Actively Dynamic Networks

Paul G. Spirakis

joint work with:

George Mertzios Othon Michail
George Skretas Michail Theofilatos

Department of Computer Science, University of Liverpool, UK
Computer Engineering and Informatics Department, University of Patras, Greece
Department of Computer Science, Durham University, UK

Durham and Liverpool Joint Seminar
14 October 2021

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go

Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go

Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go

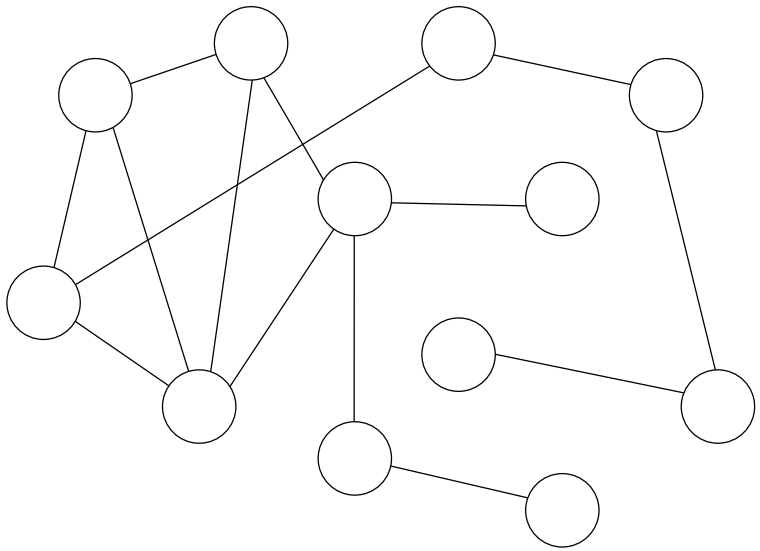
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



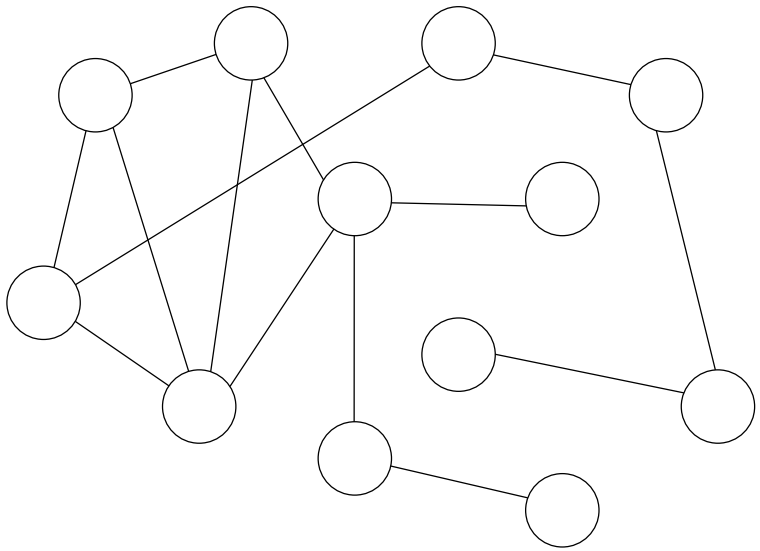
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



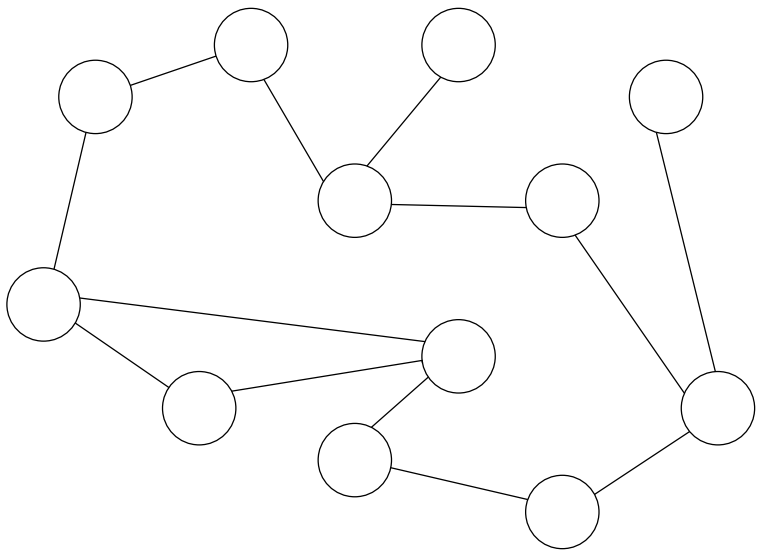
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



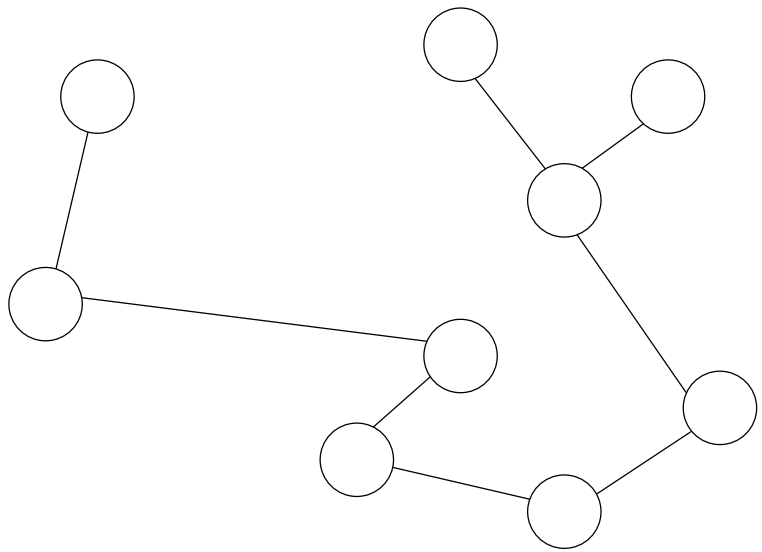
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



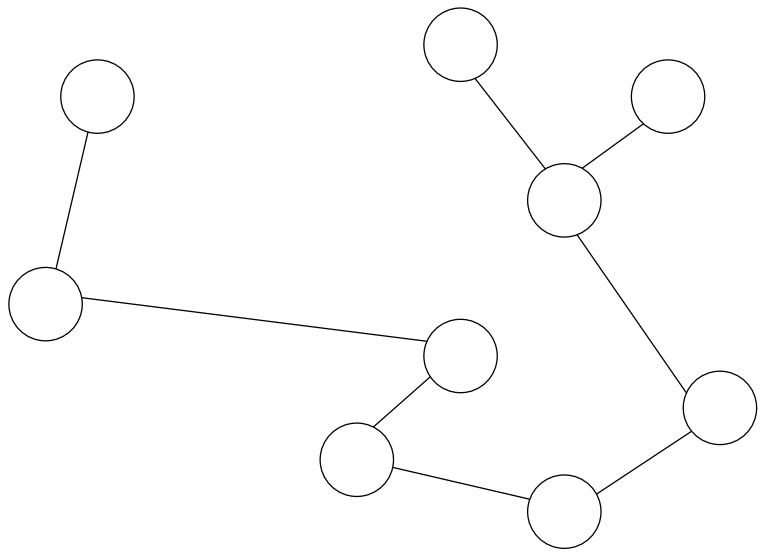
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



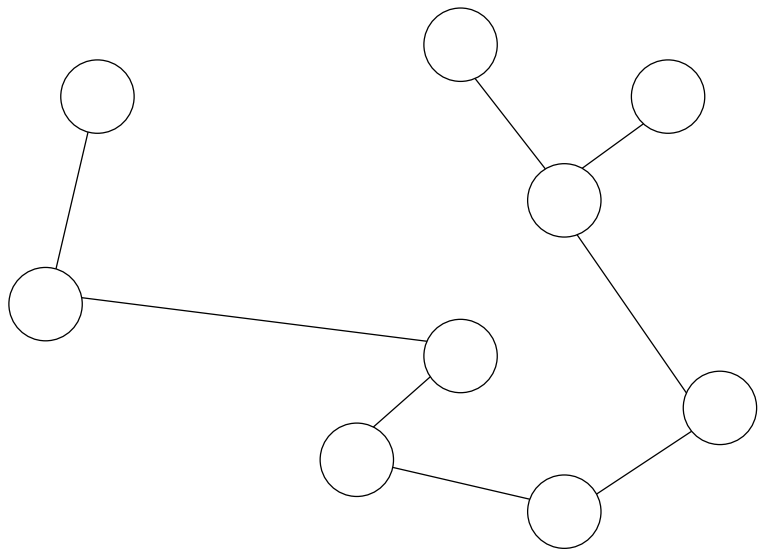
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



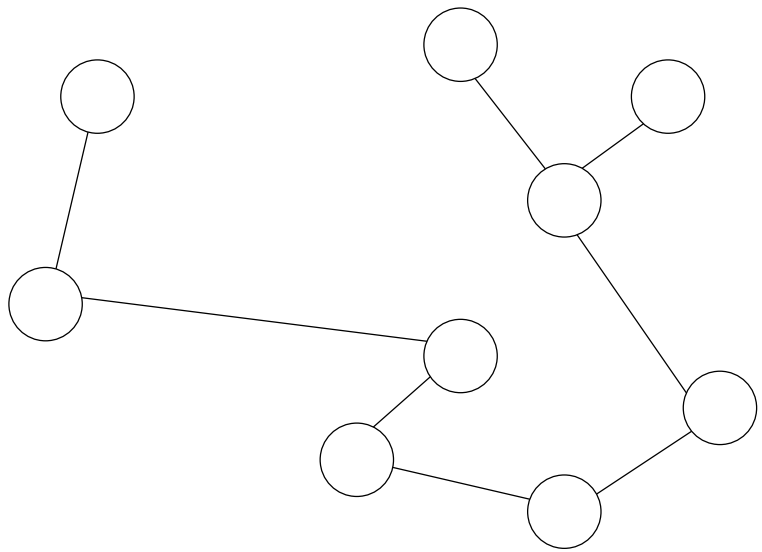
Dynamics and Control:

- Active or Passive
- Centralized or Distributed

Dynamic Networks

General Description:

- Networks whose structure may change
- Usually represented by a graph
- Edges and/or nodes come and go



Dynamics and Control:

- Active or Passive
- Centralized or Distributed

- Temporal Graphs
  - Initiated by Berman [Be96] and Kempe, Kleinberg, and Kumar [JCSS00]
  - Kostakos [Ko09]
  - Multi-labeled, Mertzios, Michail, Chatzigiannakis and Spirakis [ICALP13]
  - Recent work: Molter, Renken, and Zschoche [MFCS21], Enright, Meeks, Mertzios and Zamaraev [JCSS21], Deligkas and Potapov [AAAI20], Akrida, Mertzios, Spirakis and Raptopoulos [JCSS21], Mertzios, Molter, Renken, Spirakis and Zschoche [MFCS21].

- Distributed Computation in Passively Dynamic Networks
  - Population Protocols: Angluin, Aspnes, Diamadi, Fischer and Peralta [DC06], Michail and Spirakis [DC14], Michail, Spirakis, Theofilatos [SSS18], Doty, Eftekhari, Gasieniec, Severson, Stachowiak Uznanski [FOCS21]
  - More powerful devices: Kuhn, Lynch, Oshman [STOC10], Michail, Chatzigiannakis and Spirakis [TCS11]

Related Work

- Construction of Overlay Networks
  - Stoica, Morris, Karger, Kaashoek and Balakrishnan [SIGCOMM01]: Chord (distributed hash table)
  - Aspnes and Shah [TALG07]: Skip Graphs (distributed data structure, balanced tree)
  - Construction: Angluin, Aspnes, Chen, Wu and Yi [SPAA05], Aspnes and Wu [OPODIS07], Gmyr, Hinnenthal Scheideler and Sohler, [ICALP17], Gotte, Hinnenthal and Scheideler, [SIROCCO19] Gilbert, Pandurangan, Robinson, and Trehan [PODC20]

- Programmable Matter
  - Geometry plays a big role.
  - Derakhshandeh, Dolev, Gmyr, Richa, Scheideler and Strothmann [SPAA14], Michail, Skretas and Spirakis [JCSS19], Akitaya *et al.* [ESA19]

Two Models in this Talk

UNIVERSITY OF LIVERPOOL

Distributed Computation and Reconfiguration

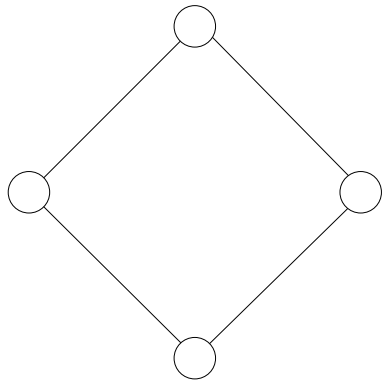
- Static set of nodes, dynamic set of edges
- Edge activations/deactivations
- Transform the initial network G_s into a target network G_f from a family of target networks
- Exploit some good properties of G_f in order to more efficiently solve a distributed task

The Complexity of Growing a Graph

- Dynamic set of nodes and edges
- Node generations and edge activations/deactivations
- Construct an input graph G starting from graph G_0 (graph with a single node) via node generations and edge activations

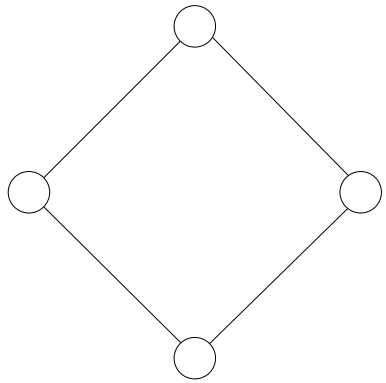
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



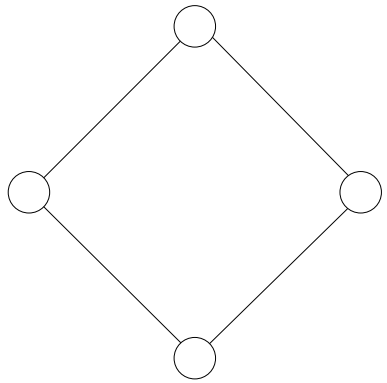
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



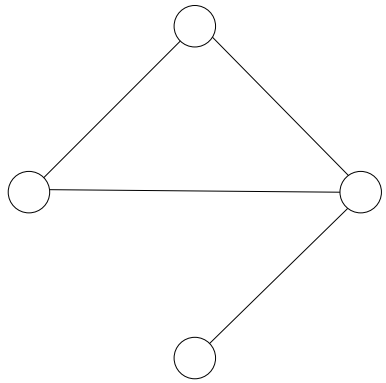
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



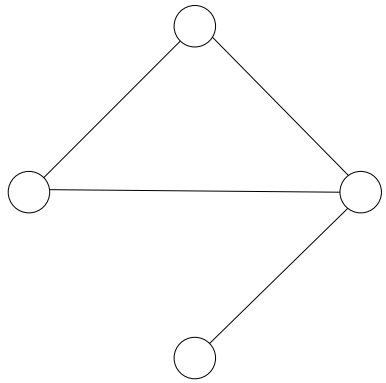
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



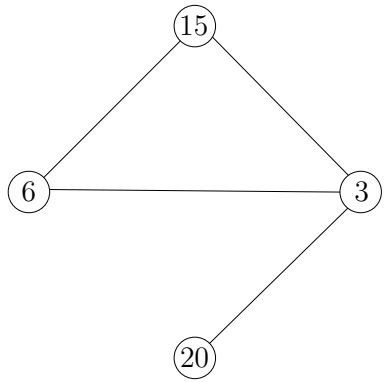
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



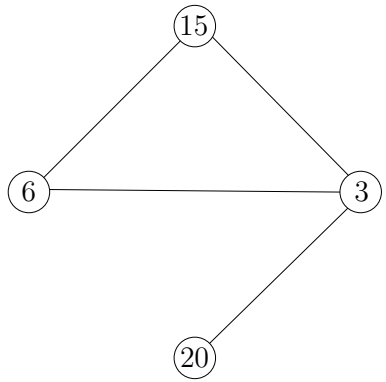
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



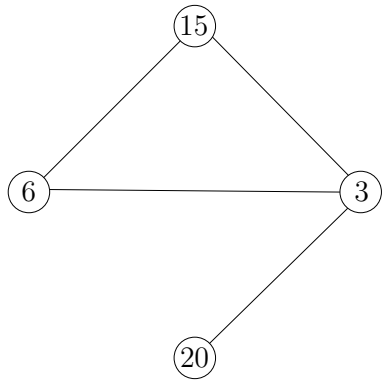
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



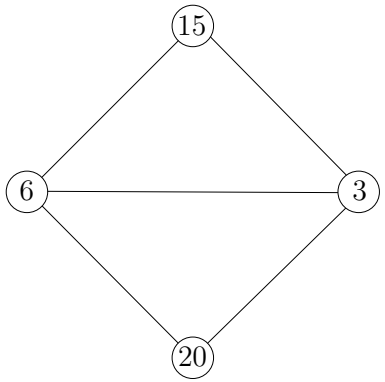
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



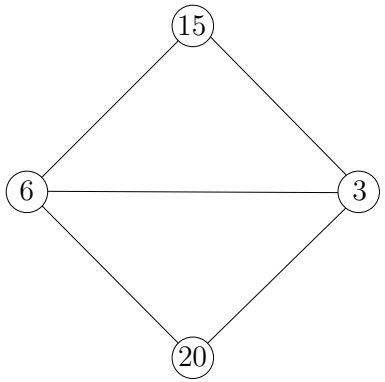
The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



The First Model

- Initial connected network G_s
- Static set V of n nodes
- Dynamic set $E(i)$ of m active edges
- Standard synchronous message passing model
- Each node has a unique ID
- Nodes can only compare unique IDs
- Node u can activate an edge with node v if they have a common neighbor
- One edge between two nodes



The Problems

Problems:

- Leader Election: Elect a unique leader
- Token Dissemination: Each node has a unique piece of information. Every piece of information has to be disseminated to every node
- Depth-d Tree: Transform the initial network into a tree of depth d

The Problems

Problems:

- Leader Election: Elect a unique leader
- Token Dissemination: Each node has a unique piece of information. Every piece of information has to be disseminated to every node
- Depth-d Tree: Transform the initial network into a tree of depth d

The Problems

Problems:

- Leader Election: Elect a unique leader
- Token Dissemination: Each node has a unique piece of information. Every piece of information has to be disseminated to every node
- Depth-d Tree: Transform the initial network into a tree of depth d

The Problems

Problems:

- Leader Election: Elect a unique leader
- Token Dissemination: Each node has a unique piece of information. Every piece of information has to be disseminated to every node
- Depth-d Tree: Transform the initial network into a tree of depth d

The Clique Formation Strategy

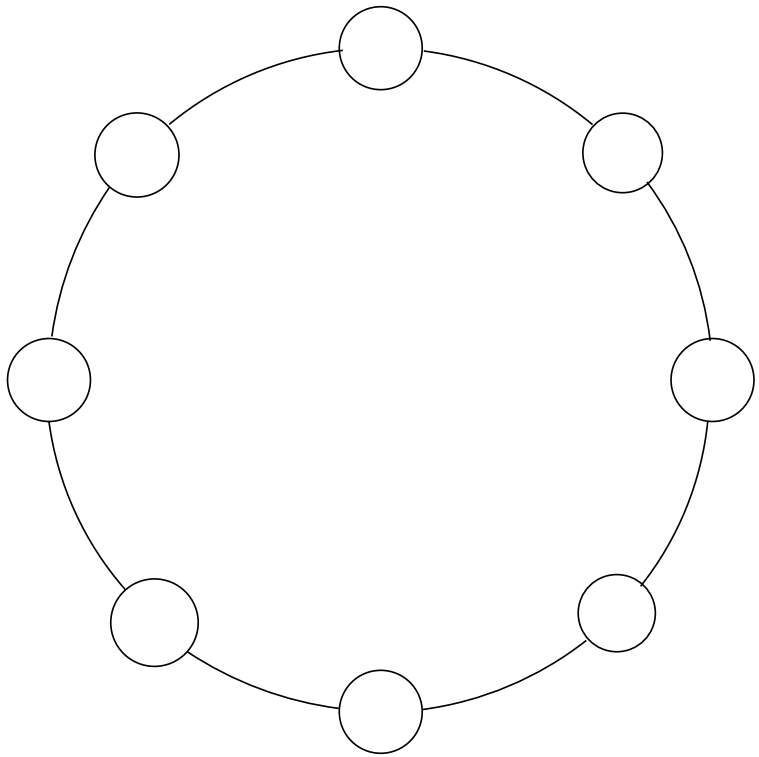
Very simple strategy:

- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges

The Clique Formation Strategy

Very simple strategy:

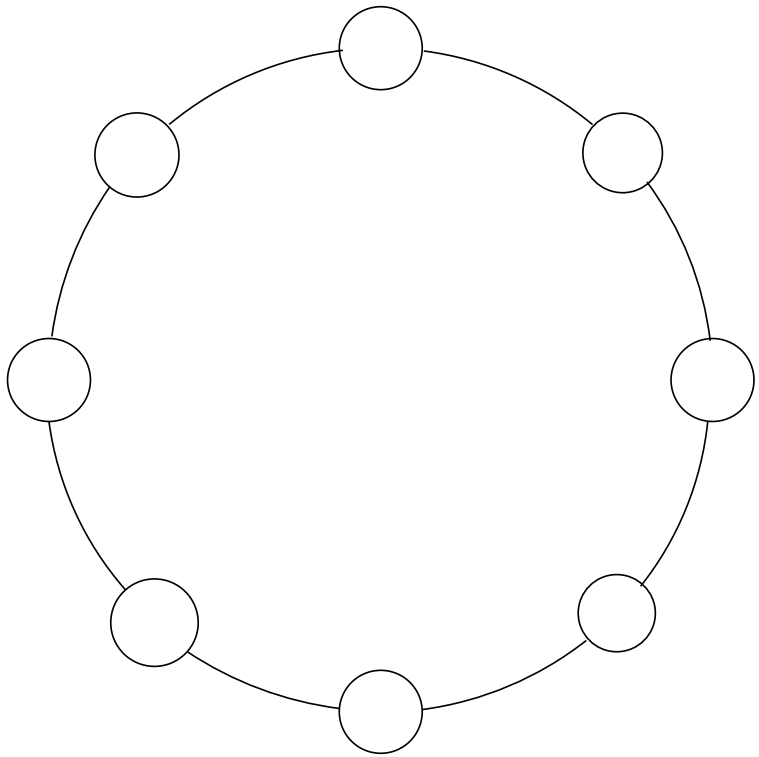
- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

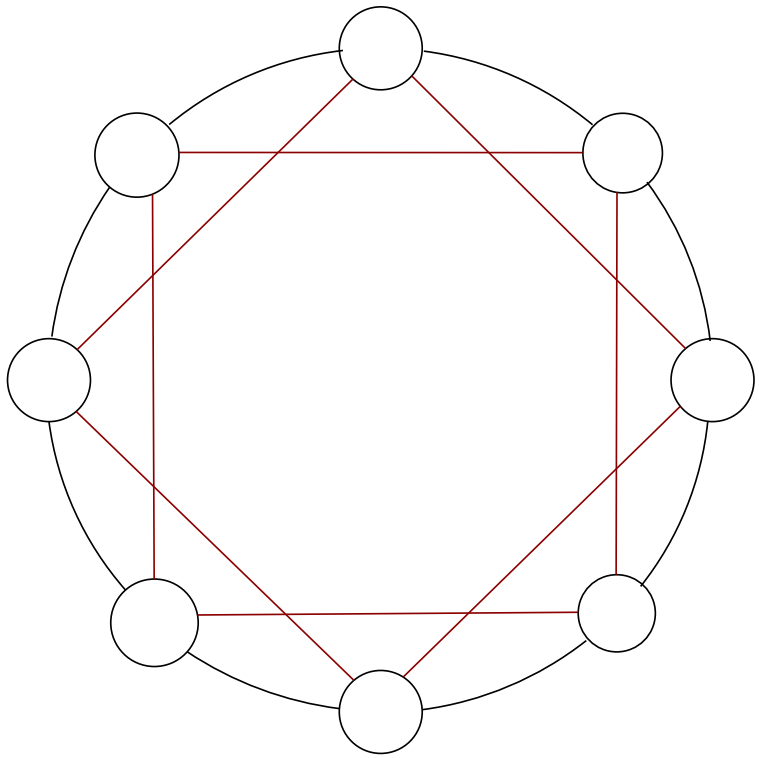
- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

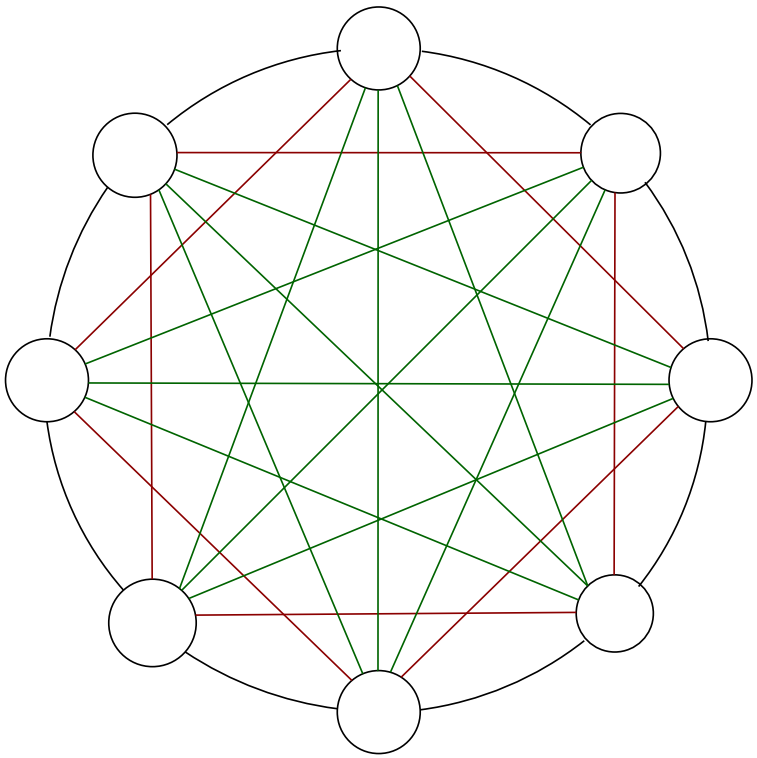
- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

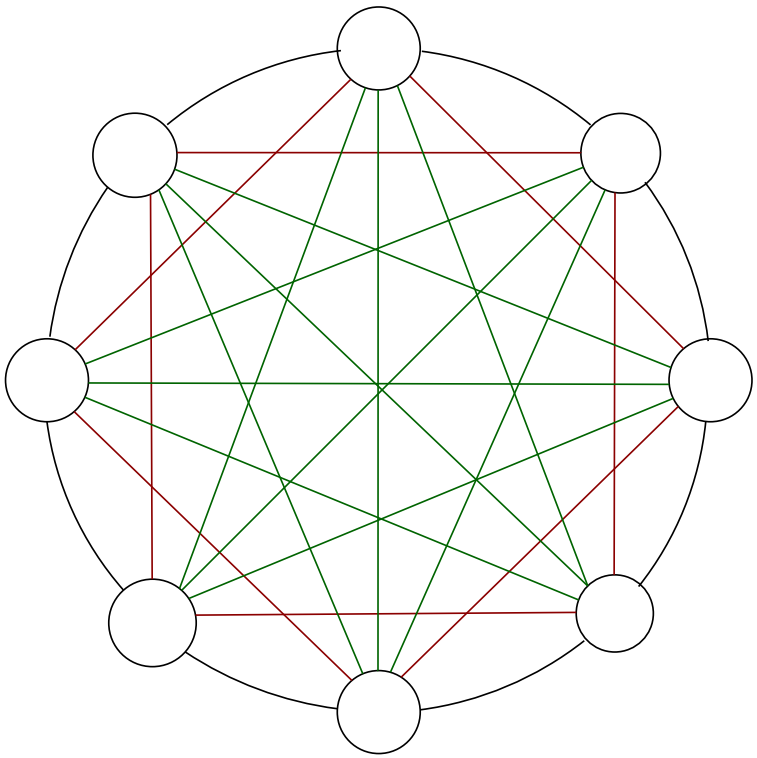
- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

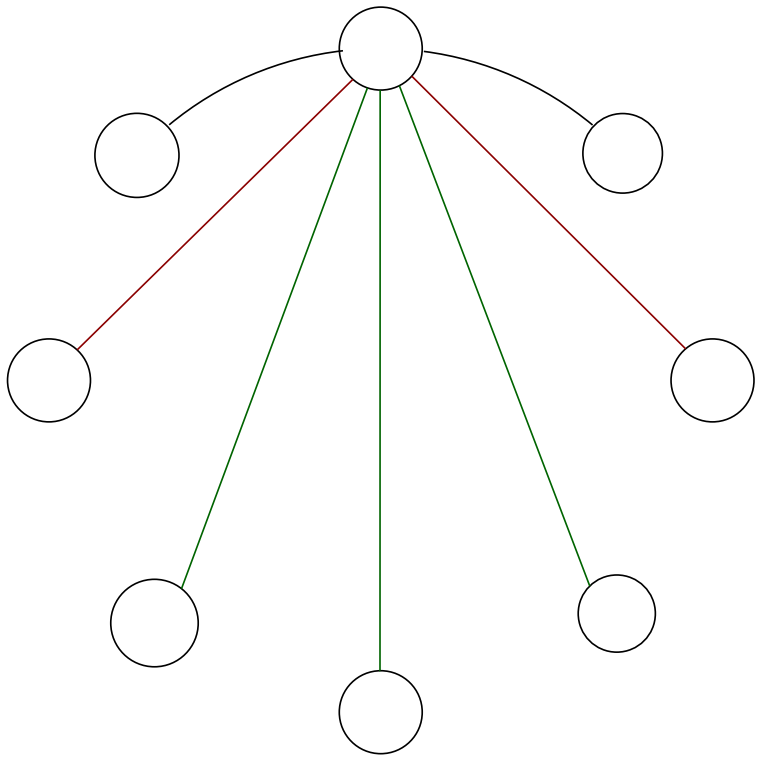
- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges

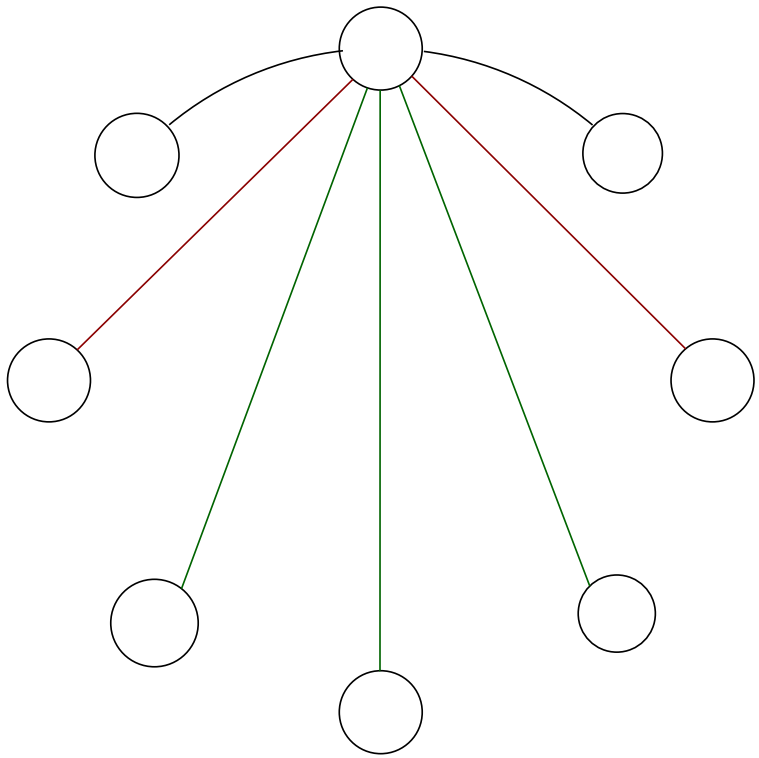


The Clique Formation Strategy

Very simple strategy:

- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges

BUT



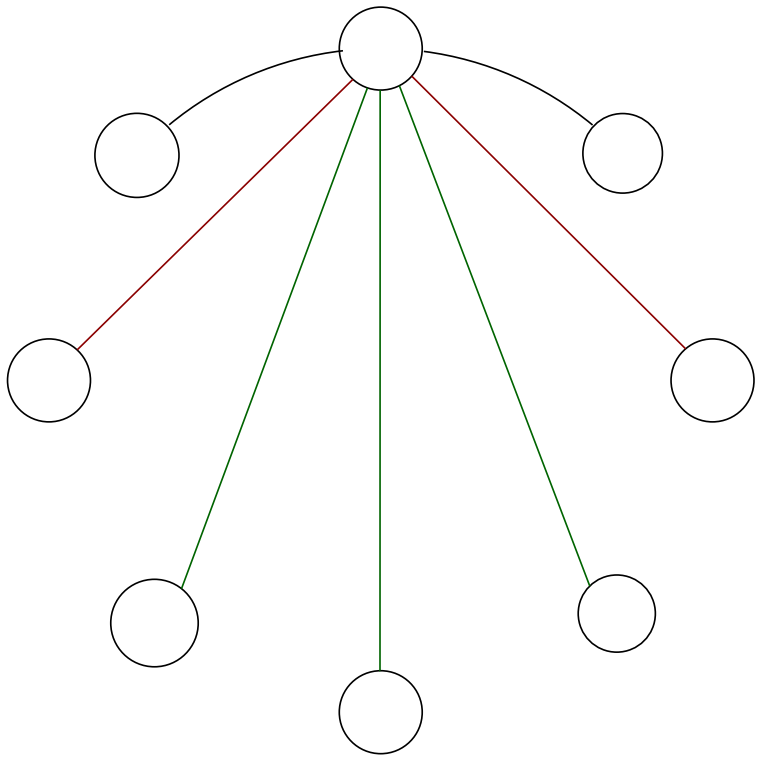
The Clique Formation Strategy

Very simple strategy:

- Initial Cycle network, target Star network
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges

BUT

- Activating and maintaining a connection does not come for free



The Measures

Measures:

- Total Edge Activations: The number of edges activated by the algorithm
- Maximum Activated Edges: The maximum number of activated edges by the algorithm per round
- Maximum Activated Degree: The maximum degree of the network based only on the activated edges by the algorithm

The Measures

Measures:

- Total Edge Activations: The number of edges activated by the algorithm
- Maximum Activated Edges: The maximum number of activated edges by the algorithm per round
- Maximum Activated Degree: The maximum degree of the network based only on the activated edges by the algorithm

The Measures

Measures:

- Total Edge Activations: The number of edges activated by the algorithm
- Maximum Activated Edges: The maximum number of activated edges by the algorithm per round
- Maximum Activated Degree: The maximum degree of the network based only on the activated edges by the algorithm

The Measures

Measures:

- Total Edge Activations: The number of edges activated by the algorithm
- Maximum Activated Edges: The maximum number of activated edges by the algorithm per round
- Maximum Activated Degree: The maximum degree of the network based only on the activated edges by the algorithm

Objective

Give (poly)log(n) time algorithms for the general task of transforming any G_s into a G_f of diameter (poly)log(n), while minimizing the edge-complexity, and solve leader election, depth-(poly)log(n) tree, and token dissemination at the same time.

Our Results

Algorithm	Time	Total Edge Activations	Maximum Activated Edges	Maximum Activated Degree
GraphToStar	$O(\log n)$	$O(n \log n)$	$O(n)$	$O(n)$
GraphToWreath	$O(\log^2 n)$	$O(n \log^2 n)$	$O(n)$	$O(1)$
GraphToThinWreath	$O(\log^2 n / \log \log n)$	$O(n \log^2 n)$	$O(n)$	$O(\log^2 n)$

Table: Algorithms

Bounds	Time	Total Edge Activations	Maximum Activated Edges	Maximum Activated Degree
Centralized Lower	$\Omega(\log n)$	$\Omega(n)$	$\Omega(n/\log n)$	
Centralized Upper	$O(\log n)$	$\Theta(n)$		
Distributed Lower	$O(\log n)$	$\Omega(n \log n)$		

Table: Bounds for Depth-d tree

Introduction on the General Strategy

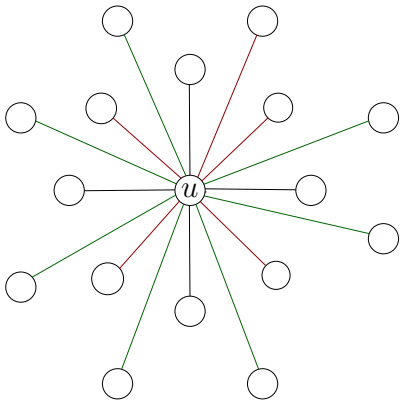
- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

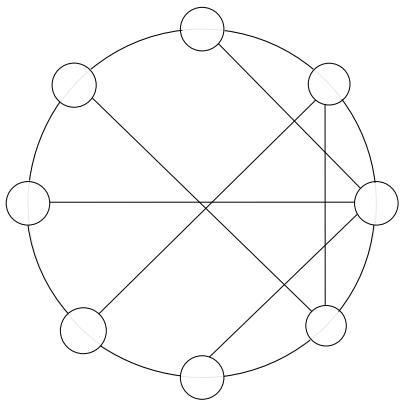
Introduction on the General Strategy

- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

Introduction on the General Strategy

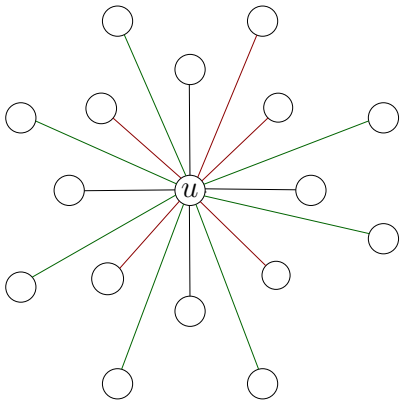
- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

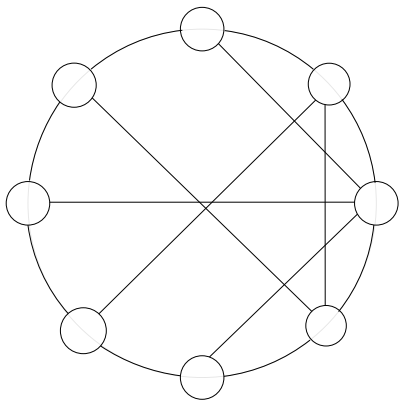




Introduction on the General Strategy

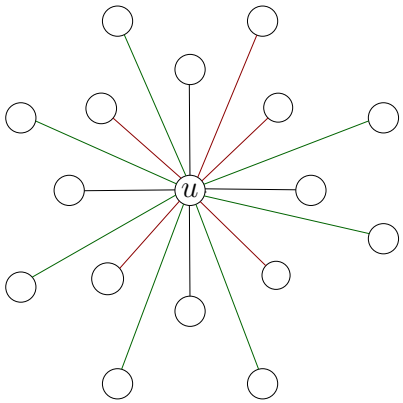
- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

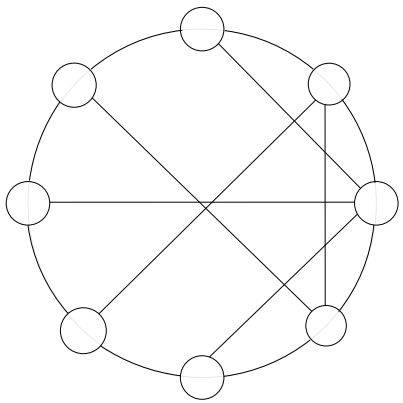




Introduction on the General Strategy

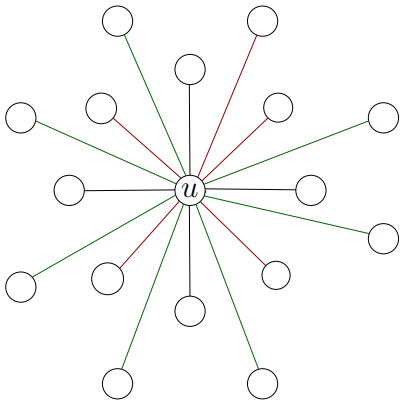
- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

