Densest subgraphs, iterative peeling, and supermodularity

November 20, 2024

Rebecca Kempe

COMP 5112 Project Presentation

Outline

1. Densest subgraph problem (**DSP**): motivation and definitions
2. Peeling, iterative peeling for graphs (**Greedy++**); associated guarantees
3. Densest supermodular set problem (**DSSP**)
   i. Iterative peeling for DSSP (**SuperGreedy++**)
   ii. **Convergence** of iterative peeling for DSSP
4. References

This presentation is primarily based on the work of Boob et. al (2019)

Flowless: Extracting Densest Subgraphs Without Flow Computations

and the work of Chekuri, Quanrud, Torres (2022)

Densest Subgraph: Supermodularity, Iterative Peeling, and Flow

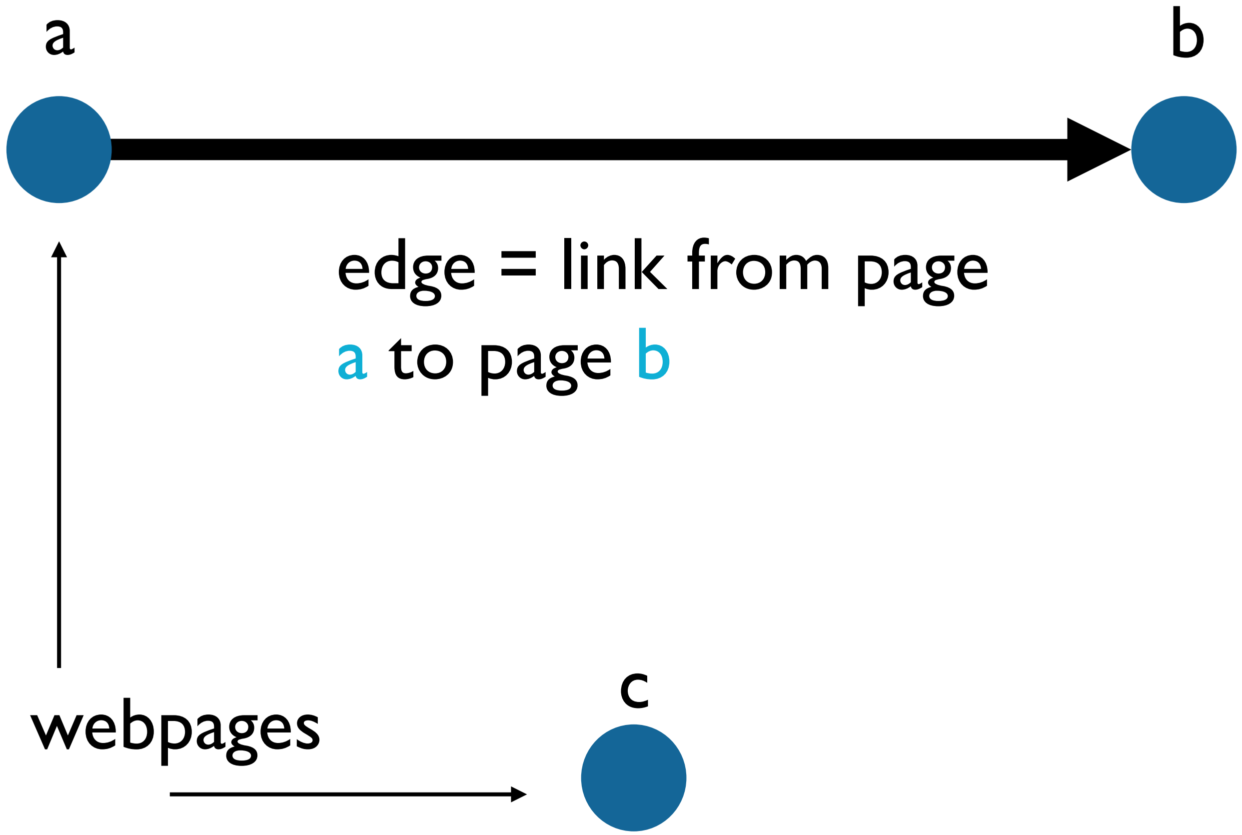
Part 1:

The Densest Subgraph Problem (DSP)

(motivation and definitions)

Many real-world problems can be formulated as finding “clusters” in graphs or optimizing “density measures” on graphs.

community detection



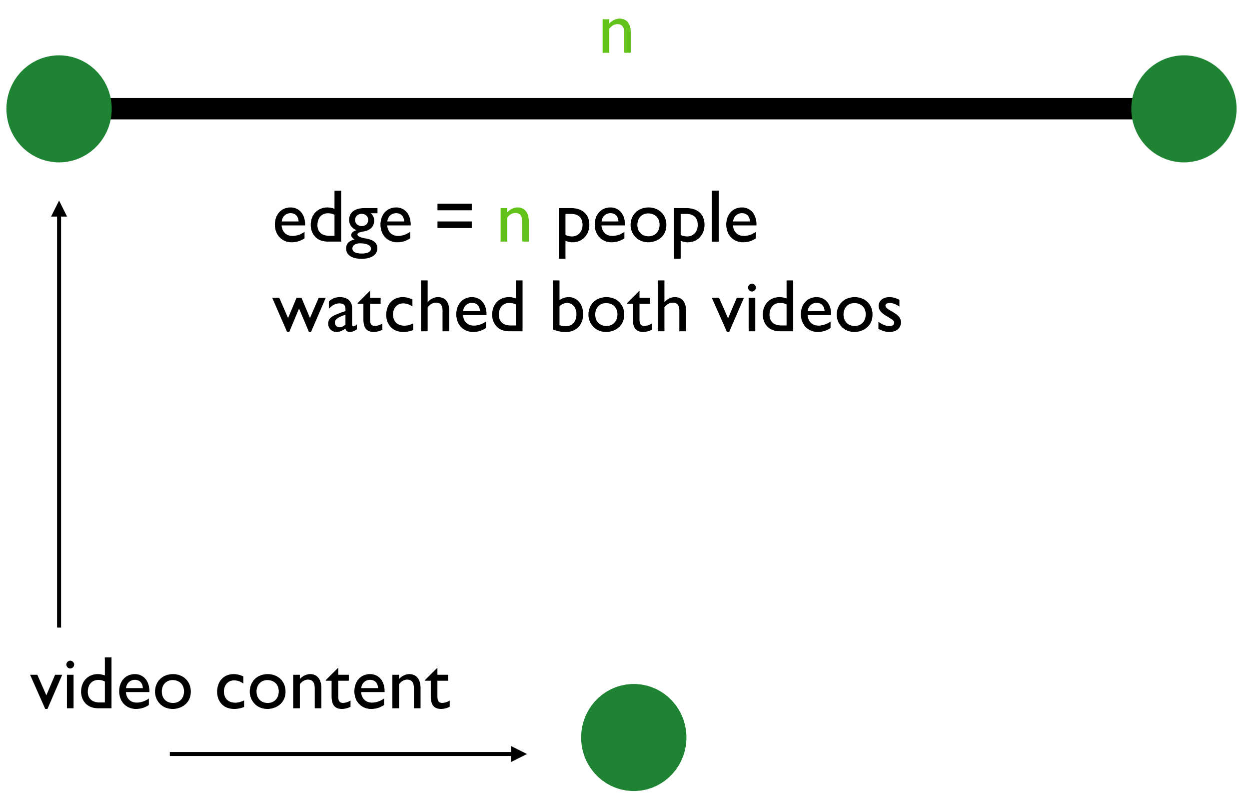
(Kleinberg, 1999)

community detection

- how do we detect groups of web pages that are related to each other?
- how do we find the "authoritative" web pages?

(Kleinberg, 1999)

topic clustering



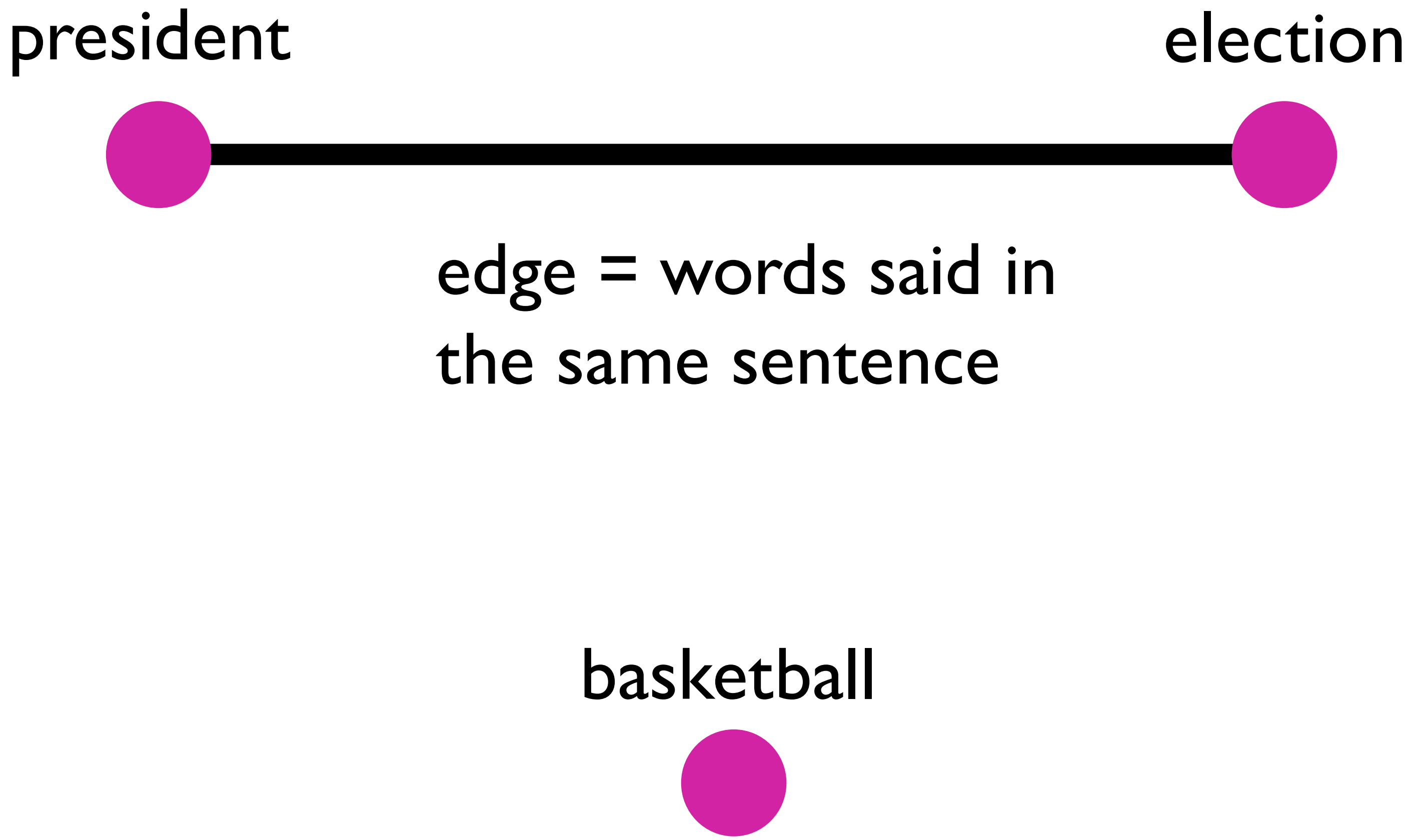
(Tsourakakis, Chen, SDM 2021)

topic clustering

- how do we decide which content is similar?
- which groups of related content are the most popular?
- “large near-clique extraction” based on “thematic coherence”

(Tsourakakis, Chen, SDM 2021)

“text networks”



(Chen, Saad, 2012)
(Corman et. al, 2002)

“text networks”

- which groups of words appear most frequently together?
- what is the topic of the text?
- used in linguistics
- “centering resonance analysis”

(Chen, Saad, 2012)
(Corman et. al, 2002)

This has motivated the study of associated techniques:

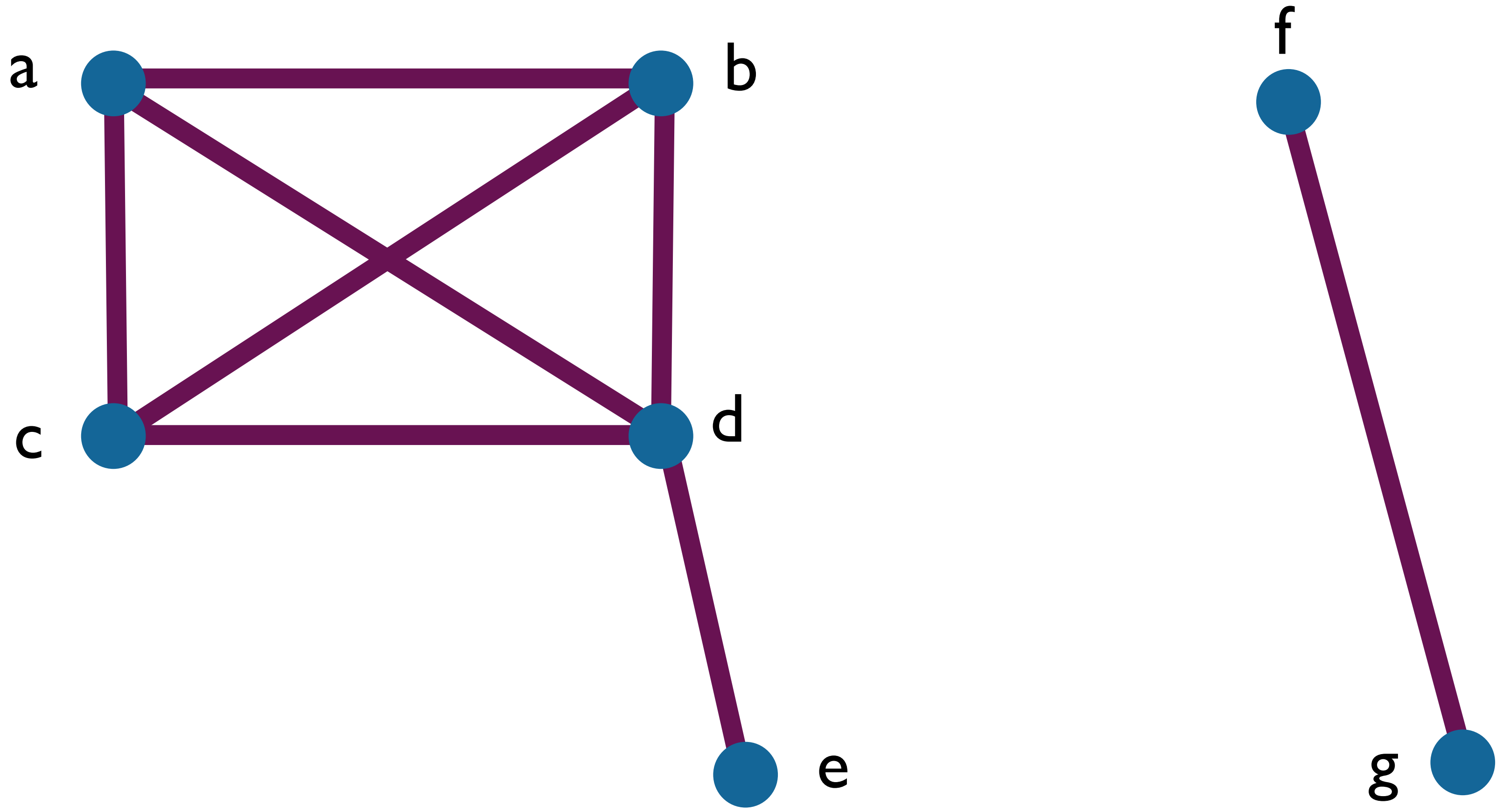
- correlation mining (finance, neuroscience, genetics, etc.)
- graph clustering, graph compression
- "dense subgraph discovery" (there are multiple variants)

In the broadest sense, the DSP asks us to find the subgraph that maximizes some measure of density.

- input: a graph G, a density function f
- output: the subgraph of G with the highest density under f

Formally, we often find the subset of the vertices that induces the densest subgraph.

graph theory (definitions)

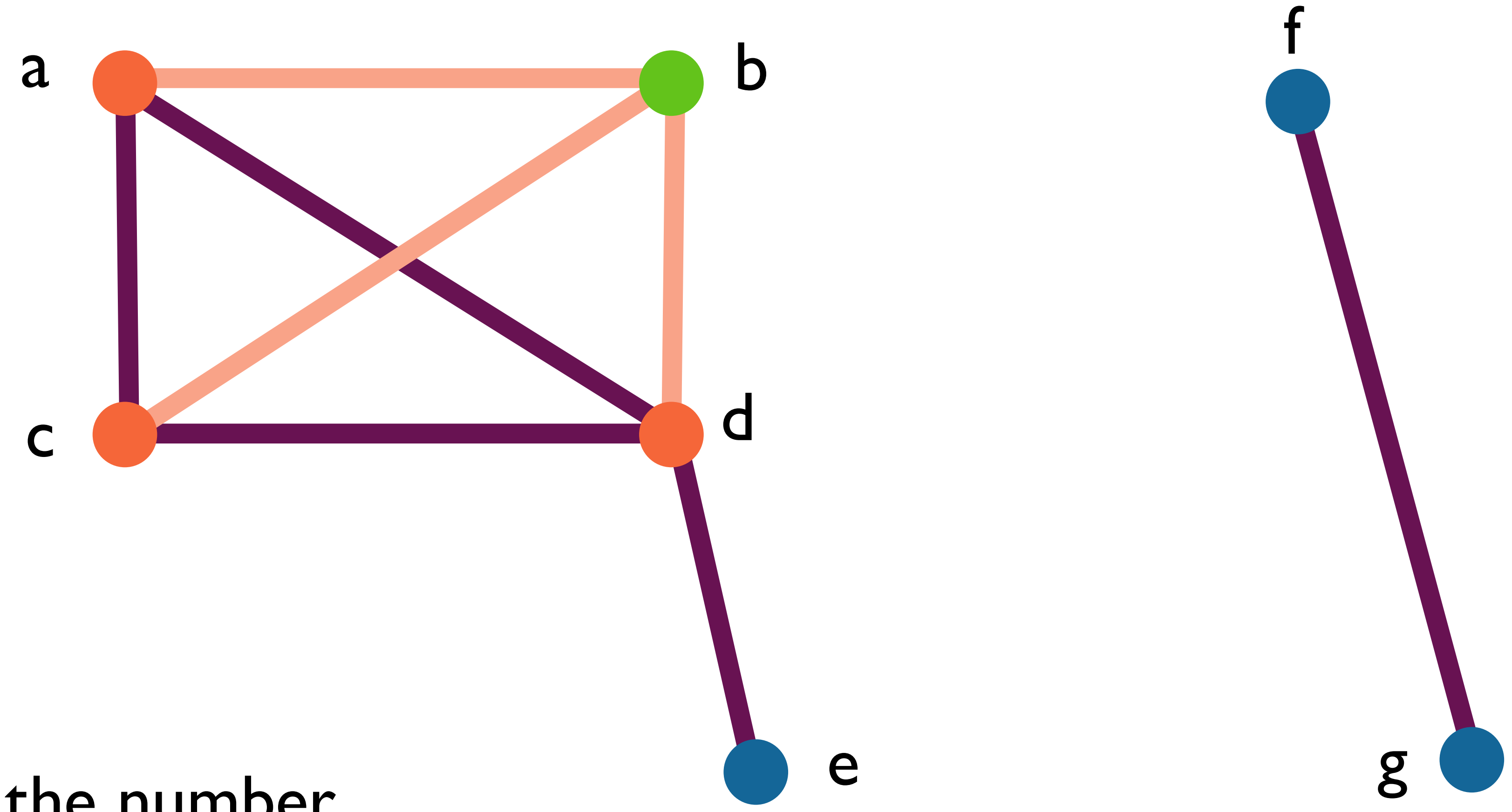


G = (V, E)

order = |V| = n = 7

size = |E| = m = 8

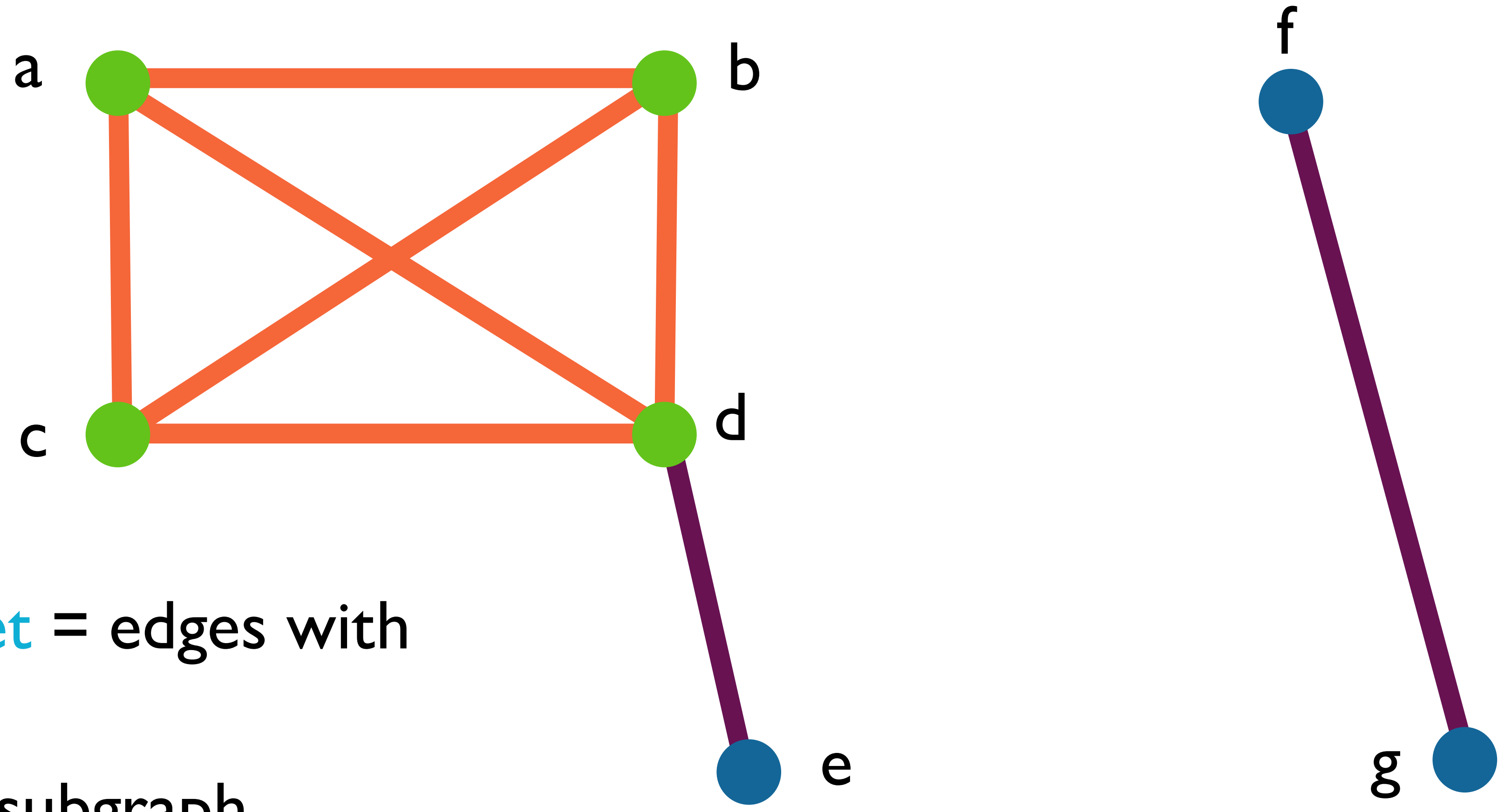
graph theory (definitions)



deg(b) = |nbr(b)| = 3

alternatively, deg(b) is the number of edges connected to b

graph theory (definitions)



S = {a, b, c, d}

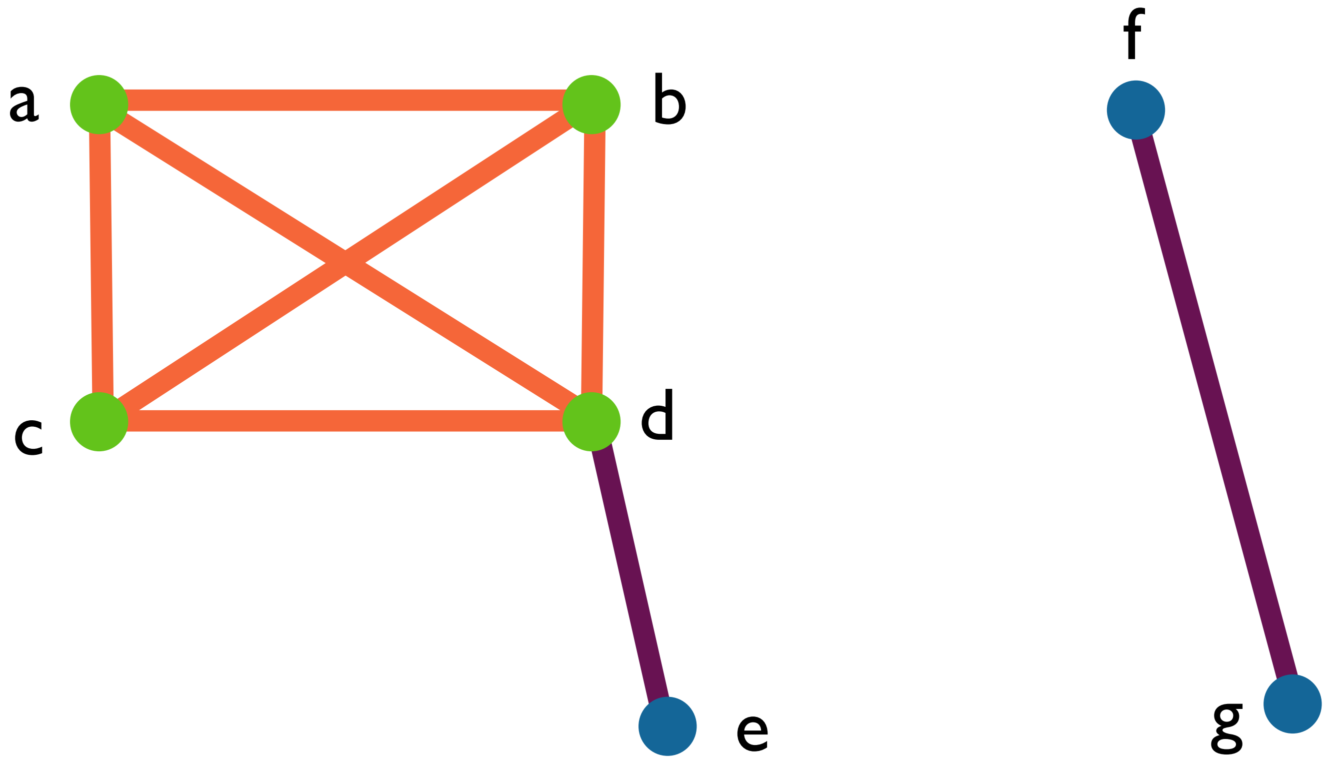
E(S) = induced edge set = edges with both endpoints in S

G[S] = (S, E(S)) is the subgraph induced by S

the densest subgraph problem

The density of a graph is the number of edges it has divided by the number of vertices.

