Distributed Algorithms for Actively Dynamic Networks

Othon Michail George Skretas Paul G. Spirakis

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

Algorithmic Aspects of Temporal Graphs III
Satellite workshop of ICALP 2020
7th July 2020

Dynamic Networks

General Description:

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

Dynamics:

- 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:

- 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:

- Active or Passive
- Centralized or Distributed

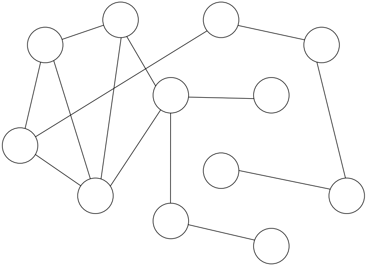
Dynamic Networks

General Description:

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

Dynamics:

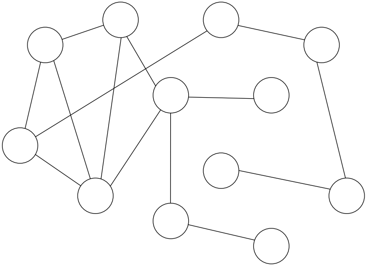
- Active or Passive
- Centralized or Distributed



Dynamic Networks

General Description:

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



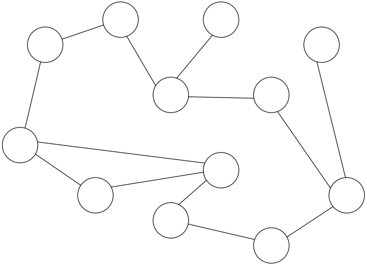
Dynamics:

- 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



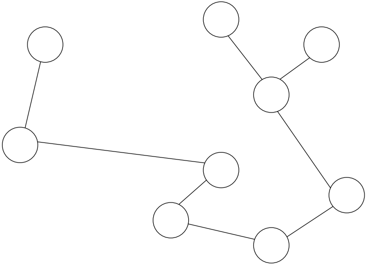
Dynamics:

- 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:

- Active or Passive
- Centralized or Distributed

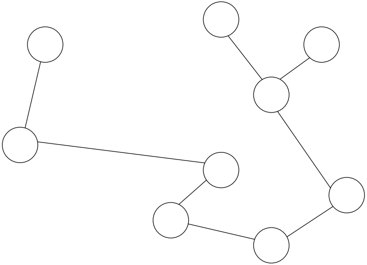
Dynamic Networks

General Description:

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

Dynamics:

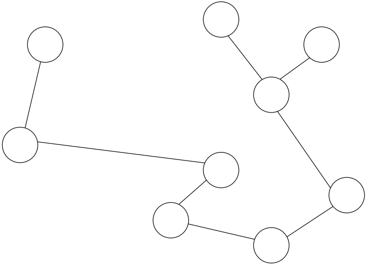
- Active or Passive
- Centralized or Distributed



Dynamic Networks

General Description:

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



Dynamics:

- Active or Passive
- Centralized or Distributed

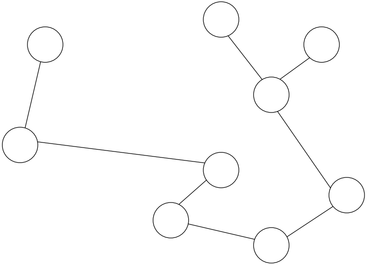
Dynamic Networks

General Description:

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

Dynamics:

- Active or Passive
- Centralized or Distributed



Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

- 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

Active Dynamics with Distributed Control

- Activate connections to new neighbors or deactivate connections
- No central coordinator

General Goal:

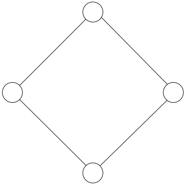
- Transform the initial network G_s into a target network G_f from a family of target networks
- Exploit some good properties of G_f in order to more efficiently solve a distributed task

Related Systems

- Temporal Graphs
  - Initiated by Berman [Be09] and Kempe et al. [KKK00]
- Distributed Computation in Passively Dynamic Networks
  - Seminal paper by [KLO10]
  - Population Protocols [AADFP06]
- Construction of Overlay Networks
  - Most similar to our model
  - Introduced by Stoica et al. [SMK01] and [AS07]
- Programmable Matter
  - Geometry plays a big role

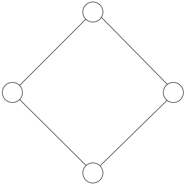
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



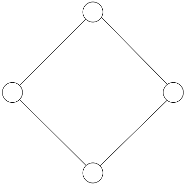
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



