Randomized Algorithms II

Geometric & Combinatorial Optimization

Pat Morin

Carleton University

University of Sydney

National ICT Australia

Geometric Optimization Problems

Examples:

- Given a finite point set $S \subset \mathbb{R}^2$
  - $\max\{|pq| : p,q \in S\}$ — diameter
  - $\min\{|pq| : p,q \in S, p \neq q\}$ — closest pair
  - smallest ball that contains S — smallest enclosing ball
  - smallest square that contains k points of S — smallest k-cluster

Geometric Decision Problems

Examples:

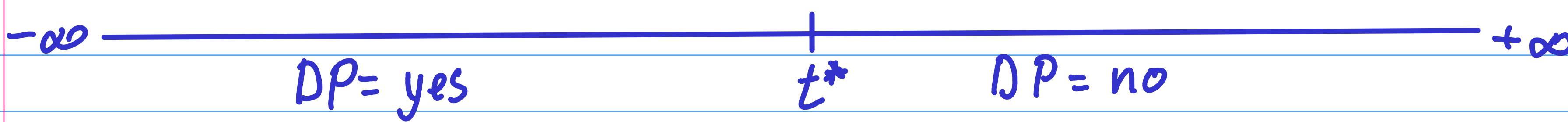
- Given a point set S and a real value t, does there exist

- $p, q \in S$ such that $|pq| \geq t$ — diameter
- $p, q \in S$ such that $|pq| \leq t$ — closest pair
- a ball of radius t that contains S — smallest enclosing ball
- a square of side-length t that contains k points of S — smallest k-cluster

Decision Algorithm to Optimization Algorithm

Decision problems are often easier to solve than the corresponding optimization problem

Can we use an algorithm for the decision problem to solve the optimization problem?



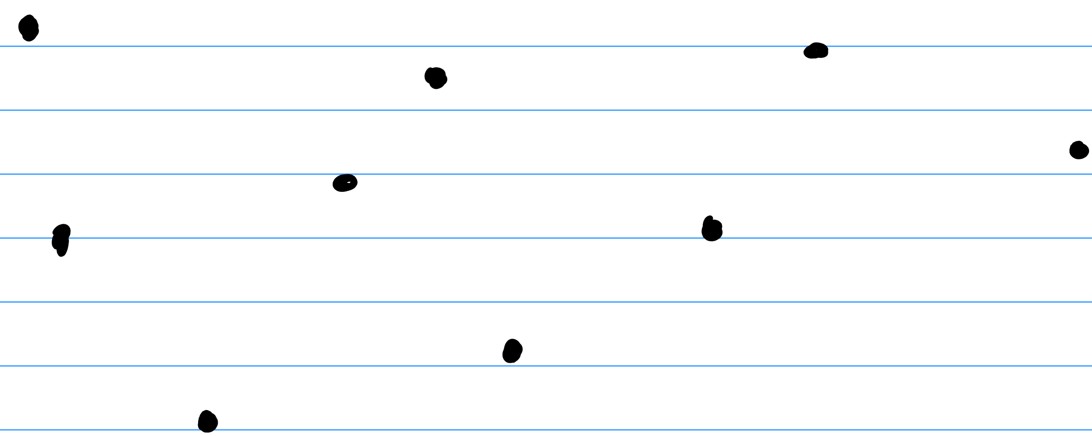
Binary search on t will not give an exact solution

(t^* is a real number)

Example: Planar Closest Pair

- $S \subseteq \mathbb{R}^2$, $|S| = n$
- $t \in \mathbb{R}$, $t > 0$

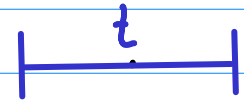
Does there exist $p, q \in S$ such that $|pq| \leq t$?



