COT 6936: Topics in Algorithms

Giri Narasimhan
ECS 254A / EC 2443; Phone: x3748
giri@cs.fiu.edu
http://www.cs.fiu.edu/~giri/teach/COT6936_S12.html
https://moodle.cis.fiu.edu/v2.1/course/view.php?id=**174**

Polynomial-time algorithms for LP

- Simplex is not poly-time in the worst-case
- Khachiyan's ellipsoid algorithm: LP is in $\mathcal{P}$
- Karmarkar's interior-point algorithm
- Good implementations for LP exist
  - Works very well in practice
  - More competitive than the poly-time methods for LP

Integer Linear Programming

Integer Linear Programming

- LP with integral solutions
- NP-hard
- If A is a totally unimodular matrix, then the LP solution is always integral.
  - A TUM is a matrix for which every nonsingular submatrix has determinant 0, +1 or -1.
  - A TUM is a matrix for which every nonsingular submatrix has integral inverse.

Vertex Cover as an LP?

- For vertex v, create variable x_v
- For edge (u,v), create constraint $x_u + x_v \geq 1$
- Objective function: Σx_v
- Additional constraints: $x_v \leq 1$

- Doesn't work because x_v needs to be from {0,1}

Set Cover

- Given a universe of items $U = \{e_1, \ldots, e_n\}$ and a collection of subsets $S = \{S_1, \ldots, S_m\}$ such that each S_i is contained in U
- Find the minimum set of subsets from S that will cover all items in U (i.e., the union of these subsets must equal U)
- Weighted Set Cover: Given universe U and collection S, and a cost $c(S_i)$ for each subset S_i in S, find the minimum cost set cover

The Greedy Set Cover Algorithm

The Integer Linear Program (ILP)

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}} c(S) x_S \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1, \quad e \in U \\ & x_S \in \{0, 1\}, \quad S \in \mathcal{S} \end{aligned}$$

The LP Relaxation

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}} c(S) x_S \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1, \quad e \in U \\ & x_S \geq 0, \quad S \in \mathcal{S} \end{aligned}$$

The Dual LP

$$\begin{aligned} \max \quad & \sum_{e \in U} y_e \\ \text{subject to} \quad & \sum_{e : e \in S} y_e \leq c(S), \quad S \in \mathcal{S} \\ & y_e \geq 0, \quad e \in U \end{aligned}$$

Fractional may be better than integral

- $U = \{e, f, g\}$
- $S_1 = \{e, f\}$
- $S_2 = \{f, g\}$
- $S_3 = \{e, g\}$
- Optimal set cover = $\{S_1, S_2\}$
- Fractional optimal set cover assigns $\frac{1}{2}$ to each of three sets giving a total optimal value of 3/2.

The LP Relaxation

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}} c(S) x_S \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1, \quad e \in U \\ & x_S \geq 0, \qquad S \in \mathcal{S} \end{aligned}$$

The Dual LP Relaxation

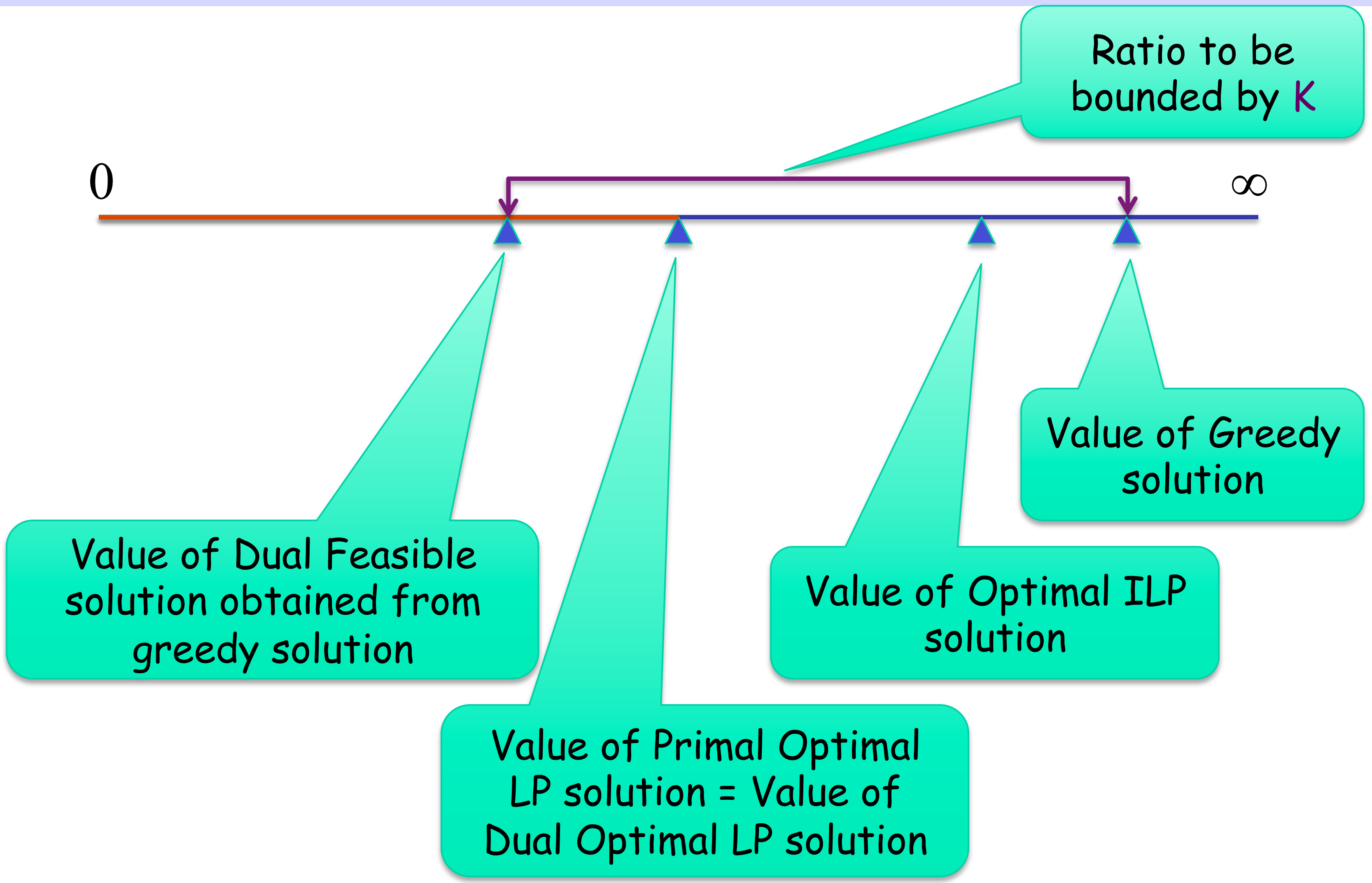
$$\begin{aligned} \max \quad & \sum_{e \in U} y_e \\ \text{subject to} \quad & \sum_{e : e \in S} y_e \leq c(S), \quad S \in \mathcal{S} \\ & y_e \geq 0, \qquad e \in U \end{aligned}$$

Weak Duality Principle

If $\bar{x}$ is primal feasible and $\bar{y}$ is dual feasible then

$$\boxed{\sum_{S \in \mathcal{S}} c(S) x_S \geq \sum_{e \in U} y_e}$$

K-Approximation Alg using Dual Fitting



Analysis of Greedy Weighted Set Cover

- In each iteration, greedy algorithm picks the set with the least price for each uncovered item.
- In iteration j, let S_j be the set picked covering m previously uncovered items. Let

$$\text{price}(e) = c(S_j)/m$$

be the price of each item e covered in this iteration.

- If $S_1, \ldots, S_k$ are sets chosen by greedy algorithm,

$$\begin{aligned}\text{Total Cost of Greedy Solution} &= \sum_{j=1}^{k} c(S_j) \\ &= \sum_{e \in U} price(e)\end{aligned}$$

Analysis of Greedy Set Cover

Let $\text{price}(e) = \frac{c(S_j)}{m}$

Consider the following dual variables:

$$y_e = \frac{\text{price}(e)}{H_n}$$

Claim: All dual constraints are satisfied.

$$\sum_{i=1}^{k} y_{e_i} \leq \frac{c(S)}{H_n} \cdot \left(\frac{1}{k} + \frac{1}{k-1} + \ldots + \frac{1}{1}\right) = \frac{H_k}{H_n} c(S) \leq c(S)$$

Thus $(y_{e_1}, \ldots, y_{e_n})$ gives us a dual feasible point.

$$\sum_{e \in U} price(e) = H_n \left(\sum_{e \in U} y_e\right) \leq H_n \cdot \text{OPT}_f \leq H_n \cdot \text{OPT}$$

Rounding Algorithm for Set Cover

- Algorithm
  - Find an optimal solution to the LP Relaxation
  - Pick all sets S for which $x_S \geq 1/f$ in this solution
    - f = frequency of most frequent item

- Analysis
  - Is the resulting solution a valid set cover?
  - How good is the solution? How close is to the optimal set cover?

Analysis of Rounding Algorithm

Let $\mathcal{C}$ = sets picked by **Rounding Algorithm**.

Claim 1: $\mathcal{C}$ is a valid set cover. Arbitrary item e appears in at most f sets. At least one of these sets is assigned value $1/f$. Thus, e will get picked.

Claim 2: The rounding algorithm is f-approximate. Rounding increases the value of each set by a factor of at most f.

COT 6936: Topics in Algorithms

Randomized Algorithms

Randomization

- Randomized Algorithms: Uses values generated by random number generator to decide next step
- Often easier to implement and/or more efficient
- Applications
  - Used in protocol in "Ethernet Cards" to decide when it next tries to access the shared medium
  - Primality testing & cryptography
  - Monte Carlo simulations

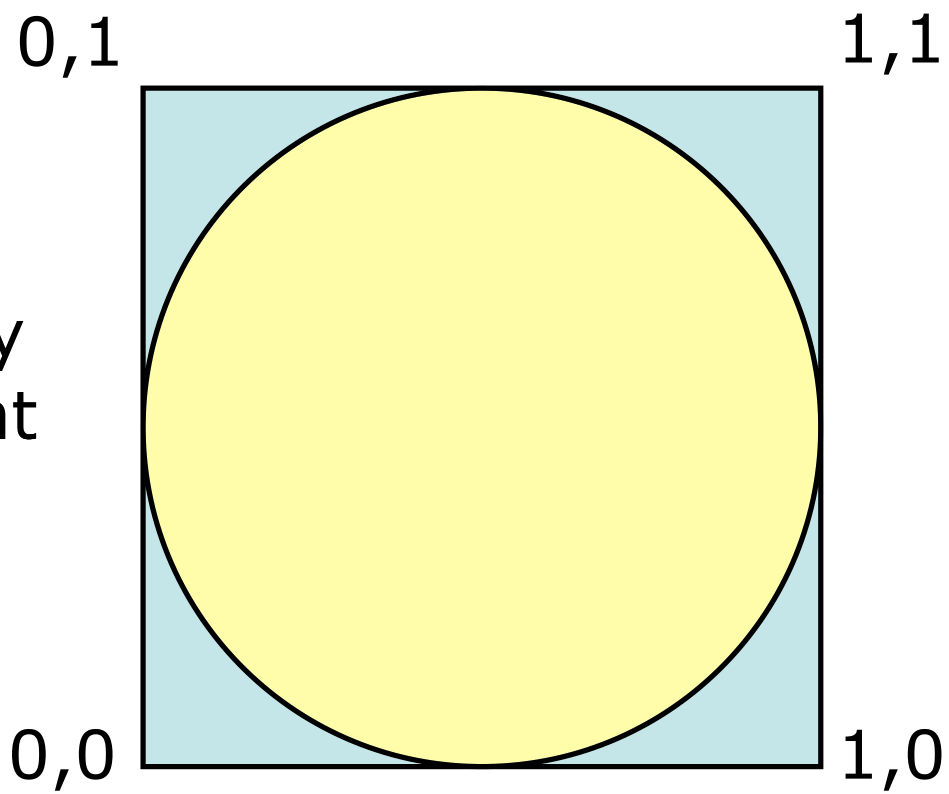
Monte Carlo Simulations

Slide by David Evans

Determining π

Square = 1
Circle = π/4

The probability a random point in square is in circle:
= π/4



π = 4 * points in circle/points

QuickSort vs Randomized QuickSort

QuickSort

- Pick a fixed pivot
- Partition input based on pivot into two sets
- Recursively sort the two partitions

Randomized QuickSort

- Pick a random pivot
- Partition input based on pivot into two sets
- Recursively sort the two partitions

QuickSort: Probabilistic Analysis

- Expected rank of pivot = n/2 (Why?)
- Thus expected size of sublists after partition = n/2
- Hence the recurrence $T(n) = 2T(n/2) + O(n)$
- Average time complexity = $T(n) = O(n \log n)$

