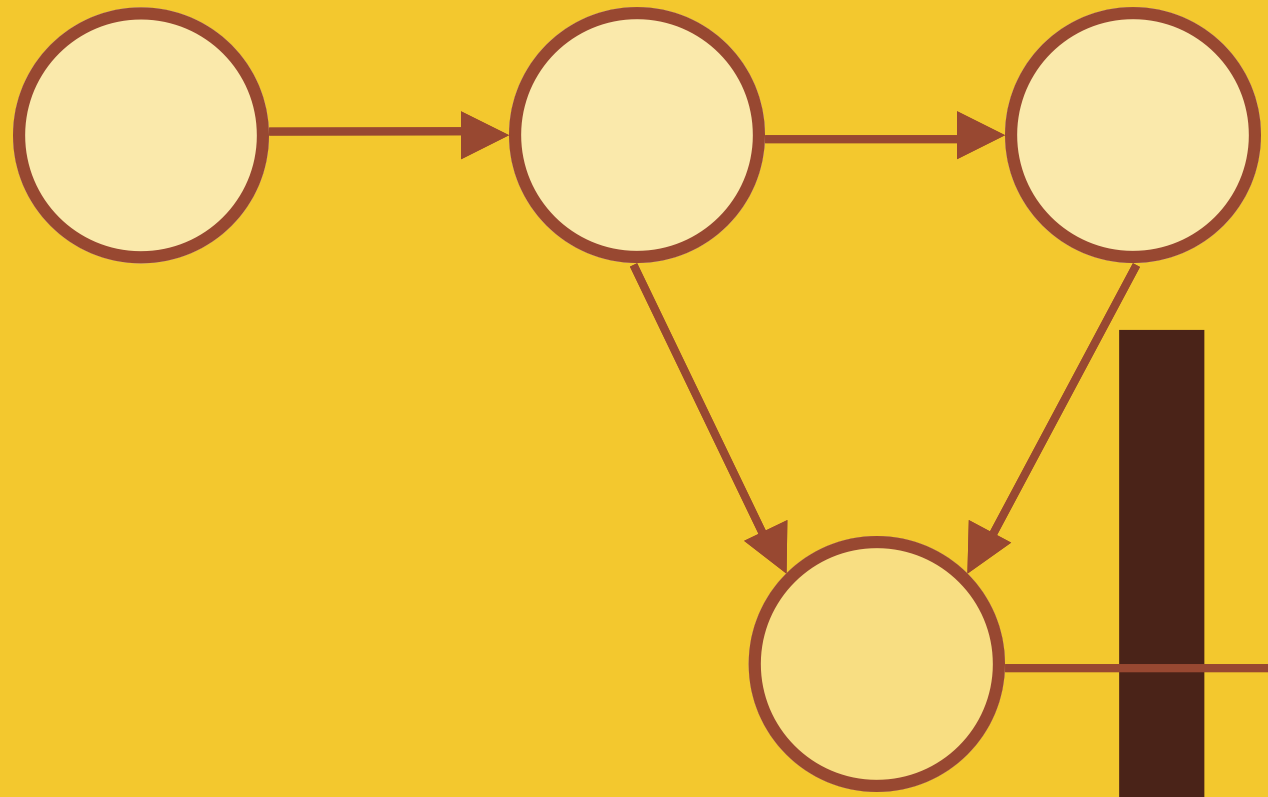
