Geometric Optimization

Piotr Indyk

Geometric Optimization

- Minimize/maximize something subject to some constraints
- Have seen:
  - Linear Programming
  - Minimum Enclosing Ball
  - Diameter/NN (?)
- All had easy polynomial time algorithms for d=2

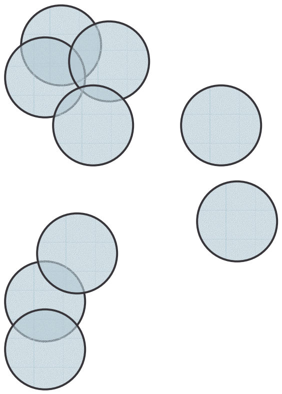
Today

- NP-hard problems in the plane
  - Packing and piercing [Hochbaum-Maass'85]
  - TSP, Steiner trees, and a whole lot more [Arora'96] (cf. [Mitchell'96])
- Best exact algorithms exponential in n
- We will see $(1+\varepsilon)$-approximation algorithms that run in poly time for any fixed $\varepsilon>0$ (so, e.g., $n^{O(1/\varepsilon)}$ is OK)
- Such an algorithm is called a PTAS

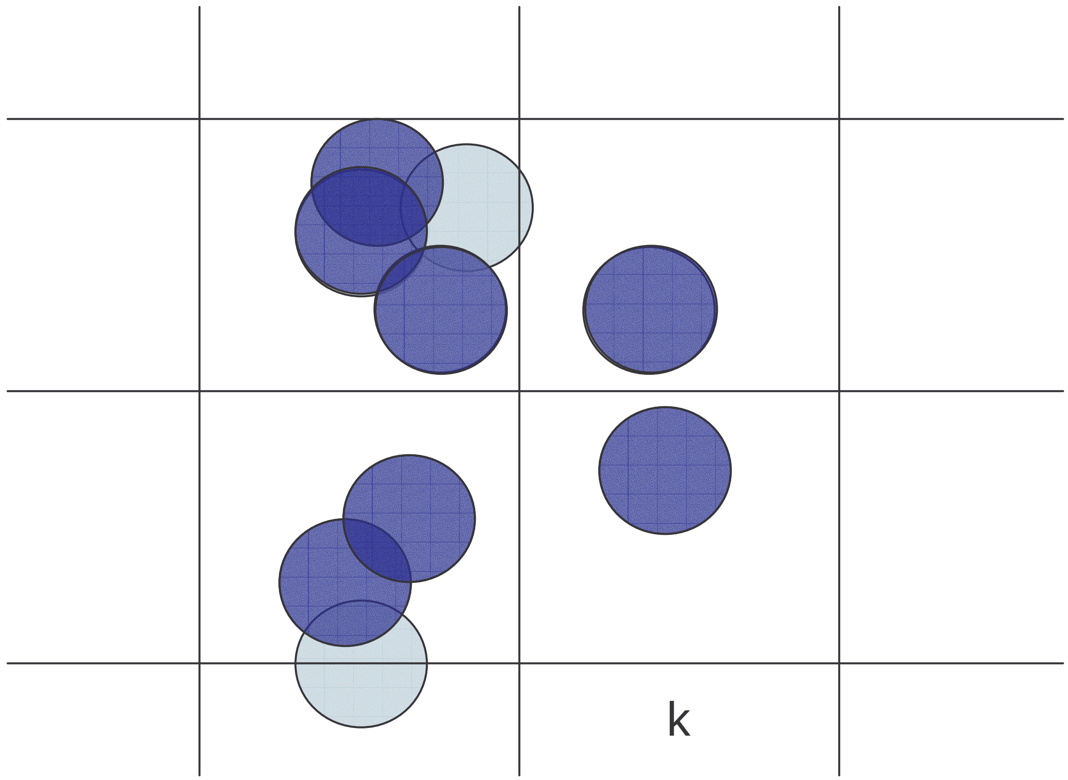
(attention: falafel fans)

Packing

- Given: a set C of unit disks $c_1 \ldots c_n$
- Goal: find a subset C' of C such that:
  - All disks in C' are pair-wise disjoint
  - |C'| is maximized
- How to get a solution of size $\geq (1-\varepsilon)$ times optimum ?