We want to find the subgraph that induces the highest density (largest average degree).



Here, the densest subgraph is induced by the set S = {a, b, c, d}.

The DSP, formally.

Given a graph $G = (V, E)$, let $S \subseteq V$. Then $G[S] = (S, E(S))$ and

$$density(S) = \frac{|E(S)|}{|S|}$$

We want to find the subgraph with the optimal density, λ^*:

$$\lambda^* = \max_{S \subseteq V} \frac{|E(S)|}{|S|}$$

Part 2:

Iterative peeling for graphs (Greedy++)

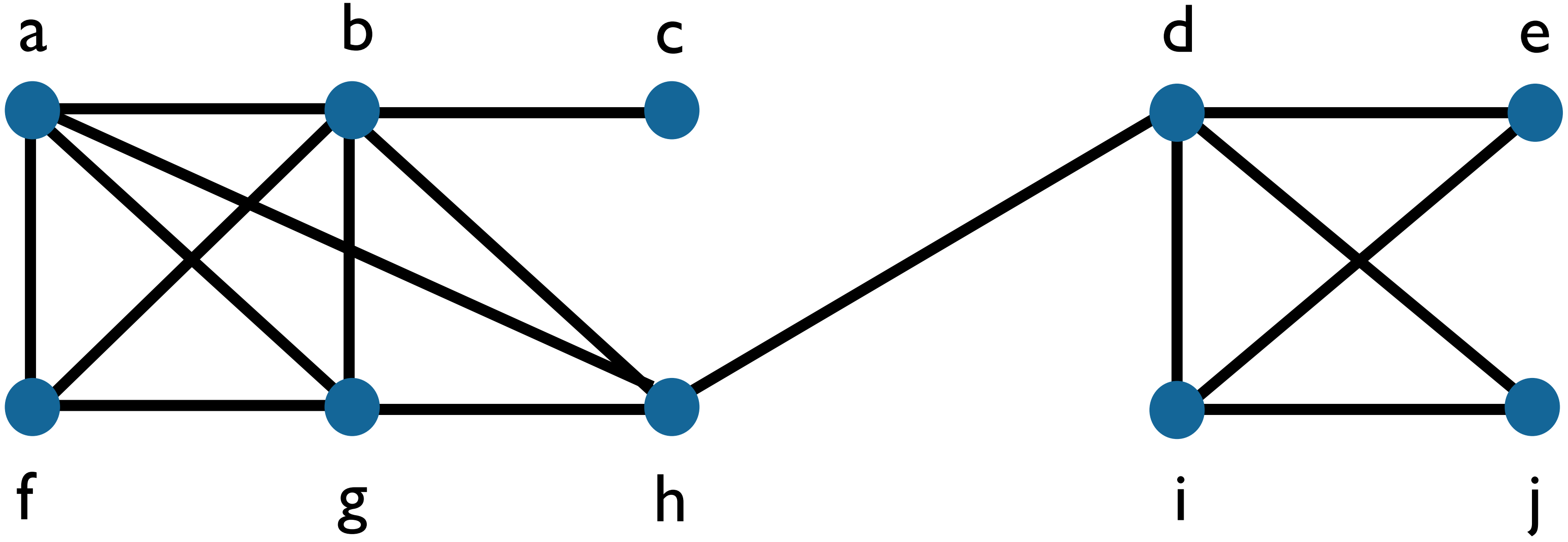
(history, algorithms, and associated results)

peeling: an algorithm

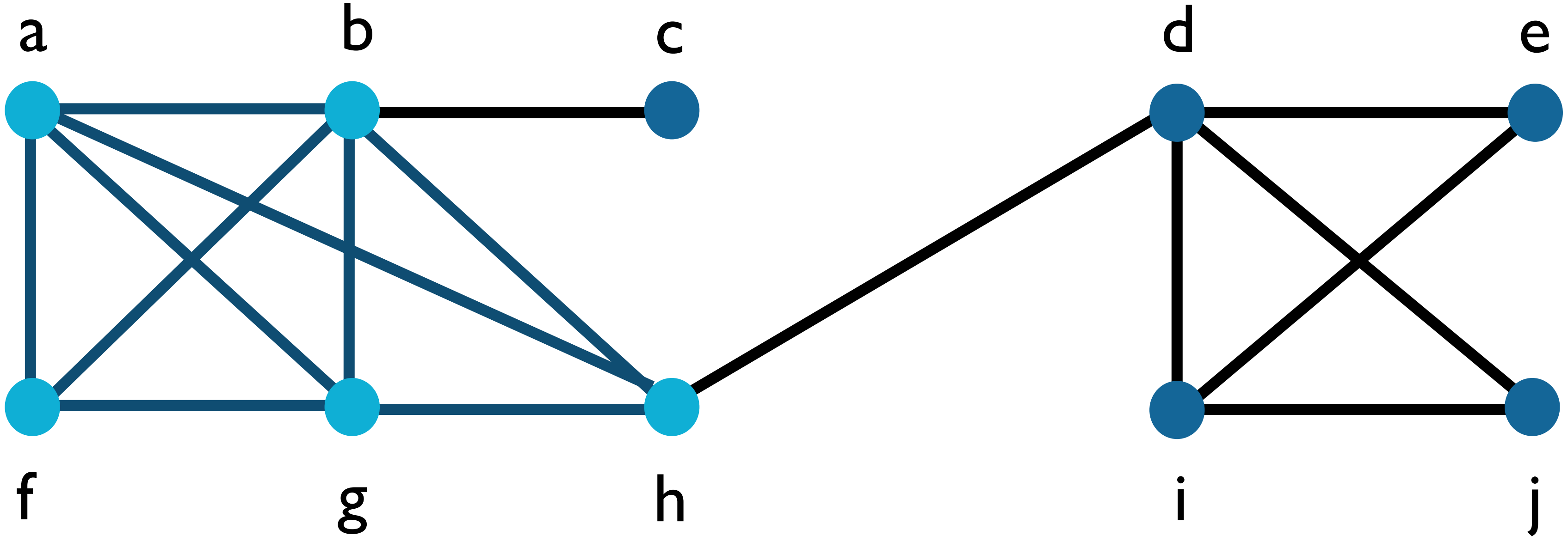
- Given a graph, repeatedly remove the vertex with the current lowest degree, as well as all edges attached to it.
- From this, we get an ordering $v_1, v_2, \ldots, v_n$ of vertices, where v_i is the i_{th} vertex in the removal order.
- We choose the suffix $S_i = \{v_i, v_{i+1}, \ldots, v_n\}$ that induces the subgraph with the highest density λ.

(Asahiro et. al, 1996; Charikar, 2000)

peeling: an algorithm



peeling: an algorithm



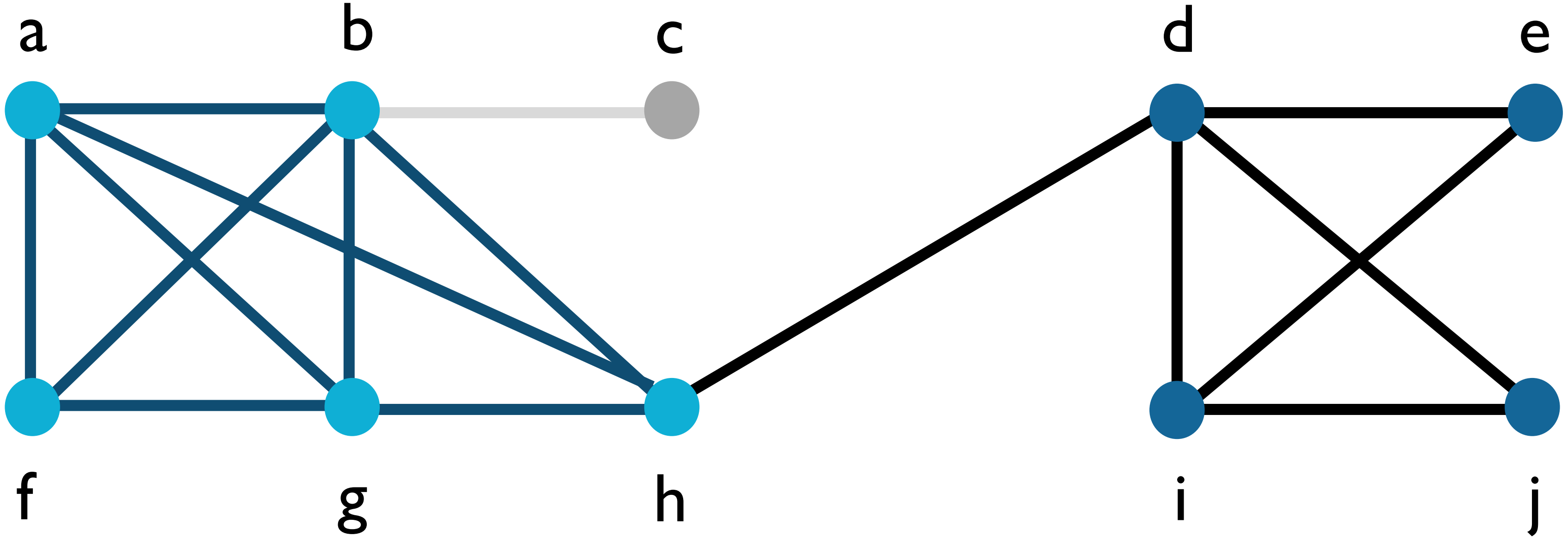
optimal density:
9/5 = 1.8

current density:
16/10 = 1.6

highest density:
16/10 = 1.6

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	5	1	4	2	3	4	3	3	2	curr. degree
										final degree

peeling: an algorithm



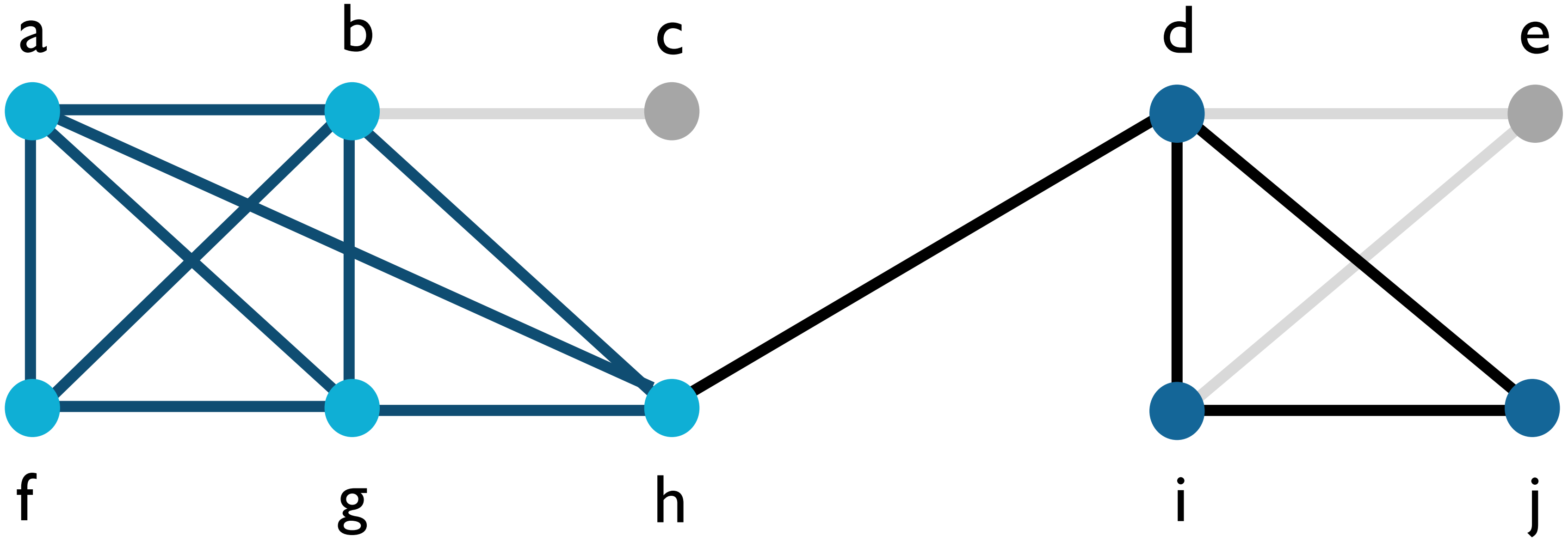
optimal density:
9/5 = 1.8

current density:
15/9 = 1.6666667

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	4	2	3	4	3	3	2	curr. degree
		1								final degree