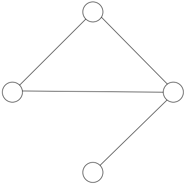
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



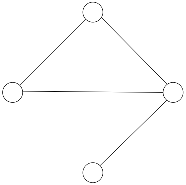
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



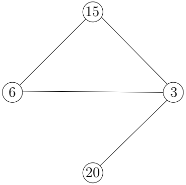
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



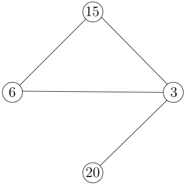
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



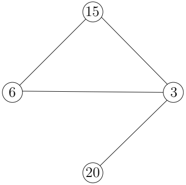
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



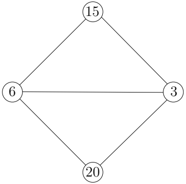
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



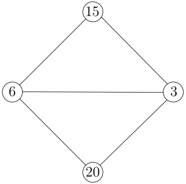
Our Model

- 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 $uv \notin E(i)$, uw and wv are active in round i
- One edge between two nodes



Our 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 $uv \notin E(i)$, uw and wv are active in round i
- 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

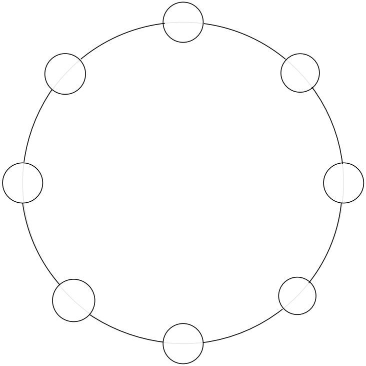
Very simple strategy:

- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges

The Clique Formation Strategy

Very simple strategy:

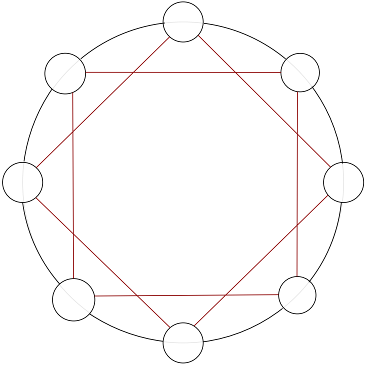
- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

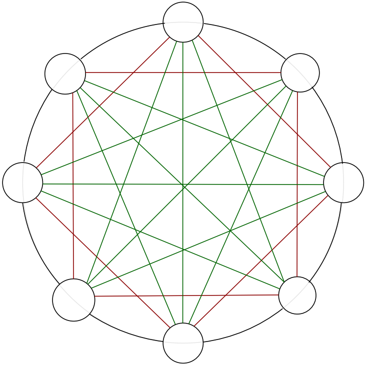
- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

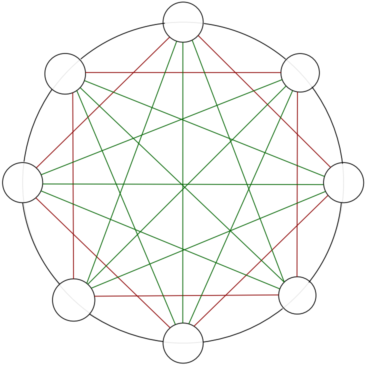
- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

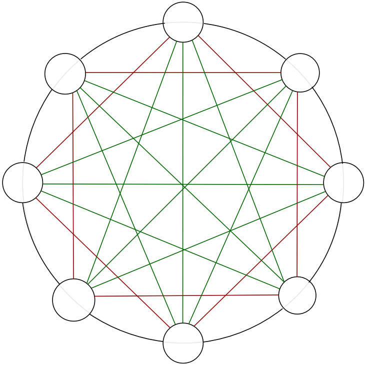
- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

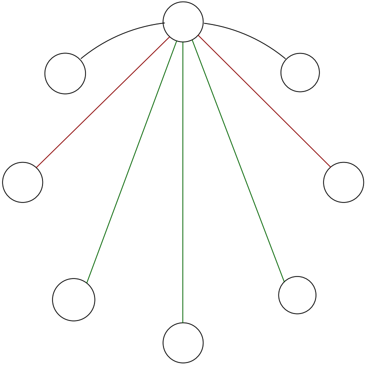
- Activate an edge with every distance 2 neighbor
- Spanning clique, $\log n$ rounds
- Eliminate extra edges



The Clique Formation Strategy

Very simple strategy:

- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges

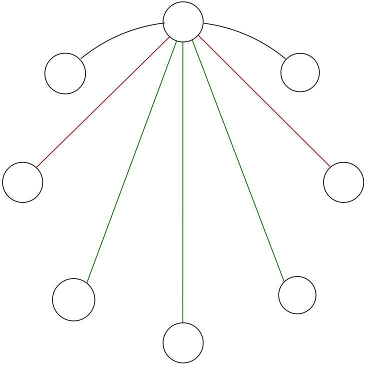


The Clique Formation Strategy

Very simple strategy:

- Activate an edge with every distance 2 neighbor
- Spanning clique, log n rounds
- Eliminate extra edges

BUT



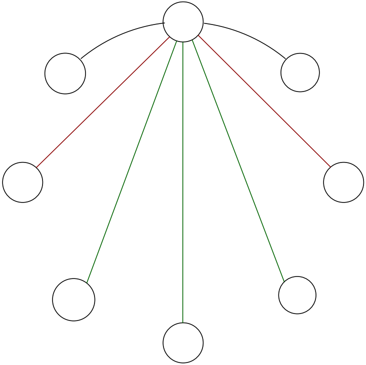
The Clique Formation Strategy

Very simple strategy:

- 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

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 Leader Election

Introduction on the General Strategy

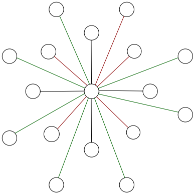
- Nodes are partitioned into committees
- Committees organised into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Logarithmic running time
- Time: phases*gadgetdiameter

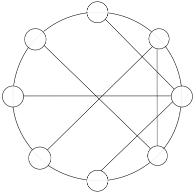
Introduction on the General Strategy

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

Introduction on the General Strategy

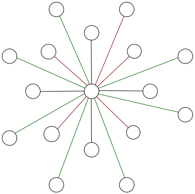
- Nodes are partitioned into committees
- Committees organised into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Logarithmic running time
- Time: phases*gadgetdiameter

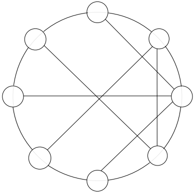




Introduction on the General Strategy

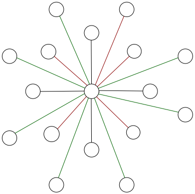
- Nodes are partitioned into committees
- Committees organised into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Logarithmic running time
- Time: phases*gadgetdiameter

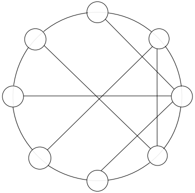




Introduction on the General Strategy

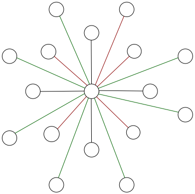
- Nodes are partitioned into committees
- Committees organised into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Logarithmic running time
- Time: phases*gadgetdiameter

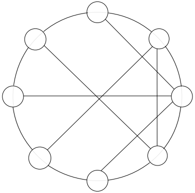




Introduction on the General Strategy

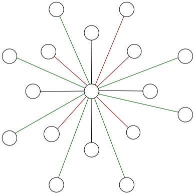
- Nodes are partitioned into committees
- Committees organised into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Logarithmic running time
- Time: phases*gadgetdiameter

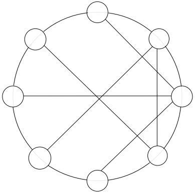




Introduction on the General Strategy

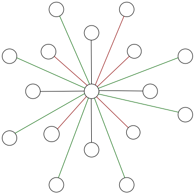
- Nodes are partitioned into committees
- Committees organised into gadget networks
- Each node forms its own committee
- Committees compete and merge
- Final network has a single committee
- Logarithmic running time
- Time: phases*gadgetdiameter

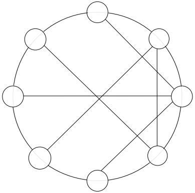




Introduction on the General Strategy

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





GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

TreeToStar

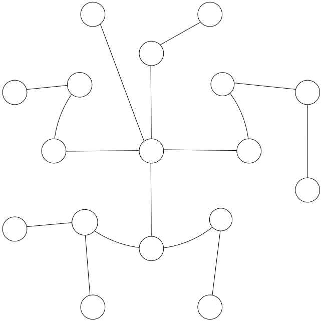
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent

GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

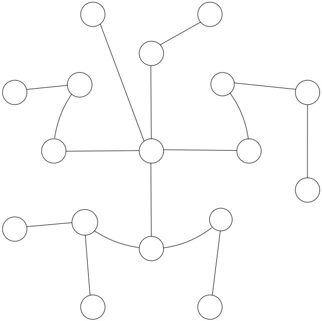


GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

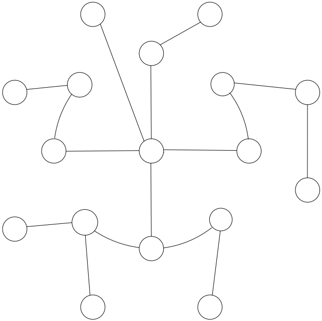


GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

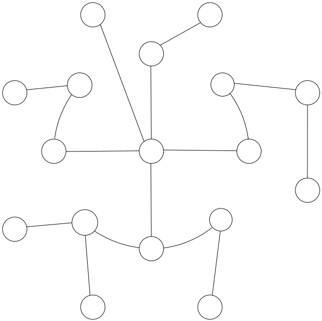


GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

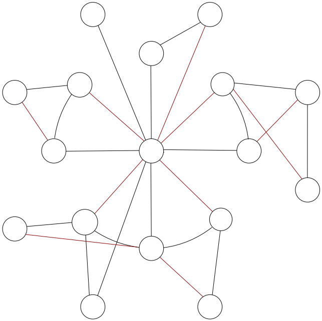


GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

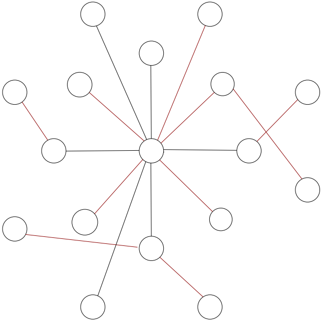


GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

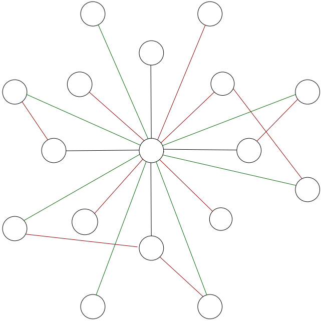


GraphToStar Algorithm

- Gadget network: Star
- TreeToStar subroutine

TreeToStar

- 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

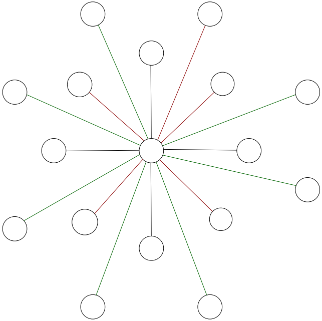


GraphToStar Algorithm

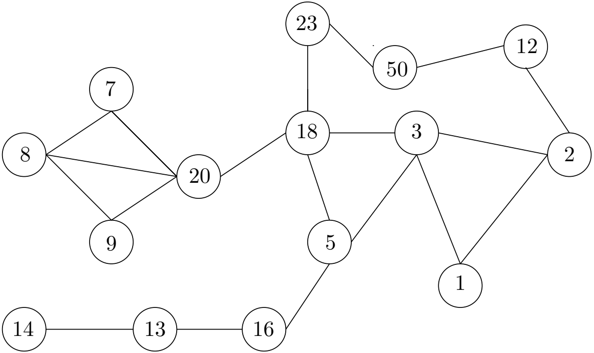
- Gadget network: Star
- TreeToStar subroutine

TreeToStar

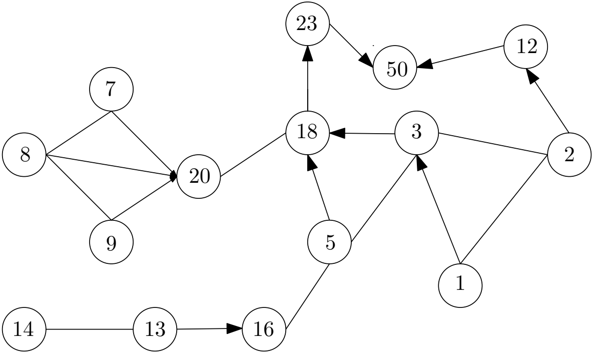
- Transforms any initial oriented Tree graph into a spanning Star graph in $\log n$ time
- Activates an edge with grandparent
- Deactivate an edge with parent



