Recognizing Partial Cubes in Quadratic Time

David Eppstein
Computer Science Dept.
Univ. of California, Irvine

ACM-SIAM Symposium on Discrete Algorithms, 2008

Context: Geometric graphs and metric embedding

Graph theory:

Unweighted graphs

Weighted graphs

Finite metric spaces

Geometry:

Real vector spaces

Integer lattices

Euclidean distances

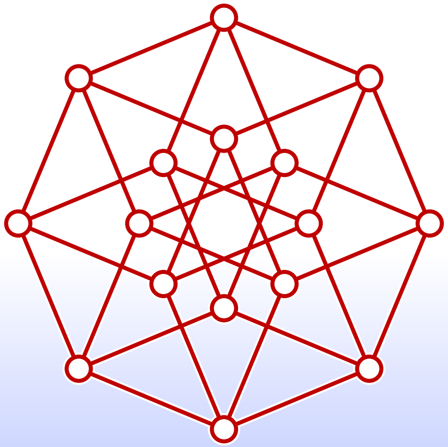
L_1 distances

L_∞ distances

Probabilistic tree embedding

Bourgain's theorem

Johnson-Lindenstrauss lemma ...



Context: Geometric graphs and metric embedding

Graph theory:

Unweighted graphs

Weighted graphs

Finite metric spaces

Geometry:

Real vector spaces

Integer lattices

Euclidean distances

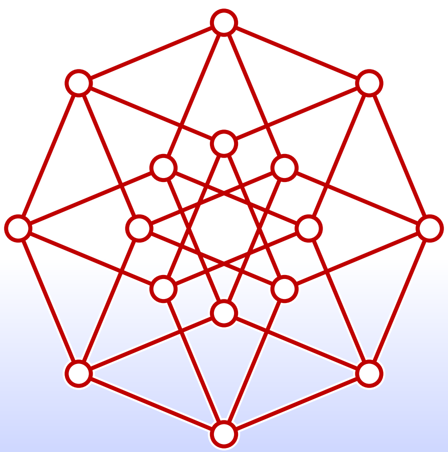
L_1 distances

L_∞ distances

Probabilistic tree embedding

Bourgain's theorem

Johnson-Lindenstrauss lemma ...

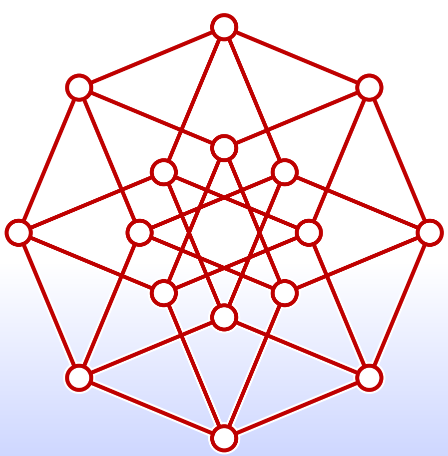


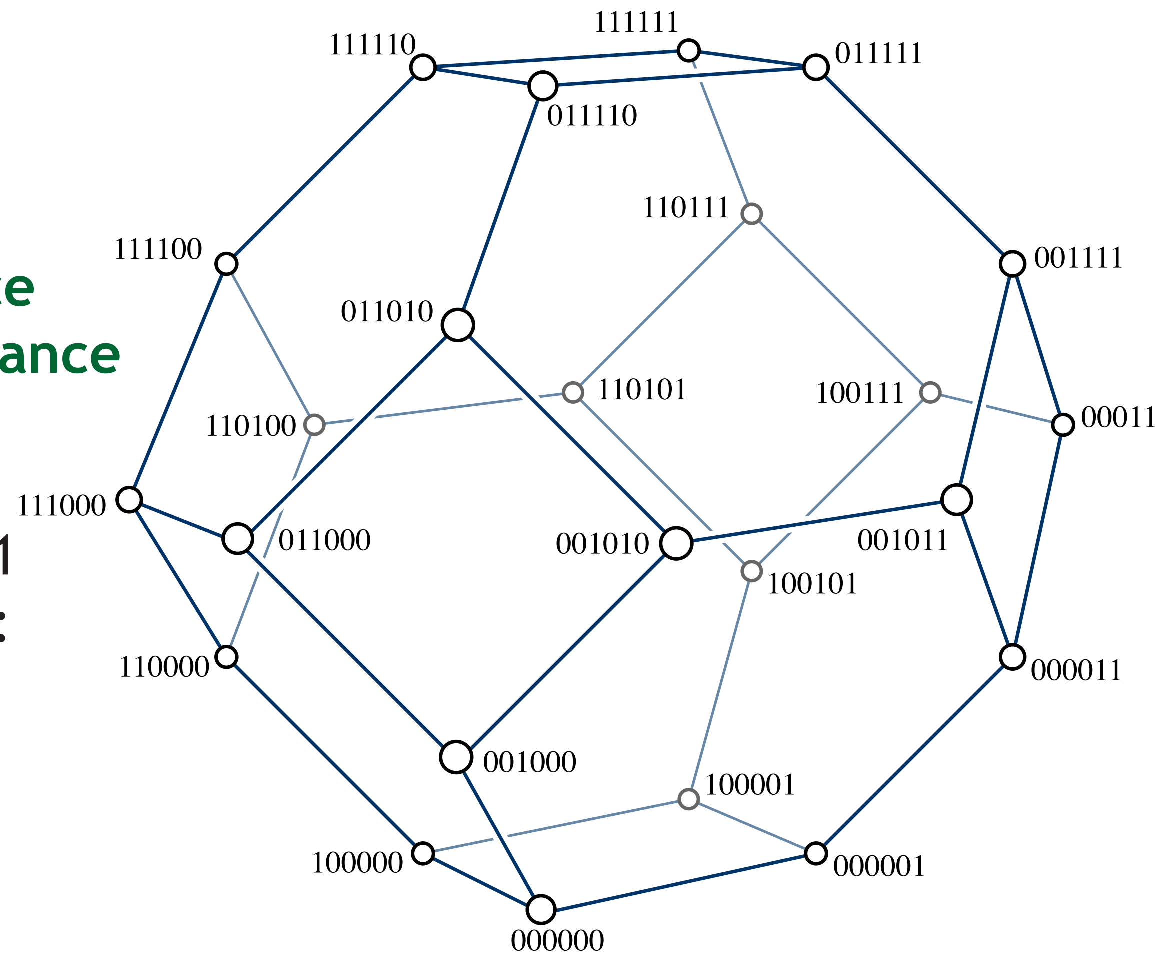
Partial cubes as geometric graphs

Partial cube:

Undirected graph that can be embedded into an **integer lattice** so that **graph distance = L_1 distance**

At expense of high dimension can restrict coordinates to 0 or 1
L_1 distance = Hamming distance:
isometric hypercube subgraph





Example: permutahedron
(vertices = permutations of 4 items
edges = flips of adjacent items)

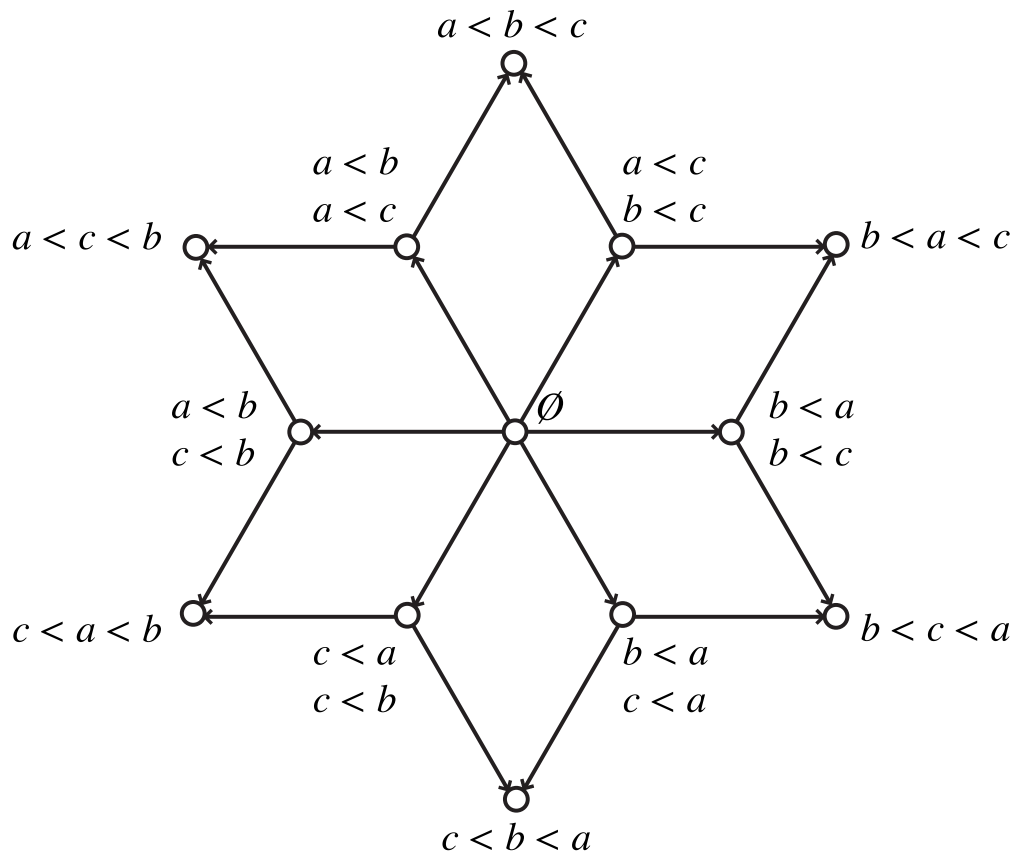
Application:
Preference modeling in mathematical behavioral sciences

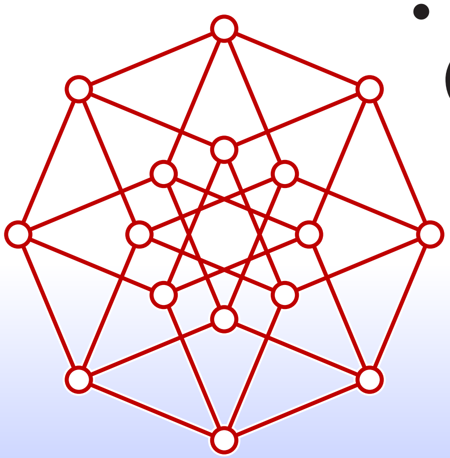
Given a fixed set of candidates

Model voter states as vertices
Possible state transitions as edges

Several natural families of orderings
define partial cubes in this way:

- total orderings
- partial orderings
- weak orderings
  (total orders with ties)





Application: Modeling knowledge of students

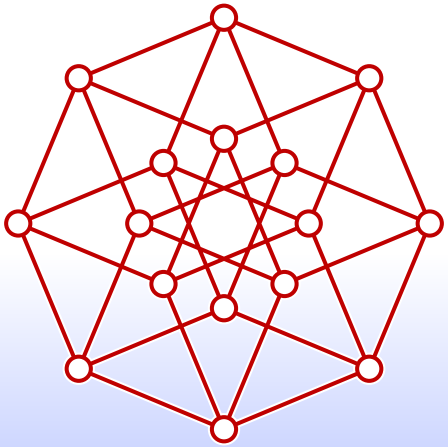
State of knowledge
= set of concepts the student understands