peeling: an algorithm



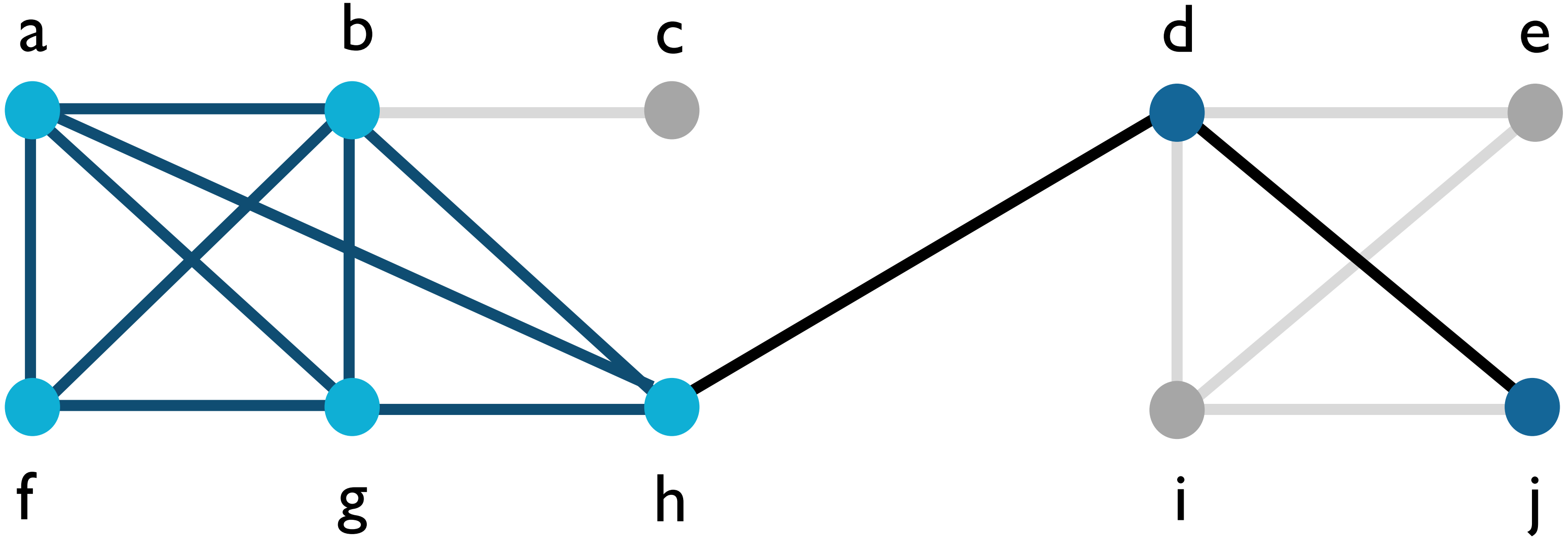
optimal density:
9/5 = 1.8

current density:
13/8 = 1.625

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	3	X	3	4	3	2	2	curr. degree
		1		2						final degree

peeling: an algorithm



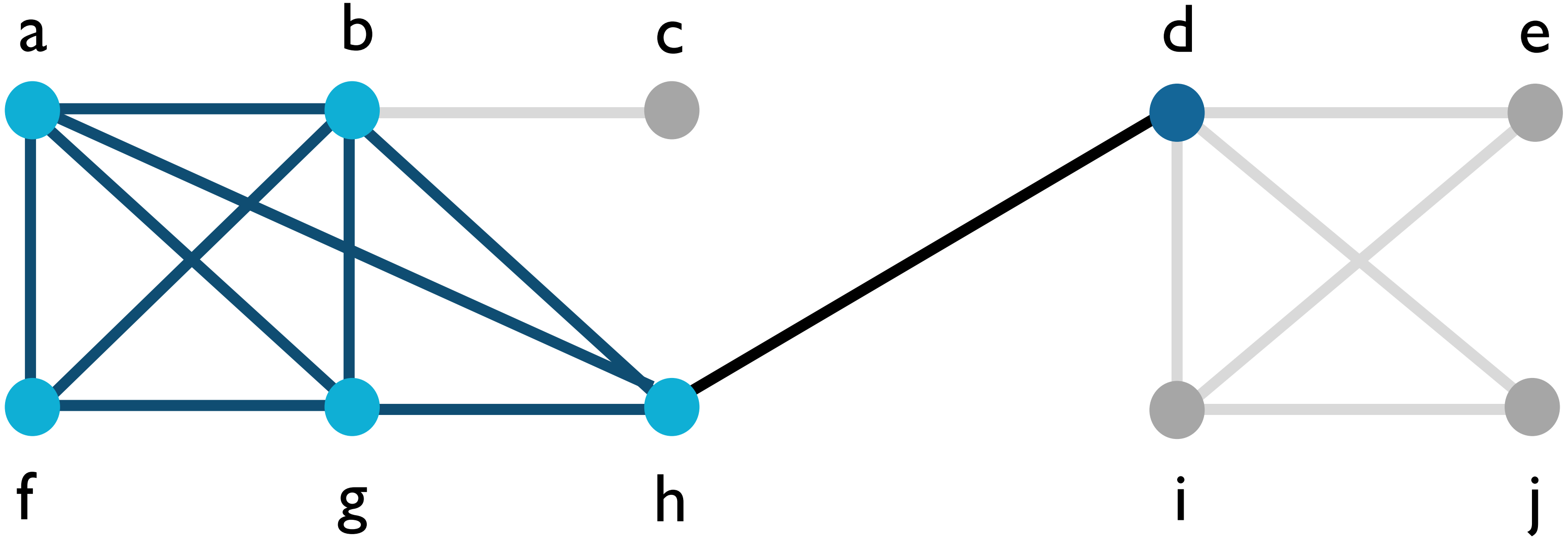
optimal density:
9/5 = 1.8

current density:
11/7 = 1.625

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	2	X	3	4	3	X	1	curr. degree
		1		2				2		final degree

peeling: an algorithm



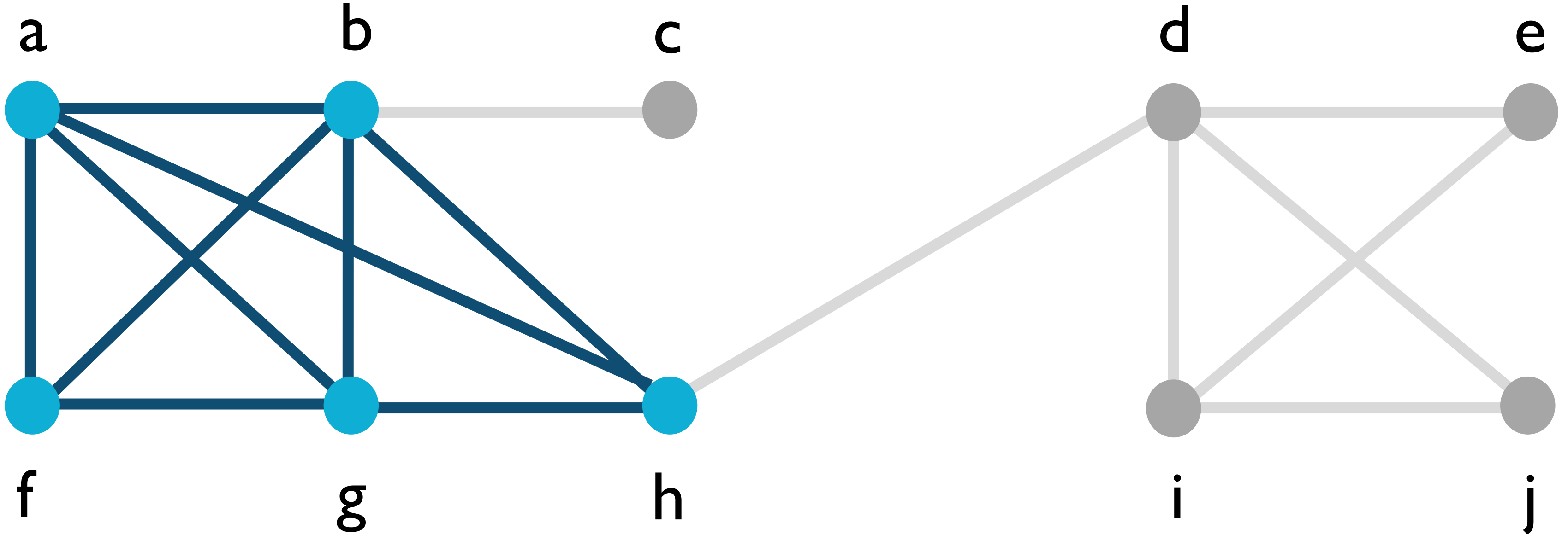
optimal density:
9/5 = 1.8

current density:
10/6 = 1.6666667

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	1	X	3	4	3	X	X	curr. degree
		1		2				2	1	final degree

peeling: an algorithm



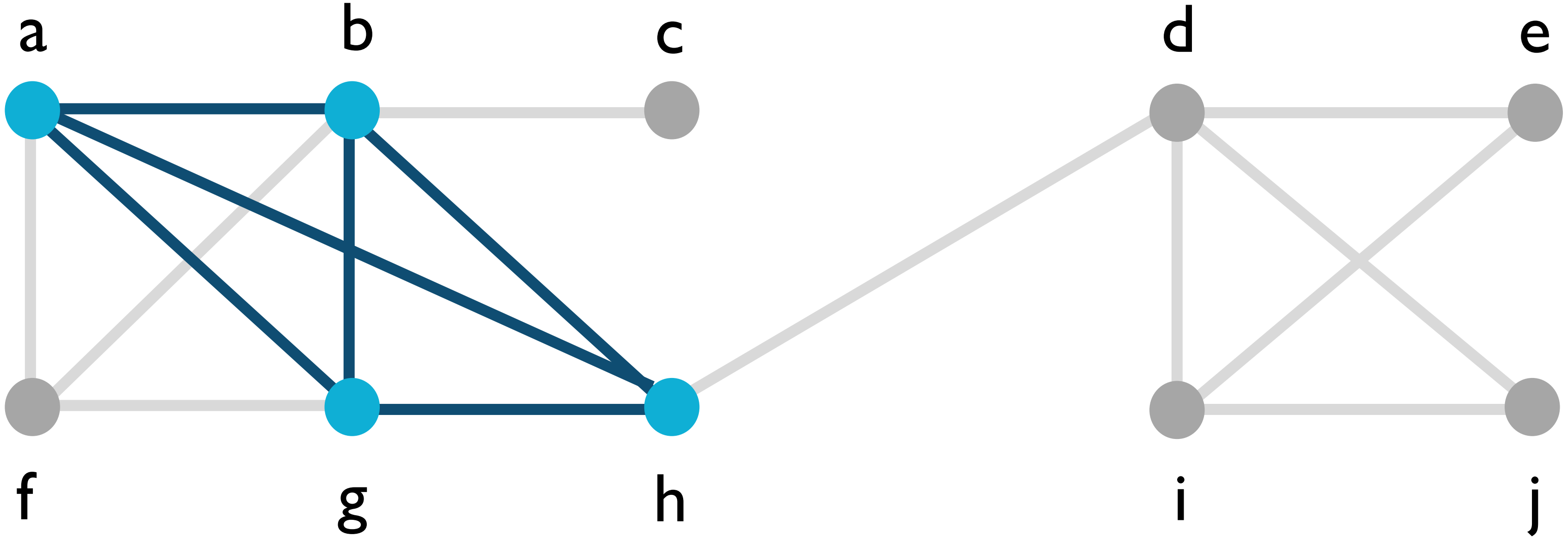
optimal density:
9/5 = 1.8

current density:
9/5 = 1.8

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	X	X	3	4	3	X	X	curr. degree
		1	1	2				2	1	final degree

peeling: an algorithm



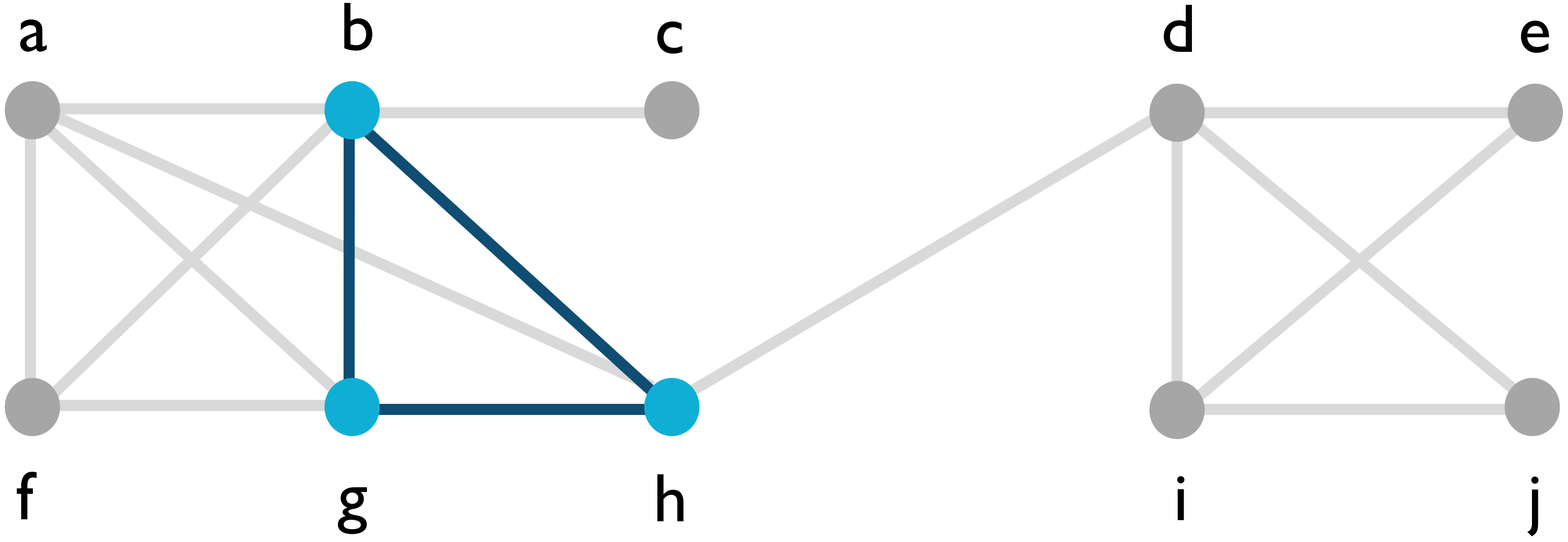
optimal density:
9/5 = 1.8

current density:
6/4 = 1.5

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
3	3	X	X	X	X	3	3	X	X	curr. degree
		1	1	2	3			2	1	final degree

peeling: an algorithm



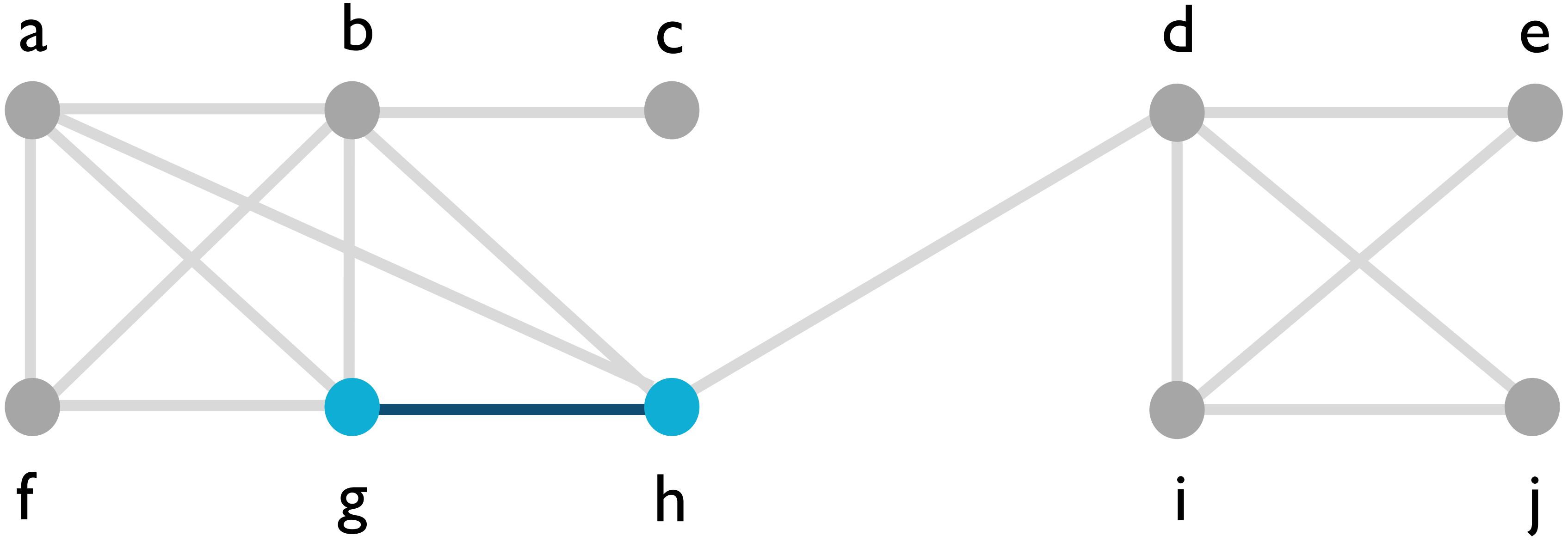
optimal density:
9/5 = 1.8

current density:
3/3 = 1

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	2	X	X	X	X	2	2	X	X	curr. degree
3		1	1	2	3			2	1	final degree

peeling: an algorithm



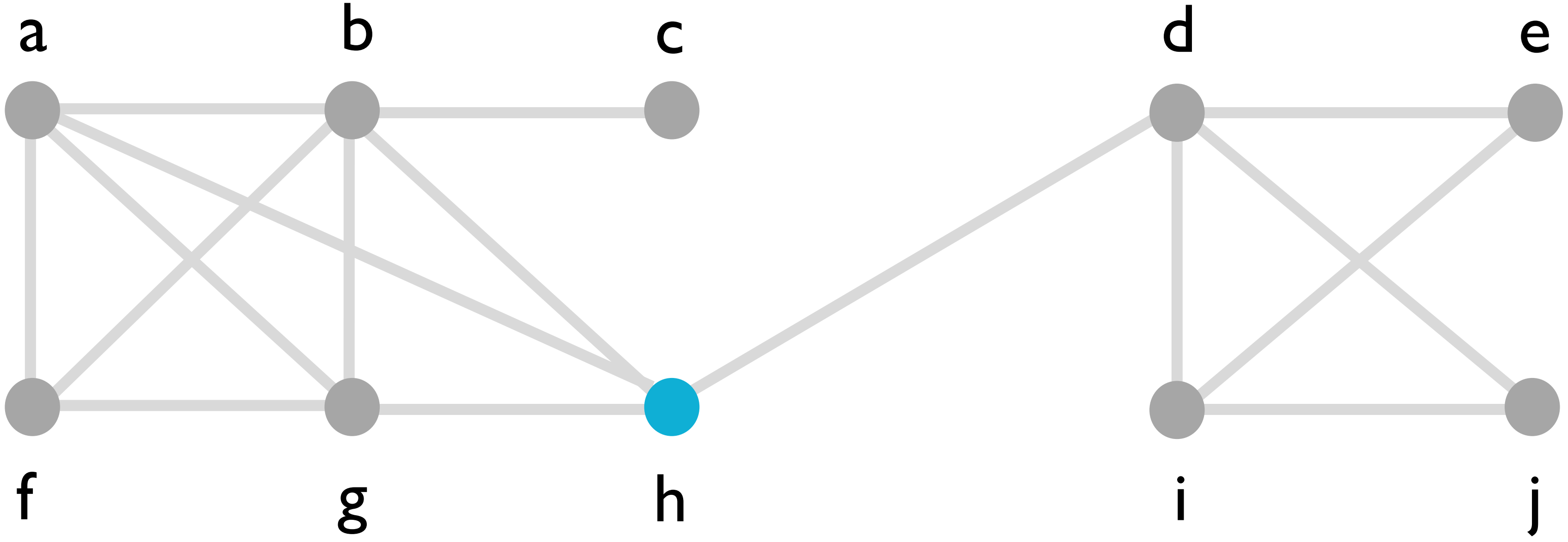
optimal density:
9/5 = 1.8

current density:
1/2 = 0.5

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	1	1	X	X	curr. degree
3	2	1	1	2	3			2	1	final degree

peeling: an algorithm



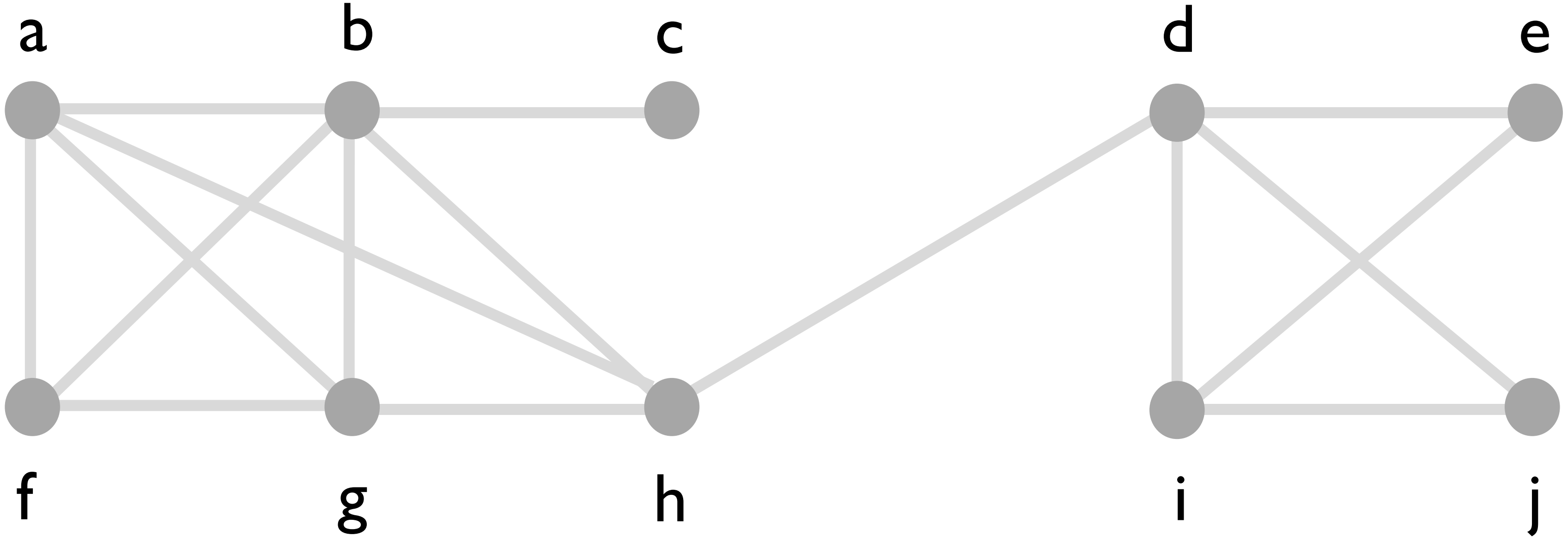
optimal density:
9/5 = 1.8

current density:
0/1 = 0

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	0	X	X	curr. degree
3	2	1	1	2	3	1		2	1	final degree

