Investigating Methods of Controlling Algebraic Connectivity

Daniel Rodrigues, December 11th

Part of a Three Project Series

CS 229R: THIS PAPER

1. Investigating Methods of Controlling Algebraic Connectivity

CS 286

2. Dictating Algebraic Connectivity as a Topology in Networked Multi-Agent Systems

EXTENSION

3. Using Multi-Path Routing to Identify Malicious Agents in Consensus

Introduction

Project Motivation & Problem Statement

Can we enable a system engineer or real-time supervisor to constrain the algebraic connectivity (i.e. Fiedler value) of a mobile multi-agent system by augmenting a flock's network topology?

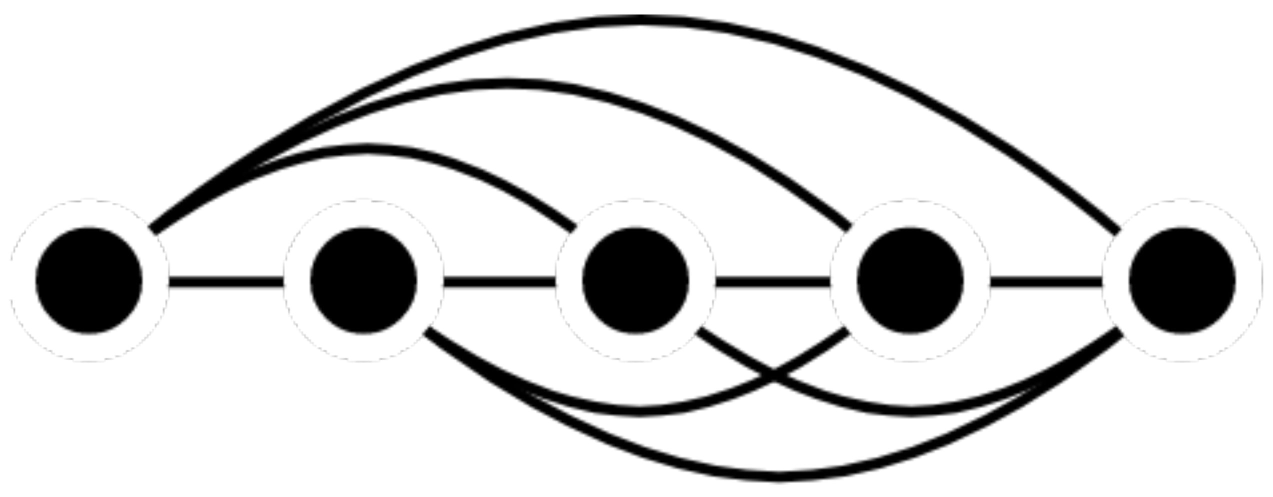
λ_2 = 5.00

λ_2 = 0.38

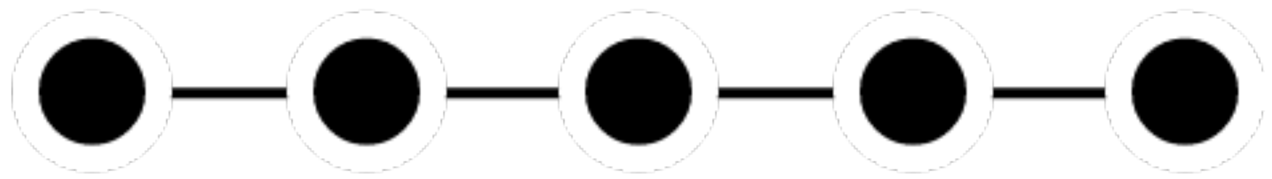
Project Motivation & Problem Statement

Instance: Given an undirected graph $G = (V, E)$ and a non-negative threshold t_1 and t_2.

Question: Is there a subset $B \subseteq E$ such that the graph $H = (V, E - B)$ satisfies $t_1 \leq \lambda_2(H) \leq t_2$.