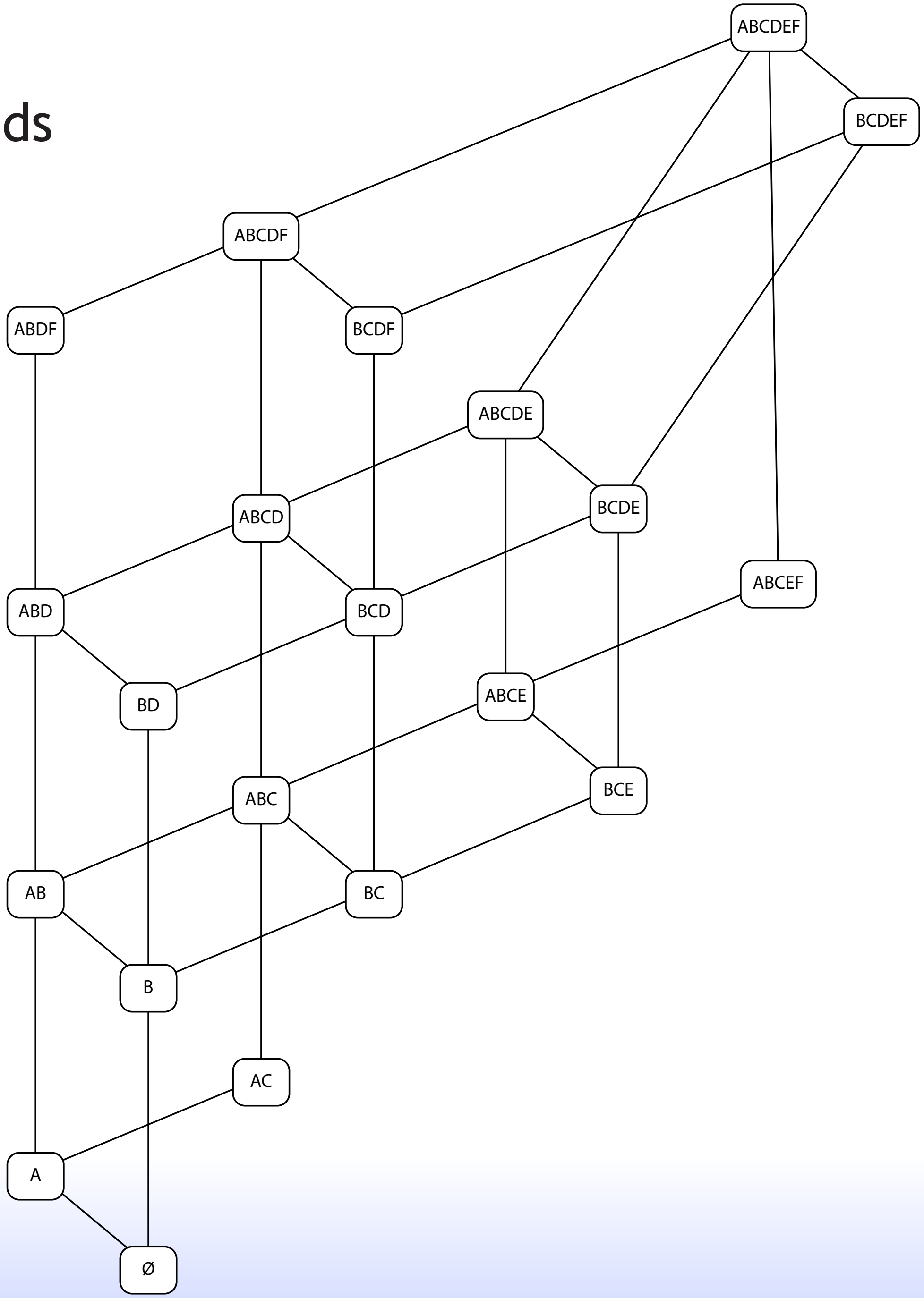
Assume:
Any state can be reached by learning one concept at a time

Union of two states is another state

Then family of states is an antimatroid, a special case of a partial cube

This theory is used by ALEKS Corp. in their educational software for high school mathematics





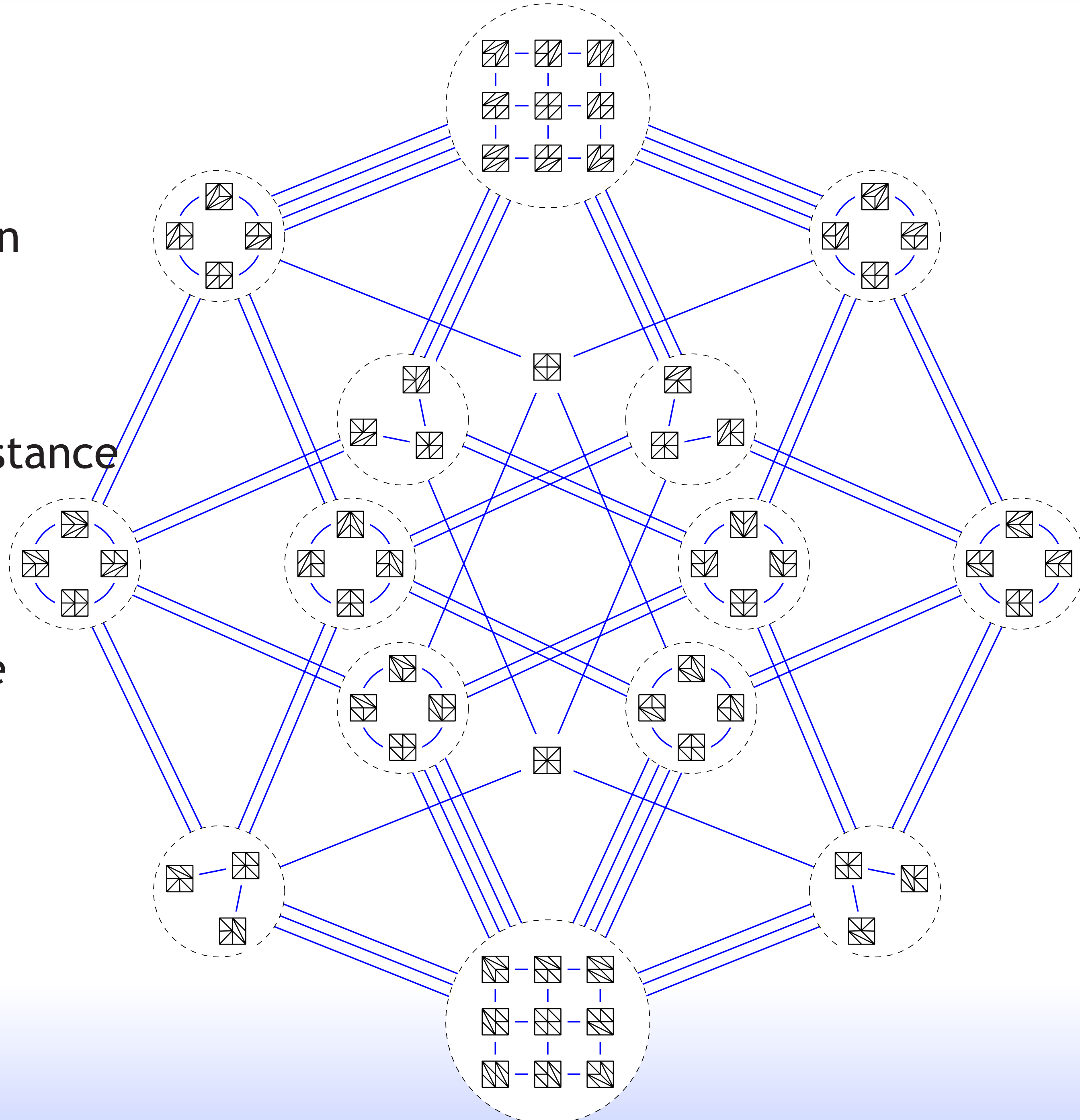
Application: flip distances in computational geometry

