Learning Circuits and Networks by Injecting Values

Lev Reyzin
Yale University

Papers

- Dana Angluin, James Aspnes, Jiang Chen, and Yinghua Wu. **Learning a Circuit by Injecting Values.** In *Proceedings of the 38th ACM Symposium on Theory of Computing* (STOC), May 2006

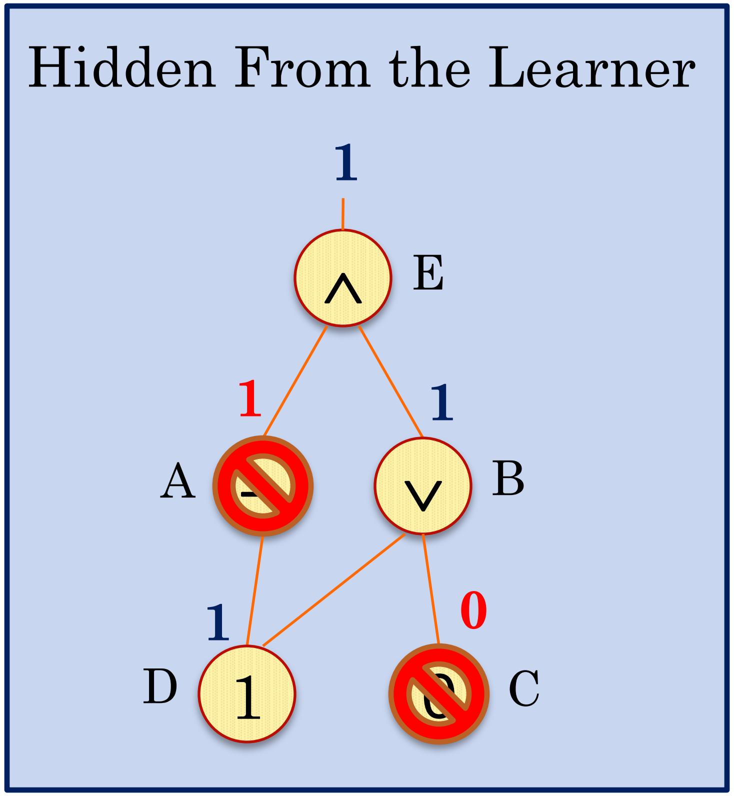
- Dana Angluin, James Aspnes, Jiang Chen, and Lev Reyzin. **Learning Large-Alphabet and Analog Circuits with Value Injection Queries.** In *Proceedings of the 20th Annual Conference on Learning Theory* (COLT), June 2007

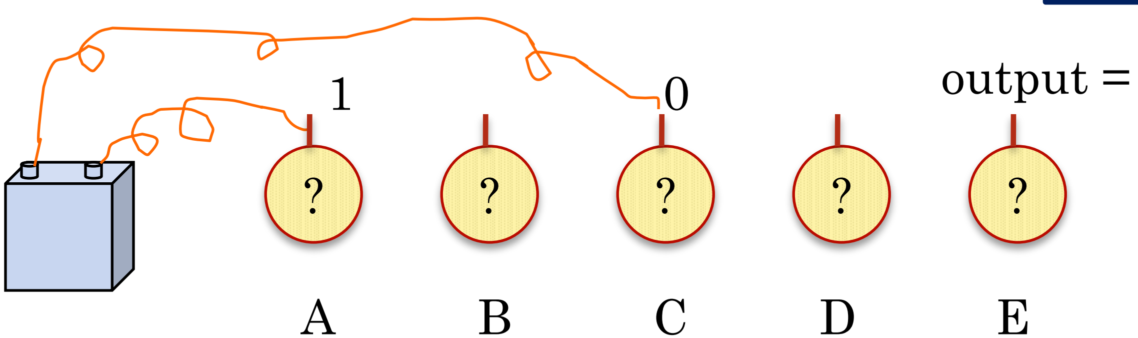
- Dana Angluin, James Aspnes, Jiang Chen, David Eisenstat, and Lev Reyzin. **Learning Acyclic Probabilistic Circuits Using Test Paths.** To appear in *Proceedings of the 21st Annual Conference on Learning Theory* (COLT), July 2008

- Dana Angluin, James Aspnes, and Lev Reyzin. **Optimally Learning Social Networks with Activations and Suppressions.** *In preparation*, February 2008

The Value Injection Query Model

- Introduced by [AACW '06]
- Experiments on a hidden Circuit.
  - a gate output may be fixed
  - a gate may be left free
- Query
  - given an experiment, we can observe its output
- Example:

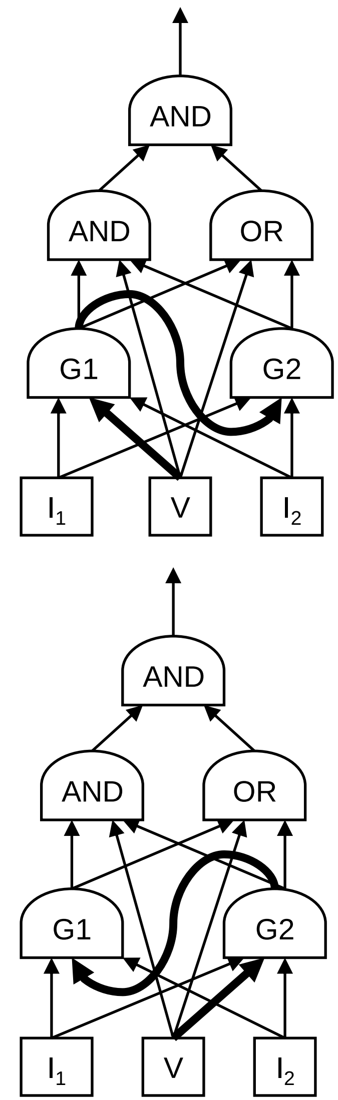




THE LEARNING PROBLEM

- **Behavioral equivalence:** Two circuits C and C' are behaviorally equivalent if for any experiment s, C(s)=C'(s).

- **The Problem:** Given qeuery access to a hidden circuit C*, find a circuit C behaviorally equivalent to C* by making value-injection queries.



[ACCW '06]

Motivation for The Model

- To model gene regulatory networks as boolean networks
- to represent gene expressions and disruptions

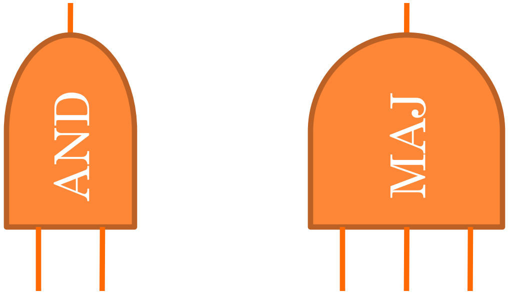
Previous gene regulatory network model	Fully controllable.	All gates are observable.
Existing circuit learning models	Only inputs can be manipulated.	Only the output is observable.
[AACW ’06] model	Fully controllable.	Only the output is observable.

[AACW '06] Results for Boolean Circuits

Depth	**Fan-in**	**Gates**	**Learnability**
Unbounded	Unbounded	AND/OR	$2^{\Omega(N)}$ queries
Unbounded	2	AND/OR	NP-hard
Constant	Unbounded	AND/OR/Θ_2	NP-hard
Log	Constant	Arbitrary	Poly-time **(NC1)**
Constant	Unbounded	AND/OR/NOT	Poly-time **(AC0)**

Large-Alphabet Circuits

Gates in Boolean Circuits



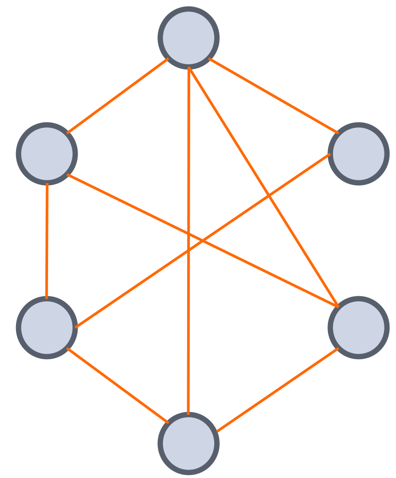
Input 1	Input 2	Output
1	1	O_1
1	0	O_2
0	1	O_3
0	0	O_4

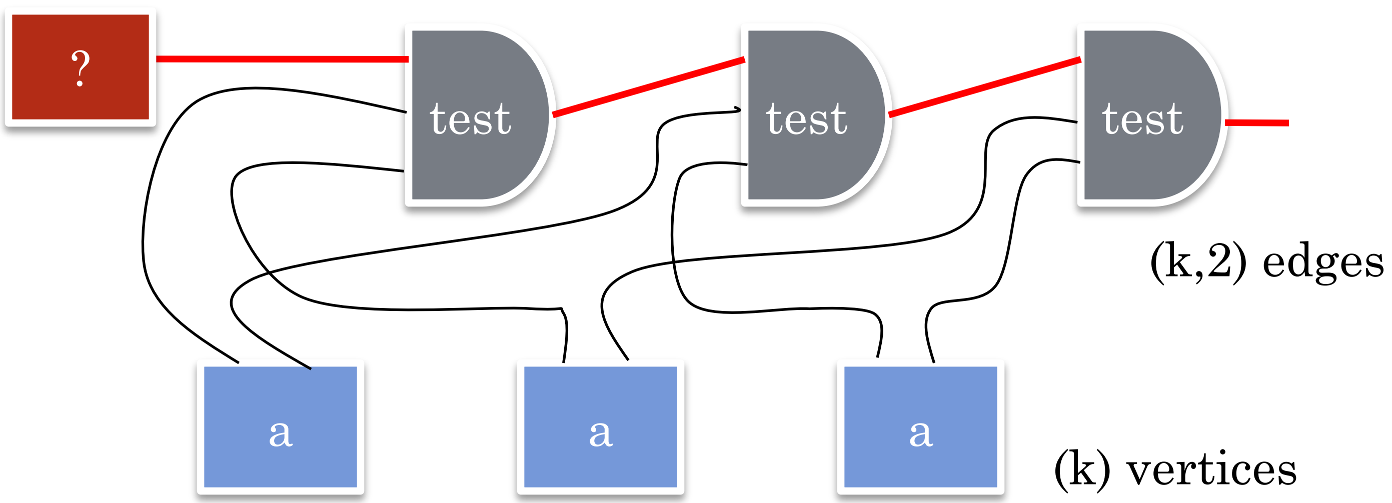
Gates in Large-Alphabet circuits

Input 1	Input 2	Output
A	A	O_1
A	B	O_2
A	C	O_3
B	A	O_4
B	B	O_5
B	C	O_6
C	A	O_7
C	B	O_8
C	C	O_9

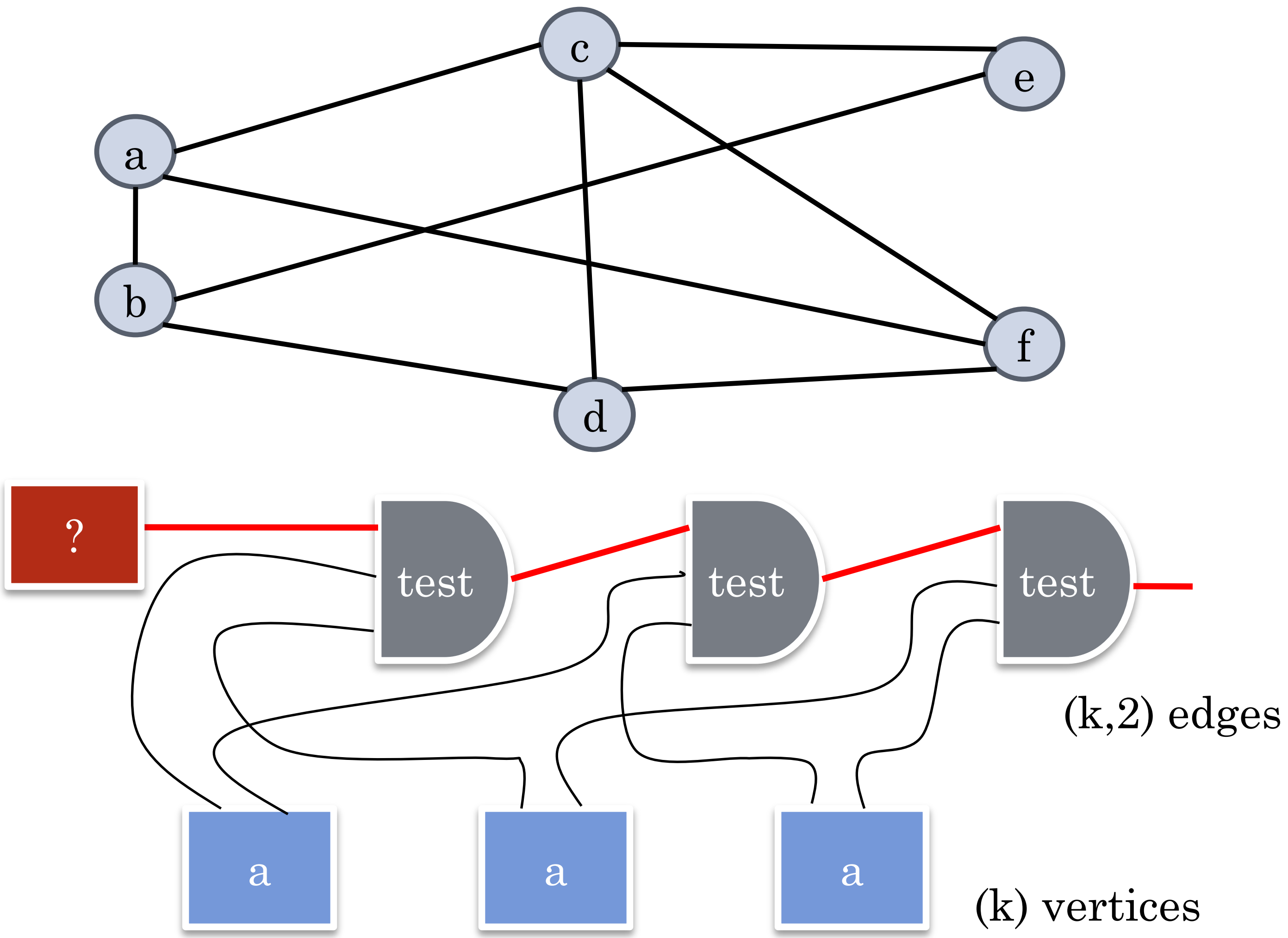
Hardness of Learning Large Alphabet Circuits

- Consider the problem on input (G,k) of telling whether the graph G on n vertices has a clique of size k
- We give a reduction that turns a large-alphabet circuit learning algorithm into a clique tester

