Online Social Networks and Media

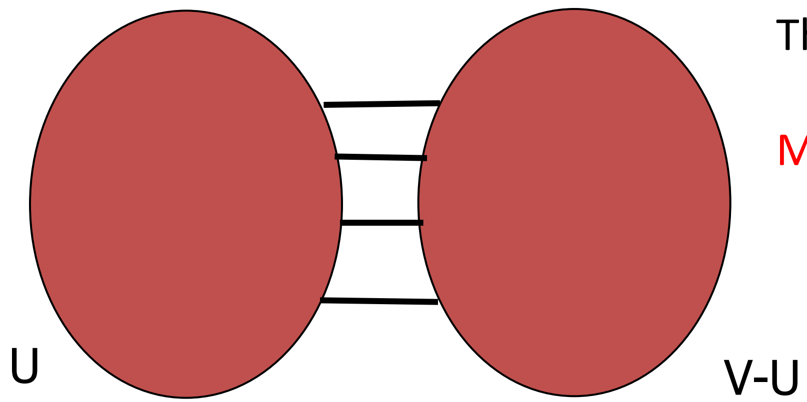
Graph partitioning

- The general problem
  - Input: a graph G=(V,E)
    - edge (u,v) denotes similarity between u and v
    - weighted graphs: weight of edge captures the degree of similarity
  - Partitioning as an optimization problem:
    - Partition the nodes in the graph such that nodes within clusters are well interconnected (high edge weights), and nodes across clusters are sparsely interconnected (low edge weights)
    - most graph partitioning problems are NP hard

Measuring connectivity

- What does it mean that a set of nodes are well or sparsely interconnected?

- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  - small min-cut implies sparse connectivity
  - $\min_{U} E(U, V-U) = \sum_{i \in U} \sum_{j \in V-U} A[i,j]$



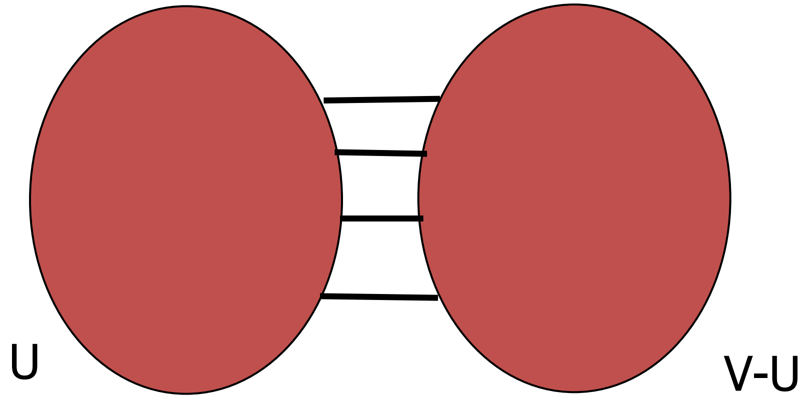
This problem can be solved in polynomial time

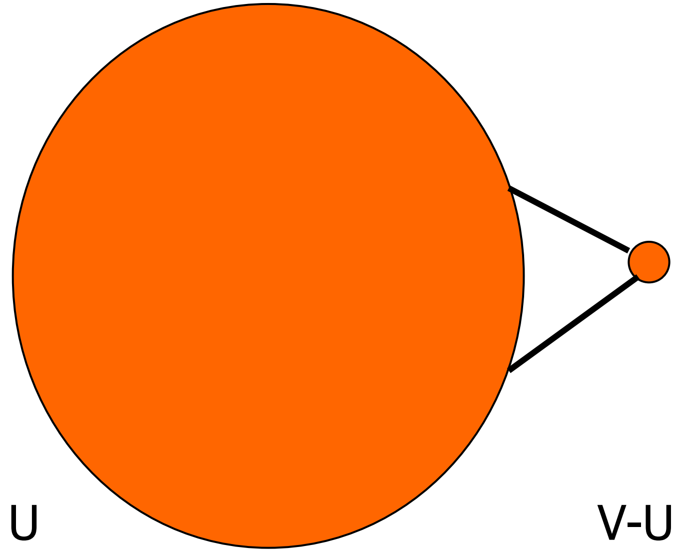
Min-cut/Max-flow algorithm

Measuring connectivity

- What does it mean that a set of nodes are well interconnected?