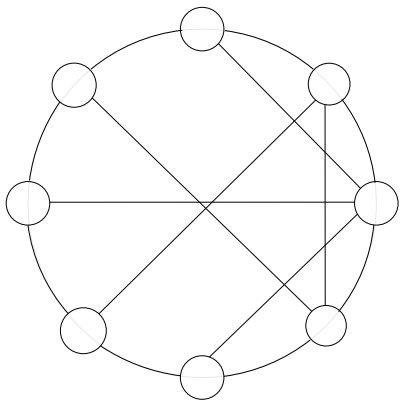




Introduction on the General Strategy

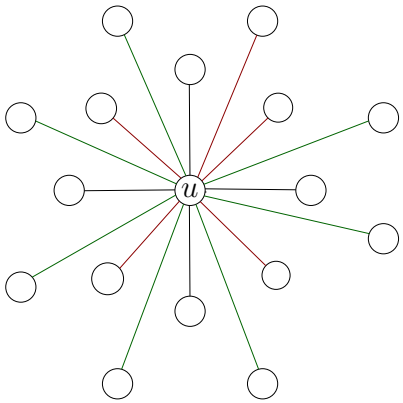
- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

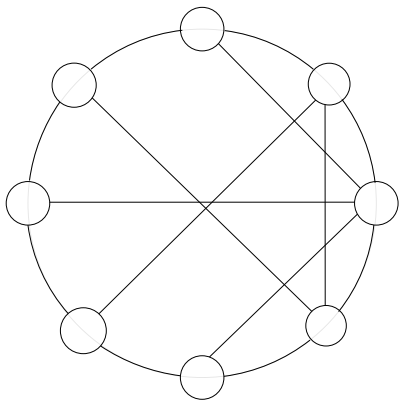




Introduction on the General Strategy

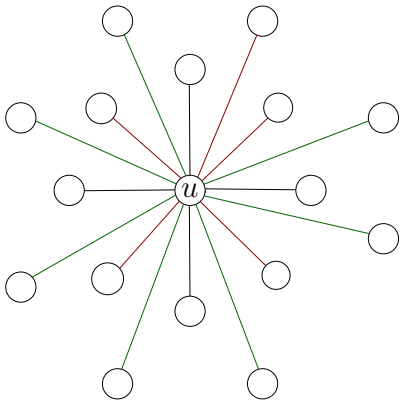
- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter

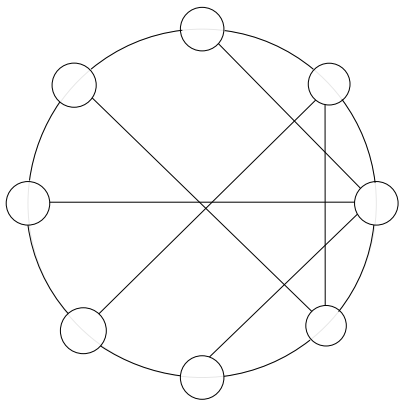




Introduction on the General Strategy

- Nodes are partitioned into committees
- Committees organized into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Polylogarithmic running time
- Time: phases*gadgetdiameter





GraphToStar Algorithm

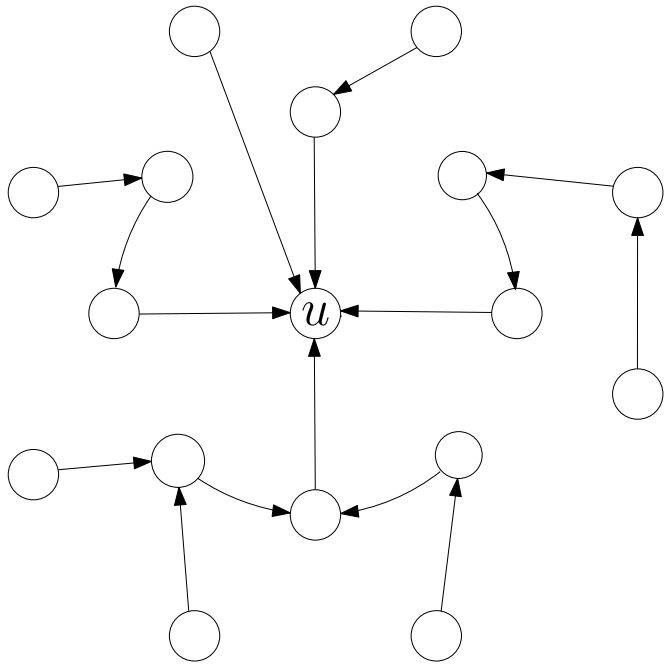
- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent

GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent

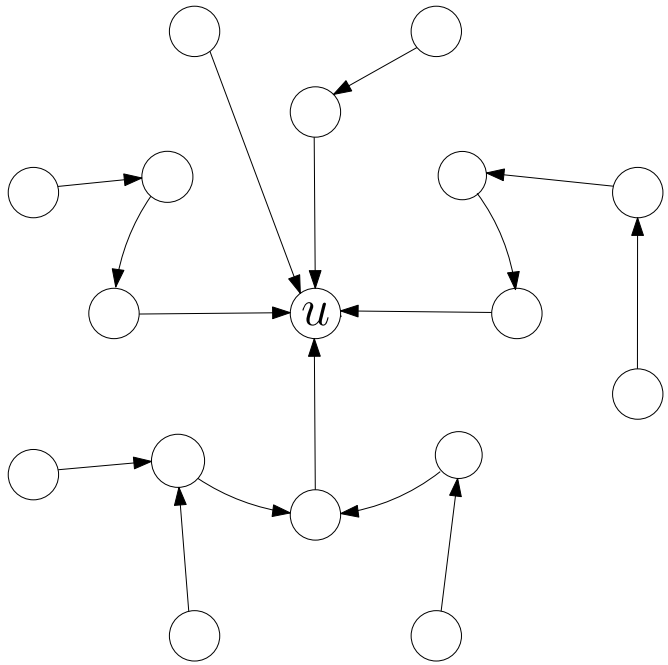
GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



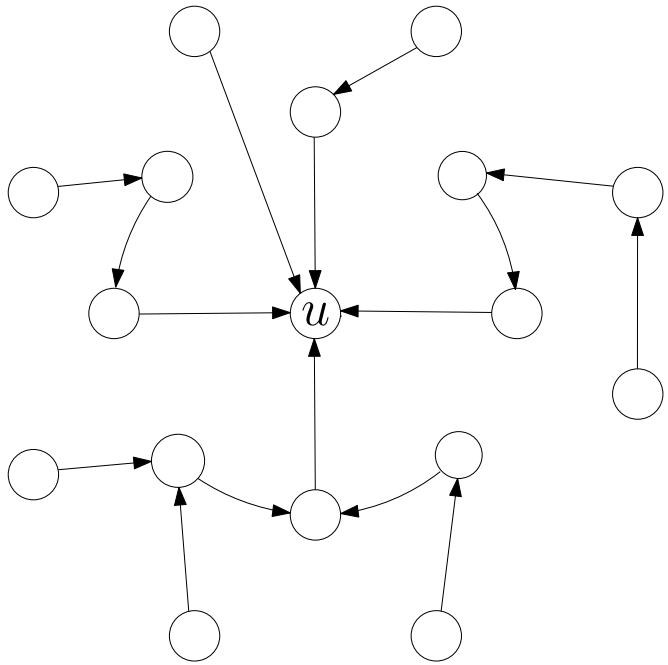
GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



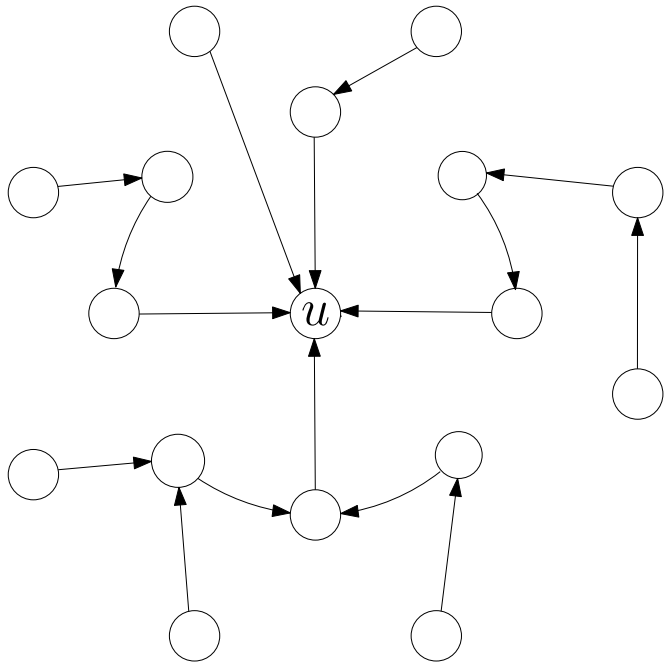
GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



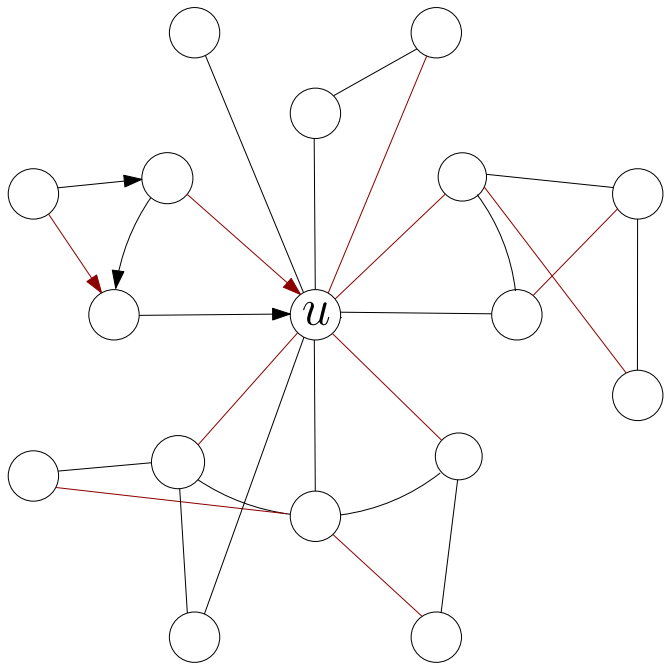
GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



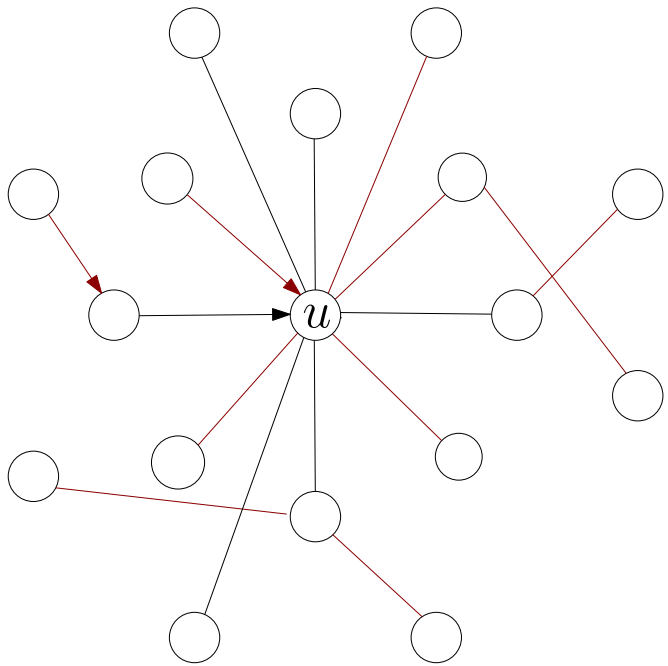
GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



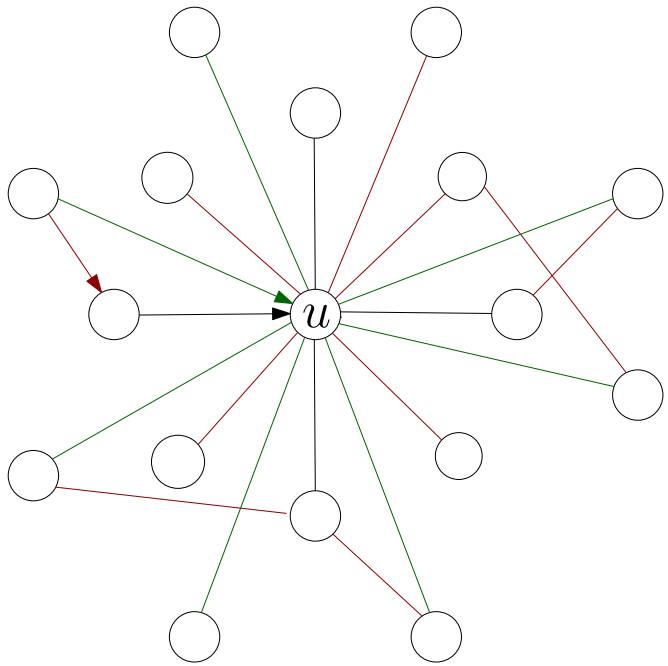
GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



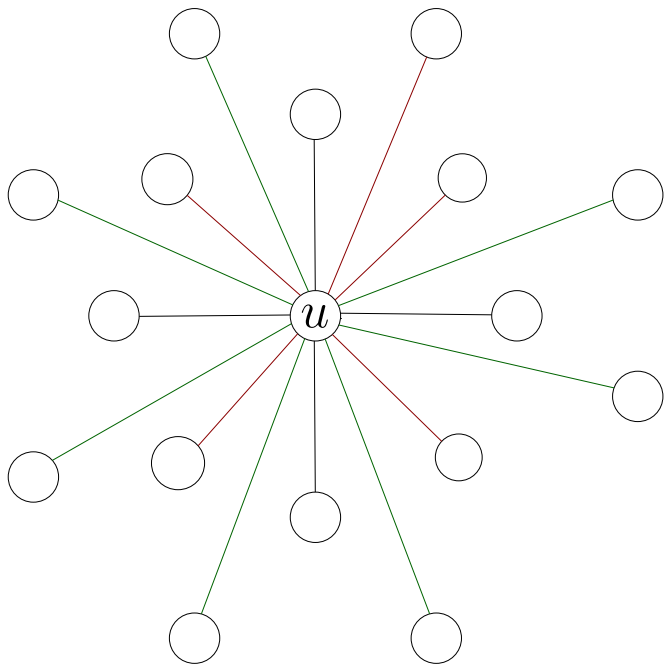
GraphToStar Algorithm

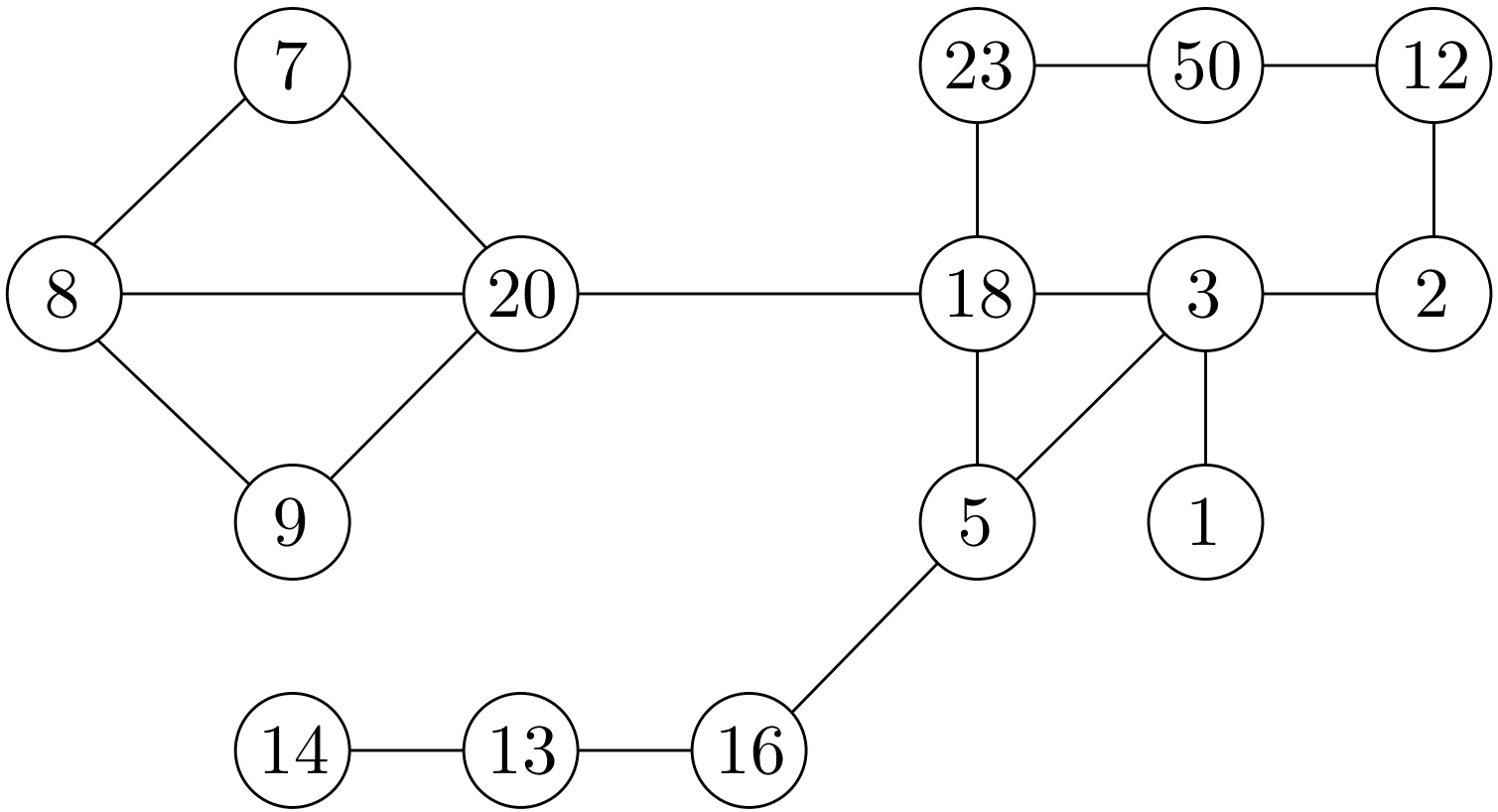
- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent

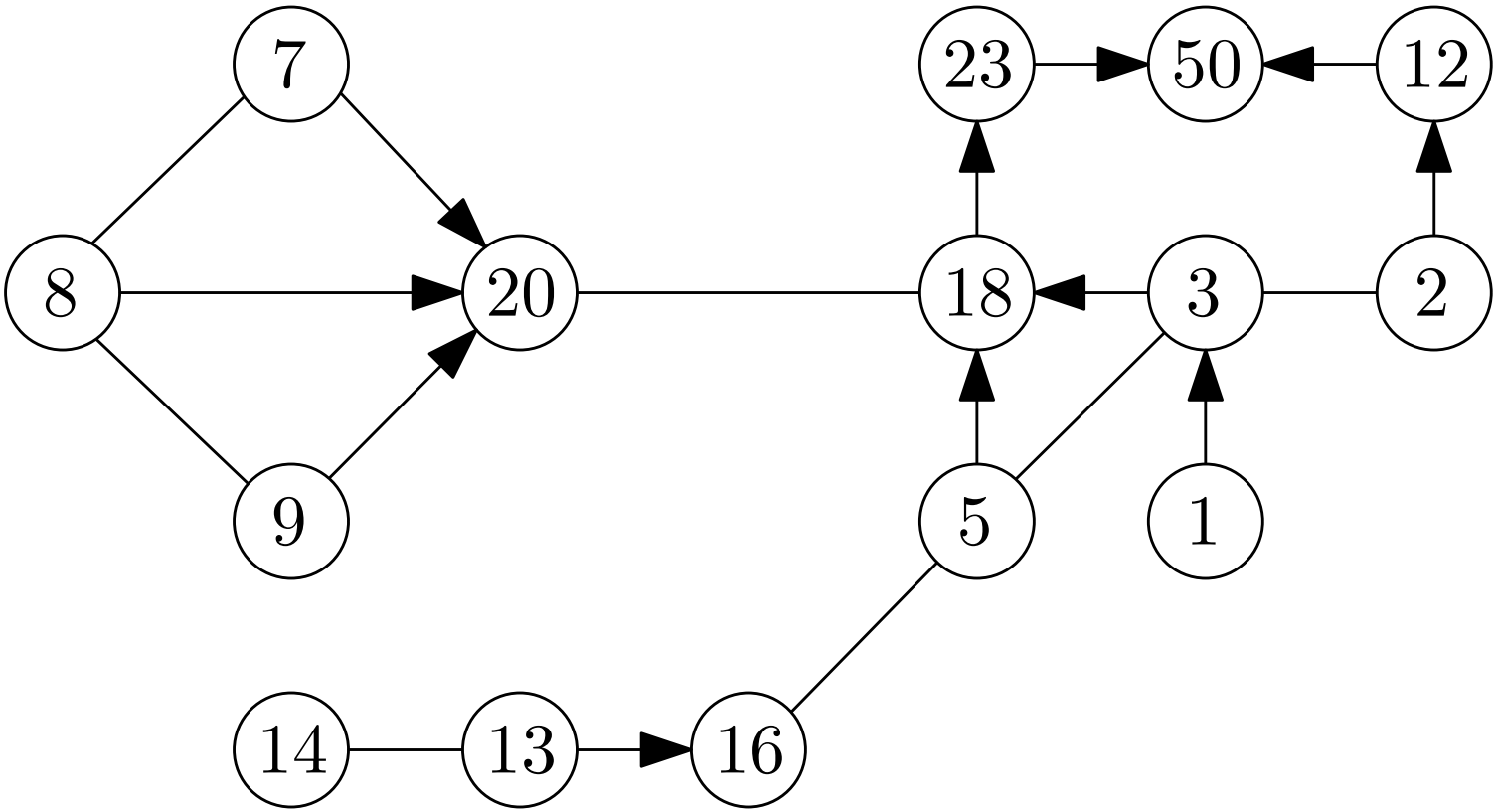


GraphToStar Algorithm

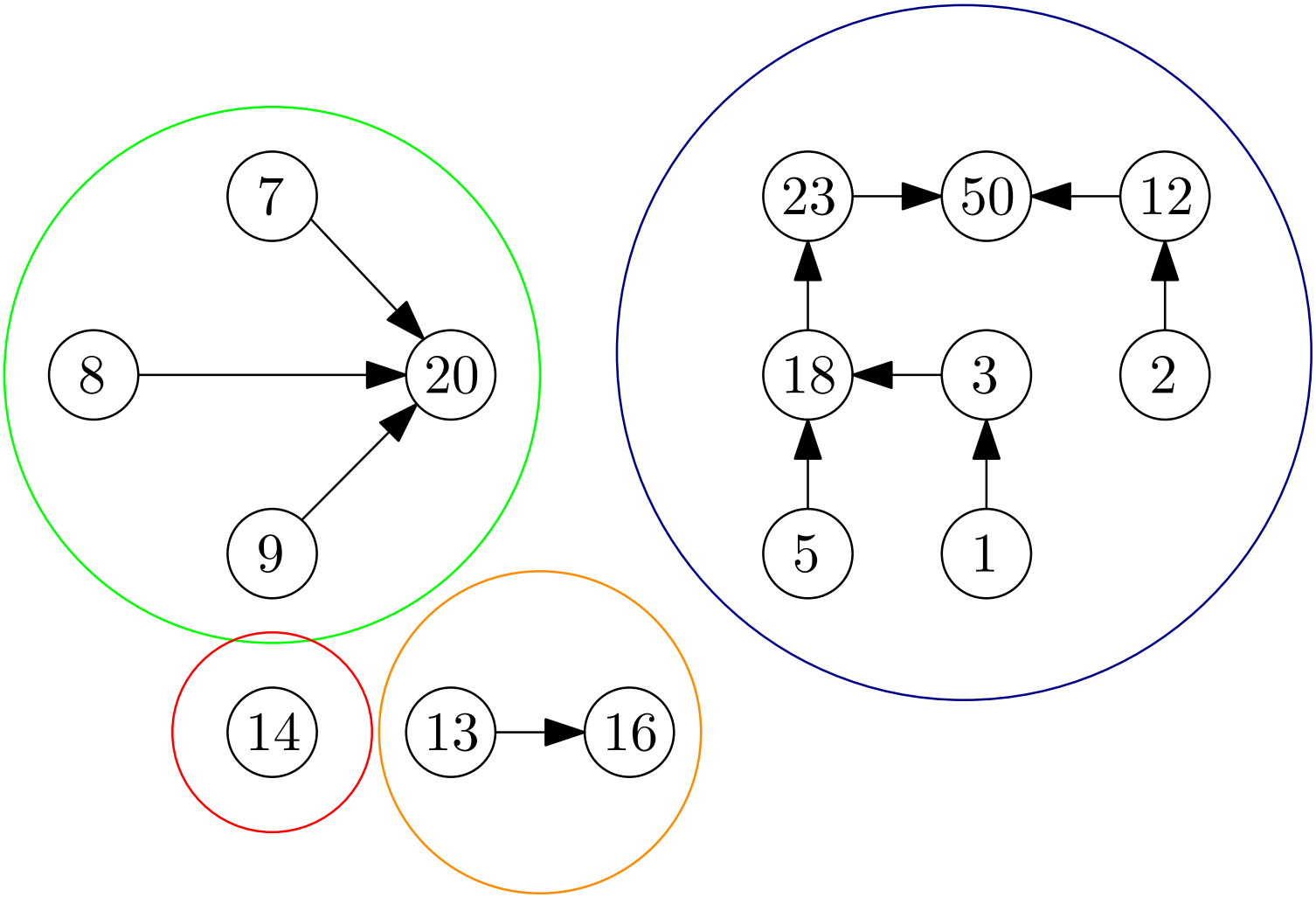
- Gadget network: Star
- TreeToStar subroutine
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



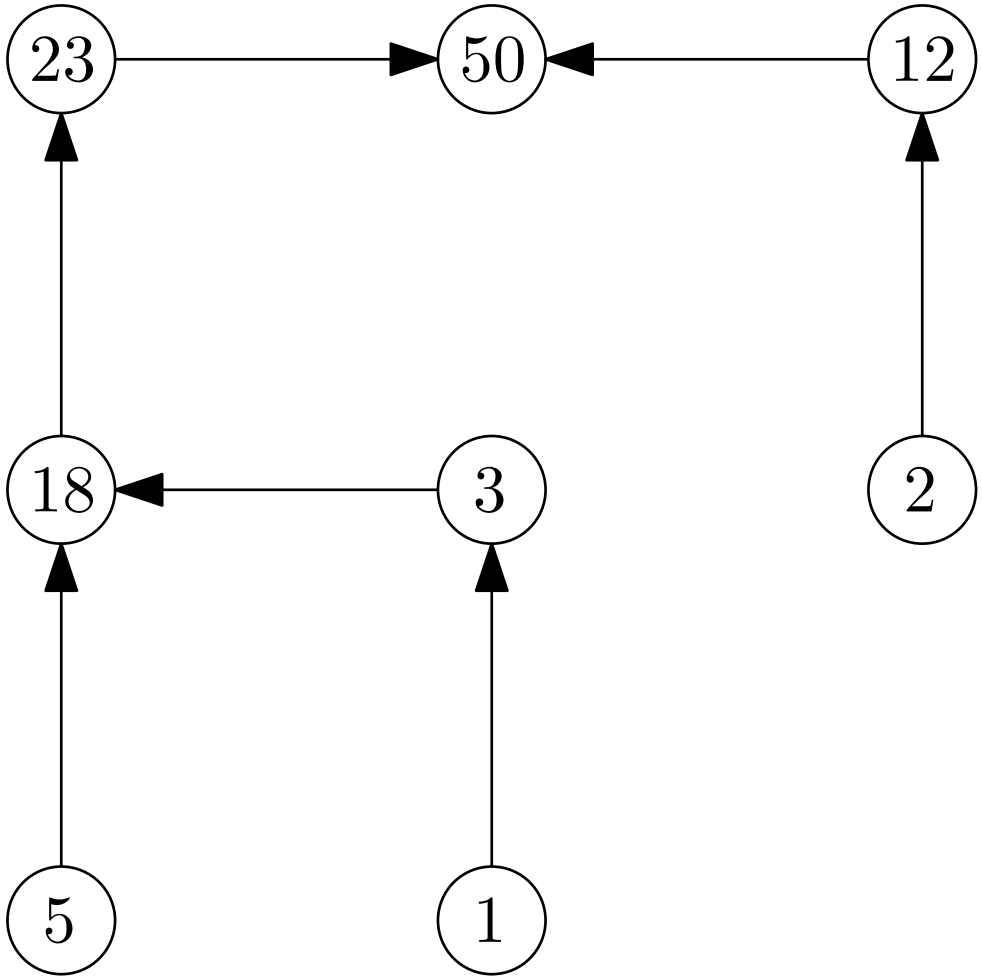

7
23
50
12
8
20
18
3
2
9
5
1
14
13
16



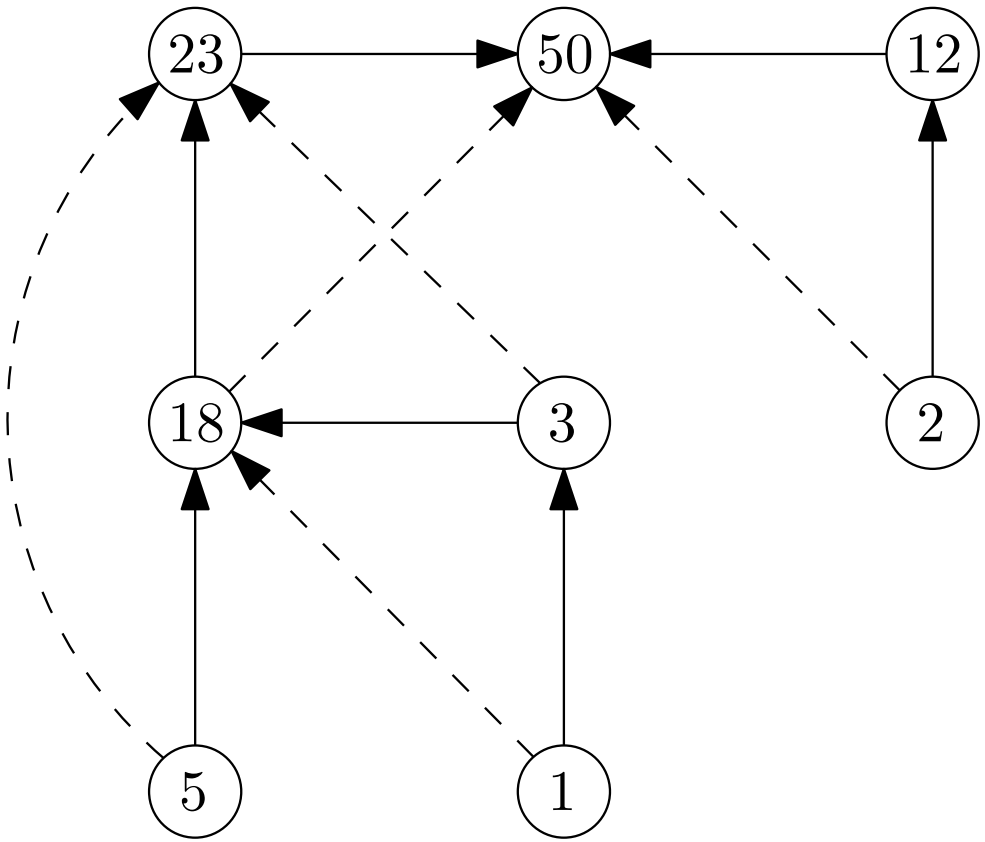
- **Selection:** Each committee leader u selects the neighboring committee leader v with the highest UID, where $UID_v > UID_u$.



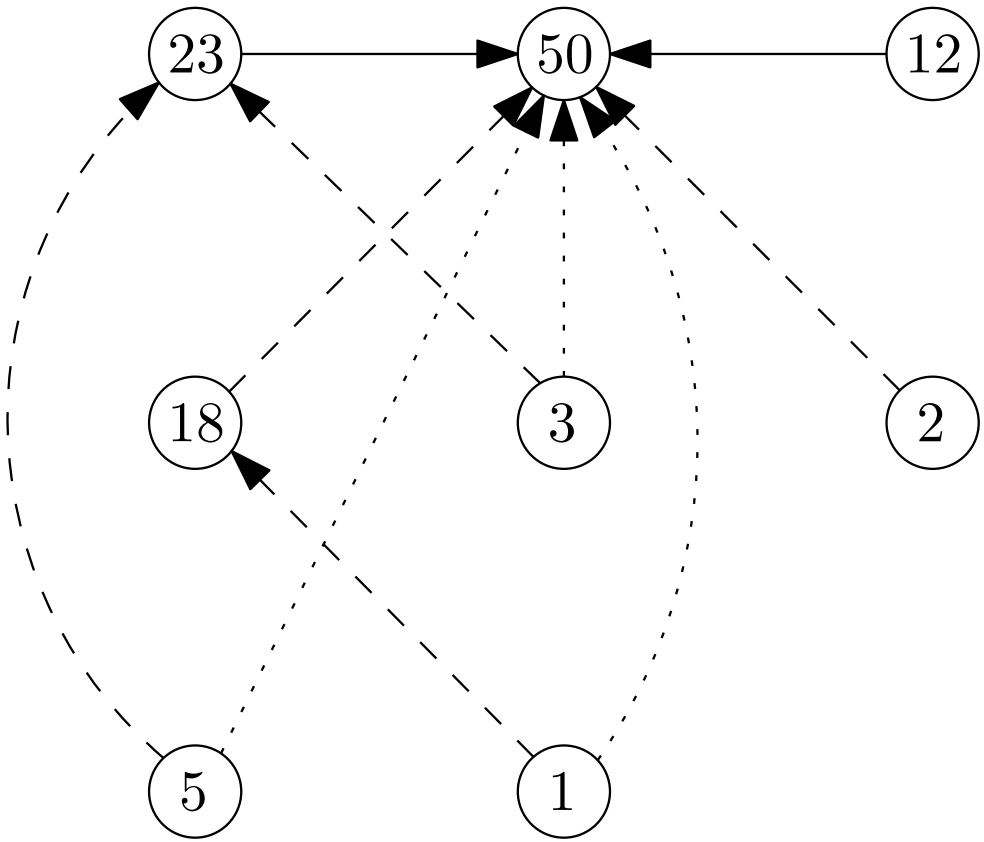
- **Partitioning:** We can partition the graph into directed trees



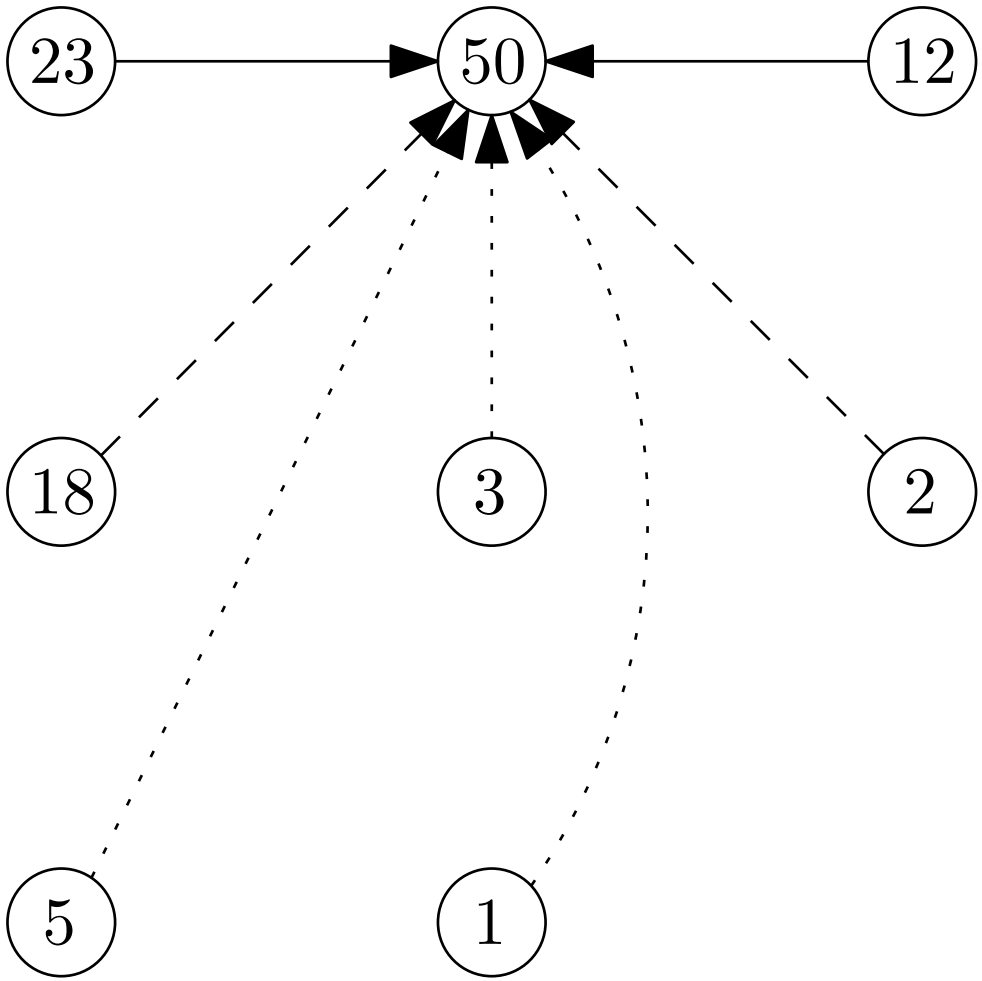
- **Pulling:** Each committee leader u activates an edge with the next committee leader v.



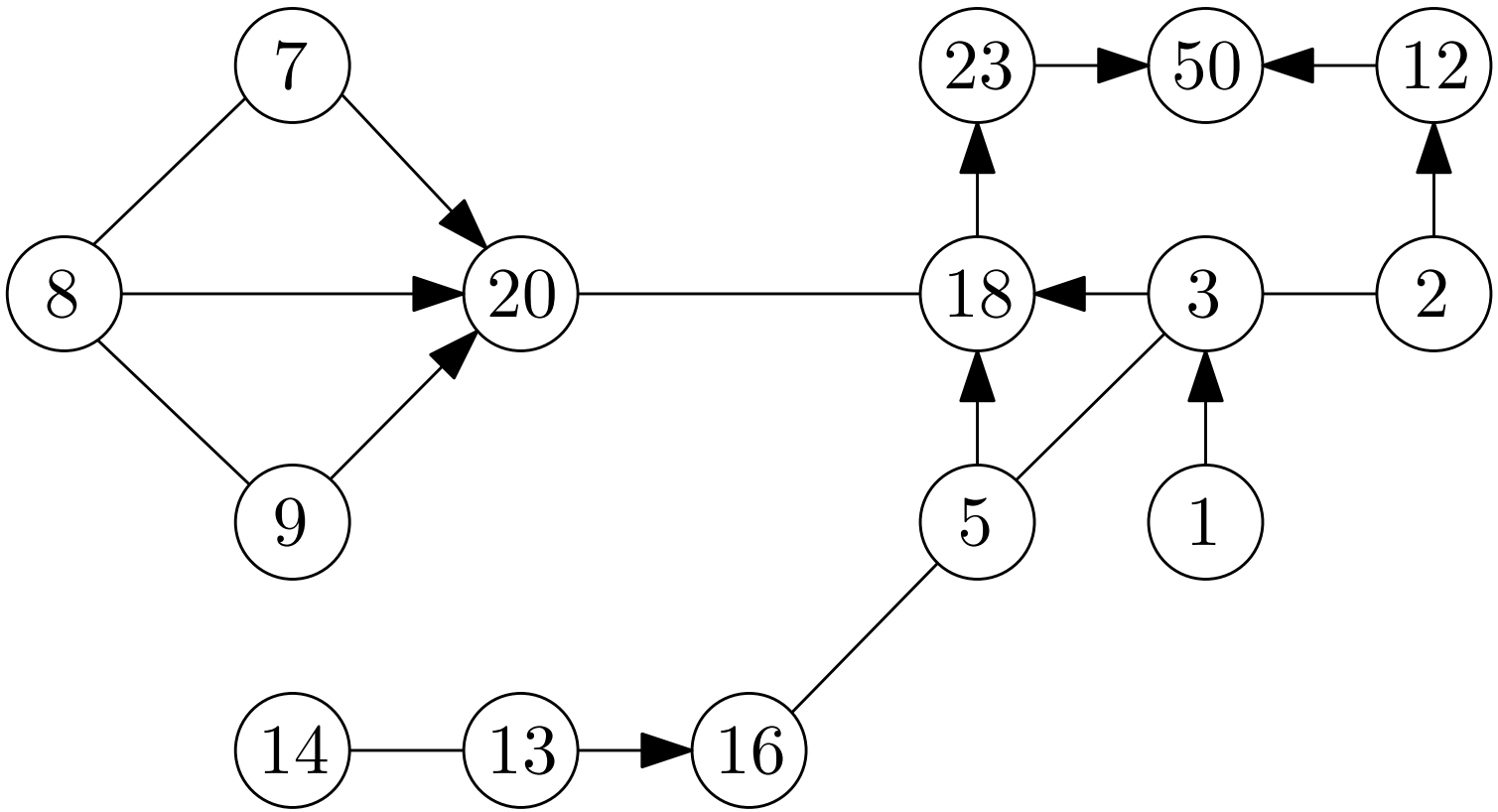
- **Pulling:** Each committee leader u activates an edge with the next committee leader v.



