Outline

• coping with intractibility
  – approximation algorithms
    • set cover
    • TSP
    • center selection

• randomness in algorithms

Optimization Problems

• many hard problems (especially \textbf{NP}-hard) are optimization problems
  – e.g. find shortest TSP tour
  – e.g. find smallest vertex cover
  – e.g. find largest clique

– may be minimization or maximization problem
– “OPT” = value of optimal solution

Approximation Algorithms

• often happy with \textit{approximately optimal} solution
  – warning: lots of heuristics
  – we want \textit{approximation algorithm} with guaranteed approximation ratio of \( r \)
  – meaning: on every input \( x \), output is guaranteed to have value
    at most \( r \times \text{opt} \) for minimization
    at least \( \text{opt} / r \) for maximization

Set Cover

• Given subsets \( S_1, S_2, \ldots, S_n \) of a universe \( U \) of size \( m \), and an integer \( k \)
  – is there a cover \( J \) of size \( k \)
  – “cover”: \( \bigcup_{j \in J} S_j = U \)

\textbf{Theorem:} set-cover is \textbf{NP}-complete
  – in \textbf{NP} (why?)
  – reduce from vertex cover (how?)

\textbf{Theorem:} greedy set cover algorithm achieves an approximation ratio of \( \left( \ln m + 1 \right) \)
Set cover

**Theorem:** greedy set cover algorithm achieves an approximation ratio of \((\ln m + 1)\)

**Proof:**
- let \(r_i\) be # of items remaining after iteration \(i\)
- \(r_0 = |U| = m\)
- Claim: \(r_i \leq (1 - 1/OPT)r_{i-1}\)
  - proof: OPT sets cover all remaining items so some set covers at least 1/OPT fraction

May 29, 2014  CS38 Lecture 18  7

TSP approximation algorithm

**Theorem:** this approximation algorithm achieves approximation ratio 2

**Proof:**
- optimal tour includes a MST, so \(wt(T) \leq OPT\)
- tour we output has weight at most 2 \(wt(T)\)

May 29, 2014  CS38 Lecture 18  11
Christofide’s algorithm

- Second approximation algorithm:
  - find a Minimum Spanning Tree \( T \)
  - even number of odd-degree vertices (why?)
  - find a min-weight matching \( M \) on these
  - output an Euler tour on \( M \cup T \) (with short-circuiting)

**Theorem:** this approximation algorithm achieves approximation ratio 1.5

---

**Theorem:** this approximation algorithm achieves approximation ratio 1.5

**Proof:**
- as before \( \text{OPT} \geq \text{wt}(T) \)
- let \( R \) be opt. tour on odd deg. vertices \( W \) only
- even/odd edges of \( R \) both constitute perfect matchings on \( W \)
- thus \( \text{wt}(M) \leq \text{wt}(R)/2 \leq \text{OPT}/2 \)
- total: \( \text{wt}(M) + \text{wt}(T) \leq 1.5 \cdot \text{OPT} \)

---

**Center selection problem**

**Input.** Set of \( n \) sites \( s_1, \ldots, s_n \) and an integer \( k > 0 \).

**Center selection problem.** Select set of \( k \) centers \( C \) so that maximum distance \( r(C) \) from a site to nearest center is minimized.

**Notation.**
- \( \text{dist}(x, y) = \) distance between sites \( x \) and \( y \).
- \( \text{dist}(s_i, C) = \min_{c \in C} \text{dist}(s_i, c) = \) distance from \( s_i \) to closest center.
- \( r(C) = \max \text{dist}(s, C) = \) smallest covering radius.

**Goal.** Find set of centers \( C \) that minimizes \( r(C) \), subject to \( |C| = k \).

**Distance function properties.**
- \( \text{dist}(x, x) = 0 \) [identity]
- \( \text{dist}(x, y) = \text{dist}(y, x) \) [symmetry]
- \( \text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y) \) [triangle inequality]

---

**Greedy algorithm:** a false start

**Greedy algorithm.** Put the first center at the best possible location for a single center, and then keep adding centers so as to reduce the covering radius each time by as much as possible.

**Remark:** arbitrarily bad!
Center selection: greedy algorithm

Repeatedly choose next center to be site farthest from any existing center.

**Algorithm**

```
CENTER-SELECTION (K, n, s1, s2, ..., sn)
C ← Ø
REPEAT k times
    Select a site s with maximum distance dist(s, C).
    C ← C ∪ {s}
RETURN C
```

Property. Upon termination, all centers in C are pairwise at least \( r(C) \) apart.

Proof. By construction of algorithm.