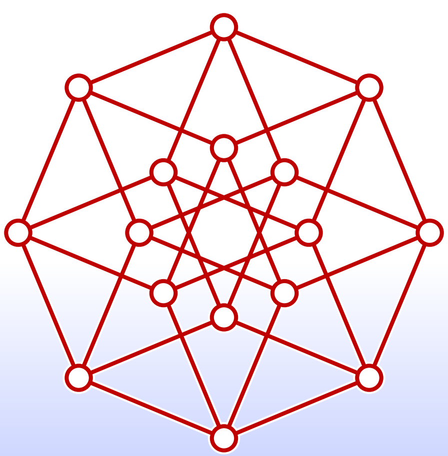
Vertices = triangulations
(here, of 3x3 grid)

Edges = change triangulation
by one edge ("flip")

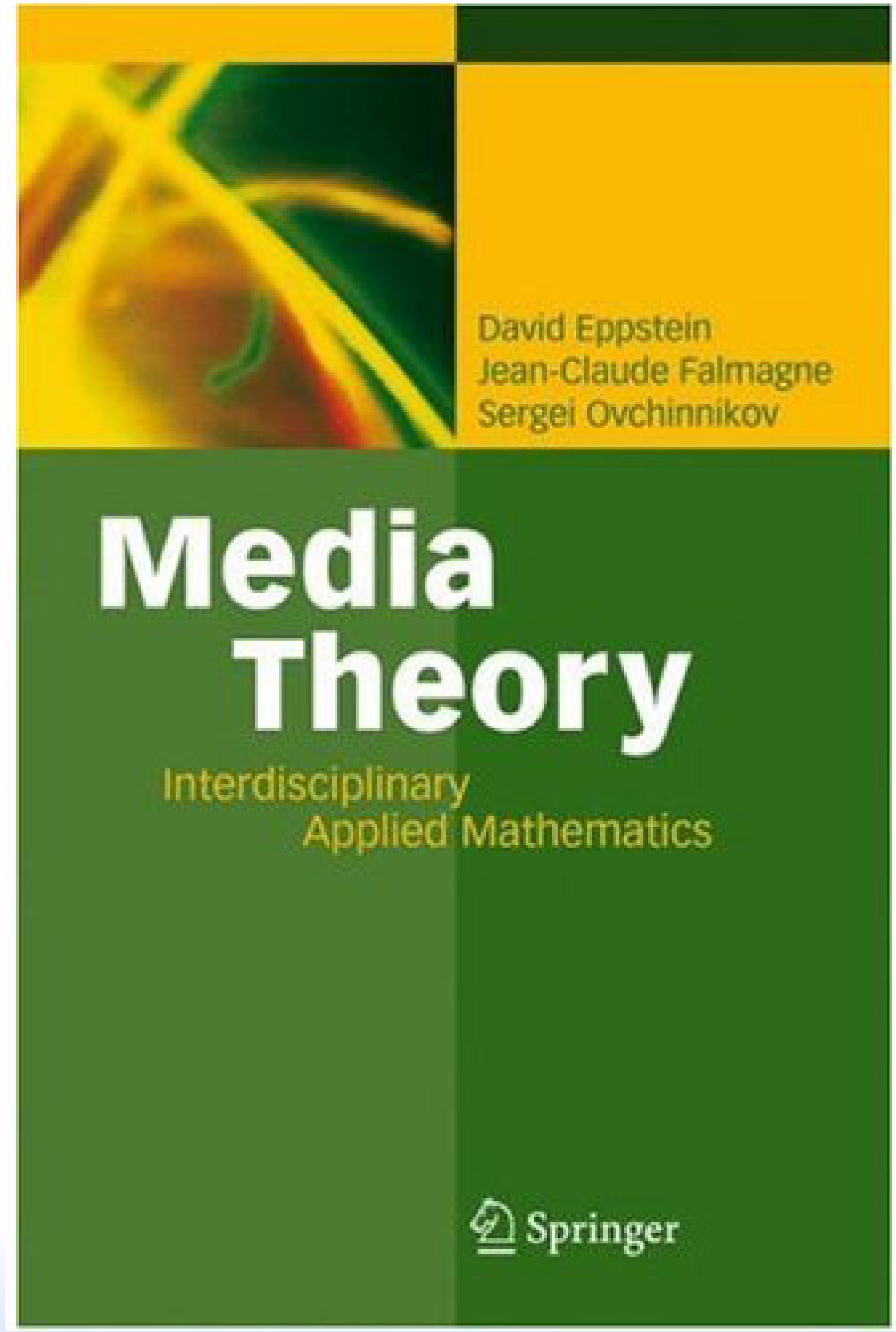
Important open problem in
algorithms: compute flip distance

Flip graph is a partial cube
iff no empty pentagon,
polynomial time in this case





For more applications...