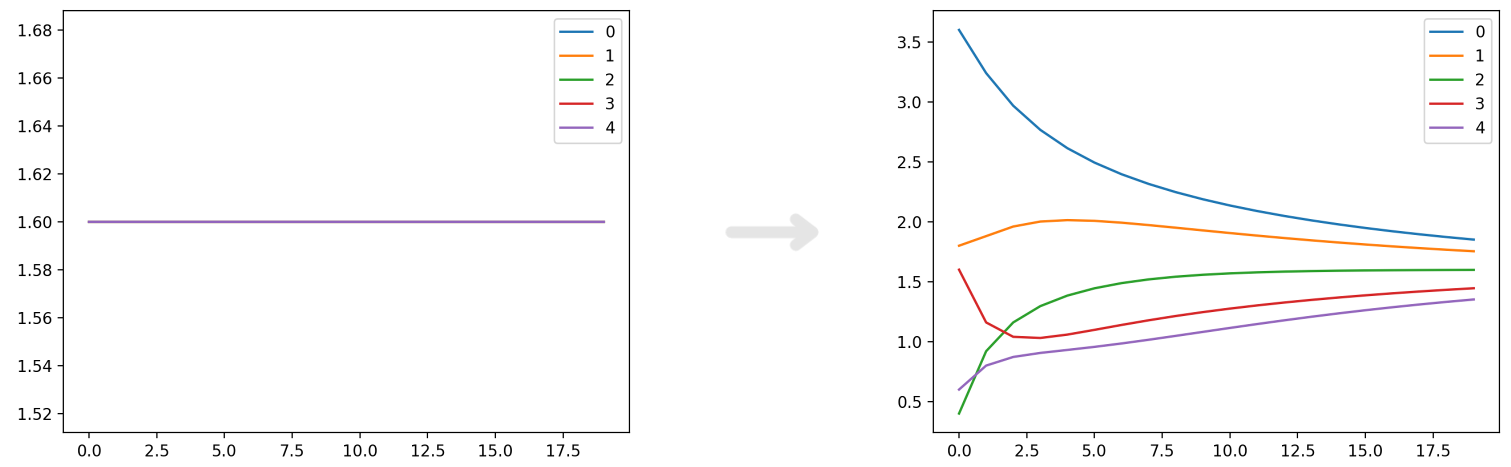
λ_2 = **5.00**



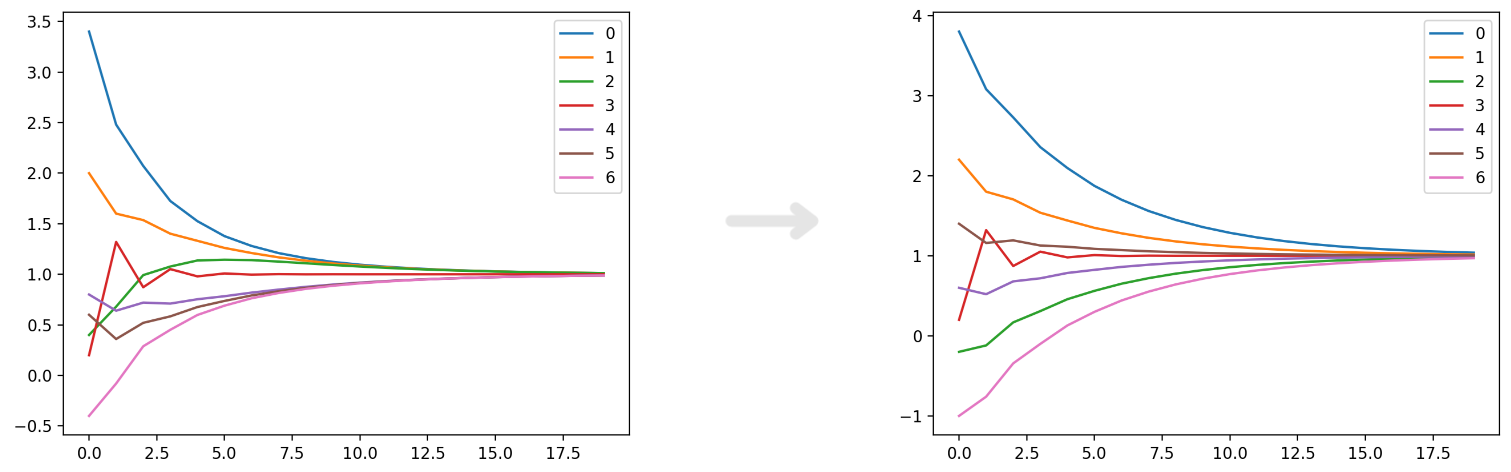
λ_2 = **0.38**

Baseline Results

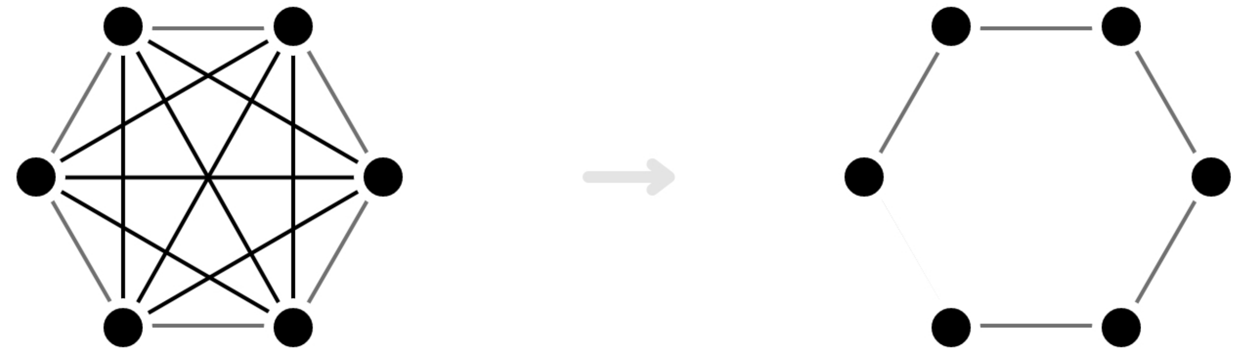
Line Formation

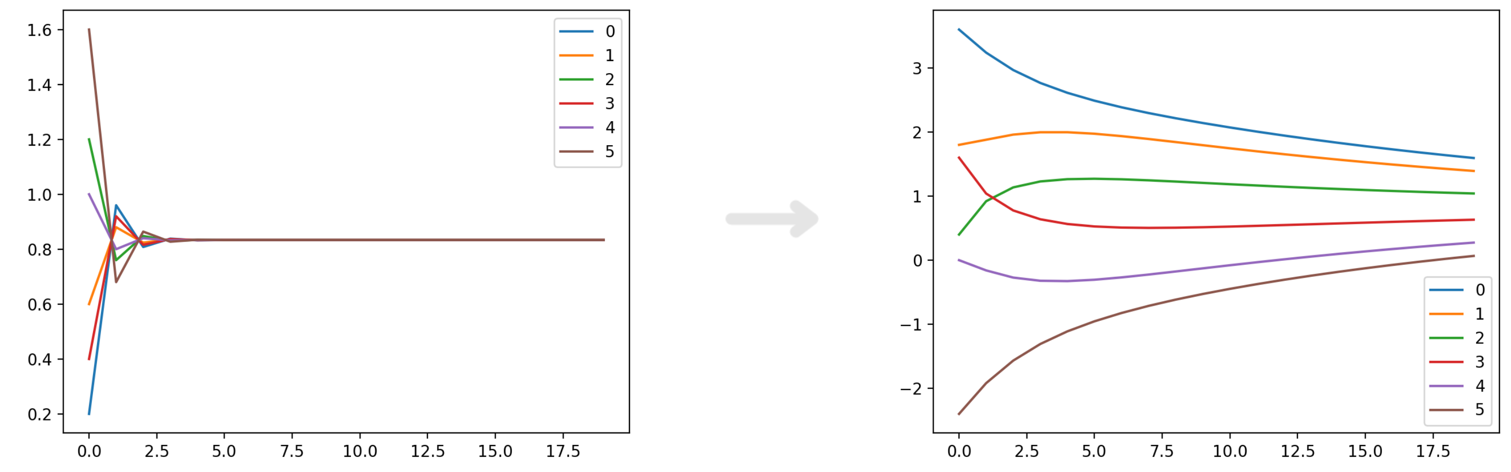


Wedge Formation



Circle Formation





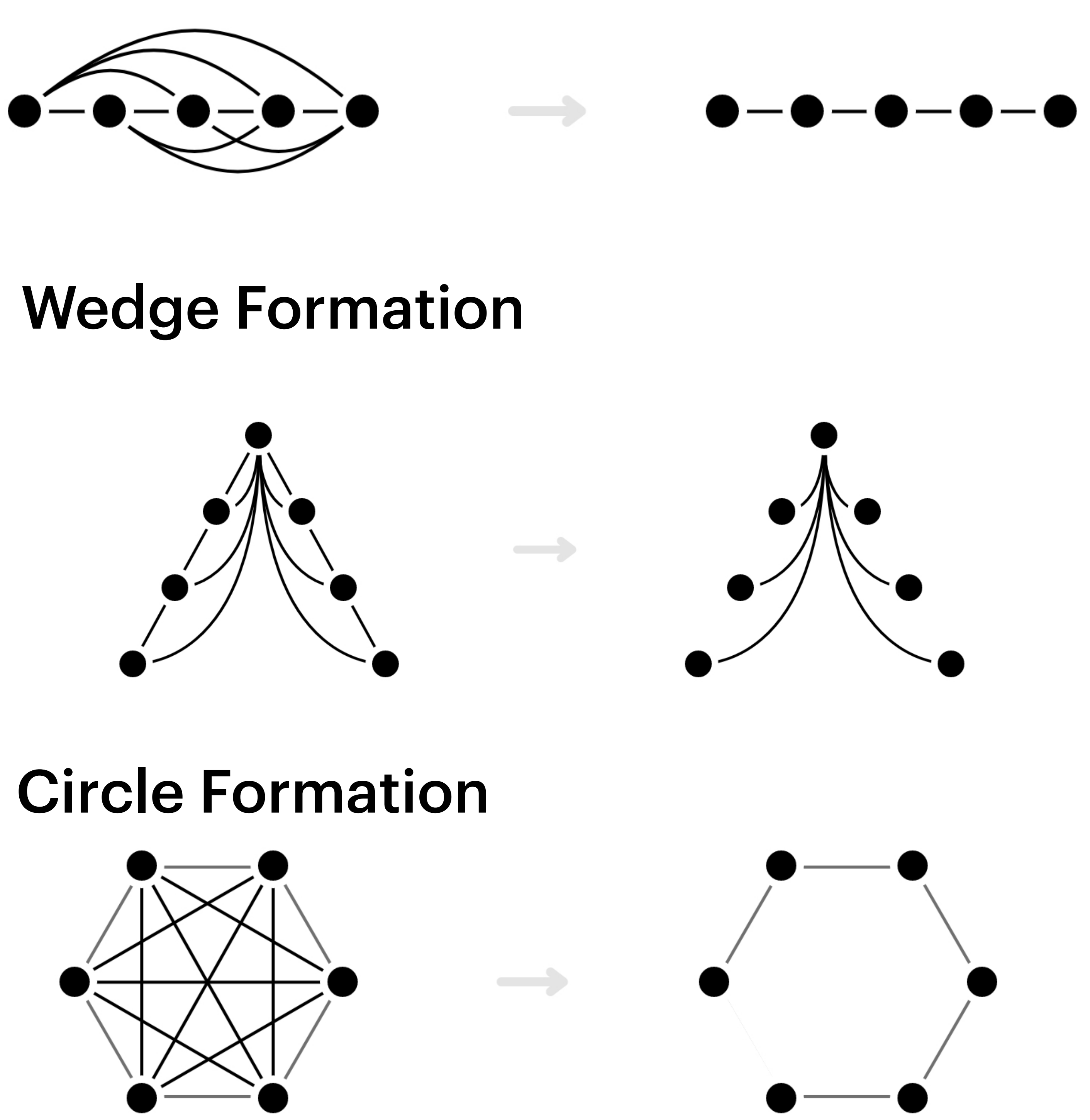
Takeaways: A less connected graph converges slower (obviously).
Which edges are chosen can significantly affect convergence time.