- min-cut: the min number of edges such that when removed cause the graph to become disconnected
  - not always a good idea!





A bad example

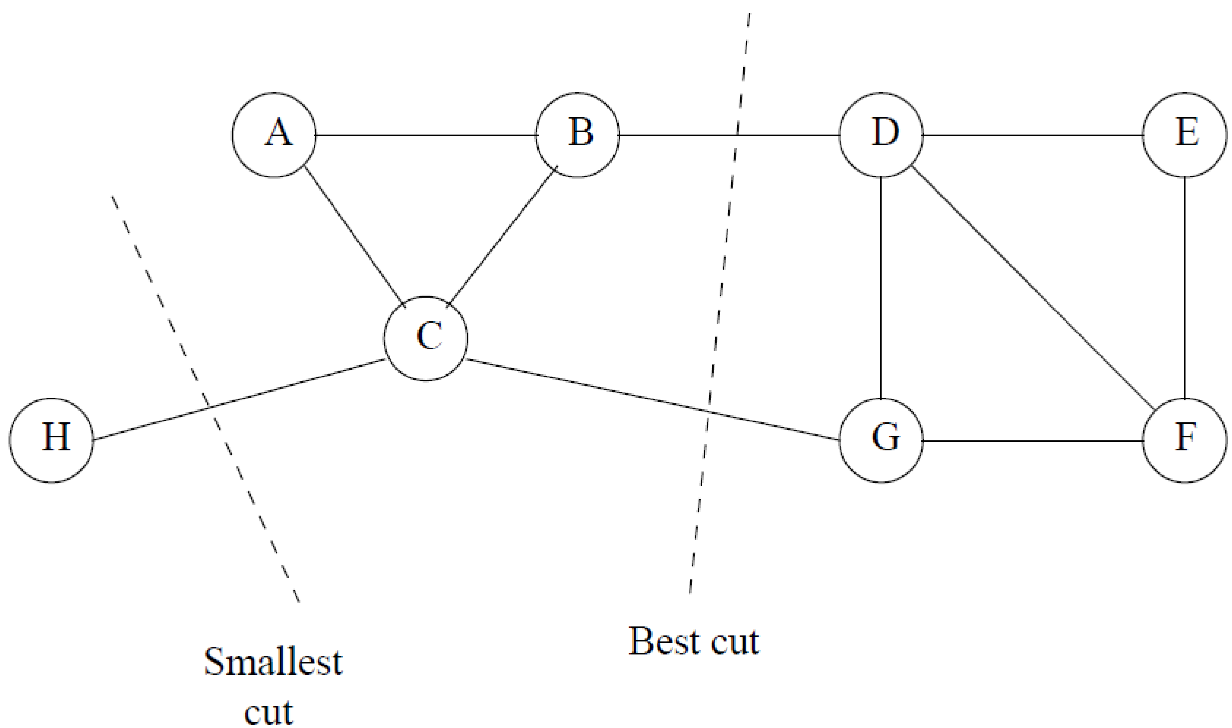


Figure 10.11: The smallest cut might not be the best cut

Graph Bisection

- Since the minimum cut does always yield good results we need an extra constraints to make the problem meaningful.
- Graph Bisection refers to the problem of partitioning the nodes of the graph into two equal sets.
- Kernighan-Lin algorithm: Start with random equal partitions and then swap nodes to improve some quality metric (e.g., cut, modularity, etc).