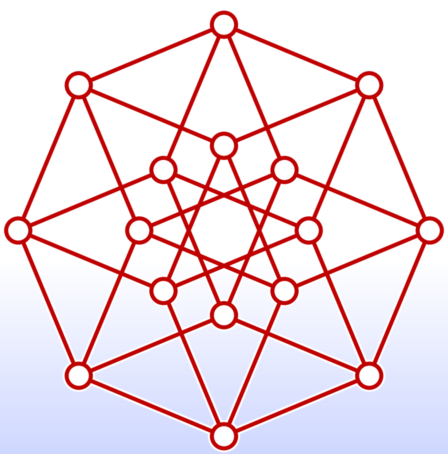
Algorithmic problem: efficiently recognize partial cubes

Given as input an undirected graph,
produce as output a labeling, and check that the labeling preserves distances

Known: $O(nm)$ time [Aurenhammer and Hagauer, 1995]

Note that $O(nm)$ is $O(n^2 \log n)$ because partial cubes have $O(n \log n)$ edges

Lower bound: output may have $\Omega(n^2)$ bits (e.g. when input is a tree)

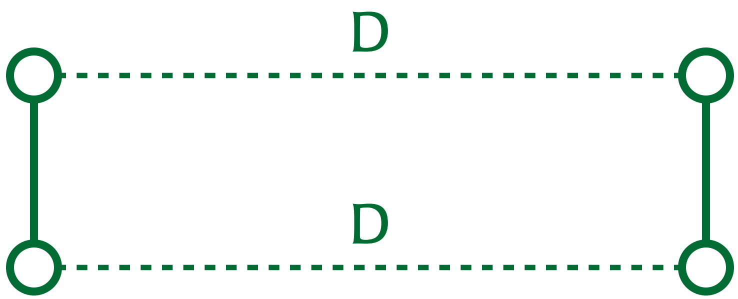


New result: **$O(n^2)$ time**

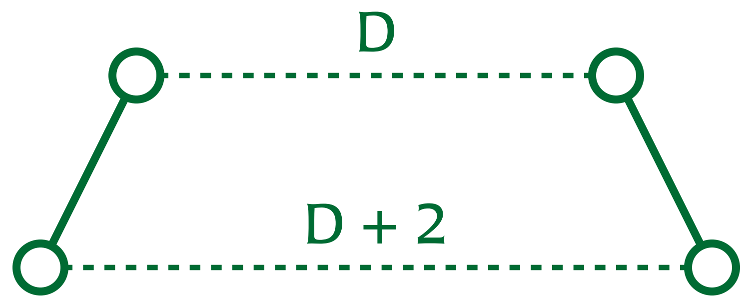
Graph-theoretic characterization

Djokovic–Winkler relation on graph edges [Djokovic 1973, Winkler 1984]:

(p,q) ~ (r,s) iff
d(p,r) + d(q,s) ≠ d(p,s) + d(q,r)



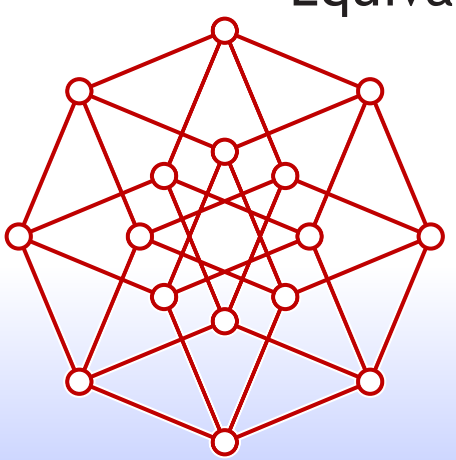
related edges



unrelated edges

G is a partial cube iff it is bipartite and DW-relation is an equivalence relation

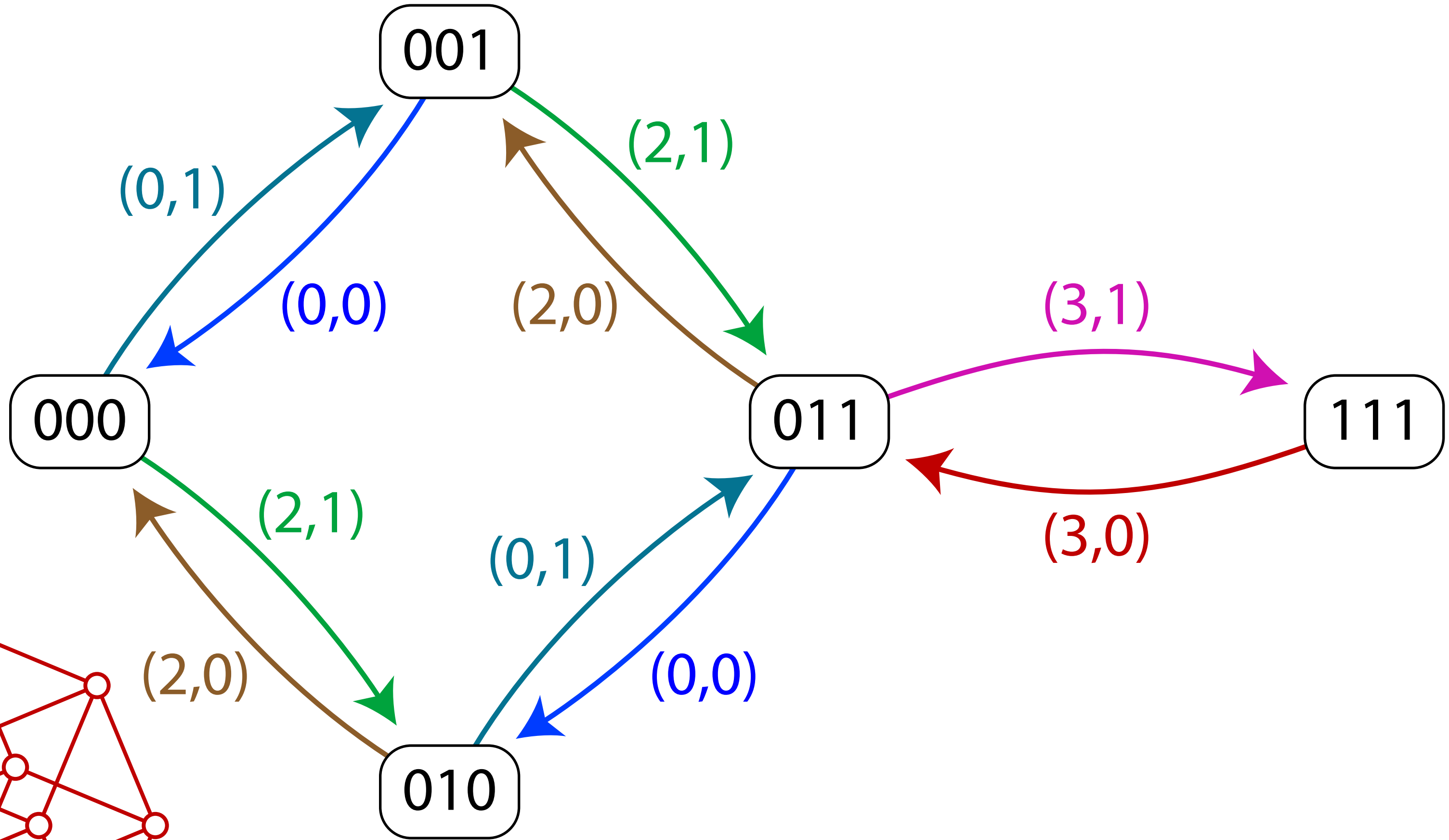
Equivalence classes cut graph into two connected subgraphs

