Reductions Among High-Dimensional Proximity Problems

Goel, Indyk, Varadarajan, SODA 2001

Introduction

Problem:

- Proximity problems:
  - Nearest neighbour
  - Furthest neighbour
  - Diameter
  - ...
- Approximate versions for faster running time
  - Approximation factor: $c = 1 + \varepsilon$
- Reduction to Approximate Nearest Neighbour Search
  - Best known (randomized) running time: $\tilde{O}(d\, n^{1/(1+\varepsilon)})$ per query/update in a dynamic setting [Indyk-Motwani STOC '98]

Problems

- Approximate **Furthest Neighbour**
- Approximate **Diameter**
- Approximate **Discrete Centre**
  - *Approximate **Line Centre***
- Approximate **Bottleneck Matching**
- Approximate **Minimum Weight Matching**
- *Approximate **Metric Facility Location***

Contribution

Subquadratic running time

for all the problems

Outline

- *(Warm-up exercise)* $\sqrt{2}$-approximation algorithm for Furthest Neighbour Search
  - Can be used to obtain $\sqrt{2}$-approximations for Diameter and Discrete Centre problems
- $(1 + \varepsilon)$-approximation algorithm for Diameter
  - Also gives $(1 + \varepsilon)$-approximation for Furthest Neighbour
- $2(1 + \varepsilon)$-approximate Bottleneck Matching
- $(2 + O(\varepsilon))$-approximate Minimum Weight Matching

c-Furthest Neighbour Search

A $\sqrt{2}$-approximation

c-Furthest Neighbour Search

- **FNS**: Given a set $P \subset \mathrm{R}^d$ and a query point q, return the element of P furthest from q
- The approximate version (***c*-FNS**): Return a point of P that c-approximates the furthest neighbour
  - Precisely, return p such that

    $$\mathrm{d}(q, p) \geq (1/c) \max_{p' \in P} \mathrm{d}(q, p')$$
- We look for a $(\sqrt{2} + 1/n^{\theta(1)})$-approximation

Other problems reducible to *c*-FNS

- Given: an n-point set $P \subset \mathrm{R}^d$
- Approximate Discrete Centre Problem (***c*-DCP**): Find $s \in P$ such that:

$$\max_{p \in P} \mathrm{d}(p, s) \le c \min_{s \in P} \max_{p \in P} \mathrm{d}(p, s)$$

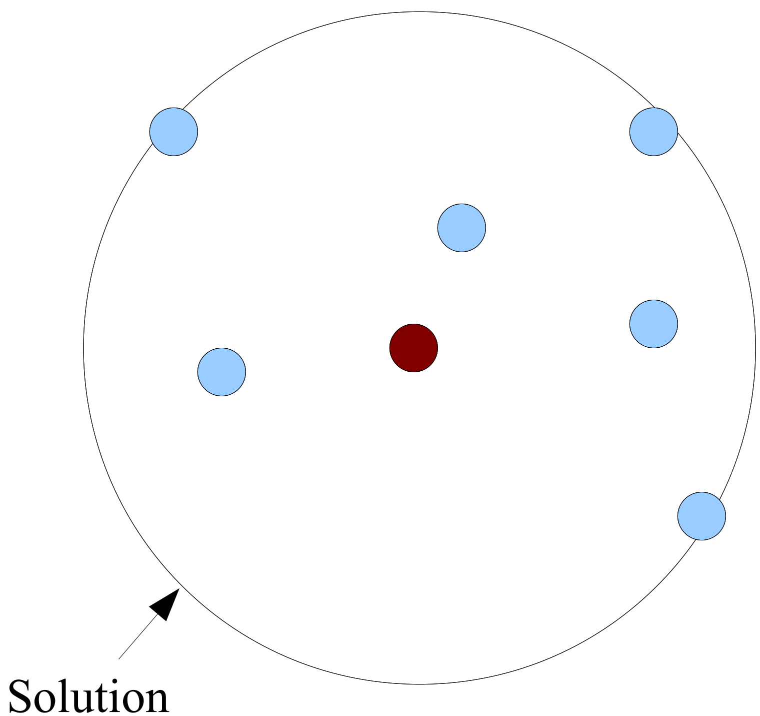
- Approximate Diameter Problem: Find $s \in P$ such that

$$\mathrm{d}(p, q) \ge (1/c) \max_{p, q \in P} \mathrm{d}(p, q)$$

c-FNS: The Reduction

- From c'-Approximate Minimum Enclosing Ball (**c'-MEB**): Given $P \subset \mathrm{R}^d$, find $s \in \mathrm{R}^d$ such that

$$\max_{p \in P} \mathrm{d}(p, s) \leq c \min_{s \in \mathrm{R}^d} \max_{p \in P} \mathrm{d}(p, s)$$



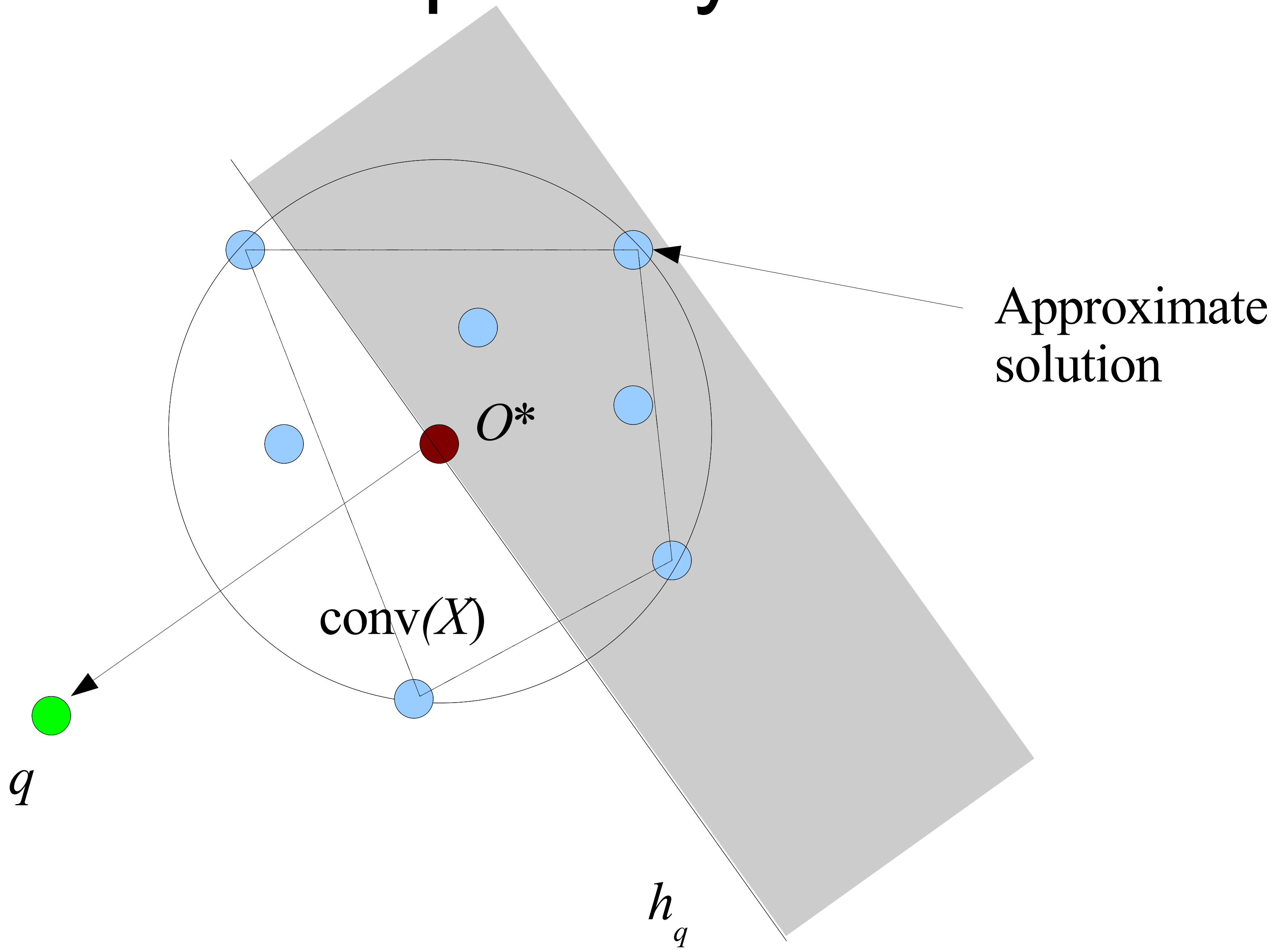
Also known as:
c'-Approximate Continuous Centre Problem