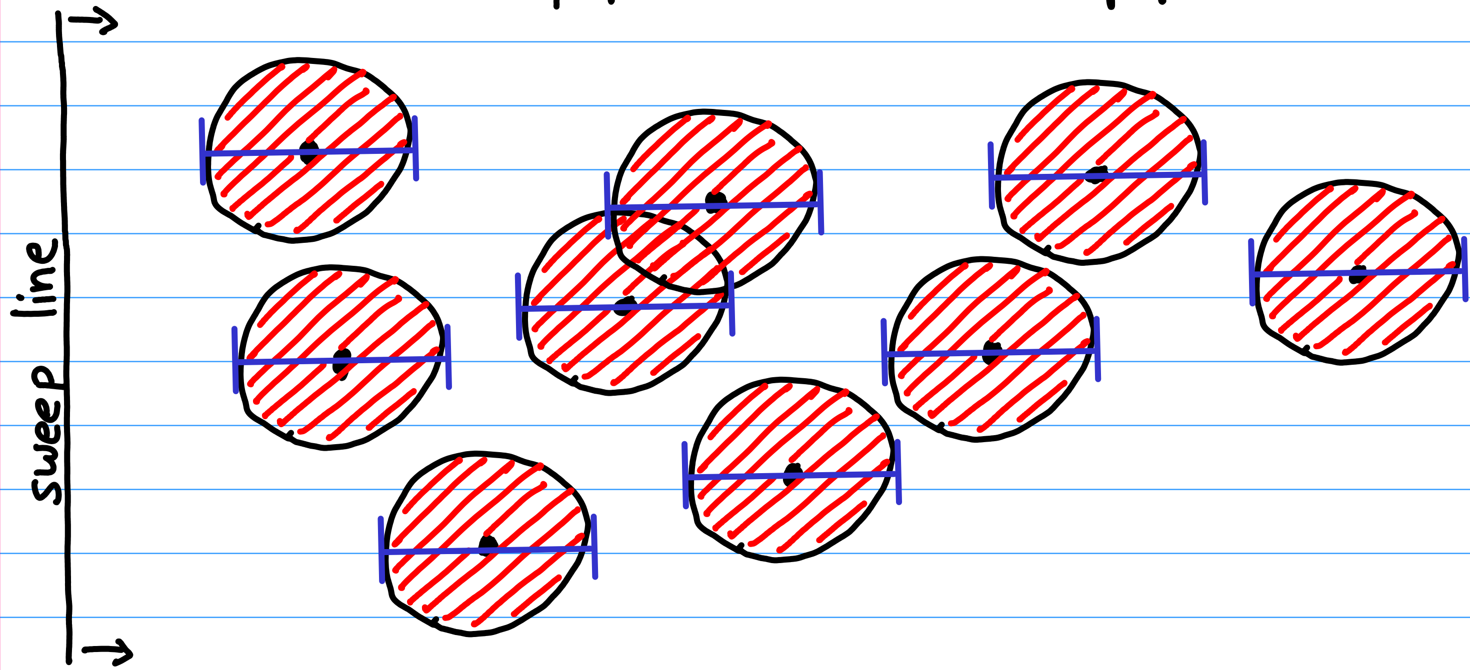
Note: There exists a trivial $O(n^2)$ time algorithm that also solves the optimization problem.

Example: Planar Closest Pair

- $S \subseteq \mathbb{R}^2$, $|S| = n$
- $t \in \mathbb{R}$, $t > 0$



Does there exist $p, q \in S$ such that $|pq| \leq t$?



Can be solved in $O(n \log n)$ time using a plane sweep algorithm (using Quicksort and a treap)

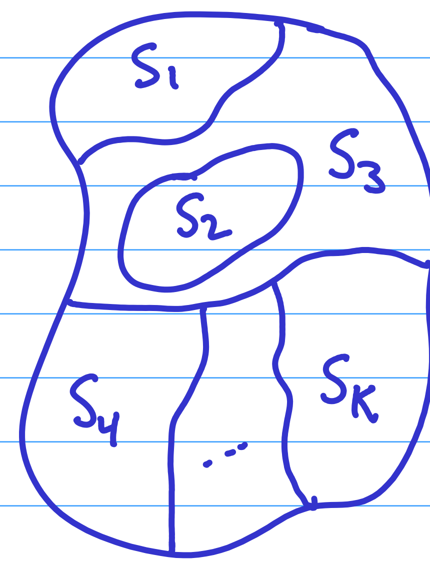
Decision to Optimization

Theorem: There exists an $O(n \log n)$ time algorithm to solve the closest pair decision problem.

Is there a fast algorithm for the optimization problem?

Divide-and-Conquer (Recursion)

- If $|S| \le 2$ then the problem is easy.
- Otherwise, partition S into K sets $S_1, \dots, S_K$ with $|S_i| = n/K$.
- Recursively solve for each $1 \le i < j \le K$ in the set $S_i \cup S_j$
- Return the closest pair found among all $\binom{K}{2}$ subproblems



Divide-and-Conquer

- If $|S| = 2$ then the problem is easy to solve
- Otherwise, partition S into k sets $S_1, \ldots, S_k$ with $|S_i| = n/k$
- Recursively solve for each $1 \leq i < j \leq k$ in the set $S_i \cup S_j$
- Return the closest pair found among all $\binom{k}{2}$ subproblems

Running Time: $T(n) = O(n) + \binom{k}{2} T(2n/k)$

$$= O\left(n^{\underbrace{(\log k + \log(k-1) - 1)/(\log k - 1)}_{\geq 2}} \right)$$

This is worse than the trivial algorithm!

Divide-and-Conquer (More carefully)

ClosestPair(S)

  if $|S| \le 20$ then return $\min\{|pq| : p, q \in S,\ p \ne q\}$

  partition S into $S_1, \dots, S_K$

  $X = \{S_i \cup S_j : 1 \le i < j \le K\}$

  $m \leftarrow \infty$

  for each subproblem $Y \in X$ in random order

    if ClosestPairDecision(Y, m) $<$ [illegible] then

      $m \leftarrow$ ClosestPair(Y)

  return m

Divide-and-Conquer (More carefully)

```
ClosestPair(S)
   if |S| ≤ 20 then return min{|pq| : p,q ∈ S, p ≠ q}
   partition S into S_1,...,S_K
   X = {S_i ∪ S_j : 1 ≤ i < j ≤ K}
   m ← ∞
   for each subproblem Y ∈ X in random order
        if ClosestPairDecision(Y) < m then
            m ← ClosestPair(Y)
   return m
```

look familiar?

Finding the Minimum

```
FindMin(A_1,..., A_n)
m ← ∞
for i=1 to n do
    if A_i < m then
        m ← A_i
return m
```

Flashback.

$$E\left[\sum_{i=1}^{n} I_i\right] = \sum_{i=1}^{n} E[I_i] = \sum_{i=1}^{n} 1/i \stackrel{\text{def}}{=} H_n$$

"nth harmonic number"

Divide-and-Conquer (More carefully)

ClosestPair(S)
  if $|S| \leq 20$ then return $\min\{|pq| : p,q \in S,\ p \neq q\}$
  partition S into $S_1, \ldots, S_K$
  $X = \{S_i \cup S_j : 1 \leq i < j \leq K\}$
  $m \leftarrow \infty$
  for each subproblem $Y \in X$ in random order
    if ClosestPairDecision(Y) $< m$ then
      $m \leftarrow$ ClosestPair(Y)
  return m

$$T(n) = O\left(\binom{K}{2} \cdot n \log n\right) + H_{\binom{K}{2}} \cdot T(2n/K)$$

$$= O(n \log n), \quad \text{for } K=5.$$

Chan's Optimization Theorem

$0 < \alpha < 1$
$r > 1$
constants

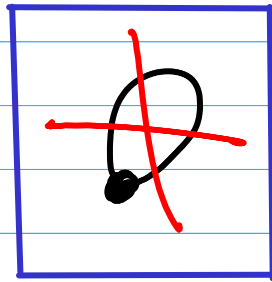
Theorem: Let P be any maximization problem such that

1. Given an input S of size $O(1)$ we can compute $val(S)$ in $O(1)$ time
2. Given an input S of size n we can test if $val(S) \geq t$ in $D(n)$ time, for any real number t.
3. Given an input S of size n, we can partition S into subproblems $S_1, \ldots, S_r$ each of size at most αn and such that

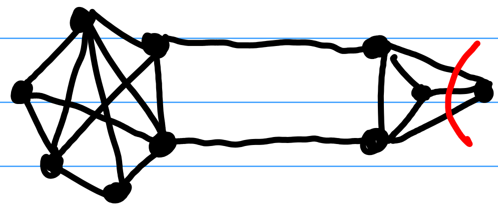
$$val(S) = \min\{val(S_i) : i \in \{1, \ldots, r\}\}$$

Then we can compute $val(S)$ in $O(D(n))$ expected time.