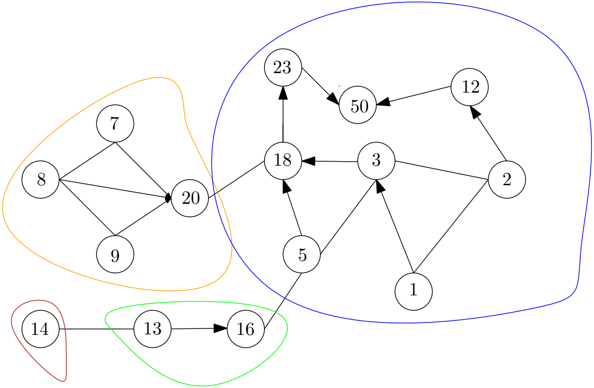
High Level



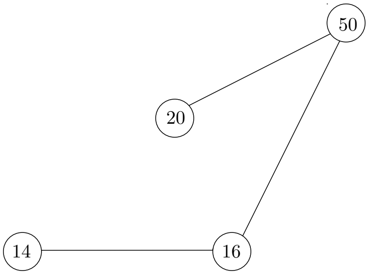
High Level



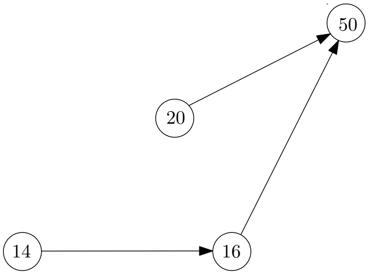
High Level



High Level

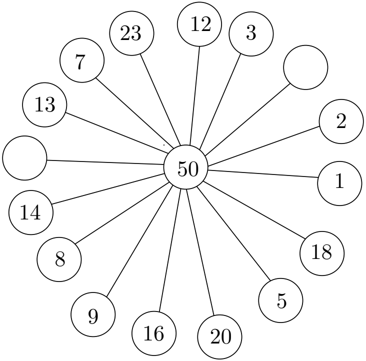


High Level



High Level

Low Level



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

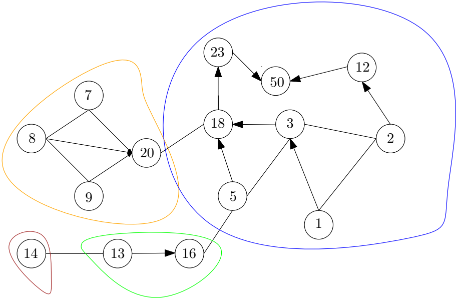
- Committees keep merging
- Can't get isolated indefinitely

Time Complexity

- Committees "double" in size
- Isolated

Edge Complexity

- Omitted



GraphToWreath Algorithm

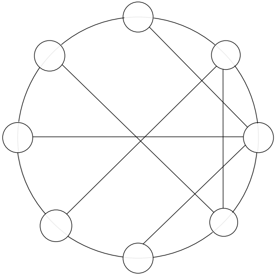
- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine

LineToCompleteBinaryTree

- 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

LineToCompleteBinaryTree

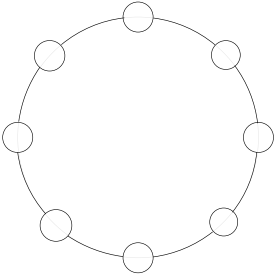
- Transforms any initial oriented line into a spanning Complete Binary Tree in log n time
- Activates an edge with grandparent
- Deactivate an edge with parent
- Stop when grandparent has two children

GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine

LineToCompleteBinaryTree

- 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

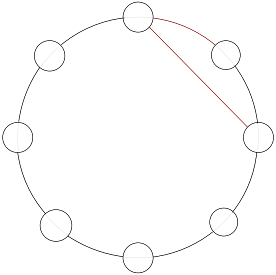
LineToCompleteBinaryTree

- 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

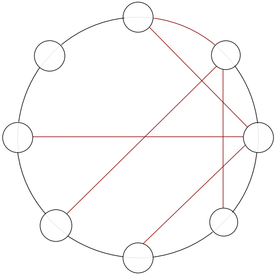


GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine

LineToCompleteBinaryTree

- 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

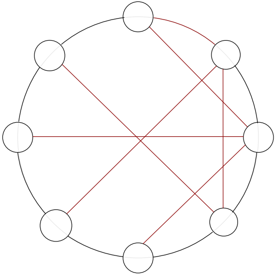


GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine

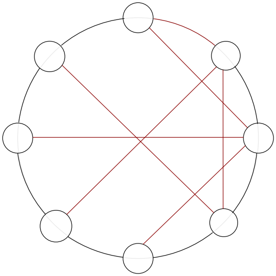
LineToCompleteBinaryTree

- 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

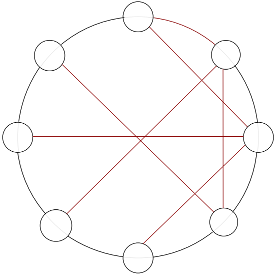


LineToCompleteBinaryTree

- 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

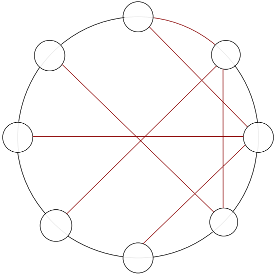


LineToCompleteBinaryTree

- 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

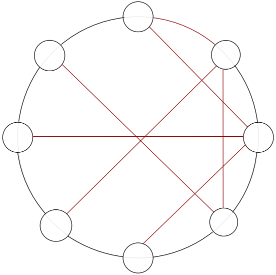


LineToCompleteBinaryTree

- 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

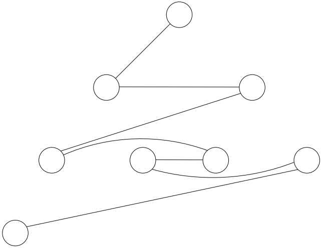
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine



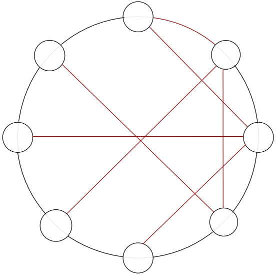
LineToCompleteBinaryTree

- 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



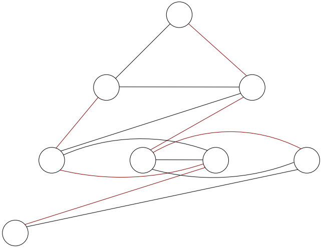
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine



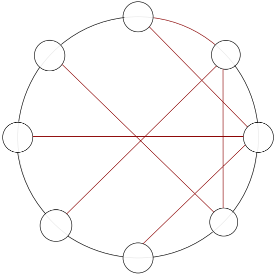
LineToCompleteBinaryTree

- 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



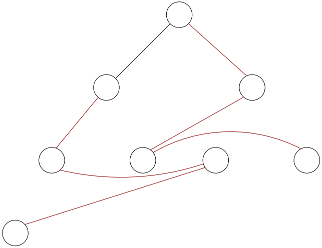
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine



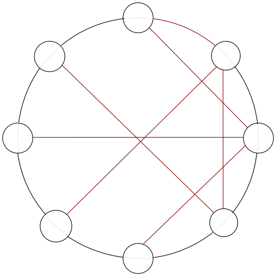
LineToCompleteBinaryTree

- 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



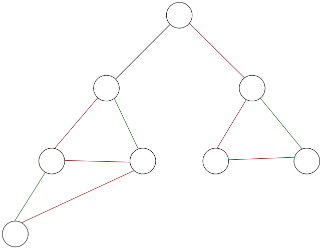
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine



LineToCompleteBinaryTree

- 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

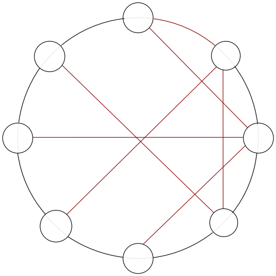


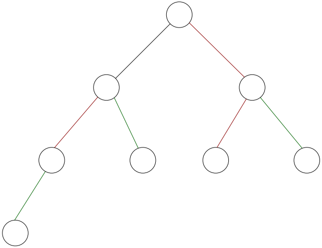
GraphToWreath Algorithm

- Gadget network: Wreath
- LineToCompleteBinaryTree subroutine

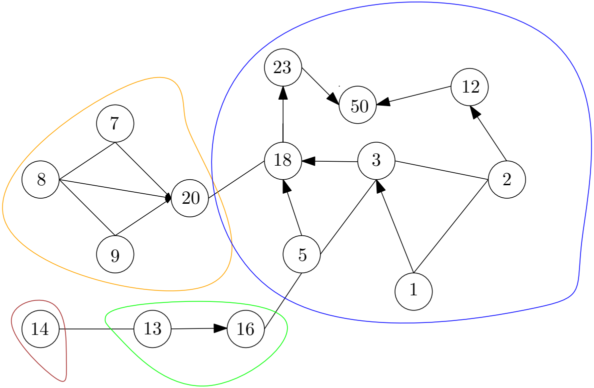
LineToCompleteBinaryTree

- 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

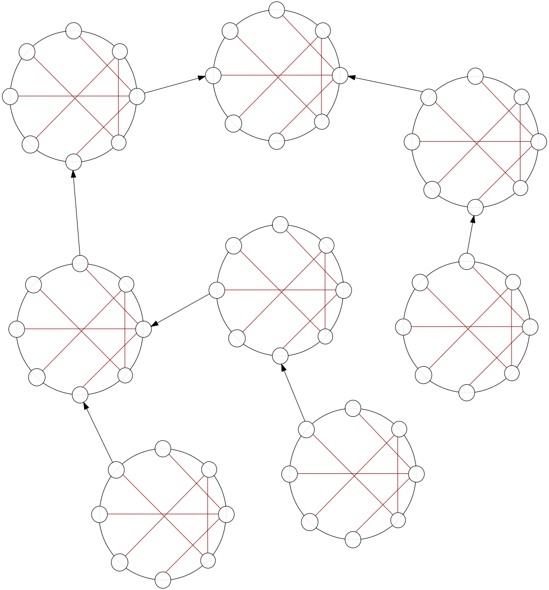




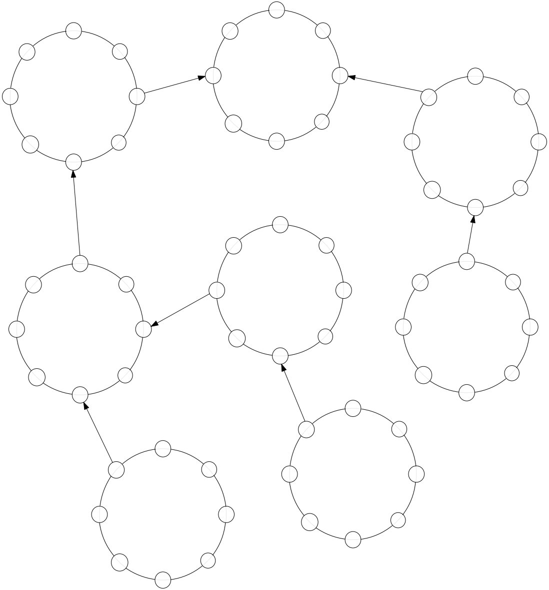
High Level



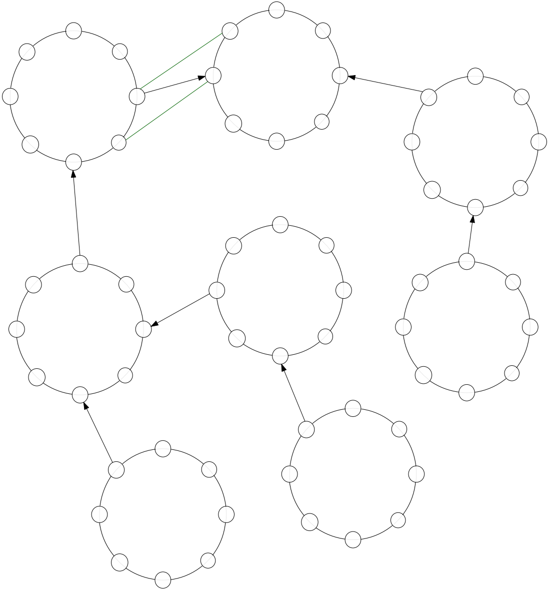
Low Level



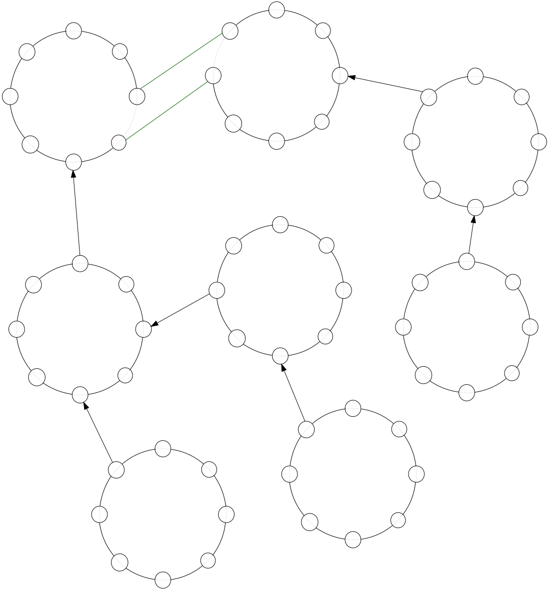
Low Level



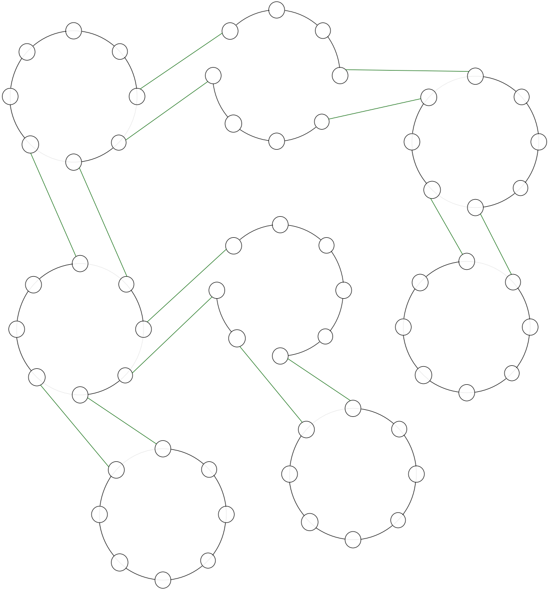
Low Level



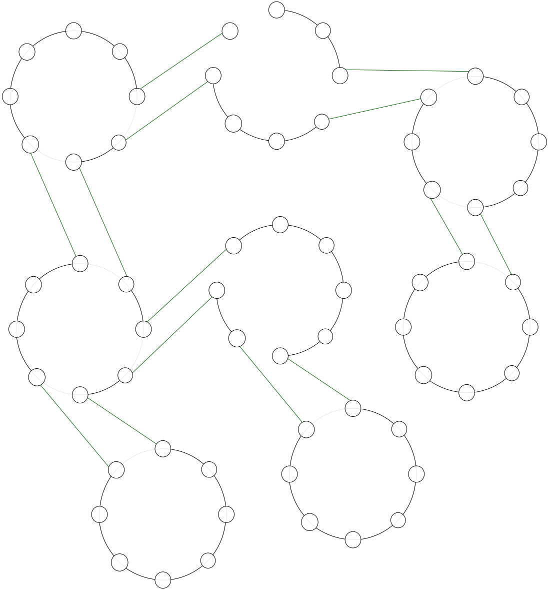
Low Level



Low Level



Low Level



GraphToThinWreath

Theorem: For any initial connected graph with polylogarithmic degree, the GraphToThinWreath algorithm solves Depth-$\log n$ Tree in $O(\frac{\log^2 n}{\log\log n})$ time with $O(n\log^2 n)$ total edge activations, $O(n)$ active edges per round and $O(1)$ maximum activated degree

Few Words:

- Gadget network: ThinWreath
- LineToCompletePolylogarithmicTree subroutine
- Requires knowledge of the size of the network
- Changes in low and high level strategy

GraphToThinWreath

Theorem: For any initial connected graph with polylogarithmic degree, the GraphToThinWreath algorithm solves Depth-$\log n$ Tree in $O(\frac{\log^2 n}{\log \log n})$ time with $O(n \log^2 n)$ total edge activations, $O(n)$ active edges per round and $O(1)$ maximum activated degree