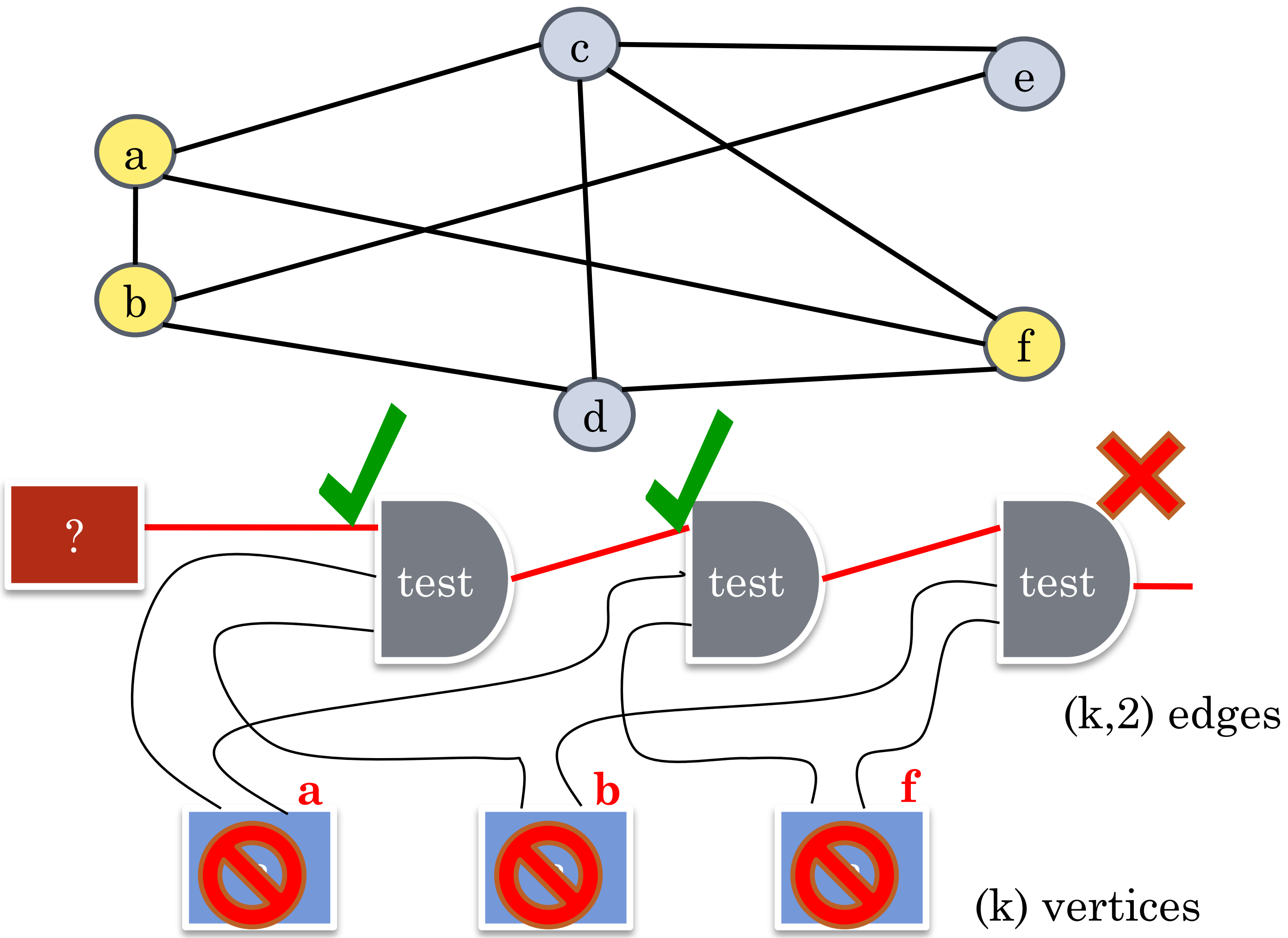




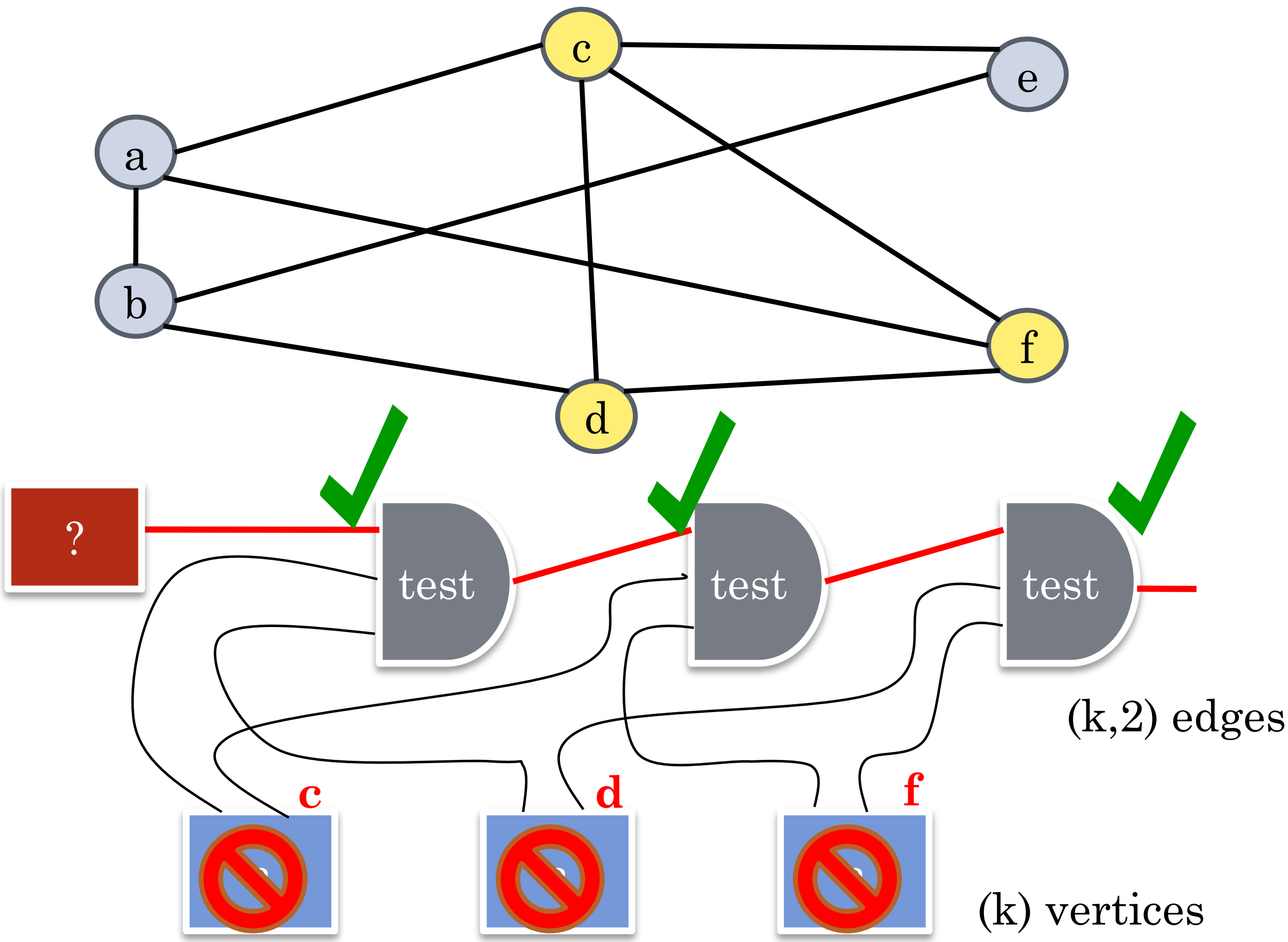
Reducing the Clique Problem to Circuit Learning



Reducing the Clique Problem to Circuit Learning



Reducing the Clique Problem to Circuit Learning



Hardness of Learning Circuits of Unrestricted Topology

- The clique problem is complete for the **parameterized complexity class** W[1]
  - There is no known algorithm for the clique problem that runs in time $f(k)n^c$ (and we believe one doesn't exist)

- **Theorem** **An algorithm for learning circuits polynomial in the number of wires and alphabet size would imply fixed parameter tractability for all problems in W[1]**

To Compare with the Boolean Case

Boolean Circuits [AACW '06]:

Depth	Fan-in	Gates	Learnability
Log	Constant	Arbitrary	Poly-time

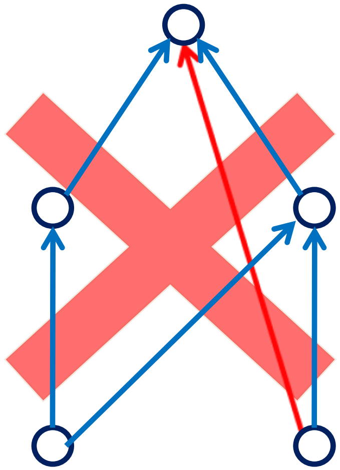
Large Alphabet Circuits:

Depth	Fan-in	Gates	Learnability
Log	Constant	Arbitrary	W[1] Hard

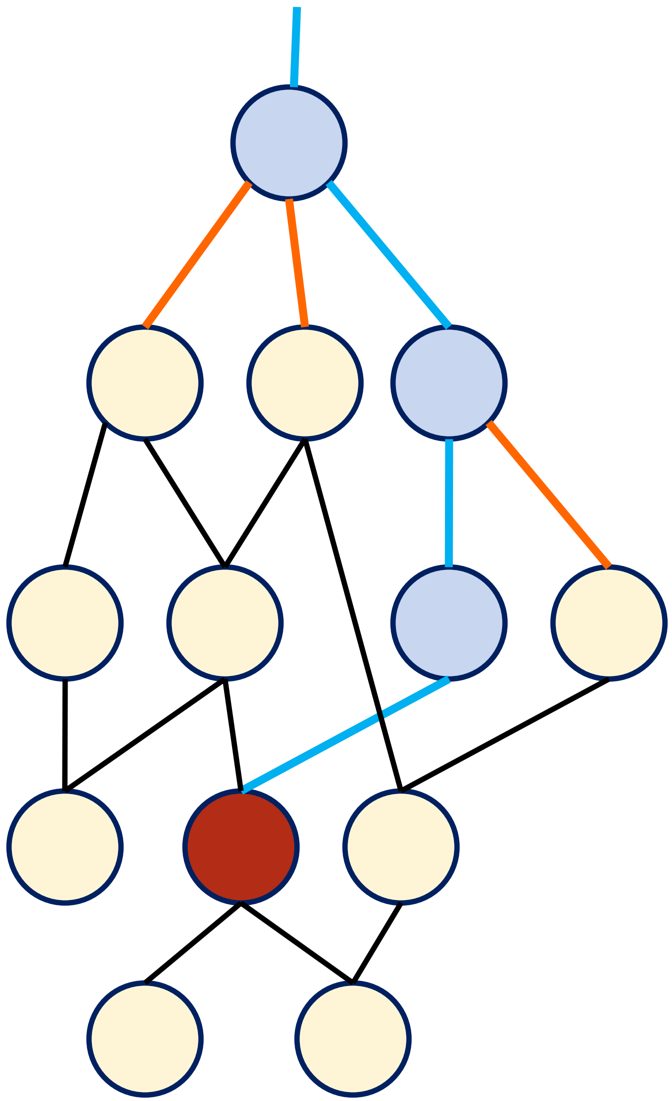
This motivates looking at classes of large-alphabet circuits with restricted topology

TRANSITIVELY REDUCED CIRCUITS

A circuit is transitively reduced if its underlying directed graph has no shortcuts. If (u,v) is an edge and there is a path of length ≥ 2 from u to v, then (u,v) is a **shortcut edge**



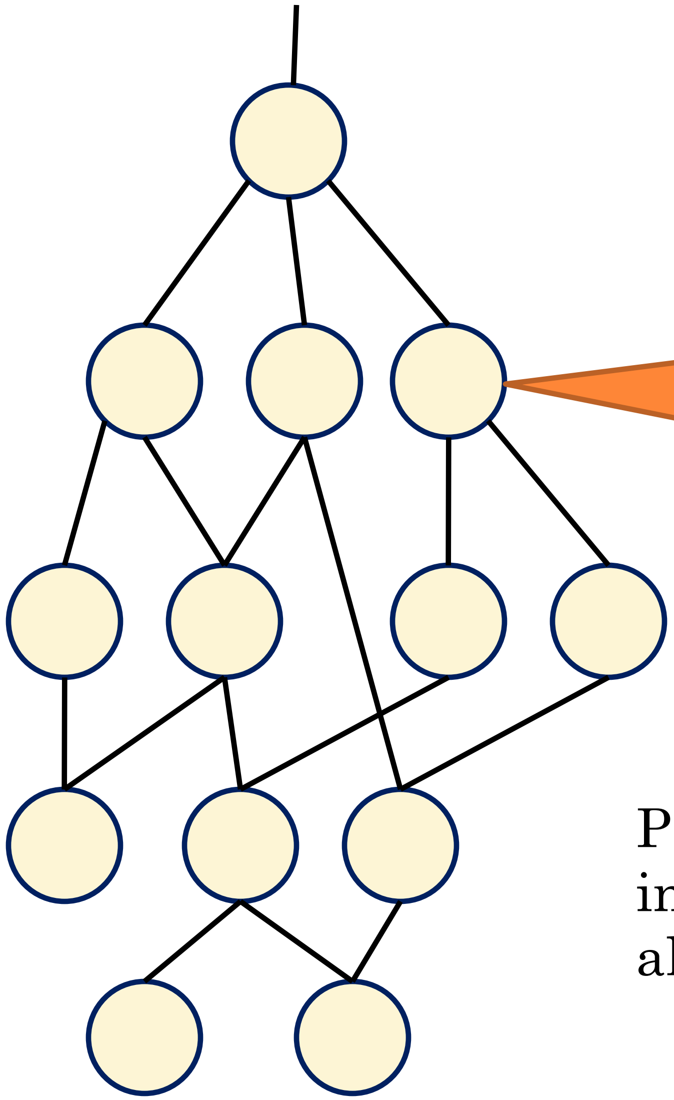
Paths to the Root



Large Alphabet Circuit Result

Theorem **We can learn the class of circuits having n wires, alphabet size s, fan-in bound k, and shortcut width bounded by b, using $ns^{O(k+b)}$ value injection queries and time polynomial in the number of queries.**

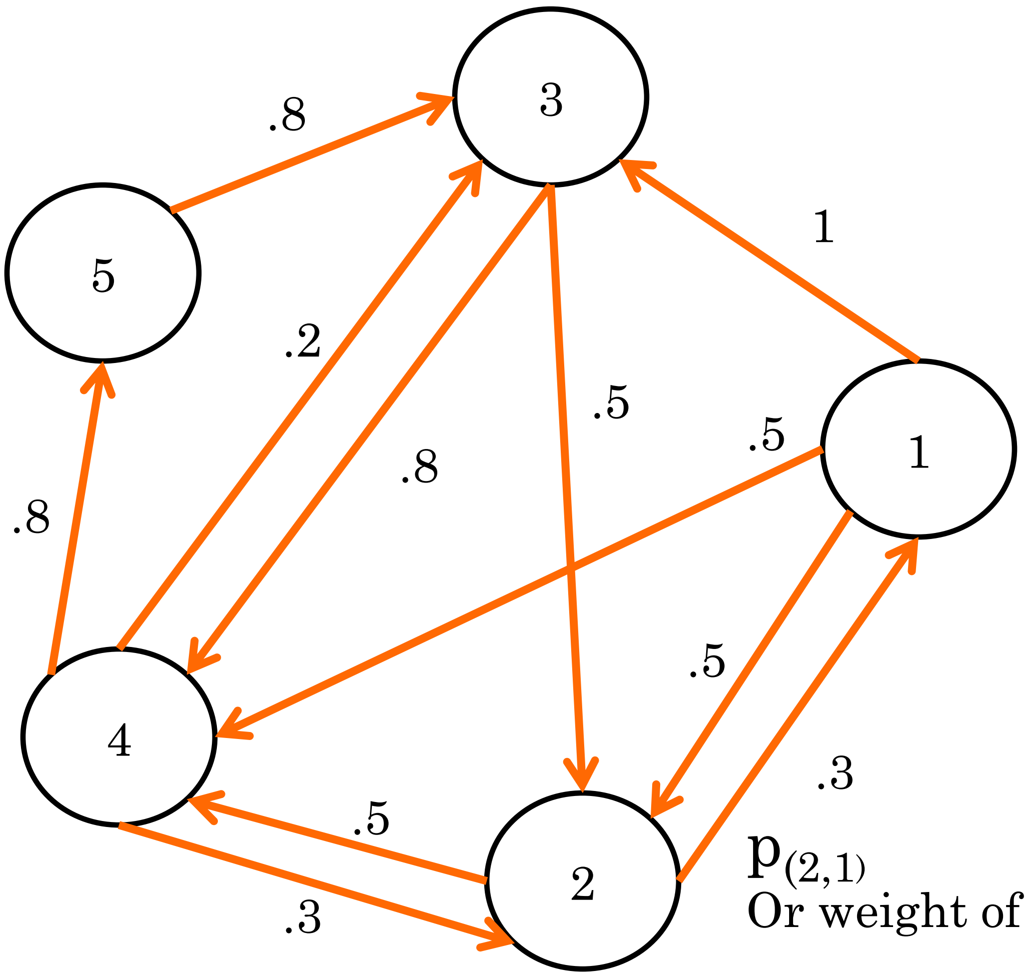
PROBABILISTIC CIRCUITS



	P(0)	P(1)
00	.5	.5
01	1	0
10	.8	.2
11	.3	.7

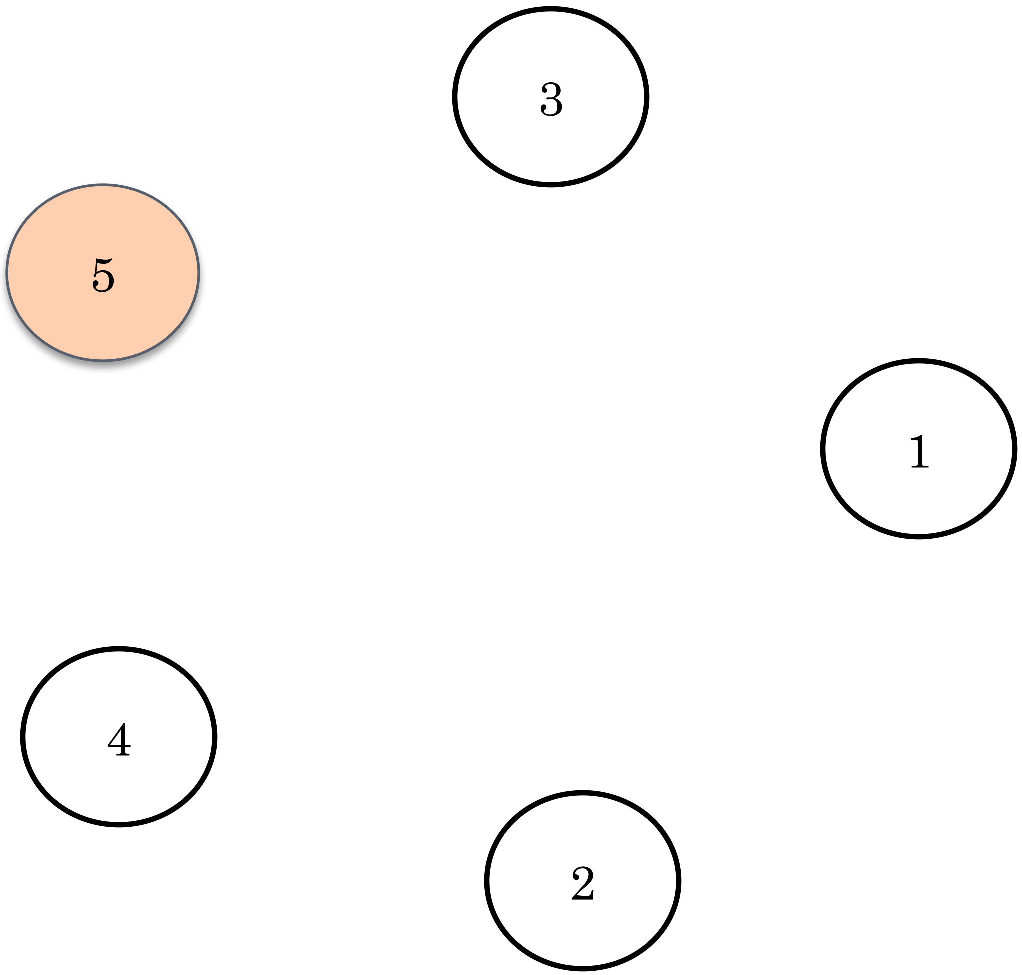
Path-based methods no longer work in the probabilistic case (for large alphabets).