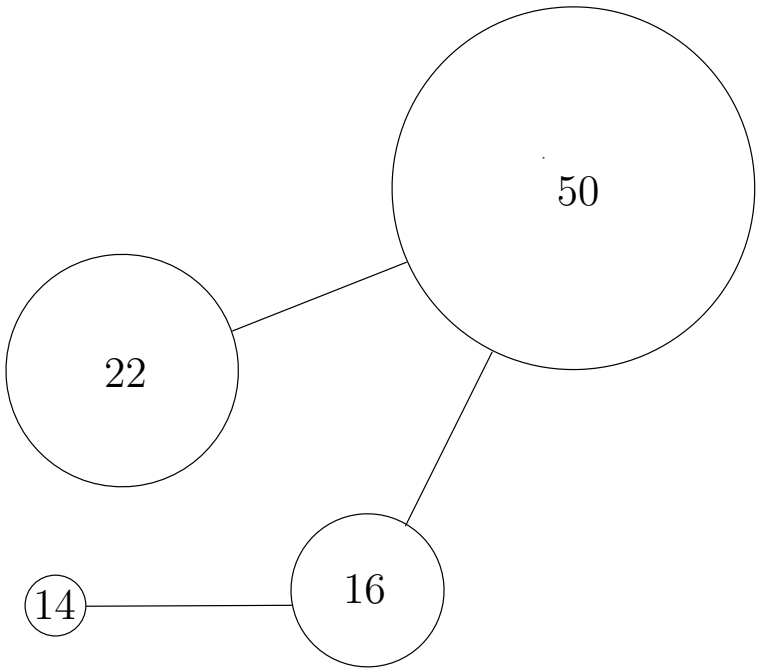
- **Pulling:** Each committee leader u activates an edge with the next committee leader v.



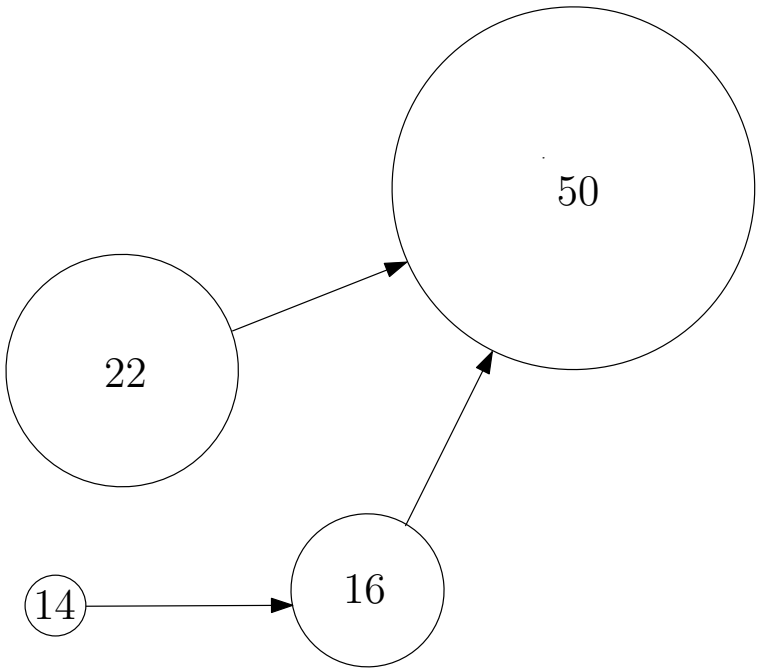
- **Pulling:** Each committee leader u activates an edge with the next committee leader v.


7
23
50
12
8
20
18
3
2
9
5
1
14
13
16

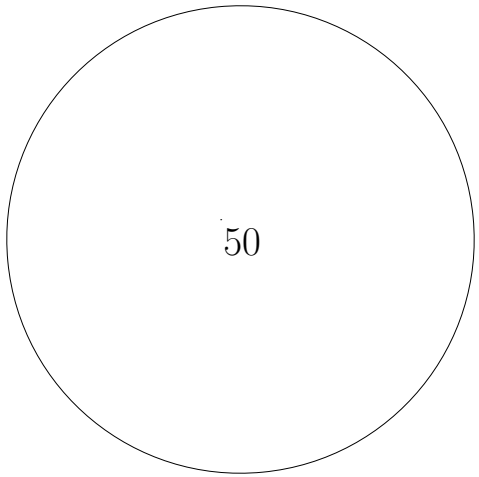
High Level

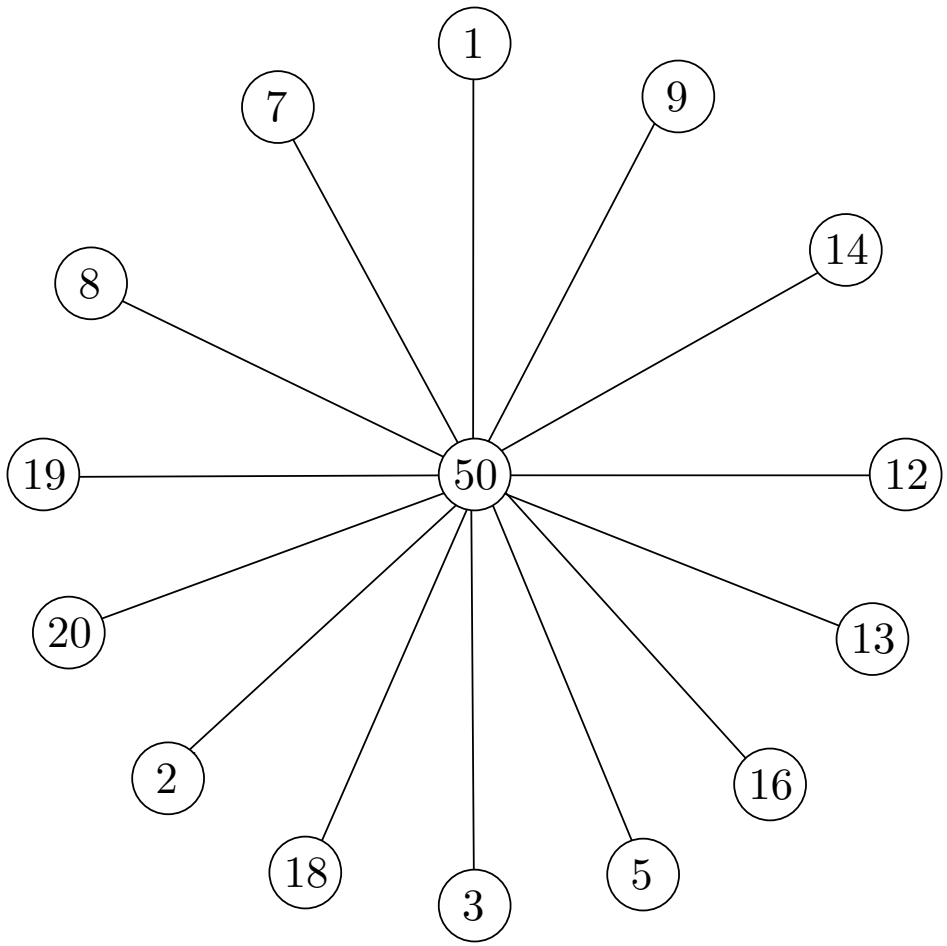


High Level



High Level




1
7
9
14
8
19
50
12
20
13
2
16
18
3
5

GraphToStar Proof Sketch

Theorem: For any initial connected graph G_s, the GraphToStar algorithm solves the Depth-1 Tree problem in $O(\log n)$ time with at most $O(n \log n)$ total edge activations and $O(n)$ active edges per round

Correctness

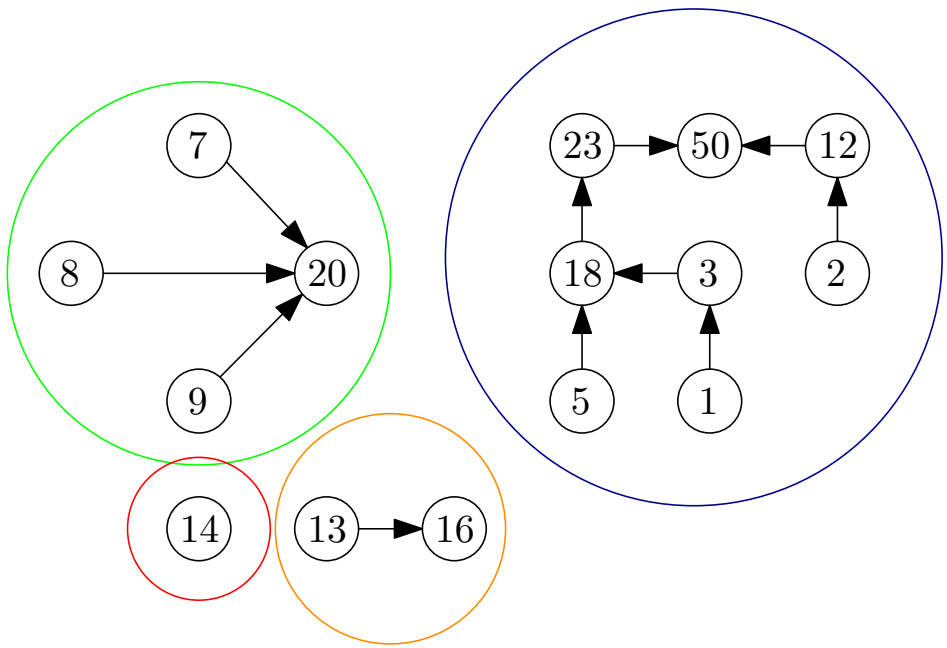
- Committees keep merging
- Can't get isolated indefinitely

Time Complexity

- Committees "double" in size
- Isolated

Edge Complexity

- Very simple

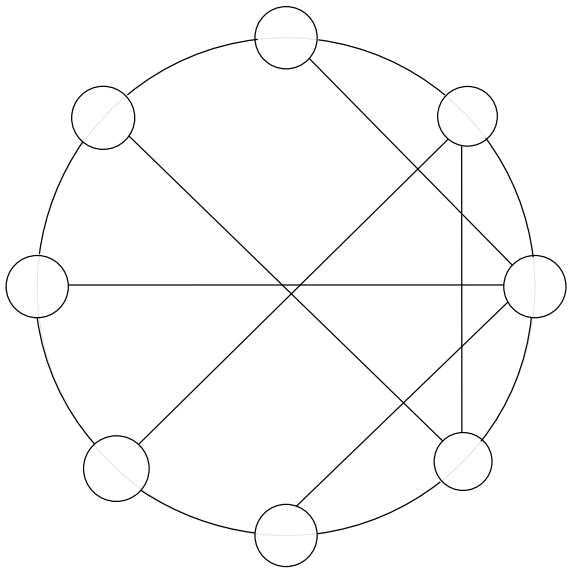


GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children

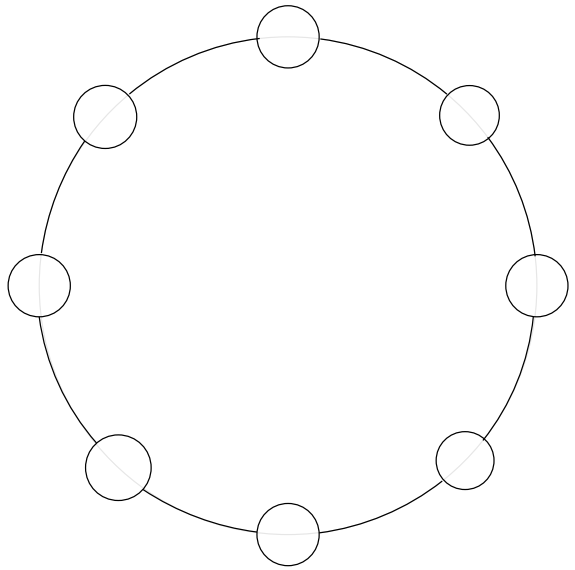
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



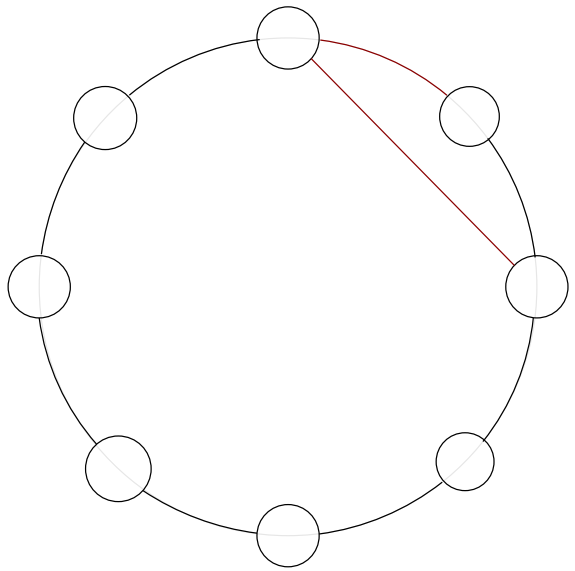
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



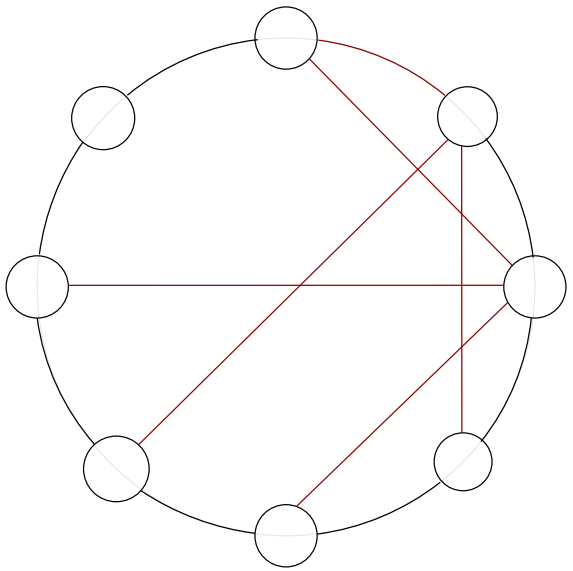
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



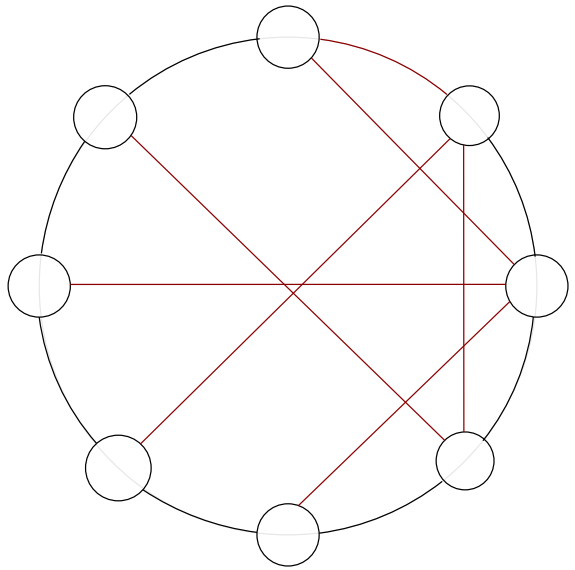
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



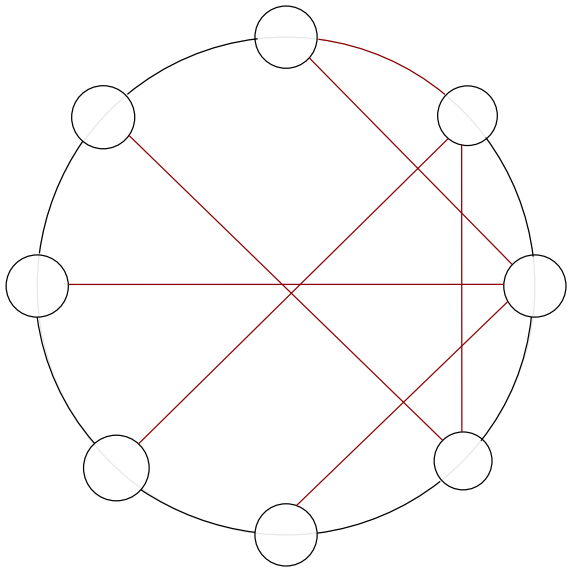
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



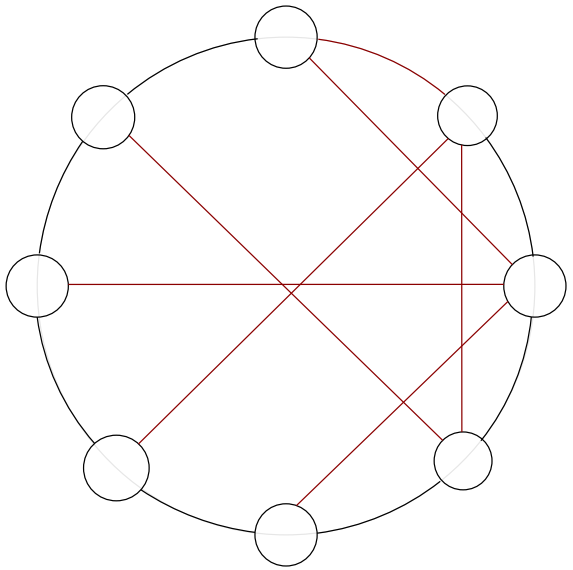
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



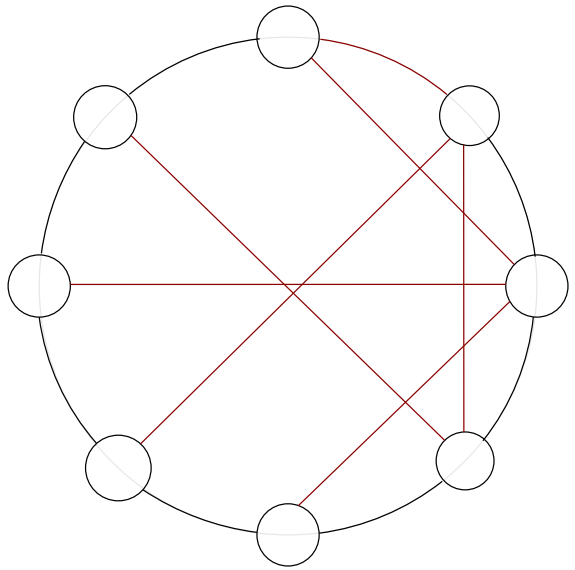
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



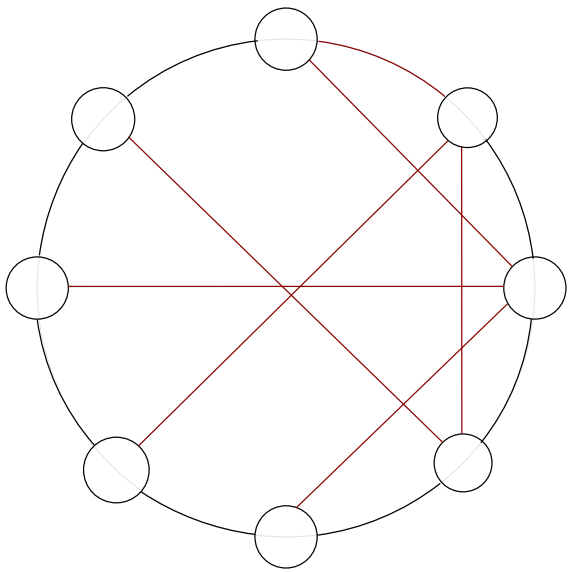
GraphToWreath Algorithm

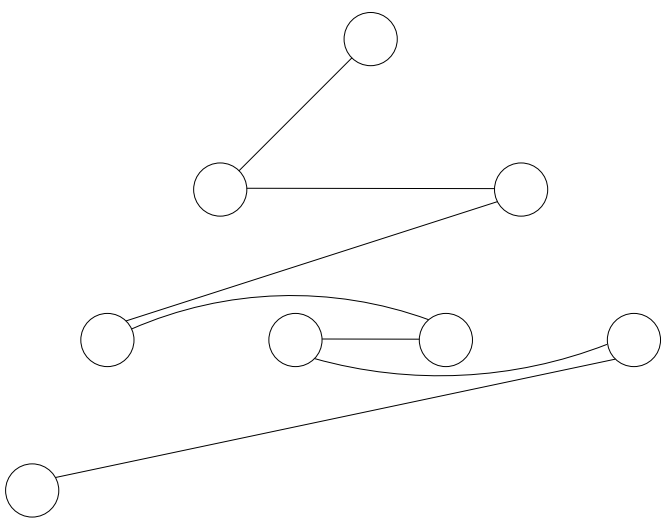
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



