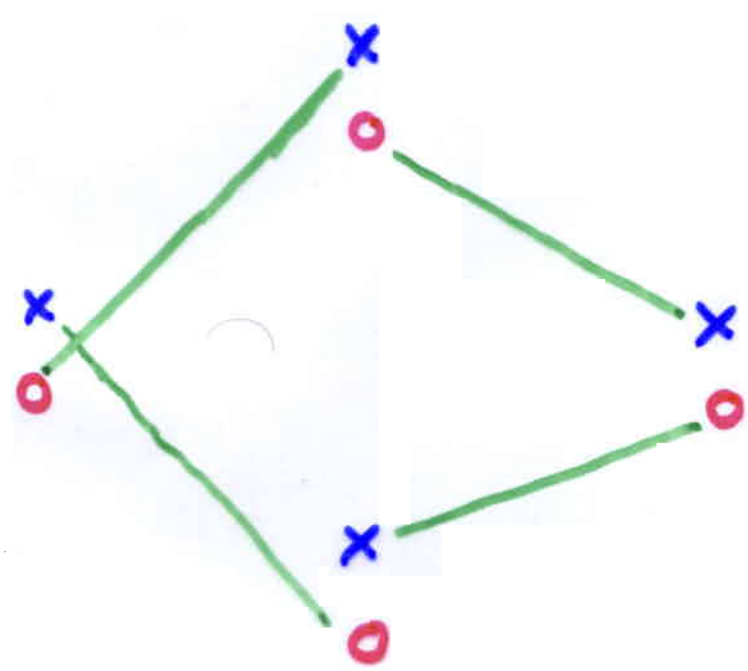
Bipartite Euclidean Matching

Kasturi Varadarajan

Given a set R of n red points and a set B of n blue points. Find a perfect red-blue matching that minimizes sum of distances between matched points.



Hungarian Algorithm

- $O(|V|^3)$
- $O(|E| \times |V|)$

Scaling

$O(|V|^{2.5})$

Vaidya 89

- $O(n^{2.5})$ in the plane

AES 95

- $O(n^2)$

AV 99

- $O(n^{1.5})$ for $(1+\varepsilon)$-approx. matching

(Hungarian algo + Scaling)

Non-bipartite matching

$O(n^{2.5})$ - Vaidya 89

$O(n^{1.5})$ - V 98

$O(n \log^{O(\frac{1}{\varepsilon})} n)$ - Arora 97

for $(1+\varepsilon)$-approx.

(Improves $O(n^{1.5})$ of Vaidya 89)

$O\left(\frac{n \log^6 n}{\varepsilon^3}\right)$ - AV 99

Is there a near-linear algo. for a constant factor or $(1+\varepsilon)$-approximation for bipartite matching?

This talk (Joint work with Pankaj Agarwal): For any $\varepsilon > 0$, $O(n^{1+\varepsilon})$ time algo to compute $\mathrm{poly}(\frac{1}{\varepsilon})$-approx.

Aside: Bottleneck matching bipart. in the plane (min-max instead of min-sum) can be approx. to within a constant factor in near-linear time.

$\downarrow$

Existence of perfect matching in planar bipart. graph

$\downarrow$ Miller-Naor

Single-source shortest path in planar graph

FR

Assume

- Points lie on integer grid
- Inside box of length $\leq n^3$.

$K \leftarrow n^{\varepsilon}$

Match(Q)

L = side-length of cube Q

If $L \leq K$, compute the best matching of the points in Q.

$G \leftarrow$ Randomly shifted grid of size $\frac{L}{K}$

From each cell C that intersects Q,

1. If C contains more red points

than blue points, then remove extra red points.

