ enters } v \\ -1 & \text{if arc } e \text{ leaves } v \\ 0 & \text{otherwise} \end{cases}$

$P = P(A) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$
- $P$ can be highly degenerate [Provan (1994)]
Example II: Flow Polytope

- Given a directed graph $G = (V, E)$
- Let $A$ be the node–arc incidence matrix of $G$: for $v \in V, e \in E$,
  $$a_{v,e} = \begin{cases} 
  1 & \text{if arc } e \text{ enters } v \\
  -1 & \text{if arc } e \text{ leaves } v \\
  0 & \text{otherwise}
  \end{cases}$$
- $P = P(A) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$
- $P$ can be highly degenerate [Provan (1994)]
- For any $x \in \mathcal{V}(P)$ all polyhedral edges adjacent to $x$ can be enumerated with polynomial delay using “Flashlight”
Example II: Flow Polytope

- Given a directed graph $G = (V, E)$
- Let $A$ be the node–arc incidence matrix of $G$: for $v \in V, e \in E,$

$$a_{v,e} = \begin{cases} 
1 & \text{if arc } e \text{ enters } v \\
-1 & \text{if arc } e \text{ leaves } v \\
0 & \text{otherwise}
\end{cases}$$

- $P = P(A) = \{x \in \mathbb{R}^n : Ax = b, \ x \geq 0\}$
- $P$ can be highly degenerate [Provan (1994)]
- For any $x \in \mathcal{V}(P)$ all polyhedral edges adjacent to $x$ can be enumerated with polynomial delay using “Flashlight”

**Theorem (Provan (1994))**

If $P$ is a flow polytope then $\mathcal{V}(P)$ can be enumerated with incremental polynomial time.
**HYP-TRANS:**

Given a hypergraph $\mathcal{H} \subseteq 2^V$, generate the hypergraph $\text{Tr}(\mathcal{H})$ of all minimal hitting sets (transversals) of $\mathcal{H}$.

\[
\mathcal{H} = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\} \quad \text{Tr}(\mathcal{H}) = \{\{1, 3\}, \{2, 3\}, \{2, 4\}\}.
\]

Complements of minimal transversals are **maximal independent sets**
A Neighborhood for Minimal Transversals

Given $X \in \text{Tr}({\mathcal{H}})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a \textit{minimal} subset of elements $Y$ to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).
Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a \textit{minimal} subset of elements $Y$ to restore transversalility, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.
A Neighborhood for Minimal Transversals

- Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a minimal subset of elements $Y$ to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

- In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.
Given \( X \in \text{Tr}(\mathcal{H}) \), define \( \mathcal{N}(X) \) is the family of all sets that can be obtained from \( X \) by deleting an element from \( i \in X \), adding a \textit{minimal} subset of elements \( Y \) to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, \( X' = Z \cup Y \setminus \{i\} \), for some \( i \in X \setminus X' \) and \( Y \subseteq X' \setminus X \), such that \( Z \subseteq X \setminus \{i\} \) is minimal with the property that \( Z \cup Y \setminus \{i\} \) is a transversal.
Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a *minimal* subset of elements $Y$ to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.
Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a \textit{minimal} subset of elements $Y$ to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.
Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a minimal subset of elements $Y$ to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.

Strong connectivity is straightforward.
Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a minimal subset of elements $Y$ to restore transversality, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.

Strong connectivity is straightforward

Generating neighbors can be done recursively!
Given $X \in \text{Tr}(\mathcal{H})$, define $\mathcal{N}(X)$ is the family of all sets that can be obtained from $X$ by deleting an element from $i \in X$, adding a \textit{minimal} subset of elements $Y$ to restore transversalility, and finally reducing the resulting set to a minimal transversal in a specified way (say in reverse-lexicographic order).

In other words, $X' = Z \cup Y \setminus \{i\}$, for some $i \in X \setminus X'$ and $Y \subseteq X' \setminus X$, such that $Z \subseteq X \setminus \{i\}$ is minimal with the property that $Z \cup Y \setminus \{i\}$ is a transversal.

Strong connectivity is straightforward

Generating neighbors can be done recursively!

Gives incremental polynomial enumeration algorithm if every pair of hyperedges of $\mathcal{H}$ intersect in a constant number of vertices.
Two Open Questions

Is there a ”reasonable” neighborhood definition for any of the following problems:

1. Minimal hypergraph transversals
2. Vertex enumeration is equivalent to enumerating simplices:

Enumeration of Simplices

Given a set of vectors \( A \subseteq \mathbb{R}^d \) containing 0 in their convex hull, enumerate all minimal subsets containing 0 in their convex hull
Two Open Questions

Is there a "reasonable" neighborhood definition for any of the following problems:

1. Minimal hypergraph transversals
2. Vertex enumeration is equivalent to enumerating simplices:

**Enumeration of Simplices**

Given a set of vectors $A \subseteq \mathbb{R}^d$ containing $0$ in their convex hull, enumerate all minimal subsets containing $0$ in their convex hull.
Two Open Questions

Is there a "reasonable" neighborhood definition for any of the following problems:

1. Minimal hypergraph transversals
2. Vertex enumeration is equivalent to enumerating simplices:

**Enumeration of Simplices**

Given a set of vectors $\mathcal{A} \subseteq \mathbb{R}^d$ containing $0$ in their convex hull, enumerate all minimal subsets containing $0$ in their convex hull.
Thank you

QUESTIONS?