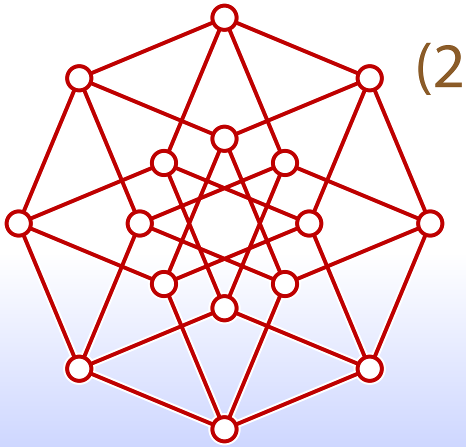


0-1 lattice embedding: coordinate per class,
0 in one subgraph, 1 in the other
unique up to hypercube symmetries

Partial cube as finite state machine

Input token (i,j): set ith bit to j, if possible
otherwise, leave state unchanged





Automaton-theoretic characterization

Medium [e.g. Falmagne and Ovchinnikov 2002]:

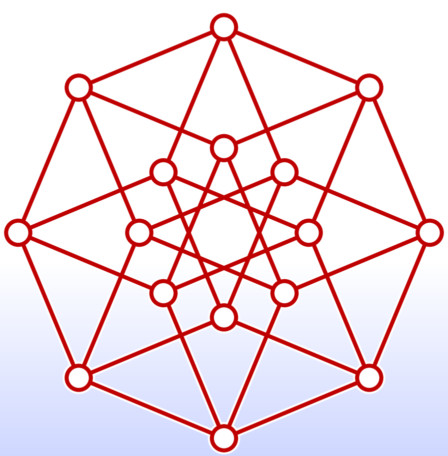
System of states and transformations of states ("tokens")

Every token τ has a "reverse" τ^R:
for any two states $S \neq V$, $S\tau = V$ iff $V\tau^R = S$

Any two states can be connected by a "concise message":
sequence of at most one from each token-reverse pair

If a sequence of effective tokens returns a state to itself
then its tokens can be matched into token-reverse pairs

States and adjacencies between states form vertices and edges of a partial cube



Fundamental components of a partial cube

Vertices and **edges**,
as in any graph, but also:

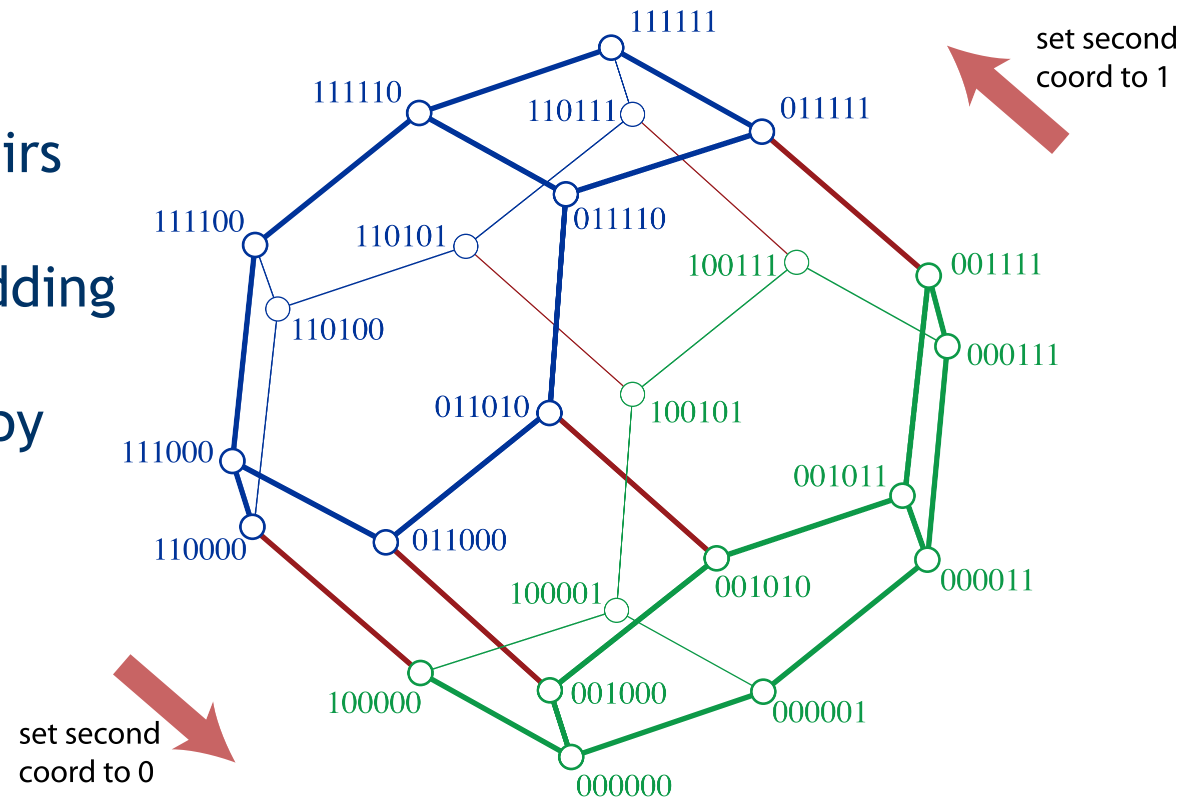
equivalence classes of DW-relation ("zones")