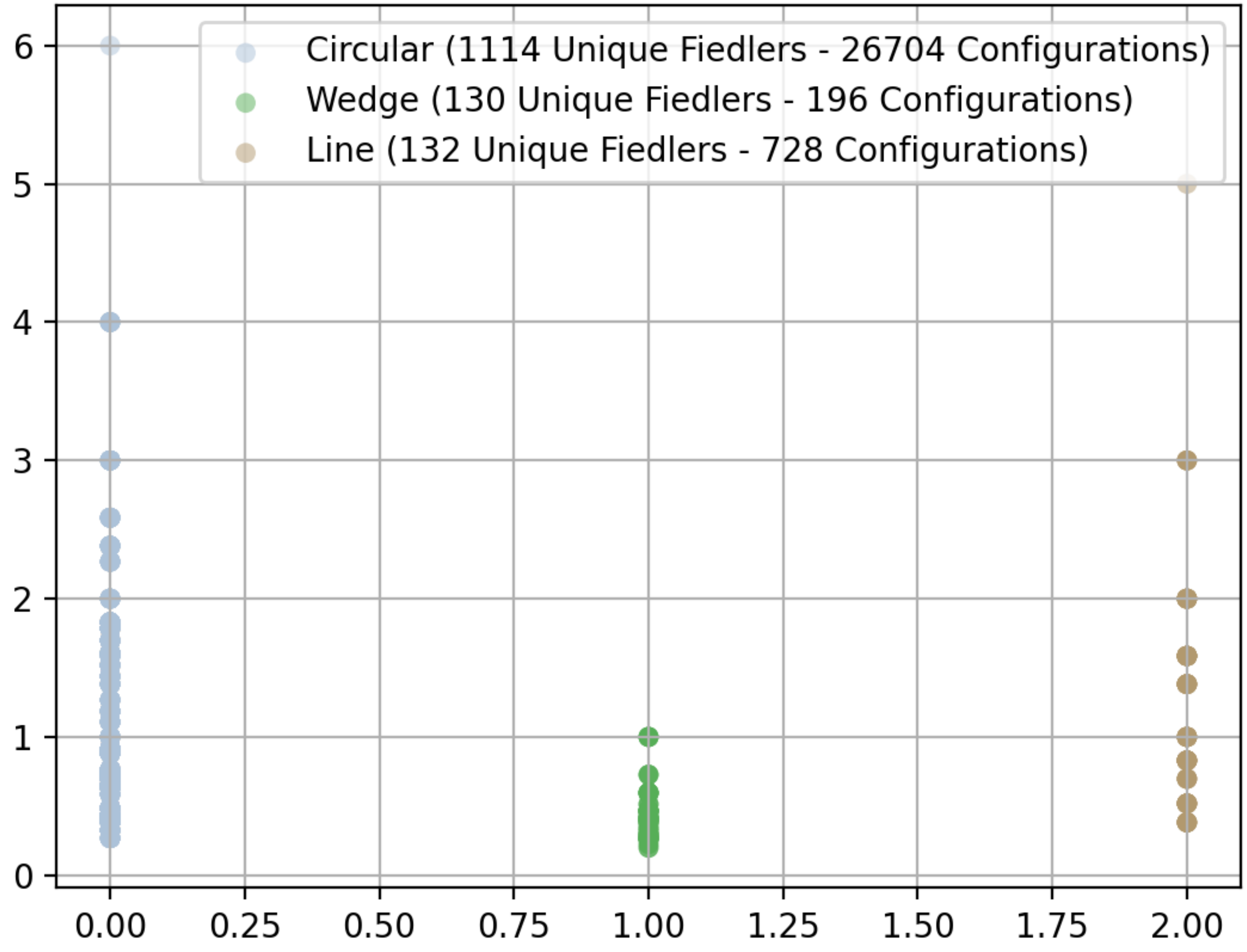
Configuration and Result Space Size

Line Formation

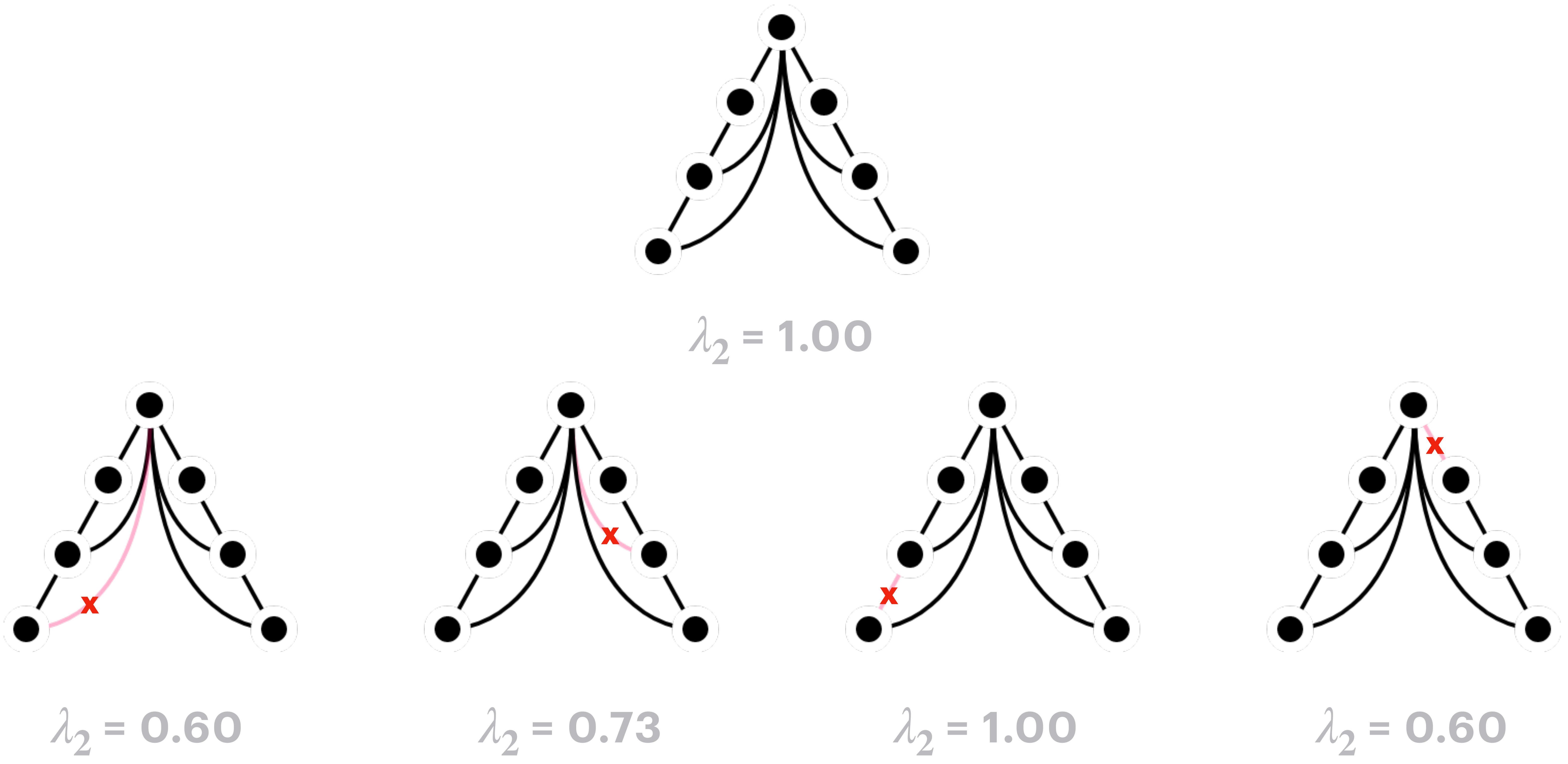
Wedge Formation

Circle Formation





Control Over Connectivity Reduction



DETERMINISTIC PROPOSED ALGORITHMS RANDOMIZED

Experimental Studies

Deterministic Algorithm

Removes an Edge with Min/Max Fiedler Impact

Input:

G **Original Graph**

λ_2' **Target Fiedler**

Output:

$\lambda_2(G')$ **Reached Fiedler**

G' **Augmented Graph**

Flags:

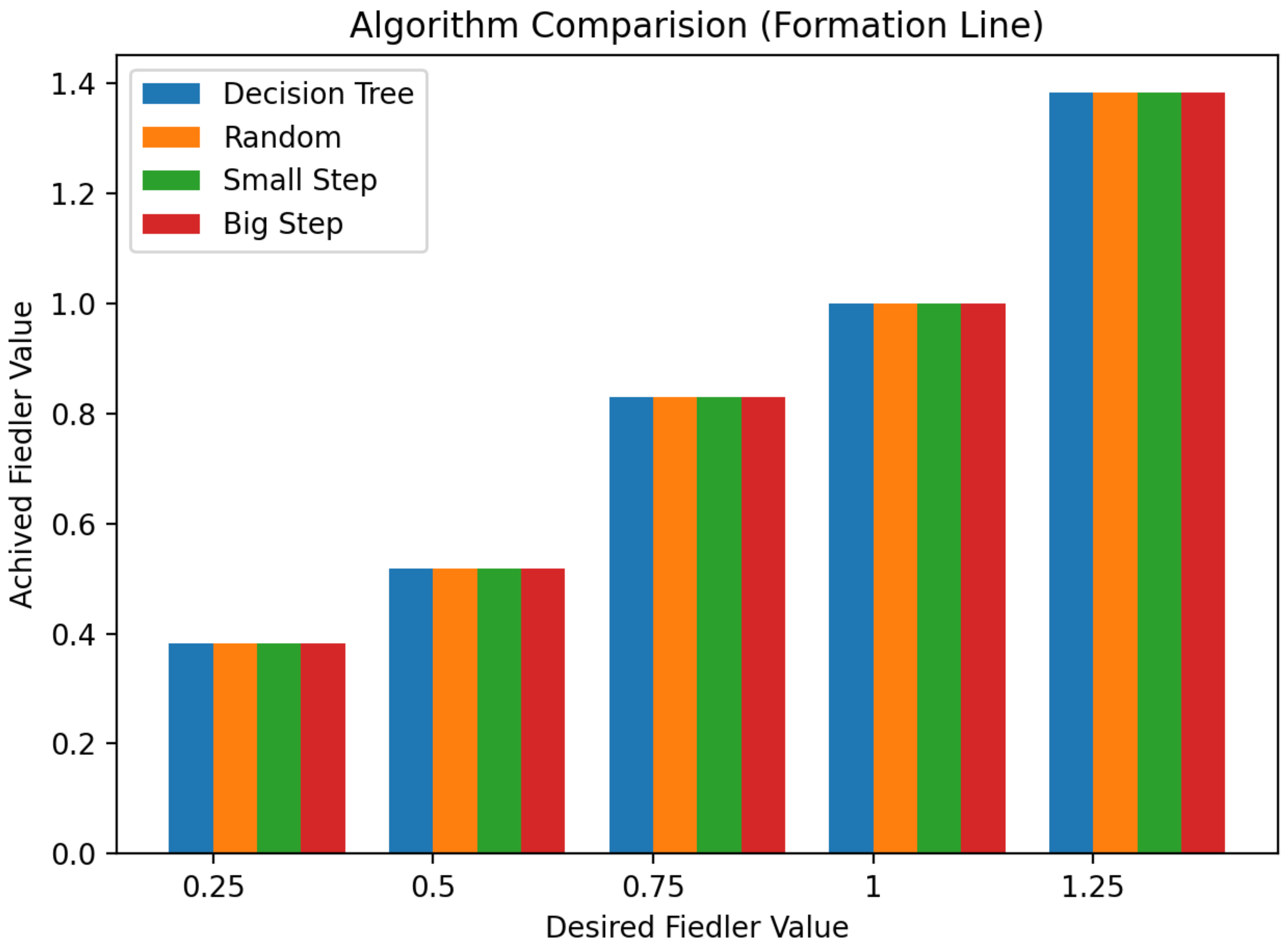
Allow Disconnect

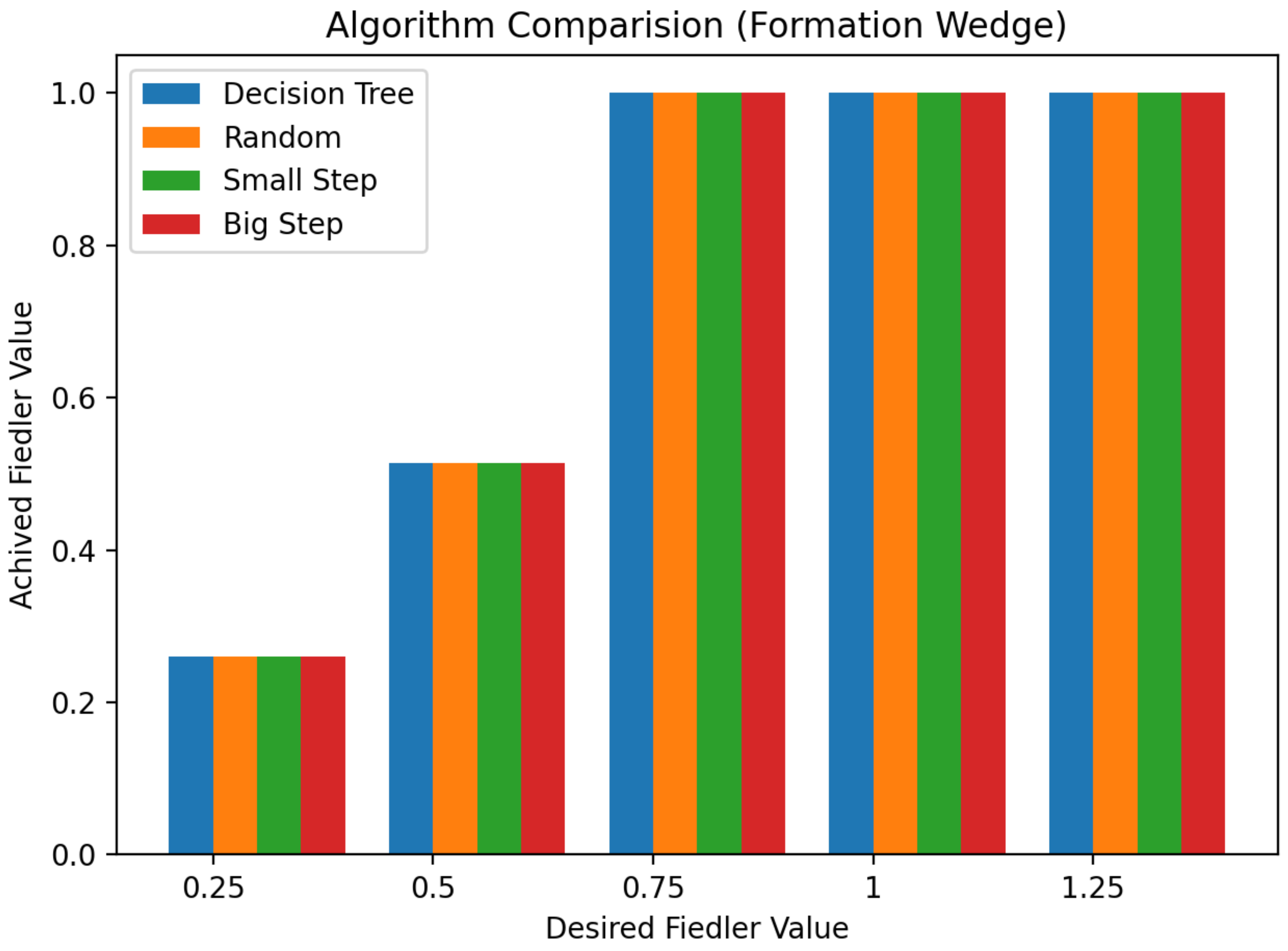
One/Two Sided Bound

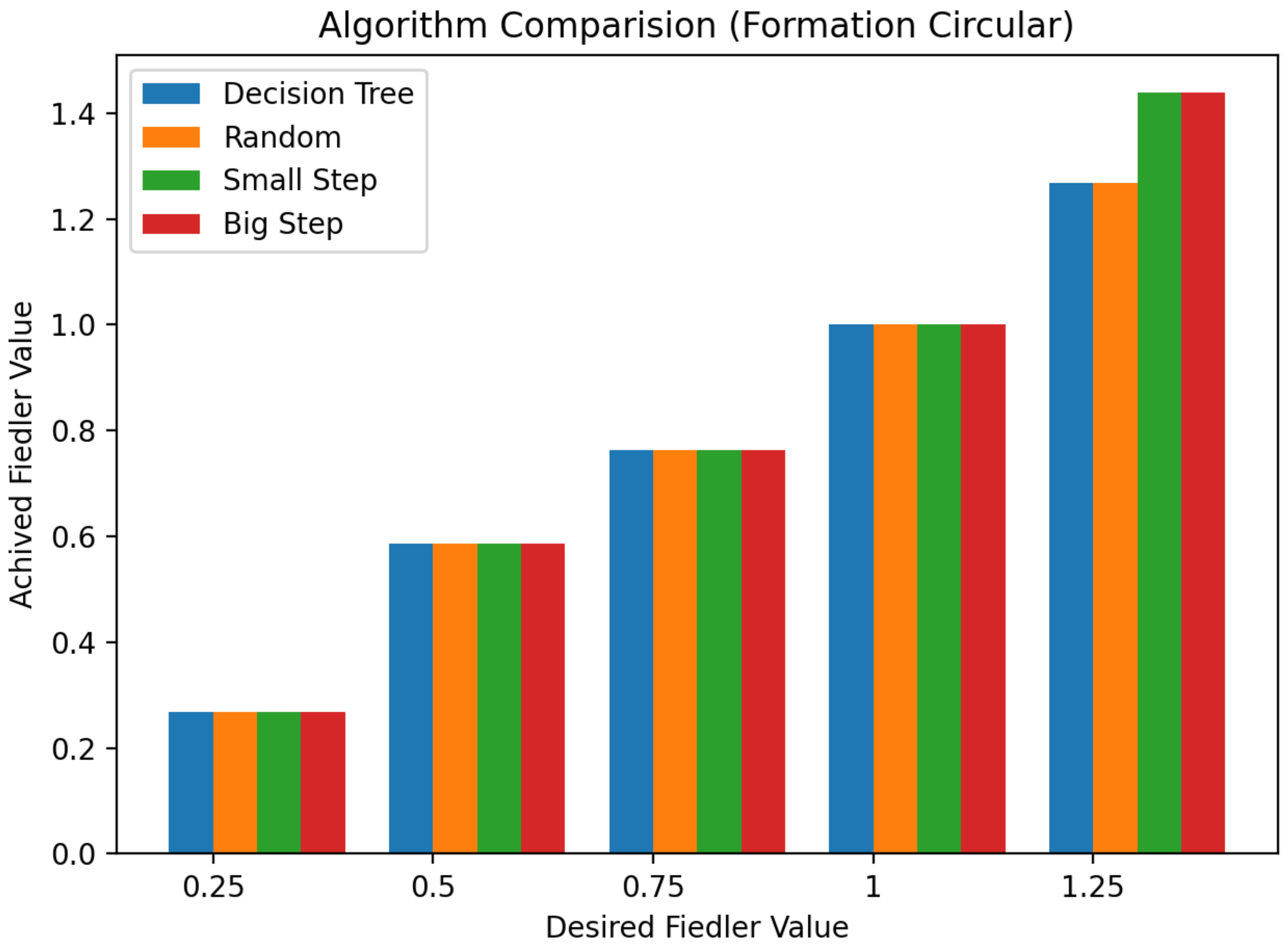
```
Function CreateGraph(g, target, runs) do
    return CutEdges(g, target)
end
```