New Quicksort: Randomized Analysis

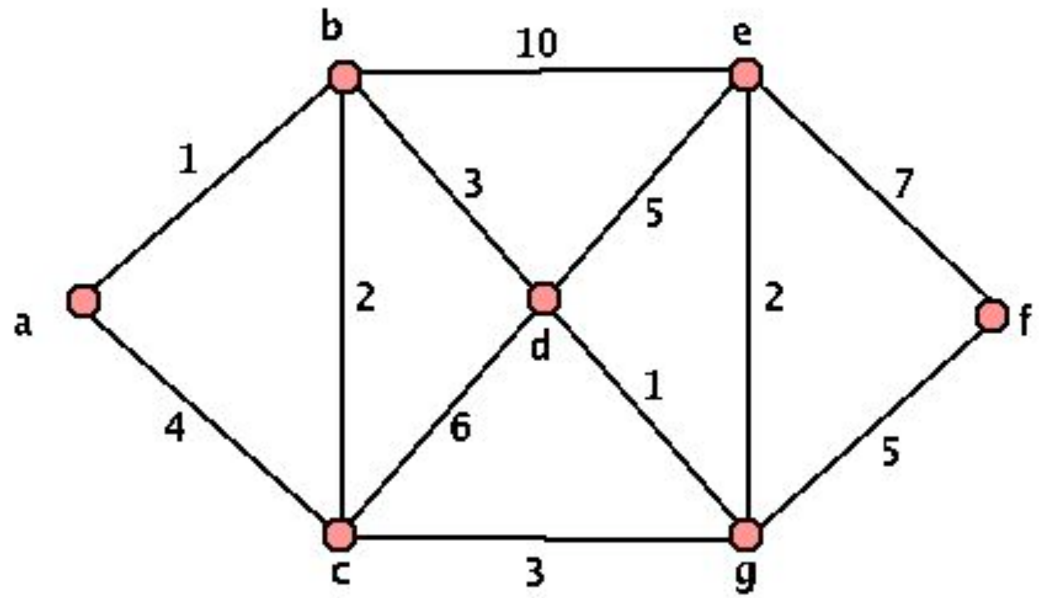
- Let X_{ij} be a random variable representing the number of times items i and j are compared by the algorithm.
- Expected time complexity = expected value of sum of all random variables X_{ij}.
- $Pr(X_{ij} = 1) = 2/(j - i + 1)$ (Why?)
- $T(n) = ?$

New Quicksort: Randomized Analysis

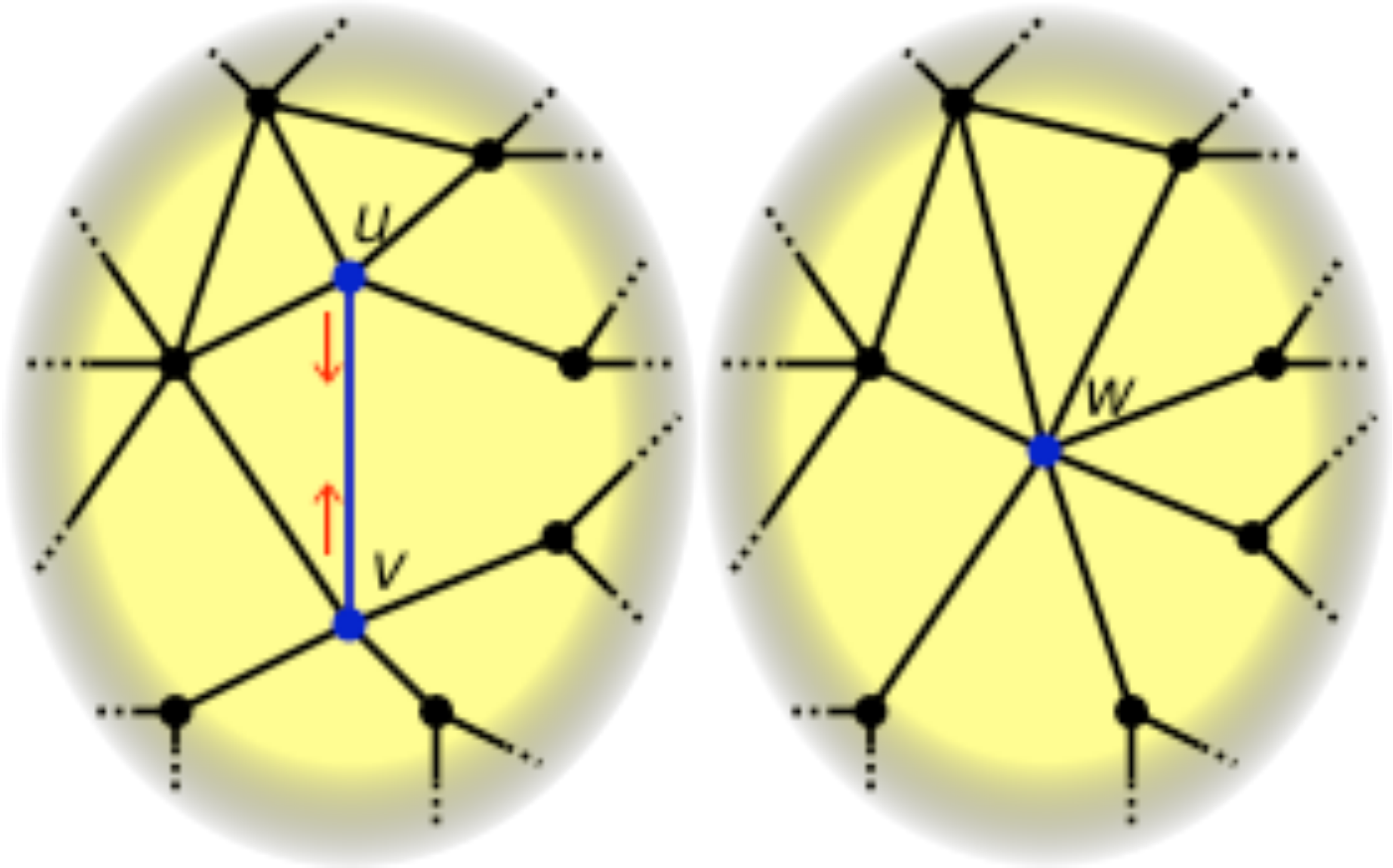
$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} \\
&= \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k)\frac{2}{k} \\
&= \left((n+1) \sum_{k=2}^{n} \frac{2}{k}\right) - 2(n-1) \\
&= (2n+2) \sum_{k=1}^{n} \frac{1}{k} - 4n \\
&= \boxed{2n \ln n + \Theta(n)}
\end{aligned}
$$

