Enumeration of Perfect Sequences in Chordal Graphs

Yasuko Matsui, Ryuhei Uehara, Takeaki Uno

Center for Computational Epidemiology and Response Analysis
University of North Texas

May 4, 2011
THE PROBLEM: Algorithm to enumerating all perfect sequences (PS).

A PS is a sequence of maximal cliques obtained by using the reverse order of repeatedly removing the leaves of a clique tree.

Difficulties developing this type of algorithms.

- A chordal graph does not generally have a unique clique tree.
- A PS can normally be generated by two or more distinct clique trees.

It follows: Hard to uses a straightforward way to generate the PS from each possible clique tree.

SOLUTION: A method to enumerate PS without constructing clique trees. Average of $O(1)$ for each sequence.
Approaches

- Naive: Generate all clique trees and generate all PS for each clique tree (can not avoid redundancy)
- Contribution: Make a weighted intersection graph of maximal cliques
  1. Unique construction.
  2. Each MWST of the intersecting graph gives a clique tree.
- Generate each PS from the union of MWST without any repetitions.

OBS: PSs are related to the set of PEOs.
Overview

A graph $G=(V,E)$ is chordal iff has no chordless cycle of length more than three.

The set of maximal cliques in chordal graph $G$ admits special tree structures called clique trees.

Figure: Chordal Graph

Figure: Weighted clique graph
Overview CONT’

- A PS is a sequence of maximal cliques obtained by using the reverse order of repeatedly removing the leaves of a clique tree.
- PS are required and must not have repetitions.

Figure: Weighted clique graph

Figure: Clique trees and perfect sequences
Definition: Weighted Clique Graph

Let \( G = (V, E) \) a chordal graph and the set \( C(G) \) of all maximal cliques. Let \( G(G) = (C(G), \mathcal{E}) \) with a weighted function let \( w: \mathcal{E} \rightarrow \mathbb{Z} \). \( \mathcal{E} \) contains the edge \( C_1, C_2 \) iff \( C_1 \cap C_2 \neq \emptyset \). For each edge in \( E \), \( w(C_1, C_2) \) is defined by \( |C_1 \cap C_2| \); therefore, \( \mathcal{E} \) has a positive integer weight less than \( |V| \).
A chordal graph $G = (V, E)$ is an intersection graph. That is, each vertex $v$ of $G$ corresponds to a subtree $T_v$ of $T$, and $u, v \in E$ iff $T_v$ intersects with $T_u$.

Now, like before, let’s make each $c_i \in T$ correspond to a maximal clique $C_i \in G$. $C_i$ consists of all vertices in $G$ such that $T_v$ contains the node $c_i$. Therefore the tree $T$ is called a clique tree of $G$.

Let’s make an ordering $\pi$ from the set of maximal cliques $C_1, C_2, \ldots, C_k$ of $G$ such that $C_{\pi(i)}$ is a leaf of tree $T_i$. $T_i$ is a subgraph of $T$ induced by $C_{\pi(1)}, C_{\pi(2)}, \ldots C_{\pi(i)}$ for each $i$.

One such a sequence from $T$ can be obtain by repeatedly pruning leaves and put them on top of the sequence until $T$ empty.
For a chordal graph $G$, $C(G)$ is the set of maximal cliques.

**FACT:** $|C(G)| \leq |V|$. Let $k = |C(G)|$.

$C(G) = C_1, C_2, \ldots, C_k$ and $\pi$ be a permutation of $k$ elements.
Lemma (1)

Let $\mathcal{G}(G)$ be the weighted clique graph of a chordal graph $G$ with a weight function $w$. A spanning tree $T$ of this graph is a clique tree of $G$ iff it has the maximum weight.

OBS: We note that any chordal graph of $n$ vertices contains $n$ maximal cliques at most. Therefore, $\mathcal{G}(G)$ contains $\Theta(n)$ nodes. On the other hand, although a star $S_n$ of $n$ vertices contains $E(S) = n - 1$, and $n - 1$ maximal cliques, the $\mathcal{G}(S_n)$ is a complete graph $K_{n-1}$ with $n - 1$ nodes that contains. $P(n - 1, 2) = \Theta(n^2)$ edges. Thus, a trivial upper bound $\Theta(|V|^2)$ for the number of edges in $\mathcal{G}(G)$.

Figure: Star $S_5$

Figure: $\mathcal{G}(S_5)$
Suppose $T$ is a clique tree of a chordal $G = (V, E)$ that consists of at least two maximal cliques; hence, $G$ is not complete (are you sure?).