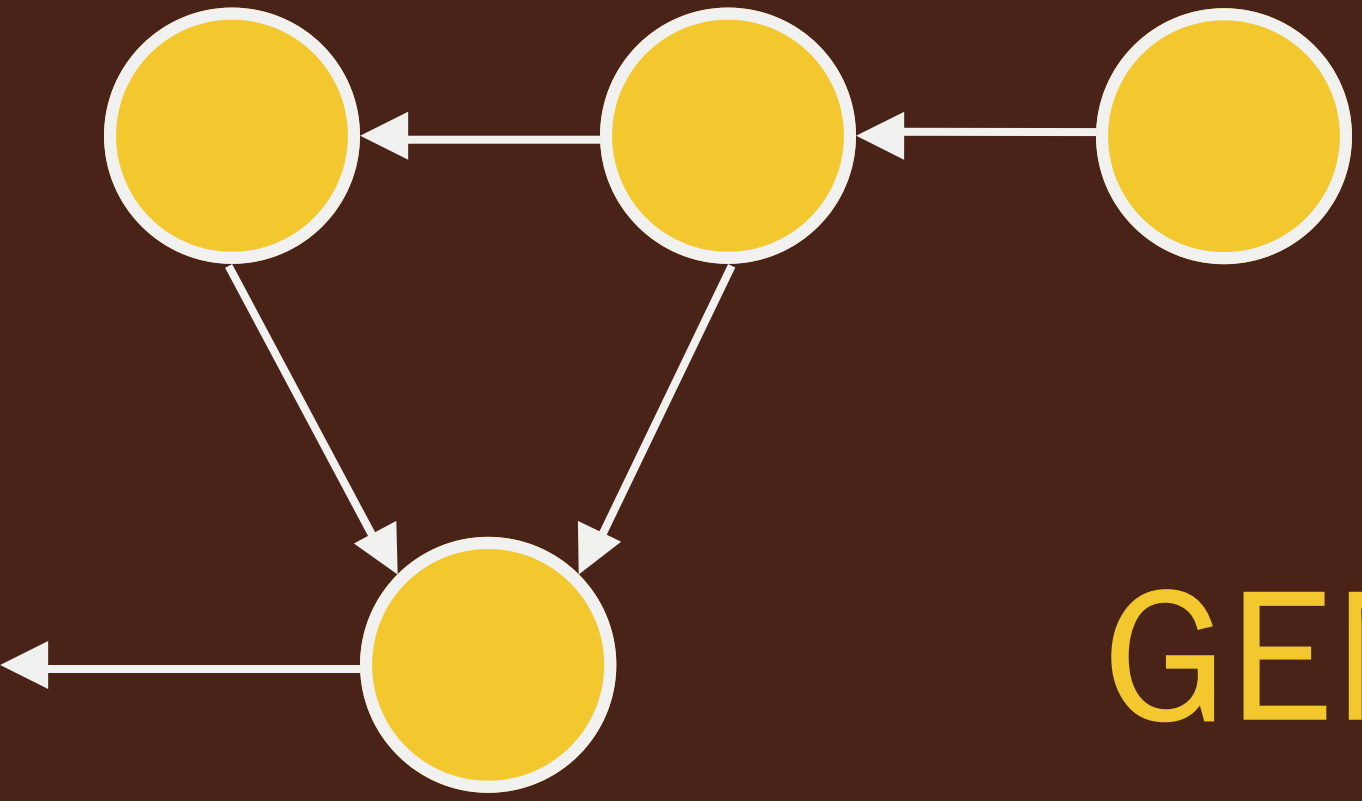


