HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI
MATEMAATTIS-LUONNONTIETEELLINEN TIEDEKUNTA
MATEMATISK-NATURVETENSKAPLIGA FAKULTETEN
FACULTY OF SCIENCE

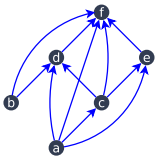
COUNTING LINEAR EXTENSIONS OF SPARSE POSETS

Kustaa Kangas, Teemu Hankala, Teppo Niinimäki, Mikko Koivisto
Department of Computer Science

Counting the linear extensions of a partially ordered set (poset) is a fundamental problem with several applications. We present two exact algorithms that target sparse posets in particular. The first algorithm breaks the counting task into subproblems recursively. The second algorithm uses variable elimination via inclusion–exclusion and runs in polynomial time for posets with a cover graph of bounded treewidth.

PROBLEM

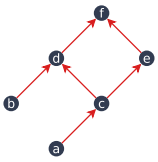
A *poset* P is a set of n elements ordered by a transitive asymmetric relation $<_P$.



For all $x, y \in P$ either

1. $x <_P y$
2. $y <_P x$
3. x and y incomparable

An element x with no $y <_P x$ is *minimal*.

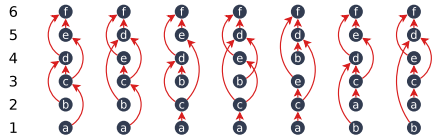


The *cover relation* $\prec_P$ is the transitive reduction of $<_P$.

Represented visually by the *cover graph*.

A *linear extension* is a bijection $\sigma : P \to [n]$ such that

$$x \prec_P y \Rightarrow \sigma_x < \sigma_y$$



Problem: Compute $\ell(P)$, the number of linear extensions. (#P-complete)

RECURSIVE COUNTING

Denoting minimal elements by $\min(P)$,

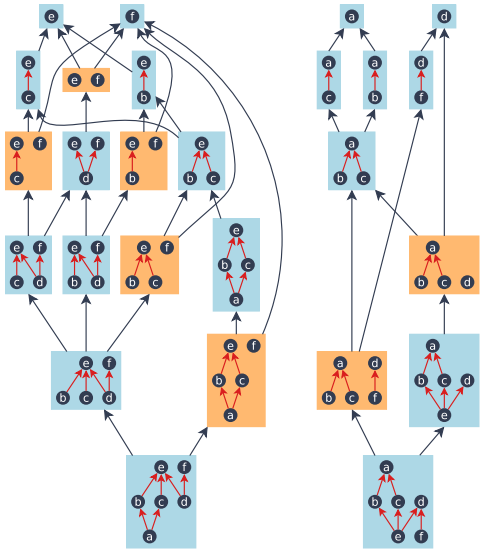
I $$\ell(P) = \sum_{x \in \min(P)} \ell(P \setminus x)$$

If P can be partitioned into pairwise disconnected sets A and B,

II $$\ell(P) = \ell(A) \cdot \ell(B) \cdot \binom{|P|}{|A|}$$

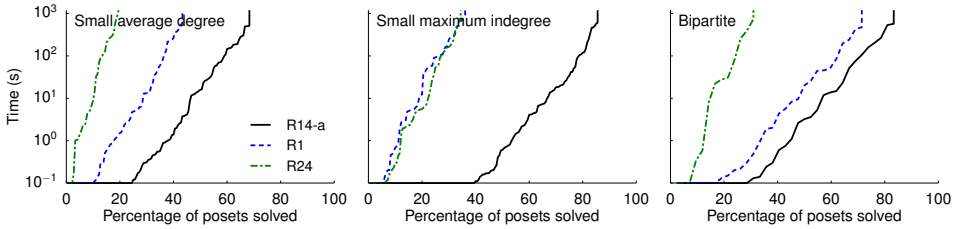
Implementations [1]:

- R1: Applies rule I, runs in $O(2^n)$ time in the worst case
- R14-a: Applies rules I and II, often greatly expedites R1
- R24: Applies rule II together with other techniques [2]



Rules I and II break the poset into subproblems recursively.

The inverse may lead to fewer subproblems.



VARIABLE ELIMINATION VIA INCLUSION–EXCLUSION

For every mapping $\sigma : P \to [n]$ let

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

Then, if σ is a bijection, we have

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

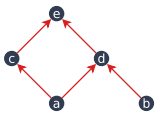
Therefore, $$\ell(P) = \sum_{\substack{\sigma : P \to [n] \\ \text{bijection}}} \Phi(\sigma) .$$

The **inclusion–exclusion** principle removes the bijectivity constraint:

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

→ can use **variable elimination** (e.g. [3,4])

EXAMPLE



The summation task is

$$\sum_{a,b,c,d,e} [a<c]\,[a<d]\,[b<d]\,[c<e]\,[d<e]$$

where $a = \sigma_a$, $b = \sigma_b$, …

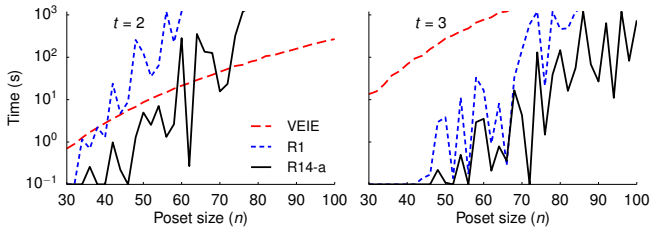
With the *elimination order* (d, b, e, a, c) the sum factorizes:

$$\sum_d \underbrace{\left(\sum_b [b<d] \right)}_{\lambda_b(d)} \underbrace{\left(\sum_e [d<e] \underbrace{\left(\sum_a [a<d] \underbrace{\left(\sum_c [a<c][c<e] \right)}_{\lambda_c(a,e)} \right)}_{\lambda_a(d,e)} \right)}_{\lambda_e(d)}$$

- Good order → the λ are small and fast to compute
- A cover graph of low *treewidth* has good orders

IMPLEMENTATION

The implementation VEIE runs in $O(n^{t+4})$ time for n elements and a cover graph of treewidth t.



[1] Implementation available at www.cs.helsinki.fi/u/jwkangas/lecount/
[2] M. Peczarski. *New results in minimum-comparison sorting*. *Algorithmica*, 40(2):133–145, 2004.
[3] R. Dechter. *Bucket elimination: A unifying framework for reasoning*. *Artificial Intelligence*, 113(12):41–85, 1999.
[4] U. Bertelè and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY