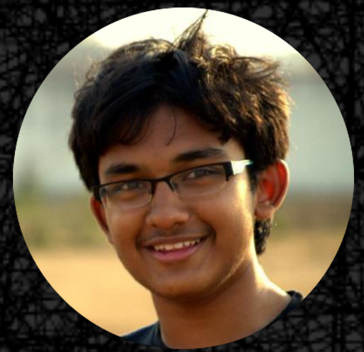
Atlas

Local Graph Exploration in a Global Context

IUI 2019

Fred Hohman
@fredhohman
Georgia Tech

James Abello
Rutgers



Varun Bezzam
Georgia Tech

Polo Chau
Georgia Tech

Polo Club of DATA SCIENCE

Georgia Tech

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Graph Sensemaking

Global View

Graph Sensemaking

Local View

Global View

Free Exploration

Graph Sensemaking

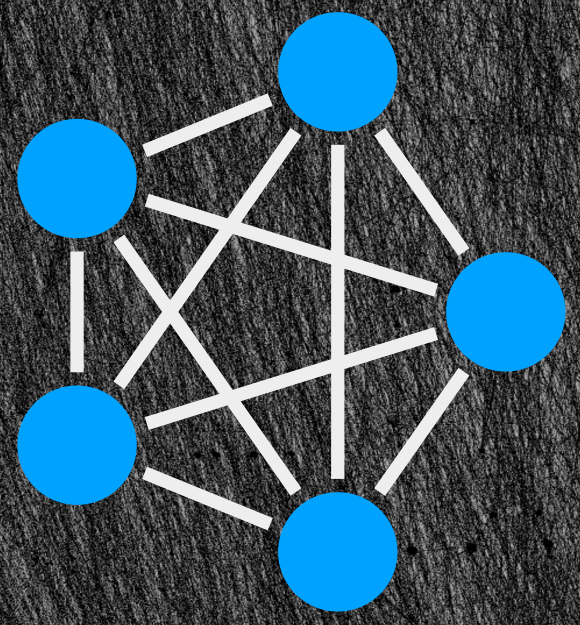
Local View

Targeted Exploration

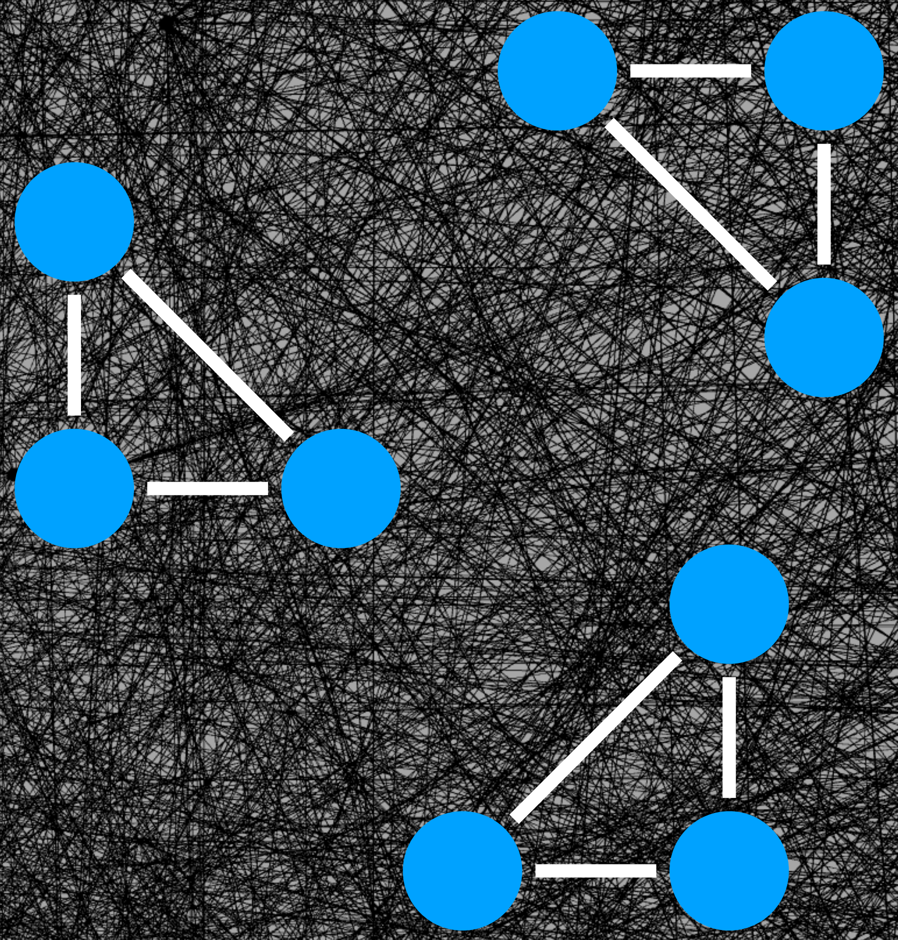
Important Structure

Graph Sensemaking

Important Nodes

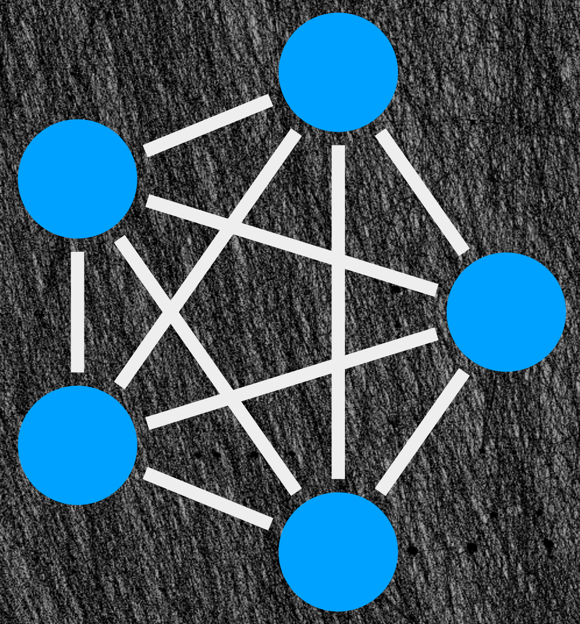


Important Structure

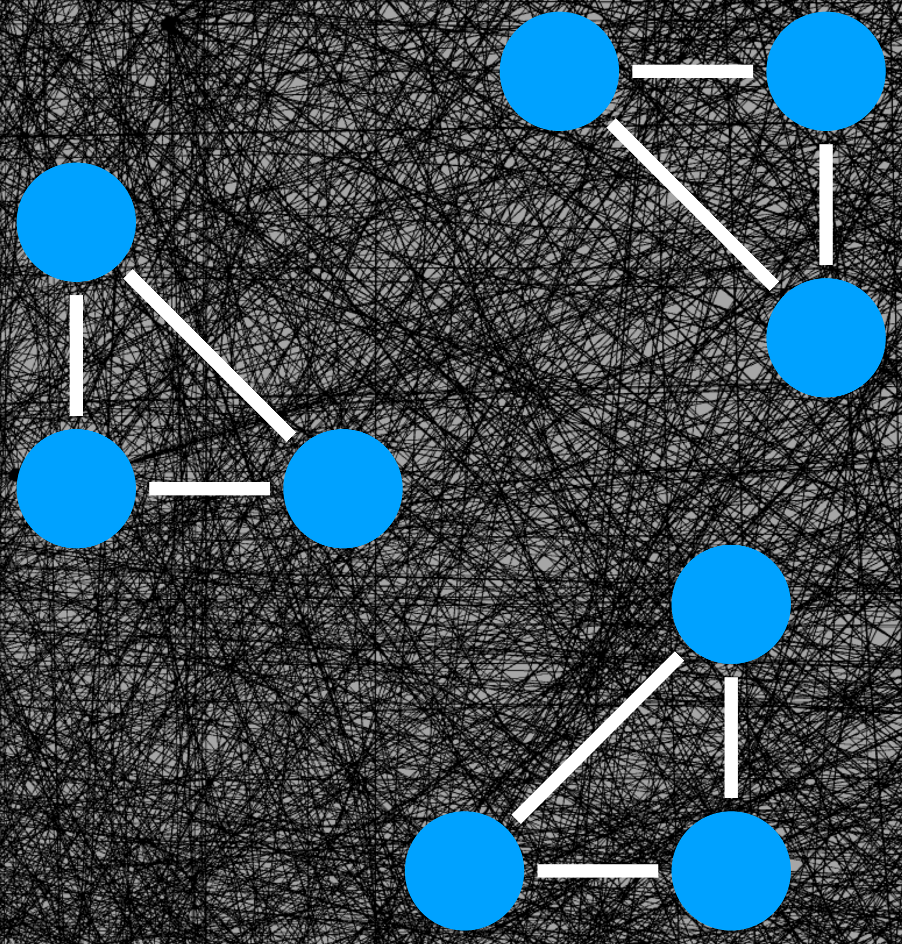


Graph Sensemaking

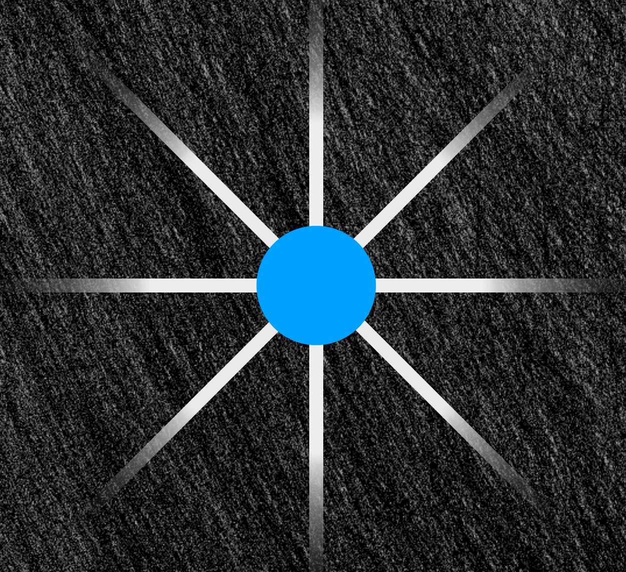
Important Nodes



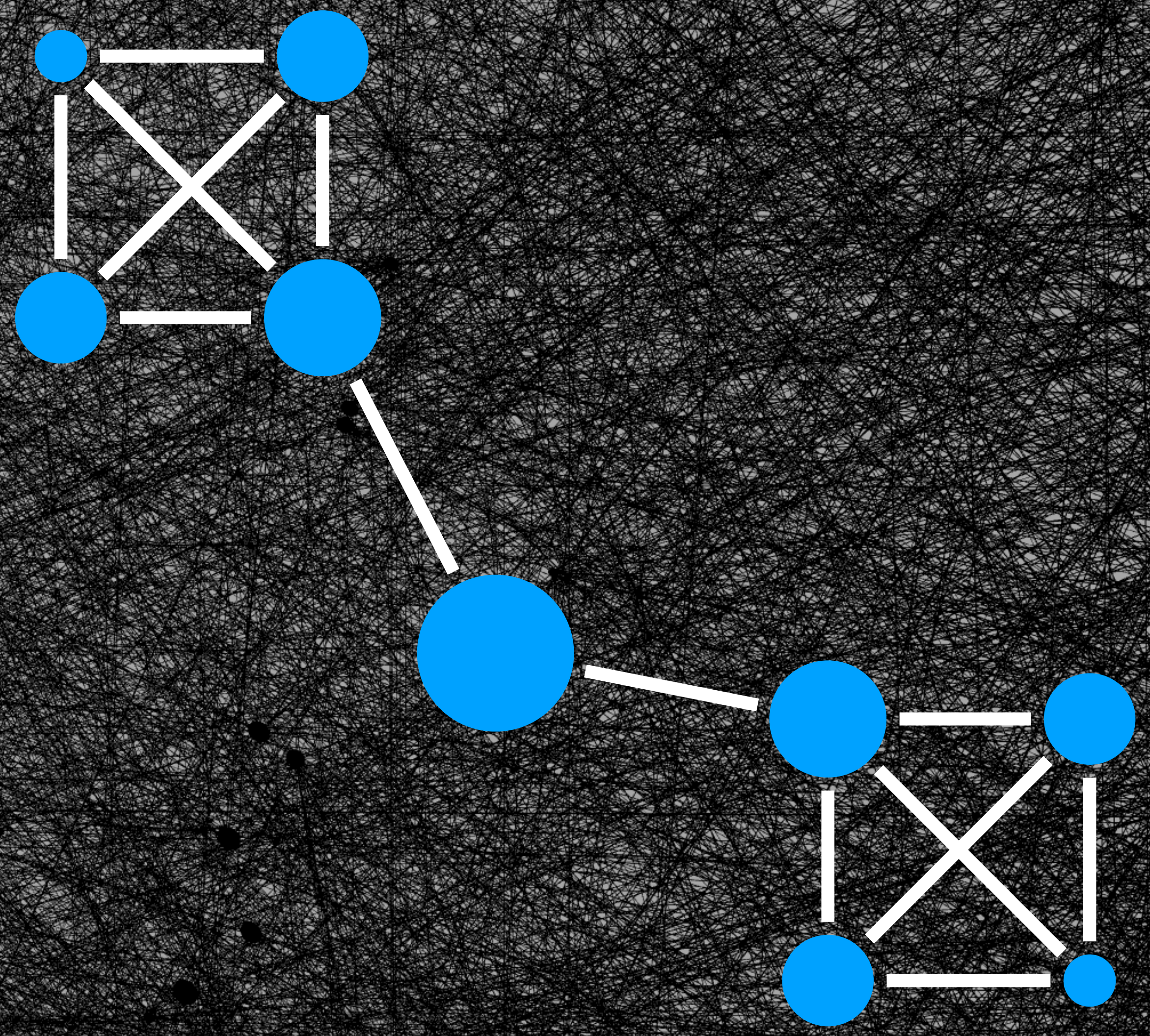
Important Structure



Graph Sensemaking

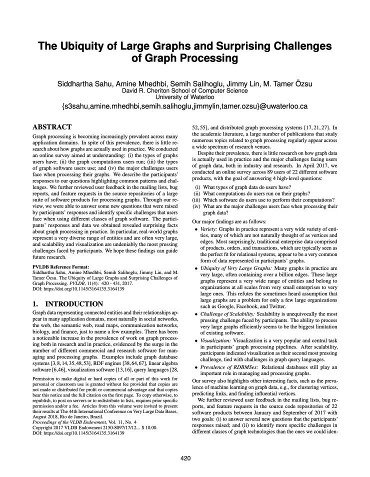


Important Nodes



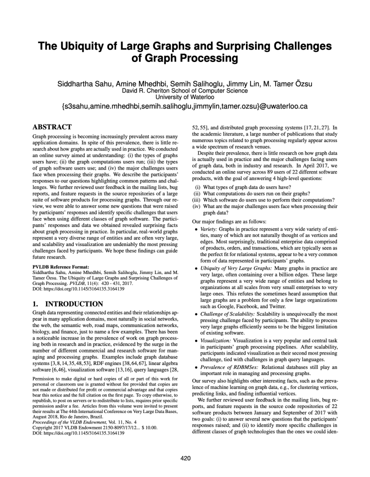
Data Mining	HCI Human-computer Interaction
Automatic	User-driven, iterative
Summarization, clustering, classification	Interactive, visualization
Millions of nodes	Thousands of nodes

Data Mining	HCI Human-computer Interaction
Automatic	User-driven, iterative
Summarization, clustering, classification	Interactive, visualization
Millions of nodes	Thousands of nodes

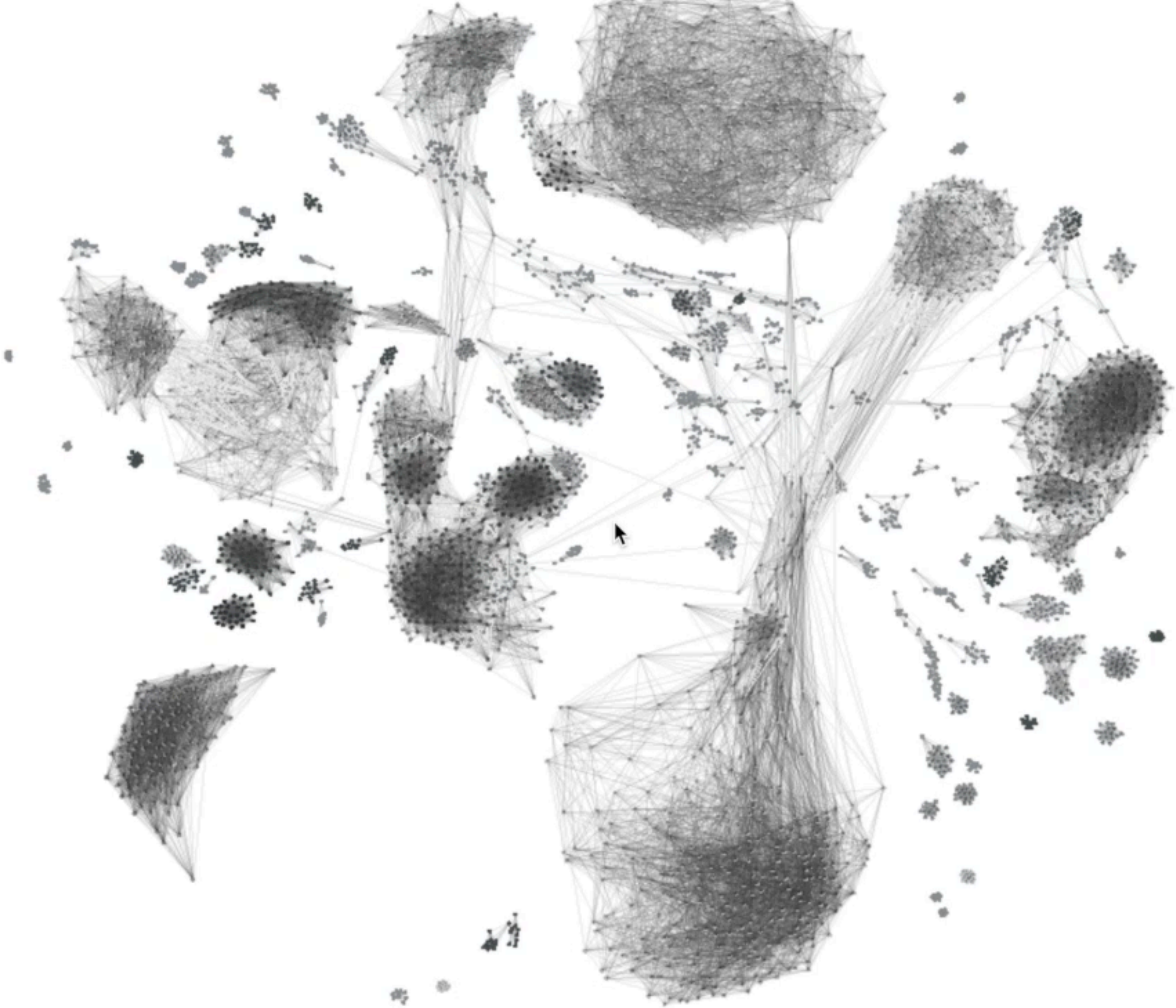

The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing

The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. Sahu, et al. *VLDB, 2017.*

Data Mining	**HCI** Human-computer Interaction
Automatic	User-driven, iterative
Summarization, clustering, classification	Interactive, **visualization**
Millions of nodes	Thousands of nodes


The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing

The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. Sahu, et al. *VLDB, 2017.*



Atlas

bit.ly/atlas-iui

interactive graph exploration via **scalable edge decomposition**

Atlas

bit.ly/**atlas-iui**

interactive graph exploration via **scalable edge decomposition**

- separate graph into ***graph layers***

Atlas

bit.ly/**atlas-iui**

interactive graph exploration via **scalable edge decomposition**

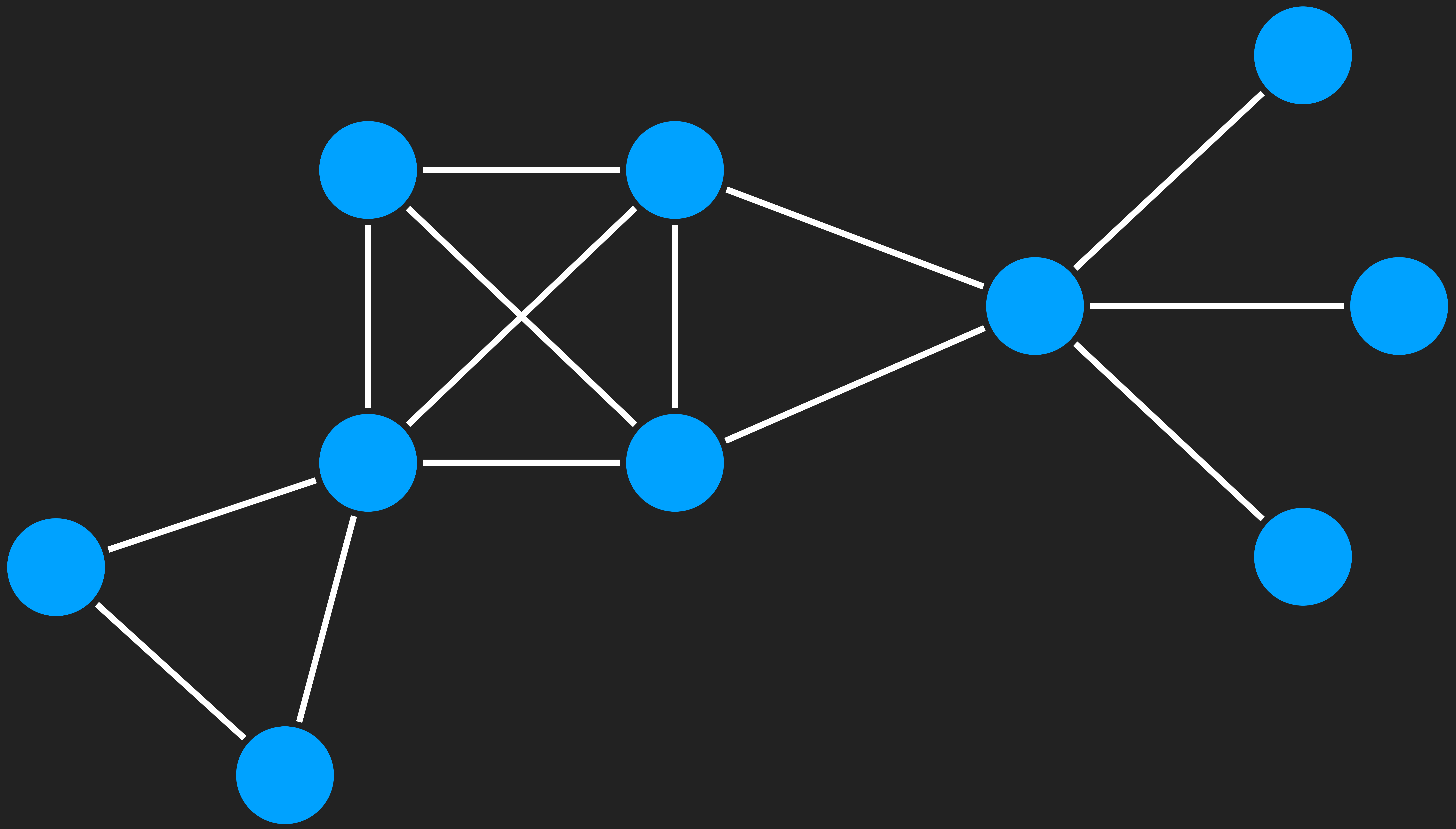
- separate graph into ***graph layers***
- reveal peculiar subgraph

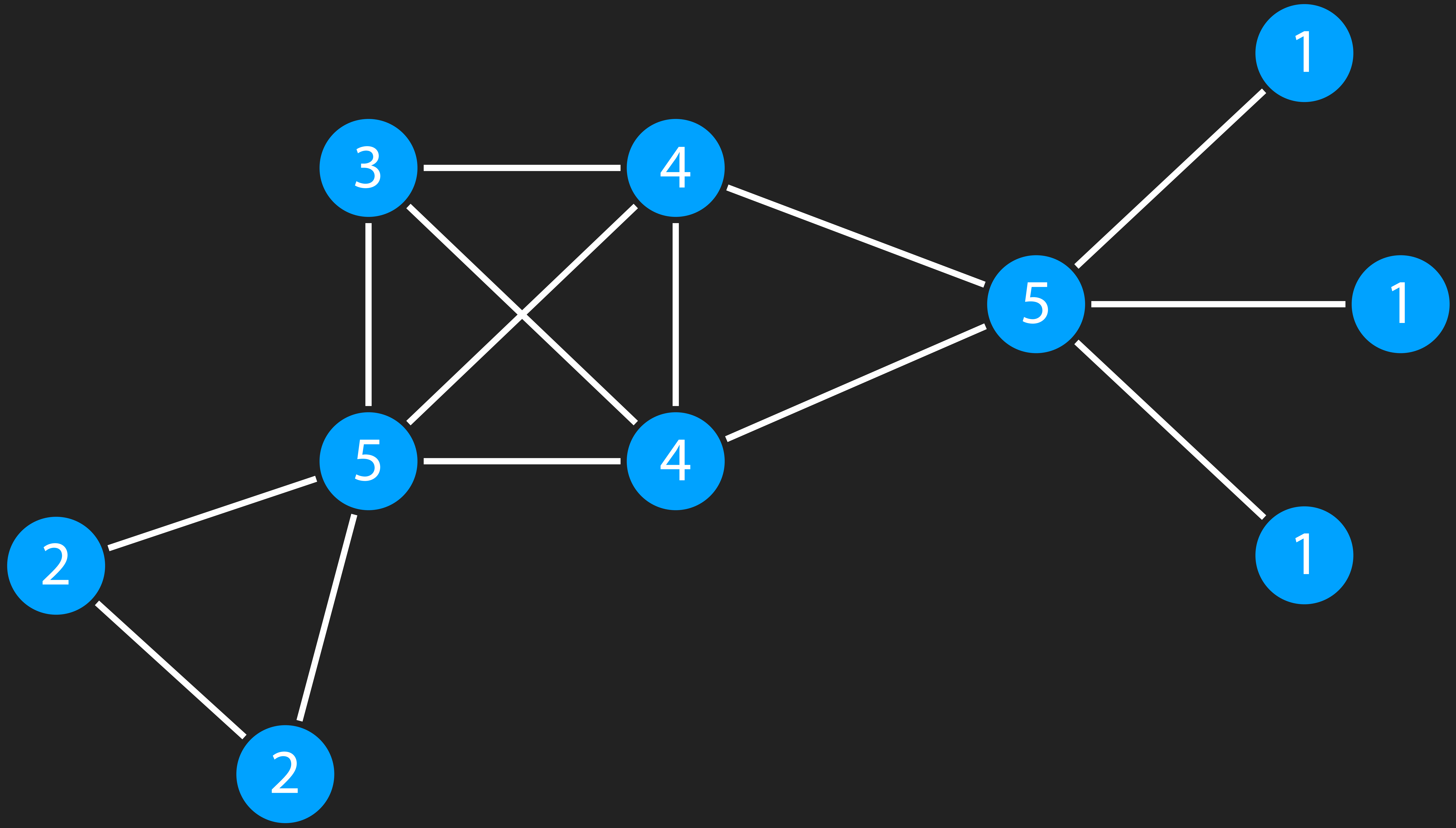
Atlas

bit.ly/atlas-iui

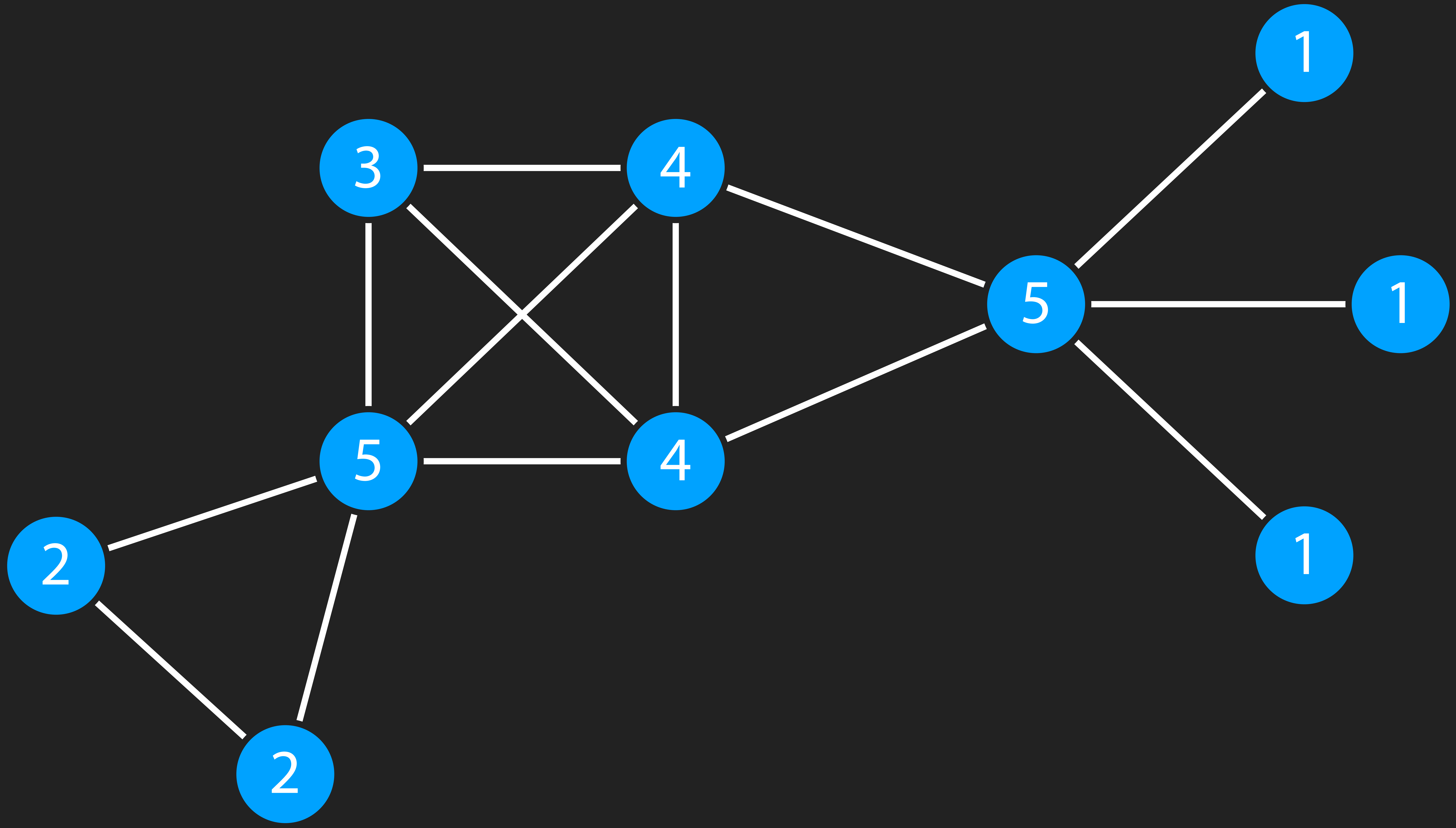
interactive graph exploration via **scalable edge decomposition**

- separate graph into ***graph layers***
- reveal peculiar subgraph
- visualize local + global structure

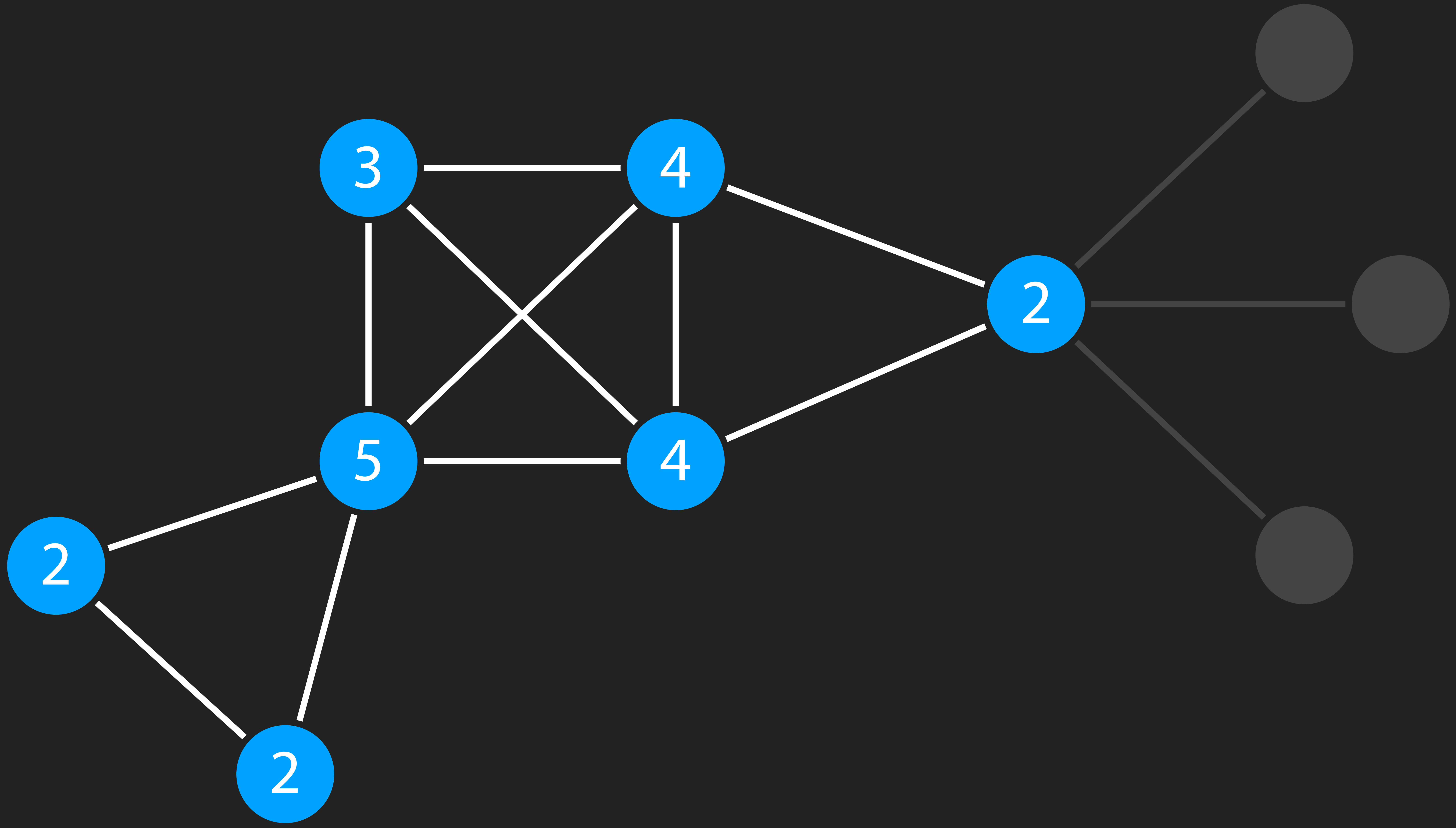



1
3
4
5
1
5
4
1
2
2

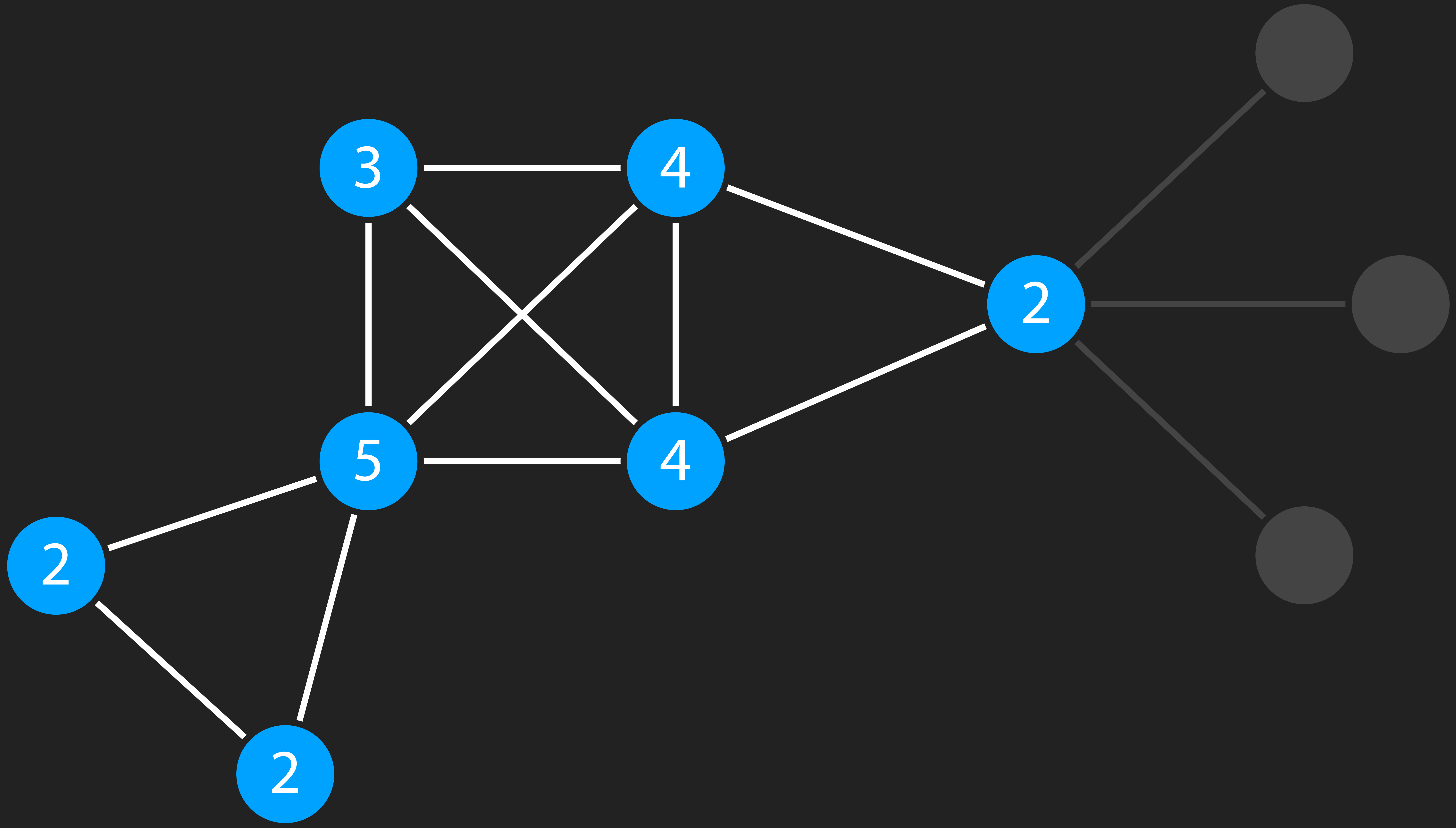
```
peel = 1
```



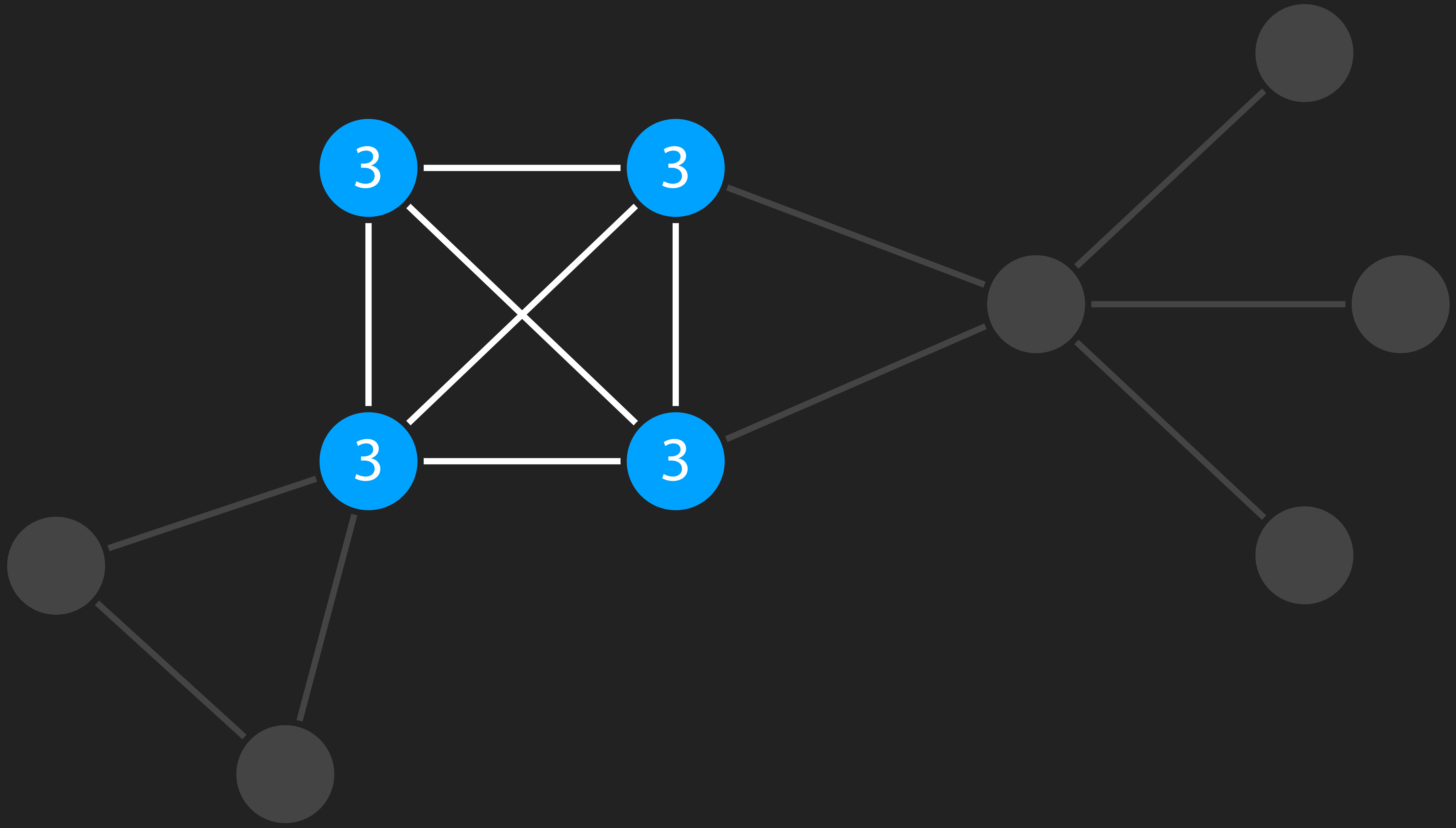
`peel = 1`



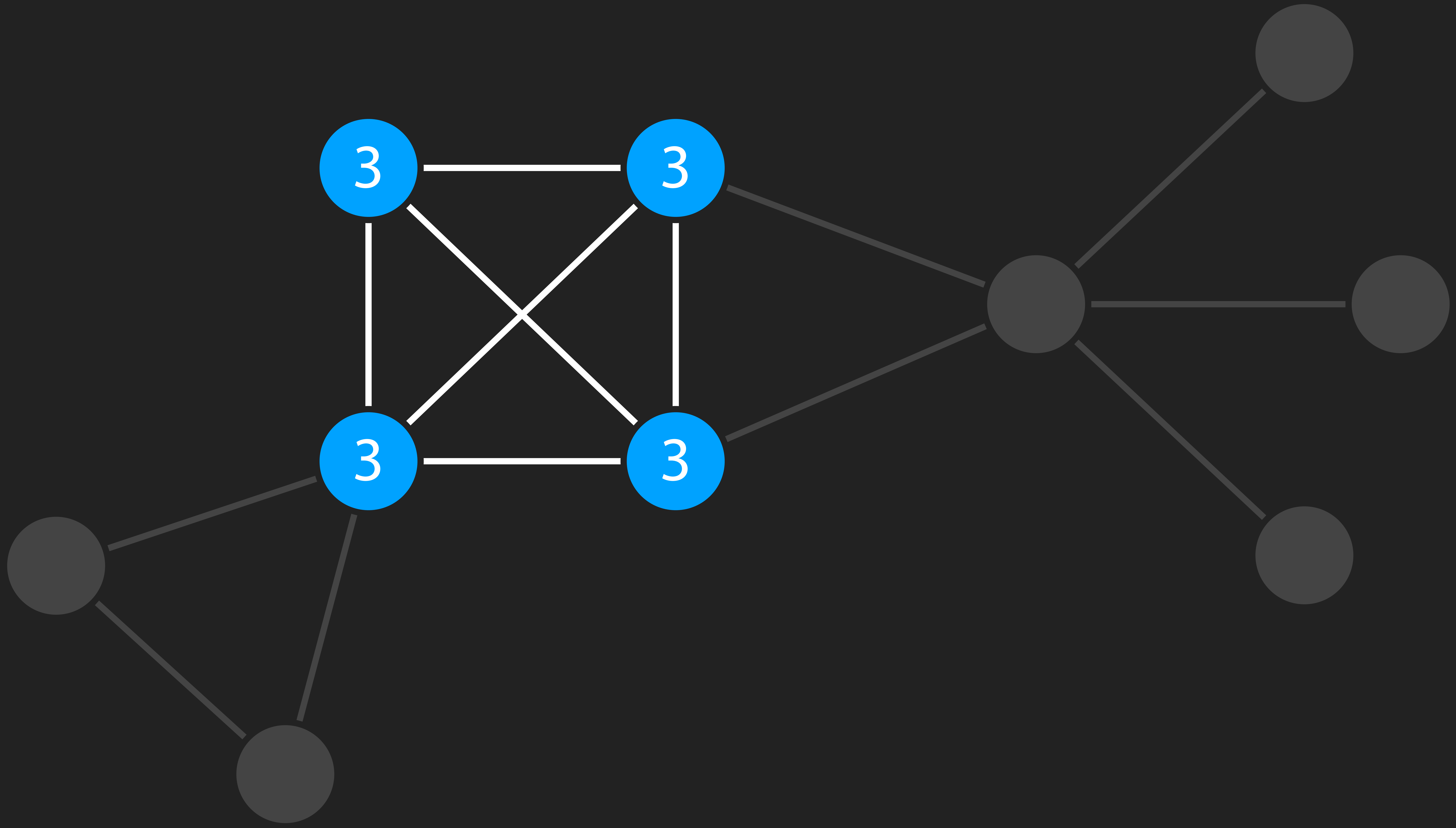
```
peel = 2
```



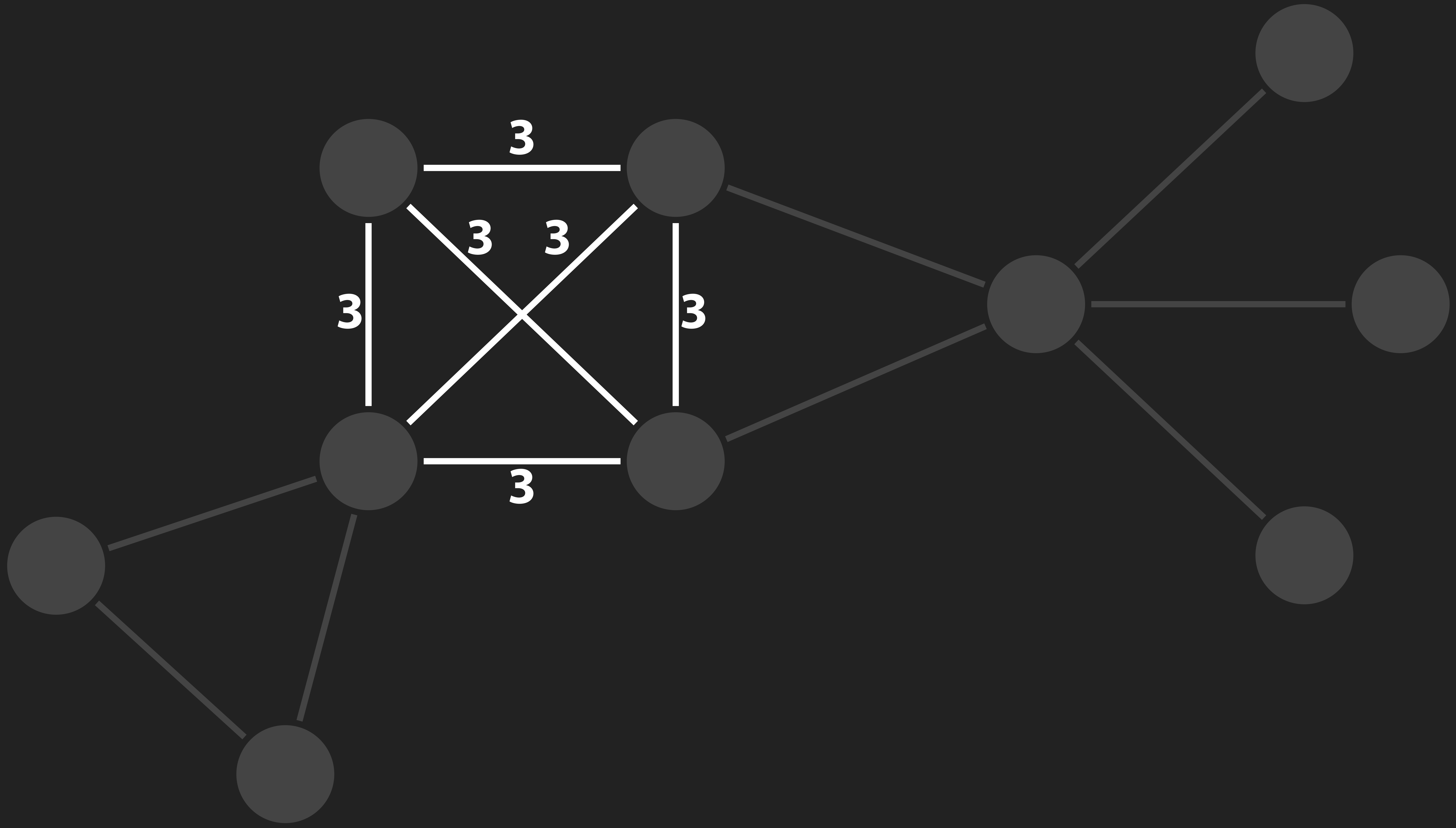
```
peel = 2
```



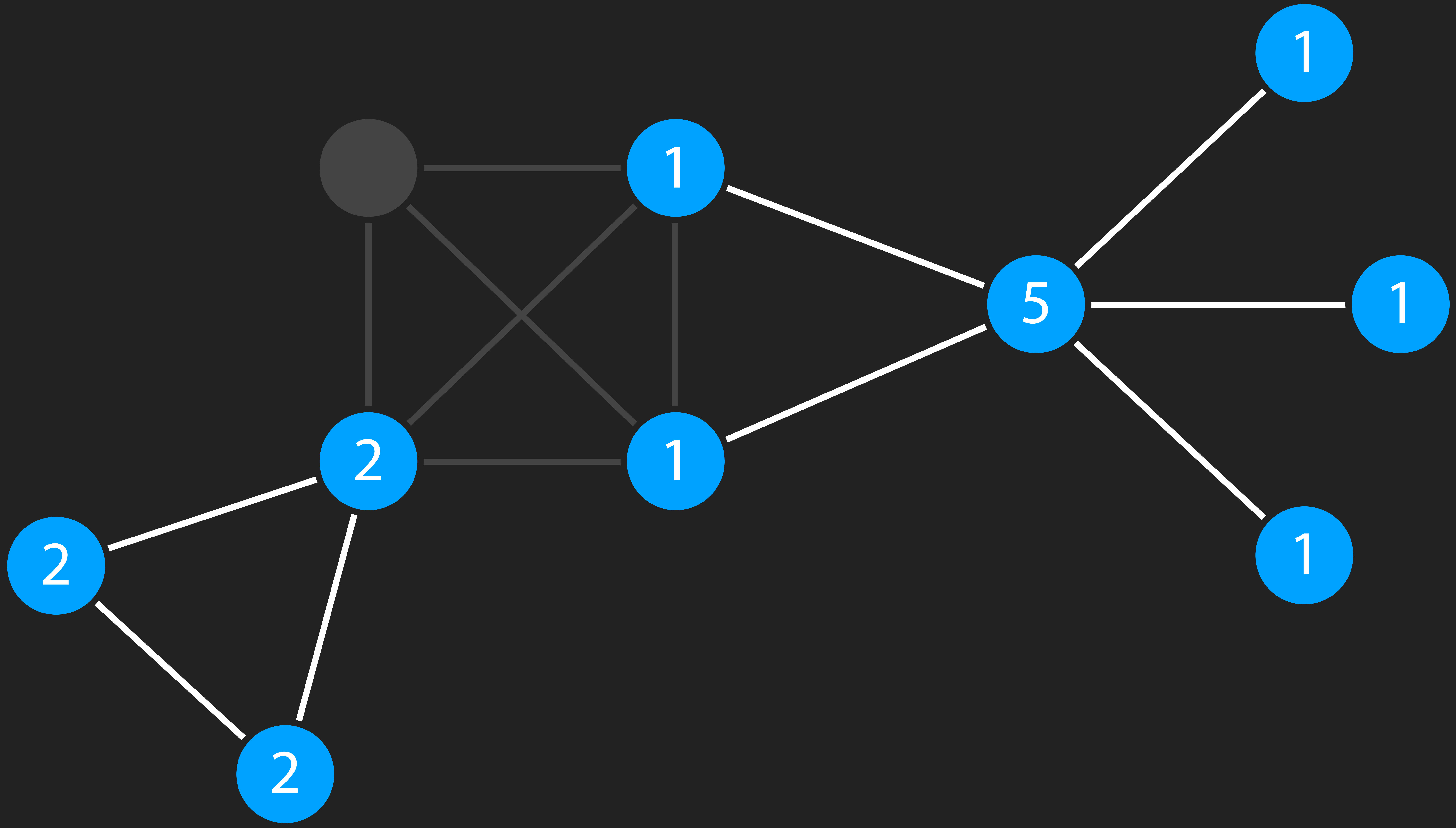
```
peel = 3
```



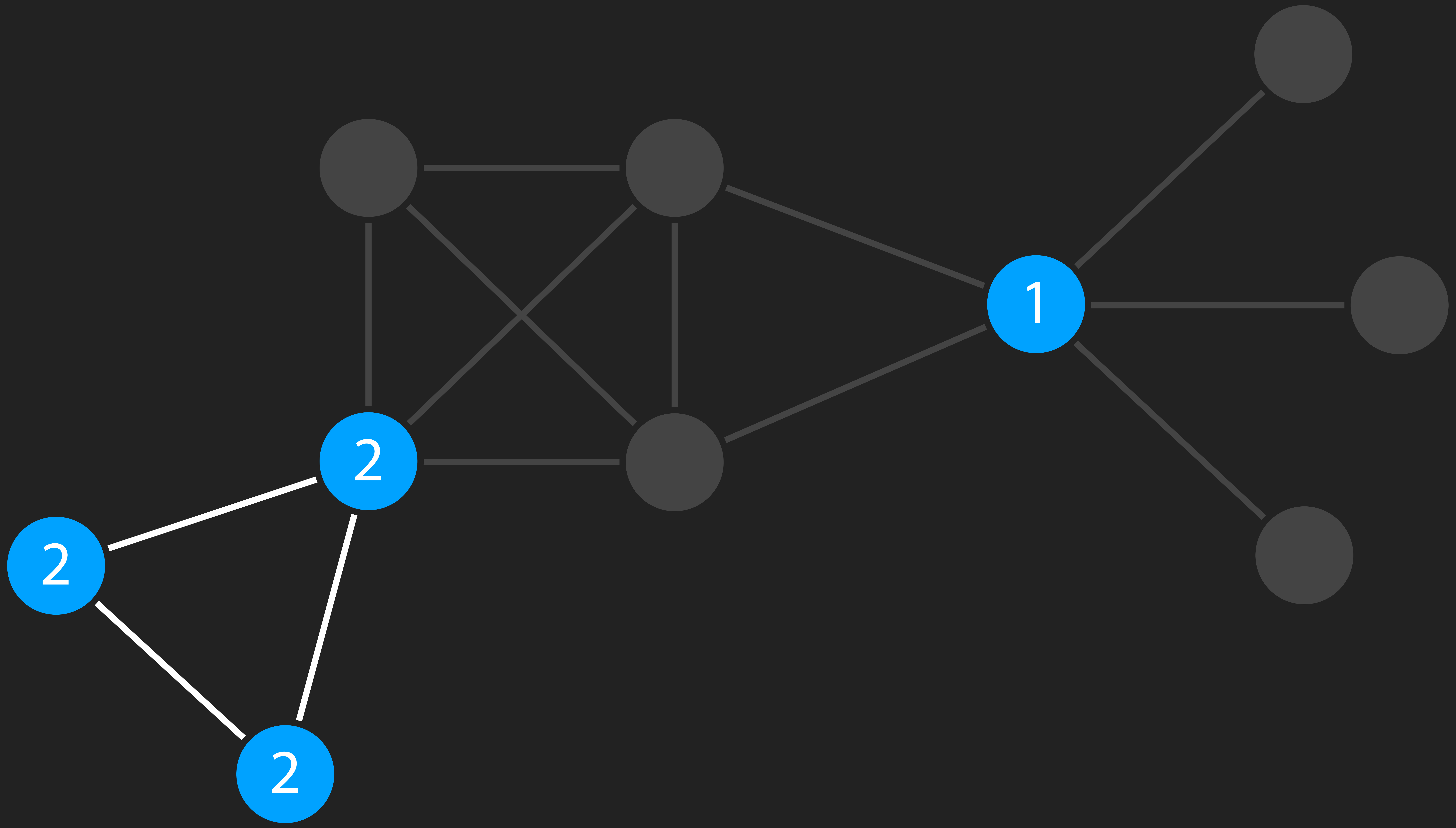
```
peel = 3
```



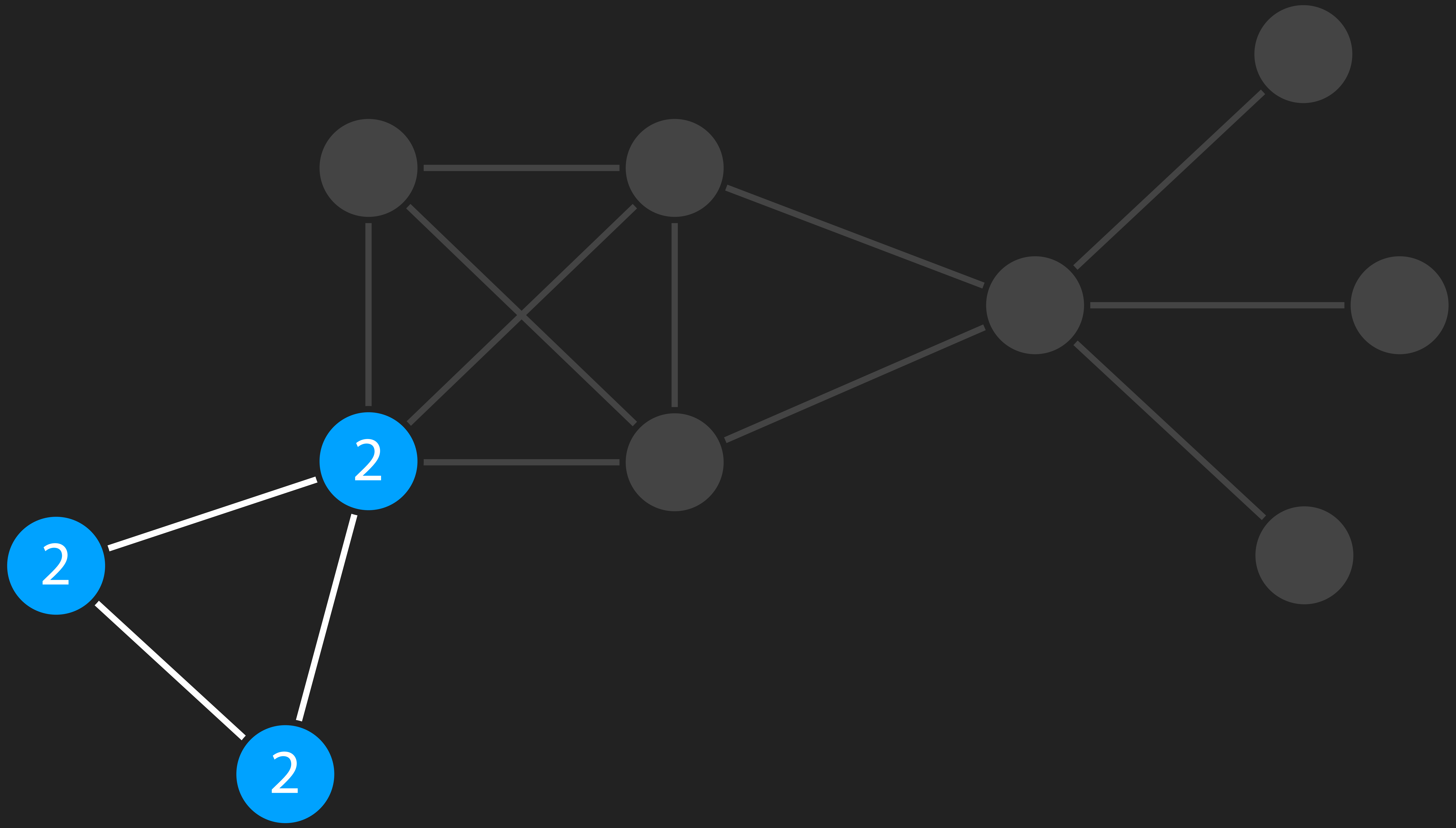
```
peel = 1
```



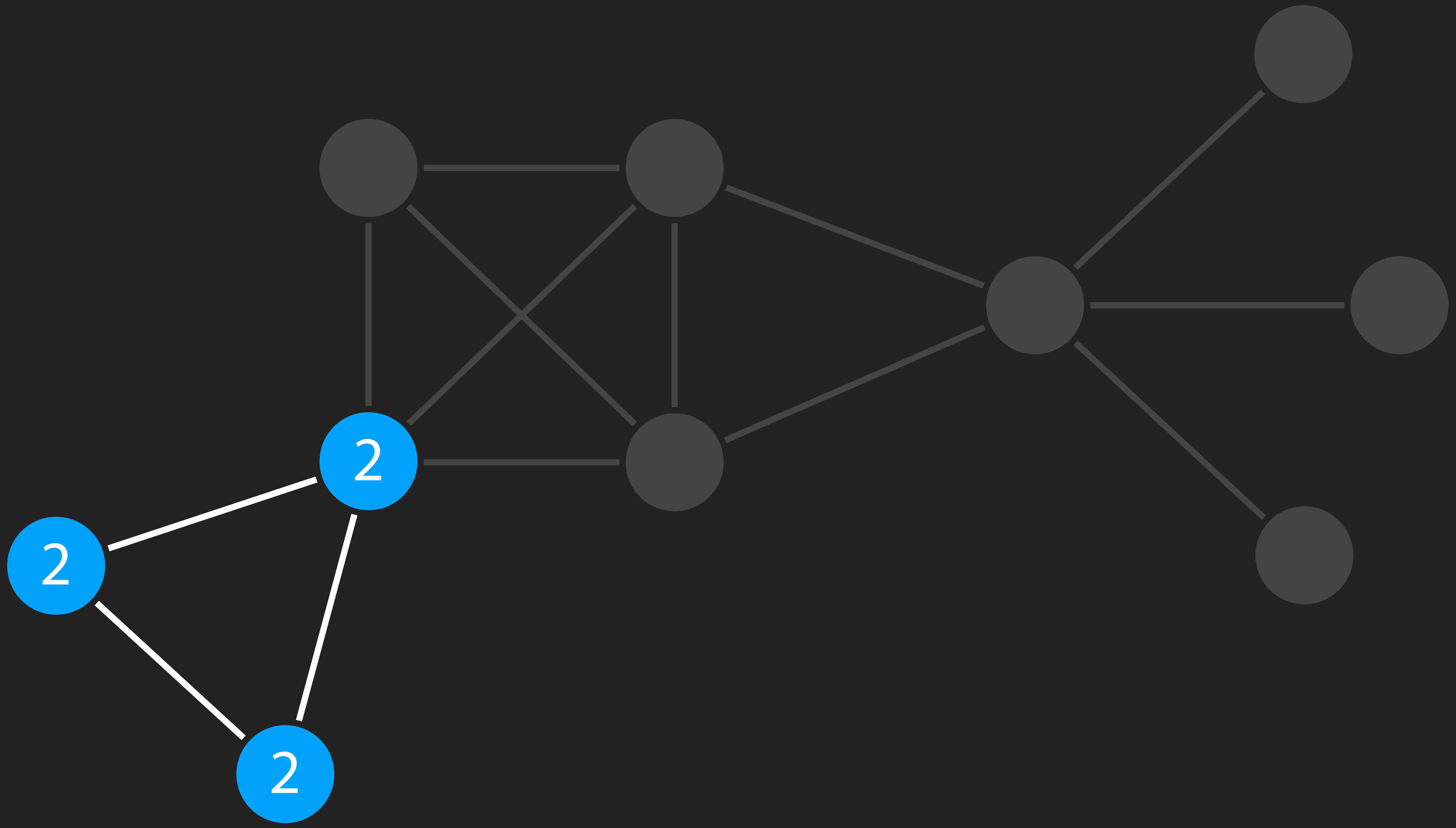
peel = 1



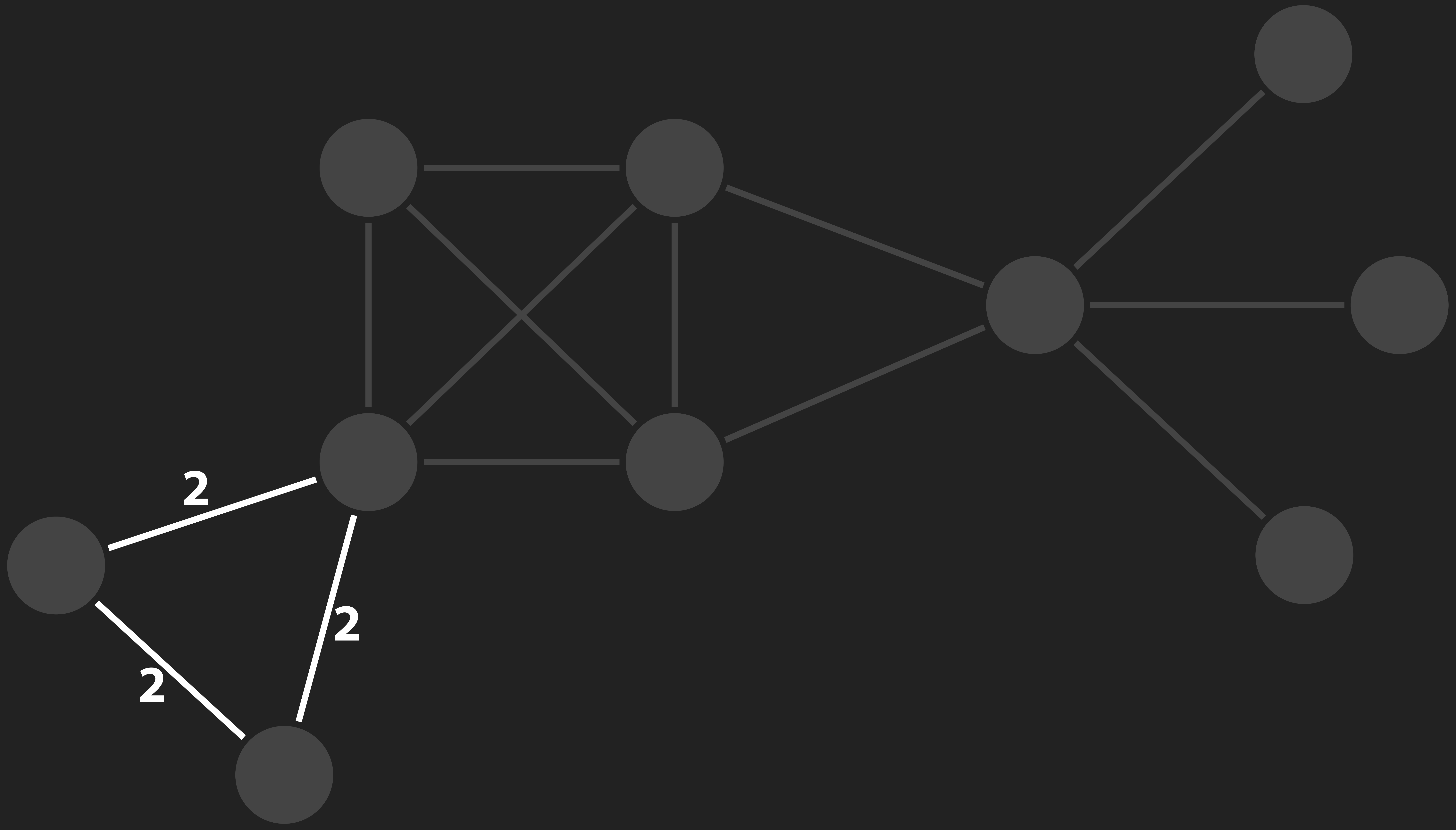
`peel = 1`



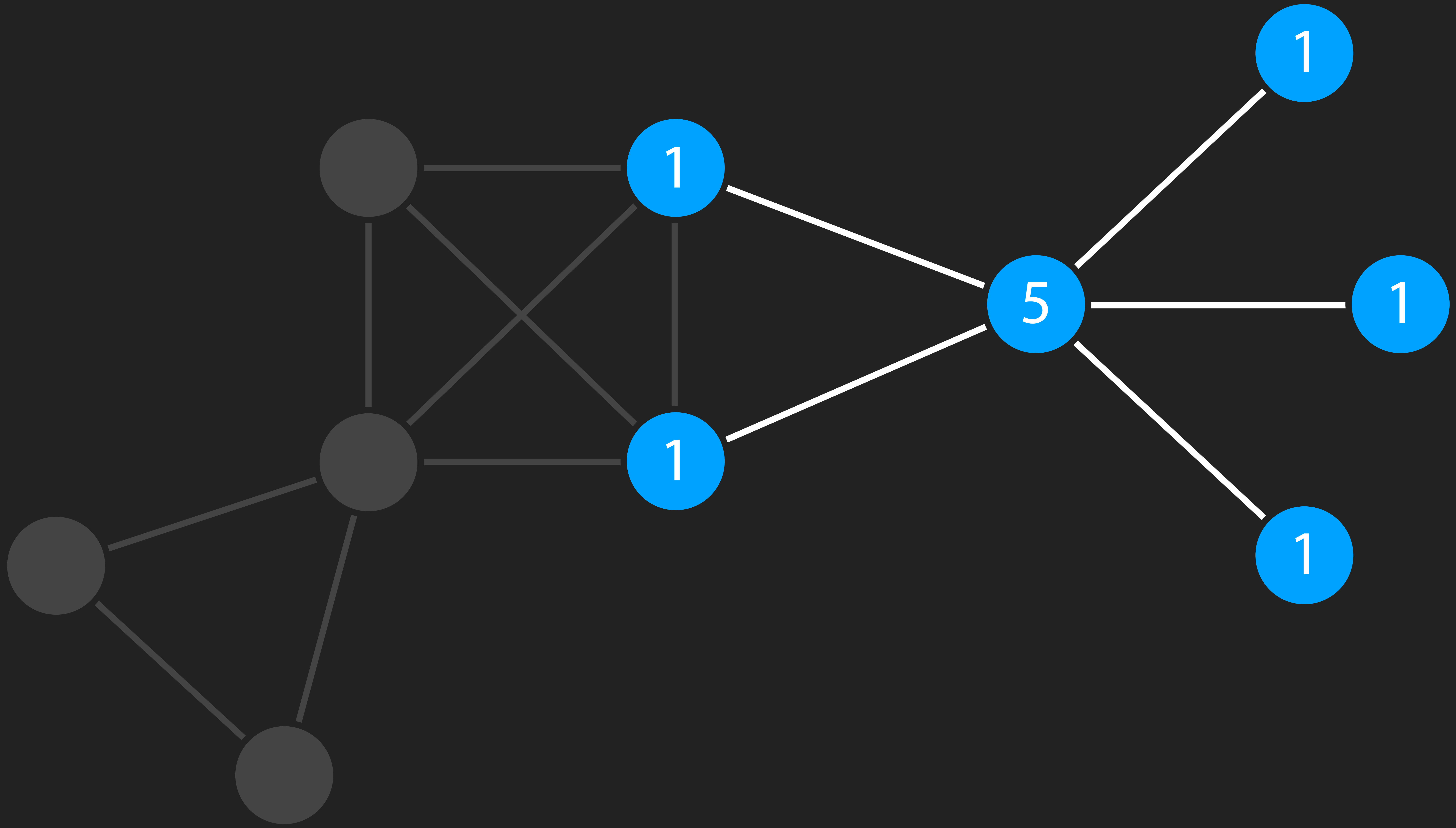
`peel = 2`



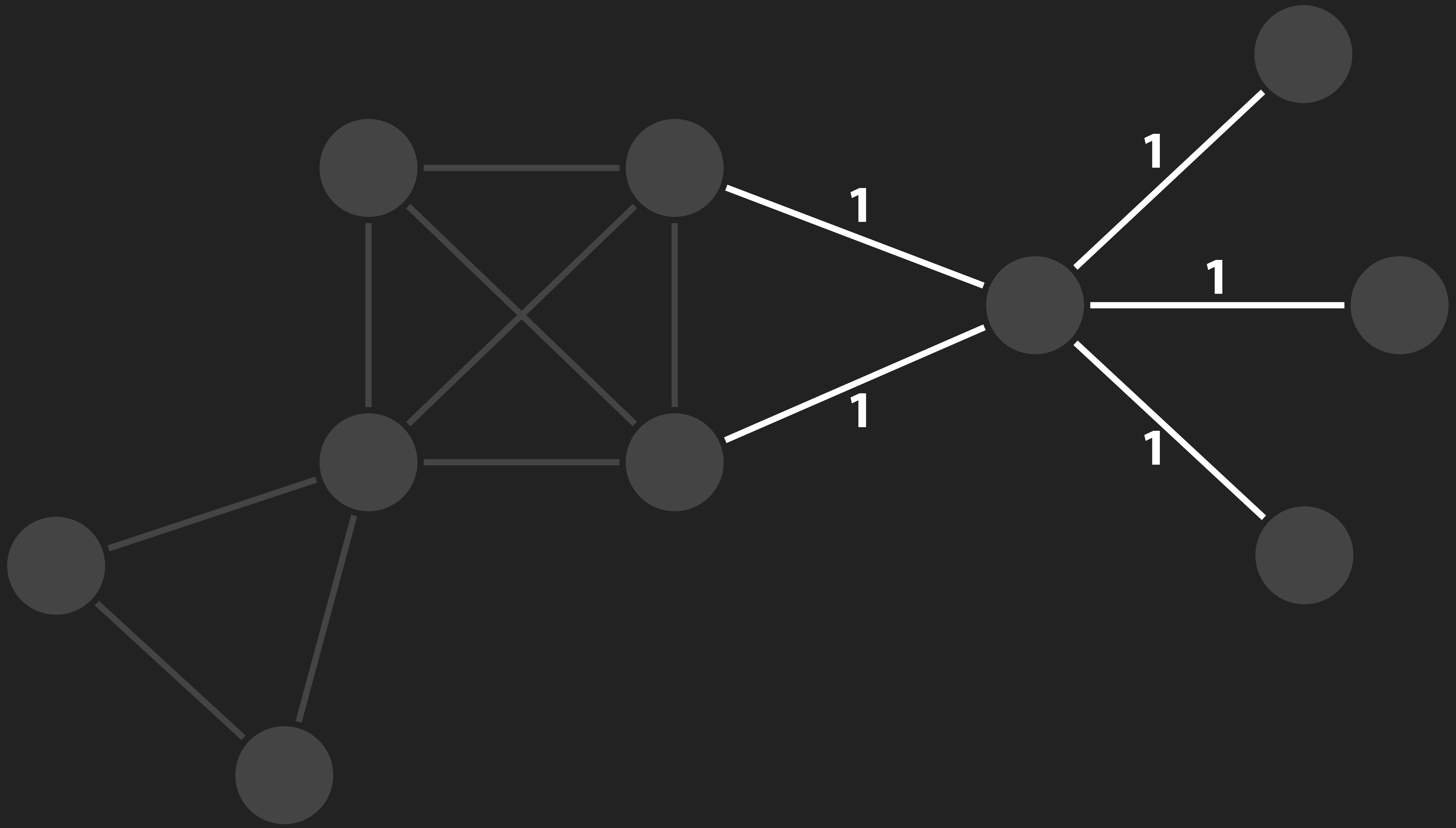
peel = 2

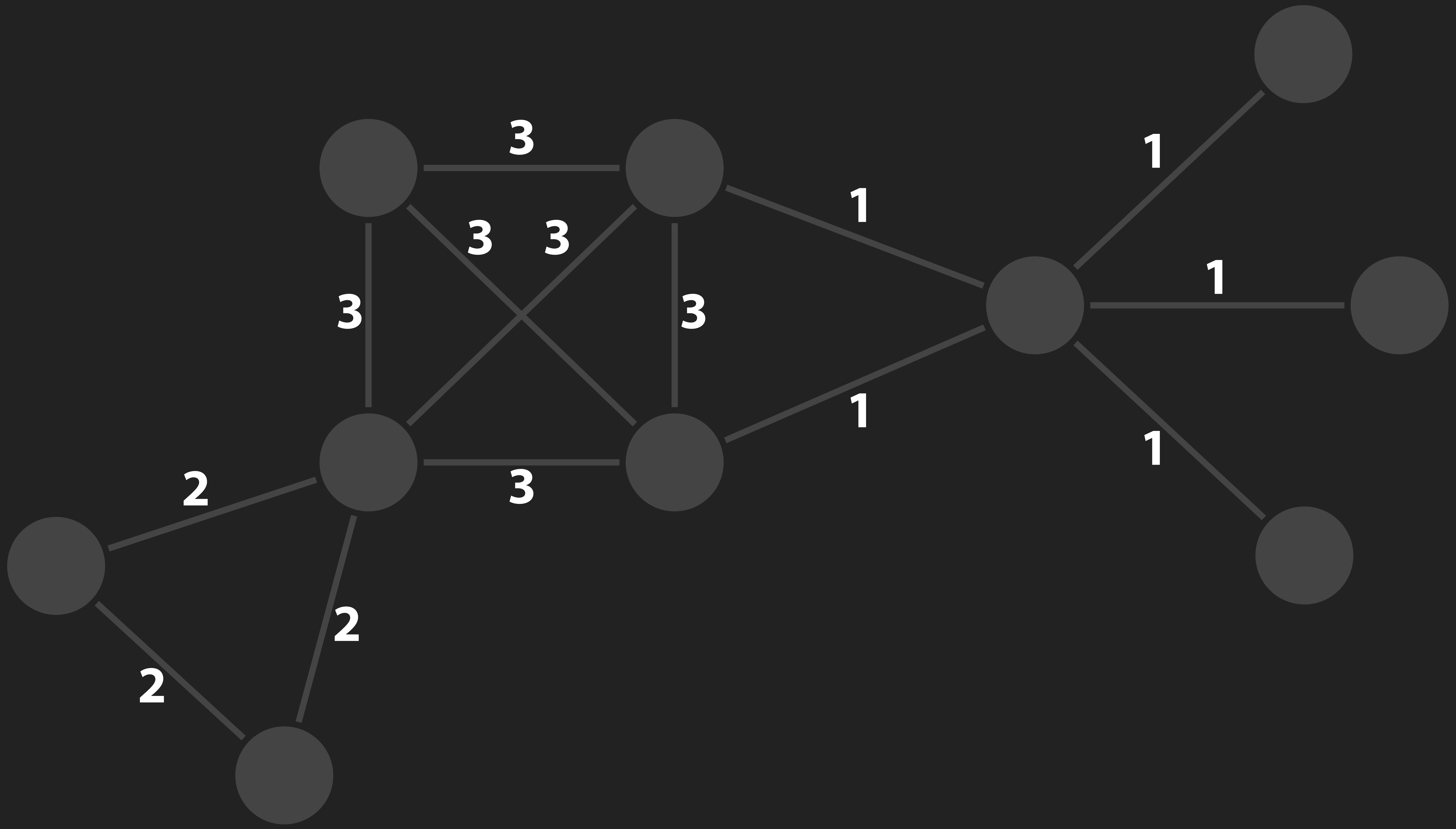


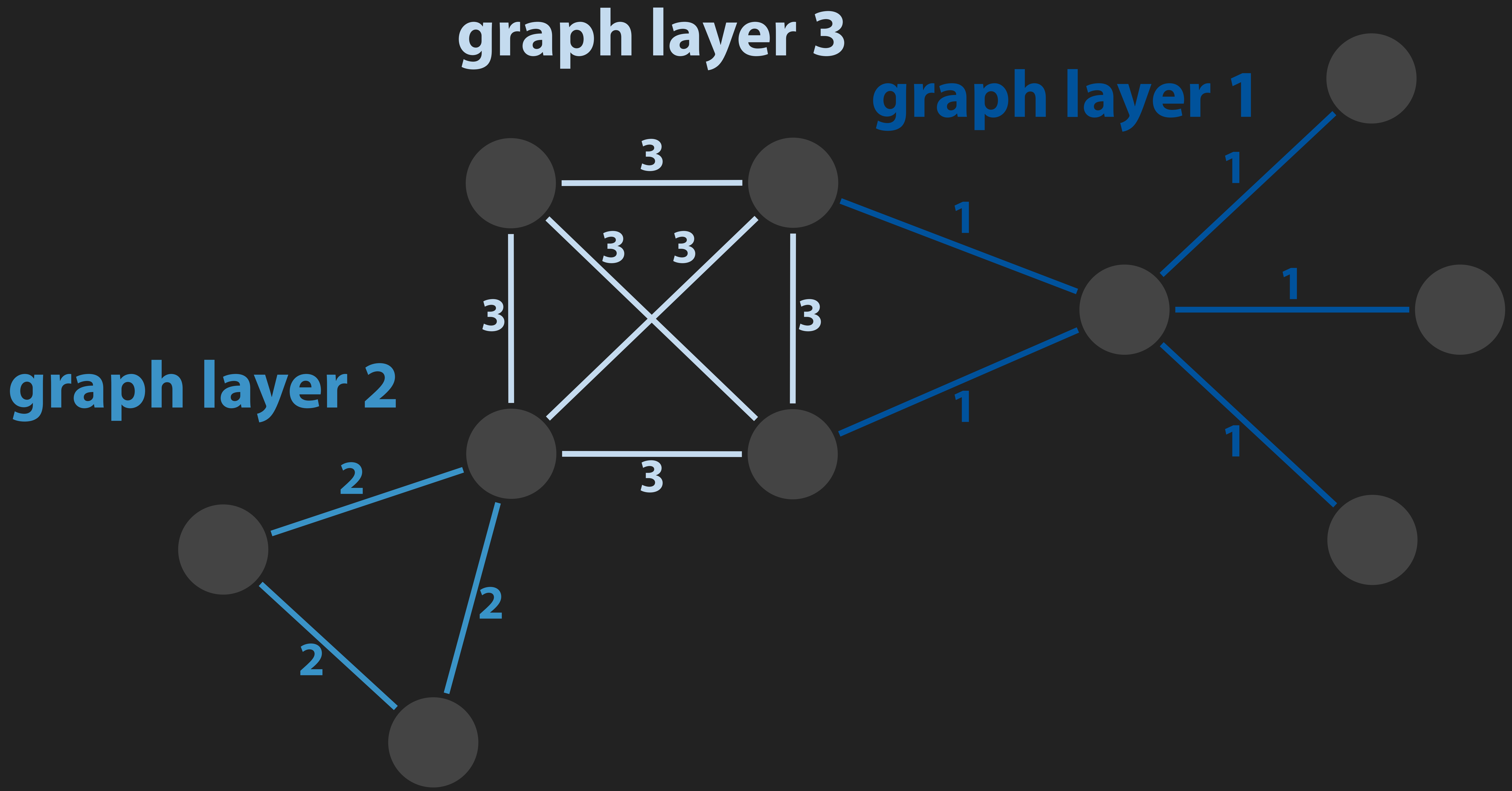
```
peel = 1
```

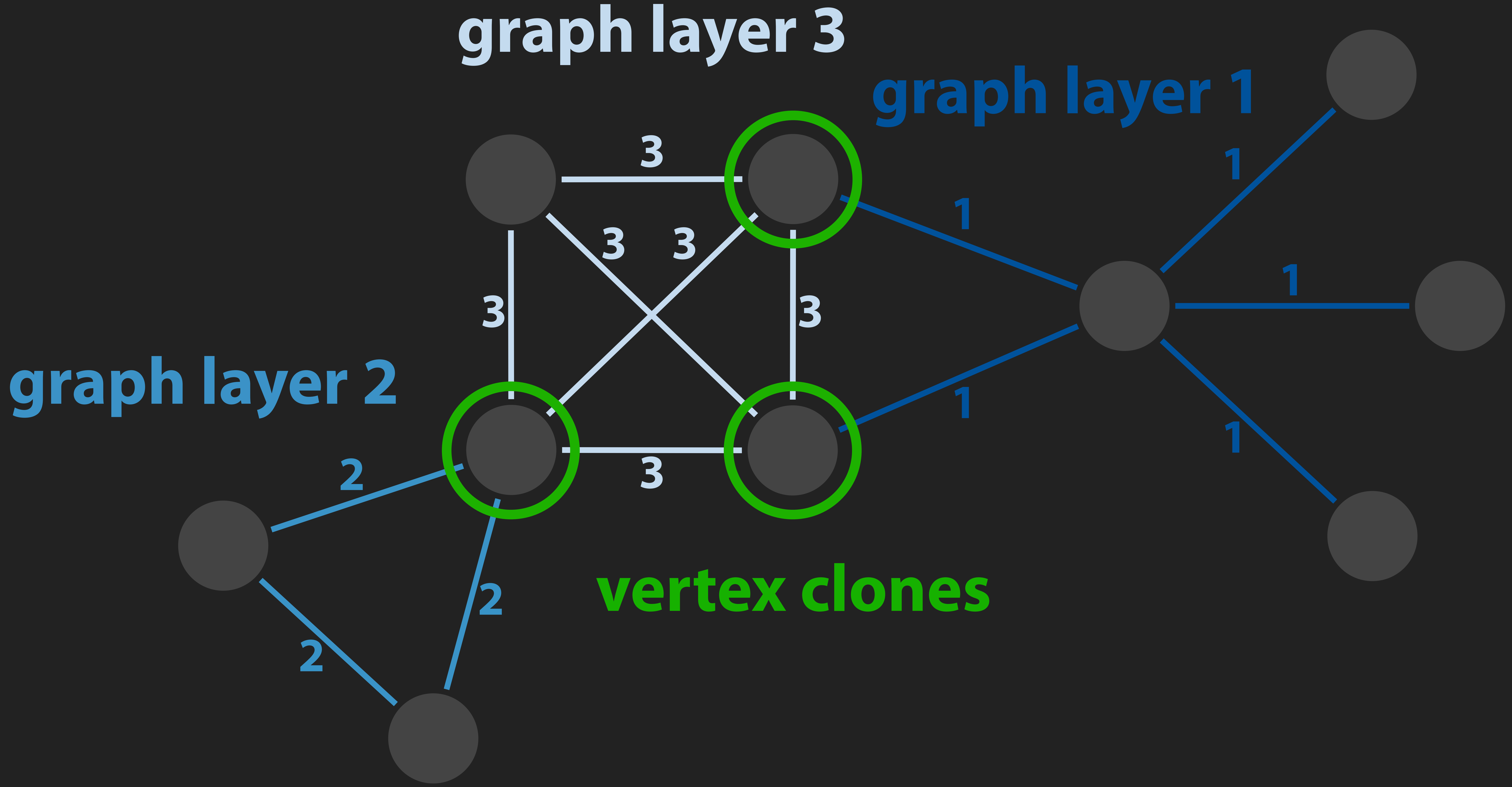


peel = 1




3
3
3
3
3
3
1
1
1
1
1
1
2
2
2


graph layer 3
graph layer 1
graph layer 2
3
3
3
3
3
3
1
1
1
1
1
1
2
2
2

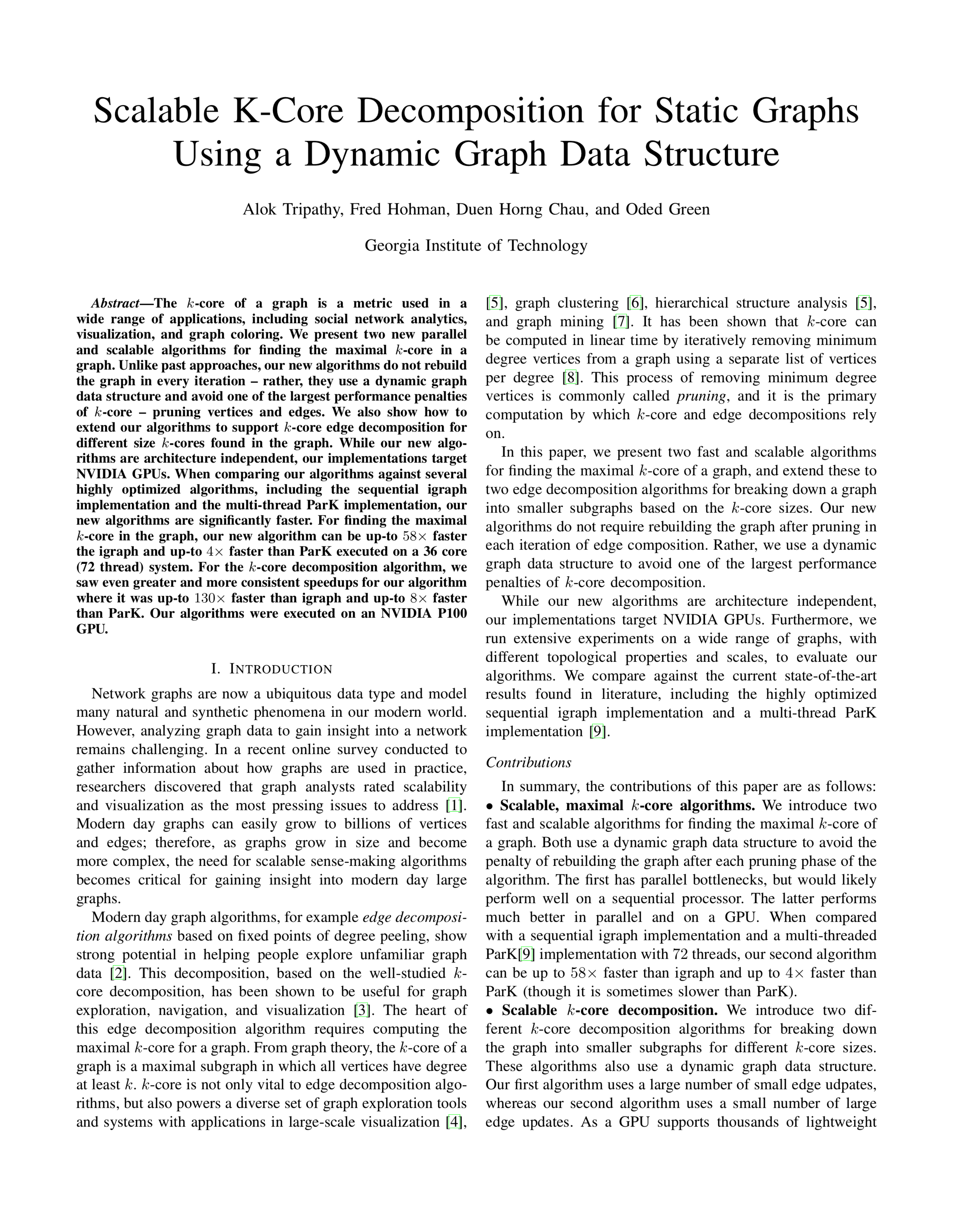

graph layer 3
graph layer 1
graph layer 2
vertex clones
3
3
3
3
3
3
1
1
1
1
1
1
2
2
2

Graph	Vertices	Edges	Time (s)	Layers
Google+	24K	39K	~0	10
arXiv astro-ph	19K	198K	~0	47
Amazon	**335K**	**925K**	**~0**	**6**
US Patents	3.8M	17M	11	41
Wikipedia (German)	3.2M	82M	225	320
Orkut	**3.1M**	**117M**	**92**	**91**

Graph	Vertices	Edges	Time (s)	Layers
Google+	24K	39K	~0	10
arXiv astro-ph	19K	198K	~0	47
Amazon	**335K**	**925K**	**~0**	**6**
US Patents	3.8M	17M	11	41
Wikipedia (German)	3.2M	82M	225	320
Orkut	**3.1M**	**117M**	**92**	**91**

Time complexity: ***O(#edges x #layers)***

layers << edges



Scalable K-Core Decomposition for Static Graphs Using a Dynamic Graph Data Structure

Alok Tripathy, Fred Hohman, Duen Horng Chau, and Oded Green

Georgia Institute of Technology