peeling: an algorithm



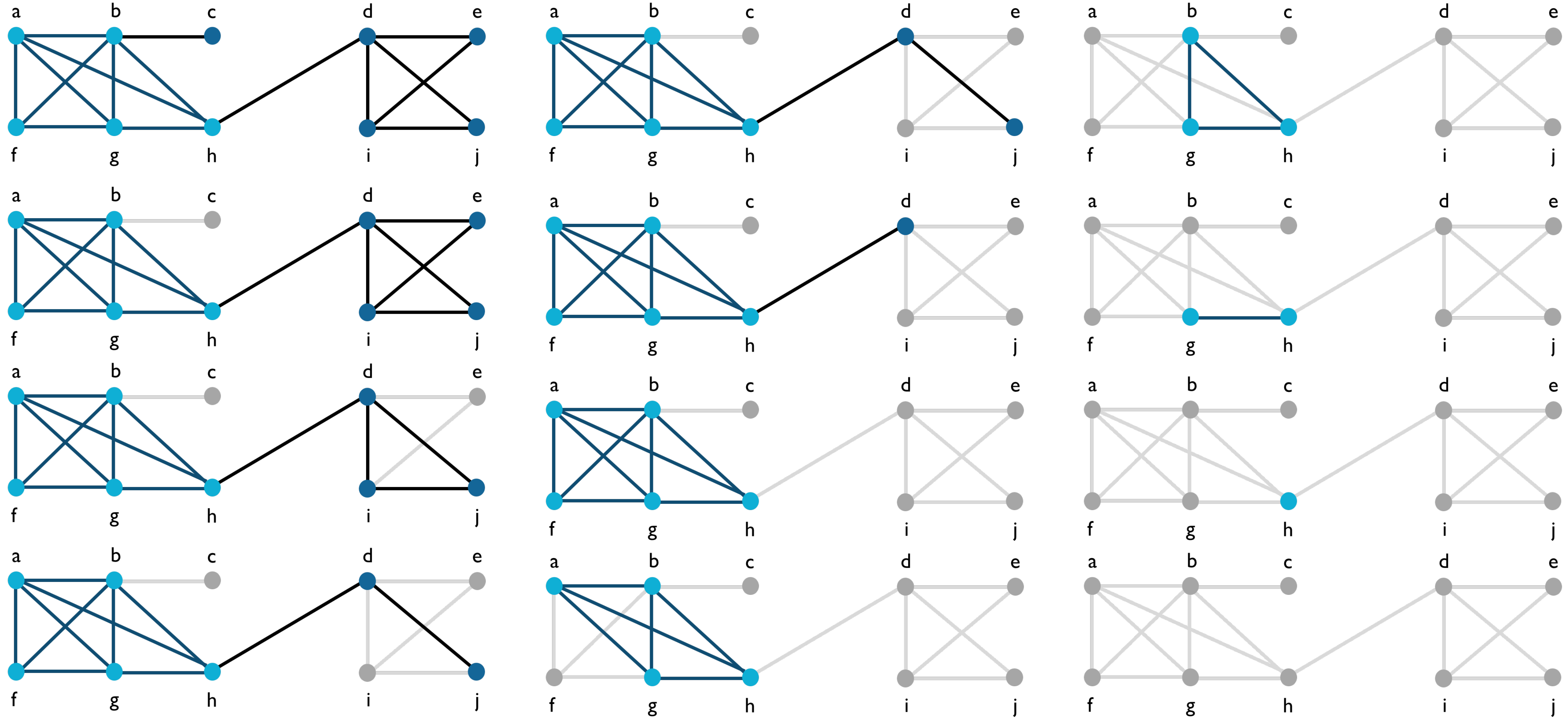
optimal density:
9/5 = 1.8

current density:
0/0 = undefined

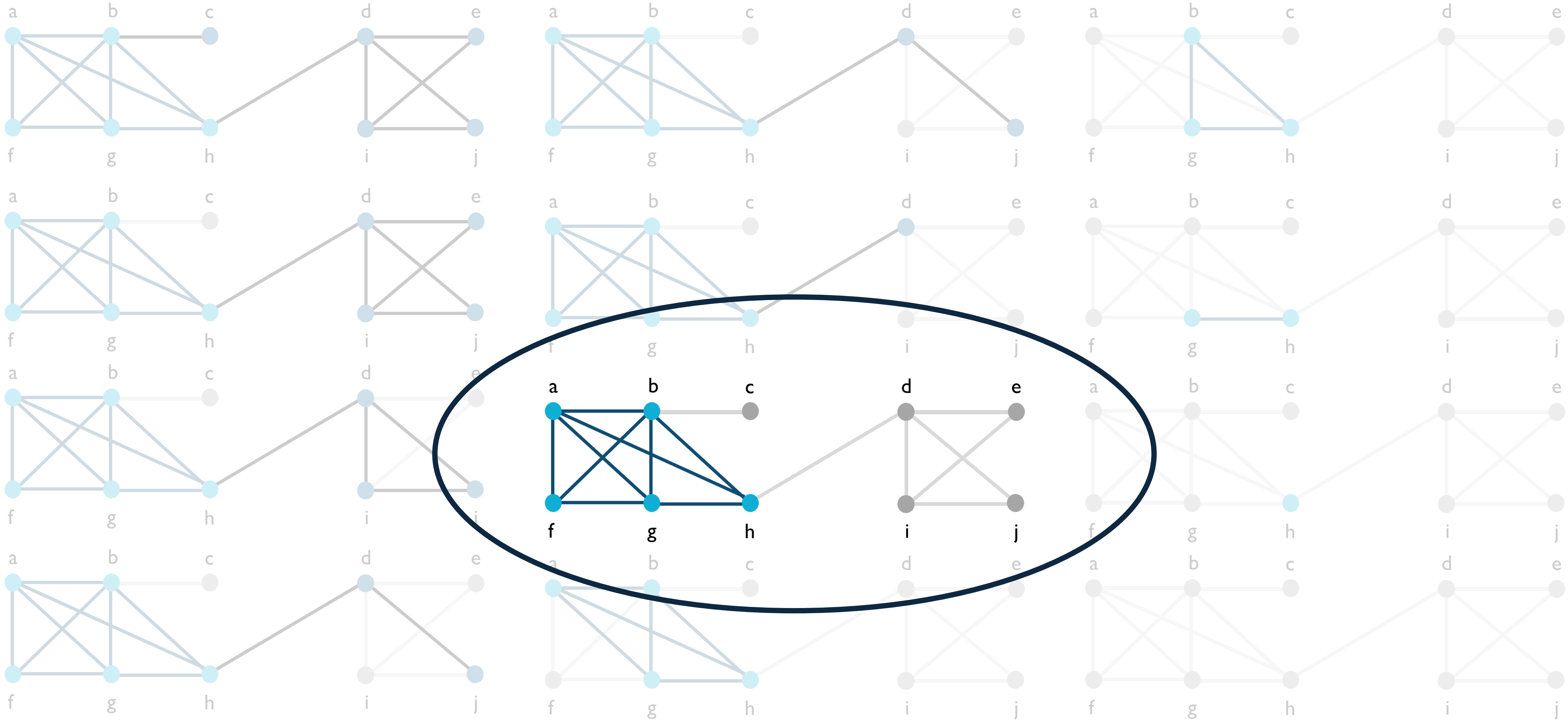
highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	1	X	X	curr. degree
3	2	1	1	2	3	1	0	2	1	final degree

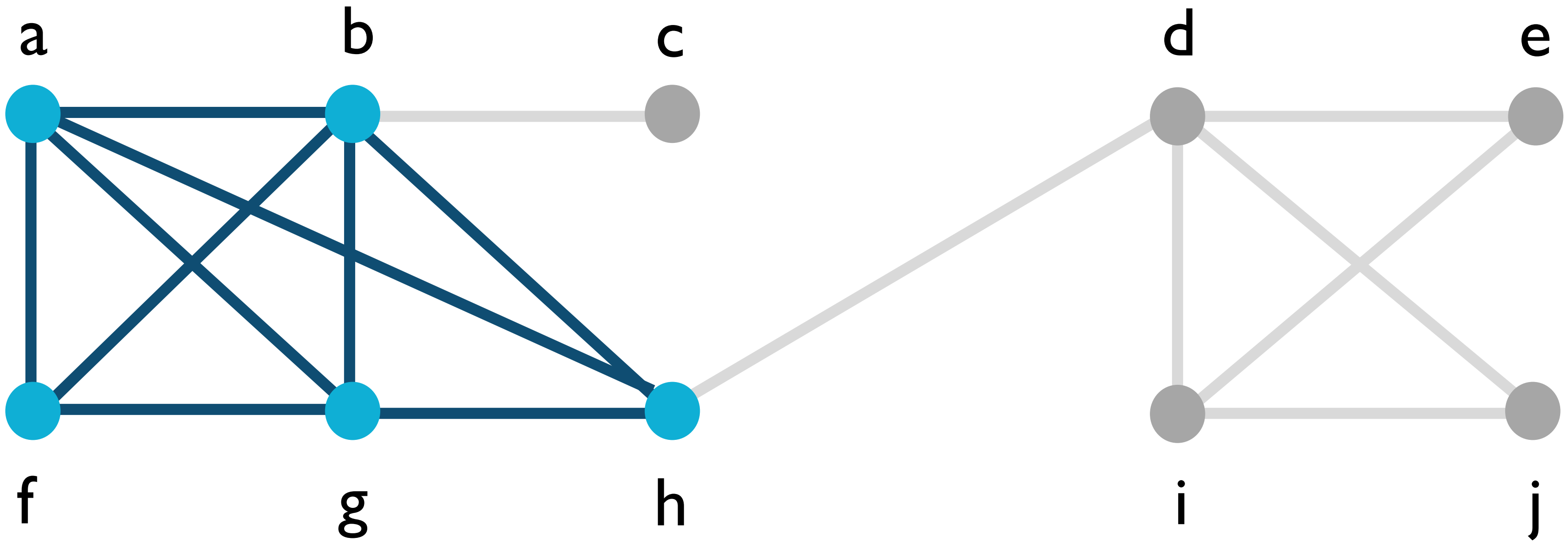
peeling: an algorithm



peeling: an algorithm



peeling: an algorithm



In this case, the algorithm did produce the expected densest subgraph. However, this is not always the case.

peeling: summary

- very fast! (runs in linear time)
- ½-approximation for DSP (Charikar, 2000)
- *usually* about 80% good on real world graphs
- used in practice (real applications)
- how can we do better?

iterative peeling (Greedy++)

- Decide on a number of iterations T;
- Keep an array of loads for each vertex, all initialized to 0. Let the load of v during iteration i = $\mathrm{load}_i(v)$;
- In each iteration i, repeatedly find the vertex that minimizes $\mathrm{current\_deg}(v) + \mathrm{load}_{i-1}(v)$, and remove it;
- Let $\mathrm{load}_i(v)$ = $\mathrm{load}_{i-1}(v)$ + the degree of v when it was removed;

(Boob et. al, 2019)

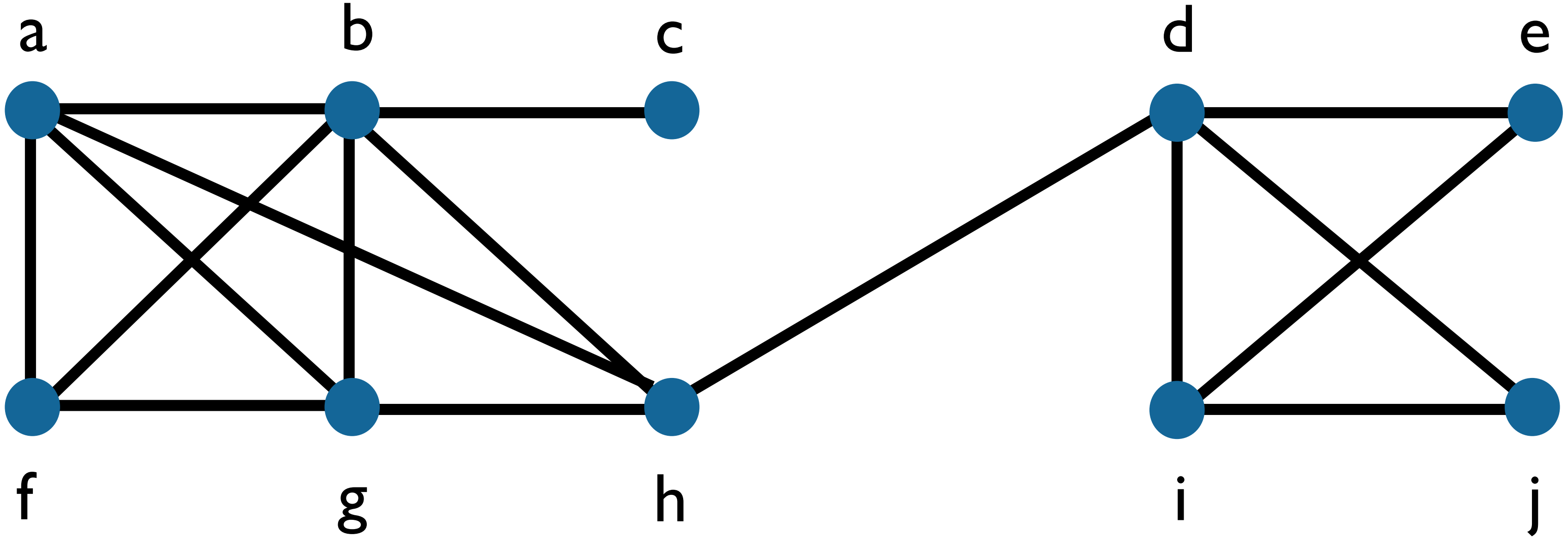
iterative peeling (Greedy++)

- Let $v_{1,j}\,, v_{2,j}\,, \ldots, v_{n,j}$ be the ordering from vertex removal during iteration j;
- After T iterations, we choose the suffix over all orderings $S_{i,j} = \{v_{i,j},\, v_{i+1,j}, \ldots, v_{n,j}\}$ that induces the subgraph with the highest density λ;
- Typically, this is a suffix $S_{i,T} = \{v_{i,T},\, v_{i+1,T}, \ldots, v_{n,T}\}$.