Remainder:

Given a graph $G = (V, E)$ a $v \in V$ is simplical in $G$ if $N(v)$ is a clique in $G$.

**Lemma (2)**

Let $C$ be a leaf in $T$ and $C'$ be the unique neighbor of $C$. Then for each $v \in C$, $v$ is simplical in $G$ iff $v \in C \setminus C'$. 
Proof.

If \( v \in C \setminus C' \), it is easy to see that \( N(v) = C \setminus v \), and therefore, \( v \) is simplicial. Now suppose a simplicial vertex \( v \in C \cap C' \) to derive a contradiction. Since \( v \in C \), \( N(v) \) contains all the vertices in \( C \), except \( v \). On the other hand, if \( v \) is also in \( C' \), \( N(v) \) contains all the vertices in \( C' \), except \( v \). However, \( C \) and \( C' \) are distinct maximal cliques. Therefore, there are two vertices \( u \in C \) and \( w \in C' \) with \( u, w \notin E \), which contradicts that \( v \) is simplicial. Therefore, \( v \) is in \( C \setminus C' \).

\( \square \)

e.g.

Let \( C = \{a, b, c\} \) and \( C' = \{c, d, e\} \) be neighbors in a clique tree, then the set of \( v \in C \setminus C' = \{a, b\} \) (a clique)
Algorithm 1. Outline of Enumeration

- **Input**: Chordal graph $G = (V, E)$;
- **Output**: All perfect sequences of $G$;
1. Construct weighted clique graph $G(G)$;
2. Compute arbitrary maximum weighted spanning tree $T^*$ of $G(G)$;
3. Construct graph $G(G)^*$ composed of edges that can be included in clique trees from $G(G)$ and $T^*$;
4. Enumerate all sequences of maximal cliques obtained by repeatedly removing maximal cliques that can be leaves of some clique trees.

Algorithm: Perfect Sequences

Efficiently find maximal cliques that can be leaves.

1. Compute maximum weighted spanning tree $T^*$
2. Produce unweighted graph $G(G)^*$ from $G(G)$ and $T^*$
   - An edge $e \in G(G)$ is **unnecessary** if it cannot be included in any maximum weighted spanning tree of $G(G)$.
   - An $e \in G(G)$ is **indispensable** if it appears in any maximum weighted spanning tree of $G(G)$.
   - Other edges are called **dispensable**, appear in some (but not all) $T^*$.
Let $e$ be an edge not in $T^*$. Since $T^*$ is a MST of $G(G)$, the $\{e\} + T^*$ produces a unique cycle $C_e$ which consists of $e$ and the other edges in $T^*$. We call $C_e$ an elementary cycle of $e$.

**Lemma (3)**

For an edge $e \notin T^*$, $w(e) \leq w(e')$ holds for any $e' \in C_e \setminus e$. Moreover, $e$ is unnecessary iff $w(e) < w(e')$ holds for any $e' \in C_e \setminus e$. On the other hand, $e$ is dispensable iff $w(e) = w(e')$ holds for some $e' \in C_e \setminus e$.

**By contradiction.**

If we have $w(e) > w(e_i)$ for some $1 \leq i < k$, by swapping $e$ and $e_i$, we can obtain a heavier spanning tree, which contradicts the fact that $T^*$ is a MST. Therefore, $w(e) \leq w(e_i)$ for each $1 \leq i < k$. When $w(e) = w(e_i)$ for some $1 \leq i < k$, we can obtain a MST $T'$ by removing $e'$ and adding $e$ to $T'$. $T$ does not include $e$ while $T'$ includes $e$, which implies $e$ is dispensable.
Lemma (4)

An edge $e$ in $T^*$ is an **indispensable** edge if $w(e) > w(e')$ for all edges $e'$ such that $e'$ is not on $T$ and $C_{e'}$ contains $e$.

**Proof: Observation.**

There is no edge $e'$ not in $T^*$ such that $C_{e'}$ contains $e$ and $w(e') \geq w(e)$.