Graph expansion

- Normalize the cut by the size of the smallest component
- Cut ratio:

$$\alpha = \frac{E(U, V - U)}{\min\{|U|, |V-U|\}}$$

- Graph expansion:

$$\alpha(G) = \min_{U} \frac{E(U, V - U)}{\min\{|U|, |V-U|\}}$$

- Other Normalized Cut Ratio:

$$\beta = \frac{E(U,V-U)}{Vol(U)} + \frac{E(U,V-U)}{Vol(V-U)}$$

Vol(U) = number of edges with one endpoint in U
= total degree of nodes in U

Spectral analysis

- The Laplacian matrix $L = D - A$ where
  - A = the adjacency matrix
  - $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$
    - d_i = degree of node i

- Therefore
  - $L(i,i) = d_i$
  - $L(i,j) = -1$, if there is an edge (i,j)

Laplacian Matrix properties

- The matrix L is symmetric and positive semi-definite
  - all eigenvalues of L are positive
- The matrix L has 0 as an eigenvalue, and corresponding eigenvector $w_1 = (1,1,\ldots,1)$
  - $\lambda_1 = 0$ is the smallest eigenvalue

The second smallest eigenvalue

- The second smallest eigenvalue (also known as Fielder value) λ_2 satisfies

$$\lambda_2 = \min_{x \perp w_1, \|x\|=1} x^T L x$$

- The eigenvector for eigenvalue λ_2 is called the Fielder vector. It minimizes

$$\lambda_2 = \min_{x \neq 0} \sum_{(i,j) \in E} (x_i - x_j)^2 \quad \text{where} \quad \sum_i x_i = 0$$

Spectral ordering

- The values of x minimize

$$\min_{x \neq 0} \sum_{(i,j) \in E} \left(x_i - x_j\right)^2 \qquad \sum_i x_i = 0$$

- For weighted matrices

$$\min_{x \neq 0} \sum_{(i,j)} A[i,j]\left(x_i - x_j\right)^2 \qquad \sum_i x_i = 0$$

- The ordering according to the x_i values will group similar (connected) nodes together
- Physical interpretation: The stable state of springs placed on the edges of the graph

Spectral partition

- Partition the nodes according to the ordering induced by the Fielder vector
- If $u = (u_1, u_2, \ldots, u_n)$ is the Fielder vector, then split nodes according to a threshold value s
  - bisection: s is the median value in u
  - ratio cut: s is the value that minimizes α
  - sign: separate positive and negative values ($s=0$)
  - gap: separate according to the largest gap in the values of u
- This works well (provably for special cases)

Fielder Value

- The value λ_2 is a good approximation of the graph expansion

$$\frac{\alpha(G)^2}{2d} \le \lambda_2 \le 2\alpha(G)$$

d = maximum degree

$$\frac{\lambda_2}{2} \le \alpha(G) \le \sqrt{\lambda_2(2d - \lambda_2)}$$

- For the minimum ratio cut of the Fielder vector we have that

$$\frac{\alpha^2}{2d} \le \lambda_2 \le 2\alpha(G)$$

- If the max degree d is bounded we obtain a good approximation of the minimum expansion cut

Thanks to Aris Gionis

MAXIMUM DENSEST SUBGRAPH

Finding dense subgraphs

- Dense subgraph: A collection of vertices such that there are a lot of edges between them
  - E.g., find the subset of email users that talk the most between them
  - Or, find the subset of genes that are most commonly expressed together
- Similar to community identification but we do not require that the dense subgraph is sparsely connected with the rest of the graph.

Definitions

- Input: undirected graph $G = (V, E)$.
- Degree of node u: $\deg(u)$
- For two sets $S \subseteq V$ and $T \subseteq V$:

$$E(S, T) = \{(\mathrm{u}, \mathrm{v}) \in E : u \in S, v \in T\}$$

- $E(S) = E(S, S)$: edges within nodes in S
- Graph Cut defined by nodes in $S \subseteq V$:

$E(S, \bar{S})$: edges between S and the rest of the graph

- Induced Subgraph by set S : $G_S = (S, E(S))$

Definitions

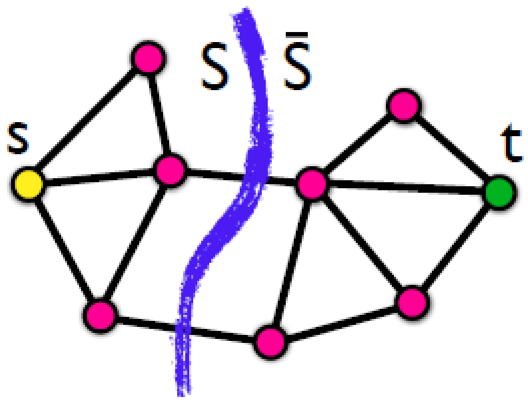
- How do we define the density of a subgraph?