Algorithm



Algorithm, ctd.

- Impose a grid of granularity k
- For each cell a
  - Compute c(a) = the set of disks fully contained in a
  - Solve the problem for c(a) obtaining c'(a)
- Report C' =union of c'(a) for all cells a

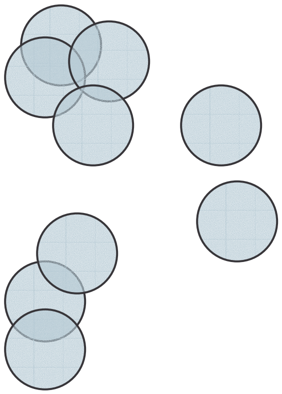
Computing c'(a)

- The grid cell a has side length = k
- Can pack at most k^2 disks into a
- Solve via exhaustive enumeration → running time $O(n^{k^2})$
- This also bounds the total running time

Analysis

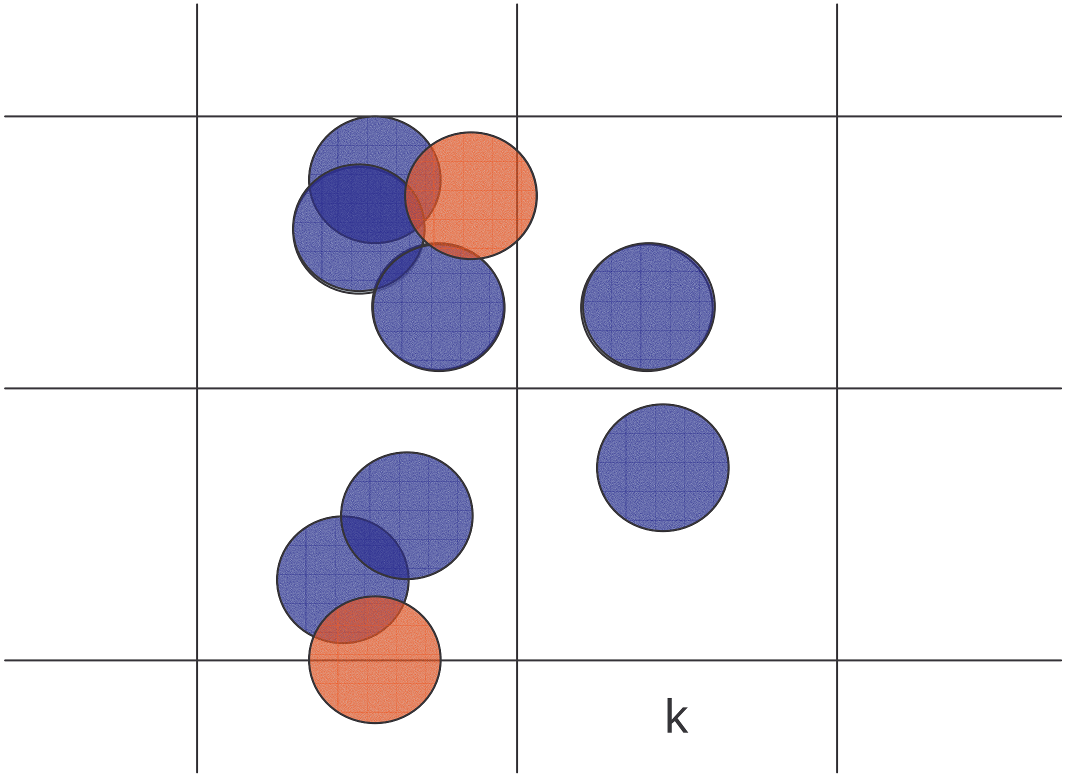
- Approximation factor:
  - Consider the optimal collection of disks C
  - Shift the grid at random
  - The probability that c not fully contained in some a is at most O(1)/k
  - The expected number of removed disks is |C|/k
  - Setting $k=O(1/\varepsilon)$ suffices
- Altogether: running time $n^{O(1/\varepsilon^2)}$

Piercing



- Given: a set C of unit disks $c_1 \ldots c_n$
- Goal: find a set P of points such that:
  - $P \cap c_i \neq \emptyset$ for $i=1 \ldots n$
  - $|P|$ is minimized
- How to get a solution of size $\geq (1+\varepsilon)$ times optimum ?