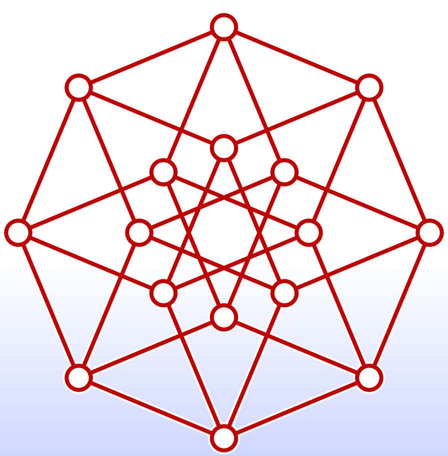
alternatively:

tokens or token-reverse pairs

coordinates of cube embedding

semicubes (subgraphs cut by equivalence classes)





The Algorithm — overall outline

I. Find a labeling
(distance-preserving iff the input is a partial cube)

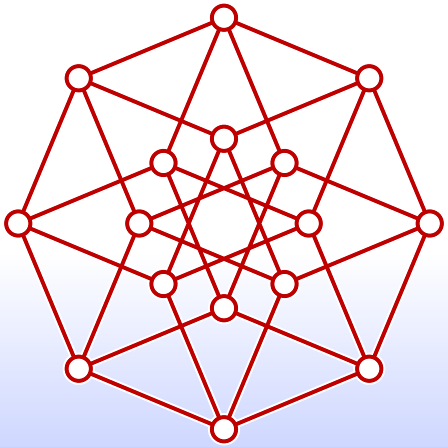
Uses Djokovic–Winkler relation

Sped up by bit-parallel programming techniques

II. Check whether it's distance-preserving

Based on fast all-pairs shortest path algorithm for media

Uses media-theoretic characterization



The Algorithm — finding a labeling

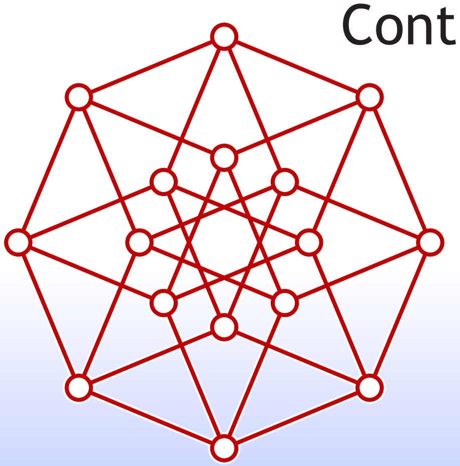
Perform a breadth first search from a high-degree root vertex

Label each node by a bitvector
Indicating which neighbors of root it can connect through

Label edge by exclusive or of endpoint labels
(should be either zero or single bit)

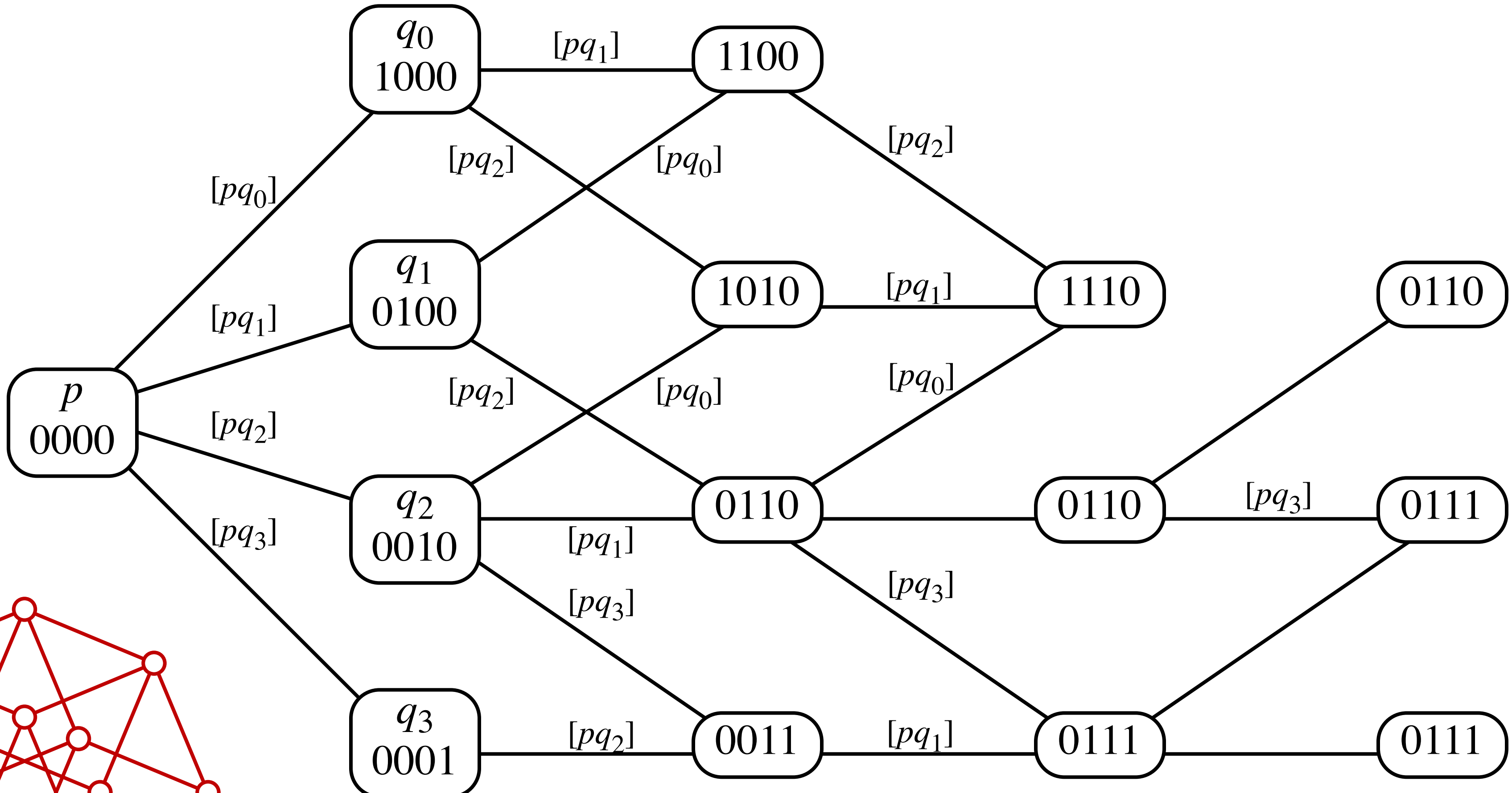
Sets of edges with same nonzero labels
= Djokovic-Winkler classes