AAAI-16 Tutorial

CP-nets

- Thomas E. Allen
- Judy Goldsmith
- Francesca Rossi



GENERATION

Thomas E. Allen

Motivation

Why generate CP-nets uniformly at random?

- *No human-subjects CP-net databases (yet)*
- *Test algorithms experimentally*
- *Effective Monte Carlo algorithms*
- *Better understand properties of CP-nets*
- *Simulations for decision making and social choice*

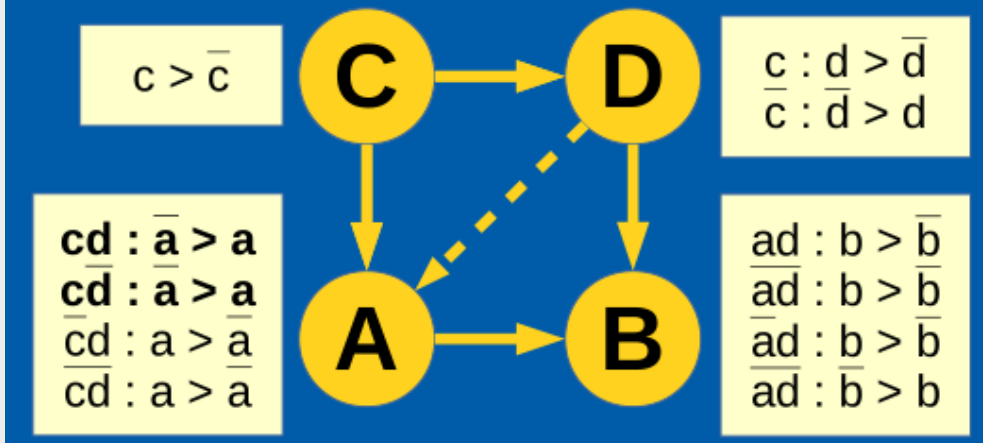
Naïve Generation

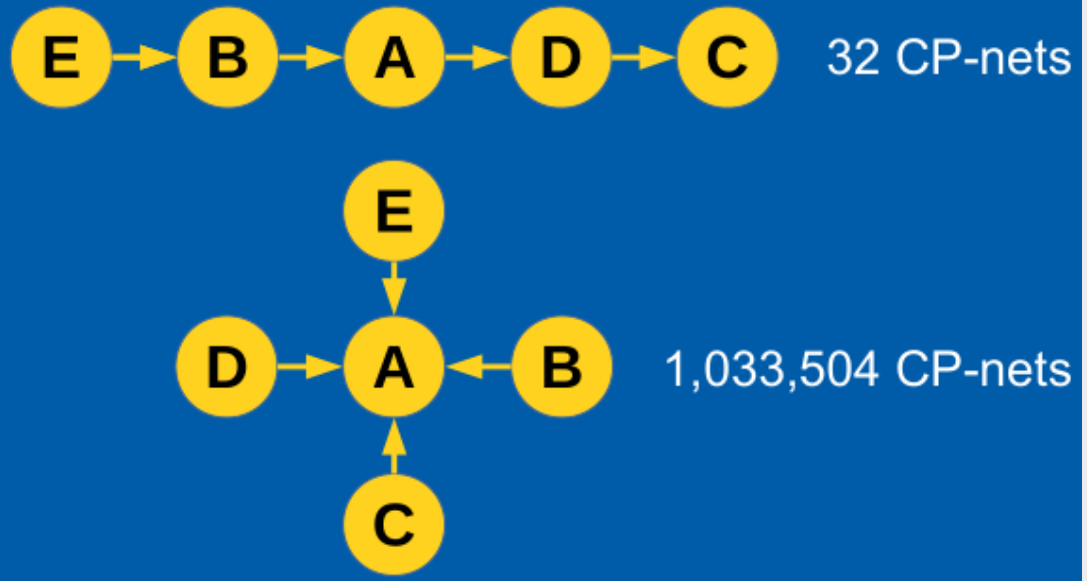
- Initialize CP-net with *n* nodes, no edges, and empty CPTs
- Choose a random subset of pairs , such that
- Insert edge from each to
- Initialize a CPT for each node with the appropriate number of entries
- Generate each rule of the CPT by randomly permuting the domain

The Trouble with Naïve Generation

3

Naïve generation leads to **degeneracy** and **bias**.





Don’t play with loaded dice!

Why we care

- Example: Testing DT algorithm as number of edges increases

Generating Outcomes and Pairs

- Outcomes and pairs of outcomes are easy to generate!
- Select value of each feature independently and uniformly
- Pairs: Generate 2 outcomes; if equivalent, regenerate
- Pairs with constrained Hamming distance h
  - *Generate first outcome*
  - *Randomly select h of n variables and "flip" their values*
  - *Subset selection also easy (see Knuth TAOCP, vol. 2)*

Generating Acyclic CP-nets Uniformly

- Early ideas that didn't work:
  - *Generate all cases (works up to 4 nodes; see next slide)*
  - *Generate all DAGs and count # combinations of CPTs*
- Important considerations
  - *Avoid cycles*
  - *Bound on indegree (# parents)*
  - *Multivalued domains*