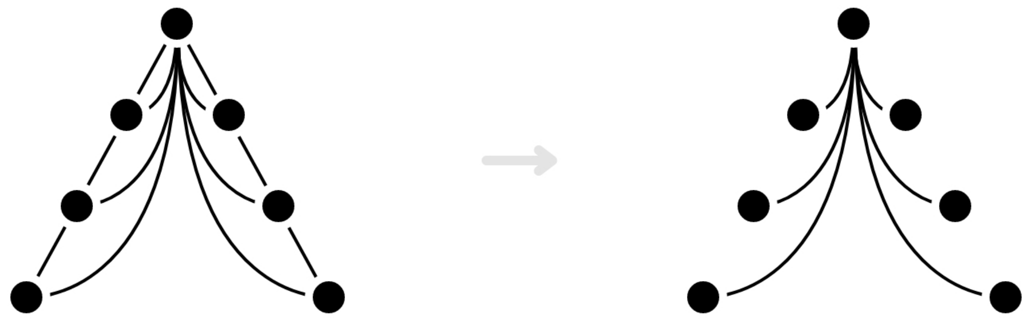
```
Function FindBestEdgeRemoval(graph, edge_set, f_current, target)
do
    options = []
    foreach (u,v) in edge_set do
        f_next, g_next = self.graph_without_edge(graph,u,v)
        // Consider all edges that bring us closer to the target
        if abs(f_next - target) > abs(f_current - target) do
            options.append((f_next, g_next, (u,v)))
        end
    end
    // Can use min for SmallStep or max for BigStep Variation
    dist_to_target = [abs(f - target) for (f,g) in graphs]
    return min(dist_to_target, key=lambda o: o[0], default=null)
end

Function CutEdges(g, target) do
    edge_set = g_edges_as_list(g)
    f_current = CalcFiedler(g)
    while f_current > target do
        res = FindBestEdgeRemoval(g, edge_set, f_current, target)
        if res == null do
            // No valid edges remain to remove
            break
        end
        // Remove edge, update graph, continue
        f_next, g_next, edge = res
        edge_set.remove(edge)
        f_current, g = f_next, g_next
    end
    return f_current, g
end
```

Results


Algorithm Comparision (Formation Line)
Decision Tree
Random
Small Step
Big Step
Achived Fiedler Value
Desired Fiedler Value
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.25
0.5
0.75
1
1.25


Algorithm Comparision (Formation Wedge)
Decision Tree
Random
Small Step
Big Step
Achived Fiedler Value
Desired Fiedler Value
0.0
0.2
0.4
0.6
0.8
1.0
0.25
0.5
0.75
1
1.25


Algorithm Comparision (Formation Circular)
Decision Tree
Random
Small Step
Big Step
Achived Fiedler Value
Desired Fiedler Value
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.25
0.5
0.75
1
1.25