(Boob et. al, 2019)

iterative peeling

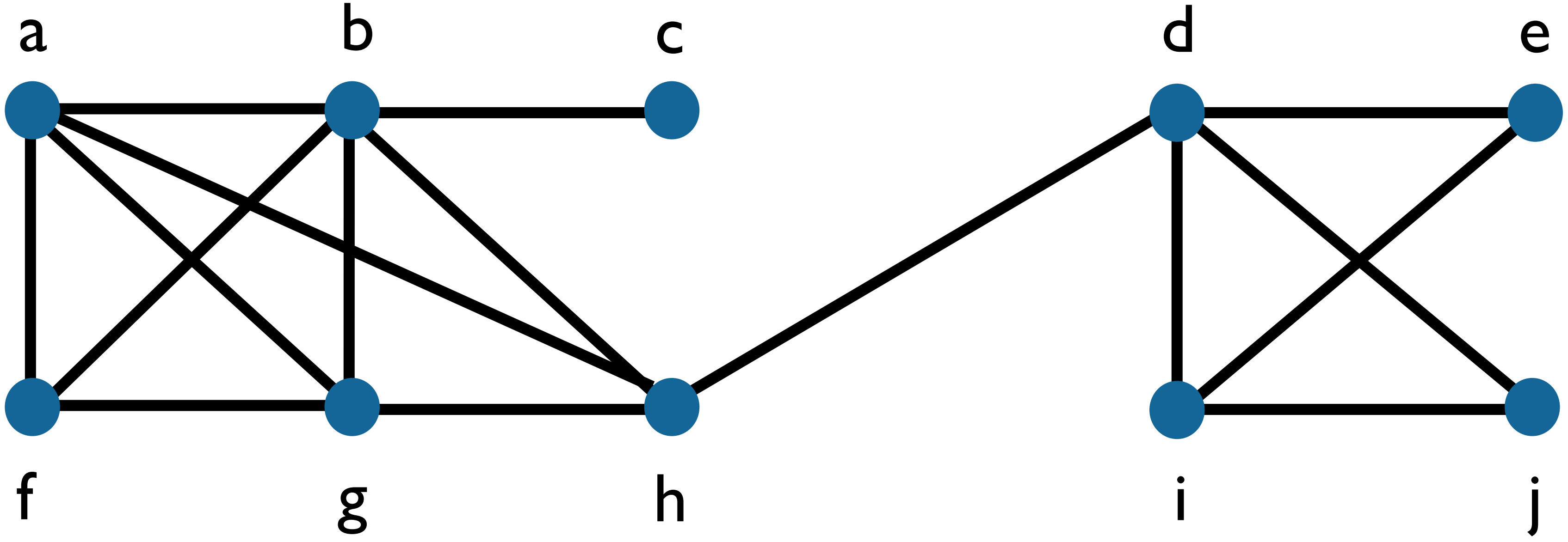
(Boob et. al, 2019)



a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	1	X	X	curr. degree
3	2	1	1	2	3	1	0	2	1	final degree

iterative peeling

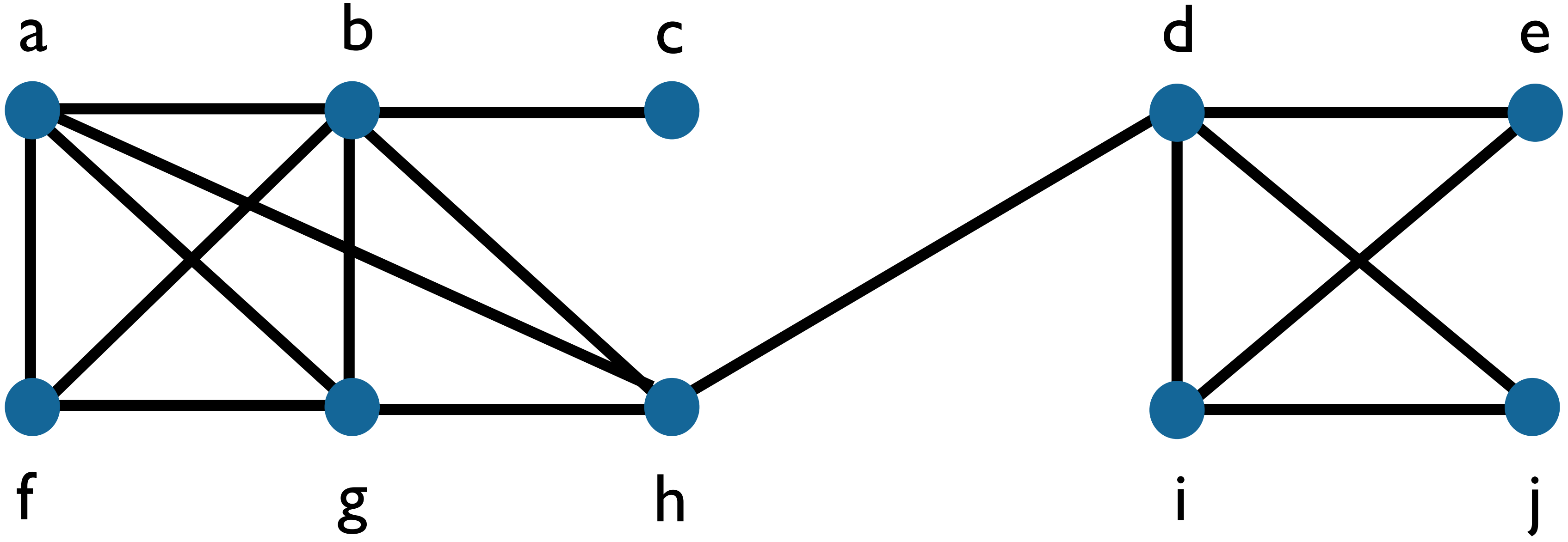
(Boob et. al, 2019)



a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	1	X	X	curr. degree
3	2	1	1	2	3	1	0	2	1	loads

iterative peeling

(Boob et. al, 2019)

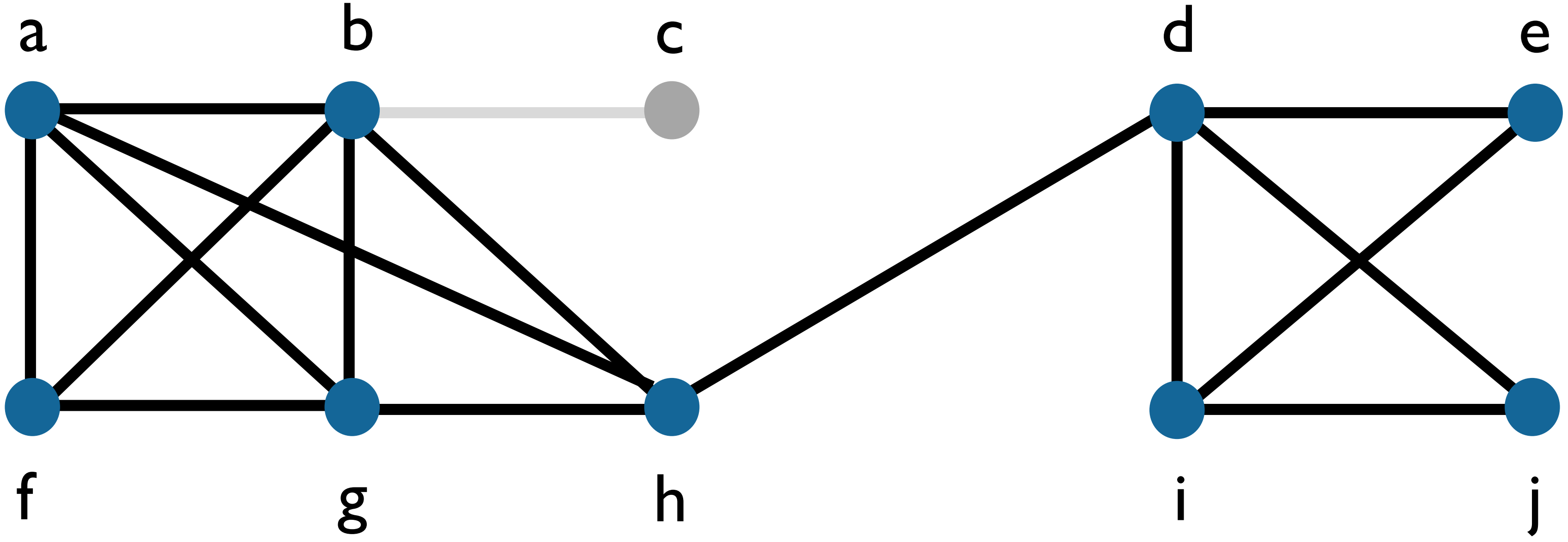


We obtain a new peeling order that changes with further iterations and eventually stabilizes.

a	b	c	d	e	f	g	h	i	j	
3	2	1	1	2	3	1	0	2	1	load
4	5	1	4	2	3	4	4	3	2	degree
7	7	2	5	4	6	5	4	5	3	curr load + deg
										load update

iterative peeling

(Boob et. al, 2019)



We obtain a new peeling order that changes with further iterations and eventually stabilizes.

a	b	c	d	e	f	g	h	i	j	
3	2	1	1	2	3	1	0	2	1	load
4	5	X	4	2	3	4	4	3	2	degree
7	7	1	5	4	6	5	4	5	3	curr load + deg
		1								load update

iterative peeling: summary

- Boob et. al experimentally showed that this seems to always eventually work!
- Conjecture: this is a $(1-\epsilon)$–approximation for DSP with $(O\frac{1}{\epsilon^2})$ iterations required
- can we use this method to solve other problems?

Part 3:

The Densest Supermodular Set Problem (DSSP)

(iterative peeling for DSSP; convergence)

set functions

A set function assigns values to subsets of a set. We call the overall set we are working with the ground set.

In other words, a set function is a function from the powerset of S to the real numbers.

$$f : 2^S \to \mathbb{R} \cup \{\pm\infty\}$$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- normalized if $f(\emptyset) = 0$
- monotone increasing if
$A \subset B \implies f(A) < f(B)$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- normalized if $f(\emptyset) = 0$
- monotone decreasing if
$A \subset B \implies f(A) > f(B)$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- additive if $f(A \dot{\cup} B) = f(A) + f(B)$
- subadditive if
  $f(A \dot{\cup} B) \leq f(A) + f(B)$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- additive if $f(A \dot\cup B) = f(A) + f(B)$
- superadditive if
  $f(A \dot\cup B) \geq f(A) + f(B)$

set functions (marginal values)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

The marginal value of adding a new element to a set is the gain or loss incurred by adding that element to the set. Formally,

$$f(v|S) = f(S \cup \{v\}) - f(S), \;\; S \subsetneq V, v \notin S$$

submodularity, formally

A submodular function is a real-valued set function characterized by diminishing returns:

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$$

$$A \subsetneq B, \ \ v \notin B.$$

supermodularity, formally

A supermodular function is a real-valued set function characterized by increasing returns:

$$f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B)$$

$$A \subsetneq B, \;\; v \notin B.$$

modularity, formally

A modular function is both submodular and supermodular.

$$f(A \cup \{v\}) - f(A) = f(B \cup \{v\}) - f(B)$$

$$A \subsetneq B, \;\; v \notin B.$$

Notice that modular functions are additive!

rewritten using marginal values…

submodular:

$$f(v|A) \geq f(v|B), \ A \subsetneq B, \ v \notin B.$$

supermodular:

$$f(v|A) \leq f(v|B), \ A \subsetneq B, \ v \notin B.$$

modular:

$$f(v|A) = f(v|B), \ A \subsetneq B, \ v \notin B.$$

The densest subgraph problem (DSP) is a special case of the densest supermodular set problem (DSSP).

(Charikar, Quanrud, Torres, 2022)

The DSSP, formally.