GraphToWreath Algorithm

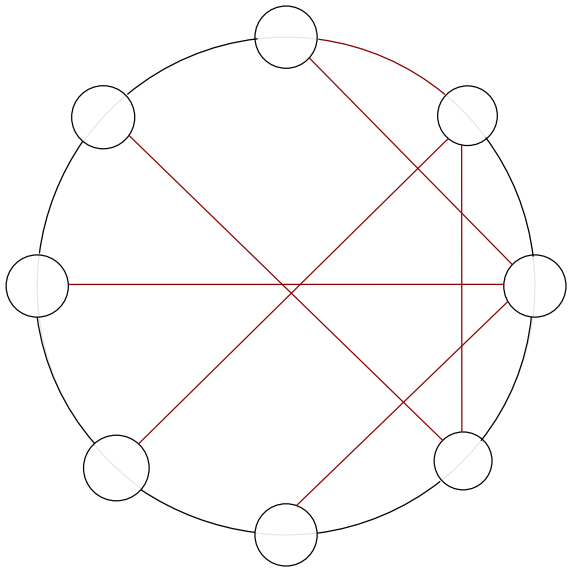
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children

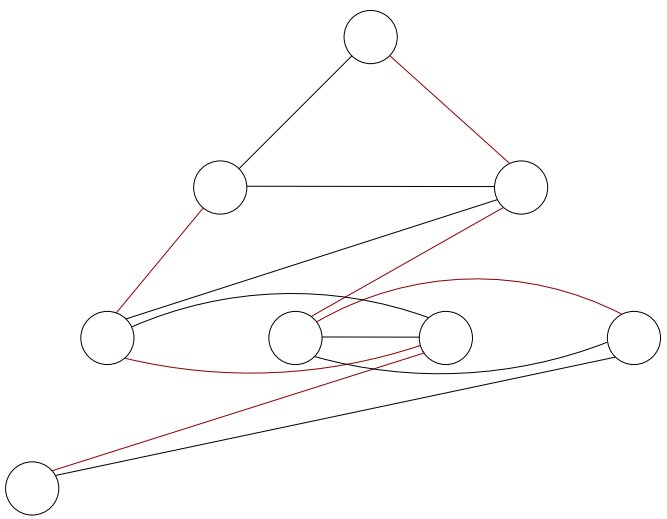




GraphToWreath Algorithm

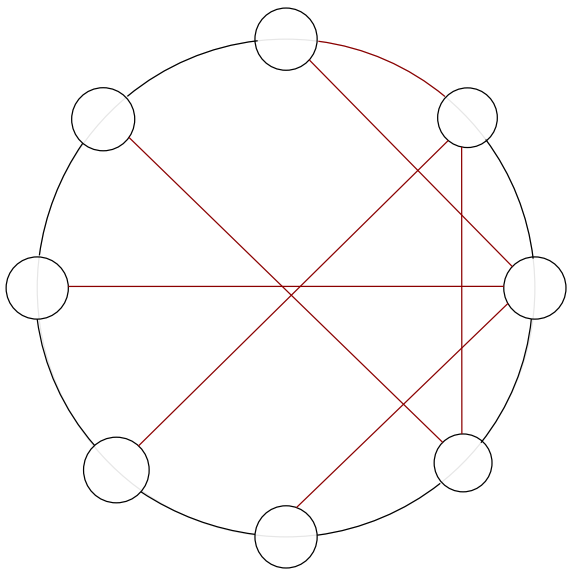
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children

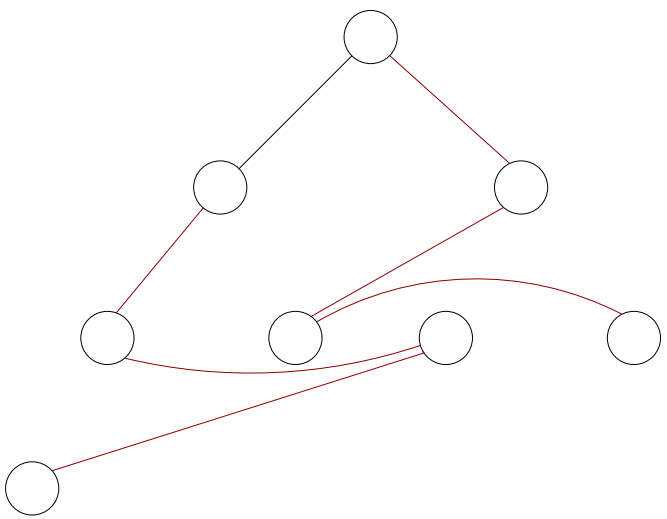




GraphToWreath Algorithm

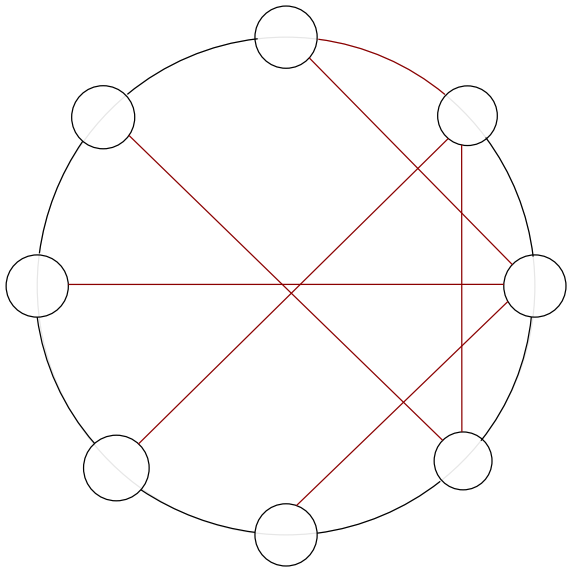
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children

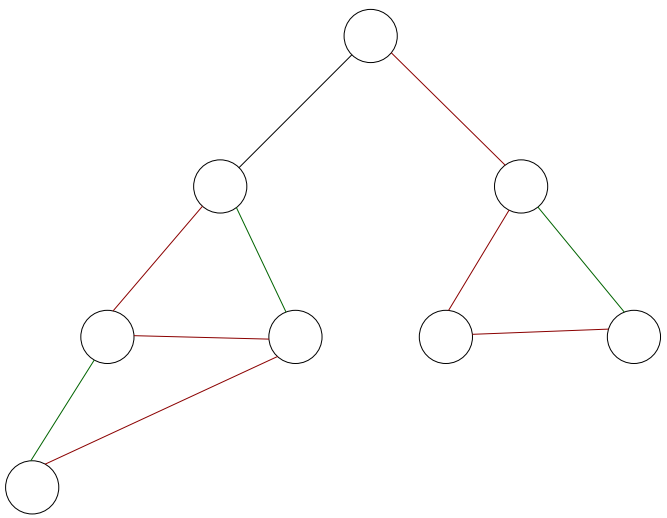




GraphToWreath Algorithm

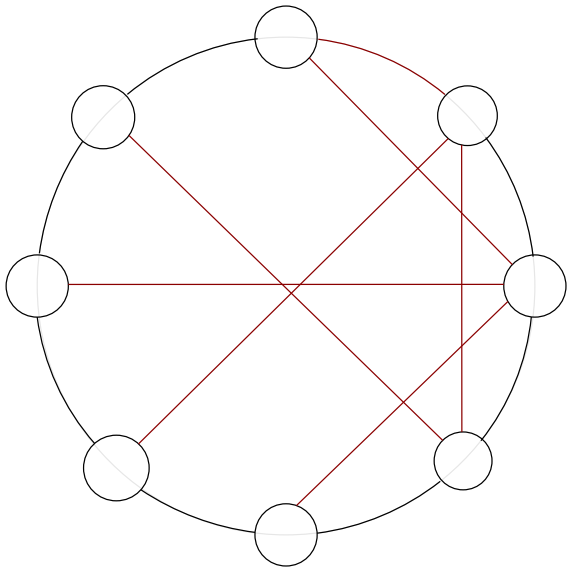
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children

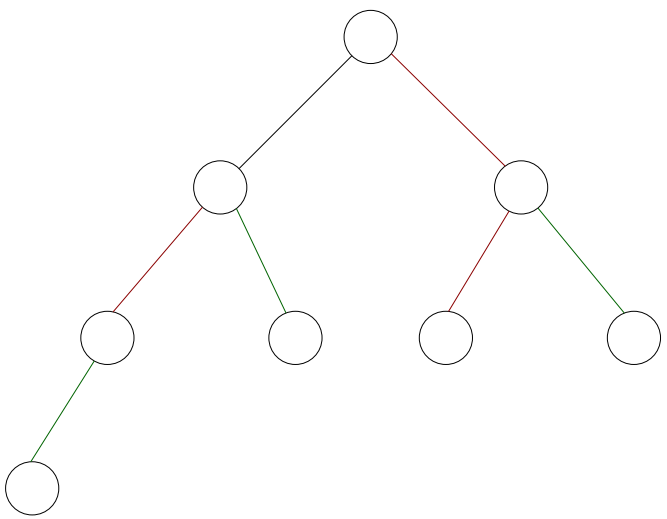


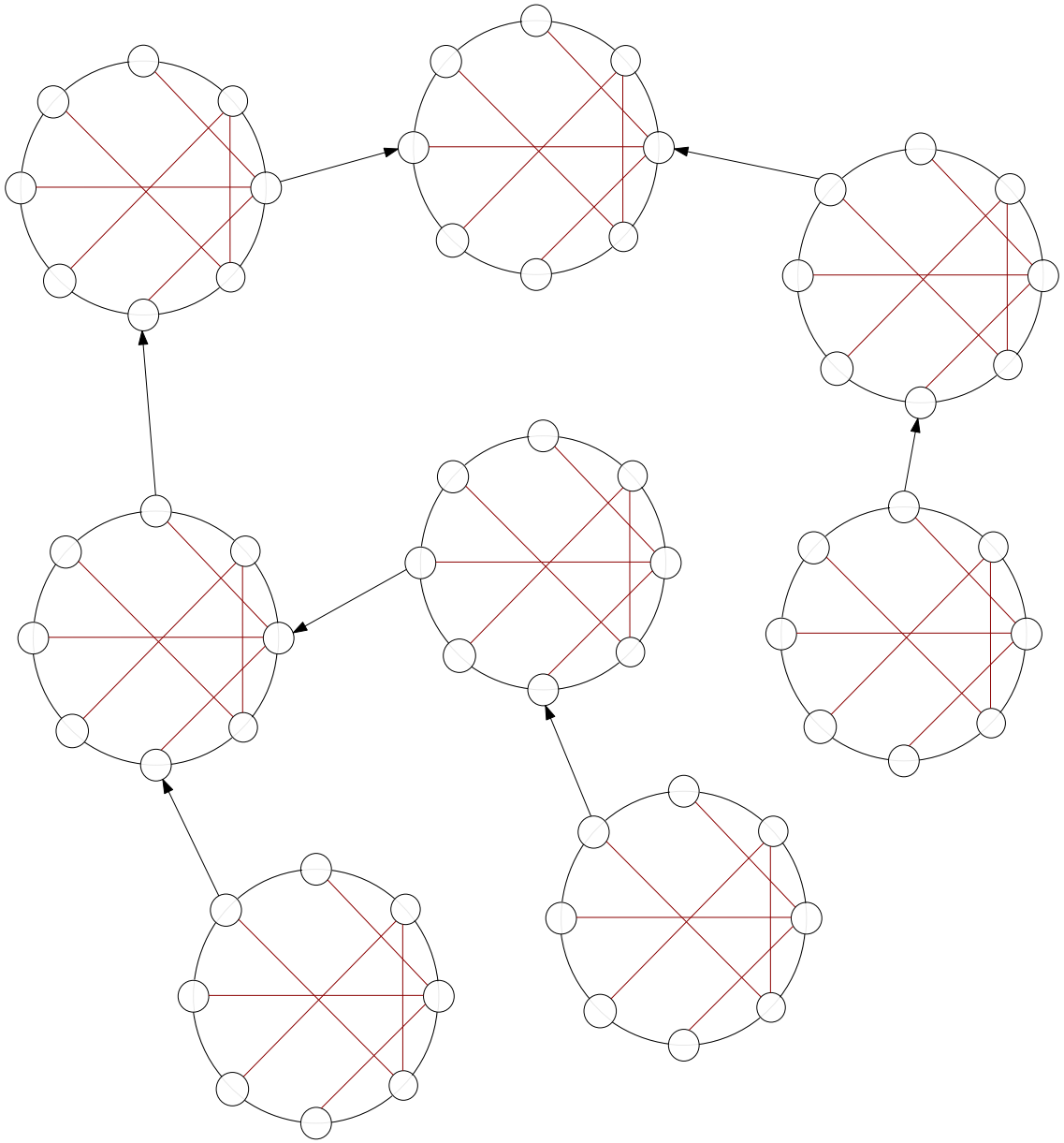


GraphToWreath Algorithm

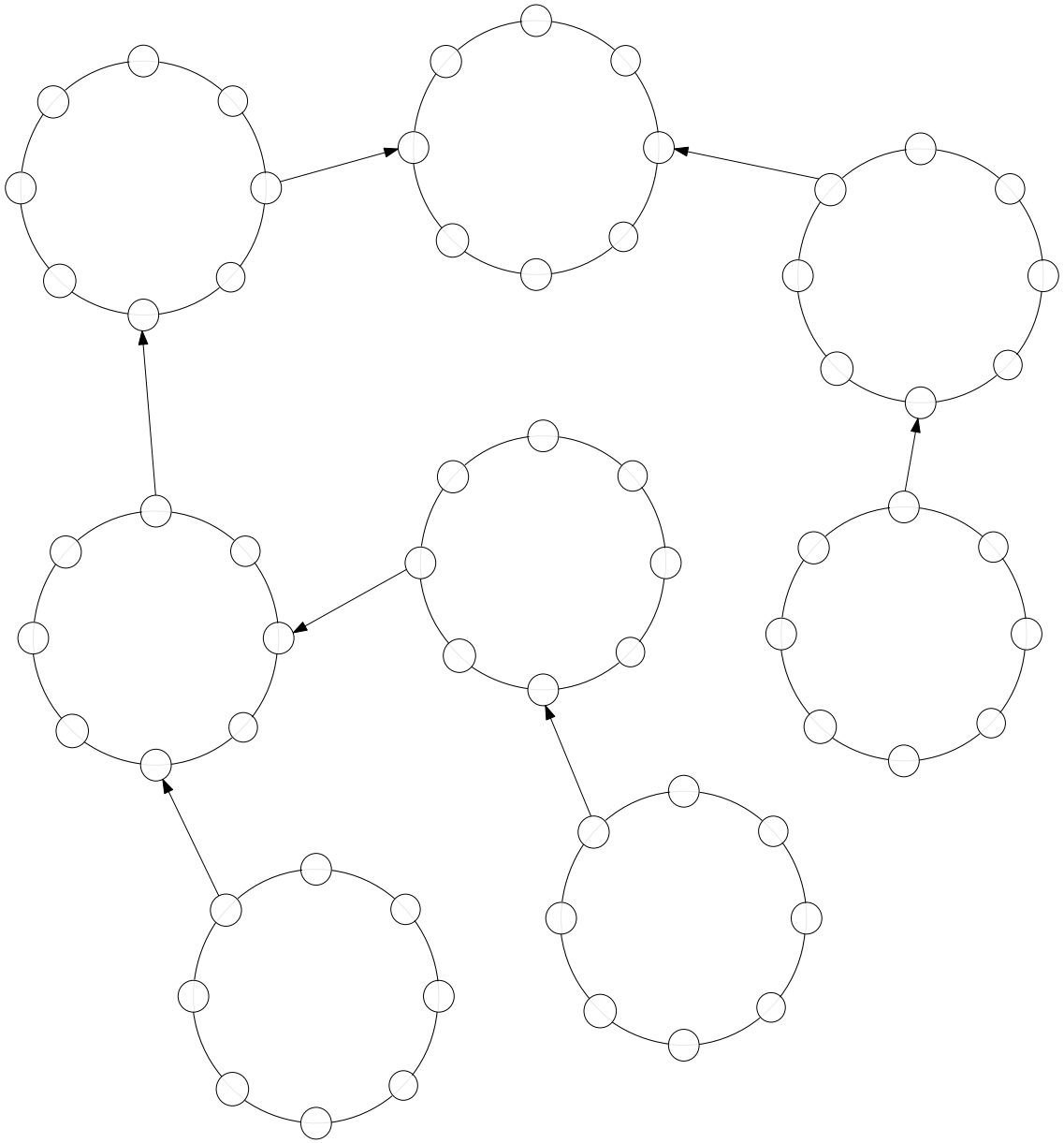
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine
- Transforms any initial oriented line into a spanning Complete Binary Tree in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children



