Counting Linear Extensions of Sparse Posets

*Kustaa Kangas*, Teemu Hankala, Teppo Niinimäki, Mikko Koivisto
July 13, 2016

University of Helsinki
Department of Computer Science
Partially ordered set (poset)

A set $P$ with an antisymmetric, transitive relation
Partially ordered set (poset)

Cover graph
Linear extensions

A *linear extension* = order preserving permutation
Counting linear extensions

#P-complete (Brightwell & Winkler, ’91)

Applications: sequence analysis, preference reasoning, sorting, learning probabilistic models, ...
Counting linear extensions

Currently we can do $O(2^n n)$ time for $n$ elements.

We give two algorithms, based on

1. recursion (exploiting low connectivity)
2. variable elimination (exploiting low treewidth)
Counting linear extensions

Currently we can do $O(2^n n)$ time for $n$ elements.

We give two algorithms, based on

1. recursion (exploiting low connectivity)
2. variable elimination (exploiting low treewidth)
Recursive counting

\[ \ell(P) = \text{the number of linear extensions of poset } P \]

**Rule 1**

\[ \ell(P) = \sum_{x \in \text{min}(P)} \ell(P \setminus x) \]

**Rule 2**

\[ \ell(P) = \sum_{(D,U)} \ell(D) \cdot \ell(U) \]

**Rule 3**

\[ \ell(P) = \prod_{i=1}^{k} \ell(S_i) \]

**Rule 4**

\[ \ell(P) = \ell(A) \cdot \ell(B) \cdot \left( \frac{|P|}{|A|} \right) \]
Recursive counting

\[ P \setminus a \]

\[ P \setminus b \]
Recursive counting

$P \setminus a$

$P \setminus b$

$P$
Recursive counting
Recursive counting

Experiments on sparse posets for $n = 30, \ldots, 100$
Counting linear extensions

Currently we can do $O(2^n n)$ time for $n$ elements.

We give two algorithms, based on

1. recursion (exploiting low connectivity)
2. variable elimination (exploiting low treewidth)
Variable elimination

\[ \sum_{a,b,c,d,e,f} \phi_1(a, b, d) \phi_2(a, c) \phi_3(b, c, e) \phi_4(d, f) \]
Variable elimination

\[ \sum_{a,b,c,d,e,f} \phi_1(a, b, d) \phi_2(a, c) \phi_3(b, c, e) \phi_4(d, f) \]
Variable elimination

\[ \sum_{a,b,c,d,e,f} \phi_1(a, b, d) \phi_2(a, c) \phi_3(b, c, e) \phi_4(d, f) \]
Variable elimination

\[
\sum_{a,b,c,d,e,f} \phi_1(a, b, d) \phi_2(a, c) \phi_3(b, c, e) \phi_4(d, f)
\]

Polynomial time for bounded **treewidth**
Variable elimination

For every permutation $\sigma : P \rightarrow [n]$ define

$$\Phi(\sigma) = \prod_{x < y} [\sigma_x < \sigma_y]$$
For every permutation $\sigma : P \rightarrow [n]$ define

$$\Phi(\sigma) = \prod_{x < y} [\sigma_x < \sigma_y]$$

$$\Phi(\sigma) = [\sigma_a < \sigma_c] [\sigma_a < \sigma_d] [\sigma_b < \sigma_d] [\sigma_c < \sigma_e] [\sigma_d < \sigma_e]$$
Variable elimination

For every permutation $\sigma : P \rightarrow [n]$ define

$$\Phi(\sigma) = \prod_{x \prec y} [\sigma_x < \sigma_y]$$

Then,

$$\Phi(\sigma) = \begin{cases} 1, & \text{if } \sigma \text{ is a linear extension,} \\ 0, & \text{otherwise.} \end{cases}$$
For every permutation $\sigma : P \rightarrow [n]$ define

$$\Phi(\sigma) = \prod_{x \prec y} [\sigma_x < \sigma_y]$$

Then,

$$\Phi(\sigma) = \begin{cases} 1, & \text{if } \sigma \text{ is a linear extension}, \\ 0, & \text{otherwise}. \end{cases}$$

As a consequence

$$\ell(P) = \sum_{\sigma : P \rightarrow [n]} \Phi(\sigma)$$
Variable elimination

$$\ell(P) = \sum_{\sigma : P \rightarrow [n]} \Phi(\sigma)$$
Variable elimination

\[ \ell(P) = \sum_{\sigma : P \rightarrow [n]} \Phi(\sigma) \]

Can’t apply variable elimination because of the bijectivity constraint.
Variable elimination

\[ \ell(P) = \sum_{\sigma : P \rightarrow [n]} \Phi(\sigma) \]

\[ = \sum_{X \subseteq [n]} (-1)^{|X|} \sum_{\sigma : P \rightarrow X} \Phi(\sigma) \]

\[ = \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} \sum_{\sigma : P \rightarrow [k]} \Phi(\sigma) \]

Inclusion–exclusion principle
Variable elimination

\[ \ell(P) = \sum_{\sigma : P \rightarrow [n]} \Phi(\sigma) \]

\[
= \sum_{X \subseteq [n]} (-1)^{n-|X|} \sum_{\sigma : P \rightarrow X} \Phi(\sigma)
\]

\[
= \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} \sum_{\sigma : P \rightarrow [k]} \Phi(\sigma)
\]

\[ O(n^{t+4}) \) time for treewidth \( t \) \]
Variable elimination

VEIE: Variable elimination via inclusion–exclusion

![Graph showing time vs. poset size for VEIE, R1, and R14-a methods. The graph is on a logarithmic scale for both axes. At t = 2, the VEIE method shows a distinct pattern compared to the other methods.](image-url)
Summary

- **Recursion**: often fast in practice
- **Variable elimination**: polynomial time for bounded treewidth

Thank you!