Solution in
$\tilde{O}(d^3\, n \log 1/\varepsilon)$ time

Reduction Method

- Assume we could compute the *exact* minimum enclosing ball $B(O^*, r^*)$ for P
- There is a subset X of P such that
  - $X \subset S(O^*, r^*)$
  - $|X| \leq d + 2$
  - $O^* \in \mathrm{conv}(X)$
- Hyperplane h_q passes through O^* and is orthogonal to $q - O^*$
- Return any point of X on the side of h_q opposite to q
  - such a point MUST exist and be a √2-approximation

Graphically...



Caveats

- Can't compute *exact* minimum enclosing ball
  - So compute an approximation (upto factor $1 + 1 / (n^{\theta(1)}\sqrt{d})$) – introduces only log factors in running time
  - X is all points within $O(1/n^{\theta(1)})$ threshold
- X may be of size $\Omega(n)$
  - So perturb points slightly (by random vectors of norm $O(1/n^{\theta(1)})$ – the "smoothed complexity" of X is $O(d \log n)$
- Running time:
  - Construction: bounded by that of c'-MEB: $\tilde{O}(d^3 n)$
  - Query: $O(d^2 \log n)$ (lin. search in X for point furthest from q)

Lower Bound?

Can we do better than $c = \sqrt{2}$ as fast?

– Unlikely, because...

- On a *random* point set, a c-approximation for FNS ($c < \sqrt{2}$) would yield a constant-factor approximation for nearest neighbour within same time bounds
- This problem was considered by Yianilios [SODA '00] and seems **very difficult** to achieve in time $\tilde{O}(d^{\theta(1)})$

$(1 + \varepsilon)$-Approximate Diameter and Furthest Neighbour Search

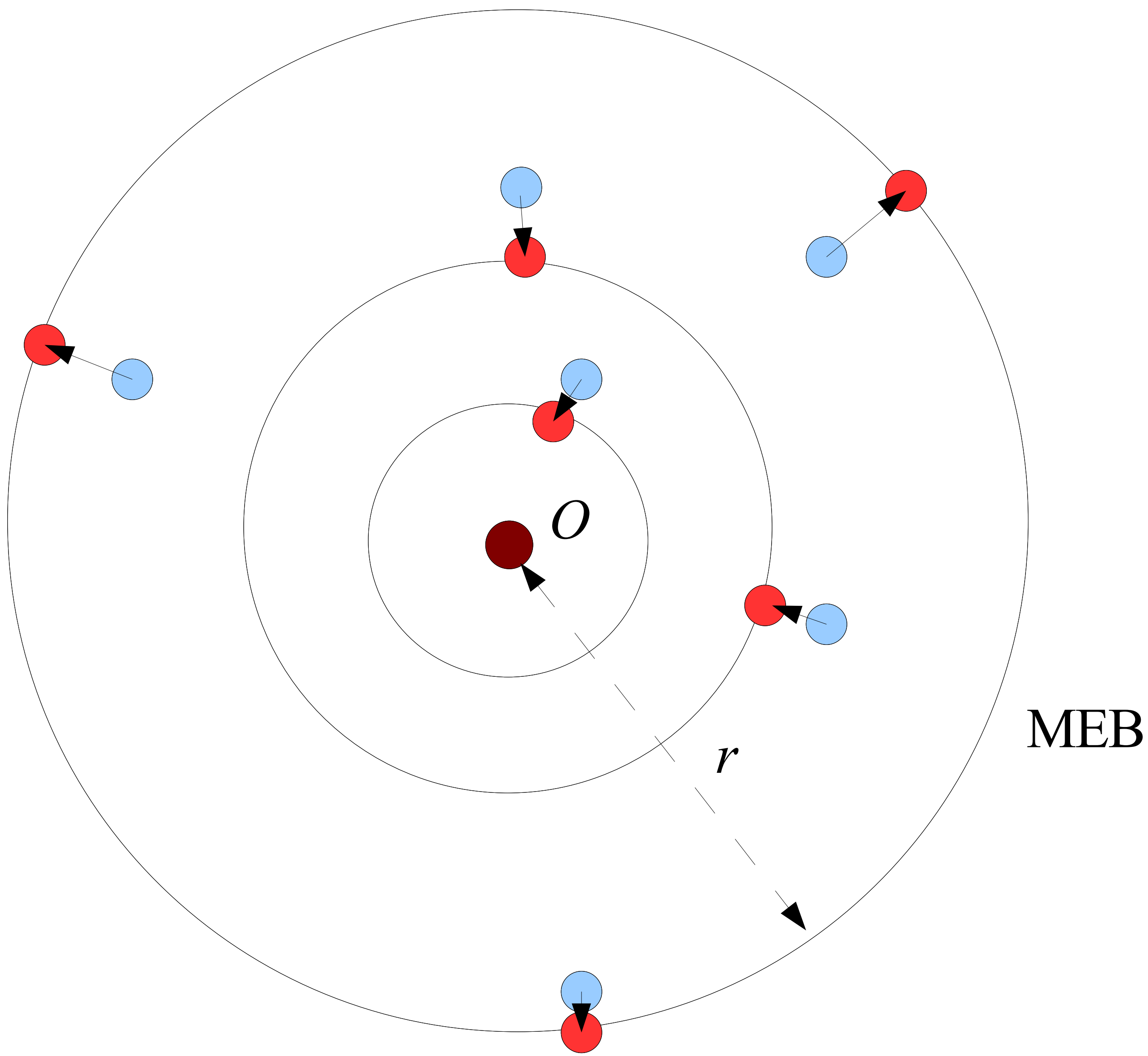
$(1 + \varepsilon)$-approximating Diameter and Furthest Neighbour Search

- Answer $(1 + \varepsilon)$-Diameter/FNS queries by using $\tilde{O}(1)$ $(1 + \varepsilon)$-NNS queries
- Preprocessing time: $\tilde{O}(d\, n^{1 + 1/(1 + \varepsilon)})$
- We will look at only the diameter problem for simplicity.