Independent Cascade Social Networks

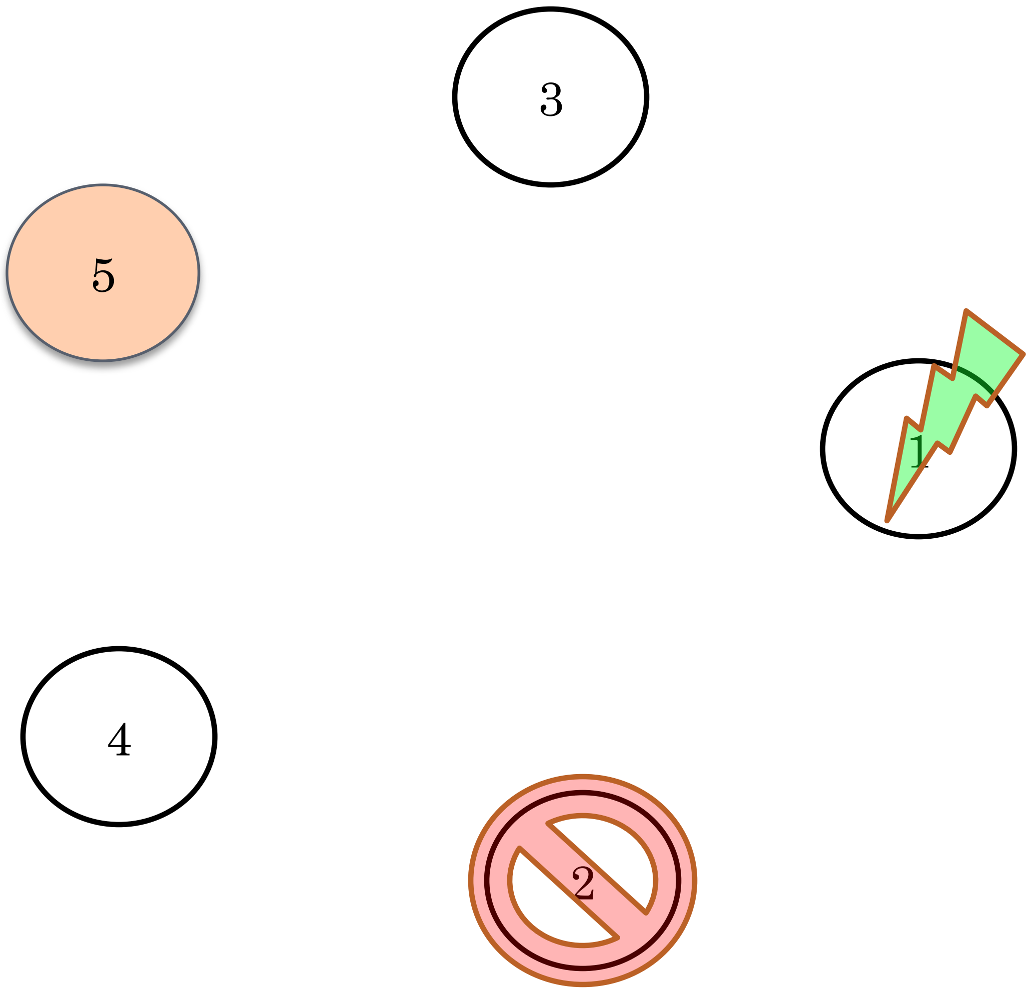


$p_{(2,1)}$
Or weight of edge (2,1)