Algorithms

BASELINE

Exhaustive Search

↓

FIRST ROUND

Deterministic-Fiedler

↙ ↘

SECOND ROUND

Deterministic-Leverage

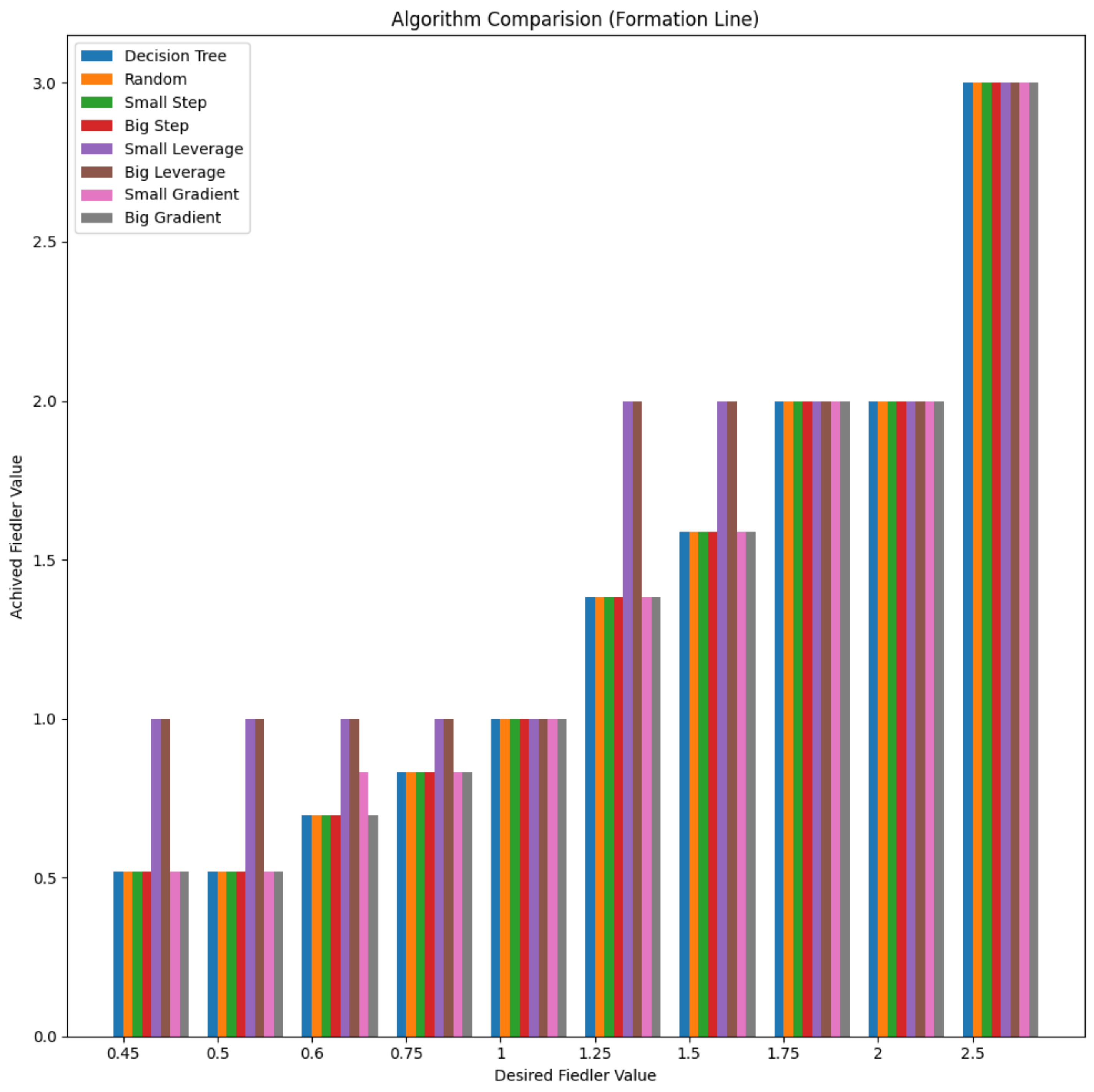
SECOND ROUND

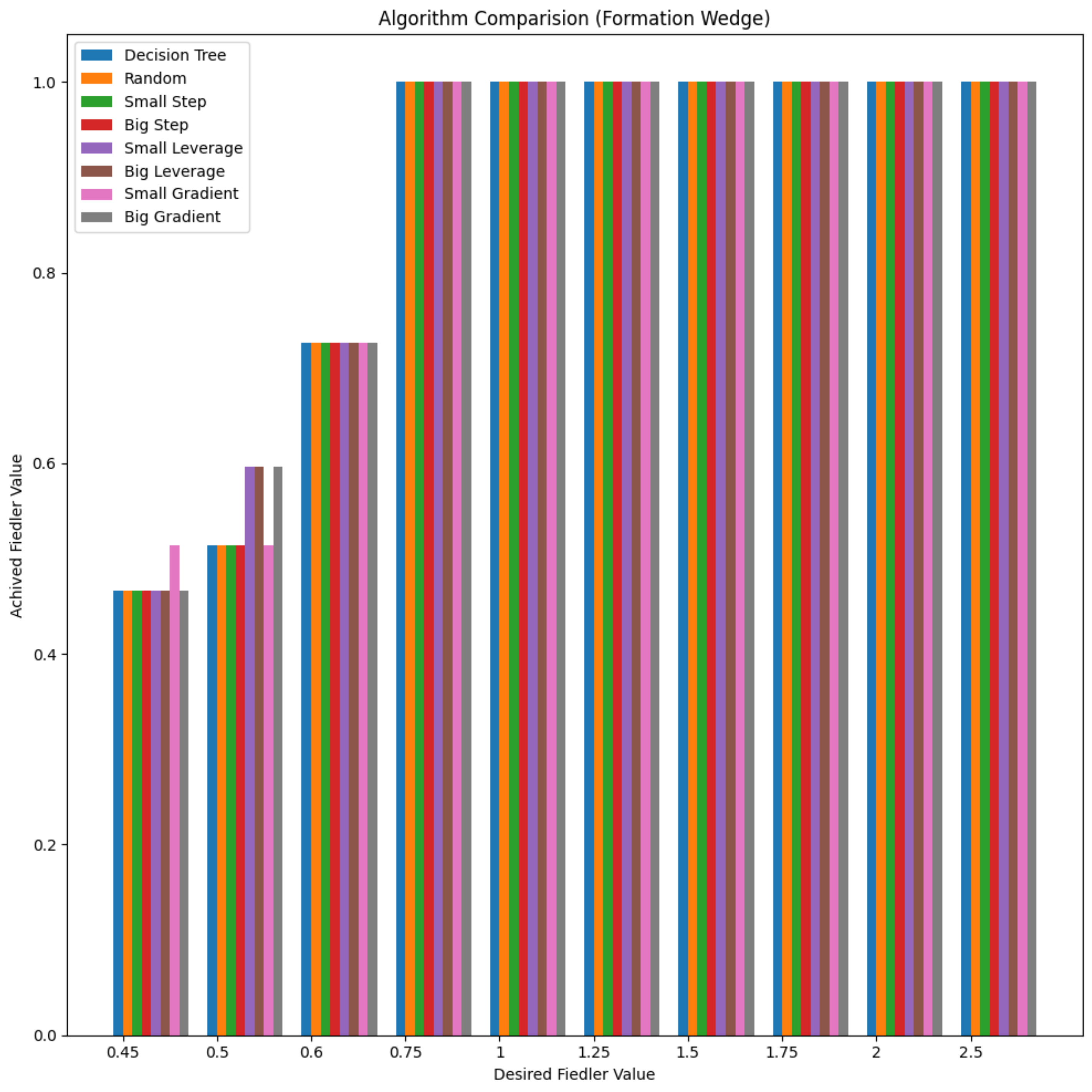
Deterministic-Gradient

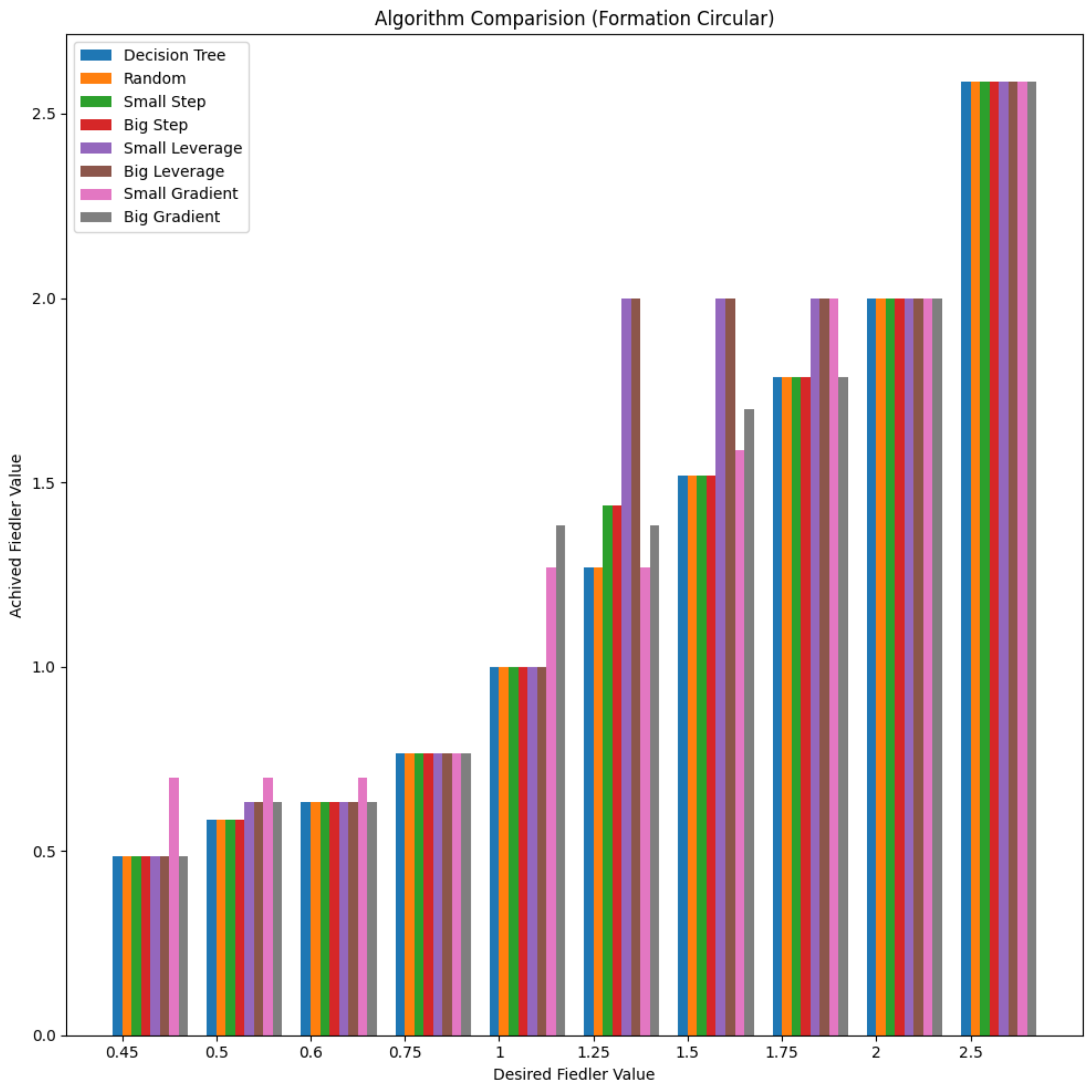
CHOSEN

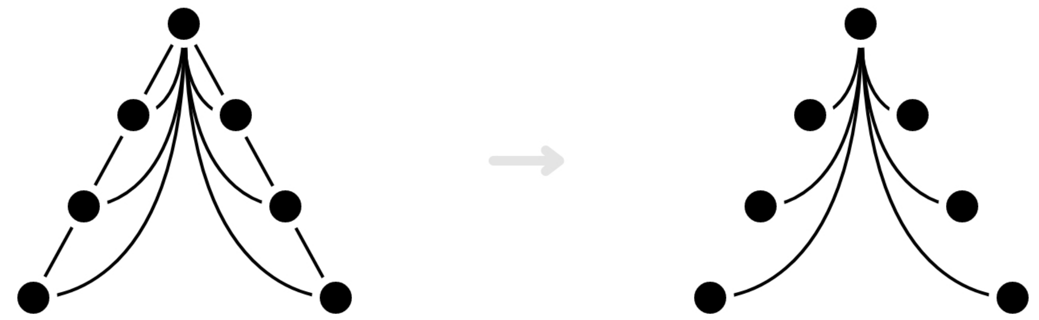
Randomized-Fiedler

Results


Algorithm Comparision (Formation Line)
Decision Tree
Random
Small Step
Big Step
Small Leverage
Big Leverage
Small Gradient
Big Gradient
Achived Fiedler Value
Desired Fiedler Value
0.45
0.5
0.6
0.75
1
1.25
1.5
1.75
2
2.5