***Abstract*—The k-core of a graph is a metric used in a wide range of applications, including social network analytics, visualization, and graph coloring. We present two new parallel and scalable algorithms for finding the maximal k-core in a graph. Unlike past approaches, our new algorithms do not rebuild the graph in every iteration – rather, they use a dynamic graph data structure and avoid one of the largest performance penalties of k-core – pruning vertices and edges. We also show how to extend our algorithms to support k-core edge decomposition for different size k-cores found in the graph. While our new algorithms are architecture independent, our implementations target NVIDIA GPUs. When comparing our algorithms against several highly optimized algorithms, including the sequential igraph implementation and the multi-thread ParK implementation, our new algorithms are significantly faster. For finding the maximal k-core in the graph, our new algorithm can be up-to $58\times$ faster the igraph and up-to $4\times$ faster than ParK executed on a 36 core (72 thread) system. For the k-core decomposition algorithm, we saw even greater and more consistent speedups for our algorithm where it was up-to $130\times$ faster than igraph and up-to $8\times$ faster than ParK. Our algorithms were executed on an NVIDIA P100 GPU.**

I. Introduction

Network graphs are now a ubiquitous data type and model many natural and synthetic phenomena in our modern world. However, analyzing graph data to gain insight into a network remains challenging. In a recent online survey conducted to gather information about how graphs are used in practice, researchers discovered that graph analysts rated scalability and visualization as the most pressing issues to address [1]. Modern day graphs can easily grow to billions of vertices and edges; therefore, as graphs grow in size and become more complex, the need for scalable sense-making algorithms becomes critical for gaining insight into modern day large graphs.