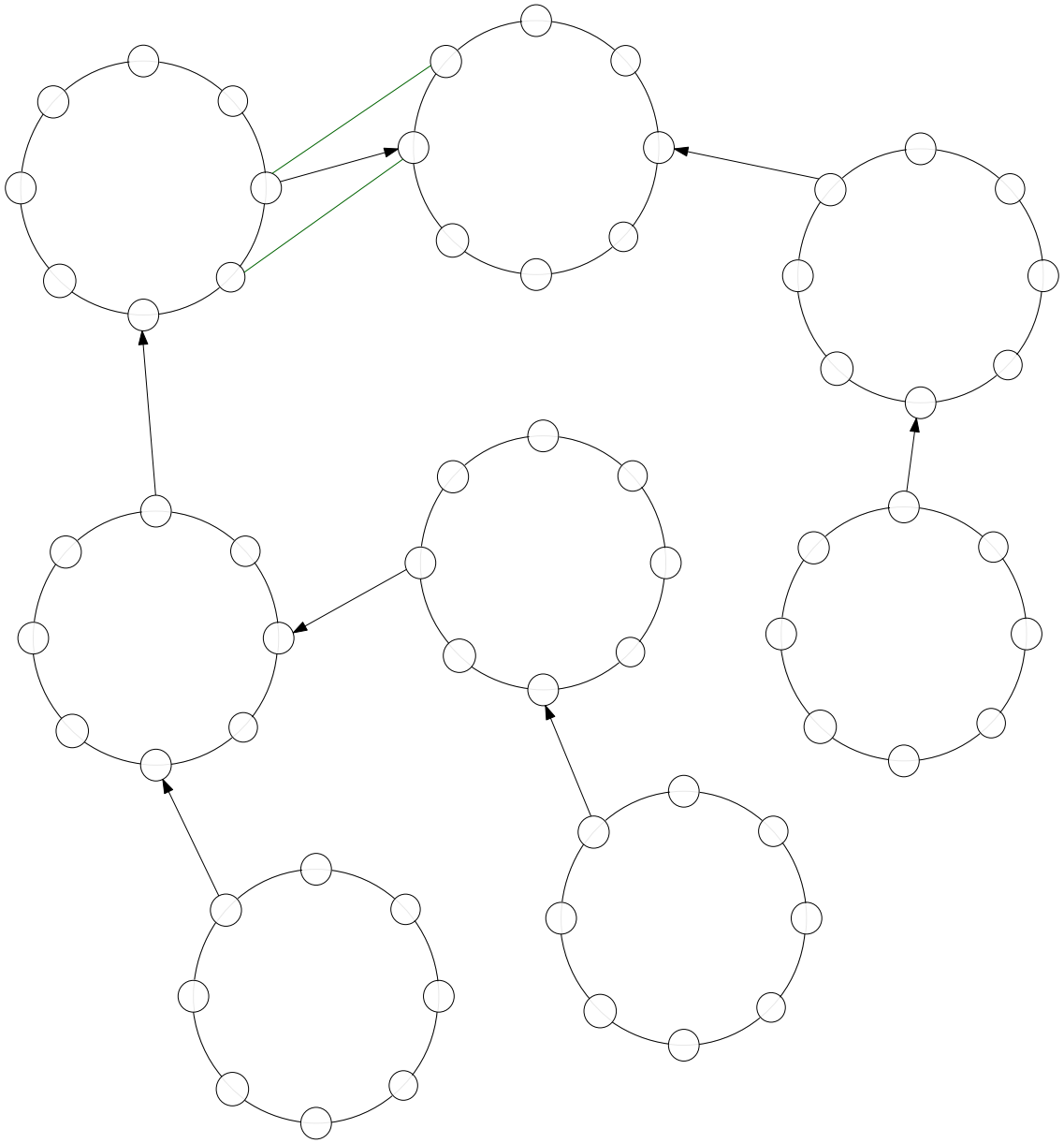


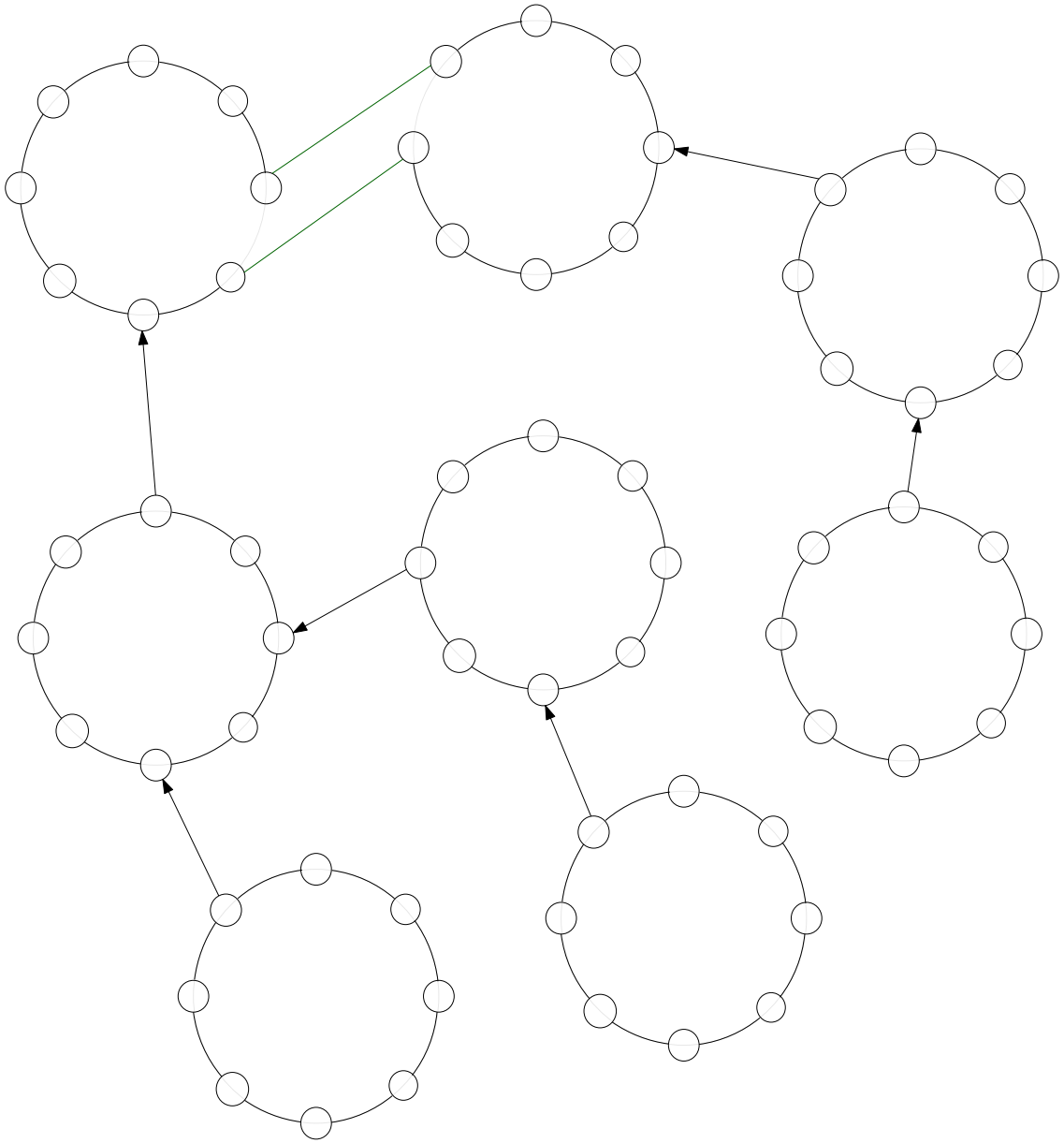


Low Level

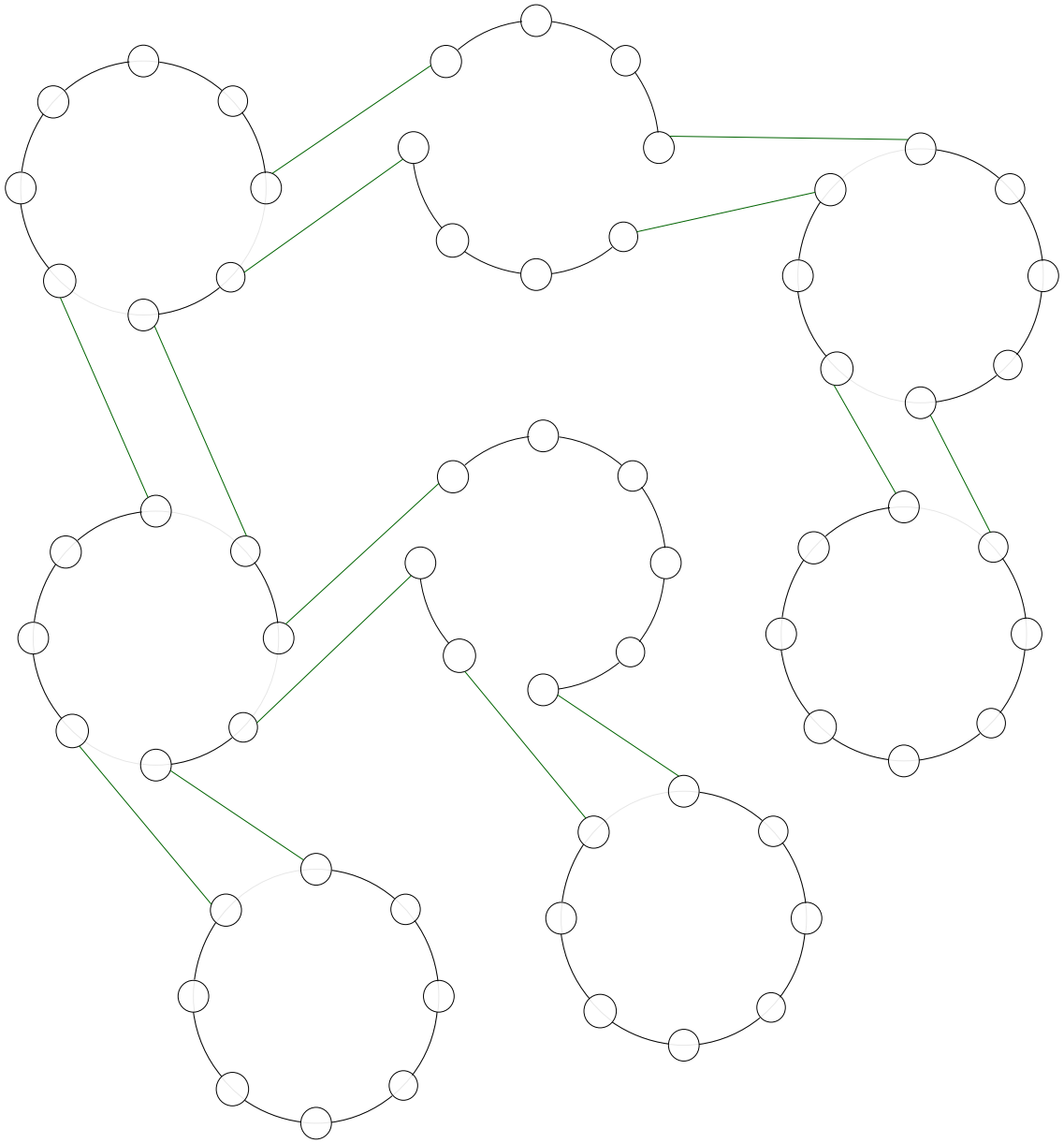


Low Level

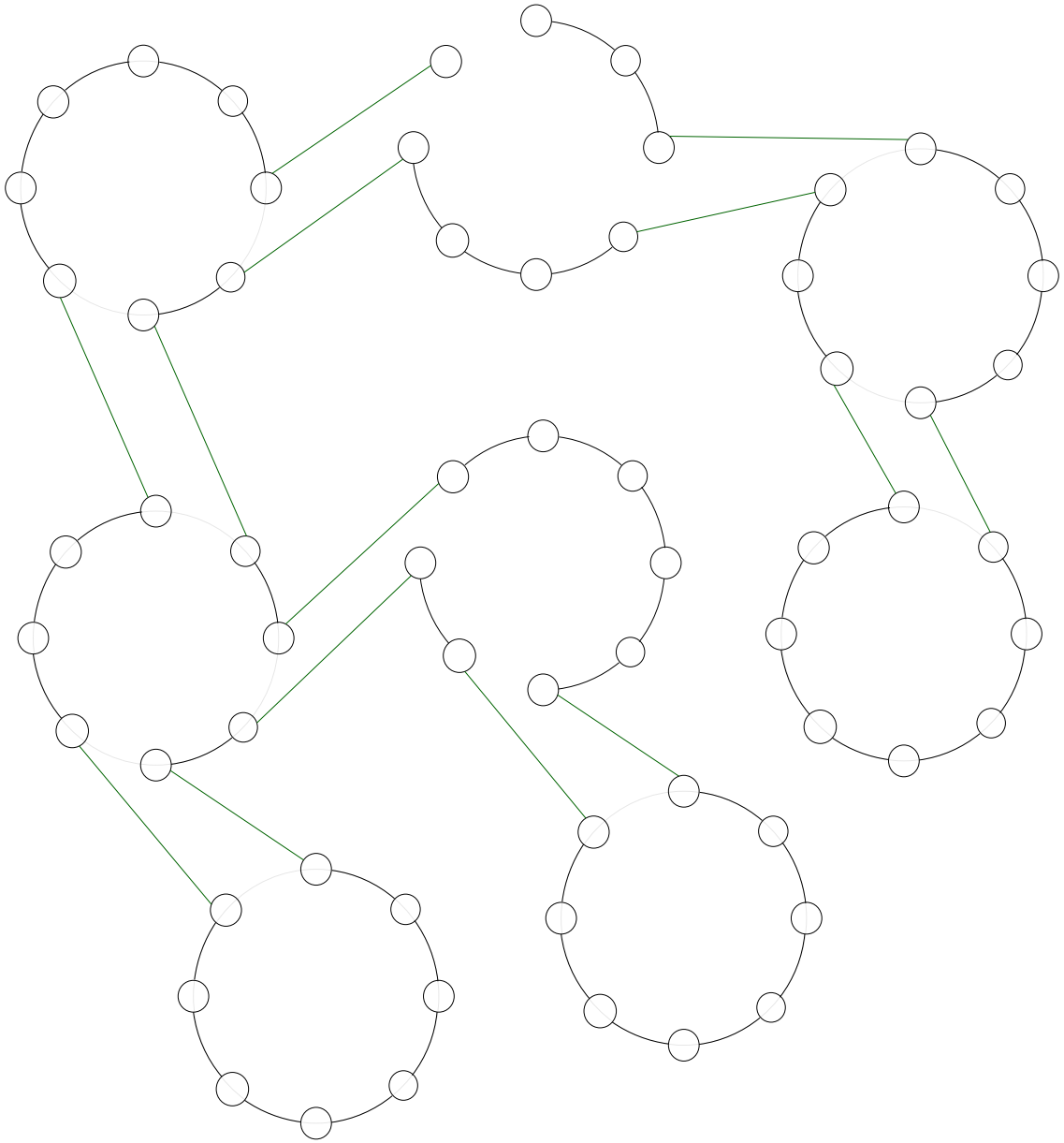




Low Level



Low Level



Our Results

Algorithm	Time	Total Edge Activations	Maximum Activated Edges	Maximum Activated Degree
GraphToStar	$O(\log n)$	$O(n \log n)$	$O(n)$	$O(n)$
GraphToWreath	$O(\log^2 n)$	$O(n \log^2 n)$	$O(n)$	$O(1)$
GraphToThinWreath	$O(\log^2 n / \log \log n)$	$O(n \log^2 n)$	$O(n)$	$O(\log^2 n)$

Table: Algorithms for Depth-d Tree

Bounds	Time	Total Edge Activations	Maximum Activated Edges	Maximum Activated Degree
Centralized Lower	$\Omega(\log n)$	$\Omega(n)$	$\Omega(n/\log n)$	
Centralized Upper	$O(\log n)$	$\Theta(n)$		
Distributed Lower	$O(\log n)$	$\Omega(n \log n)$		

Table: Bounds for Depth-d Tree

- Motivation: Abstraction of networked systems which, starting from a single entity, can grow into well-defined global networks and structures
- Not well studied apart from the Nubot model of Woods *et al.* [WCG13], which is a geometric model

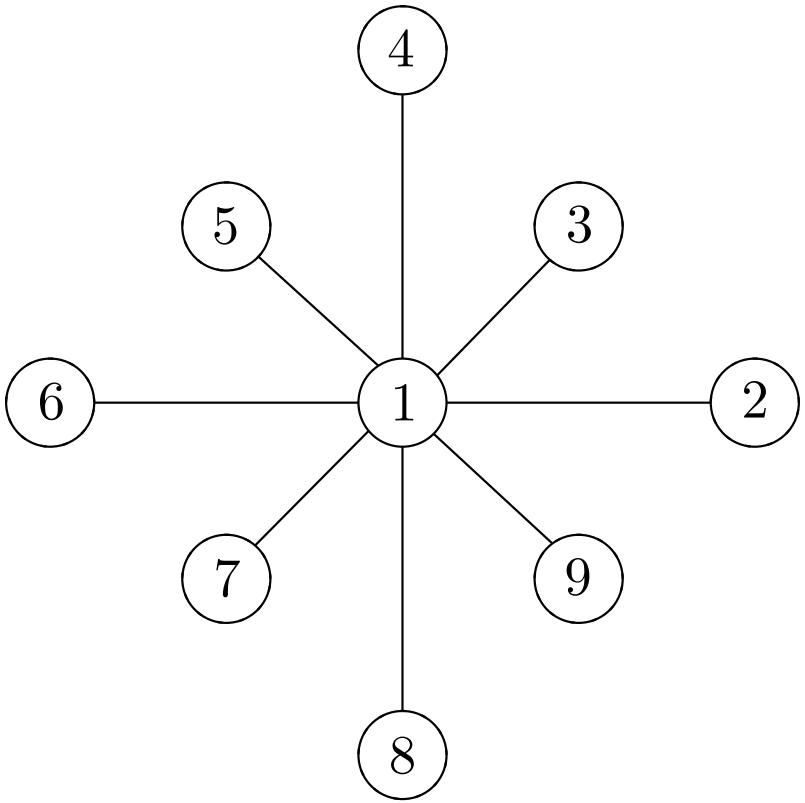
Our Focus

- Construct an input graph G starting from graph G_0 (graph with a single node) via node generations and edge activations
- Active Dynamics and Distributed Control
- BUT the centralized case is still unknown (and hard as it seems). So we will focus on the centralized case for now.

The Second Model

- Initial graph G_0
- Node generation: In every round, each node u can generate(give birth) another node v and activate edge uv
- Edge activation: At the time of its birth, node v can activate edges with nodes that are within its edge-activation distance d at the time of its birth
- Edge deletion. In every round, any node u can decide to delete some of its incident edges