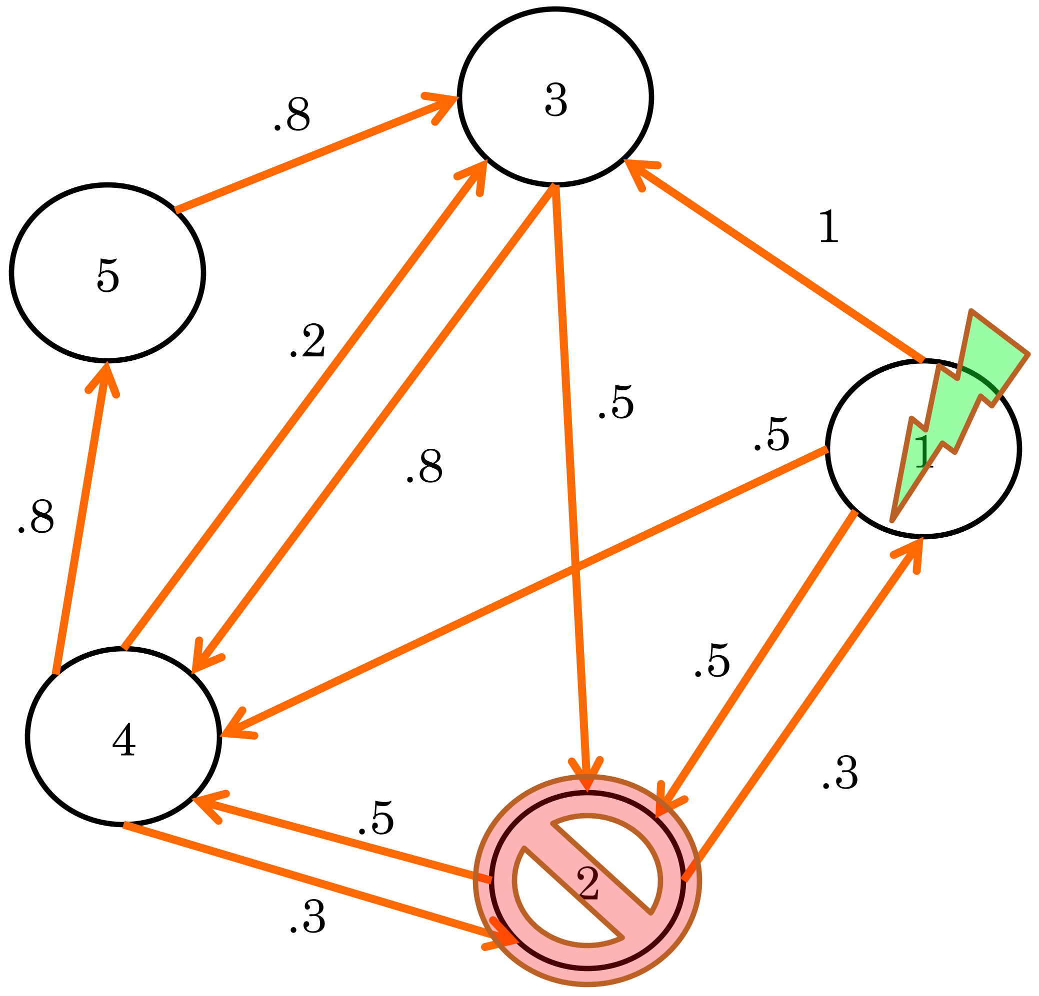
What the Learner Sees



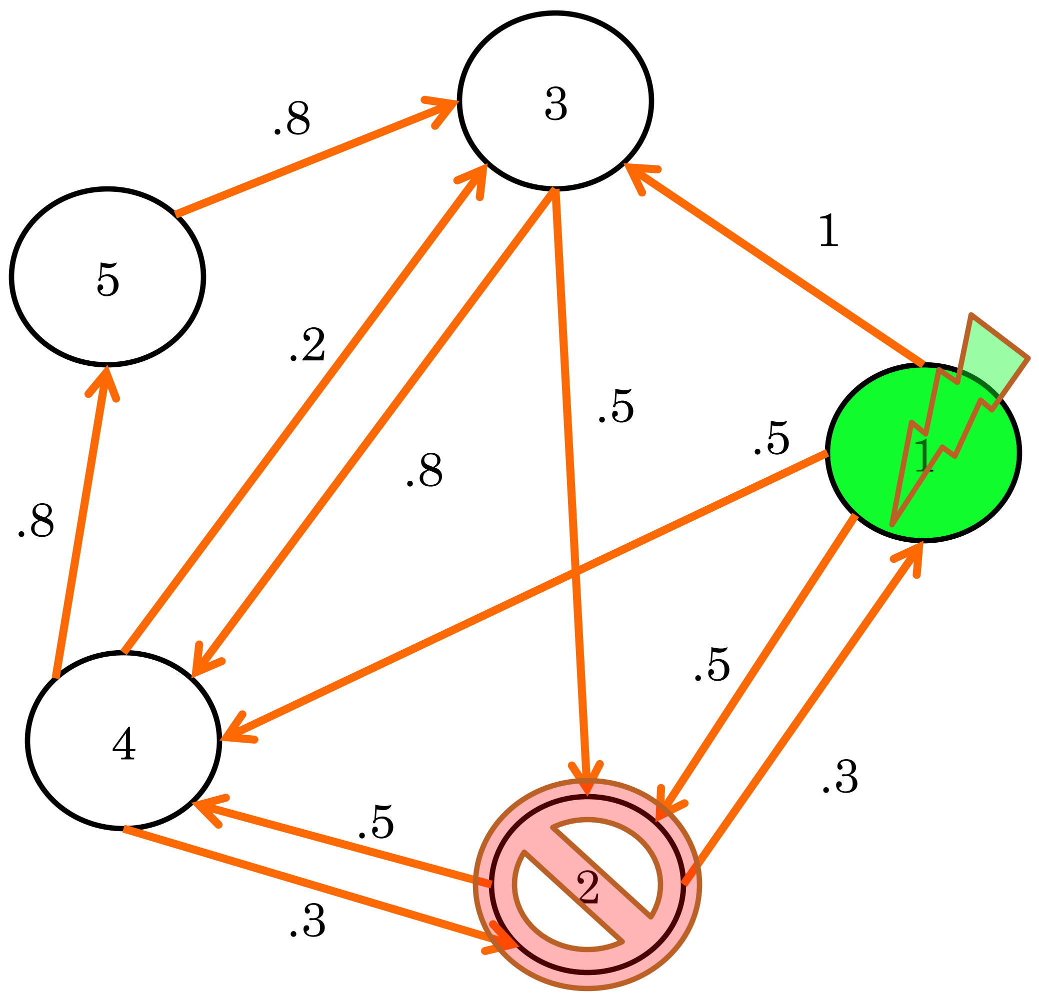
Activations and Suppressions



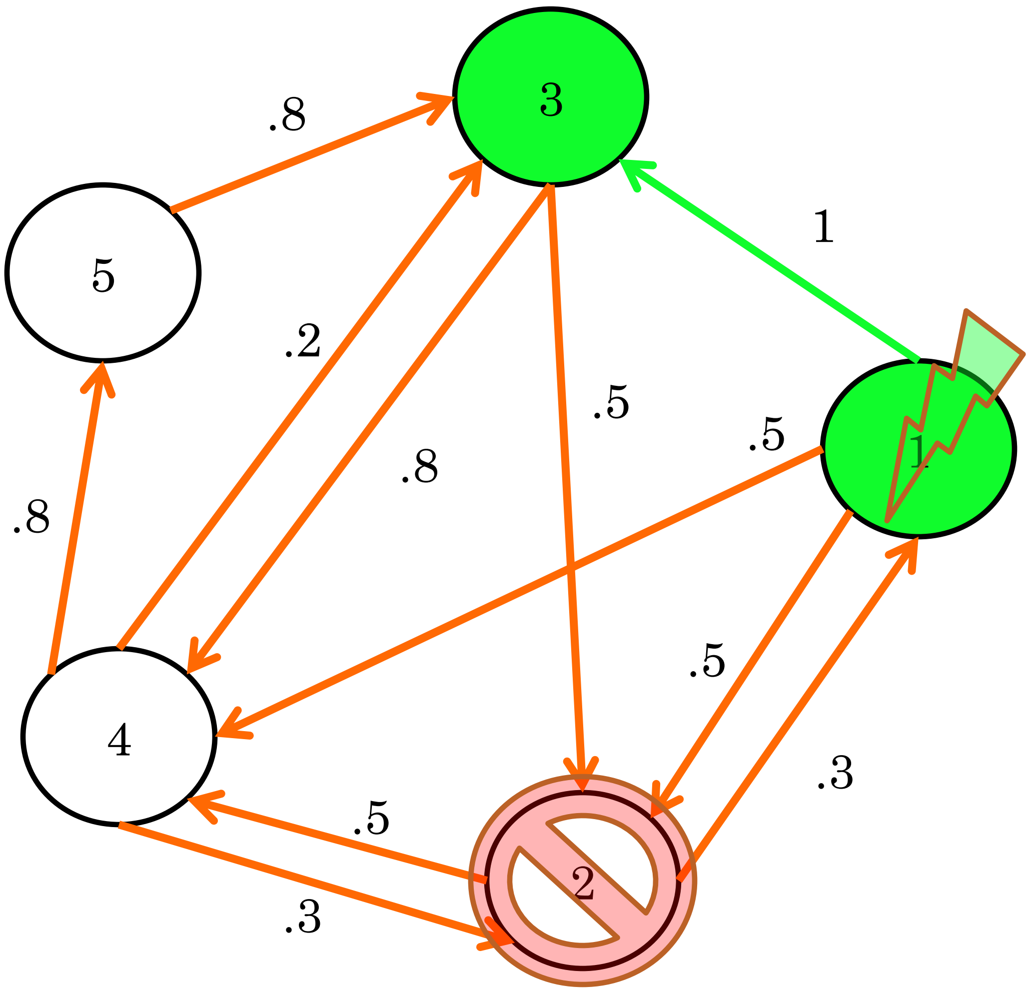
ACTIVATIONS AND SUPPRESSIONS



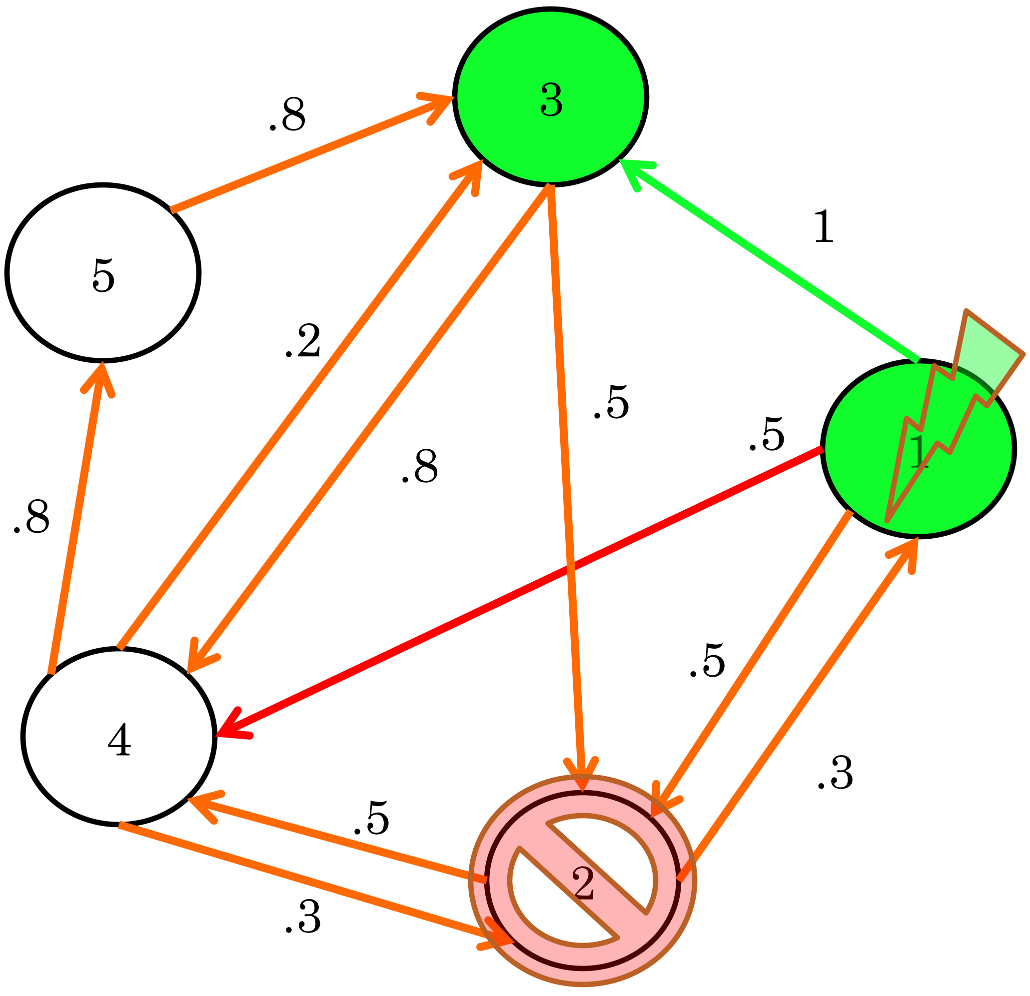
Activations and Suppressions



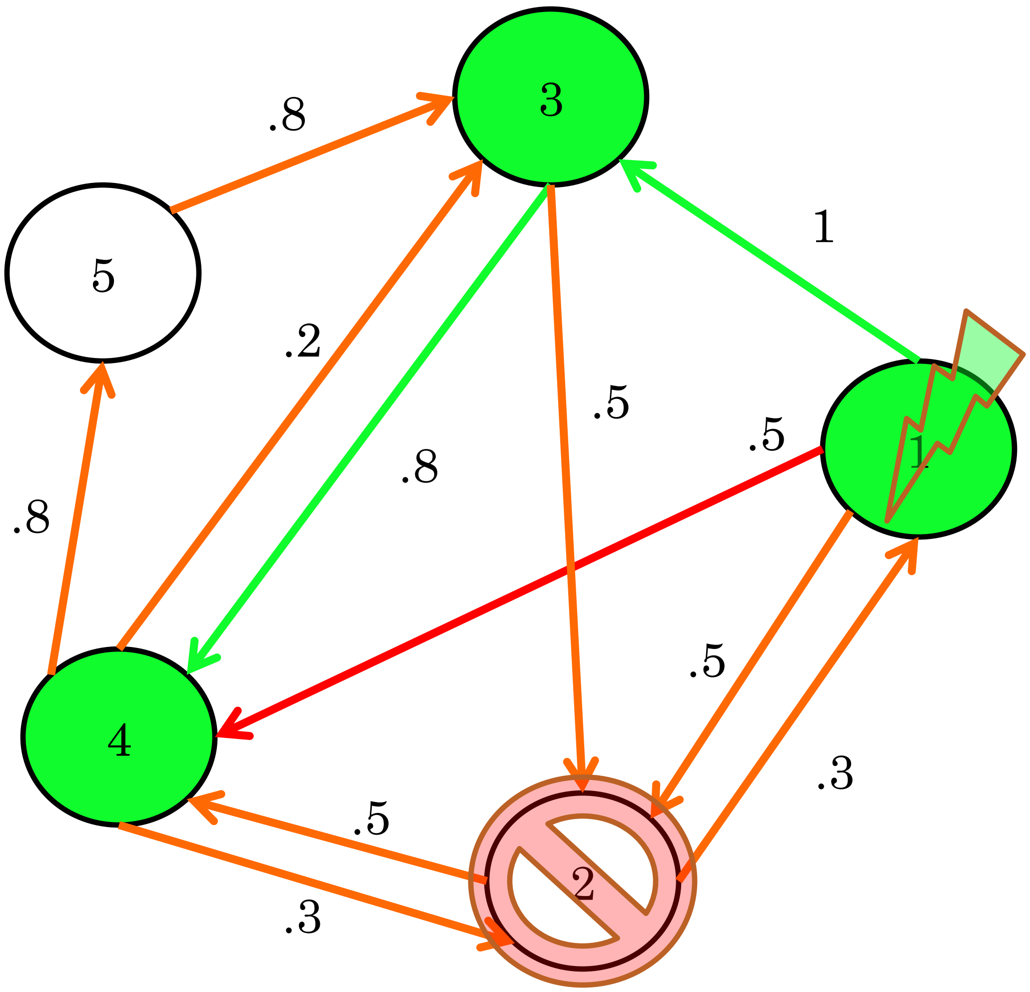
Activations and Suppressions



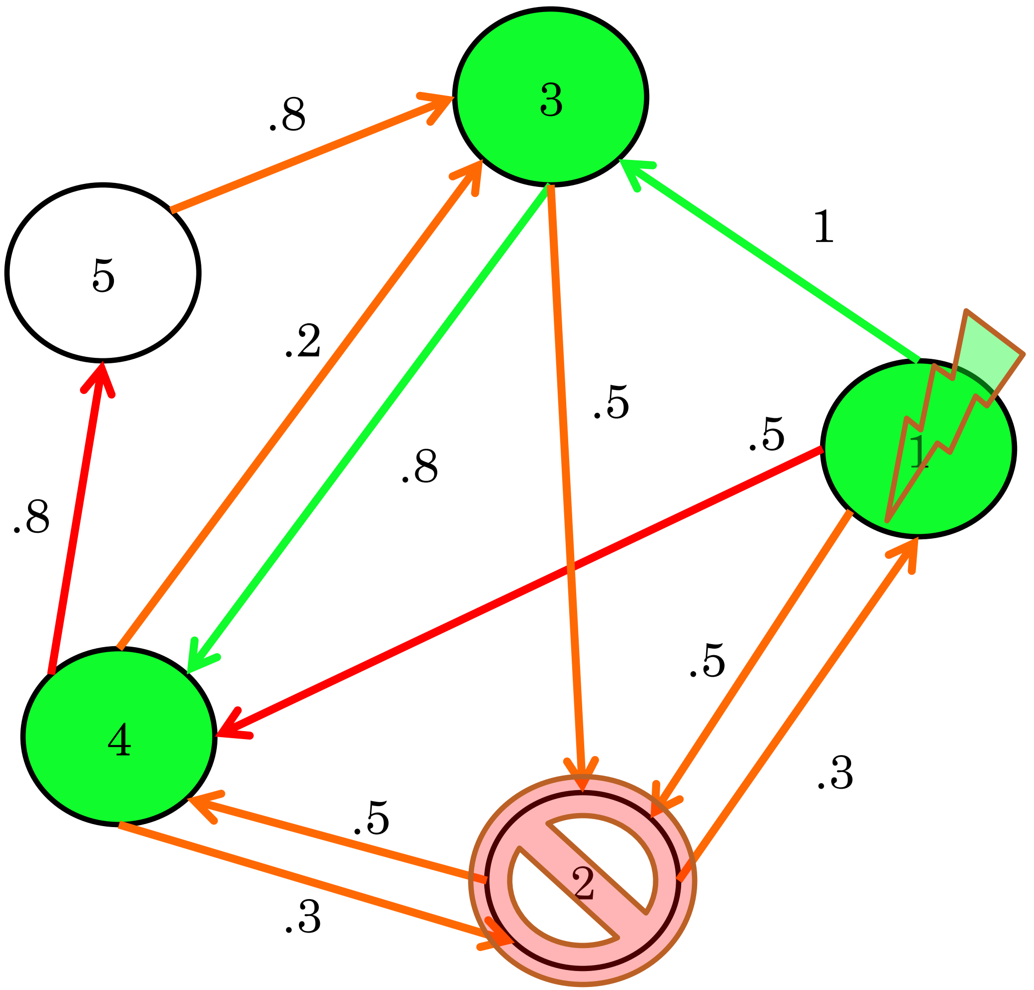
Activations and Suppressions



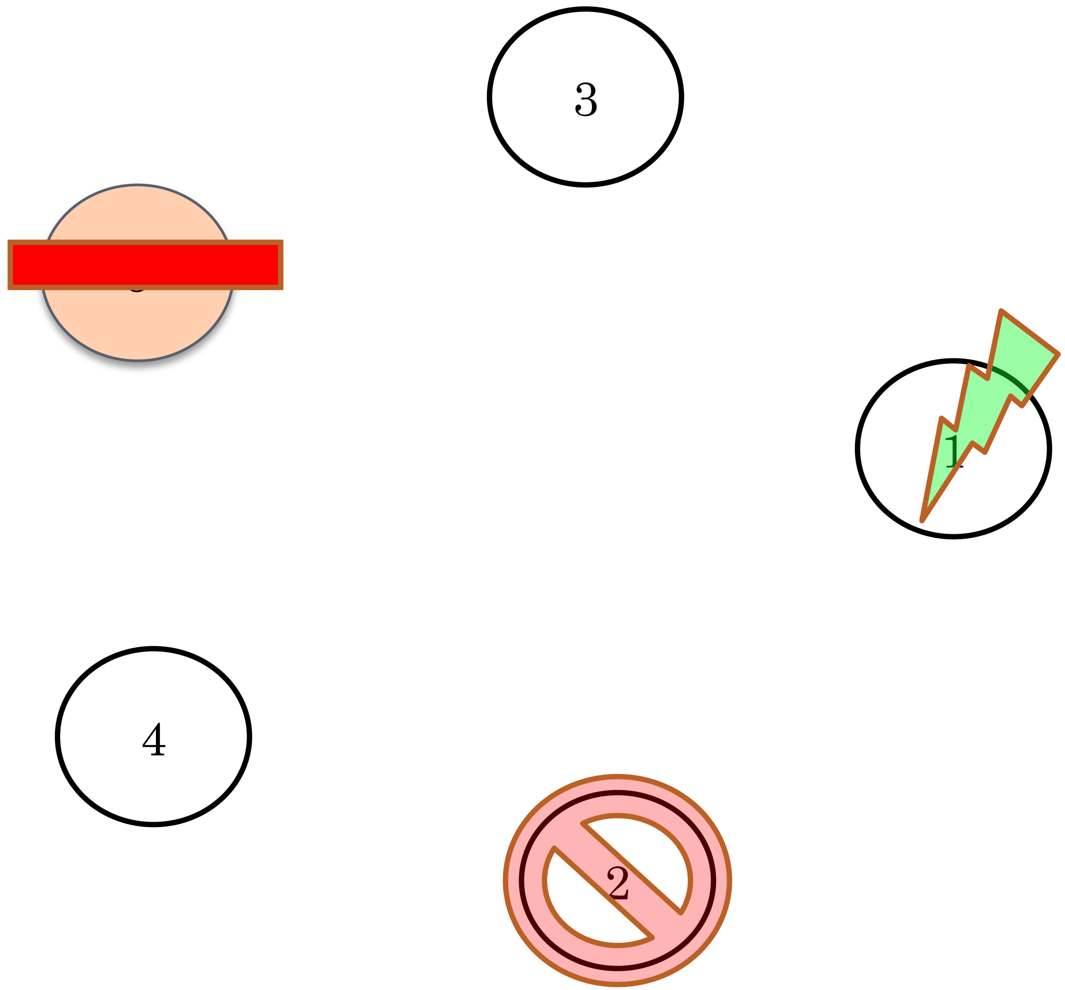
Activations and Suppressions



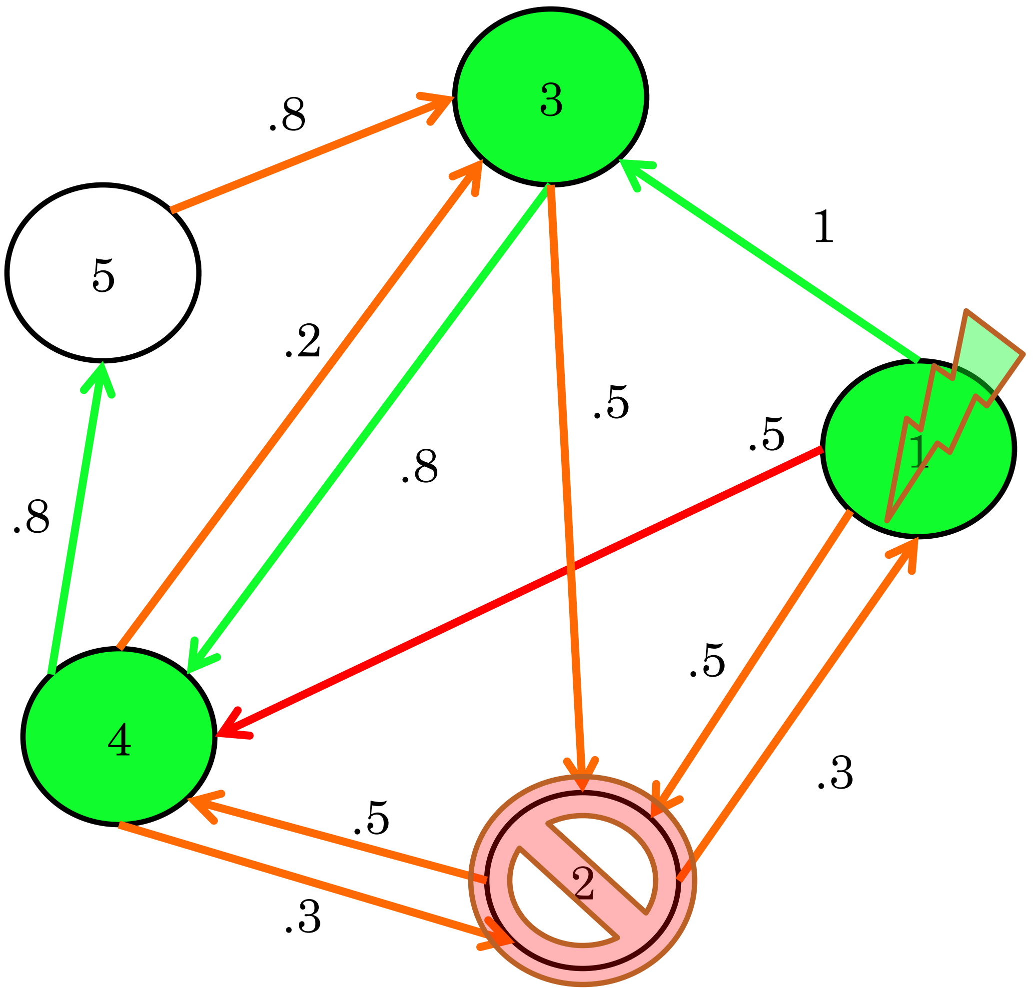
Activations and Suppressions



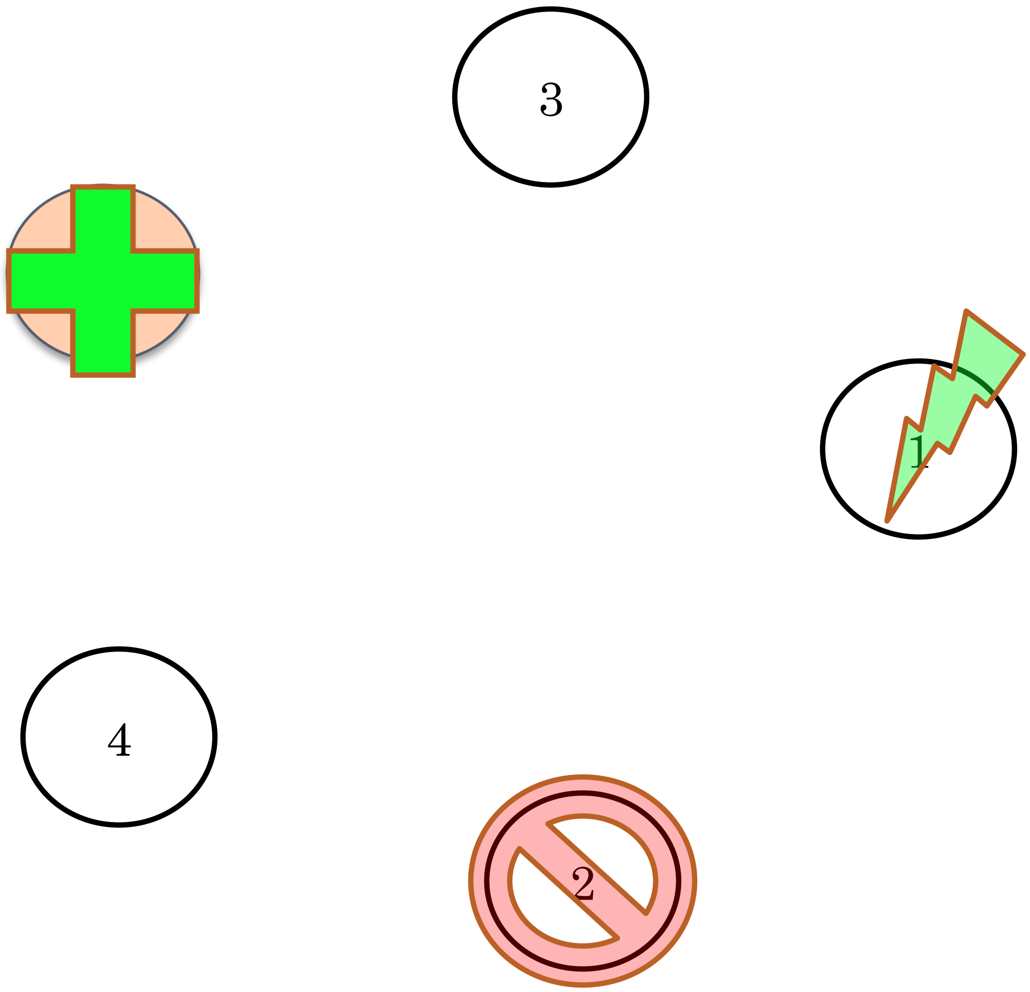
Activations and Suppressions



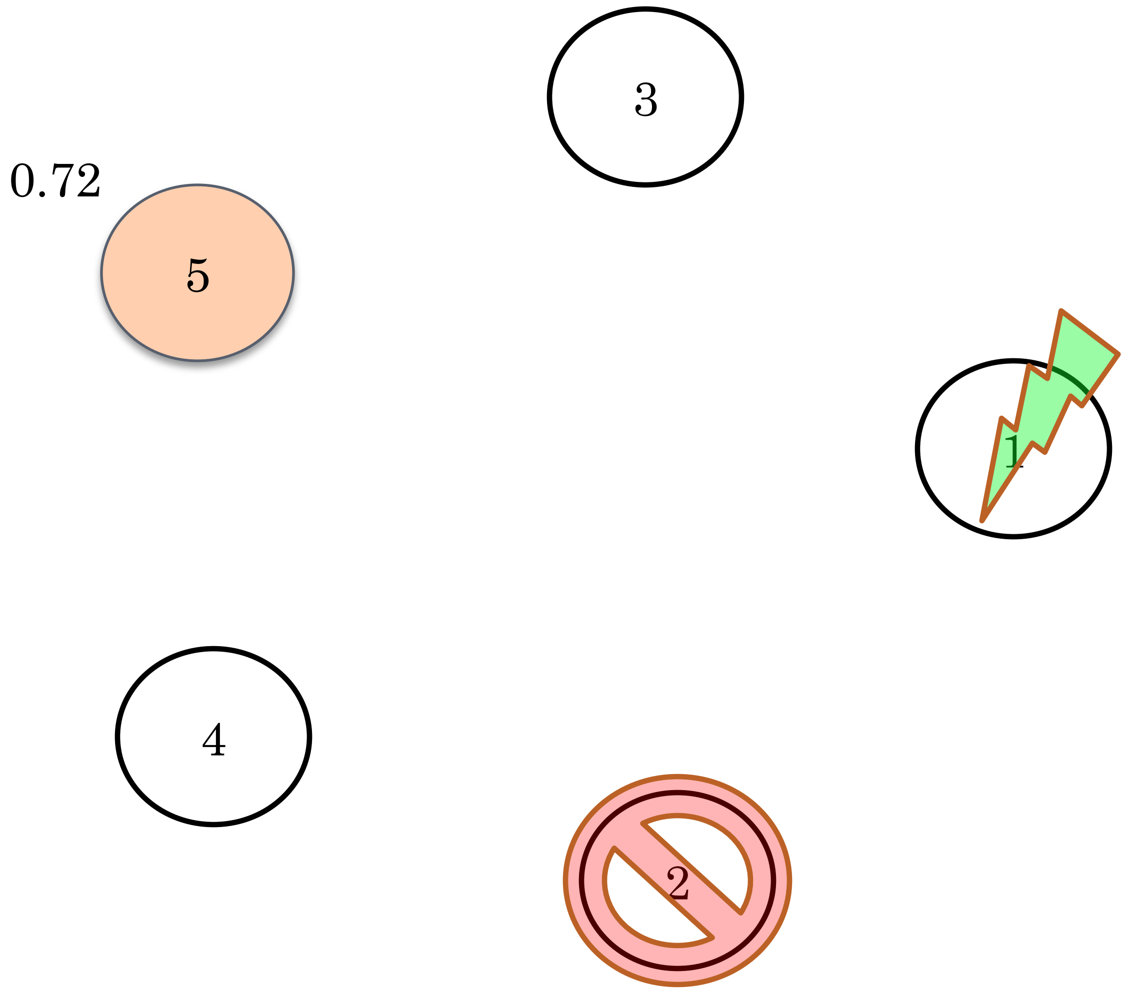
Activations and Suppressions



Activations and Suppressions



Exact Value Injection Queries



The Learning Task (a reminder)

- Two social networks S and S' are behaviorally equivalent if for any experiment e, S(e) = S'(e)
- Give access to a hidden social network S*, the learning problem is to find a social network S behaviorally equivalent to S* using value injection queries.