Combinatorial Optimization: Min-Cut



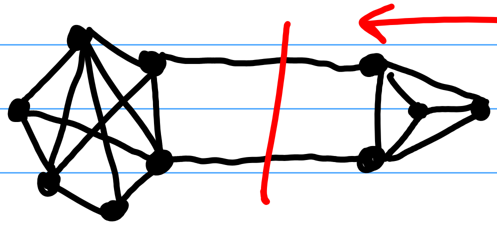
- $G=(V,E)$ is an undirected graph with no self loops
- A cut of G is a set of edges whose removal disconnects G.



← cut of size 3

- A min-cut is a cut of minimum cardinality

← cut of size 2 (min cut)



Min-Cut Background

Best deterministic MinCut algorithms run in $O(n^3)$ time

- complicated
- difficult to implement
- difficult to understand

We want something

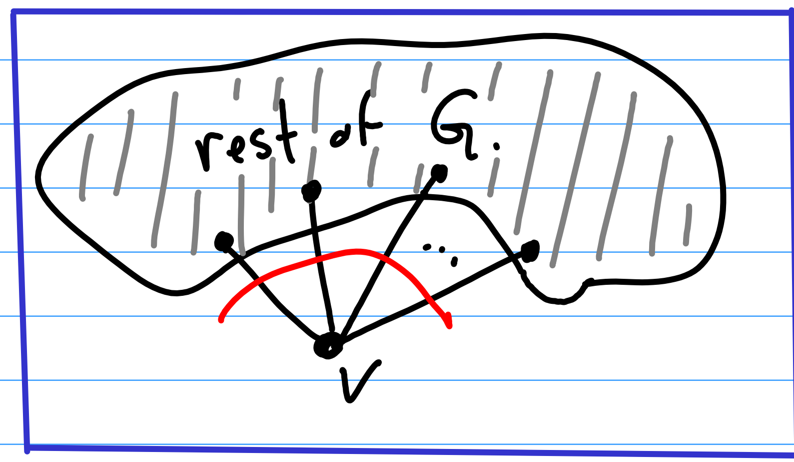
- simple
- easy to implement
- easy to understand (the algorithm, not necessarily the analysis)

How Big Can a MinCut Be?

Let C be any min-cut of a graph G with n vertices and m edges. Then

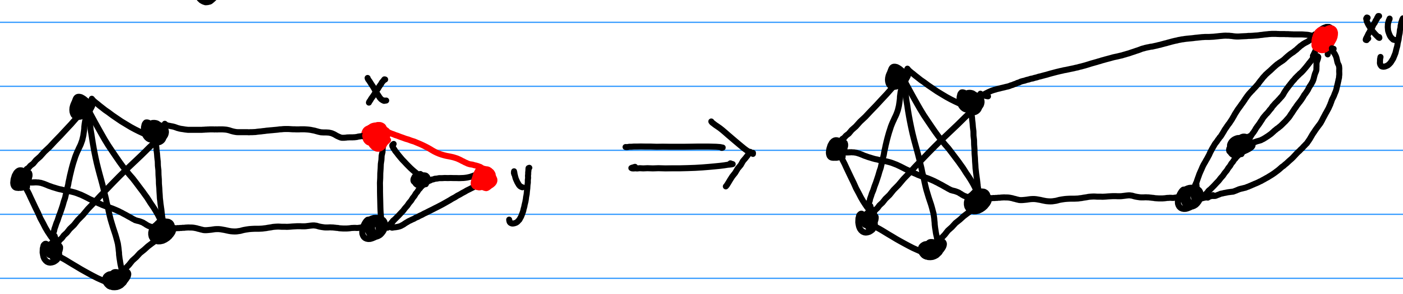
$$|C| \leq \frac{2m}{n}$$

Proof: G has a vertex v whose degree is at most $2m/n$. v's incident edges form a cut of size $\leq 2m/n$.