Method

- Compute an approximate minimum enclosing ball $B(O, r)$
- Construct a series of k "shells", each of radius $1 / (1 + \alpha)$ times that of its predecessor. The first shell is $S(O, r)$.
  - α will be specified later
  - k is $O(1 / \alpha)$
- Round each point to nearest shell
- Construct $(1 + \varepsilon)$-NNS data structure for each shell

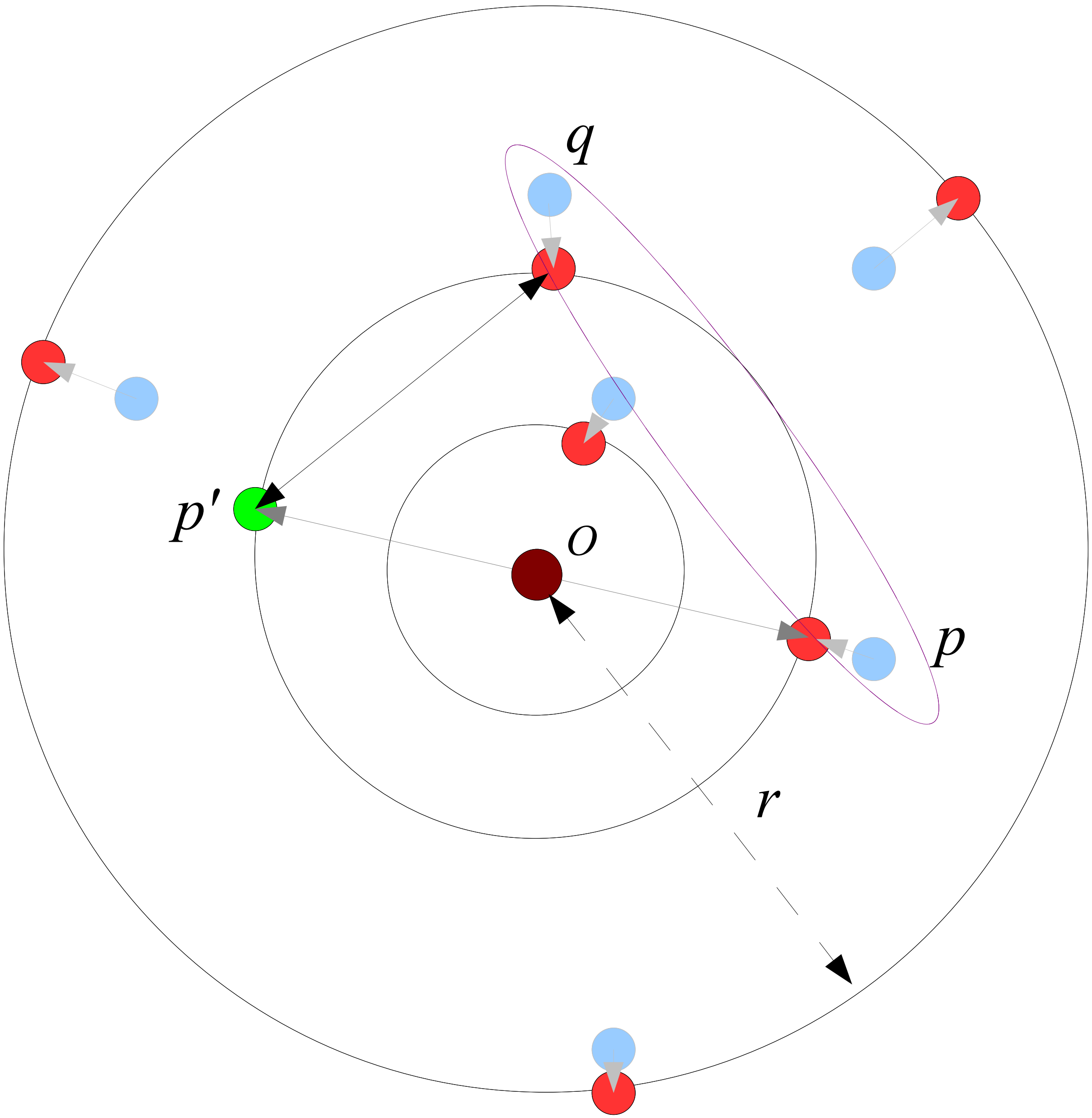
Graphically...



Method (contd)

- For each point $p \in P$ and each shell S_i
  - Reflect p in O and promote to S_i to get the "antipode" p'
  - Find (approximate) nearest neighbour q of p' from points on S_i
    - This gives a "candidate diameter pair" (p, q)
- Return the pair among the candidate pairs from all the shells that is furthest apart
  - This is a $(1 + \varepsilon)$-approximation to the diameter

Graphically...



Running Time

- Set $\alpha = 1 / (c \log n)$
- There are nk c-NNS queries
  - So running time $= \tilde{O}(nT)$, where T is the running time of c-NNS

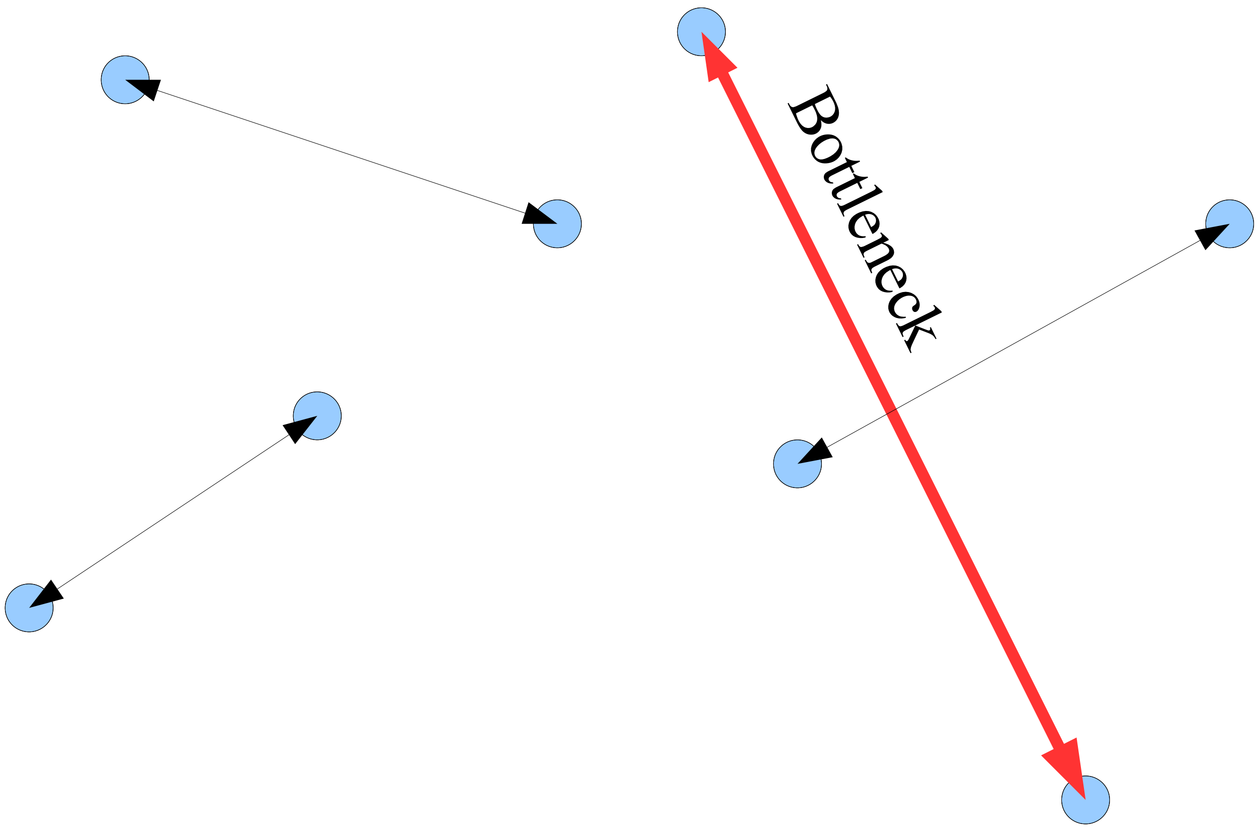
$$= \tilde{O}(d\, n^{1 + 1/(1 + \varepsilon)})$$

Approximate Bottleneck Matching

Approximate Bottleneck Matching

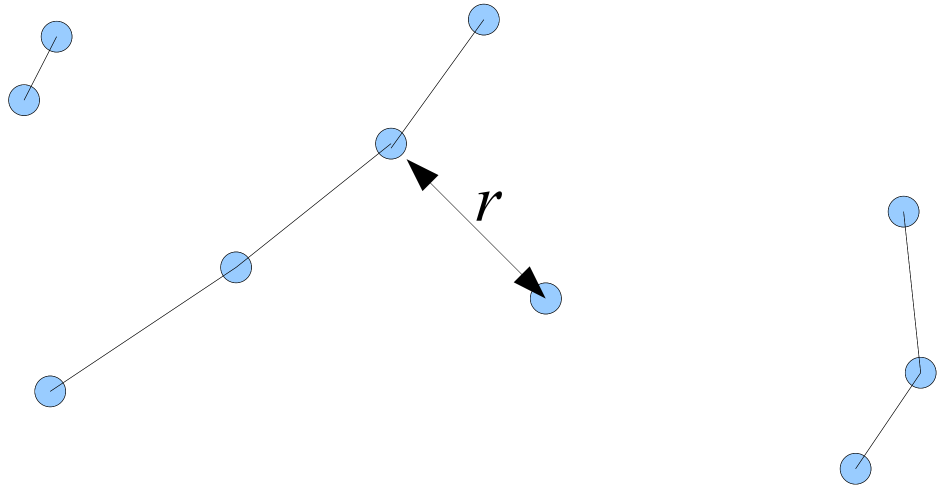
- P is a subset of R^d with $2n$ points
- Perfect Matching: Partition of P into disjoint pairs
- "Bottleneck Cost": Distance between furthest pair in matching
- Bottleneck Matching Problem: Find perfect matching with minimum bottleneck cost
  - Approximate version: we'll compute a $2(1 + \varepsilon)$ approximation

Graphically...



“Short Hop Graph”

- Let $G(r)$ be the graph with $V = P$ and E = set of pairs that are r-close
  - “short hop graph”?



- Let r^* be smallest r for which each connected component has an even number of vertices

Bounds

- **Lemma 1 (lower bound)**: The cost c^* of the optimal matching is at least r^*
- **Lemma 2 (upper bound)**: Let T be a tree on a vertex set V of even cardinality $2m$. Let l be the length of the longest edge of T. We can construct a perfect matching on V with bottleneck cost at most $2l$. Given T, construction time is $O(m)$.
  - Gives method of computing perfect matching with bottleneck cost at most $2r^* \leq 2c^*$

Reduction to NNS

- Compute a spanning forest $\{T_1, \ldots, T_k\}$ of P such that
  - Each edge has length at most $r^*(1 + \varepsilon)$
  - Each tree T_i has an even number of vertices
- Can be done by running Kruskal's MST algorithm until each connected component is even, with $n \log n$ calls to (an approximate algo for) ***k*-Chromatic Dynamic Closest Pair**
  - **k-CDCP:** Given a dynamic set of coloured points, find closest pair with different colours

Reduction to NNS (contd)

- Eppstein [DCG '95 etc] reduces $(1 + \varepsilon)$-k-CDCP to $(1 + \varepsilon)$-NNS via $(1 + \varepsilon)$-2-CDCP
  - Polylogarithmic overhead
- Now apply method of Lemma 2 to find perfect matching with bottleneck cost at most $2(1 + \varepsilon)\, c^*$
- Running time: $\tilde{O}(d\, n^{1 + 1/(1 + \varepsilon)})$