The Percolation Model

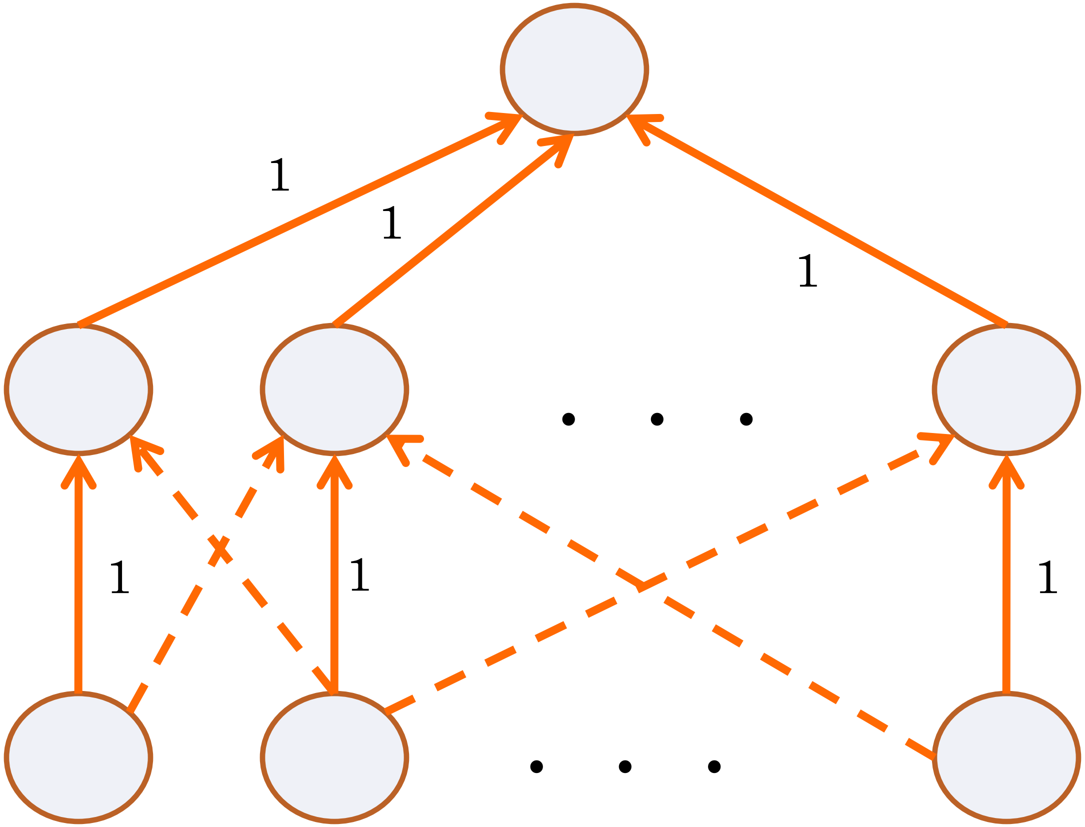
Given a network S and a VIQ

- All edges entering or leaving a suppressed node are automatically "closed."
- Each remaining edge (u,v) is "open" with probability $p_{(u,v)}$ and "closed" with probability (1- $p_{(u,v)}$)
- The result of a VIQ is the probability there is a path from a fired node to the output via open edges in S

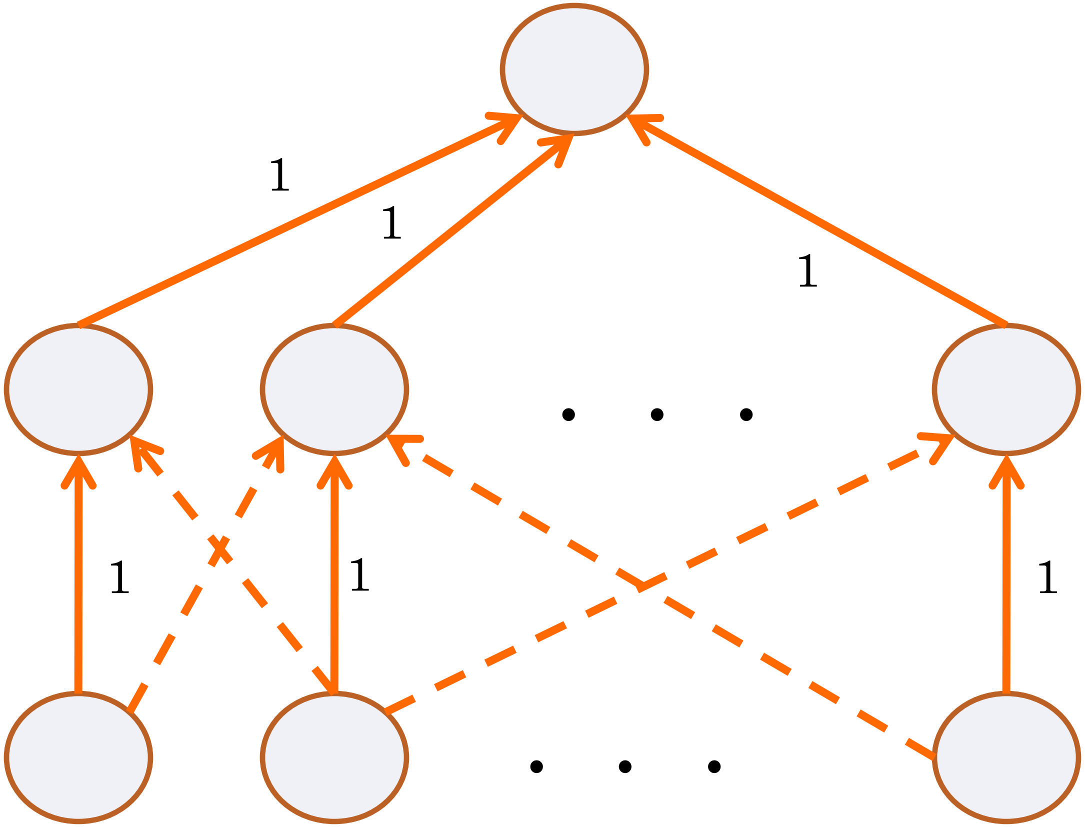
Discoverable Edges

- Let S be a social network and S' be another social network that differs from S only in edge (u,v).
- We say edge (u,v) is discoverable if there is an experiment e such that $S(e) \neq S'(e)$.
- We can view the learning problem as having to find all discoverable edges.

A Lower Bound

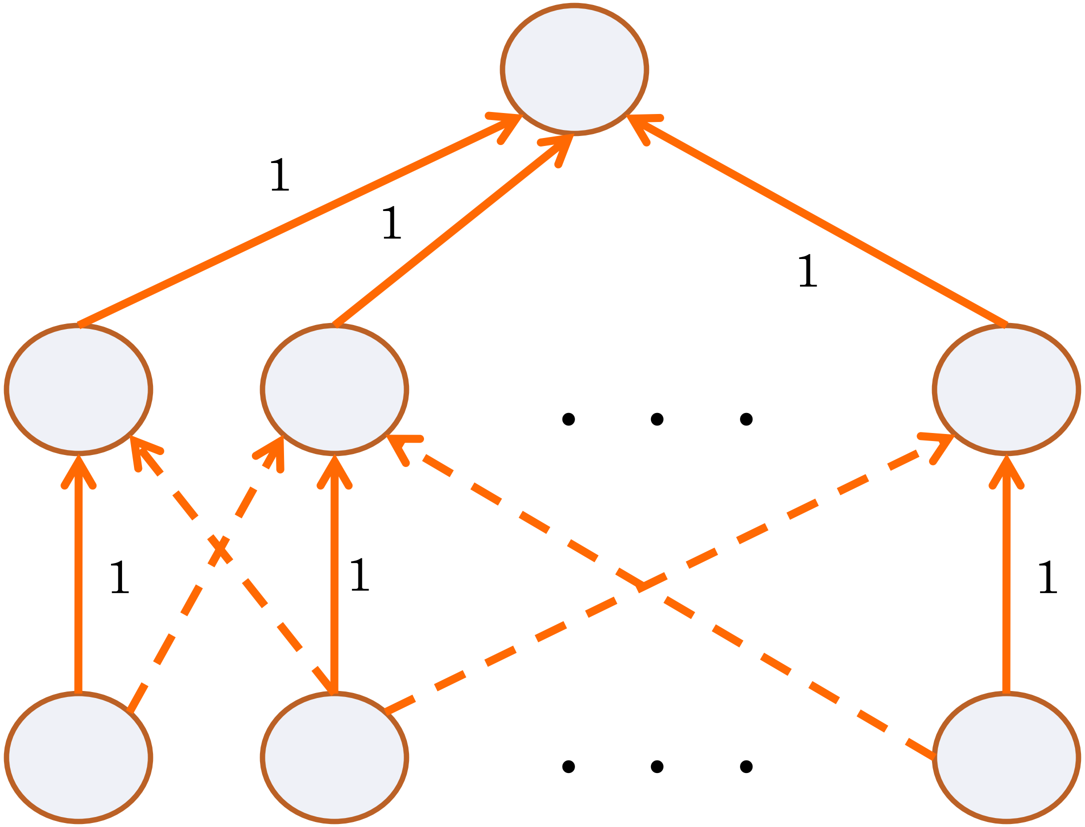


A Lower Bound



All queries give 1-bit answers

A Lower Bound



$2^{\Omega(n^2)}$ such graphs, $\Omega(n^2)$ l.b.

First Some Definitions

- The depth of a node is its distance to the root
- An Up edge is an edge from a node of larger depth to a node of smaller depth
- A Level edge is an edge between two nodes of same depth
- A Down edge is an edge from a node at smaller depth to a node at higher depth
- A leveled graph of a social network is the graph of Up edges

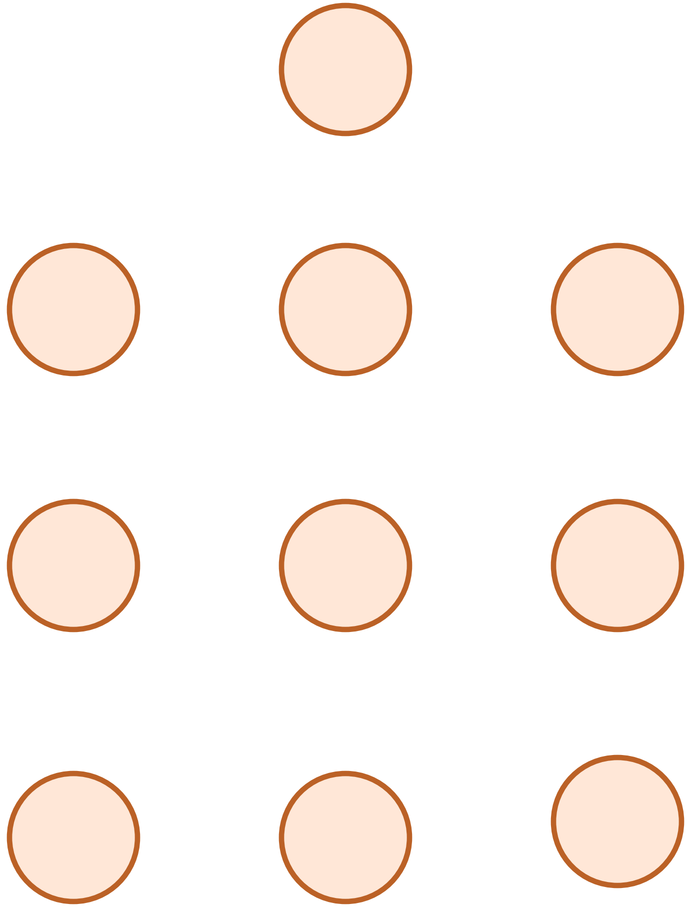
Excitation Paths

- An excitation path for a node n is a VIQ in which a subset of the free agents form a simple directed path from n to the output. All agents not on the path with inputs into the path are suppressed.
- We also have a shortest excitation path

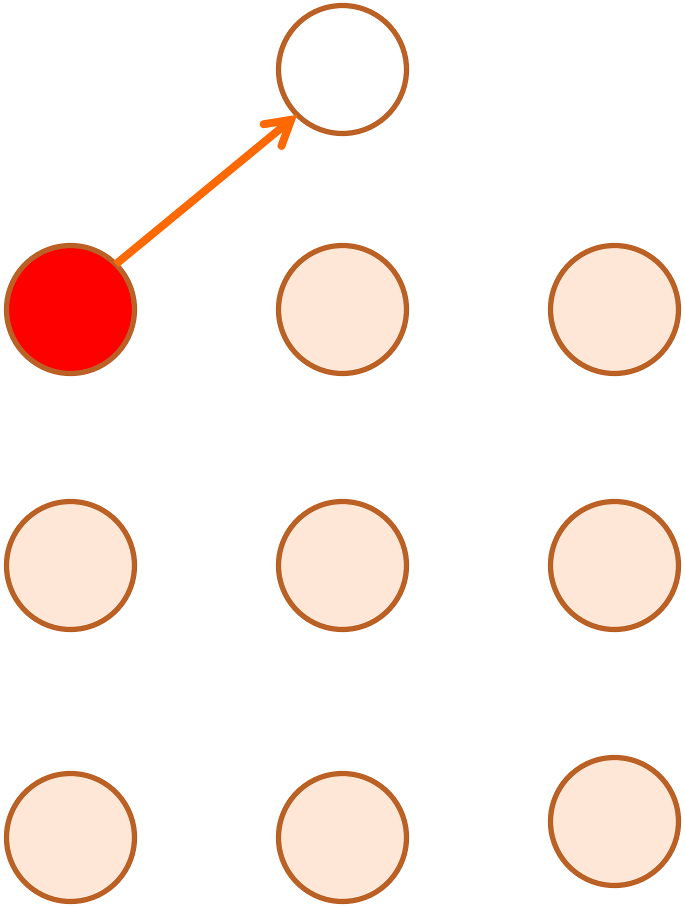
The Learning Algorithm For Networks Without 1 Edges

- First Find-Up-Edges to learn the leveled graph of S
- For each level, Find-Level-Edges
- For each level, bottom-down, Find-Down-Edges

