CS38
Introduction to Algorithms

Lecture 18

May 29, 2014

Outline

- coping with intractibility
  - approximation algorithms
    - set cover
    - TSP
    - center selection
- randomness in algorithms

Optimization Problems

- many hard problems (especially **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 approximately optimal solution
  - warning: lots of heuristics
  - we want approximation algorithm with guaranteed approximation ratio of r
  - meaning: on every input x, output is guaranteed to have value

    at most r*opt for minimization

    at least 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": $\cup_{j \in J} S_j = U$

<u>Theorem:</u> set-cover is NP-complete
  - in NP (why?)
  - reduce from vertex cover (how?)

Set cover

- Greedy approximation algorithm:
  - at each step, pick set covering largest number of remaining uncovered items

<u>Theorem</u>: greedy set cover algorithm achieves an approximation ratio of (ln m + 1)

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

Set cover

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

Proof:
- Claim: $r_i \leq (1 - 1/OPT)r_{i-1}$
- so $r_i \leq (1 - 1/OPT)^i\ m$
- after OPT·ln m + 1 iterations, # remaining elements is at most m/(2m) $\leq$ ½
- so must have covered all m elements.

$(1-1/x)^x \leq 1/e$

Travelling Salesperson Problem

- given a complete graph and edge weights satisfying the triangle inequality

  $w_{a,b} + w_{b,c} \geq w_{a,c}$ for all vertices a,b,c
  - find a shortest tour that visits every vertex

Theorem: TSP with triangle inequality is NP-complete
- in NP (why?)
- reduce from Hamilton cycle (how?)

TSP approximation algorithm

- two key observations:
  - tour that visits vertices more than once can be short-circuited without increasing cost, by triangle inequality
    - short-circuit = skip already-visited vertices
  - (multi-)graph with all even degrees has Eulerian tour: a tour that uses all edges
    - proof?

TSP approximation algorithm

- First approximation algorithm:
  - find a Minimum Spanning Tree T
  - double all the edges
  - output an Euler tour (with short-circuiting)

Theorem: this approximation algorithm achieves approximation ratio 2

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)

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

Christofide's algorithm

Theorem: this approximation algorithm achieves approximation ratio 1.5

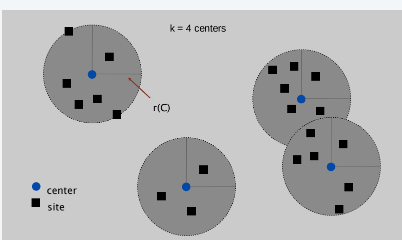
Proof:

- as before OPT $\geq$ 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 wt(M) $\leq$ wt(R)/2 $\leq$ OPT/2
- total: wt(M) + wt(T) $\leq$ 1.5·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.



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.

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

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

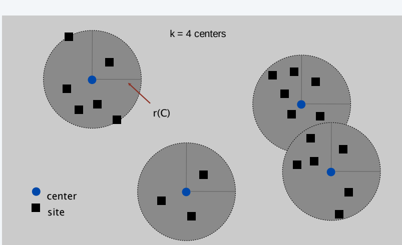
Distance function properties.

- $dist(x, x) = 0$ [identity]
- $dist(x, y) = dist(y, x)$ [symmetry]
- $dist(x, y) \leq dist(x, z) + dist(z, y)$ [triangle inequality]

Center selection example

Ex: each site is a point in the plane, a center can be any point in the plane, $dist(x, y)$ = Euclidean distance.

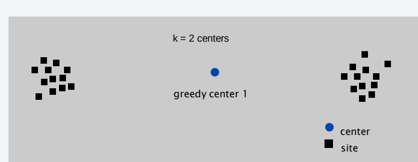
Remark: search can be infinite!



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.

GREEDY-CENTER-SELECTION (k, n, s_1, s_2, … , s_n)

$C \leftarrow \emptyset$.

REPEAT k times

  Select a site s_i with maximum distance $dist(s_i, C)$.

  $C \leftarrow C \cup s_i$.

RETURN C.

site farthest from any center

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

Pf. By construction of algorithm.

Center selection: analysis of greedy algorithm

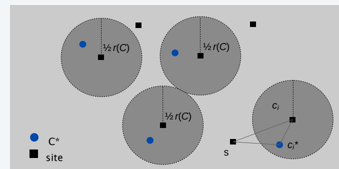
Theorem. Let C^* be an optimal set of centers. Then $r(C) \leq 2r(C^*)$.

Pf. [by contradiction] Assume $r(C^*) < ½\, r(C)$.

- For each site $c_i \in C$, consider ball of radius $½\, r(C)$ around it.
- Exactly one c_i^* in each ball; let c_i be the site paired with c_i^*.
- Consider any site s and its closest center $c_i^* \in C^*$.
- $dist(s, C) \leq dist(s, c_i) \leq dist(s, c_i^*) + dist(c_i^*, c_i) \leq 2r(C^*)$.
- Thus, $r(C) \leq 2r(C^*)$. ▪

Δ-inequality

≤ $r(C^*)$ since c_i^* is closest center



Center selection

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

Theorem. Greedy algorithm is a 2-approximation for center selection problem.

Remark. Greedy algorithm always places centers at sites, but is still within a factor of 2 of best solution that is allowed to place centers anywhere.

e.g., points in the plane

Question. Is there hope of a 3/2-approximation? 4/3?

Randomness in algorithms

Randomization

Algorithmic design patterns.

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

in practice, access to a pseudo-random number generator

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.