Algorithm Comparision (Formation Wedge)
Decision Tree
Random
Small Step
Big Step
Small Leverage
Big Leverage
Small Gradient
Big Gradient
Achived Fiedler Value
Desired Fiedler Value

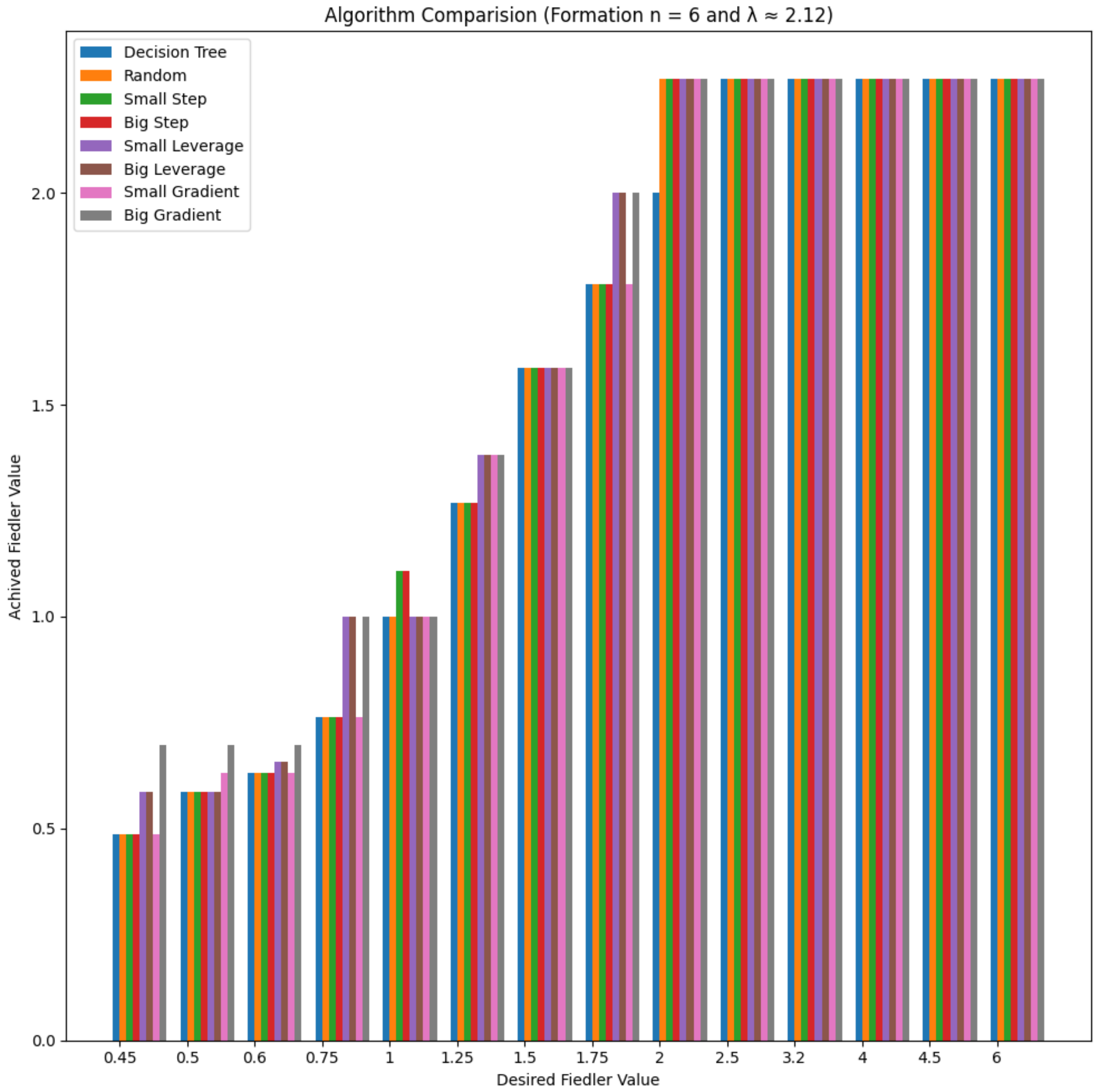

Algorithm Comparision (Formation Circular)
Decision Tree
Random
Small Step
Big Step
Small Leverage
Big Leverage
Small Gradient
Big Gradient
Achived Fiedler Value
Desired Fiedler Value