Cut-Sets & Min-Cuts

- Example 1: ({a,b,c,d}, {e,f,g})
  - Weight = 19
- Example 2: ({a,b,g}, {c,d,e,f})
  - Weight = 30
- Example 3: ({a}, {b,c,d,e,f,g})
  - Weight = 5



Edge Contraction

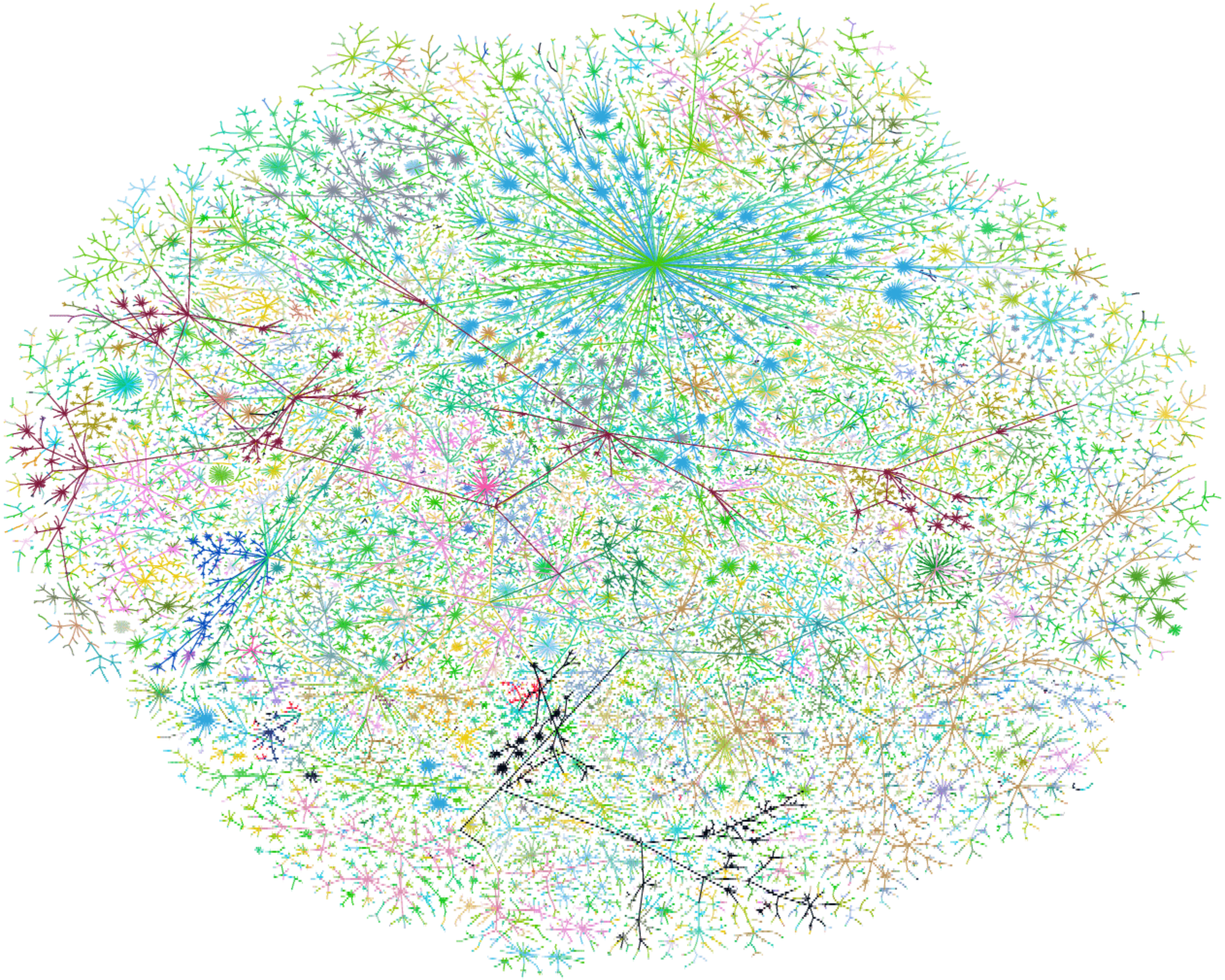


http://en.wikipedia.org/wiki/Edge_contraction

Edge Contractions and Min-Cuts

- Lemma: If you are not contracting an edge from the cut-set, edge contractions do not affect the size of min-cuts.
- Observation: Most edges are not part of the min-cut.
- Idea: Use randomization

Min-Cuts in the Internet Graph



June 1999 Internet graph, Bill Cheswick
http://research.lumeta.com/ches/map/gallery/index.html

Randomized Algorithms: Min-Cut

- Algorithm:
  - Pick a random edge and contract it until only 2 vertices are remaining.
  - Report edges connecting the 2 remaining vertices as the min cut
- Analysis
  - Assume that the Min-cut is of size k
  - Prob {edge is not in Min-cut} $\geq 1 - 2/n$ (why?)
  - Prob {Min-cut is output} $\geq 2/n(n - 1)$ (why?)

Analysis: Min-Cut Algorithm (Cont'd)

- Observation:
  - If Min-Cut is of size k, then minimum degree of every vertex is k. (Why?)
- Number of edges in graph ≥ kn/2
- Probability that <u>an</u> edge from Min-Cut is picked in iteration 1 is ≤ 2/n
