

Walking and Matrix-based Algorithms

Algorithmic Problems Around the Web #4

Yury Lifshits
<http://yury.name>

CalTech, Fall'07, CS101.2, <http://yury.name/algoweb.html>

1 / 18

Outline

- 1 Nearest Neighbors via Walking
- 2 Matrix-Based Techniques

2 / 18

Part I

Nearest Neighbors via Walking

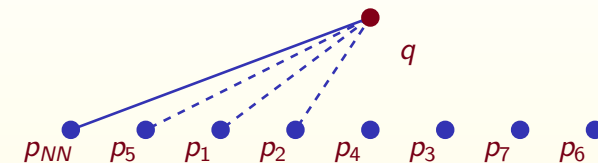
3 / 18

Orchard's Algorithm

Preprocessing:

For every object $p_i \in S$ construct a list $L(p_i)$ of all other objects sorted by their similarity to p_i

Orchard'91



Query processing:

- Start from some random p_{NN}
- Inspect members of $L(p_{NN})$ from left to right
- Whenever meet p' having $d(p', q) < d(p_{NN}, q)$, set $p_{NN} := p'$
- **Stopping condition:** we reached p' having $d(p', q) \geq 2d(p_{NN}, q)$

4 / 18

Hierarchical Orchard's Algorithm

- Randomly choose $S_1 \subset S_2 \subset \dots S_k = S$ with $|S_i|/|S_{i-1}| \approx \alpha > 1$
- Start with Orchard algorithm on S_1
- For every i from 2 to k apply Orchard's algorithm for S_i using result of the previous step as a starting point

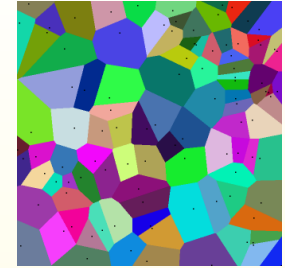
Inspired by classic [skip list](#) technique [Pugh'90](#)

5 / 18

Delaunay Graph Algorithm

Delaunay Graph:

Construct Voronoi diagram for set in Euclidean space. Draw an edge between every two points whose Voronoi cells are adjacent



Search algorithm:

- Start from a random point
- Check all Delaunay neighbors of current object p
- If some p' is closer to q , move to p' and repeat
- Otherwise return p

6 / 18

Delaunay Graph in General

Exercise: prove correctness of the above algorithm

Assume we have general metric space and full matrix of pairwise distances. How Delaunay graph should be defined?

[Navarro, 2002](#): for any distance matrix any two objects can be adjacent :-)

7 / 18

Spatial Approximation Tree: Construction

[Navarro'99](#):

- Set a random object p to be root
- Partitioning technique:
 - Inspect all other object in order by their similarity to p
 - Whenever some p' is closer to p than to any of already chosen children $Ch(p)$ add p' to children set
 - Put every other object p'' to the subtree of closet member of $Ch(p)$
- Recursively repeat

Exercise: prove that covering radius for children subtree is never exceeding covering radius of parent subtree

8 / 18

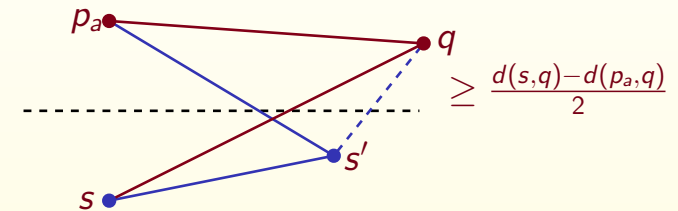
SA-Tree: Search

- Start from the root p
- For every node to be inspected:
 - keep global candidate p_{NN}
(closest object to query visited so far)
 - and p_a — closest to q among
all ancestors and brothers of current node
- Use usual depth-first or best-first tree traversal
- Processing current node t :
 - Compute distances from q to all children of t
 - Go to child s whenever $d(q, s) < d(q, p_a(s)) + 2r_{NN}$

9 / 18

SA-Tree: Correctness

Observation: fix node s , let p_a be its ancestor/brother and s' be some object in its subtree. Then s' is closer to s than to p_a



If there exists s' such that $d(s', q) < r_{NN}$ then $d(s, q) < d(p_a, q) + 2r_{NN}$