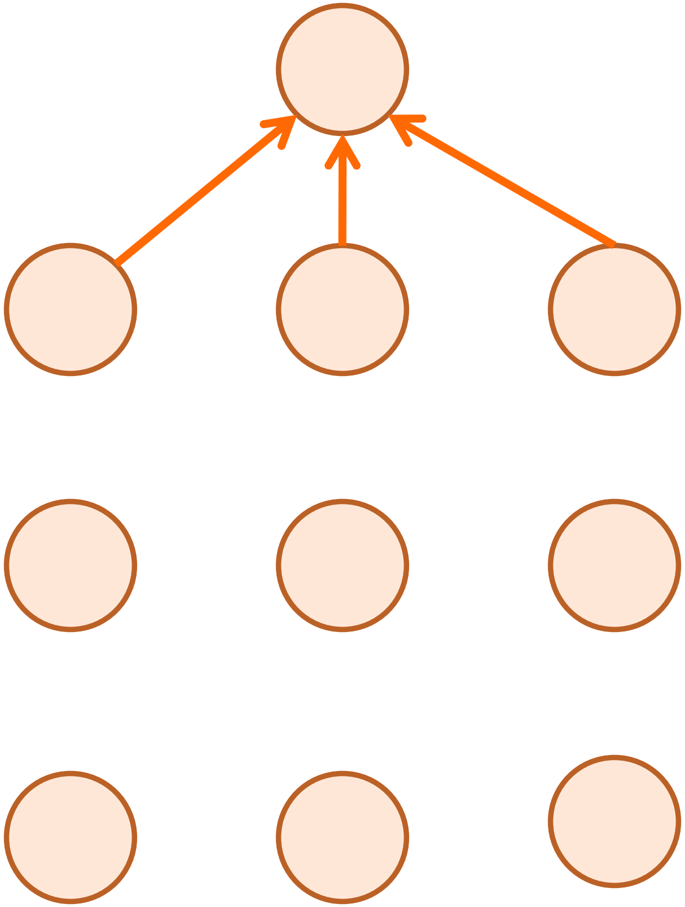
Find-Up-Edges



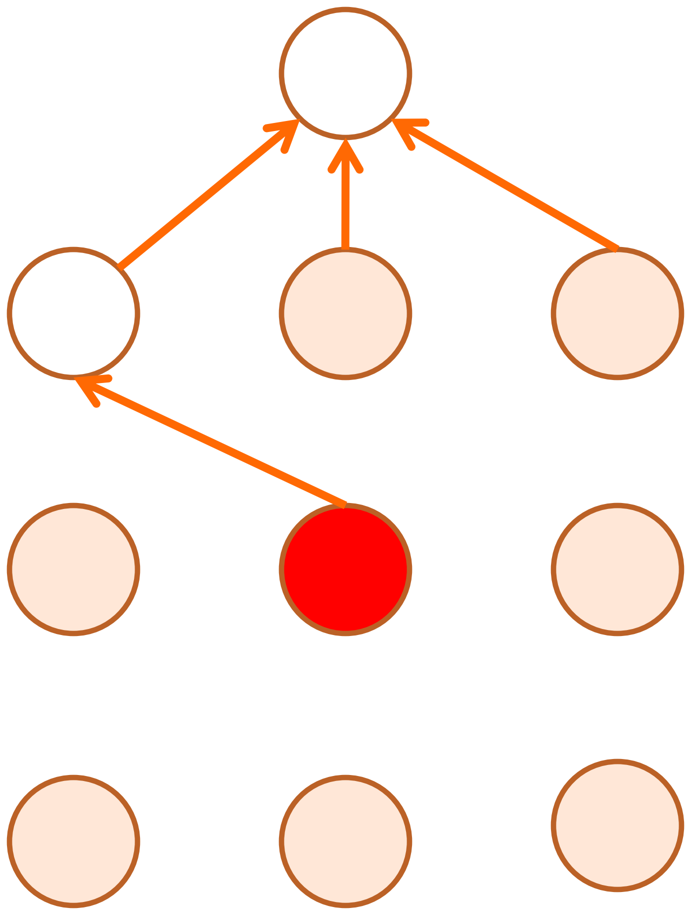
Find-Up-Edges



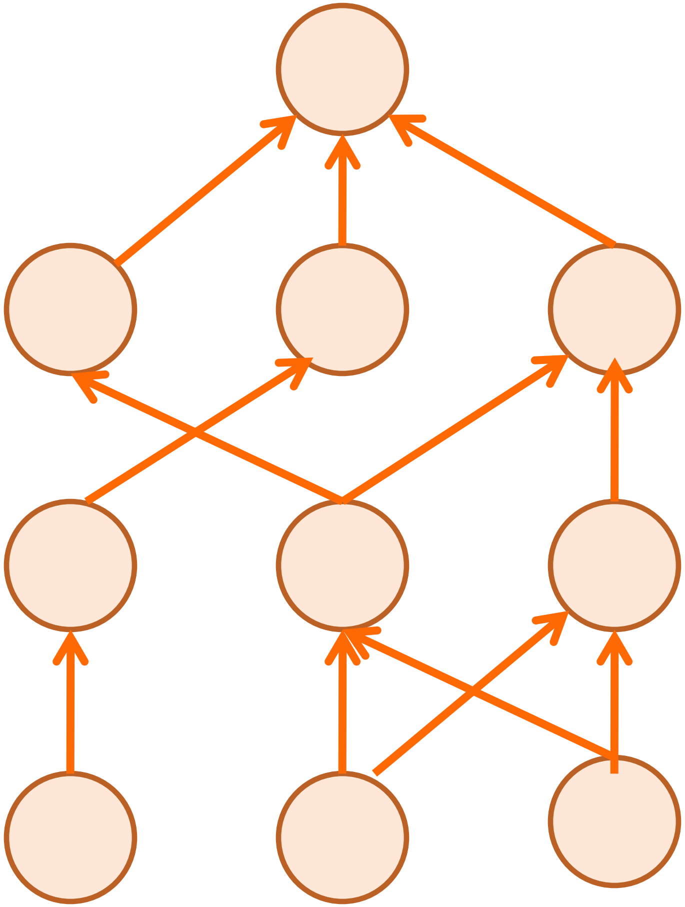
Find-Up-Edges



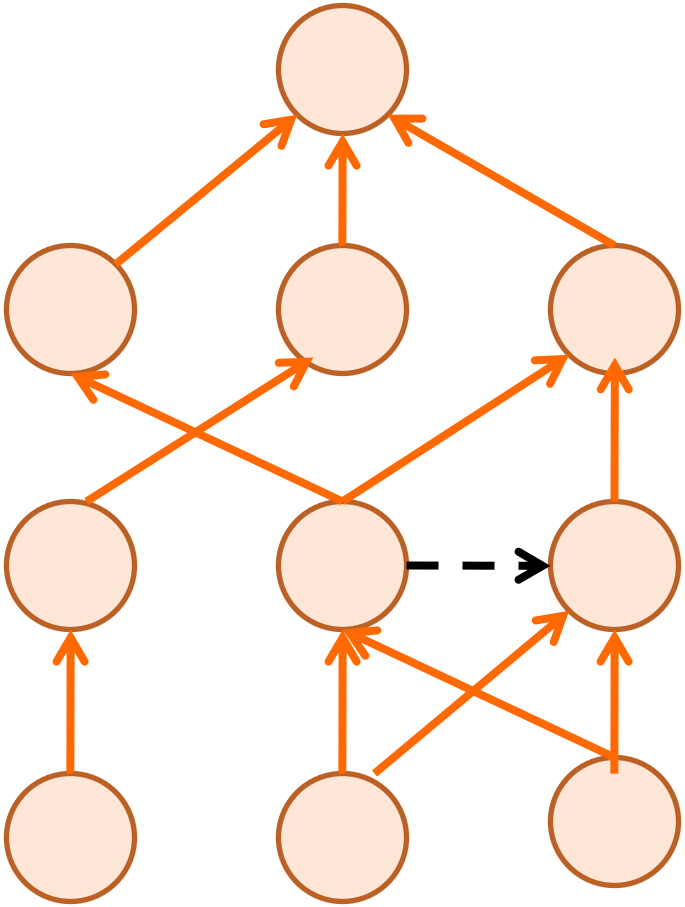
Find-Up-Edges



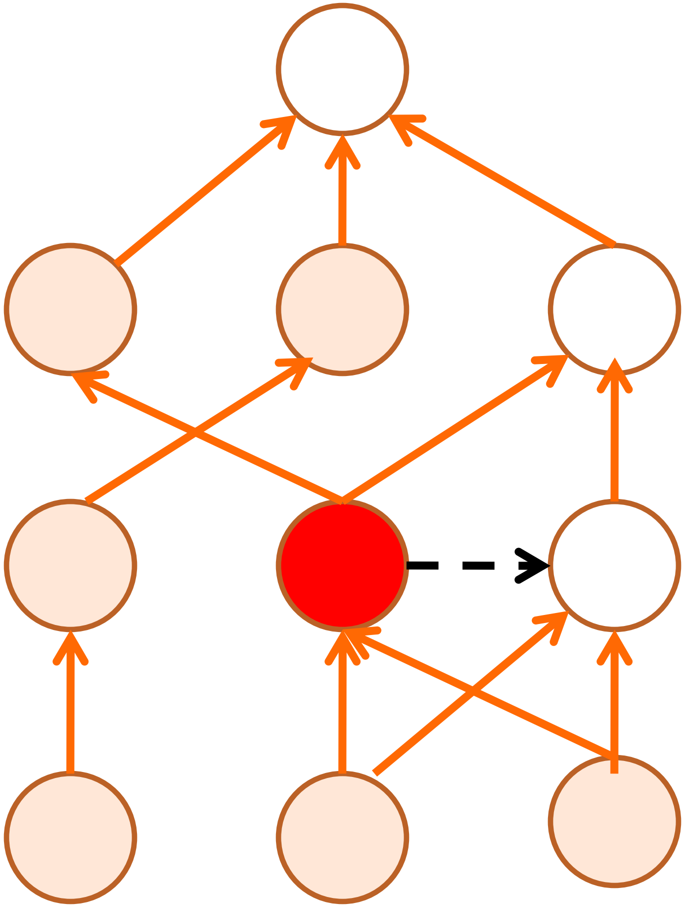
FIND-UP-EDGES



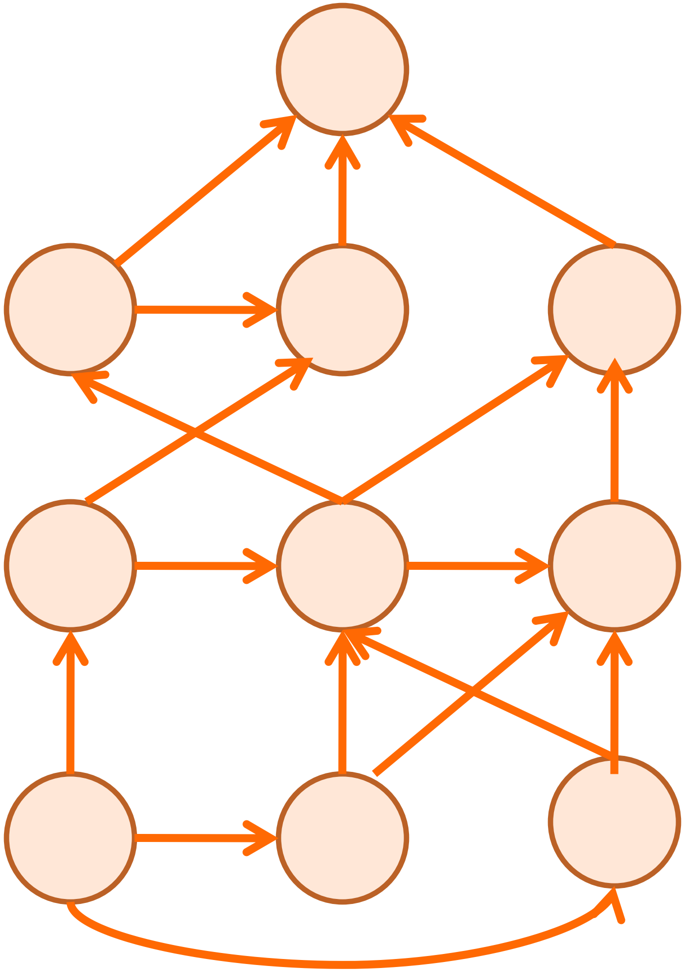
Find-Level-Edges



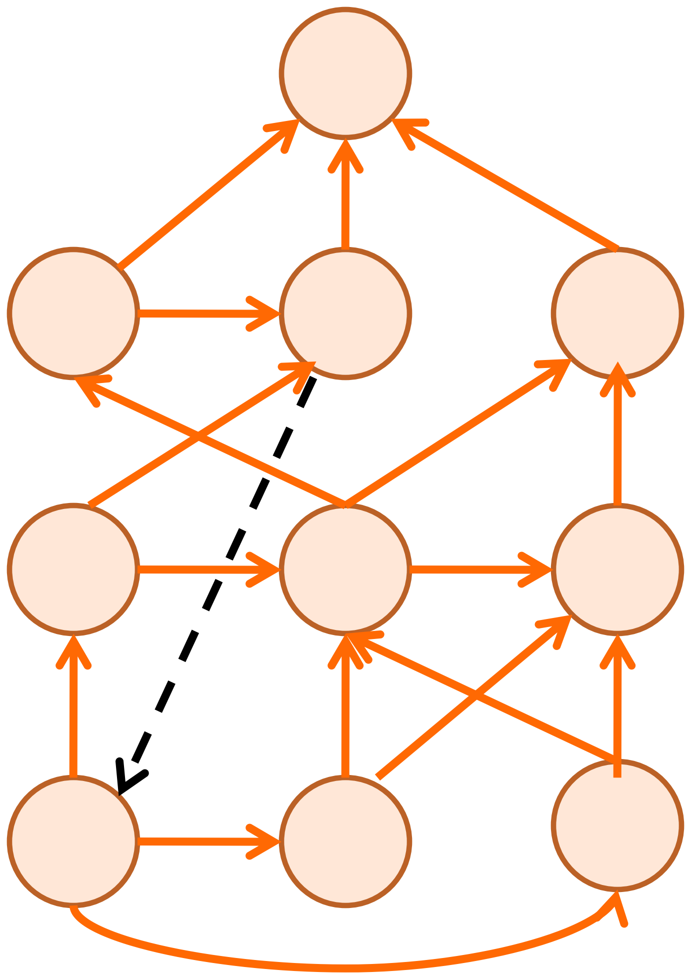
Find-Level-Edges



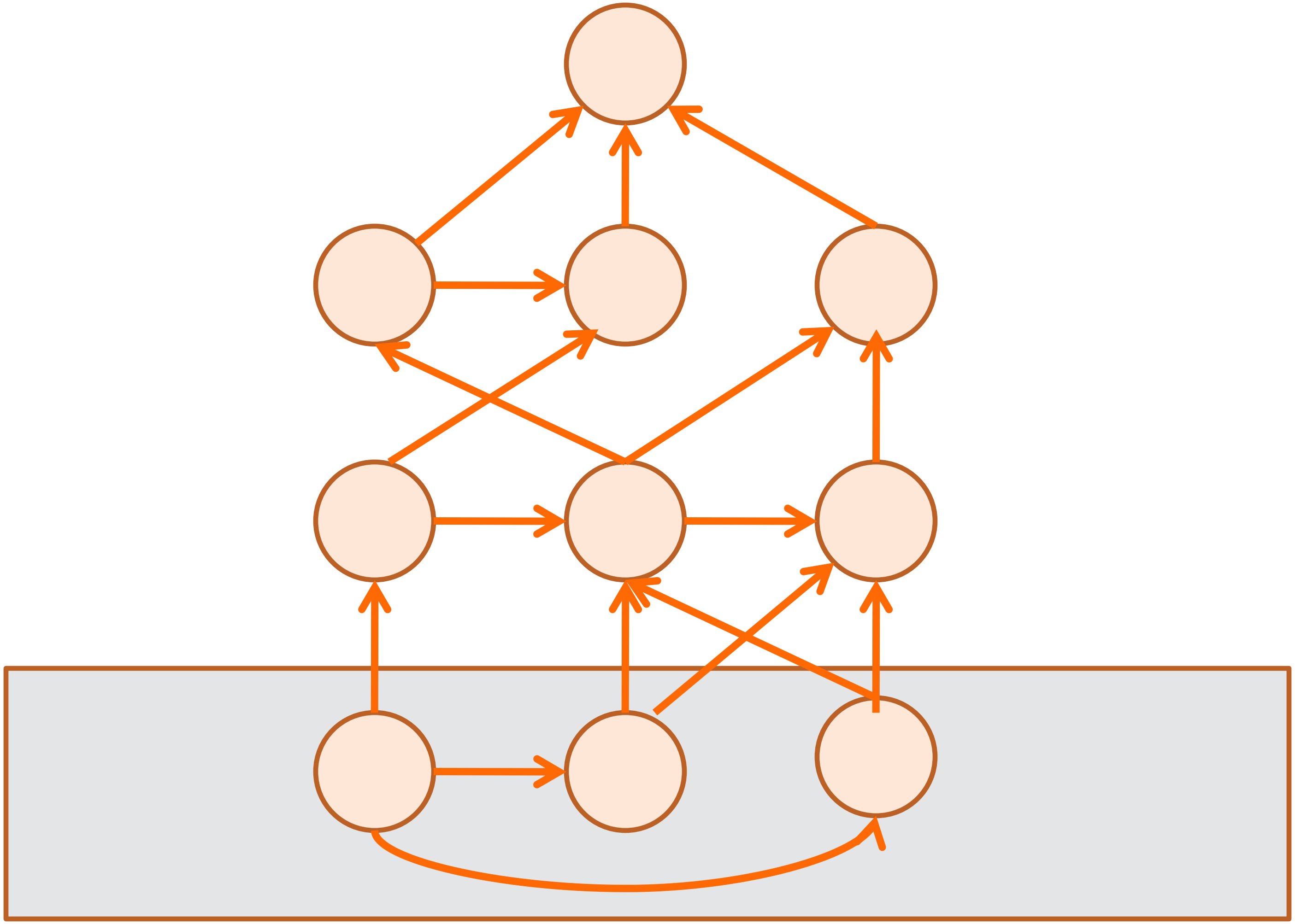
FIND-LEVEL-EDGES



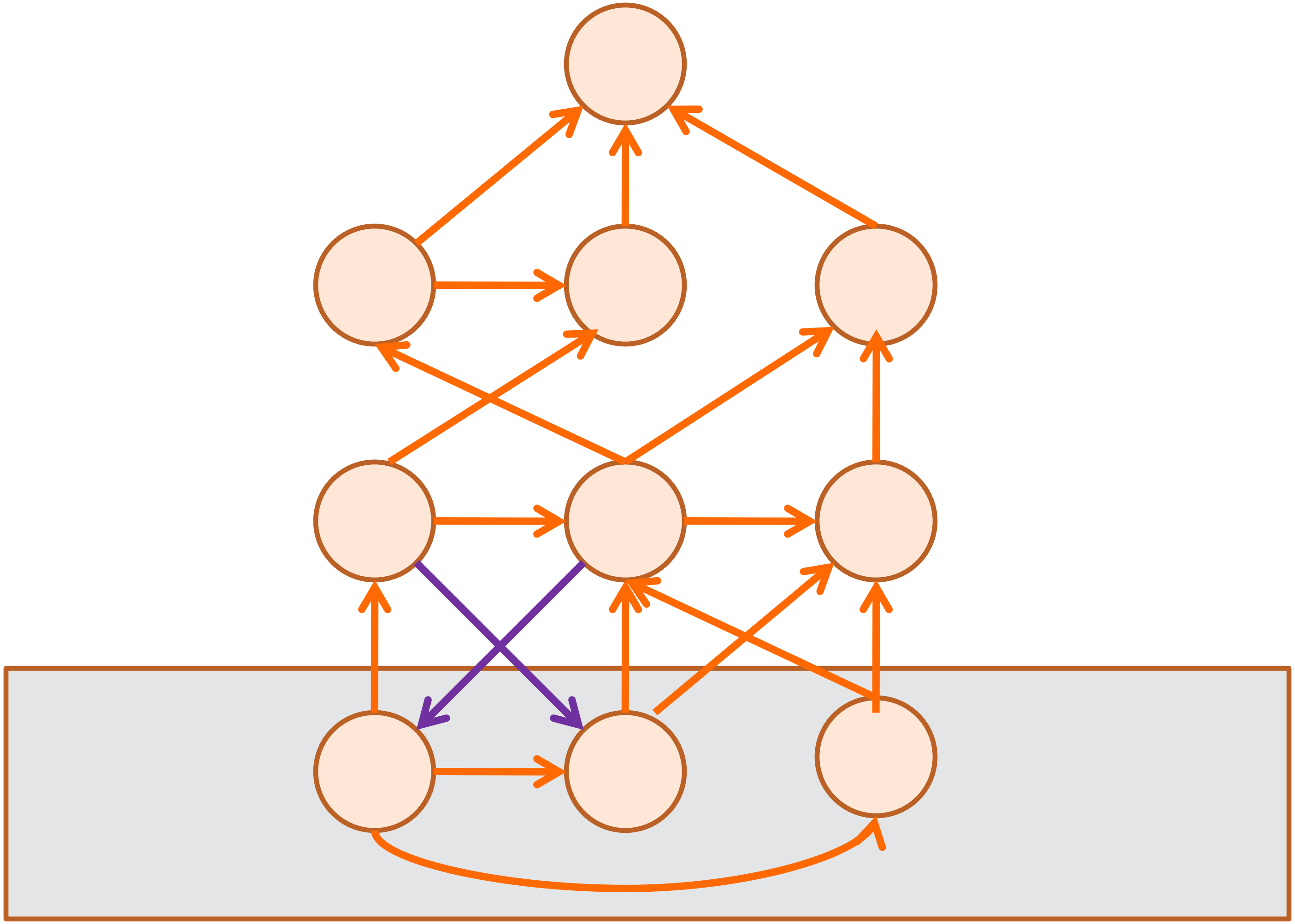
FIND-DOWN-EDGES



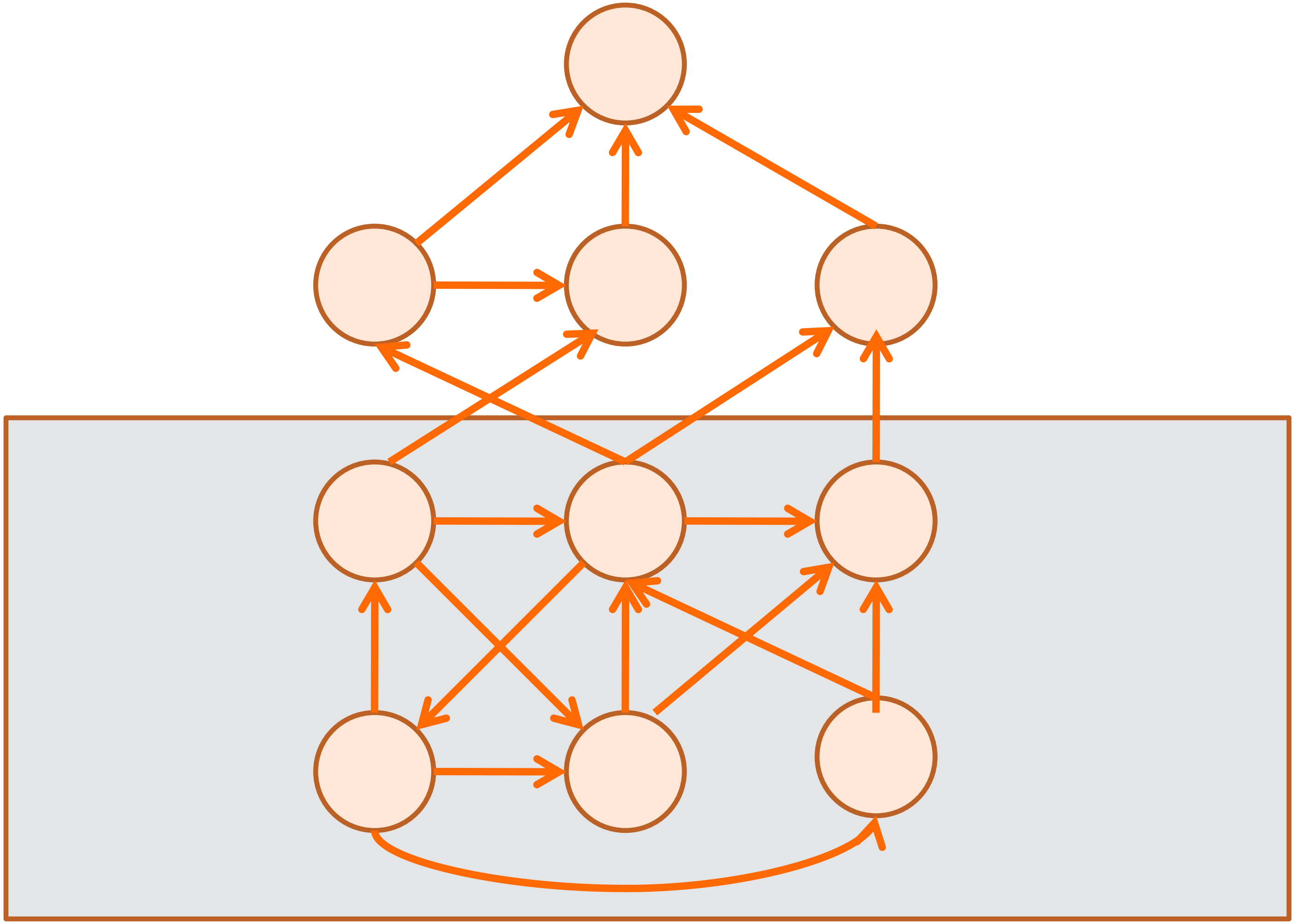
FIND-DOWN-EDGES



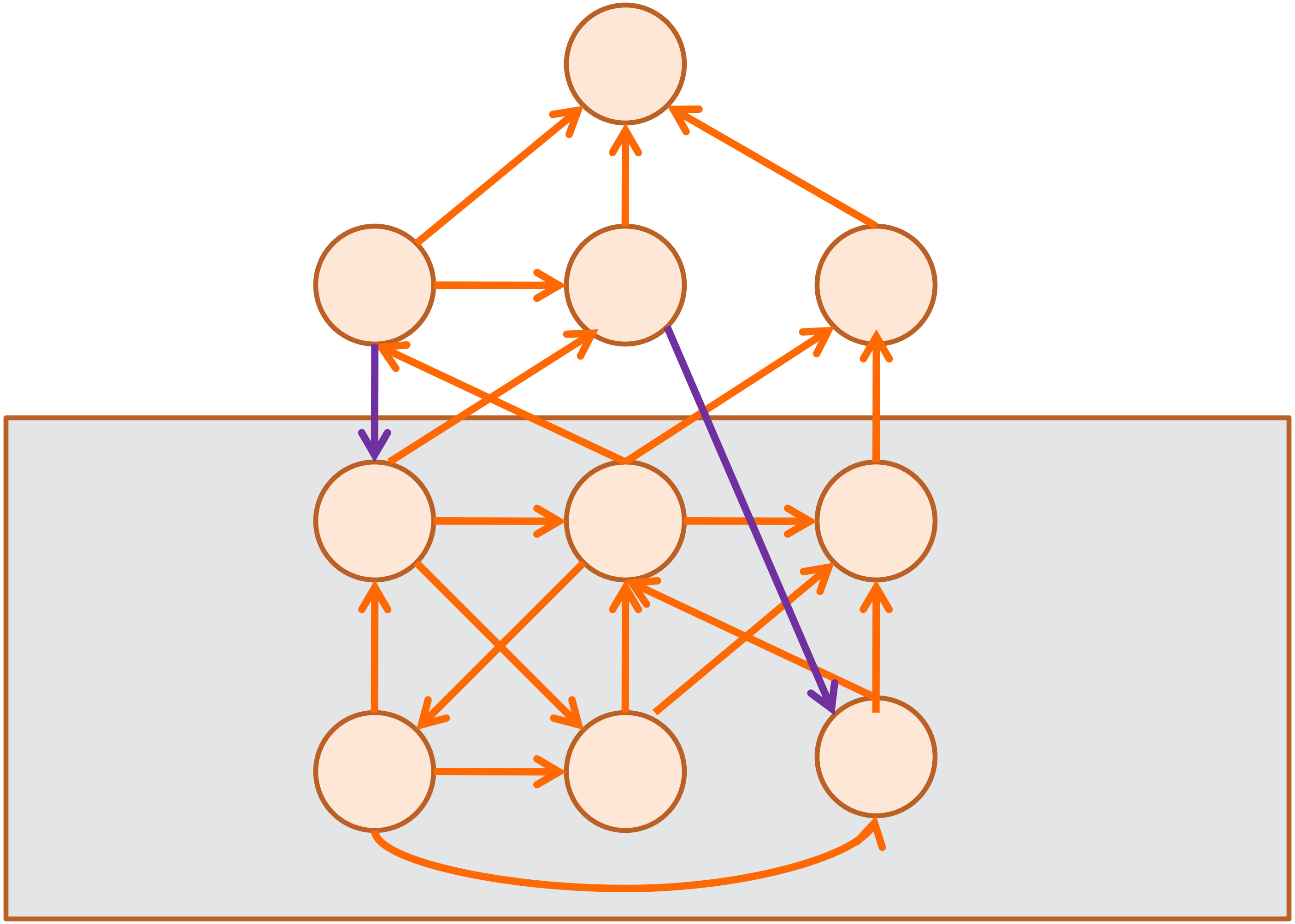
Find-Down-Edges



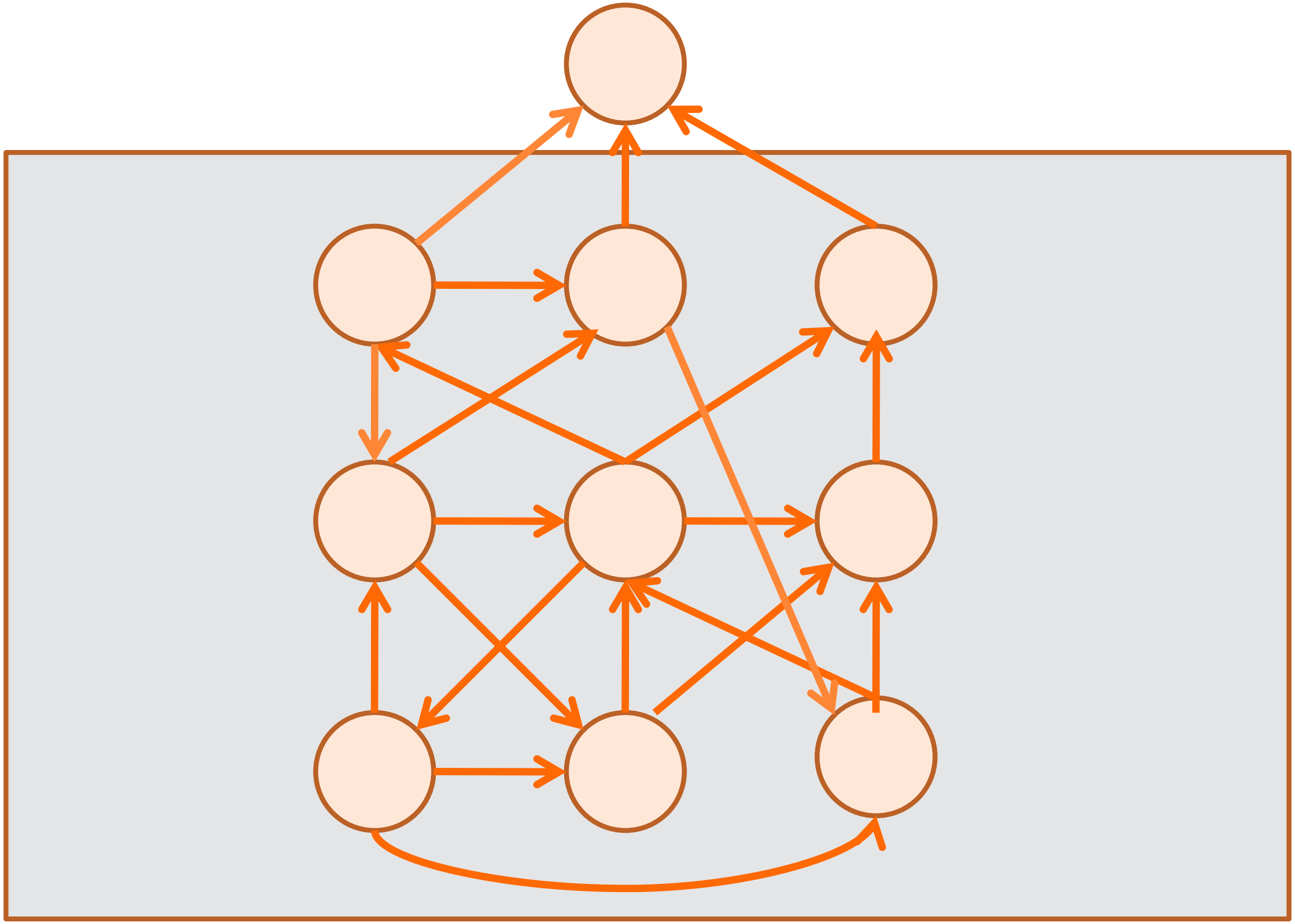
Find-Down-Edges



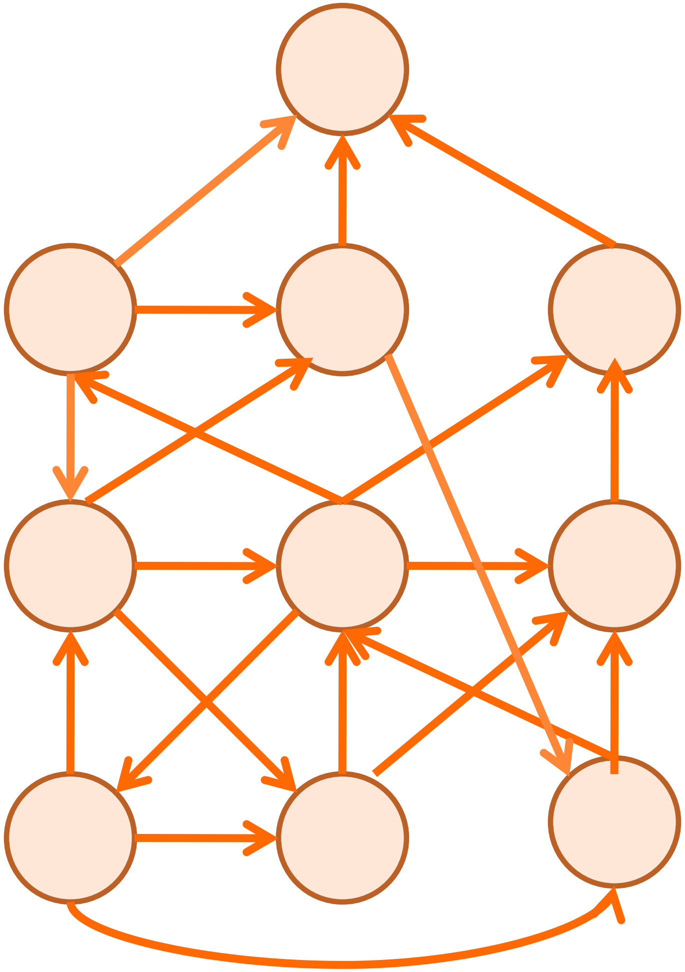
Find-Down-Edges



FIND-DOWN-EDGES



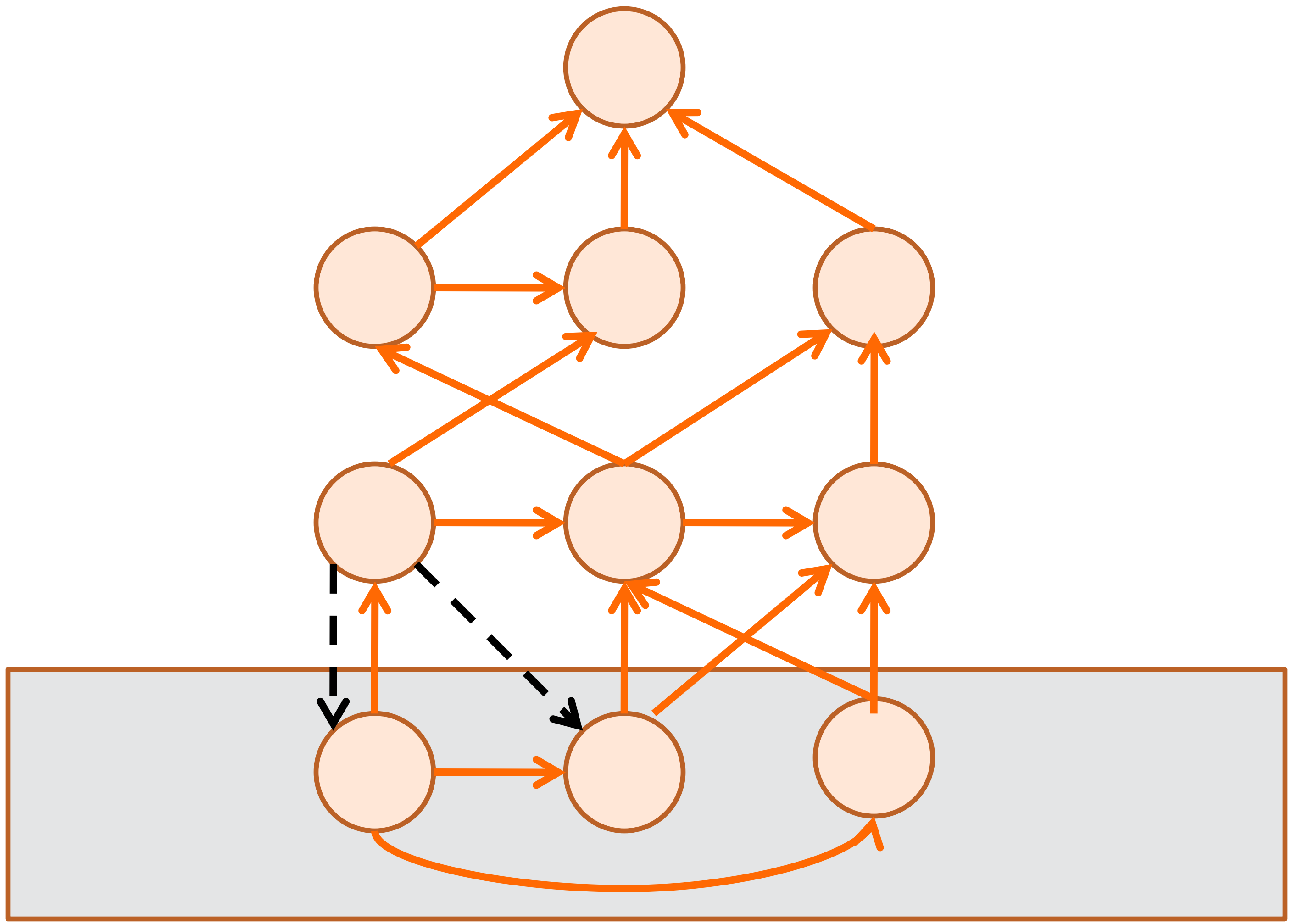
FIND-DOWN-EDGES



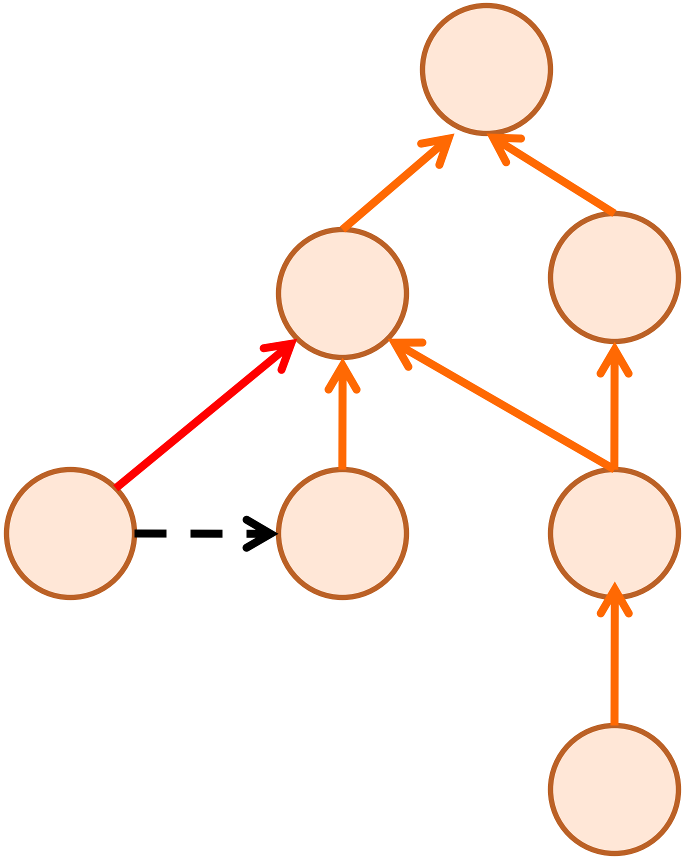
Find-Down-Edges

- For each node u at current level
  - Sort each node v_i in C (complete set) by distance to the root in $G - \{u\}$
  - Let $v_1 \ldots v_k$ be the sorted v_is
  - Let $pi_1 \ldots pi_k$ be their corresponding shortest paths to the root in $G - \{u\}$
  - For i from 1 to k
    - Do experiment of firing u, leaving pi_i free, and suppressing the rest of the nodes.

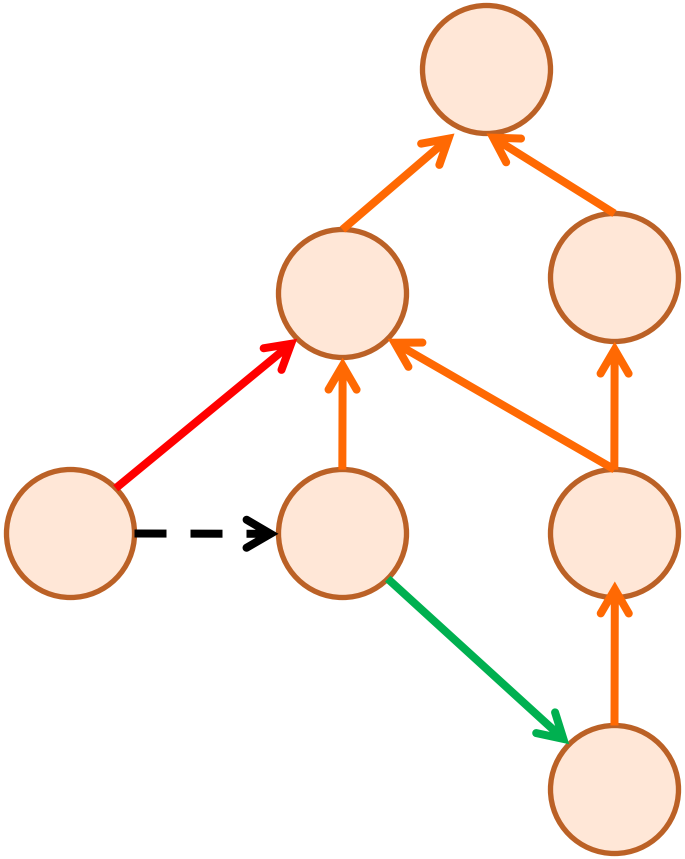
For Example



With Ones – a Problem



With Ones – A Problem



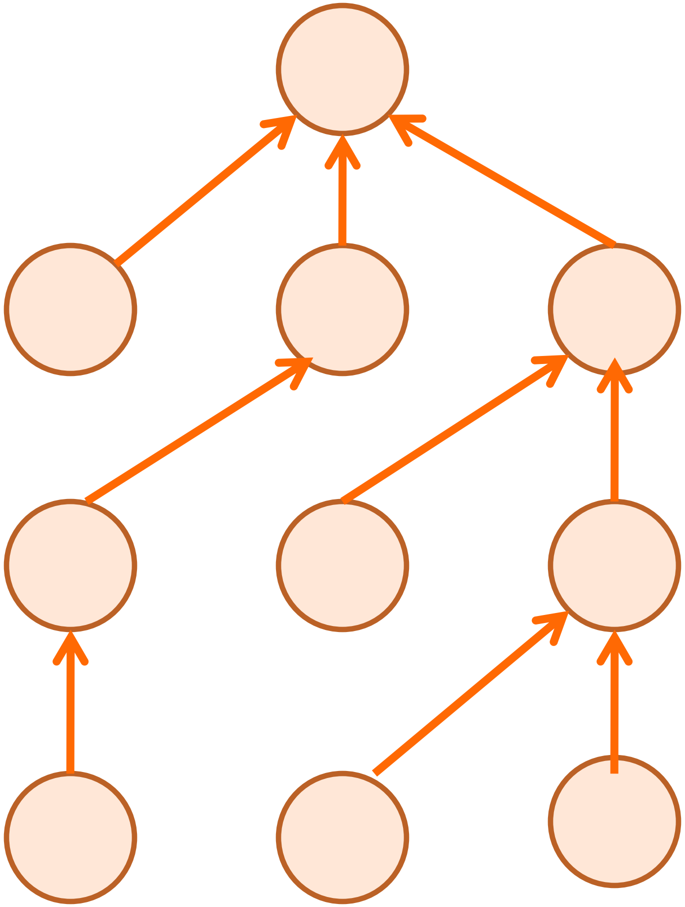
With Ones

- Algorithm gets more complicated
- Level edges and down edges are found in one subroutine