Modern day graph algorithms, for example *edge decomposition algorithms* based on fixed points of degree peeling, show strong potential in helping people explore unfamiliar graph data [2]. This decomposition, based on the well-studied k-core decomposition, has been shown to be useful for graph exploration, navigation, and visualization [3]. The heart of this edge decomposition algorithm requires computing the maximal k-core for a graph. From graph theory, the k-core of a graph is a maximal subgraph in which all vertices have degree at least k. k-core is not only vital to edge decomposition algorithms, but also powers a diverse set of graph exploration tools and systems with applications in large-scale visualization [4], [5], graph clustering [6], hierarchical structure analysis [5], and graph mining [7]. It has been shown that k-core can be computed in linear time by iteratively removing minimum degree vertices from a graph using a separate list of vertices per degree [8]. This process of removing minimum degree vertices is commonly called *pruning*, and it is the primary computation by which k-core and edge decompositions rely on.

In this paper, we present two fast and scalable algorithms for finding the maximal k-core of a graph, and extend these to two edge decomposition algorithms for breaking down a graph into smaller subgraphs based on the k-core sizes. Our new algorithms do not require rebuilding the graph after pruning in each iteration of edge composition. Rather, we use a dynamic graph data structure to avoid one of the largest performance penalties of k-core decomposition.

While our new algorithms are architecture independent, our implementations target NVIDIA GPUs. Furthermore, we run extensive experiments on a wide range of graphs, with different topological properties and scales, to evaluate our algorithms. We compare against the current state-of-the-art results found in literature, including the highly optimized sequential igraph implementation and a multi-thread ParK implementation [9].

Contributions

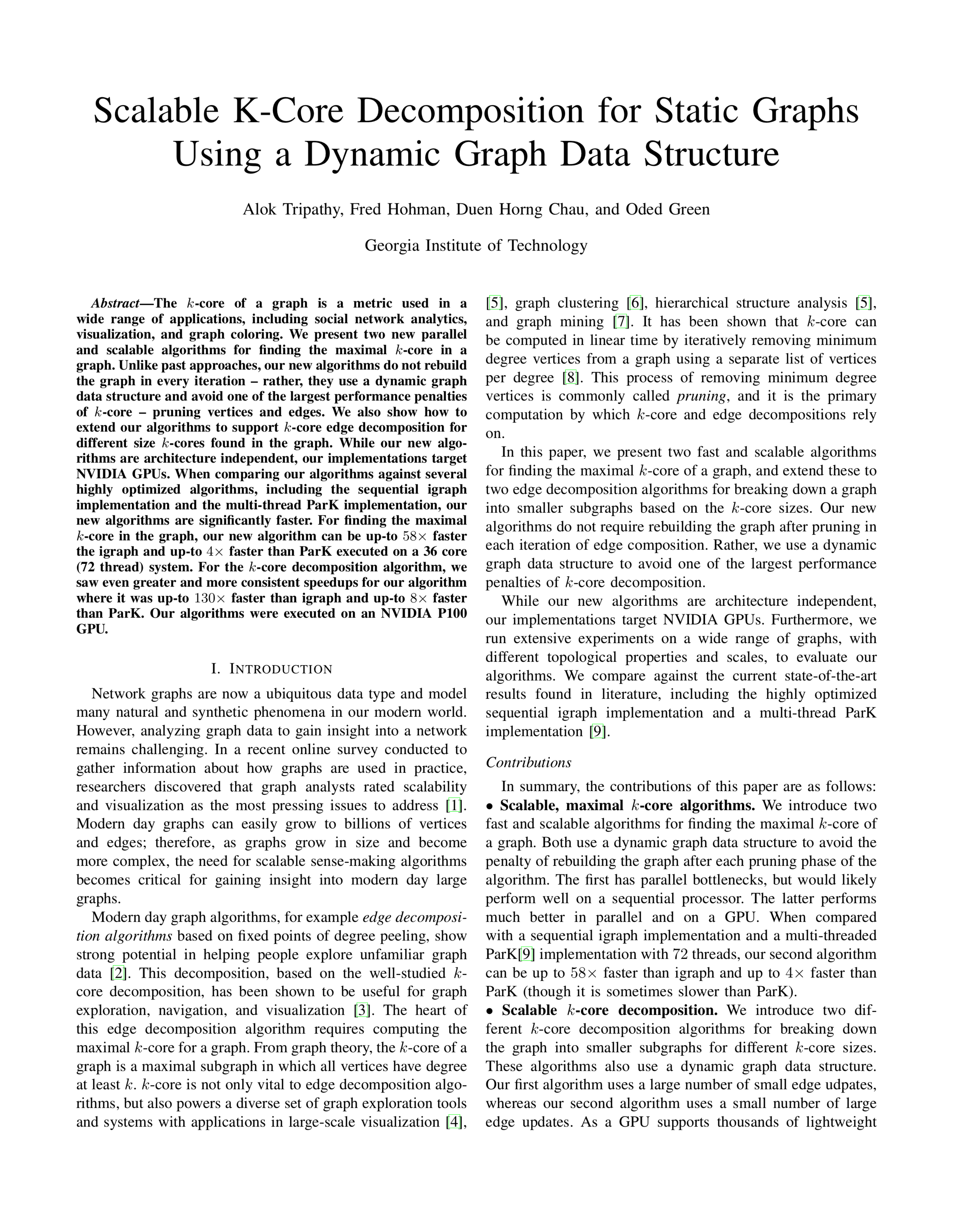
In summary, the contributions of this paper are as follows:

- **Scalable, maximal k-core algorithms.** We introduce two fast and scalable algorithms for finding the maximal k-core of a graph. Both use a dynamic graph data structure to avoid the penalty of rebuilding the graph after each pruning phase of the algorithm. The first has parallel bottlenecks, but would likely perform well on a sequential processor. The latter performs much better in parallel and on a GPU. When compared with a sequential igraph implementation and a multi-threaded ParK[9] implementation with 72 threads, our second algorithm can be up to $58\times$ faster than igraph and up to $4\times$ faster than ParK (though it is sometimes slower than ParK).
- **Scalable k-core decomposition.** We introduce two different k-core decomposition algorithms for breaking down the graph into smaller subgraphs for different k-core sizes. These algorithms also use a dynamic graph data structure. Our first algorithm uses a large number of small edge udpates, whereas our second algorithm uses a small number of large edge updates. As a GPU supports thousands of lightweight

Scalable K-Core Decomposition for Static Graphs Using a Dynamic Graph Data Structure

Alok Tripathy, Fred Hohman, Duen Horng (Polo) Chau, Oded Green

IEEE International Conference on Big Data. Seattle, WA, USA, 2018.



Scalable K-Core Decomposition for Static Graphs Using a Dynamic Graph Data Structure

Alok Tripathy, Fred Hohman, Duen Horng Chau, and Oded Green

Georgia Institute of Technology

***Abstract*—The k-core of a graph is a metric used in a wide range of applications, including social network analytics, visualization, and graph coloring. We present two new parallel and scalable algorithms for finding the maximal k-core in a graph. Unlike past approaches, our new algorithms do not rebuild the graph in every iteration – rather, they use a dynamic graph data structure and avoid one of the largest performance penalties of k-core – pruning vertices and edges. We also show how to extend our algorithms to support k-core edge decomposition for different size k-cores found in the graph. While our new algorithms are architecture independent, our implementations target NVIDIA GPUs. When comparing our algorithms against several highly optimized algorithms, including the sequential igraph implementation and the multi-thread ParK implementation, our new algorithms are significantly faster. For finding the maximal k-core in the graph, our new algorithm can be up-to $58\times$ faster the igraph and up-to $4\times$ faster than ParK executed on a 36 core (72 thread) system. For the k-core decomposition algorithm, we saw even greater and more consistent speedups for our algorithm where it was up-to $130\times$ faster than igraph and up-to $8\times$ faster than ParK. Our algorithms were executed on an NVIDIA P100 GPU.**

I. Introduction

Network graphs are now a ubiquitous data type and model many natural and synthetic phenomena in our modern world. However, analyzing graph data to gain insight into a network remains challenging. In a recent online survey conducted to gather information about how graphs are used in practice, researchers discovered that graph analysts rated scalability and visualization as the most pressing issues to address [1]. Modern day graphs can easily grow to billions of vertices and edges; therefore, as graphs grow in size and become more complex, the need for scalable sense-making algorithms becomes critical for gaining insight into modern day large graphs.

Modern day graph algorithms, for example *edge decomposition algorithms* based on fixed points of degree peeling, show strong potential in helping people explore unfamiliar graph data [2]. This decomposition, based on the well-studied k-core decomposition, has been shown to be useful for graph exploration, navigation, and visualization [3]. The heart of this edge decomposition algorithm requires computing the maximal k-core for a graph. From graph theory, the k-core of a graph is a maximal subgraph in which all vertices have degree at least k. k-core is not only vital to edge decomposition algorithms, but also powers a diverse set of graph exploration tools and systems with applications in large-scale visualization [4], [5], graph clustering [6], hierarchical structure analysis [5], and graph mining [7]. It has been shown that k-core can be computed in linear time by iteratively removing minimum degree vertices from a graph using a separate list of vertices per degree [8]. This process of removing minimum degree vertices is commonly called *pruning*, and it is the primary computation by which k-core and edge decompositions rely on.

In this paper, we present two fast and scalable algorithms for finding the maximal k-core of a graph, and extend these to two edge decomposition algorithms for breaking down a graph into smaller subgraphs based on the k-core sizes. Our new algorithms do not require rebuilding the graph after pruning in each iteration of edge composition. Rather, we use a dynamic graph data structure to avoid one of the largest performance penalties of k-core decomposition.

While our new algorithms are architecture independent, our implementations target NVIDIA GPUs. Furthermore, we run extensive experiments on a wide range of graphs, with different topological properties and scales, to evaluate our algorithms. We compare against the current state-of-the-art results found in literature, including the highly optimized sequential igraph implementation and a multi-thread ParK implementation [9].

Contributions

In summary, the contributions of this paper are as follows:

- **Scalable, maximal k-core algorithms.** We introduce two fast and scalable algorithms for finding the maximal k-core of a graph. Both use a dynamic graph data structure to avoid the penalty of rebuilding the graph after each pruning phase of the algorithm. The first has parallel bottlenecks, but would likely perform well on a sequential processor. The latter performs much better in parallel and on a GPU. When compared with a sequential igraph implementation and a multi-threaded ParK[9] implementation with 72 threads, our second algorithm can be up to $58\times$ faster than igraph and up to $4\times$ faster than ParK (though it is sometimes slower than ParK).
- **Scalable k-core decomposition.** We introduce two different k-core decomposition algorithms for breaking down the graph into smaller subgraphs for different k-core sizes. These algorithms also use a dynamic graph data structure. Our first algorithm uses a large number of small edge udpates, whereas our second algorithm uses a small number of large edge updates. As a GPU supports thousands of lightweight

Scalable K-Core Decomposition for Static Graphs Using a Dynamic Graph Data Structure

Alok Tripathy, Fred Hohman, Duen Horng (Polo) Chau, Oded Green

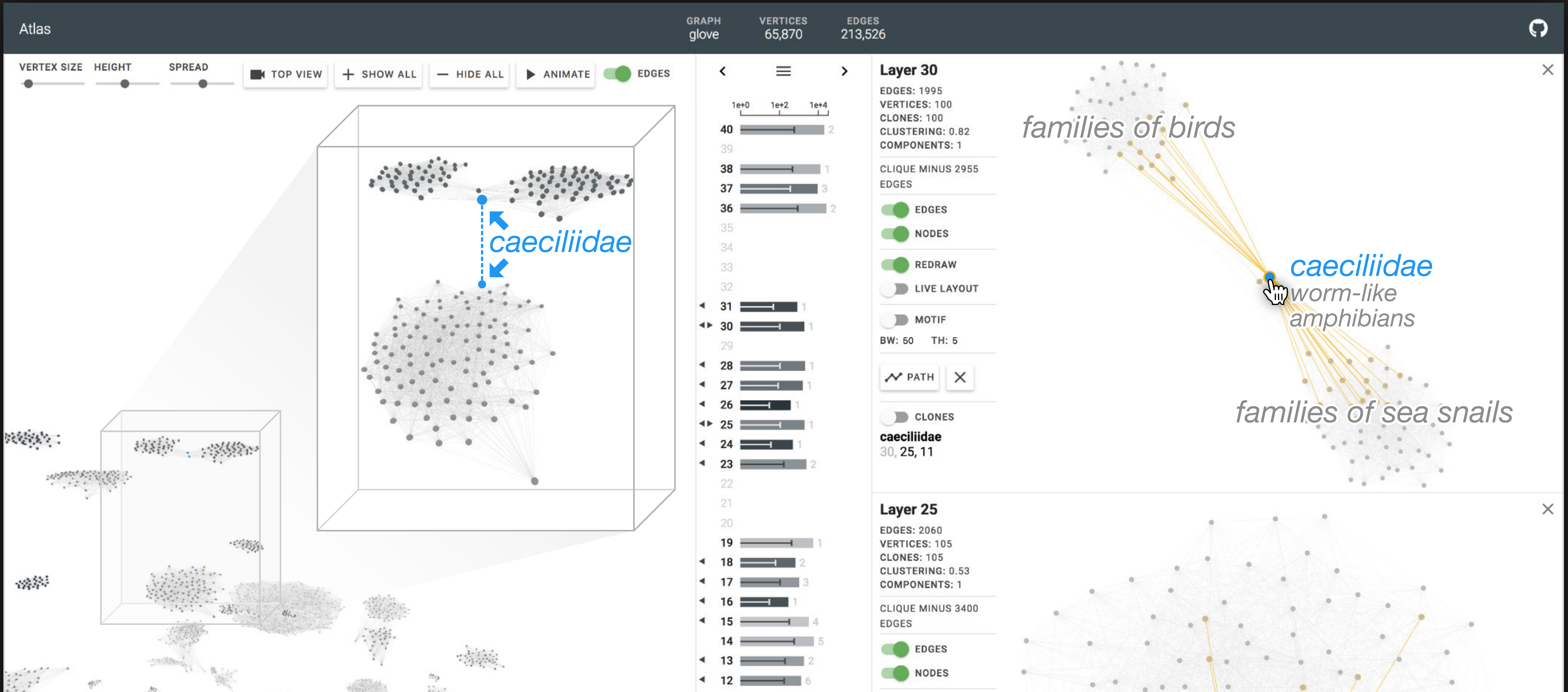
IEEE International Conference on Big Data. Seattle, WA, USA, 2018.