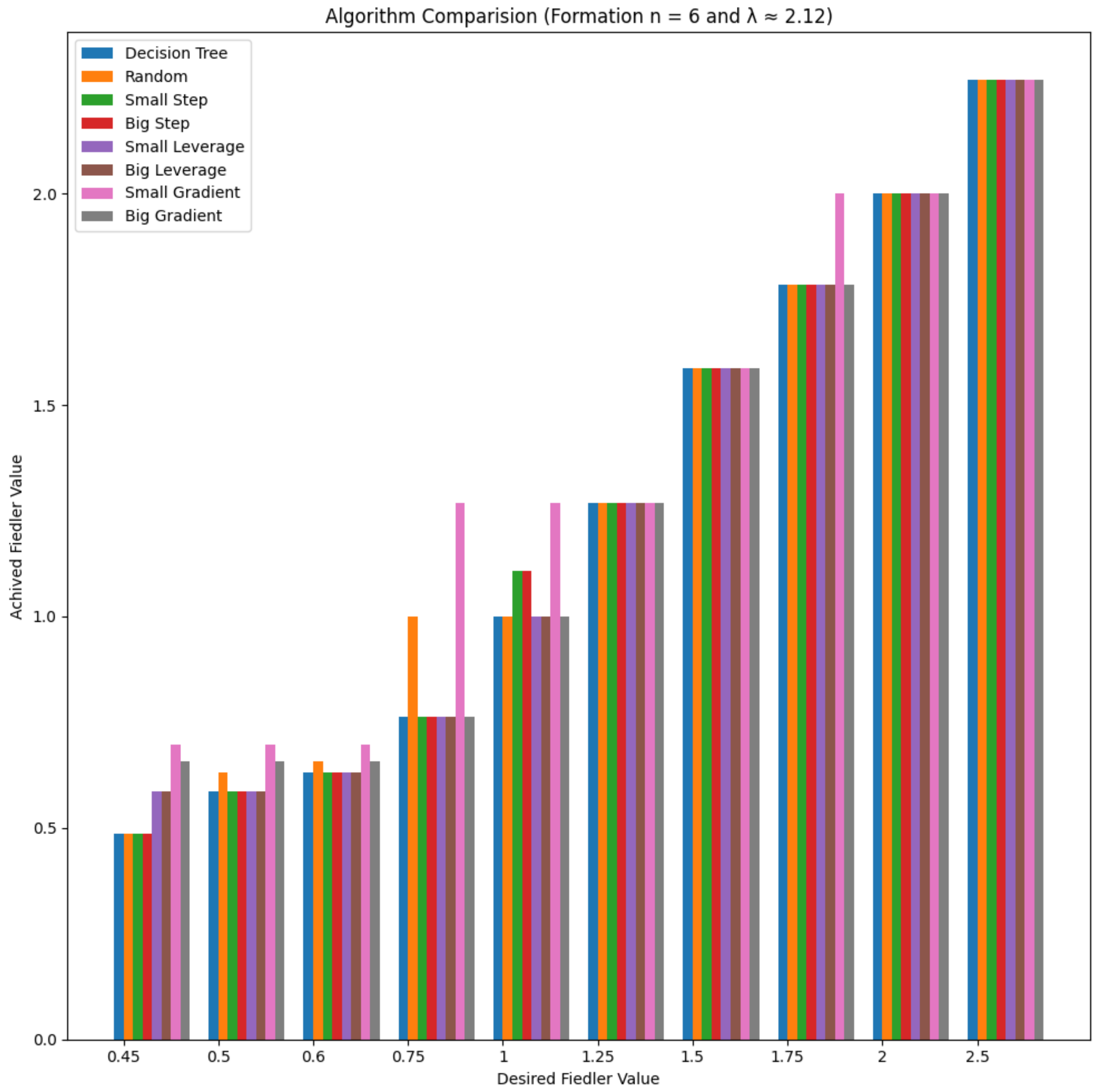


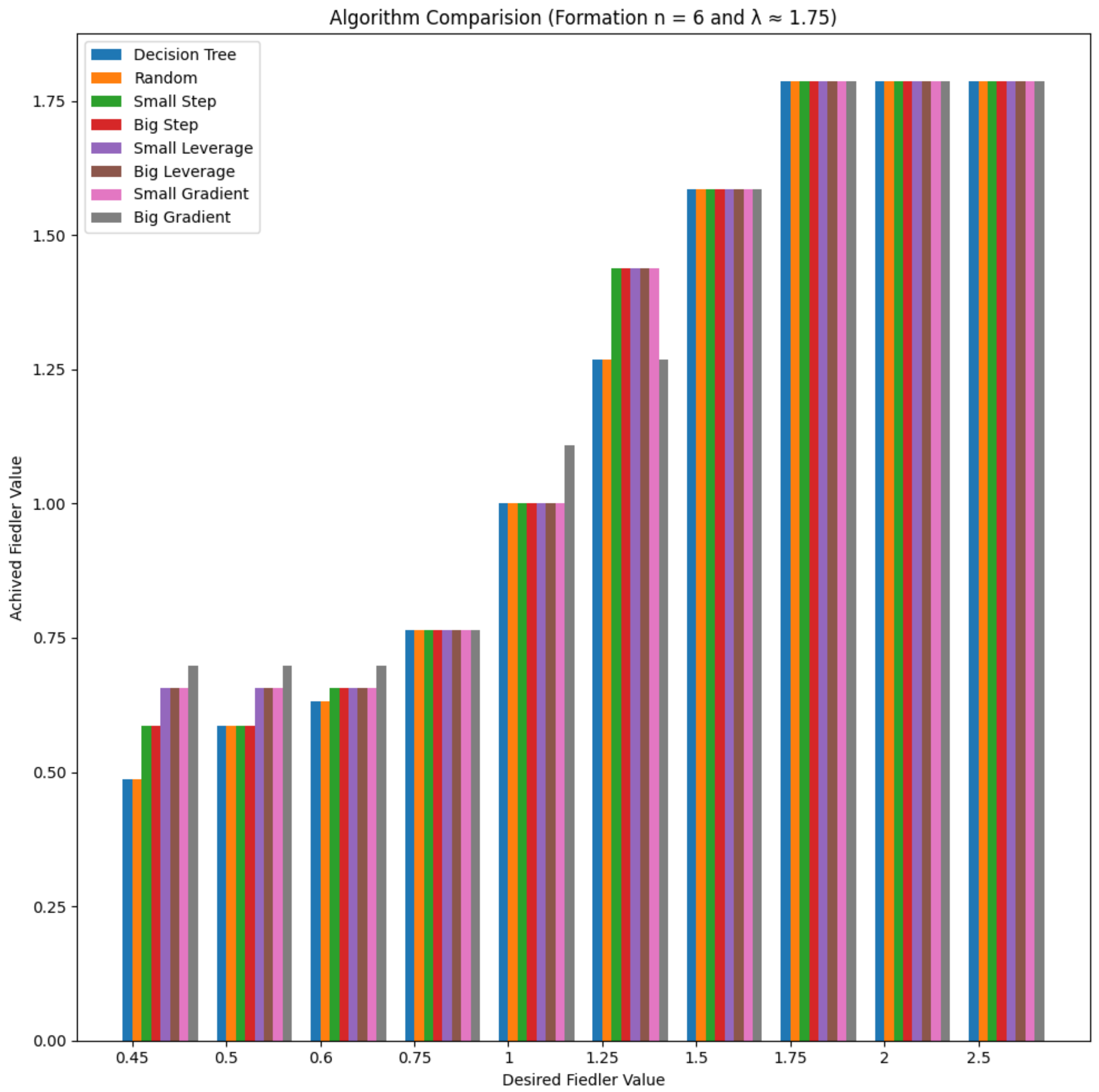
*~11/27 TESTS (N >= 5)

Results

*LOWERED AMPLIFICATION







Decision Tree is Expo(E) - Proposed Algorithms are Poly(E)

Takeaways: In our simulations, all algorithms had similar performance.

Randomized tends to perform best.

Randomized Algorithm

Removes a Random Valid Edge

Input:

G **Original Graph**

λ_2' **Target Fiedler**

r **Amplification Runs**

Output:

$\lambda_2(G')$ **Reached Fiedler**

G' **Augmented Graph**

Flags:

Allow Disconnect

One/Two Sided Bound

```
Function CreateGraph(g, target, runs) do
    graphs = []
    For i in range(runs):
        graphs += CutEdges(g, target)
    dist_to_target = [abs(f - target) for (f,g) in graphs]
    return min(dist_to_target, key=function o: o[0])
end
```

```
Function FindValidEdgeRemoval(graph, edge_set, f_current, target)
do
    edges_considering = copy_of(edge_set)
    while len(edges_considering) do
        u,v = random_edge(edge_set)
        f_next, g_next = self.graph_without_edge(graph,u,v)
        // Take edge if it brings us closer to the target
        if abs(f_next - target) > abs(f_current - target) do
            return f_next, g_next, (u,v)
        end
        edges_considering.remove((u,v))
    end
    // If no edge brought us closer, return null
    return null
end

Function CutEdges(g, target) do
    edge_set = g_edges_as_list(g)
    f_current = CalcFiedler(g)
    while f_current > target do
        res = FindValidEdgeRemoval(g, edge_set, f_current, target)
        if res == null do
            // No valid edges remain to remove
            break
        end
        // Remove edge, update graph, continue
        f_next, g_next, edge = res
        edge_set.remove(edge)
        f_current, g = f_next, g_next
    end
    return f_current, g
end
```

Maximum Algebraic Connectivity Augmentation

Damon Mosk-Aoyama. 2008. Maximum algebraic connectivity augmentation is

Instance: Given an undirected graph $G = (V, E)$, a non-negative integer k, and a non-negative threshold t.

Question: Is there a subset $A \subseteq E^C$ of size $|A| \leq k$ such that the graph $H = (V, E \cup A)$ satisfies $\lambda_2(H) \geq t$.

- ✔ **NP:** $\lambda_2(H) \geq t$ verifiable in polynomial time.
- ✔ **NP-Hard:** Reduction from 3-colorability.

NP-Complete

A Harder Version of Our Problem

Instance: Given an undirected graph $G = (V, E)$, a subset $A \subseteq E$, a non-negative integer k, and a non-negative threshold t_1 and t_2.

Question: Is there a subset $B \subseteq (E - A)$ of size $|B| \geq k$ such that the graph $H = (V, E - B)$ satisfies $t_1 \leq \lambda_2(H) \leq t_2$.

- ✔ **NP:** $\lambda_2(H) \geq t$, $B \subseteq E$, and $|B| \geq k$ verifiable in polynomial time.
- ✔ **NP-Hard:** Reduction from the "maximum algebraic connectivity augmentation problem".

NP-Complete

Future Works

- Is the original question we proposed NP-Hard?
- What about reducing weights on graphs?
- What about optimizing the number of removed edges to meet the threshold?
- Can we split a graph into separate components with desired Fiedler values (same or different)?

Networking and Robotic Applications

Problem: Can we enable a system engineer or real-time supervisor to constrain the algebraic connectivity (i.e. Fiedler value) of a mobile multi-agent system by augmenting a flock's network topology?

Solution: Selecting and maintaining a subset of edges in a graph as to dictate the final graph's algebraic connectivity (i.e. Fiedler value).

ENVIRONMENT & LIMITATIONS

Reducing Transmission Noice or Network Traffic Per Time Unit

NETWORK EFFICIENCY

More Efficient Information Distribution Through Broadcasting (Rather than Routing)

CONSENSUS & RESILIENCE

Controlling the Speed of Consensus or Identifying Malicious Agents Through Multi Path Routing

Continuing This Line of Work

CS 229R: THIS PAPER

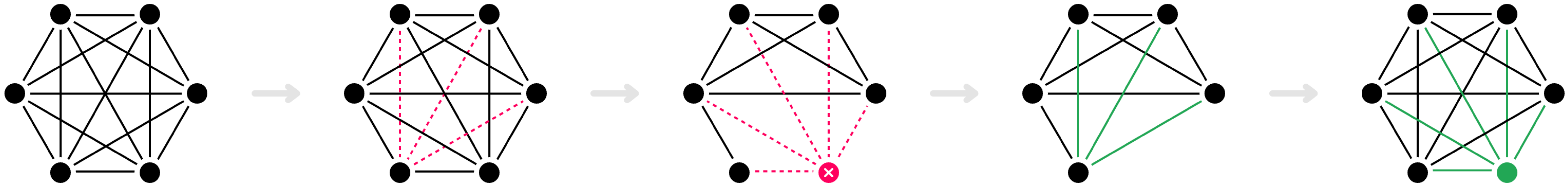
1 Investigating Methods of Controlling Algebraic Connectivity

CS 286

2 Dictating Algebraic Connectivity as a Topology in Networked Multi-Agent Systems

EXTENSION

3 Using Multi-Path Routing to Identify Malicious Agents in Consensus



Thank You!

Any Questions?