An example



$a) 1 \to 2$

$b) 1 \to 4$

$c) 1 \to 6$

$d) 1 \to 8$

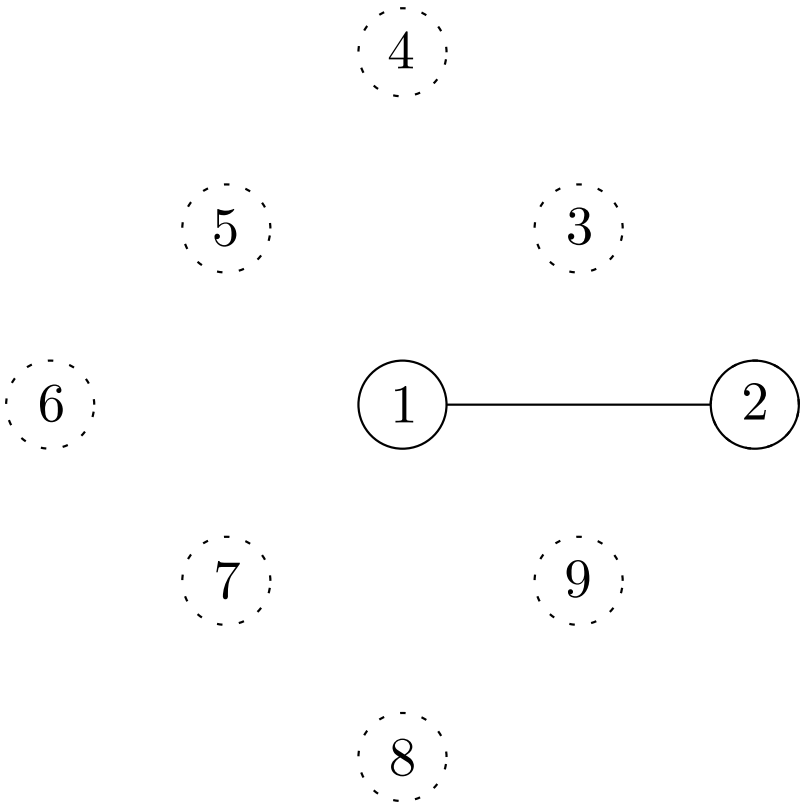
$e) 1 \to 9$

$f) 1 \to 3$

$g) 1 \to 5$

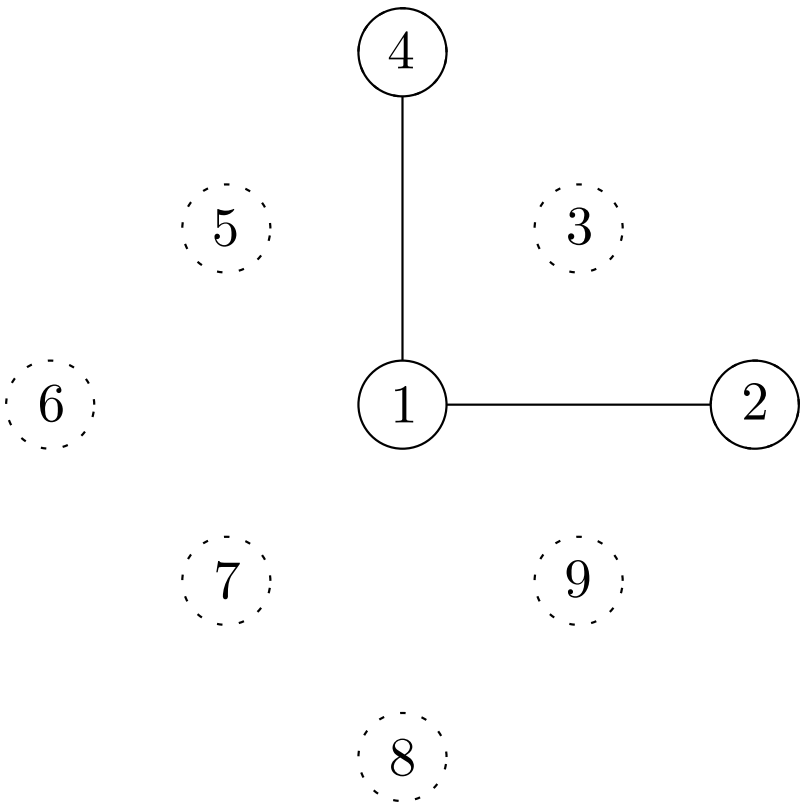
$h) 1 \to 7$

An example



$a) 1 \to 2$

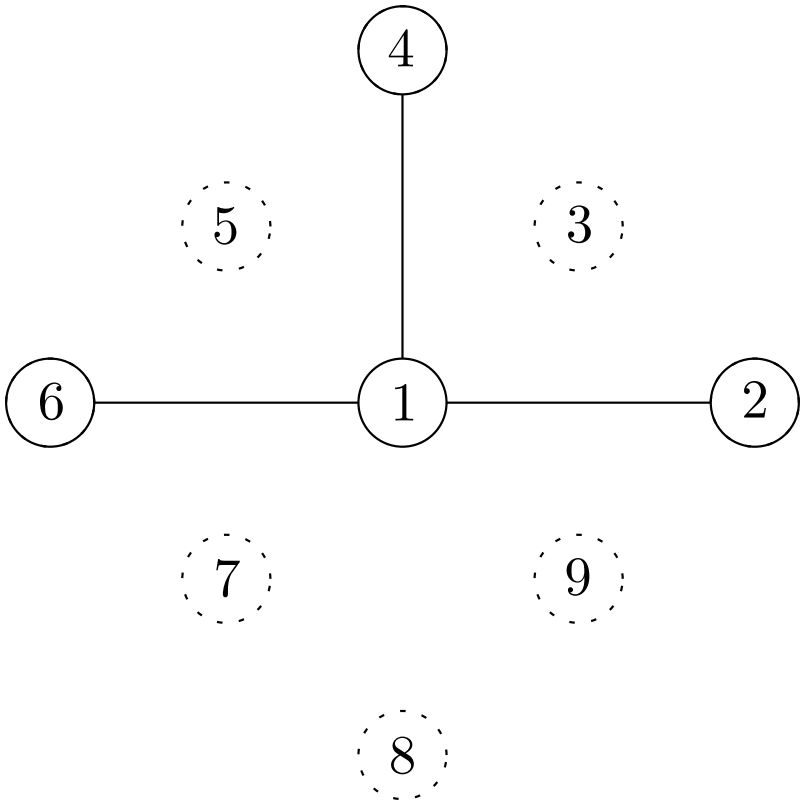
An example



$a)1 \to 2$

$b)1 \to 4$

An example

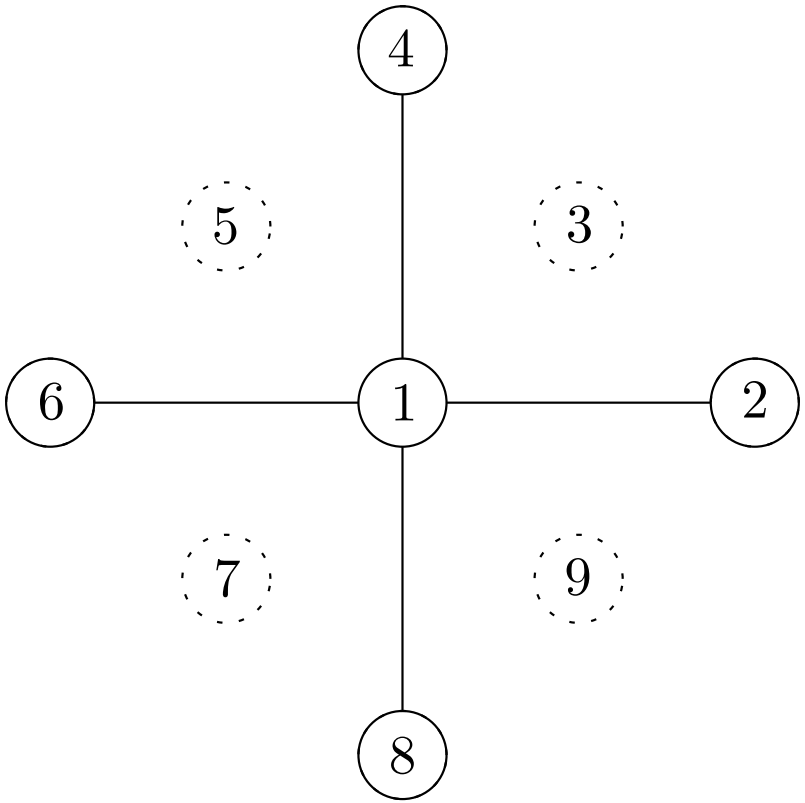


$a) 1 \to 2$

$b) 1 \to 4$

$c) 1 \to 6$

An example



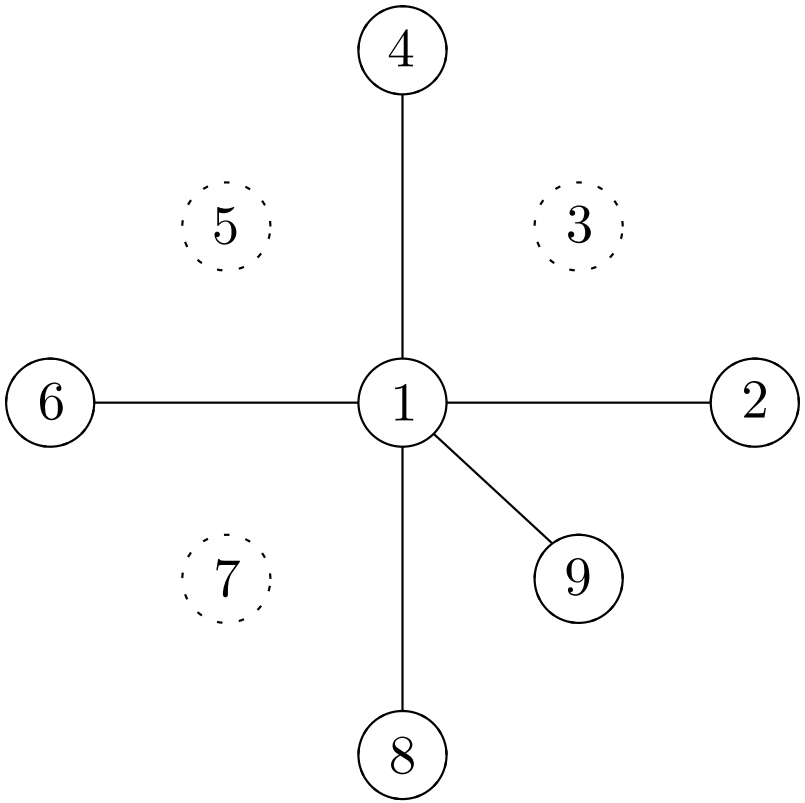
$a) 1 \to 2$

$b) 1 \to 4$

$c) 1 \to 6$

$d) 1 \to 8$

An example



$a) 1 \to 2$

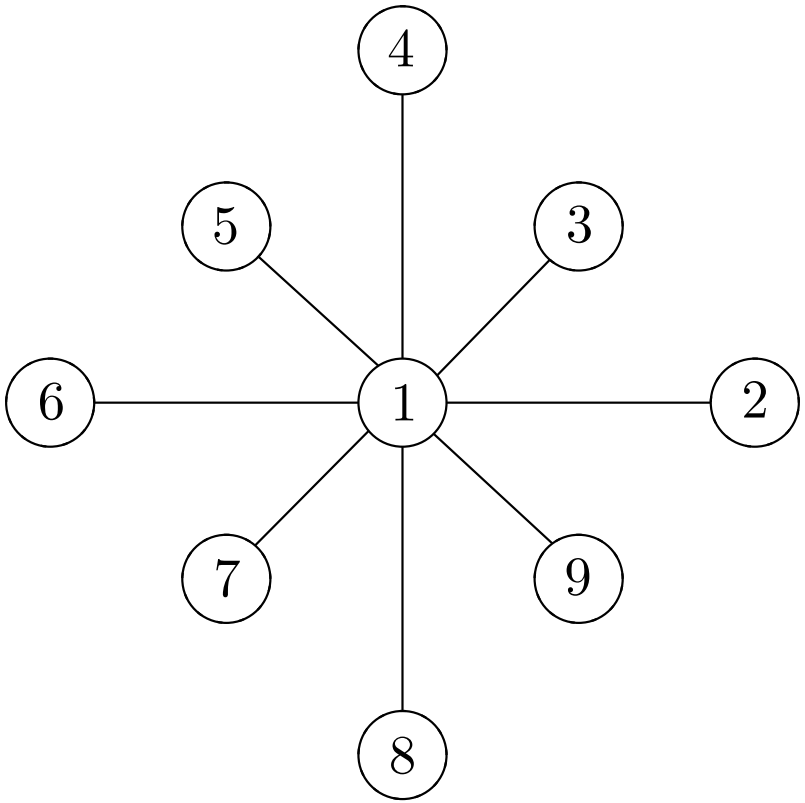
$b) 1 \to 4$

$c) 1 \to 6$

$d) 1 \to 8$

$e) 1 \to 9$

An example



$a)1 \to 2$

$b)1 \to 4$

$c)1 \to 6$

$d)1 \to 8$

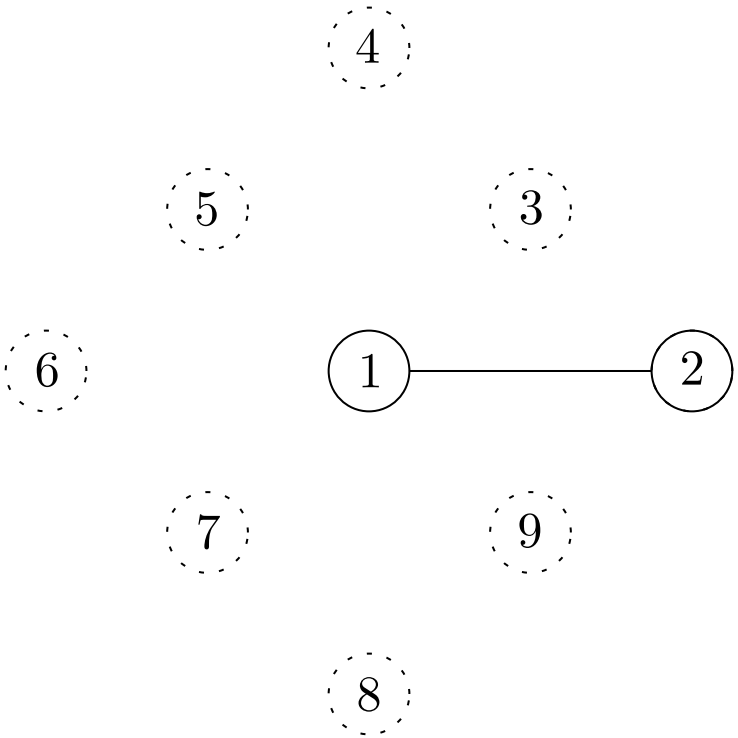
$e)1 \to 9$

$f)1 \to 3$

$g)1 \to 5$

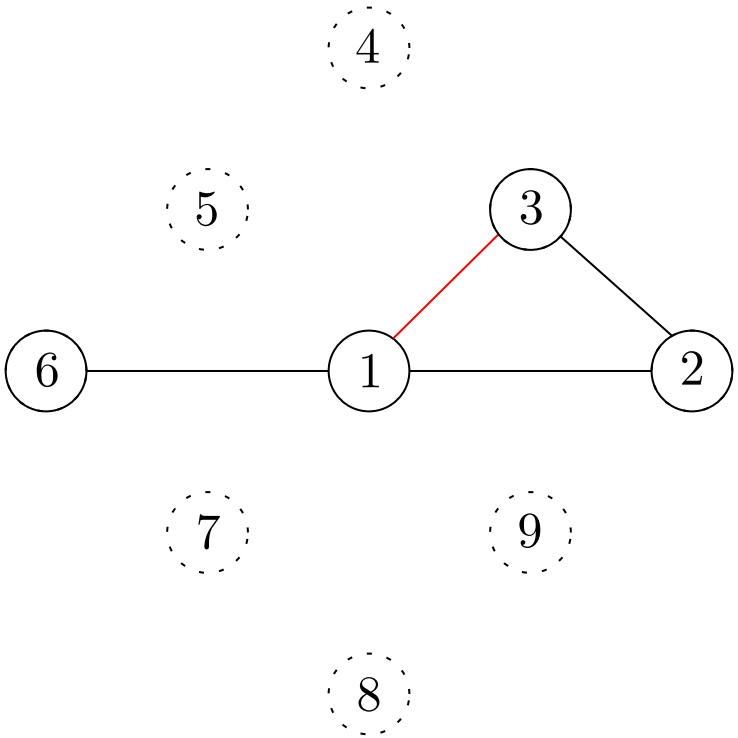
$h)1 \to 7$

An example



$a) 1 \to 2$

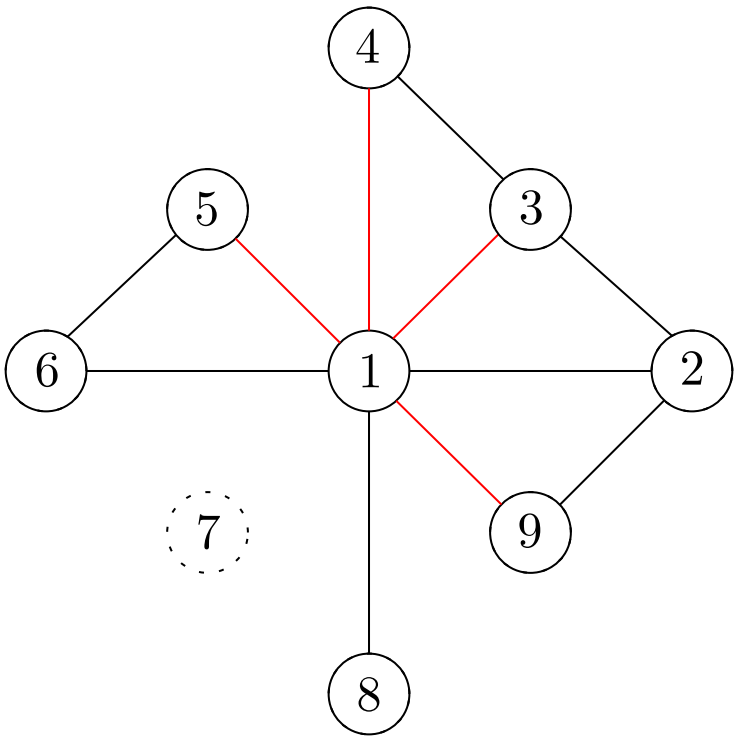
An example



$a) 1 \to 2$

$b) 1 \to 6, 2 \to 3, (1, 3)$

An example

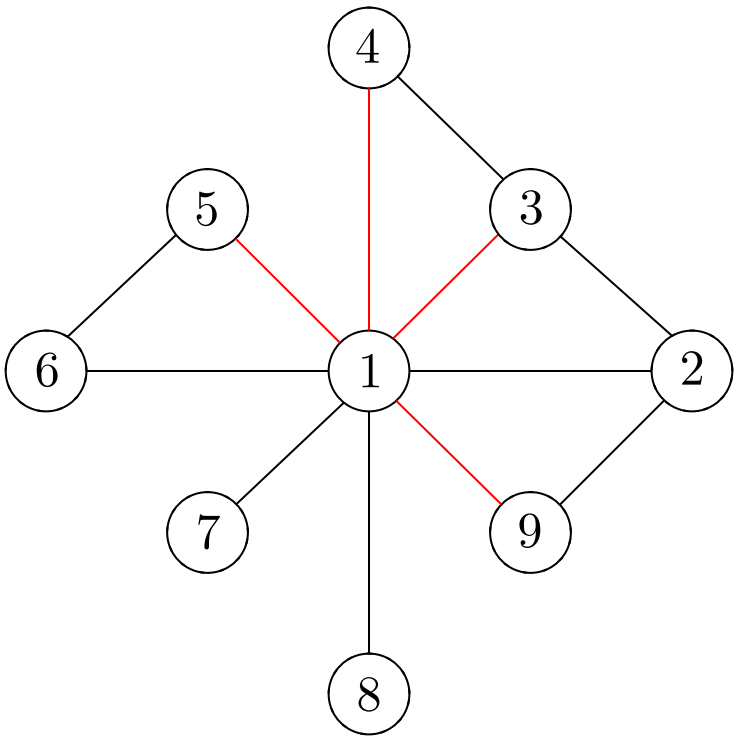


$a) 1 \to 2$

$b) 1 \to 6, 2 \to 3, (1, 3)$

$c) 1 \to 8, 2 \to 9, (1, 9), 3 \to 4, (1, 4), 6 \to 5, (1, 5)$

An example



$a) 1 \to 2$

$b) 1 \to 6, 2 \to 3, (1, 3)$

$c) 1 \to 8, 2 \to 9, (1, 9), 3 \to 4, (1, 4), 6 \to 5, (1, 5)$

$d) 1 \to 7, del((3, 4), (2, 3), (5, 6), (2, 9))$

The Problem

- Based on the previous example, we can see that there is a trade off between time slots and excess edges
- **Construction Schedule Problem**: Given an input graph G, compute in polynomial time a construction schedule of length at most k slots and with at most l excess edges, if it exists.
- **Zero Waste Construction Schedule Problem**: Given a input graph G, compute in polynomial time a construction schedule of length at most k slots and $l = 0$ excess edges, if it exists.

Edge Activation Distance

- Theorem: For $d = 1$, we provide an algorithm computes in polynomial time an optimal construction schedule with k slots for any tree graph G.
- Lemma: For $d \geq 4$, a graph $G = (V, E)$ with n nodes can be generated with a construction schedule with $\log n$ slots and with $O(n)$ excess edges.
- We will focus on $d = 2$ since its more natural and interesting

Properties:

- The nodes generated in a time slot form an independent set in the final graph.
- The progenies of two non adjacent nodes are independent from each other in the final graph.

Lower Bounds:

- Chromatic Number $\chi(G)$
- Size of largest clique c

Efficient Schedule for Trees

- The Efficient Schedule for Trees algorithm computes, in polynomial time, a construction schedule for any given tree graph G with $O(\log^2 n)$ slots and with $O(n)$ excess edges.
- Decomposition strategy where nodes are removed in phases until a single node is present
- The phases can be reversed using $O(\log^2 n)$ slots and $O(n)$ excess edges

Planar Graphs

- The Planar Graph algorithm computes, in polynomial time, a construction schedule for any given planar graph G with $O(\log n)$ slots and with $O(n \log n)$ excess edges.
- Compute a 5-coloring of the input planar graph
- Construct the nodes of each color class one by one using a line construction schedule for each color.
- $O(\log n)$ slots for each color class and $O(\log n)$ excess edges for each node.

Cop-win graphs: Undirected graph on which the pursuer (cop) can always win a pursuit-evasion game against a robber.

- A graph can be constructed with $l = 0$ iff it is a cop-win graph.
- The Recognition of Cop-win Graphs algorithm can decide in polynomial time, whether a given graph G is cop-win, and if so, it also produces a construction schedule with $n - 1$ slots and $l = 0$ excess edges.
- The Fast Cop-win algorithm computes in polynomial time a construction schedule σ for any graph $G = (V, E)$, where $|V| = 2^{\delta}$, with $\log n$ slots and $l = 0$ excess edges, if and only if such a σ exists for G.

Zero Waste Construction Schedule Problem

Theorem: The decision version of the zero-waste construction schedule problem is NP-complete.

- The reduction is from the coloring problem.
- Construct graph G' by adding a clique of size n whose nodes connect with every node in graph G.
- Show that the construction schedule for G' has $\kappa(G')$ slots such that $\kappa(G') = \chi(G) + n$.

Zero Waste Construction Schedule Problem

Theorem: Let $\epsilon > 0$. If there exists a polynomial-time algorithm, which, for every graph G, computes a $n^{1-\epsilon}$-approximate construction schedule, then P=NP.

- The reduction is from the coloring problem.
- Construct graph G'. Show that computing an approximate construction schedule for G' also computes an approximate solution for the chromatic number of G.

Open Problems

- Relationship to overlays (communication vs edges)
- Randomized algorithms, anonymous entities, less local edge activations, dynamic nodes
- Improve upper and/or lower bounds for both models
- Going back from networks to geometry (e.g., translating edge complexity to local moves)
- Distributed model for growing graphs

Open Problems

- Relationship to overlays (communication vs edges)
- Randomized algorithms, anonymous entities, less local edge activations, dynamic nodes
- Improve upper and/or lower bounds for both models
- Going back from networks to geometry (e.g., translating edge complexity to local moves)
- Distributed model for growing graphs

Open Problems

- Relationship to overlays (communication vs edges)
- Randomized algorithms, anonymous entities, less local edge activations, dynamic nodes
- Improve upper and/or lower bounds for both models
- Going back from networks to geometry (e.g., translating edge complexity to local moves)
- Distributed model for growing graphs

Open Problems

- Relationship to overlays (communication vs edges)
- Randomized algorithms, anonymous entities, less local edge activations, dynamic nodes
- Improve upper and/or lower bounds for both models
- Going back from networks to geometry (e.g., translating edge complexity to local moves)
- Distributed model for growing graphs

Open Problems

- Relationship to overlays (communication vs edges)
- Randomized algorithms, anonymous entities, less local edge activations, dynamic nodes
- Improve upper and/or lower bounds for both models
- Going back from networks to geometry (e.g., translating edge complexity to local moves)
- Distributed model for growing graphs

Thank you!!!