- Average Degree:

$$d(S) = \frac{2|E(S)|}{|S|}$$

- Problem: Given graph G, find subset S, that maximizes density d(S)
  - Surprisingly there is a polynomial-time algorithm for this problem.

Min-Cut Problem



Given a graph* $G = (V, E)$,
A source vertex $s \in V$,
A destination vertex $t \in V$

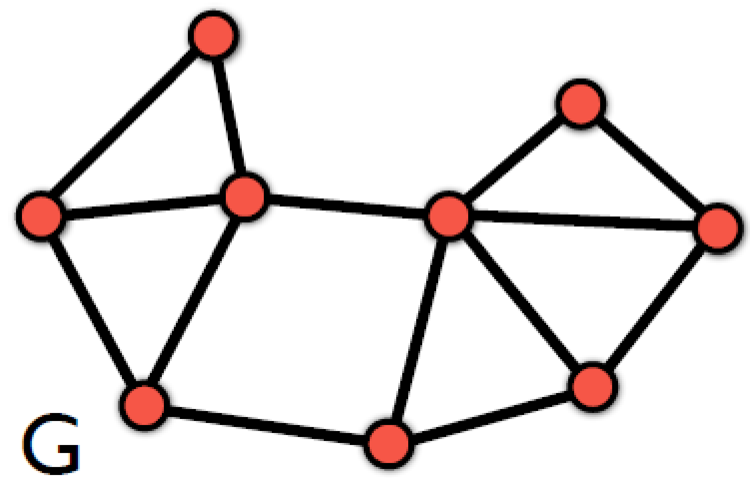
Find a set $S \subseteq V$
Such that $s \in S$ and $t \in \bar{S}$
That minimizes $E(S, \bar{S})$

* The graph may be weighted

Min-Cut = Max-Flow: the minimum cut maximizes the flow that can be sent from s to t. There is a polynomial time solution.

Decision problem

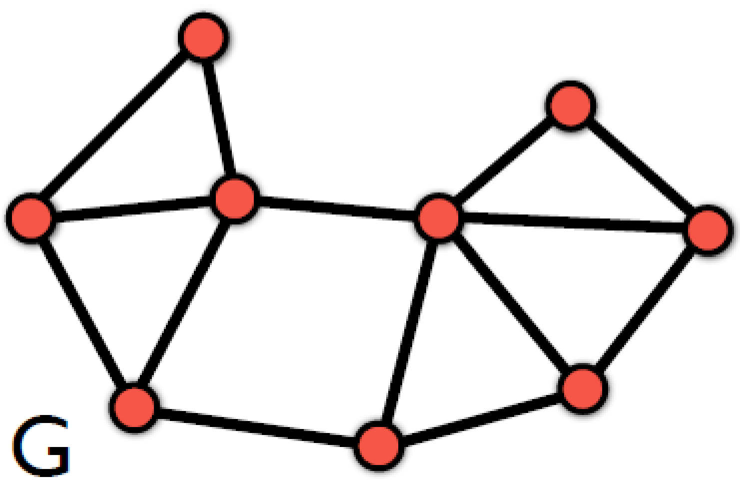
- Consider the decision problem
  - Is there a set S with $d(S) \geq c$?