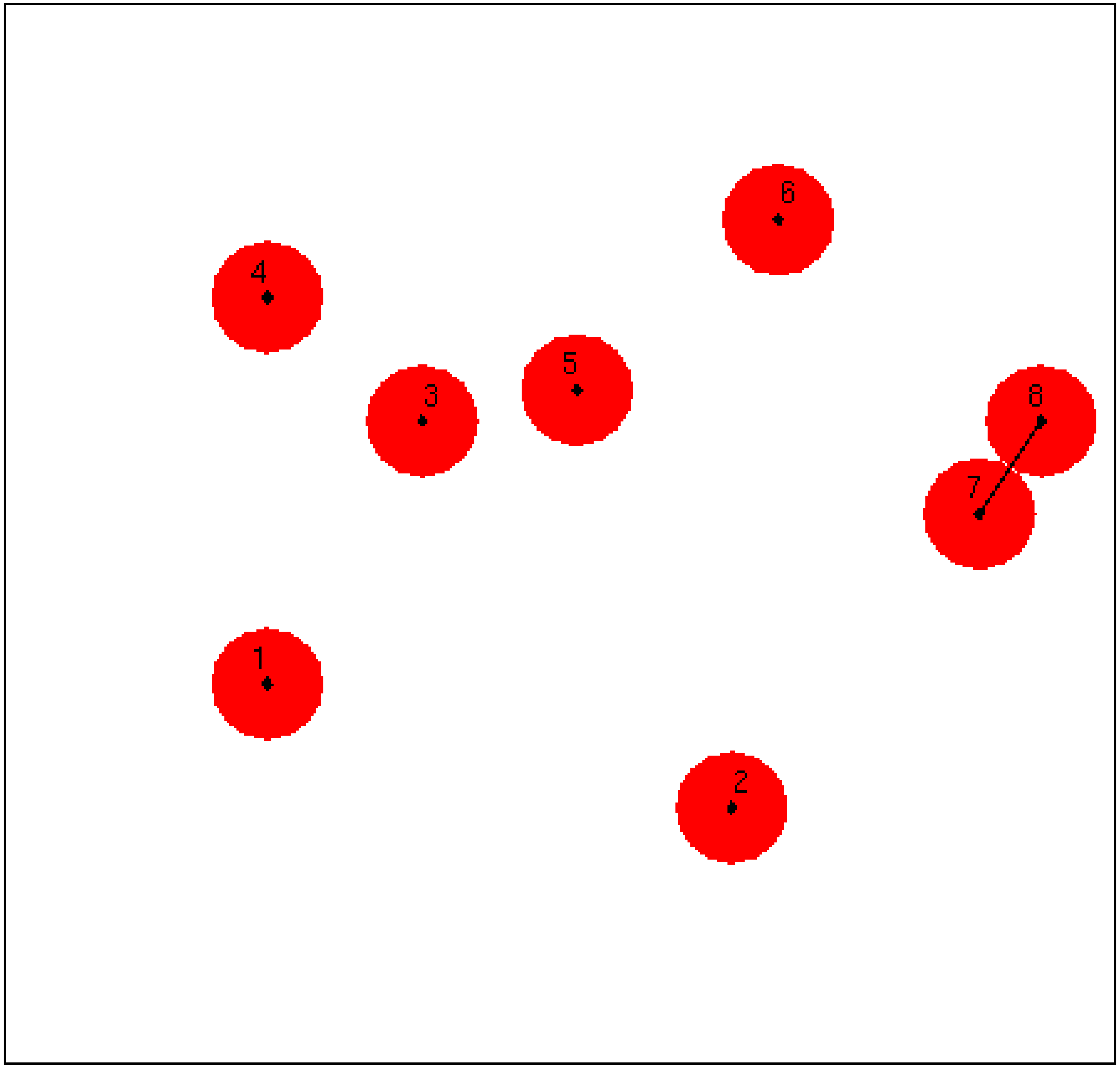
Minimum-Weight Matching

Improving the Goemans-Williamson Method for approximate MWM

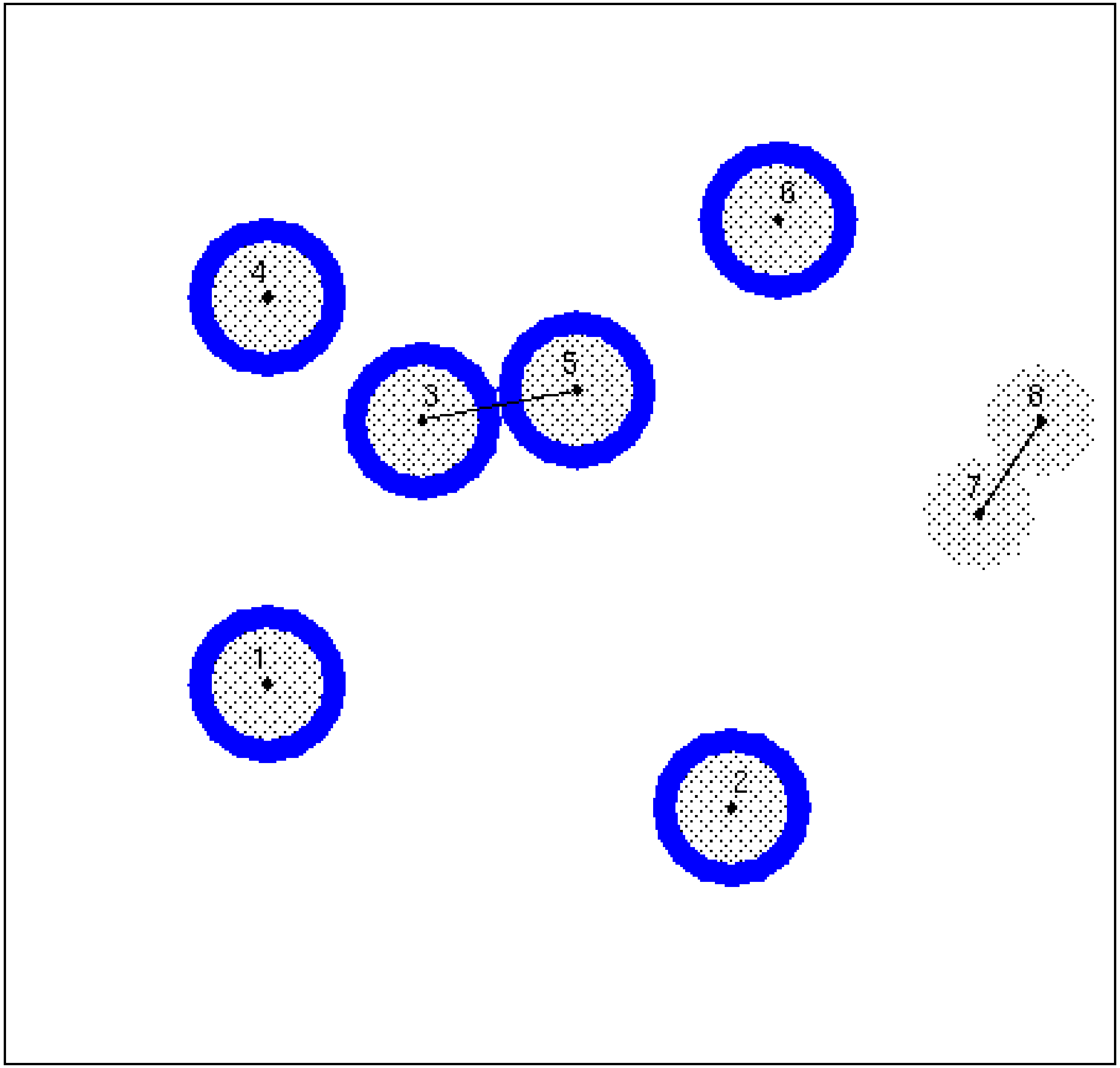
Minimum-Weight Matching

- Goemans-Williamson Method for 2-approximation
  - Active component: odd vertices
  - Inactive component: even vertices
  - Grow balls around vertices in active components until two balls collide
  - Add edge between centres of colliding balls to solution
  - Merge colliding components and continue
- Resulting forest weighs at most twice MWM
- Trivial to convert forest to matching

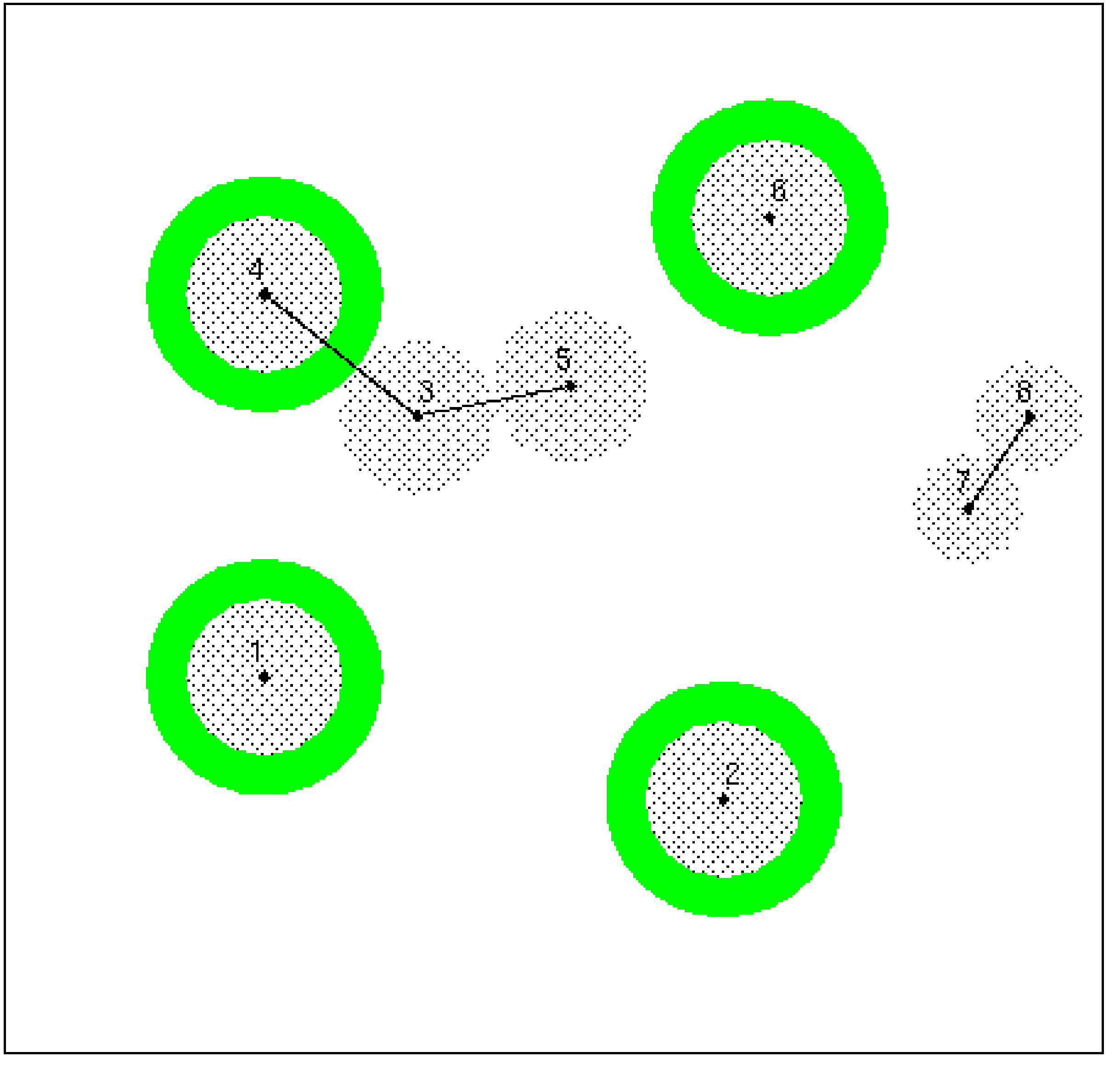
Graphically...



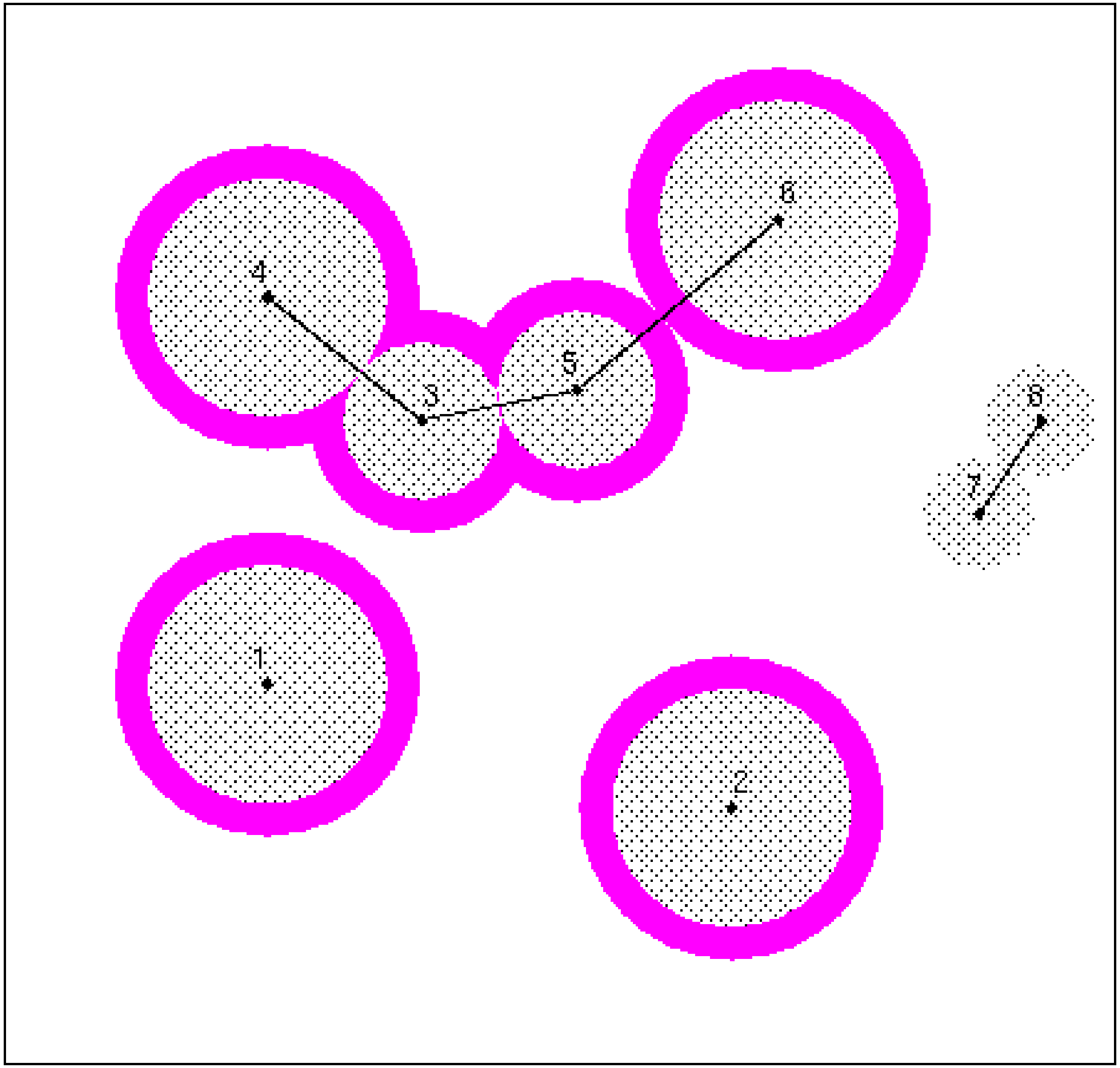
Graphically...



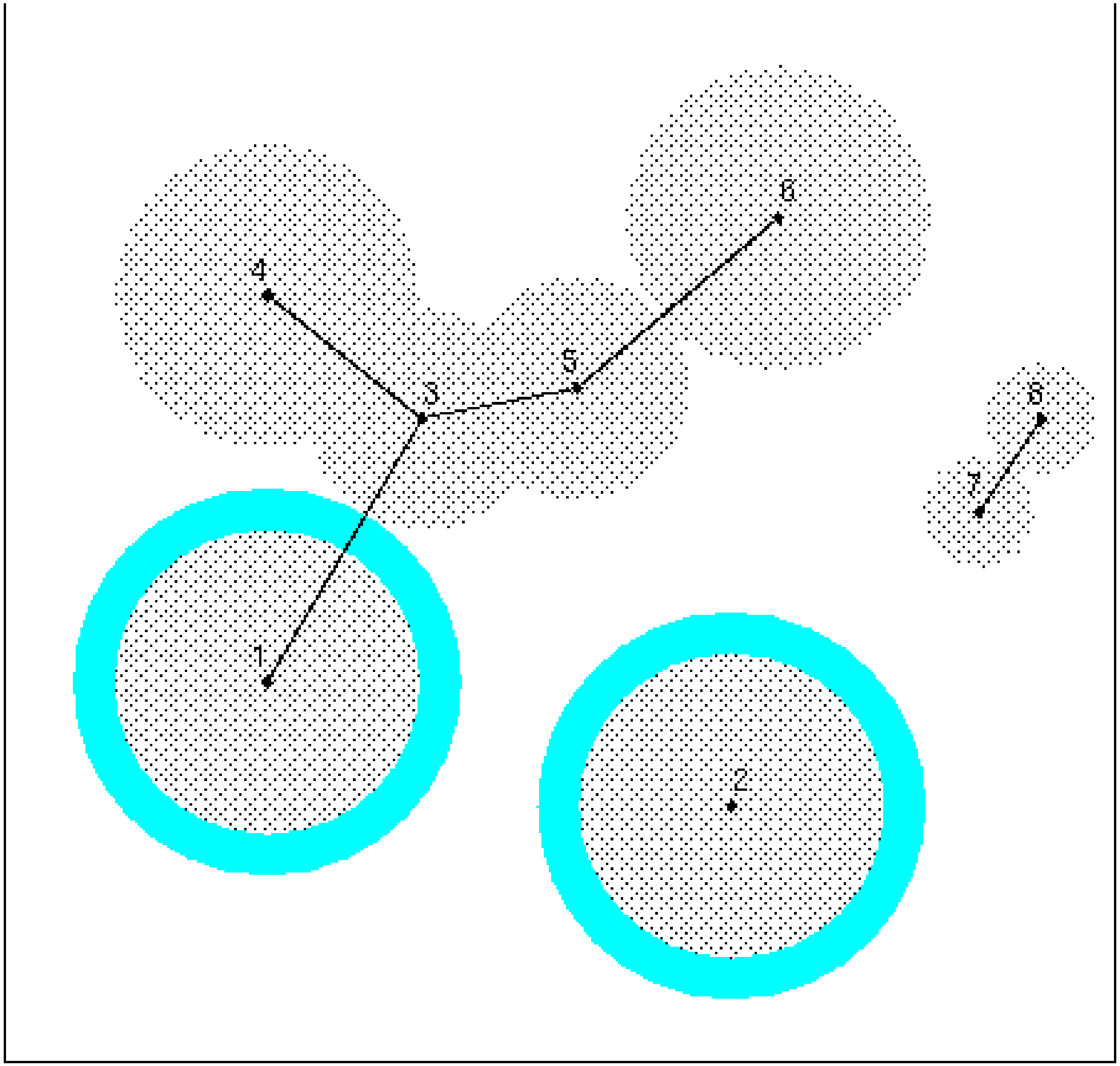
Graphically...



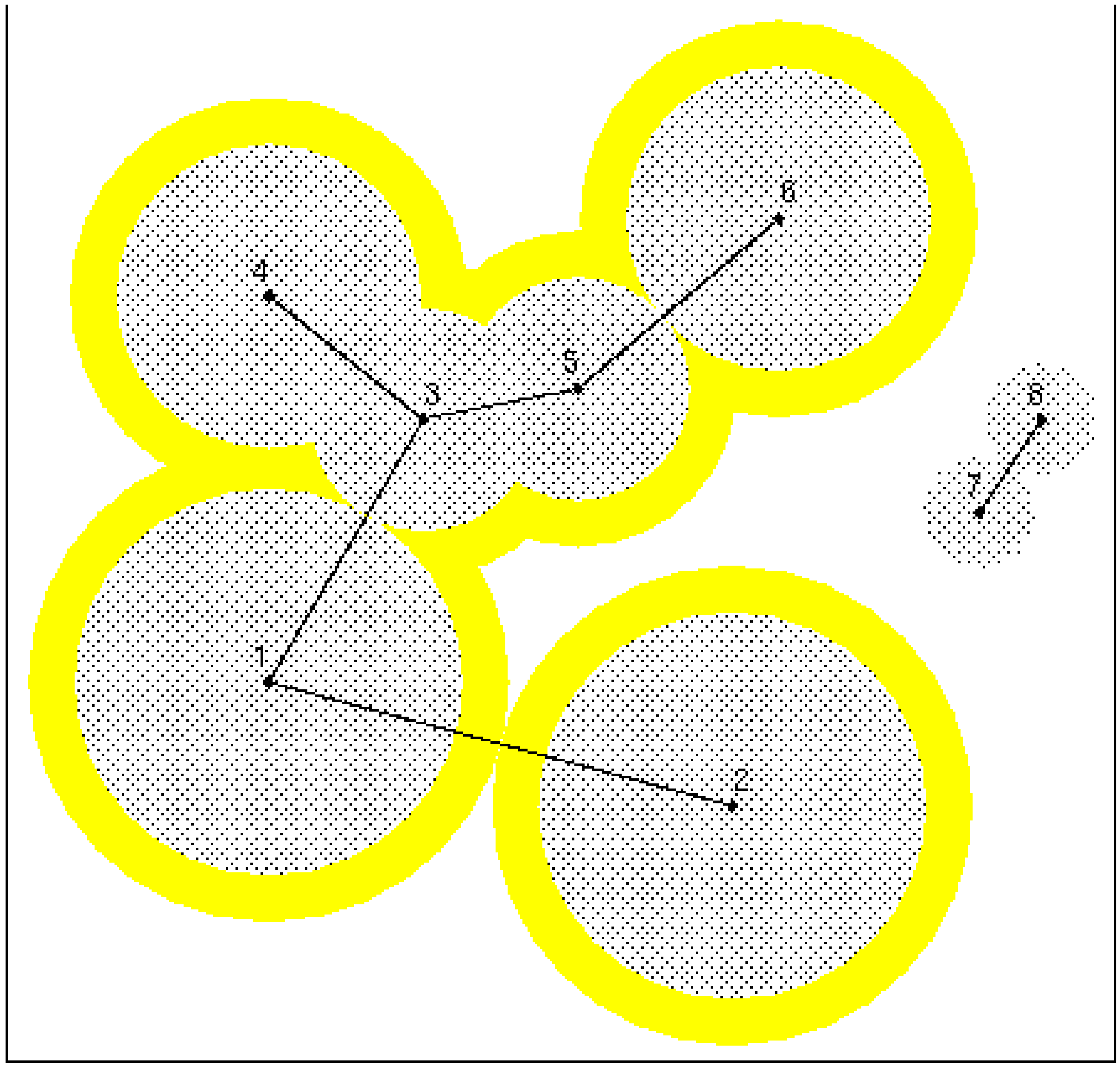
Graphically...



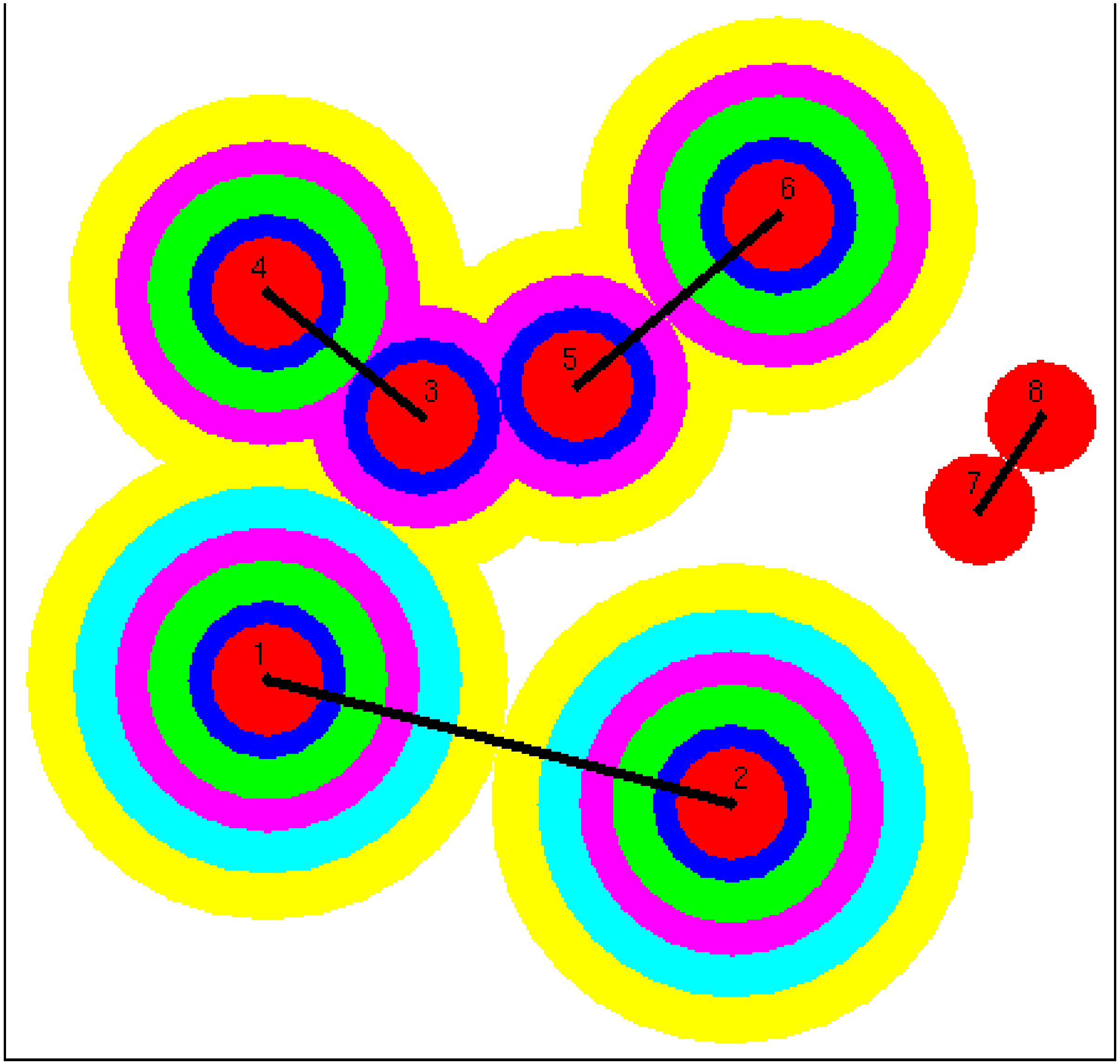
Graphically...



Graphically...

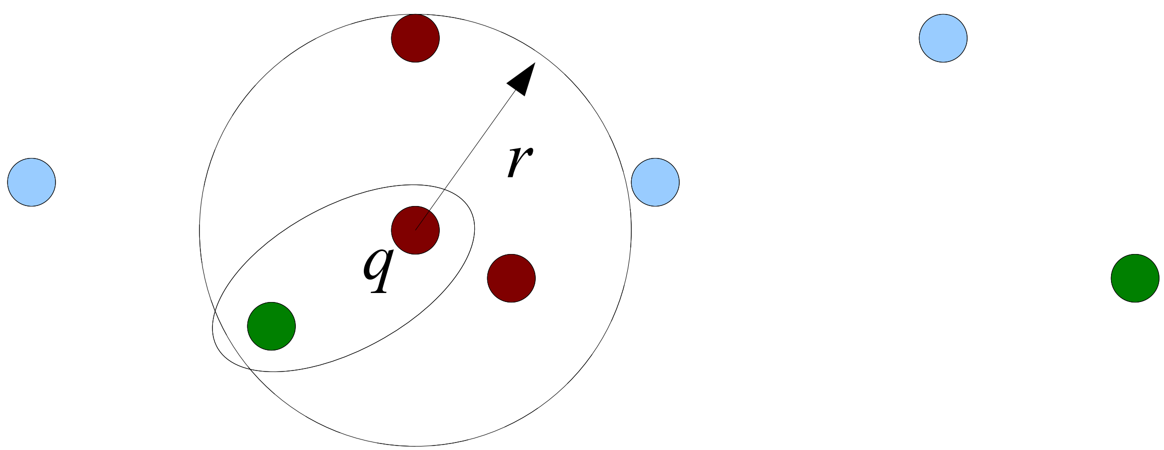


Graphically...



Minimum-Weight Matching

- Running time can be improved by use of an Approximate Multichromatic NNS data structure
- Question: Given a set of n coloured points X, a number r, and a coloured query point q, is there a point in X with colour different from q, which is $r(1 + \varepsilon)$-close to q?



Minimum-Weight Matching

- Solution: Use a set of $2[1 + \log n]$ Approximate NNS data structures
  - $N_i(b)$ is a $(1 + \varepsilon)$-NNS structure for all points whose colours have bit b in ith position
  - q has colour C, C_i is ith bit of C
  - Search for (approximate) nearest neighbour of q in each $N_i(1 - C_i)$
  - Return closest such neighbour