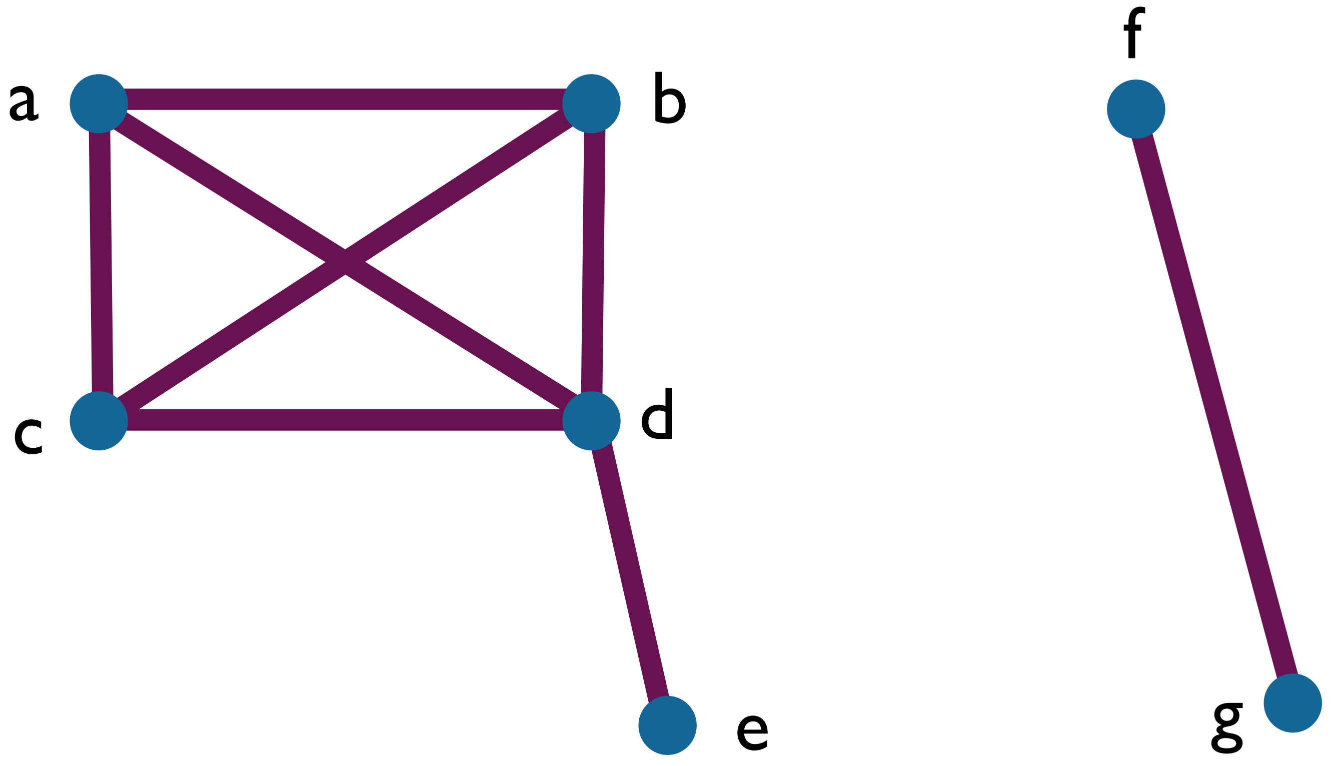
Given a non-negative supermodular function $f : 2^V \to \mathbb{R}^{\geq 0}$, let $S \subseteq V$. Then,

$$density(S) = \frac{f(S)}{|S|}$$

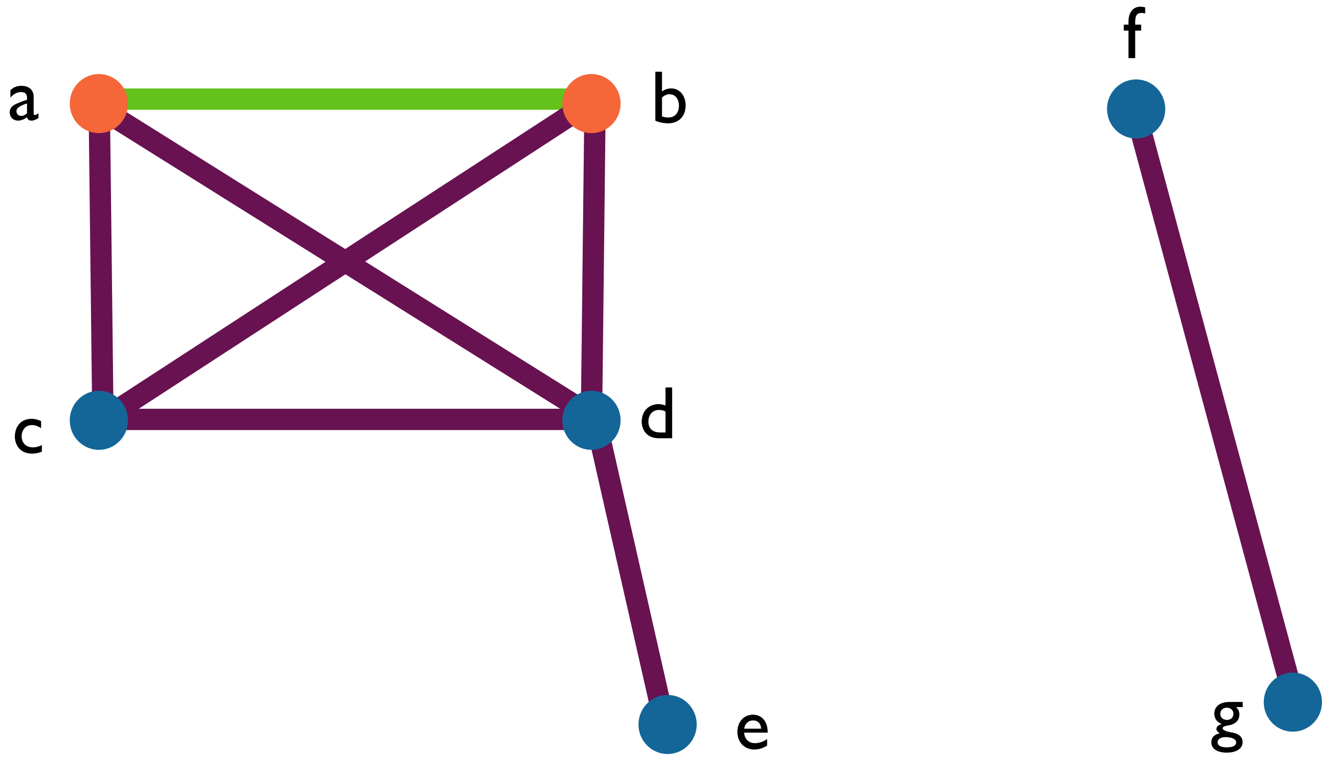
and we want to find the subset with the optimal density, λ^*:

$$\lambda^* = \max_{S \subseteq V} \frac{f(S)}{|S|}$$

|E(S)| is supermodular!

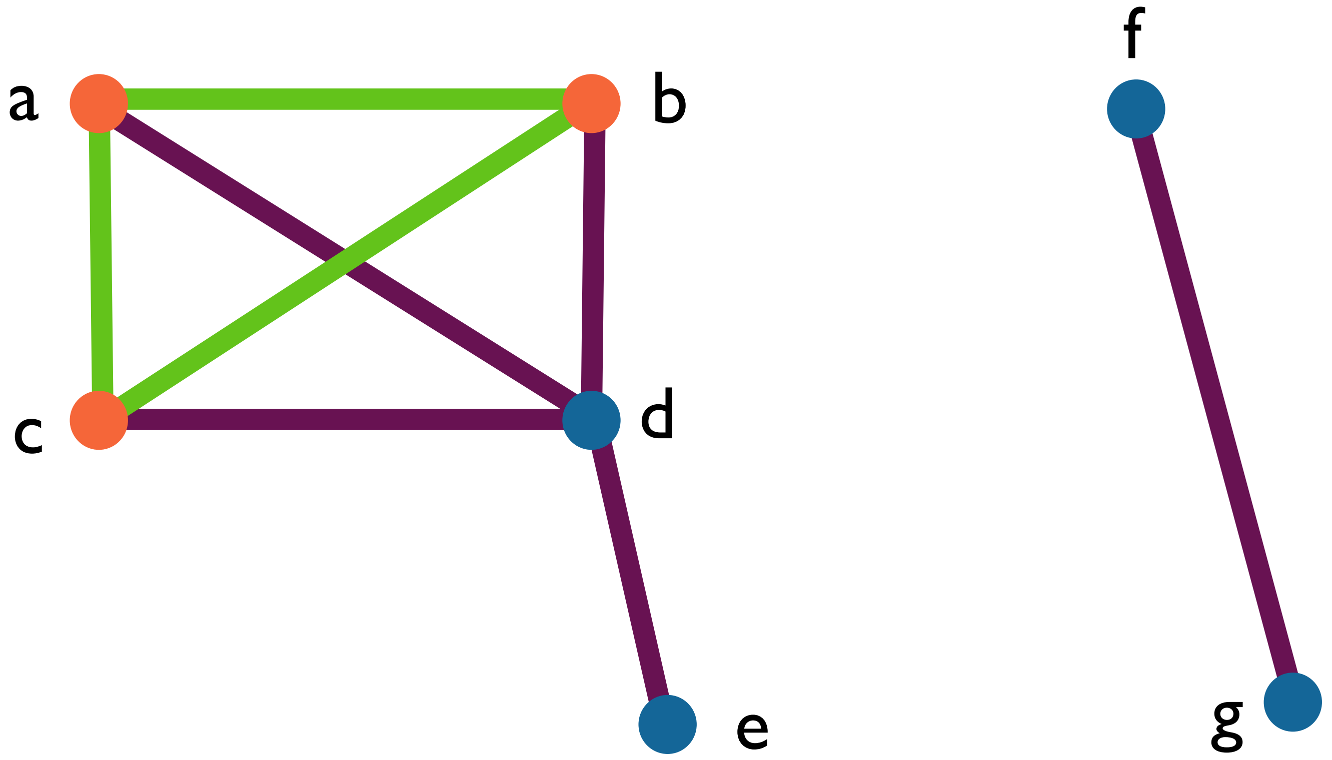


|E(S)| is supermodular!



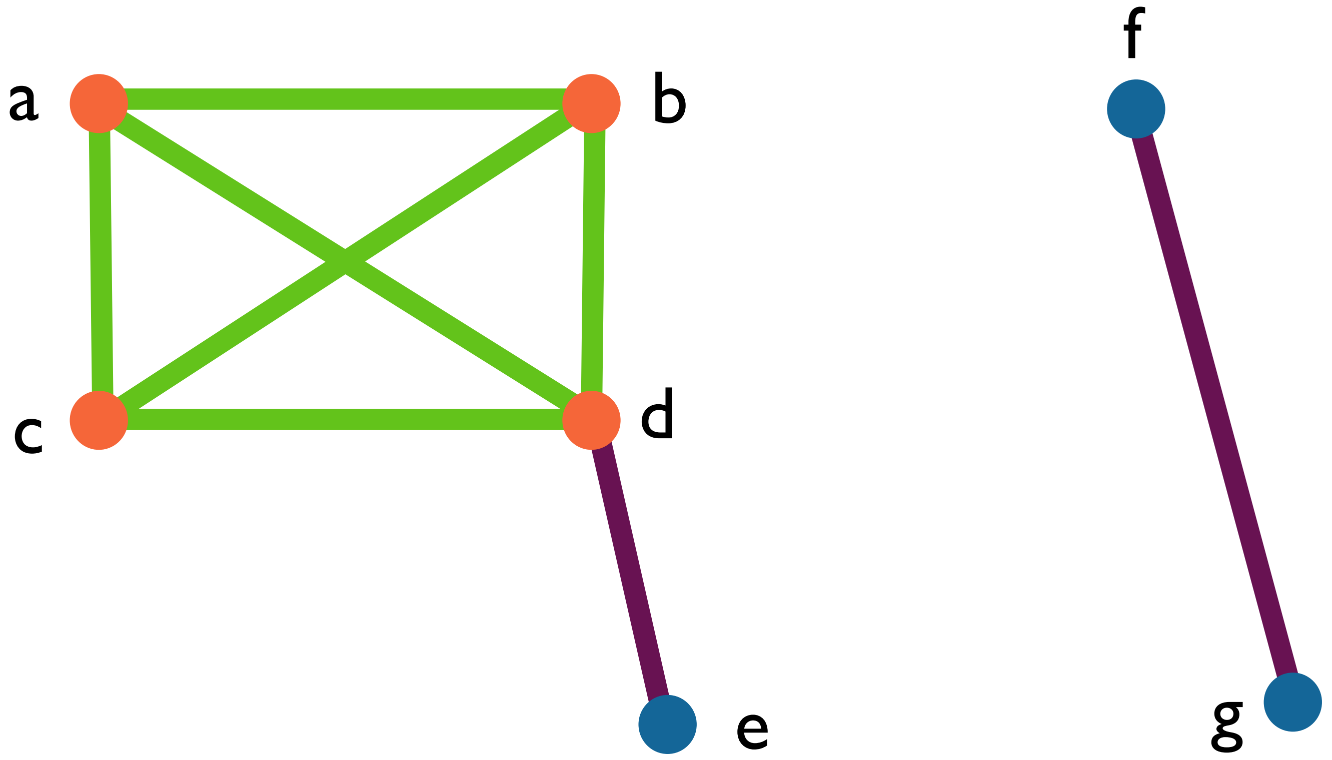
If we take S = {a, b}, then |E(S)| = 1.

|E(S)| is supermodular!



If we take T = {a, b, c}, then |E(T)| = 3. So f(c|S) = 2.

|E(S)| is supermodular!



If we take U = {a, b, c, d}, then |E(U)| = 6. So f(d|U) = 3!

SuperGreedy++, in general

- In general, we can replace the concept of current degree with the marginal value of v to the current set
- Everything else about iterative peeling remains exactly the same.

(Charikar, Quanrud, Torres, 2022)

SuperGreedy++: summary

- This is a $(1-\epsilon)$– approximation for the DSSP with $O(1 - \frac{1}{\epsilon^2} \cdot \frac{\delta}{\lambda^*} \log n)$ iterations required
- Therefore this is also a $(1-\epsilon)$– approximation for the DSP
- Iterative peeling works for any set with a supermodular function

(Charikar, Quanrud, Torres, 2022)

Main References

László Lovász. "**Submodular Functions and Convexity**". (1983)

Moses Charikar. "**Greedy Approximation Algorithms for Finding Dense Components in a Graph**". (2000)

Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. "**Flowless: Extracting Densest Subgraphs Without Flow Computations**". (2019)

Chandra Chekuri, Kent Quanrud, Manuel R. Torres. "**Densest Subgraph: Supermodularity, Iterative Peeling, and Flow**". (2022)