Center selection: analysis of greedy algorithm

**Theorem.** Let \( C^* \) be an optimal set of centers. Then \( r(C) \leq 2r(C^*) \).

**Proof.** (by contradiction) Assume \( r(C) < \frac{3}{2} r(C^*) \).

- For each site \( c \in C \), consider ball of radius \( \frac{1}{2} r(C) \) around it.
- Exactly one \( c^* \) in each ball; let \( c \) be the site paired with \( c^* \).
- Consider any site \( s \) and its closest center \( c^* \in C^* \).
- \( dist(s, C) \leq dist(s, c) \leq dist(s, c^*) + dist(c^*, C) \leq 2r(C^*) \).
- Thus, \( r(C) \leq 2r(C^*) \).

Randomness in algorithms

May 29, 2014

Randomization

**Algorithmic design patterns.**

- Greedy.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Randomization.

**Randomization.** Allow fair coin flip in unit time.

**Why randomize?** Can lead to simplest, fastest, or only known algorithm for a particular problem.

**Ex.** Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.
Contestion resolution in a distributed system

Contestion resolution. Given $n$ processes $P_1, ... , P_n$, each competing for access to a shared database. If two or more processes access the database simultaneously, all processes are locked out. Devise protocol to ensure all processes get through on a regular basis.

Restriction. Processes can't communicate.

Challenge. Need symmetry-breaking paradigm.

Contention Resolution: randomized protocol

Claim. The probability that process $i$ fails to access the database in $en$ rounds is at most $1/n$. After $o(n \log n)$ rounds, the probability is $n^{-e}$.

Proof. Let $P_i[t]$ be the event that process $i$ fails to access the database in rounds $1$ through $t$. By independence and previous claim, we have

\[ Pr[P_i[t]] \leq (1 - 1/n)^t. \]

* Choose $t = \epsilon n \ln n$:

\[ Pr[P_i[t]] \leq (1 - 1/n)^{\epsilon n \ln n} = \frac{1}{n^{\epsilon}}. \]

* Choose $t = \lceil \frac{\epsilon n \ln n}{c} \rceil$:

\[ Pr[P_i[t]] \leq \left( \frac{1}{n^c} \right)^{\epsilon n \ln n} = n^{-1}. \]

Global minimum cut

Global min cut. Given a connected, undirected graph $G = (V, E)$, find a cut $(A, B)$ of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.

* Replace every edge $(u, v)$ with two antiparallel edges $(u, v)$ and $(v, u)$.

* Pick some vertex $s$ and compute min $s-v$ cut separating $s$ from each other vertex $v \in V$.

False intuition. Global min-cut is harder than min $s-t$ cut.
Contraction algorithm

[Contraction algorithm. [Karger 1995]]

- Pick an edge $e = (u, v)$ uniformly at random.
- Contract edge $e$.
  - replace $u$ and $v$ by single new super-node $w$
  - preserve edges, updating endpoints of $u$ and $v$ to $w$
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes $v_1$ and $v_1'$
- Return the cut (all nodes that were contracted to form $v_1$).

Reference: Thore Husfeldt

Claim. The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

Pf. Consider a global min-cut $(A^*, B^*)$ of $G$.

- Let $F^*$ be edges with one endpoint in $A^*$ and the other in $B^*$.
- Let $k = |F^*| = $ size of min cut.
- Let $G'$ be graph after $j$ iterations. There are $n' = n - j$ supernodes.
- Suppose no edge in $F^*$ has been contracted. The min-cut in $G'$ is still $k$.
- Since value of min cut is $k$, $|E'| \geq 1/2 k n'$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq 2/n$.

Amplification. To amplify the probability of success, run the contraction algorithm many times.

Claim. If we repeat the contraction algorithm $n^2 \ln n$ times, then the probability of failing to find the global min cut is $\leq 1/n^2$.

Pf. By independence, the probability of failure is at most

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

with independent random choices.

Contraction algorithm: example execution

References: [Thore Husfeldt]
Global min cut: context

**Remark.** Overall running time is slow since we perform $O(n^2 \log n)$ iterations and each takes $O(m)$ time.

**Improvement.** [Karger-Stein 1996] $O(n^2 \log^3 n)$.
- Early iterations are less risky than later ones: probability of contracting an edge in min-cut hits 50% when $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph and return best of two cuts.

**Extensions.** Naturally generalizes to handle positive weights.

**Best known.** [Karger 2000] $O(m \log^7 n)$.

\[ \text{faster than best known max flow algorithm or deterministic global min cut algorithm} \]

37