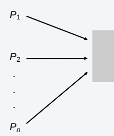
Contention resolution

Contention resolution in a distributed system

Contention resolution. Given n processes $P_1, \ldots, P_n$, 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

Protocol. Each process requests access to the database at time t with probability $p = 1/n$.

Claim. Let $S[i, t]$ = event that process i succeeds in accessing the database at time t. Then $1/(e \cdot n) \le \Pr[S(i, t)] \le 1/(2n)$.

Pf. By independence, $\Pr[S(i, t)] = p\,(1 - p)^{n-1}$.

process i requests access — none of remaining n-1 processes request access

- Setting $p = 1/n$, we have $\Pr[S(i, t)] = 1/n\,(1 - 1/n)^{n-1}$. ▪

value that maximizes Pr[S(i, t)] — between 1/e and 1/2

Useful facts from calculus. As n increases from 2, the function:
- $(1 - 1/n)^n$ converges monotonically from 1/4 up to $1/e$.
- $(1 - 1/n)^{n-1}$ converges monotonically from 1/2 down to $1/e$.

Contention Resolution: randomized protocol

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

Pf. Let $F[i, t]$ = event that process i fails to access database in rounds 1 through t. By independence and previous claim, we have $\Pr[F[i, t]] \le (1 - 1/(en))^t$.

- Choose $t = \lceil e \cdot n \rceil$: $\Pr[F(i,t)] \le \left(1 - \frac{1}{en}\right)^{\lceil en \rceil} \le \left(1 - \frac{1}{en}\right)^{en} \le \frac{1}{e}$
- Choose $t = \lceil e \cdot n \rceil \lceil c \ln n \rceil$: $\Pr[F(i,t)] \le \left(\frac{1}{e}\right)^{c \ln n} = n^{-c}$

Contention Resolution: randomized protocol

Claim. The probability that all processes succeed within $2e \cdot n \ln n$ rounds is $\ge 1 - 1/n$.

Pf. Let $F[t]$ = event that at least one of the n processes fails to access database in any of the rounds 1 through t.

$$\Pr[F[t]] = \Pr\left[\bigcup_{i=1}^{n} F[i,t]\right] \le \sum_{i=1}^{n} \Pr[F[i,t]] \le n\left(1 - \frac{1}{en}\right)^t$$

union bound — previous slide

- Choosing $t = 2 \lceil en \rceil \lceil c \ln n \rceil$ yields $\Pr[F[t]] \le n \cdot n^{-2} = 1/n$. ▪

Union bound. Given events $E_1, \ldots, E_n$, $\Pr\left[\bigcup_{i=1}^{n} E_i\right] \le \sum_{i=1}^{n} \Pr[E_i]$

Global min cut

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

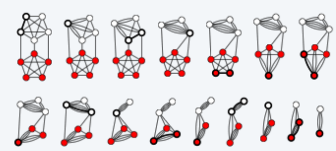
Contraction algorithm. [Karger 1995]

- 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).

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).



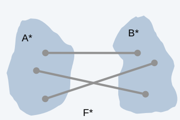
Reference: Thore Husfeldt

Contraction algorithm

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.
- In first step, algorithm contracts an edge in F^* probability $k / |E|$.
- Every node has degree $\geq k$ since otherwise (A^*, B^*) would not be a min-cut ® $|E| \geq ½ k n$.
- Thus, algorithm contracts an edge in F^* with probability $\leq 2 / n$.



Contraction algorithm

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, $|E'| \geq ½ k n'$.
- Thus, algorithm contracts an edge in F^* with probability $\leq 2 / n'$.
- Let E_j = event that an edge in F^* is not contracted in iteration j.

$$\begin{aligned}
\Pr[E_1 \cap E_2 \text{ L } \cap E_{n-2}] &= \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \text{ L } \times \Pr[E_{n-2} \mid E_1 \cap E_2 \text{ L } \cap E_{n-3}] \\
&\geq \left(1-\tfrac{2}{n}\right)\left(1-\tfrac{2}{n-1}\right) \text{ L } \left(1-\tfrac{2}{4}\right)\left(1-\tfrac{2}{3}\right) \\
&= \left(\tfrac{n-2}{n}\right)\left(\tfrac{n-3}{n-1}\right) \text{ L } \left(\tfrac{2}{4}\right)\left(\tfrac{1}{3}\right) \\
&= \tfrac{2}{n(n-1)} \\
&\geq \tfrac{2}{n^2}
\end{aligned}$$

Contraction algorithm

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

with independent random choices,

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

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

$(1 - 1/x)^x \leq 1/e$

Contraction algorithm: example execution

trial 1

trial 2

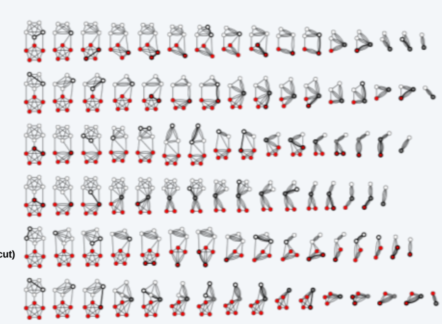
trial 3

trial 4

trial 5
(finds min cut)

trial 6

...



Reference: Thore Husfeldt

Global min cut: context

Remark. Overall running time is slow since we perform $\Theta(n^2 \log n)$ iterations and each takes $\Omega(m)$ 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] $O(m \log^3 n)$.

faster than best known max flow algorithm or deterministic global min cut algorithm

37