Edge Contraction

Contracting G at edge (x,y) involves identifying x and y, and removing any self loops to get a new graph $G/(x,y)$



Notice: The min-cut of G is no bigger than the min-cut of $G/(x,y)$

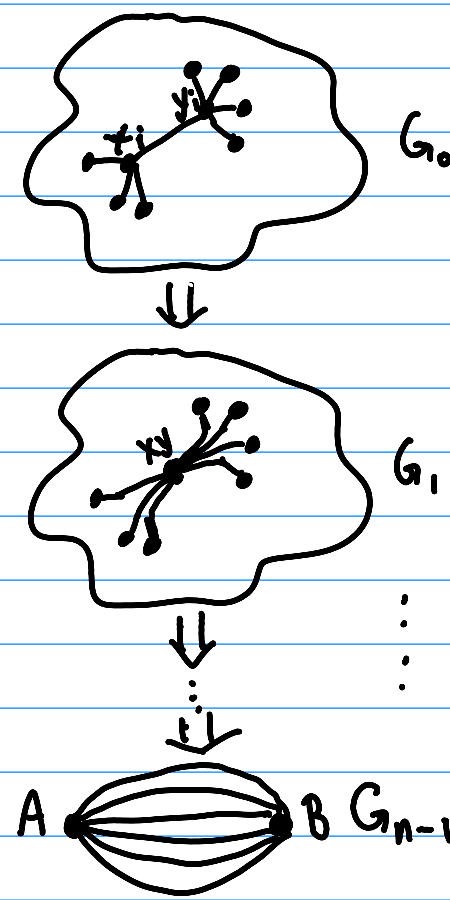
Notice: If C is a min-cut in G and $(x,y) \notin C$ then C is also a min-cut in $G/(x,y)$

Notice: If implemented carefully, edge contraction takes $O(n)$ time

Karger's Contraction Algorithm

$G_i = (V_i, E_i)$
$n_i = |V_i|$

KargerMinCut(G)
  $G_0 \leftarrow G$
  for $i = 1$ to $|V| - 2$ do
    select a random edge (x_i, y_i) in $E(G_{i-1})$
    $G_i \leftarrow G_{i-1} / (x_i, y_i)$
  return all edges in G_{n-1}



- KargerMinCut runs in $O(n^2)$ time
- Does it correctly return a min-cut of G?

Correctness of KargerMinCut

Let C be some min-cut in G

$$\Pr\{(x_1, y_1) \in C\} = \frac{|C|}{|E|} = \frac{|C|}{|E_0|} \leq \frac{2|E_0|}{|E_0| \cdot |V_0|} = \frac{2}{n}$$

$\therefore \Pr\{C \text{ survives at time } 1.\} \geq 1 - \frac{2}{n}$

$$\Pr\{C \text{ survives at time } i \mid C \text{ survives at time } i-1\} \geq 1 - \frac{2}{n-i+1}$$

$\therefore \Pr\{C \text{ survives at time } n-2\}$

$$\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdot \left(1 - \frac{2}{n-2}\right) \cdot \cdots \cdot \left(1 - \frac{2}{3}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right) \cdot \cdots \cdot \left(\frac{1}{3}\right) = \frac{2(n-2)!}{n!} = \frac{2}{n(n-1)} \geq \frac{1}{n^2}$$

$\therefore \Pr\{\text{KargerMinCut is correct}\} \geq \frac{1}{n^2}$

$$\boxed{\left(1-\frac{1}{k}\right)^k \le \frac{1}{e}}$$

Improving KargerMinCut

Theorem: Algorithm KargerMinCut runs in $O(n^2)$ time and correctly reports a min-cut with probability at least $1/n^2$

KargerMinCut*:

Run KargerMinC $cn^2 \ln n$ times and report the best solution found

$$\Pr\{\text{KargerMinCut* is incorrect}\} \le \left(1-\frac{1}{n^2}\right)^{cn^2 \ln n} \le \left(\frac{1}{e}\right)^{c \ln n} \le \frac{1}{n^c}$$

Theorem: Algorithm KargerMinCut* runs in $O(cn^4 \log n)$ time and correctly reports a min-cut with probability at least $1-\frac{1}{n^c}$.