- In looking for down edges from u, need to avoid not just u, but also all nodes reachable from u by 1 edges
- There always exists some pair of nodes, with source in L (current level) and destination in C + L where you can look for an edge.

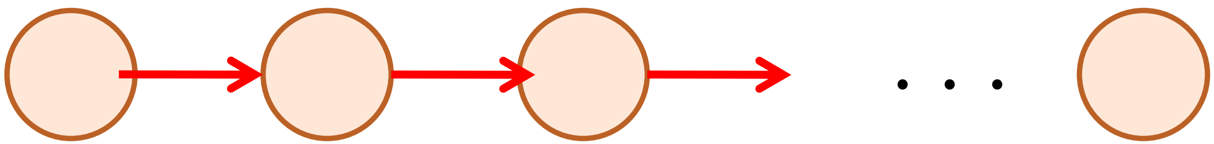
In the End

- We do 1 query per each possible edge, giving an $O(n^2)$ algorithm
- Matches the $\Omega(n^2)$ lower bound

Trees



Lower Bound



Like sorting with comparisons

Gives a $\Omega(n \log n)$ lower bound

Algorithm

- Ancestor Test. To test if u has an ancestor in set S, fire u and suppress all nodes in S.
- How to make a Parent Finder using log n Ancestor Tests and other queries? (assume no 1 edges)

Algorithm

- Ancestor Test. To test if u has an ancestor in set S, fire u and suppress all nodes in S.
- How to make a Parent Finder using log n Ancestor Tests and other queries? (assume no 1 edges)
- Remember, your parent is your deepest ancestor

Trees

- Using n queries find all distances to the root
- Using log n queries per node find all parents
- Then “sort out” all 1 edges
- Gives a O(n log n) algorithm that meets the lower bound

Finding Influential Nodes

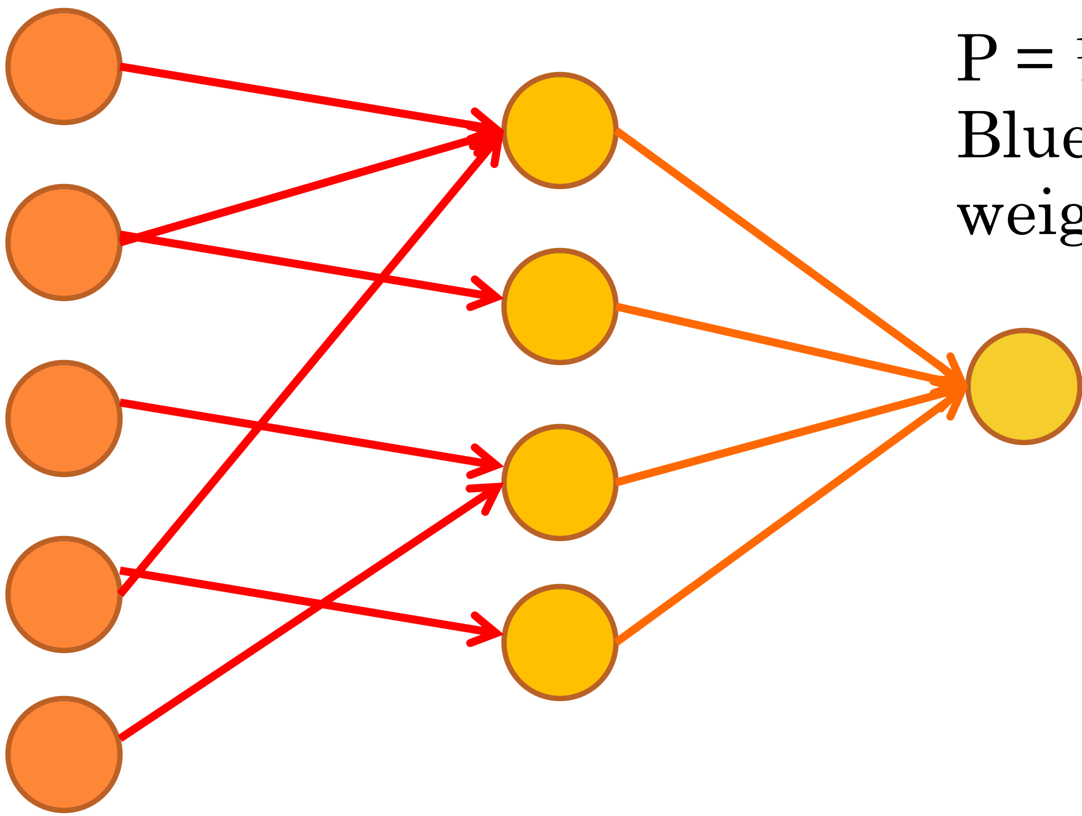
- Suppose instead of learning the social network, we wanted to find an influential set of nodes quickly.

- A set of nodes is influential if, when activated, activates the output with probability at least p

Finding Influential Nodes

- Suppose instead of learning the social network, we wanted to find an influential set of nodes quickly.