GPU + dynamic graph data structure
-> **4x - 8x** speed up over ParK

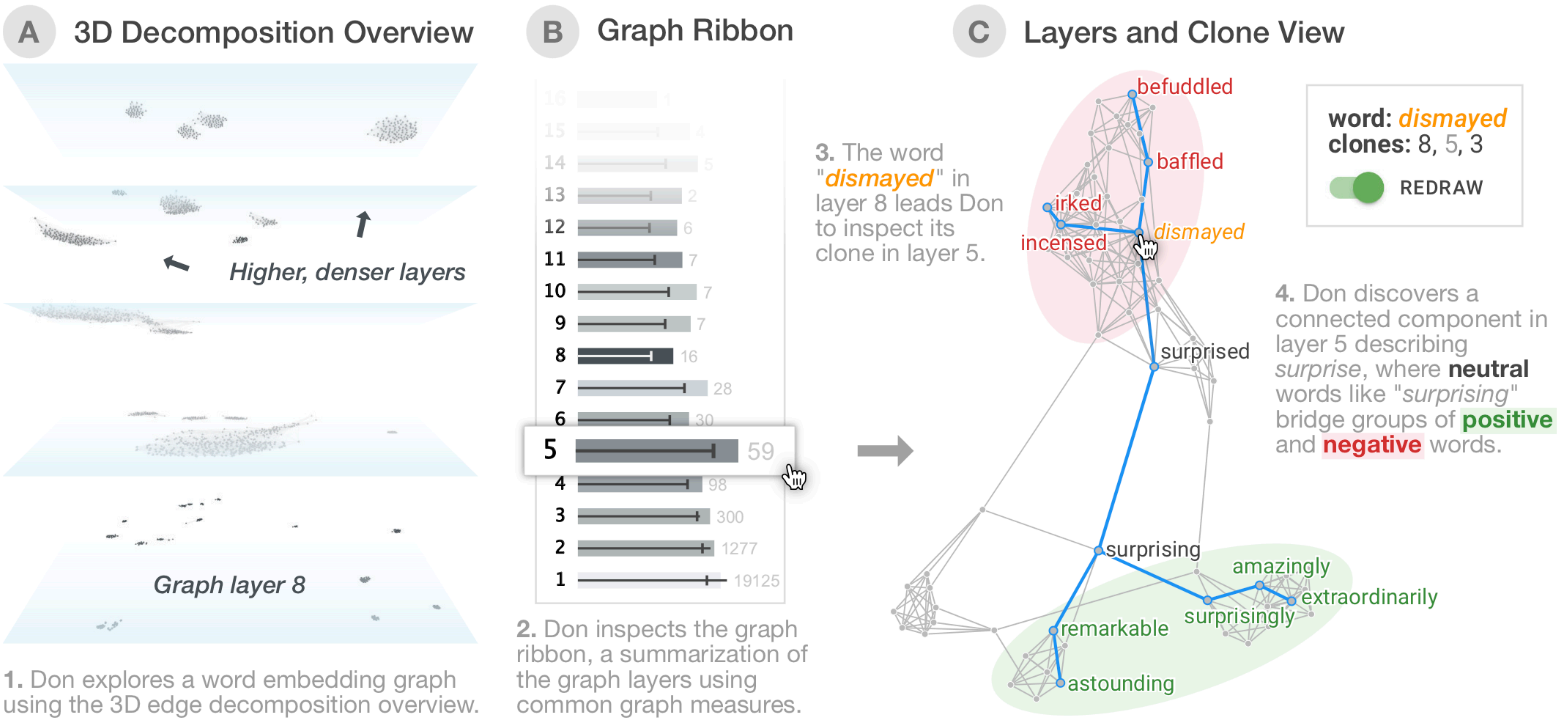
Demo: Understanding Word Embedding Graph

Nodes: 66K words from Wikipedia

Edges: 214K (connect words with small distance)



ATLAS contains three main views


A
3D Decomposition Overview
Higher, denser layers
Graph layer 8
1. Don explores a word embedding graph using the 3D edge decomposition overview.
B
Graph Ribbon
13
12
11
10
9
8
7
6
5
4
3
2
1
2
6
7
7
7
16
28
30
59
98
300
1277
19125
2. Don inspects the graph ribbon, a summarization of the graph layers using common graph measures.
3. The word "dismayed" in layer 8 leads Don to inspect its clone in layer 5.
C
Layers and Clone View
befuddled
baffled
irked
incensed
dismayed
surprised
surprising
amazingly
extraordinarily
surprisingly
remarkable
astounding
word: dismayed
clones: 8, 5, 3
REDRAW
4. Don discovers a connected component in layer 5 describing surprise, where neutral words like "surprising" bridge groups of positive and negative words.

User Study

*Goal: use **Atlas** to spot interesting patterns, mimicking their own work*

Graph Analysts	**Graphs**
Researcher, Symantec	Yelp Reviews Network
Researcher, NASA	SEC Insider Trading Graph
Systems engineer, NASA	GloVe Word Embed. Graph

All PhDs + use graphs daily or weekly

Intro questionnaire → **Atlas** tutorial → Study → Exit questionnaire

User Study Findings

User Study Findings

3D for overview, 2D for details

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a “feel” for the graph

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a “feel” for the graph
- Graph Ribbon + Layers view used more precisely

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a "feel" for the graph
- Graph Ribbon + Layers view used more precisely
- Show nearest neighbors used frequently

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a "feel" for the graph
- Graph Ribbon + Layers view used more precisely
- Show nearest neighbors used frequently

Identifying and linking meaningful graph substructures

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a “feel” for the graph
- Graph Ribbon + Layers view used more precisely
- Show nearest neighbors used frequently

Identifying and linking meaningful graph substructures

- Vertex clones as traversal mechanism between layers

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a “feel” for the graph
- Graph Ribbon + Layers view used more precisely
- Show nearest neighbors used frequently

Identifying and linking meaningful graph substructures

- Vertex clones as traversal mechanism between layers

Application to anomaly detection

User Study Findings

3D for overview, 2D for details

- 3D useful for intro to new data → get a "feel" for the graph
- Graph Ribbon + Layers view used more precisely
- Show nearest neighbors used frequently

Identifying and linking meaningful graph substructures

- Vertex clones as traversal mechanism between layers

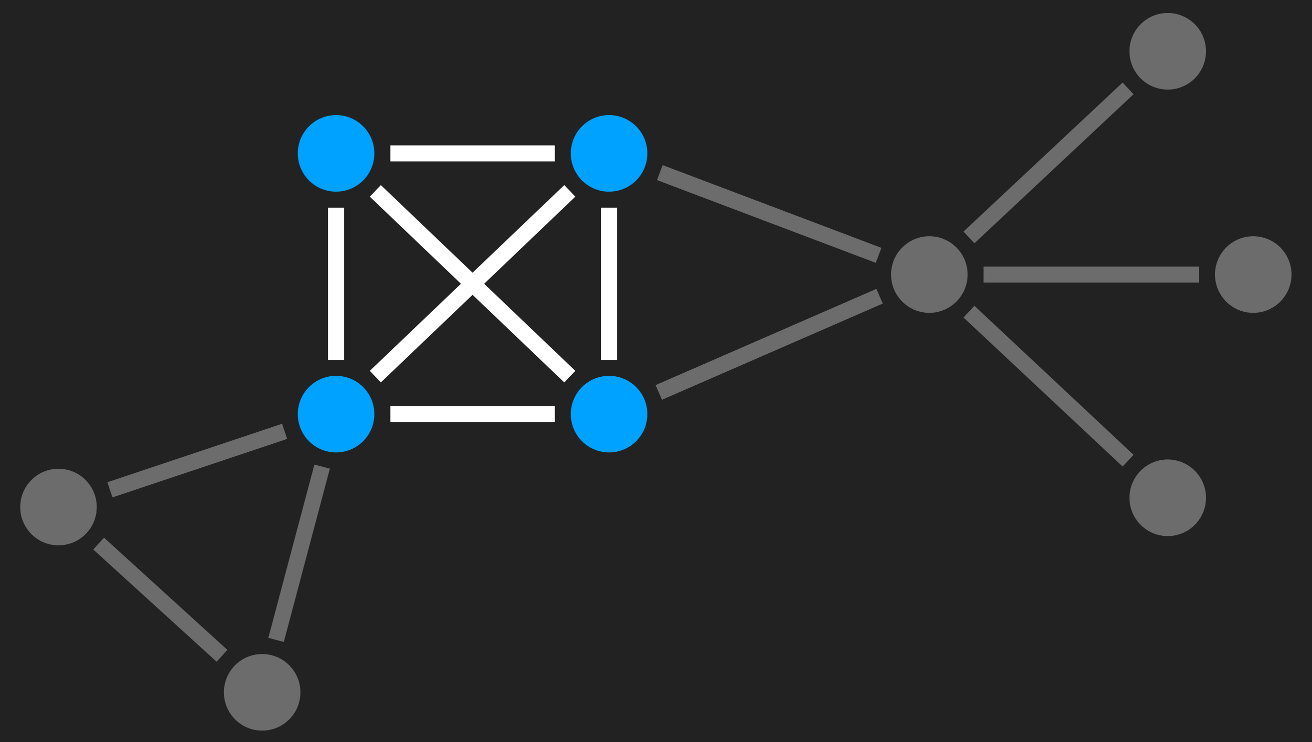
Application to anomaly detection

- *"…analysis (using [both] vertex clones and layers) naturally reveals potentially anomalous substructures and vertices. This is highly useful from a cybersecurity perspective."*

Future Work

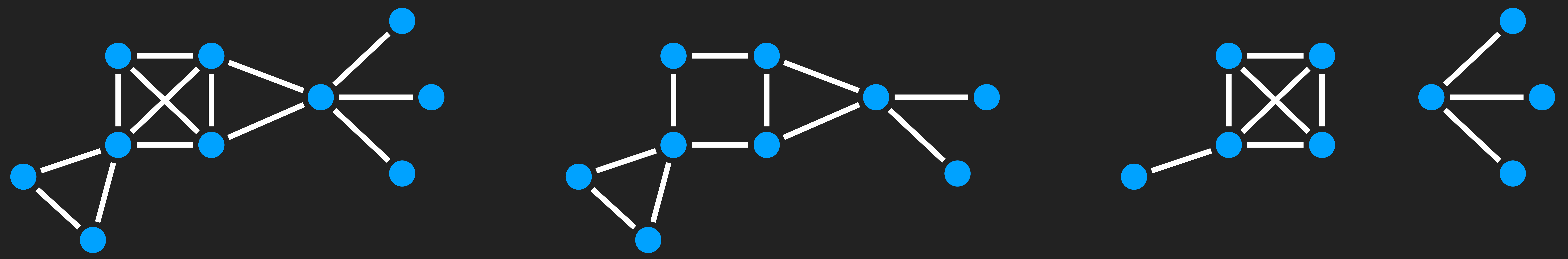
Future Work

- Automatically suggest interesting layers



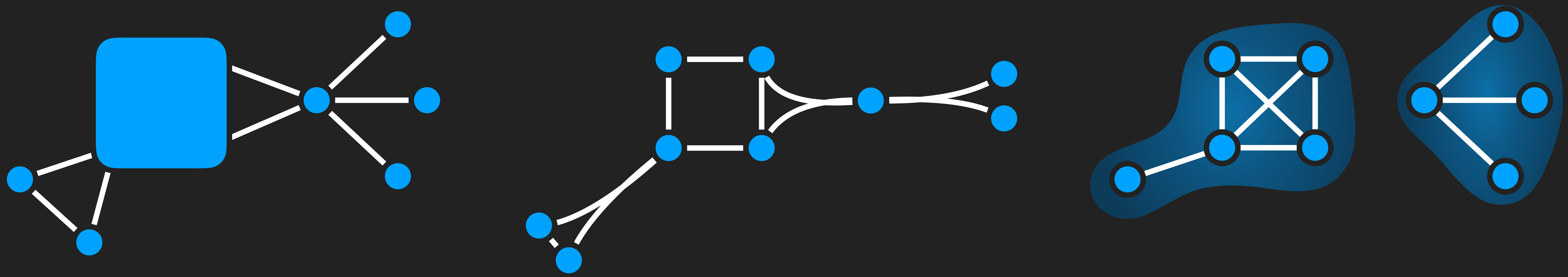
Future Work

- Automatically suggest interesting layers
- Dynamic graph decomposition visualization



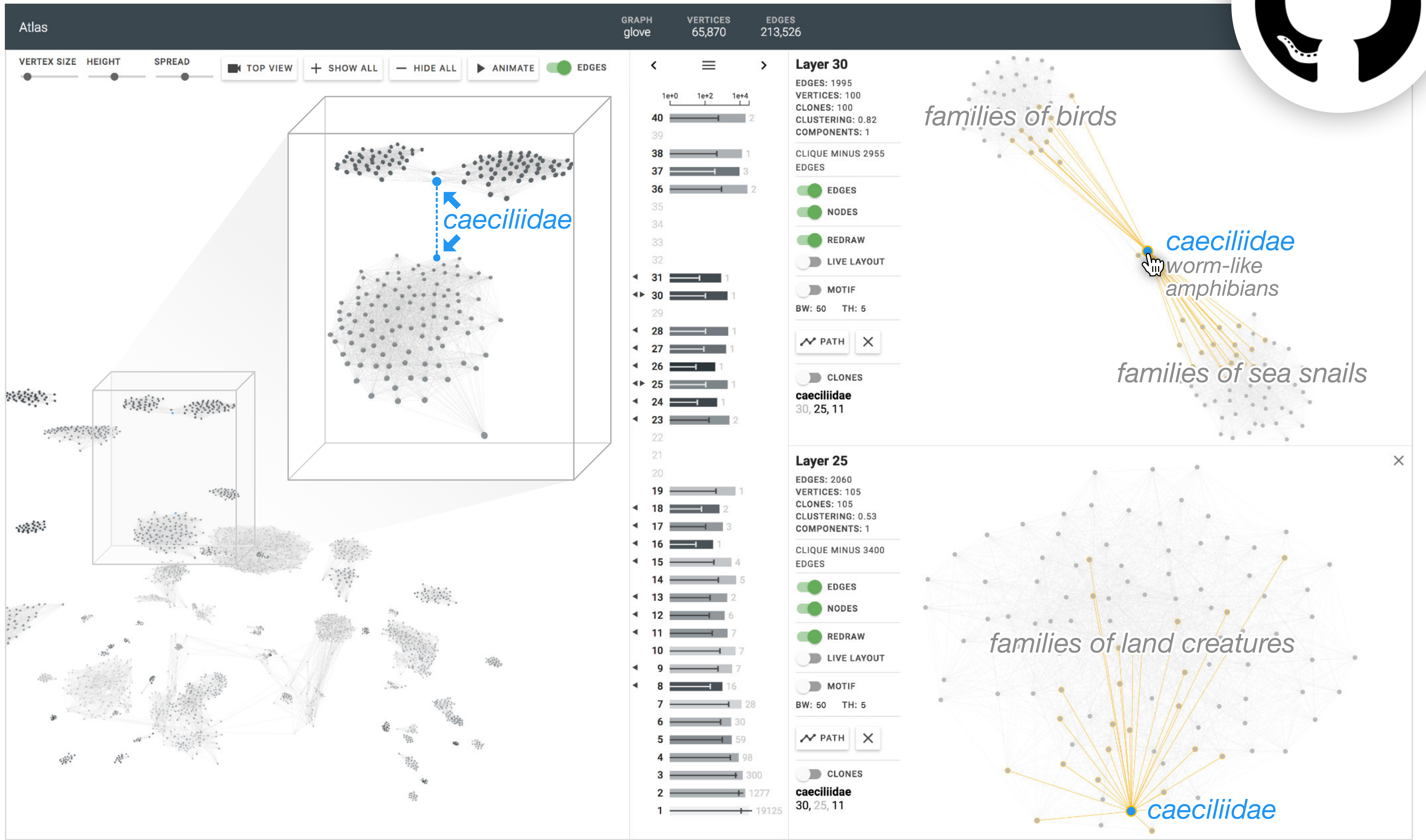
Future Work

- Automatically suggest interesting layers
- Dynamic graph decomposition visualization
- Visual scalability (e.g., super-noding, edge bundling, graph motif)



Atlas Local Graph Exploration in a Global Context

bit.ly/**atlas-iui**



Thanks!

Fred Hohman
@fredhohman
fredhohman@gatech.edu

James Abello
abelloj@cs.rutgers.edu

Varun Bezzam
varun.bezzam@gatech.edu

Polo Chau
polo@gatech.edu

We thank the anonymous reviewers for their constructive feedback.