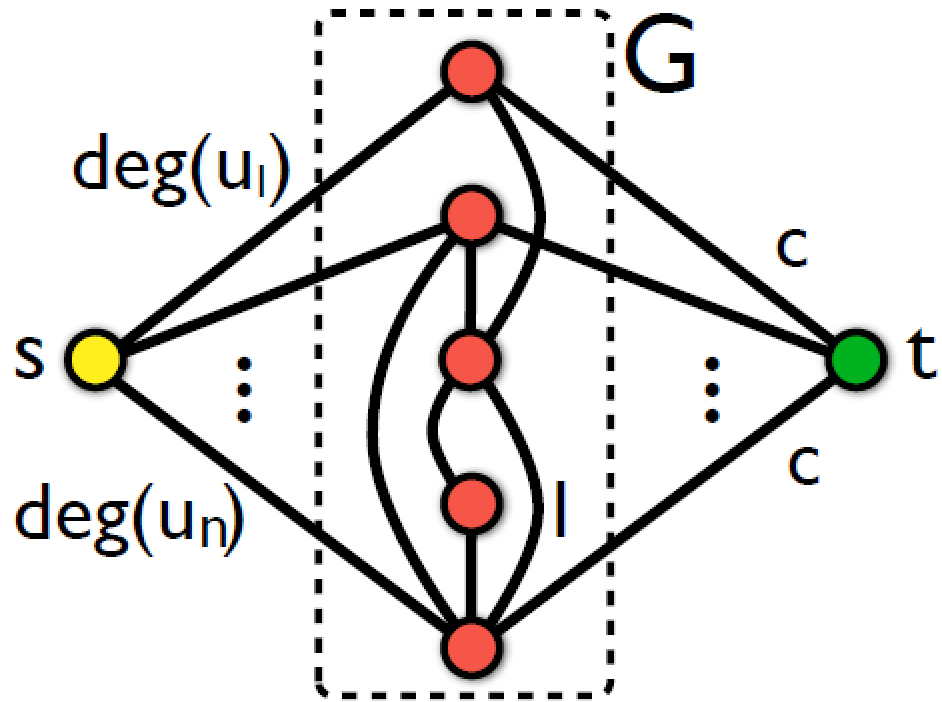


- $d(S) \geq c$
- $2|E(S)| \geq c|S|$
- $\sum_{v \in S} \deg(v) - E(S, \bar{S}) \geq c|S|$
- $2|E| - \sum_{v \in \bar{S}} \deg(v) - E(S, \bar{S}) \geq c|S|$
- $\sum_{v \in \bar{S}} \deg(v) + E(S, \bar{S}) + c|S| \leq 2|E|$