Algorithm



Algorithm, ctd.

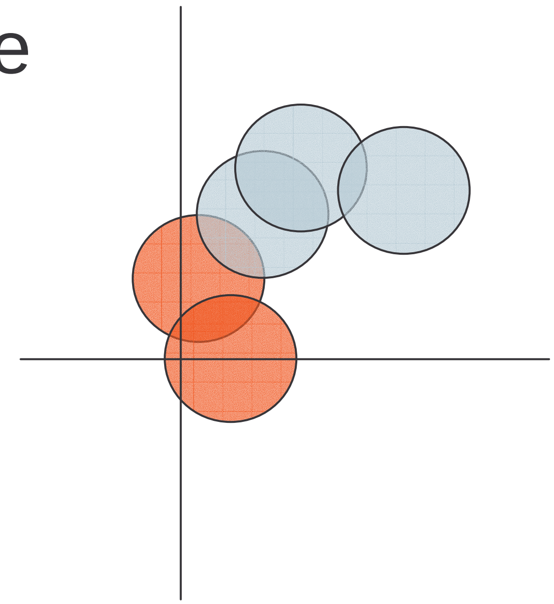
- Impose a grid G of granularity k
- For each cell a
  - Compute c(a) = the set of disks intersecting a
  - Solve the problem for c(a) obtaining P(a)
- Report P=union of P(a) for all cells a

Computing P(a)

- The grid cell a has side length = k
- Can pierce c(a) using at most k^2 points
- Sufficient to consider $O(n^2)$ choices of piercing points (for all points in P)
  - Compute arrangement of disks in c(a)
  - Points in the same cell equivalent
- Solve via exhaustive enumeration → running time $O(n^{2k^2})$
- This also bounds the total running time

Analysis

- Problem: disks B within distance 1 from the grid boundary considered up to 4 times
- If we eliminated all the occurrences of the disks in B from all c(a)'s, one could pierce the remainder with cost at most OPT(C)
- Suffices to show that the cost of piercing B is small

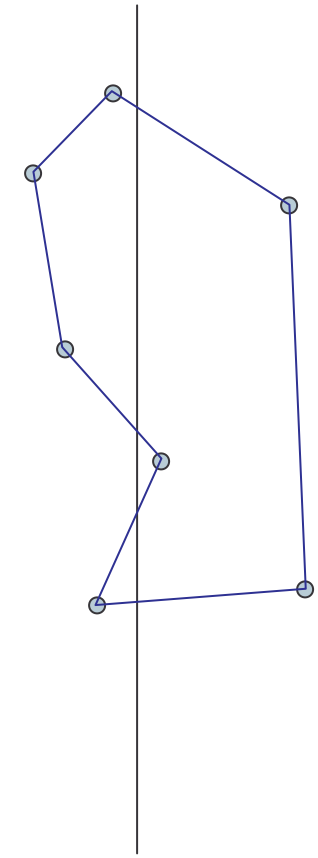


Analysis ctd.

- Shift the grid at random
- Disks in B can be covered by $P' = P \cap (G \oplus Ball(o,2))$
- The probability that a fixed $p \in P$ belongs to P' is $O(1)/k$
- The expected $|P'|$ of is $O(|P|)/k$
  - Setting $k=O(1/\varepsilon)$ suffices
- Altogether: running time $n^{O(1/\varepsilon^2)}$

Dynamic programming for TSP

- Divide the points using randomly shifted line
- Enumerate "all" possible configurations C of crossing points
- For each C, solve the two recursive problems
- Choose the C that minimizes the cost



Analysis

- Structure lemma: for any tour T, there is another tour T' with cost not much larger than the cost of T, which has only constant number of crossing points.