Sets of unnecessary, indispensable, and dispensable edges are denoted by $E_u$, $E_i$, and $E_d$, respectively. The sets can be computed by the following algorithm in $O(|G(G)|^3) = O(|V|^3)$ time.
Algorithm 2. Search for Unnecessary, Indispensable, and Dispensable Edges

Input : The weighted clique graph $CG(G) = (C(G), E)$ and an arbitrary maximum weighted spanning tree $T^*$ of $CG(G)$;

Output: Sets $E_u$, $E_i$, and $E_d$ of the unnecessary, indispensable, and dispensable edges;

1 set $E_u := \emptyset$; $E_d := \emptyset$; $E_i := \emptyset$;
2 foreach $e$ not in $T^*$ do
3     if $w(e) < w(e')$ for all $e' \in C_e$ then
4         $E_u := E_u \cup \{e\}$;
5     else
6         $E_d := E_d \cup \{e\}$;
7     foreach $e' \in C_e$ satisfying $w(e) = w(e')$ do
8         $E_d := E_d \cup \{e'\}$;
9     end
10    $E_i := E \setminus (E_u \cup E_d)$;
11 return $(E_i, E_u, E_d)$;
Define: $\mathcal{G}(G)^*$ by $(C(G), \mathcal{E}_i \cup \mathcal{E}_d)$

OBS: Any spanning tree of $\mathcal{G}(G)^*$ that contains all the edges in $\mathcal{E}_i$ gives a maximum weighted spanning tree of $\mathcal{G}(G)$

Lemma (5)

A maximal clique $C$ can be a leaf of a clique tree iff $C$ satisfies (1) $C$ is incident to at most one edge in $\mathcal{E}_i$, and (2) $C$ is not a cut vertex in $\mathcal{G}(G)^*$.

Proof.

First, we suppose that $C$ is a leaf of a clique tree $T$. Since $T$ is a clique tree of $G$, $T$ is a spanning tree in $\mathcal{G}(G)^*$ that includes all the edges in $\mathcal{E}_i$. Since $C$ is a leaf of $T$, $C$ is incident to at most one edge of $\mathcal{E}_i$, and $C$ is not a cut vertex of $\mathcal{G}(G)^*$. Thus, $C$ satisfies the conditions.

We next suppose that $C$ satisfies the conditions. We assume that $\mathcal{G}(G)$ contains two or more nodes. We choose any edge $e$ from $\mathcal{E}_i \cup \mathcal{E}_d$ that is incident to $C$. We always can choose $e$ since $\mathcal{G}(G)$ is connected. Then, we remove $C$ from $\mathcal{G}(G)^*$. Since $C$ is not a cut vertex, the resultant graph $\mathcal{G}(G)'$ is still connected. Therefore, $\mathcal{G}(G)'$ has a spanning tree $T'$ which contains all the edges in $\mathcal{E}_i \setminus e$. Then, by adding $e$ to $T'$, we have a spanning tree $T$ that contains all the edges in $\mathcal{E}_i$, and $C$ is a leaf of $T$. 

\qed
Algorithm 3. All Perfect Sequences

Input: Chordal graph $G = (V, E)$;
Output: All perfect sequences of $G$;
1. construct $\mathcal{C}G(G)$;
2. find maximum weighted spanning tree $T^*$ of $\mathcal{C}G(G)$;
3. by using $T^*$, compute sets $\mathcal{E}_u$, $\mathcal{E}_i$, $\mathcal{E}_d$ of unnecessary, indispensable, and dispensable edges, respectively;
4. set $P$ to empty sequence; // keep current perfect sequence
5. let $\mathcal{C}G(G)^* := (\mathcal{C}(G), \mathcal{E}_i \cup \mathcal{E}_d)$;
6. call Enumerate($\mathcal{C}G(G)^*$, $P$);

Procedure Enumerate($\mathcal{C}G(G)^* = (\mathcal{C}(G), \mathcal{E}_i \cup \mathcal{E}_d)$, $P$)

[H]
Output: A perfect sequence at the last node;
7. if $\mathcal{C}(G)$ contains one node $C$ then
8. output $(C + P)$; // $C + P$ denotes concatenation of node $C$ and sequence $P$

9. else
10. compute $S := \{ C \in \mathcal{C}(G) \mid C$ satisfies the leaf condition$\}$;
11. foreach $C \in S$ do
12. call Enumerate($\mathcal{C}G(G) \setminus C$, $C + P$);
13. end
14. end
Theorem (1)

For any chordal graph $G = (V,E)$, with $O(|V|^3)$ time and $O(|V|^2)$ space pre-computation, all perfect sequences can be enumerated in $O(1)$ time per sequence on average and $O(|V|^2)$ space.
Fig. 2. Part of computation tree that enumerates all perfect sequences.
References