Transform to min-cut

- For a value c we do the following transformation

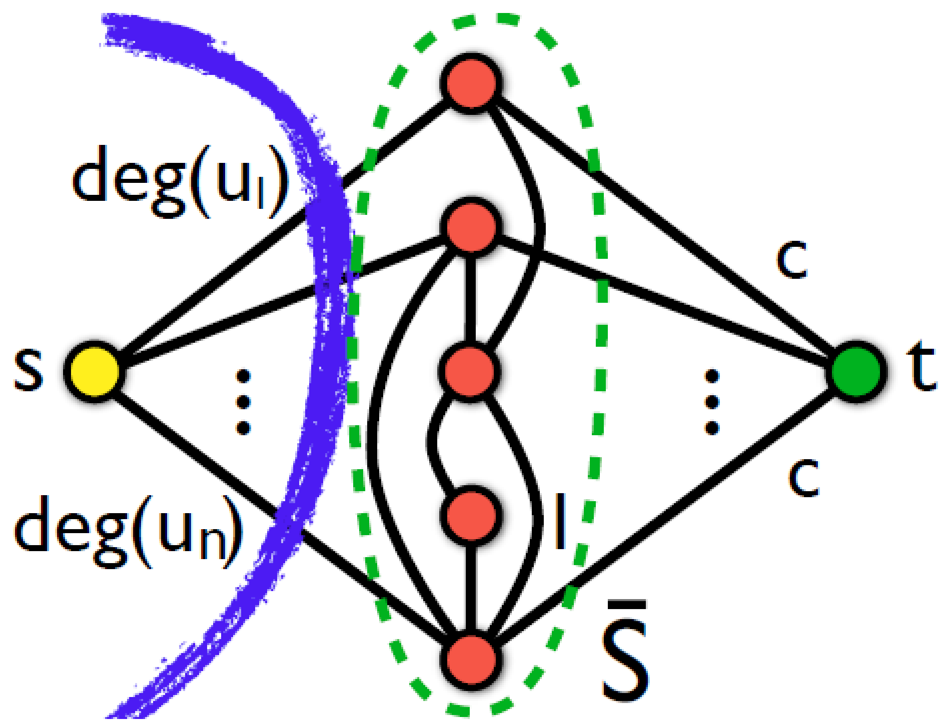




- We ask for a min s-t cut in the new graph

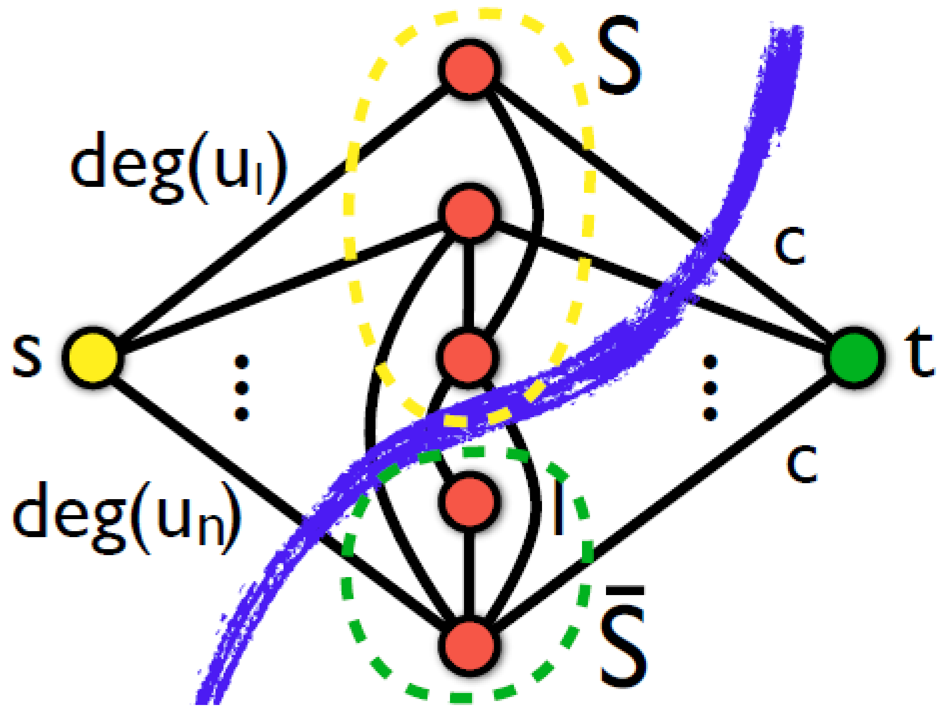
Transformation to min-cut

- There is a cut that has value $2|E|$



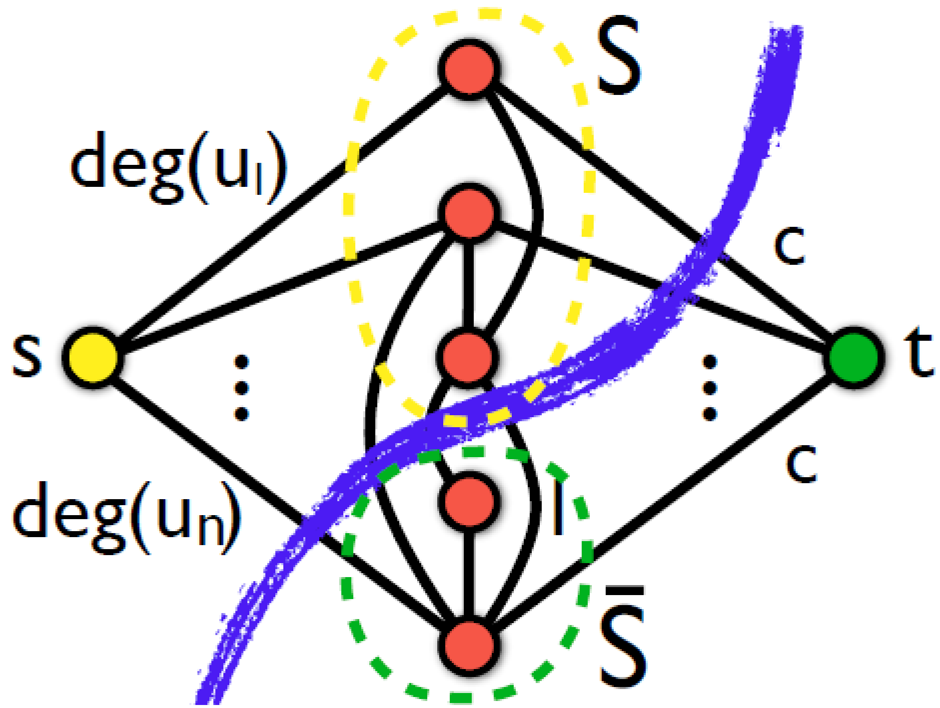
Transformation to min-cut

- Every other cut has value:
- $\sum_{v \in \bar{S}} \deg(v) + E(S, \bar{S}) + c|S|$



Transformation to min-cut

- If $\sum_{v \in \bar{S}} \deg(v) + E(S, \bar{S}) + c|S| \leq 2|E|$ then $S \neq \emptyset$ and $d(S) \geq c$



Algorithm (Goldberg)

Given the input graph G, and value c

1. Create the min-cut instance graph