Few Words:

- Gadget network: ThinWreath
- LineToCompletePolylogarithmicTree subroutine
- Requires knowledge of the size of the network
- Changes in low and high level strategy

GraphToThinWreath

Theorem: For any initial connected graph with polylogarithmic degree, the GraphToThinWreath algorithm solves Depth-log n Tree in $O(\frac{\log^2 n}{\log\log n})$ time with $O(n\log^2 n)$ total edge activations, $O(n)$ active edges per round and $O(1)$ maximum activated degree

Few Words:

- Gadget network: ThinWreath
- LineToCompletePolylogarithmicTree subroutine
- Requires knowledge of the size of the network
- Changes in low and high level strategy

GraphToThinWreath

Theorem: For any initial connected graph with polylogarithmic degree, the GraphToThinWreath algorithm solves Depth-$\log n$ Tree in $O(\frac{\log^2 n}{\log \log n})$ time with $O(n \log^2 n)$ total edge activations, $O(n)$ active edges per round and $O(1)$ maximum activated degree

Few Words:

- Gadget network: ThinWreath
- LineToCompletePolylogarithmicTree subroutine
- Requires knowledge of the size of the network
- Changes in low and high level strategy

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

Conclusion

Summary:

- Three Algorithms
- Trade off between time and degree

Open Problems:

- Improve time and degree together
- Change the property of the target network
- Lower Bounds

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 Leader Election