10 / 18

Part II

Matrix-Based Techniques

11 / 18

Approximating and Eliminating Search Algorithm

Preprocessing:

Vidal'86

Compute $n \times n$ matrix of pairwise distances in S

Query processing:

- Maintain a set C of candidate objects, initially $C := S$
- For every $p \in C$ keep the lower bound $d_l(q, p)$
- Main loop:
 - Choose $p \in C$ with smallest lower bound, compute $d(q, p)$, update $p_{NN}, r_{NN} = d(q, p_{NN})$ if necessary
 - **Approximating:** update lower bounds in C using $d(q, p') \geq d(q, p) + d(p, p')$ inequality
 - **Eliminating:** delete all elements in C whose lower bounds exceeded r_{NN}

12 / 18

Linear AESA

Advantage of AESA: small number of distance computations
Disadvantages: large storage and non-distance computation

Linear AESA: Micó, Oncina, Vidal'94
Compute $n \times m$ matrix choosing m objects as pivots

Range search:

- Compute all query-pivot distances
- Compute lower bounds for all non-pivot objects
- Eliminate objects with lower bound exceeding search range
- Explicitly check remaining non-pivots

13 / 18

TLAESA

A combination of bisector tree and LAESA

Data structure: Micó, Oncina, Carrasco'96

Usual bisector tree
Additionally, m pivots
Distances from pivots to all objects are precomputed

Query processing

Compute distances from query to pivots
Depth-first/Best-first search in bisector tree
Additional condition to prune subtree of some object s :

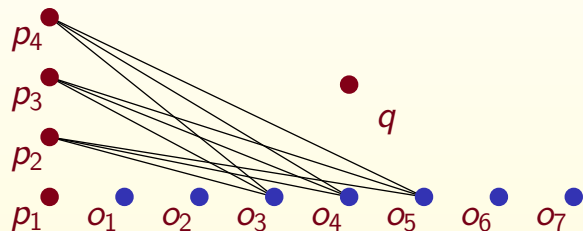
$$\exists i : |d(p_i, s) - d(p_i, q)| \geq r_c(s) + r_{NN}$$

14 / 18

Shapiro's Algorithm (1/2)

Data structure: Shapiro'77

$n \times m$ distance matrix (pivots p_1, \dots, p_m)
Non-pivot objects are sorted by their distances to first pivot $p_1 : o_1, \dots, o_n$

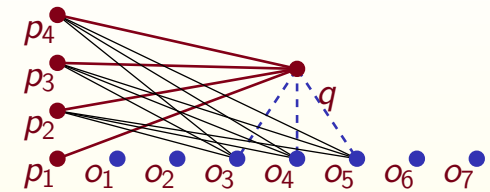


15 / 18

Shapiro's Algorithm (2/2)

Query processing

Compute distances from query to pivots
Start with o_i having $d(p_1, o_i) \approx d(p_1, q)$
Inspect other objects in order $i-1, i+1, i-2, i+2, \dots$
Whenever meet better candidate change the center of inspection
Use flags to avoid double-check
Use all pivots to skip some objects (similar to AESA)
Stopping condition: $|d(p_1, o_i) - d(p_1, q)| \geq r_{NN}$



Actually, it's a mixture of LAESA and Orchard
But published before both: 1977 vs 1991 and 1992!

16 / 18

Thanks for your attention! Questions?

References

Course homepage <http://yury.name/algoweb.html>



P. Zezula, G. Amato, V. Dohnal, M. Batko
Similarity Search: The Metric Space Approach. Springer, 2006.
<http://www.nmis.isti.cnr.it/amato/similarity-search-book/>



E. Chávez, G. Navarro, R. Baeza-Yates, J. L. Marroquín
Searching in Metric Spaces. ACM Computing Surveys, 2001.
<http://www.cs.ust.hk/~leichen/courses/comp630j/readings/acm-survey/searchinmetric.pdf>



G.R. Hjaltason, H. Samet
Index-driven similarity search in metric spaces. ACM Transactions on Database Systems, 2003
http://www.cs.utexas.edu/~abhinay/ee382v/Project/Papers/ft_gateway.cfm.pdf