- A set of nodes is influential if, when activated, activates the output with probability at least p
- NP Hard to Approximate to log n, even if we know the structure of the network

Reduction from Set Cover



P = ½
Blue edges have weight (1- ½ ^(1/k))

sets k elements

An Approximation Algorithm

- Say the optimal solution has m nodes
- Suppose we wanted to fire the output with probability (p – ε)
- Let I be the set of chosen influential nodes.
- Observation: at any point in the algorithm, greedily adding one more node w to I makes

$$S(e_{I\cup\{w\}}) \geq S(e_I) + \frac{p - S(e_I)}{m}$$

ANALYZING GREEDY

- Using a greedy algorithm, we let k be the number of rounds the algorithm is run

For

$$p\left(1-\frac{1}{m}\right)^k < \epsilon$$

it suffices that

$$e^{-\frac{k}{m}} < \frac{\epsilon}{p}$$

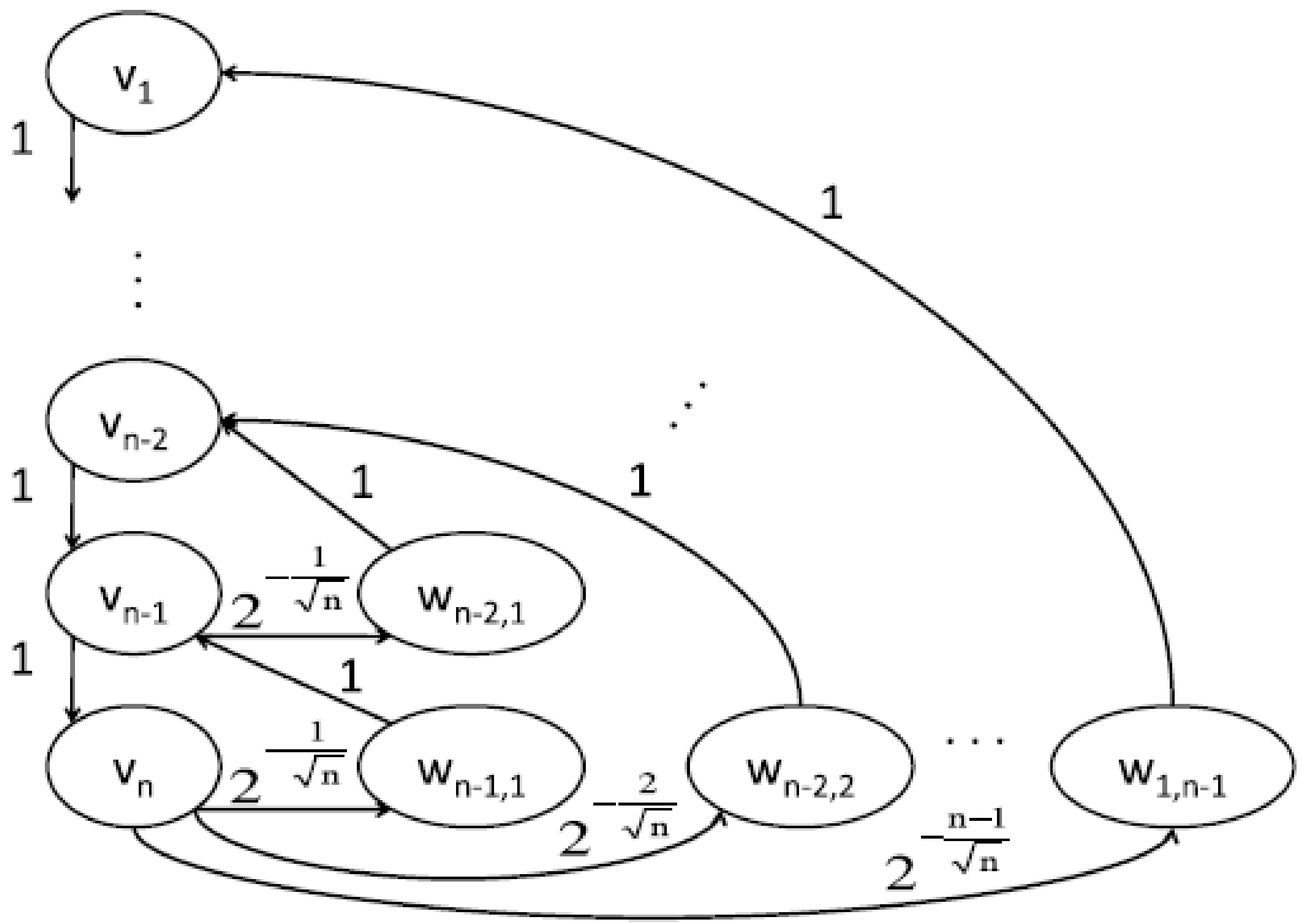
or

$$k > m\log\left(\frac{p}{\epsilon}\right).$$

Reconciling Greedy

- Therefore after $m \log(p/\varepsilon)$ rounds, we get to within ε of p.
- To reconcile with the set cover reduction, if we set $\varepsilon = ½^\wedge(1/n) - ½^\wedge(1/(n-1)) = \theta(1/n^\wedge 2)$, this forces us to cover all the elements.
- Giving a $m \log (p\ n^\wedge 2) = O(m \log (n))$ approximation. Matches the set cover lower bound.

If We Don't Have Exact VIQs



Discussions and Open Problems

- Interesting model to study various hidden structures.
- Linking value injection query model to real-world problems.
- Finding non-path based methods for social networks (and probabilistic networks)
- Reducing Query Size