2. Compute the min-cut
3. If the set S is not empty, return YES
4. Else return NO

How do we find the set with maximum density?

Min-cut algorithm

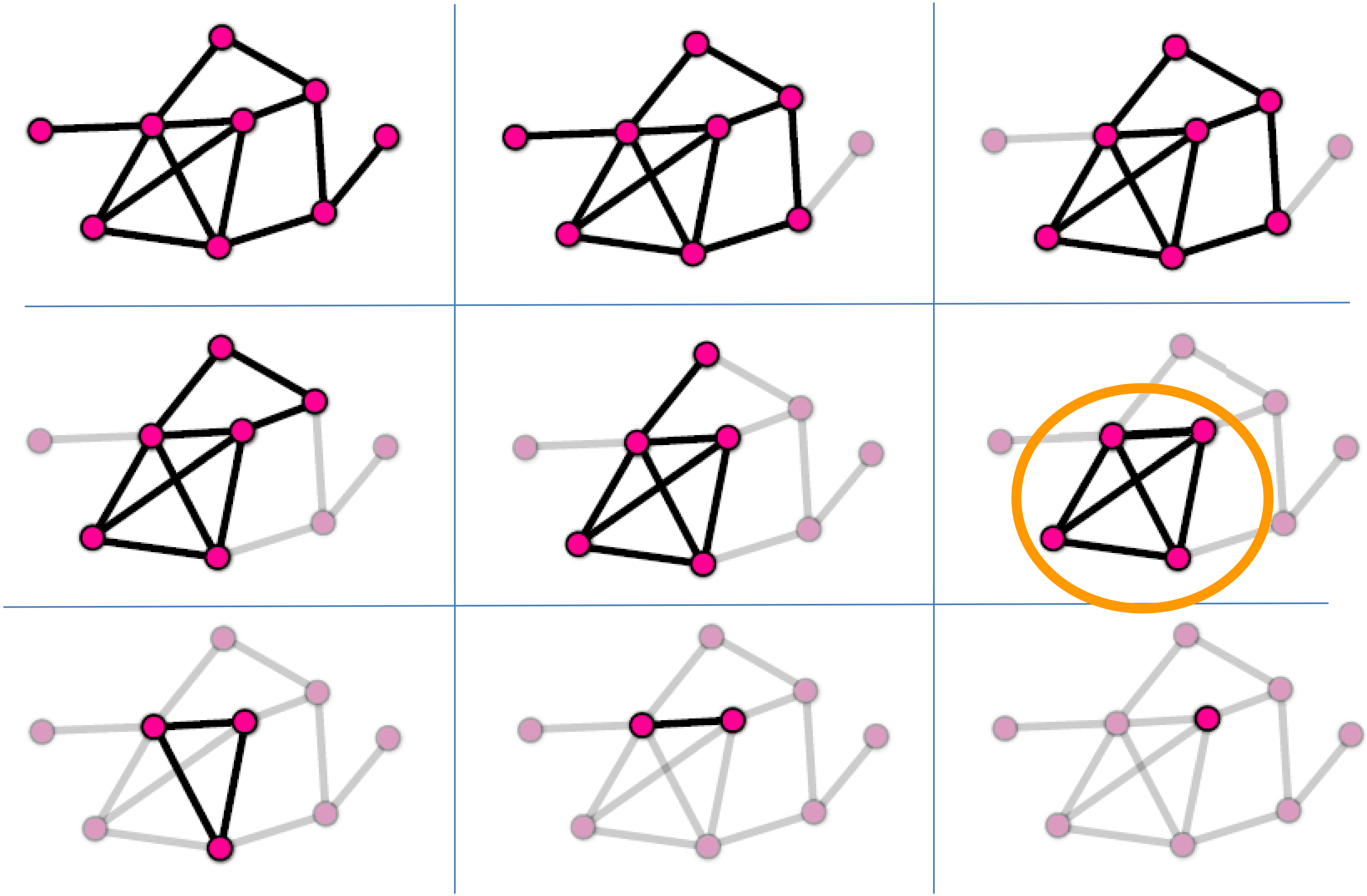
- The min-cut algorithm finds the optimal solution in polynomial time O(nm), but this is too expensive for real networks.
- We will now describe a simpler approximation algorithm that is very fast
  - Approximation algorithm: the ratio of the density of the set produced by our algorithm and that of the optimal is bounded.
    - We will show that the ratio is at most ½
    - The optimal set is at most twice as dense as that of the approximation algorithm.
- Any ideas for the algorithm?