- Probability that <u>no</u> edge from Min-Cut is picked in iteration 1 is ≥ 1 - 2/n
- Iteration i?

Analysis: Min-Cut Algorithm (Cont'd)

- E_i = Event that no edge from Min-Cut is picked in iteration i
- F_i = Event that no edge from Min-Cut is picked in iteration 1 through i

$$Pr(E_i | F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}.$$

- Need F_{n-2}!

Analysis: Min-Cut Algorithm (Cont'd)

$$
\begin{aligned}
Pr(F_{n-2}) &= Pr(E_{n-2} \cap F_{n-3}) = Pr(E_{n-2}|F_{n-3})Pr(F_{n-3}) \\
&= Pr(E_{n-2}|F_{n-3}) \cdot Pr(E_{n-3}|F_{n-4}) \ldots Pr(E_2|F_1)Pr(F_1) \\
&\geq \Pi_{i=1}^{n-2}\left(1 - \frac{2}{n-i+1}\right) = \Pi_{i=1}^{n-2}\frac{n-i-1}{n-I+1} \\
&= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\cdots\frac{4}{6}\frac{3}{5}\frac{2}{4}\frac{1}{3} \\
&= \frac{2}{n(n-1)}.
\end{aligned}
$$

Analysis: Min-Cut Algorithm (Cont'd)

- Probability of contracting only edges not from Min-Cut, i.e., ending up with exactly the Min-Cut ≥ 2/n(n-1)
  - Rather low!
- Repeat the algorithm many times.
  - How many times?
  - Goal: repeat until prob of error is very small

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\ln n} \leq e^{-2\ln n} = \frac{1}{n^2}$$