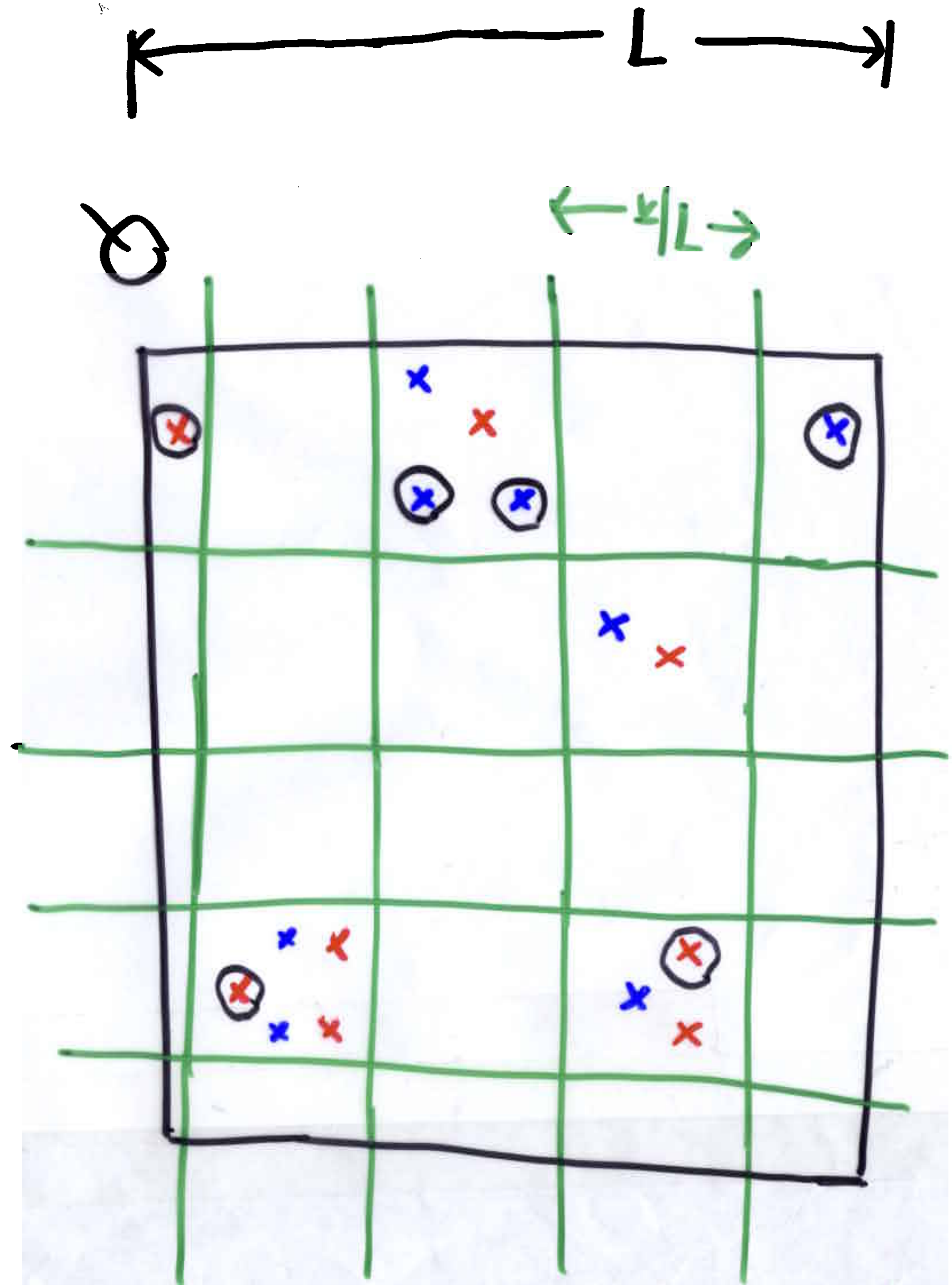
2. If C contains more blue points than red points, then remove extra blue points.

3. Match(C) to recursively compute a perfect matching of the remaining points in C.

Move each "removed" point to center of corresponding cell.

Compute the best p.m. of moved, removed points.

Return resulting p.m.


L
L/k

Running-time Analysis

In Match(Q)

- time to set up recursive sub-problems is # points in $Q + k^2$

- time for "~~Conquer~~ Step" matching moved, removed points is $(k^2)^3 = k^6$

  # of points without multiplicities $\leq k^2$

  though

  # of points with multiplicities may be $\gg k^2$.

Overall, time spent = # points + k^6.

Recursion has $O(\frac{1}{\varepsilon})$ depth.

Time spent at one level of recursion

$$= \sum_{\text{non-empty cells } Q} \#\text{points in } Q + k^6$$

$$= n + n \cdot k^6 = O(nk^6).$$

Overall running time

$$= \frac{1}{\varepsilon} n \cdot k^6 = \frac{1}{\varepsilon} n^{1+6\varepsilon}$$

Cost Analysis for Match(Q)

M^* - best perfect matching for points in Q.

$\mathbb{C}$ - $O(k^2)$ cells into which Q is partitioned

M' = matching for "removed points"

$+ \bigcup_{c \in \mathbb{C}}$ best matching for points in c

$\chi(M^*)$ = # of edges in M^* "cut" by grid.

Claim:

$$\text{Cost}(M') \leq \text{Cost}(M^*) + \chi(M^*) \cdot O\left(\frac{L}{k}\right)$$

$$E[\mathrm{Cost}(M')]$$

$$\le \mathrm{Cost}(M^*) + E[X(M^*)] \times O\left(\frac{L}{k}\right)$$

$$= \mathrm{Cost}(M^*) + O\left(\frac{L}{k}\right) \times \sum_{(u,v)\in M^*} \Pr\left[\begin{matrix}(u,v) \text{ is}\\ \text{cut}\end{matrix}\right]$$

$$\le \mathrm{Cost}(M^*) + O\left(\frac{L}{k}\right) \times \sum_{(u,v)\in M^*} \frac{d(u,v)}{2\frac{L}{k}}$$

$$\le \mathrm{Cost}(M^*) + O(\mathrm{Cost}(M^*))$$

$$= O(\mathrm{Cost}(M^*))$$

Cost increases by constant factor at each level. So cost of matching computed $\le c^{\frac{1}{\varepsilon}}\ \mathrm{Cost}(M^*)$.

Being more careful, one gets $\mathrm{poly}\left(\frac{1}{\varepsilon}\right) \times \mathrm{Cost}(M^*)$