Reductions to Approximate Nearest Neighbor

Piotr Indyk

Stanford University

The base case[IM'98]

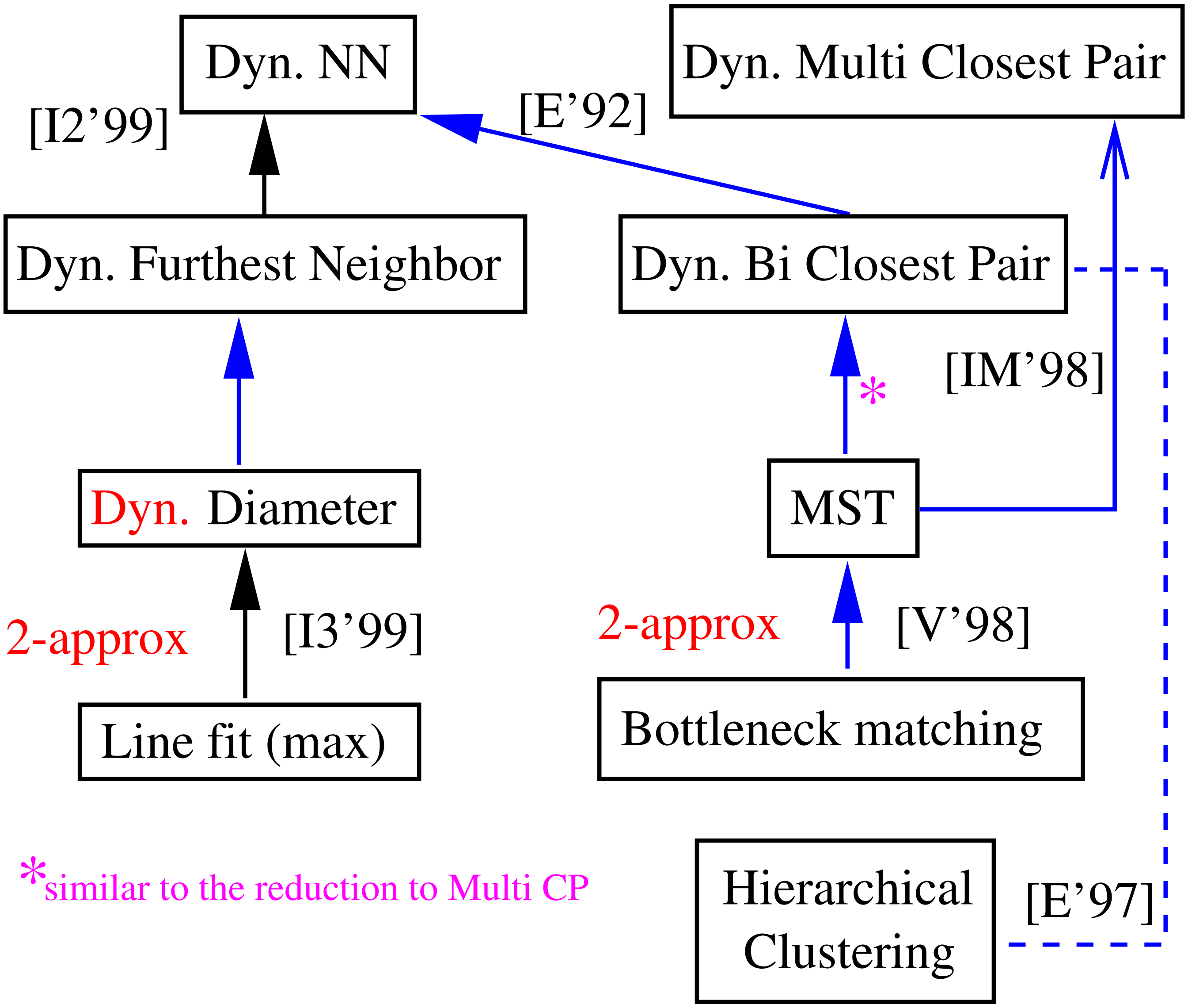
Dynamic nearest neighbor:

- $(1+\epsilon)$-approximation
- search,insert,delete in $\tilde{O}(dn^{1/(1+\epsilon)})$ time
- works for l_1/l_2

Dynamic multichromatic closest pair (optional)

- want to keep closest pair of points with different colors
- approximation, time and norms as above

The reductions



The reductions: explanation

- time overhead: $O(\log^{O(1)} n)$
- approximation: $1 + \epsilon$ (unless stated otherwise)
- "black" reductions: work for l_1/l_2
- "blue" reductions: work for any metric
- problems:
  - Bichromatic Closest Pair: find closest red-blue pair
  - Line fit: find a line minimizing max distance from a pointset
  - Bottleneck matching: perfect matching with smallest max edge

Other reductions [BOR'99].

Furthest to Nearest Neighbor

Hamming space, ϵ-PLEB with parameter R.
Reduce $(1+\epsilon)$-FN to $(1+\gamma(\epsilon)\cdot\epsilon)$-NN.

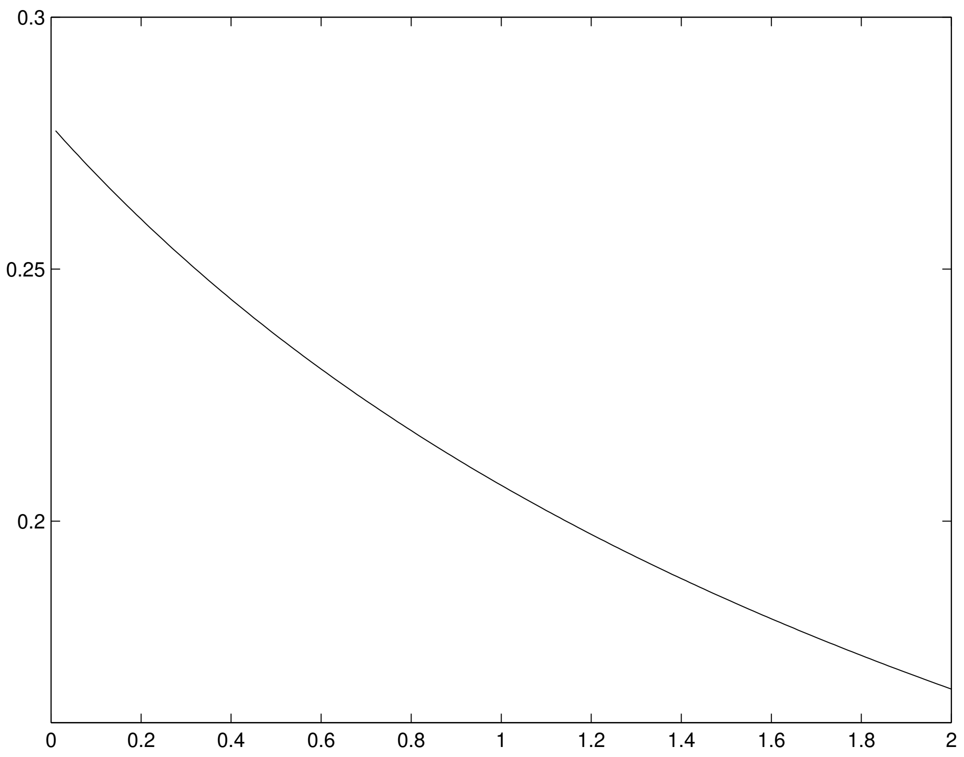


Figure 1: $\gamma(\epsilon)$ vs. ϵ

Idea

$$|pq| = d - |p\overline{q}|$$

i.e. the NN of $\overline{q}$ in P is the FN of q in P.

Does not preserve approximation:

- assume that FN p is s.t. $|pq| = R$
- if we find E-NN of $\overline{q}$ in P, we get p' s.t.

$$|\overline{q}p'| = E|\overline{q}p| = E(d - R)$$

- therefore

$$\frac{|qp|}{|qp'|} \leq \frac{R}{d - E(d - R)} = \frac{R}{(1+\epsilon)R - \epsilon d}$$

Have to reduce the dimension [KOR'98,IM'98]

Dimensionality reduction

- mapping $f : \{0,1\}^d \to \Sigma^D$, where each coordinate of $f(p)$ is a concatenation of k random bits of p

$$\text{if } |pq| = R = \rho d \text{ then } \frac{|f(p)f(q)|}{D} \approx 1 - (1-\rho)^k$$

- replace each coordinate a_i of $f(p)$ by binary error-correcting code $C(a_i)$

$$\frac{|C[f(p)]C[f(q)]|}{D} \approx \frac{1-(1-\rho)^k}{2}$$

- optimize k

MST to Multi Dynamic Closest Pair

Kruskal's MST Algorithm:

- find smallest intercluster pair - simple
- merge clusters:
  - update the color of the smaller cluster $\rightarrow$ only $O(\log n)$ overhead

Correctness of approximation

Idea: Compare the "exact" algorithm A to the "approximate" algorithm A'.

Claim: Before the "time step" t (i.e. adding the tth edge) of A', there is at least one edge (say e) among those chosen by A during its first t steps, such that e can be added to A''s tree without creating a cycle.

Proof: By applying pigeonhole principle to the number of connected components in the trees of A and A'.

Therefore we can bound the cost of the tth edge of A' by $(1 + \epsilon)$ times the cost of the tth edge of A (since the costs of A are increasing).