Problem: There are many models

n binary features	$\lvert \mathcal{N} \rvert$ number of possible CP-nets
1	2
2	12
3	488
4	481776
5	157549032992 (1.6×10^{11})
6	4059976627283664056256 (4.1×10^{21})

Generation Algorithm

Allen et al., “Generating CP-nets Uniformly at Random” (AAAI-16)

- Encode dependency graph as a **DAG code** (Steinsky, 2003)
- Encode CPTs as Boolean or multivalued discrete functions
- Recurrence: count completions of partially constructed CP-net
- Generate the network one node at a time so as to avoid:
  - *Degeneracy in tables*
  - *Bias related to in-degree*

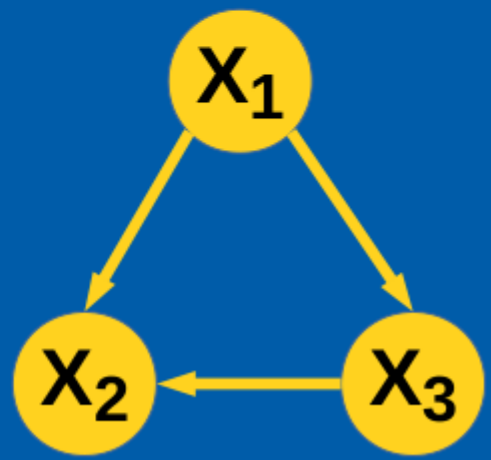
Encoding the Dependency Graph

4

- We encode the graph as a **DAG code**[3]
- The code is a **tuple of subsets** corresponding to each node's parent labels
- We can count codes (hence DAGs)

$$a_{n,c}(j,q) = \sum_{\substack{s\geq 0,\ t\geq 0,\\ s\leq q,\ s+t\leq c,\\ q+t\leq j}} \binom{q}{s}\binom{n-q}{t} a_{n,c}(j+1, q+t), \quad a_{n,c}(n,q) = 1$$

$$\langle \{1\}, \{1,3\} \rangle$$



Encoding the CPTs

5

- Encode the CPTs as Boolean (discrete multivalued) functions
- Rejection sampling is efficient for generation (randomly assign entries, check for degeneracy)
- We extend previous work[4] to obtain the number of non-degenerate discrete multi-valued functions:

$$\psi_d(m) = \sum_{k=0}^{m} (-1)^{m-k} \binom{m}{k} d!^{d^k}$$

Generating CP-nets Uniformly

6

- Generate CP-net via its dagcode and CPT, one node at a time
- Check each CPT for degeneracy; regenerate if necessary (rare)
- To avoid bias, use recurrence below counting number of CP-nets:

$$a_{n,c,d}(j,q) = \sum_{\substack{s\geq 0,\ t\geq 0,\\ s\leq q,\ s+t\leq c,\\ q+t\leq j}} \binom{q}{s}\binom{n-q}{t}\,\psi_d(s+t)\,a_{n,c,d}(j+1,q+t),\quad a_{n,c,d}(n,q)=1$$

What we can generate

- Acyclic CP-nets
- Arbitrary bound on in-degree or unbounded
- Binary or homogeneous multivalued domains
- Each CP-net (graph + combination of CPTs) equally likely
- Outcomes, pairs of outcomes, pairs with fixed HD

Introducing `GenCPnet`

- Implemented in C++ as a command line tool
- Runs on GNU/Linux systems (requires GMP big num library)
- Free and open source software (GPL3)
- *Input:* parameters (# nodes, bound on indegree, etc.)
- *Output:* XML representation (Santhanam's `Crisner` tool)
- Can also generate sets of DT problems, # CP-nets

Introducing `GenCPnet`

```
gencpnet -n 5 -d 3 TrinaryDomains

gencpnet -n 10 -c 2 -g 10 -t 1 -h 2 HammingTest

gencpnet -n 20 -c 4 -g 10 -t 10 DominanceTests
```

Other cases

- CP-nets with incomplete tables
- Tree-shaped CP-nets
- Polytree CP-nets
- Heterogeneous domains (future work)
- Cyclic CP-nets (future work)

Questions?