Improving KargerMinCut*

Theorem: Algorithm KargerMinCut* runs in $\underbrace{O(cn^4 \log n)}_{\text{not fast enough}}$ time and correctly reports a min-cut with probability at least $1 - \frac{1}{n^c}$.

$$\Pr\{C \text{ survives at time } n-1\}$$

$$\geq \left(1-\frac{2}{n}\right)\cdot\left(1-\frac{2}{n-1}\right)\cdot\left(1-\frac{2}{n-2}\right)\cdot \dots \cdot \left(1-\frac{2}{3}\right) \geq \frac{1}{n^2}$$

$\frac{1}{3}$ too small!

$$\Pr\{C \text{ survives at time } n-n/\sqrt{2}\}$$

$$\geq \left(1-\frac{2}{n}\right)\cdot\left(1-\frac{2}{n-1}\right)\cdot\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{\lceil n/\sqrt{2}\rceil}\right)$$

$$= \left(\frac{n-2}{n}\right)\cdot\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\cdots\left(\frac{\lceil n/\sqrt{2}\rceil-2}{\lceil n/\sqrt{2}\rceil}\right) \geq \frac{1}{2}$$

Randomized Shrinking

Just make the problem smaller:

```
KargerSteinReduce(G)
    G_0 ← G
    for i=1 to |V| - ⌈|V|/√2⌉
        select a random edge (x_i, y_i) in E(G_{i-1})
        G_i ← G_{i-1} / (x_i, y_i)
    return G_i
```

Lemma: KargerSteinReduce runs in $O(n^2)$ time and produces a graph G' with $n/\sqrt{2}$ vertices. Furthermore, with probability at least $1/2$ $\text{mincut}(G') = \text{mincut}(G)$.

The Main Event

KargerSteinMinCut(G)
    if $|V| \leq 20$ then return MinCut(G)
    $G_1 \leftarrow$ KargerSteinReduce(G)
    $C_1 \leftarrow$ KargerSteinMinCut(G_1)
    $G_2 \leftarrow$ KargerSteinReduce(G)
    $C_2 \leftarrow$ KargerSteinMinCut(G_2)
    return $\min\{C_1, C_2\}$.

$\boxed{\text{note: } G_1 \neq G_2}$

Running Time: $T(n) = O(n^2) + 2 \cdot T(n/\sqrt{2}) = O(n^2 \log n)$

Prob. Correctness: $S(n) = \frac{1}{4} \cdot 0 + \frac{1}{2} \cdot S(n/\sqrt{2}) + \frac{1}{4} \cdot \left(1 - \left(1 - S(n/\sqrt{2})\right)^2\right)$

$$\geq \frac{1}{\log n}$$

Theorem: KargerSteinMinCut runs in $O(n^2 \log n)$ time and correctly outputs a min-cut of G with probability at least $1/\log n$

The Main Main Event

Theorem: KargerSteinMinCut runs in $O(n^2 \log n)$ time and correctly outputs a min-cut of G with probability at least $\underbrace{1/\log n}_{\text{too small}}$

KargerSteinMinCut*:
Run KargerSteinMinCut $c \cdot \log n \cdot \ln n$ times and output the best cut found

just right!

Theorem: KargerSteinMinCut* runs in $O(n^2 \log^3 n)$ time and correctly outputs a min-cut of G with probability at least $1 - \frac{1}{n^c}$.

Summary

Theorem: Let P be any maximization problem such that
1. Given an input S of size $O(1)$ we can compute $val(S)$ in $O(1)$ time
2. Given an input S of size n we can test if $val(S) \geq t$ in $D(n)$ time, for any real number t.
3. Given an input S of size n, we can partition S into subproblems $S_1, \ldots, S_r$ each of size at most αn and such that
$$val(S) = \min\{val(S_i) : i \in \{1, \ldots, r\}\}$$

Then we can compute $val(S)$ in $O(D(n))$ expected time

Theorem: Given a graph G with n vertices, there exists an algorithm that runs in $O(cn^2 \log^3 n)$ time and outputs a cut of G that, with probability at least $1 - \frac{1}{n^c}$, is of minimum cardinality.