Greedy Algorithm

Given the graph $G = (V, E)$

1. $S_0 = V$
2. For $i = 1 \dots |V|$
   a. Find node $v \in S$ with the minimum degree
   b. $S_i = S_{i-1} \setminus \{v\}$
3. Output the densest set S_i

Example



Analysis

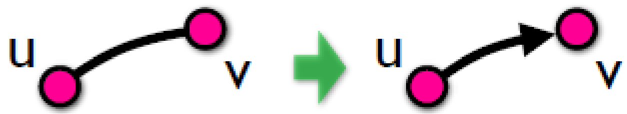
- We will prove that the optimal set has density at most 2 times that of the set produced by the Greedy algorithm.

- Density of optimal set: $d_{opt} = \max_{S \subseteq V} d(S)$
- Density of greedy algorithm d_g

- We want to show that $d_{opt} \leq 2 \cdot d_g$

Upper bound

- We will first upper-bound the solution of optimal
- Assume an arbitrary assignment of an edge (u, v) to either u or v



- Define:
  - $IN(u)$= # edges assigned to u
  - $\Delta = \max_{u \in V} IN(u)$
- We can prove that
  - $d_{opt} \leq 2 \cdot \Delta$

This is true for any assignment of the edges!

Lower bound

- We will now prove a lower bound for the density of the set produced by the greedy algorithm.
- For the lower bound we consider a specific assignment of the edges that we create as the greedy algorithm progresses:
  - When removing node u from S, assign all the edges to u
- So: $IN(u) = \text{degree of } u \text{ in } S \leq d(S) \leq d_g$
- This is true for all u so $\Delta \leq d_g$

- It follows that $d_{opt} \leq 2 \cdot d_g$

The k-densest subgraph

- The k-densest subgraph problem: Find the set of k nodes S, such that the density $d(S)$ is maximized.
  - The k-densest subgraph problem is NP-hard!