Contract labeled edges and continue in remaining graph



The Algorithm — finding a labeling

Example:



The Algorithm — checking the labeling

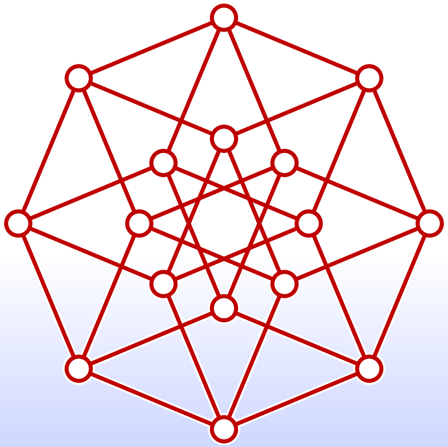
Perform a depth-first traversal of the graph, maintaining:

- a list of tokens on shortest paths to the current vertex (one token from each token-reverse pair)
- for each other vertex, the first effective token on the list

When the depth-first traversal moves to another vertex:

- remove the corresponding token from the list, and add its reverse to the end of the list
- for each vertex pointing to the removed token, search forwards for the next effective token

If the search runs off the end of the list, the graph is not a partial cube



The Algorithm — analysis

I. Finding the labeling

Search from degree d vertex finds d ≥ m/n tokens
using O(m) bitvector operations
taking time O(1 + d/log n) per bitvector operation

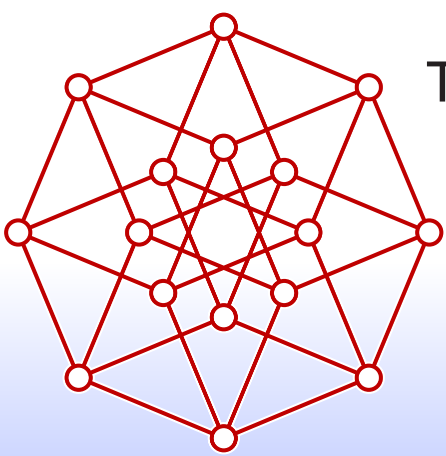
Total per token: O(m/d + m/log n) = O(n)

Whole graph has O(n) tokens, so $O(n^2)$ total

II. Checking whether it’s distance-preserving

Total number of tokens added to end of list: O(n)

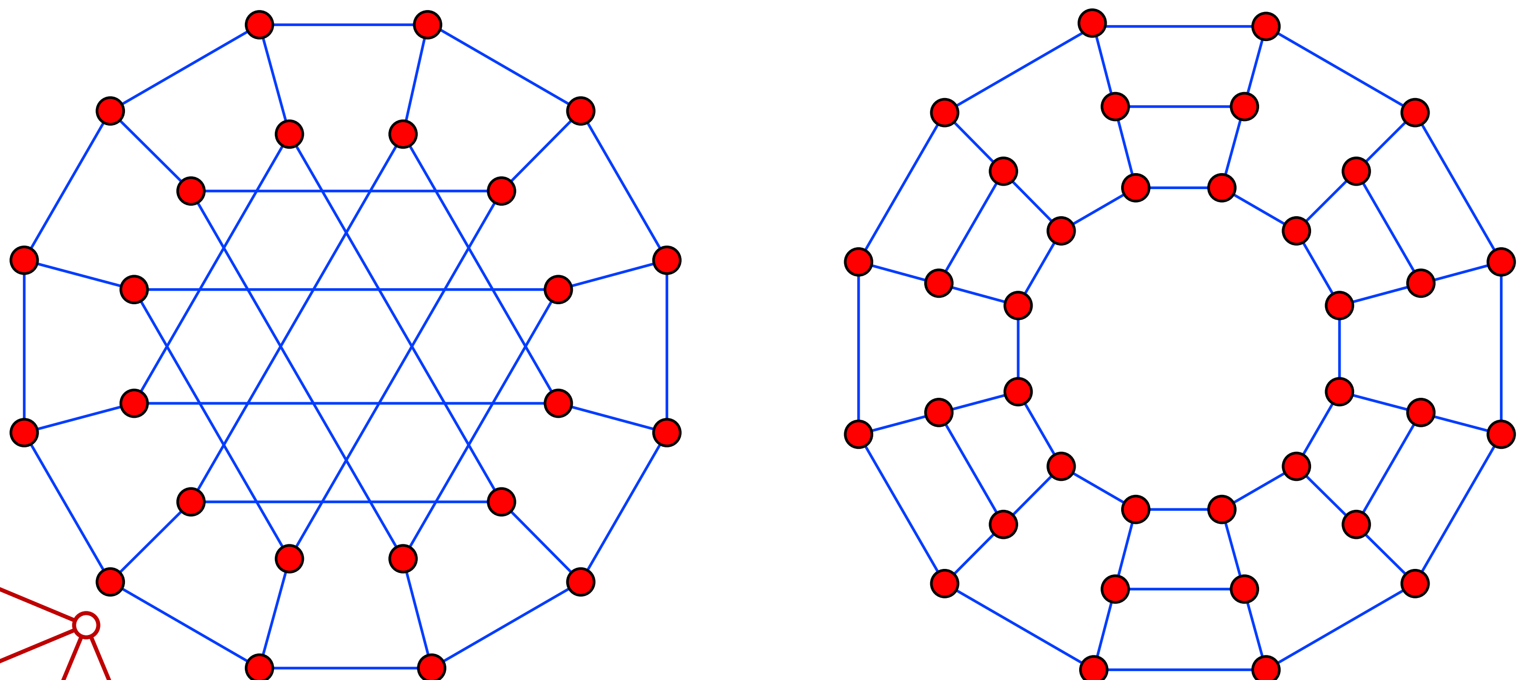
Each node scans list once, so $O(n^2)$ total



The Algorithm — implementation

220 lines of Python
(approximately 1/3 of which are unit tests)
http://www.ics.uci.edu/~eppstein/PADS/PartialCube.py

Two problematic graphs
(minor bugs in implementation, both fixed, no change to algorithm):



left: crashed the program
right: incorrectly reported as a partial cube