EasyChair Preprint

№ 2787

Data Pre-process Facilitating Efficient K-NN Queries in Spatial Database

Sheng Zhou and Jonathan Simmons

February 26, 2020

Data Pre-process Facilitating Efficient K-NN Queries in Spatial Database

Sheng Zhou[*1] and Jonathan Simmons [†1]

[1] Data Science and Analytics Team
Corporate Data Office, Ordnance Survey

21-24 April, 2020

Summary

This paper presents 2DMAX, a method to improve performance of large-scale repeated K-NN queries. Distances from a source point to its nearest neighbours are pre-computed and stored to facilitate re-use of query results for multiple queries without additional database access. For certain application scenarios this method may offer performance improvement up to one magnitude over conventional methods.

KEYWORDS: K-NN, nearest neighbour query, spatial query, spatial database, big data process.

1. Introduction

The K-NN (K nearest neighbours) query is one of the most frequently performed spatial database queries. Single K-NN query is generally regarded as a solved problem and most spatial database servers have been highly optimised to handle K-NN queries efficiently via utilisation of appropriate spatial indices such as R-Tree (e.g. Roussopoulos N et al). However, for certain use cases it is still possible to further improve K-NN query performance, for example:

- Generate an inventory of 6-nearest schools for all 28.957 million GB addresses;
- Real-time updates of 5-nearest restaurants for a user on the move

In this paper we focus on point-point queries although some techniques presented here may be extended to support linear and areal objects and queries.

In K-NN queries the range of spatial search to be performed is unknown prior to query and depends on query location and data distribution. The search process must find at least *k* data points, and ensure no other data points can be closer to the query location than the found *k* points.

1.1. K-NN of queries in proximity

Given a data point set P_i and a query location Q_0 (**Figure 1A**) with known K-NN range of R_P (i.e. its K-NN is within the circle C_P with radius R_P centred at Q_0, K = 6, and for simplicity Q_0 is coincident to a data point P_0), for a new query Q (**Figure 1B**) which is D_{PQ} away from Q_0, the K-NN of Q is within the circle C_Q with radius $R_Q = R_P + D_{PQ}$ centred at Q. A spatial query Distance(x, Q) $\leq R_Q$ will guarantee to retrieve at least K objects (i.e. K-NN of Q_0). If more data points are retrieved, simple distance comparison will find the nearest K data points for Q (Sankaranarayanan J et al 2007).

1.2. The 2DMAX Method

[*] Sheng.Zhou@os.uk
[†] Jonathan.Simmons@os.uk

The above observation may be extended to a query point set QS={Q_1| Distance(Q_1, Q_0) ≤ D_{PQMAX}}. In this case (Figure 2), the K-NN of any Q_1 is within the circle C_{PMAX} with radius $R_{MAX} = R_P + 2D_{PQMAX}$ centred at Q_0. Consequently, if we make a single spatial query to retrieve all data points in C_{PMAX} as the candidate result set S_q, the K-NN of any Q_1 may then be obtained from S_q without further access to the database server.

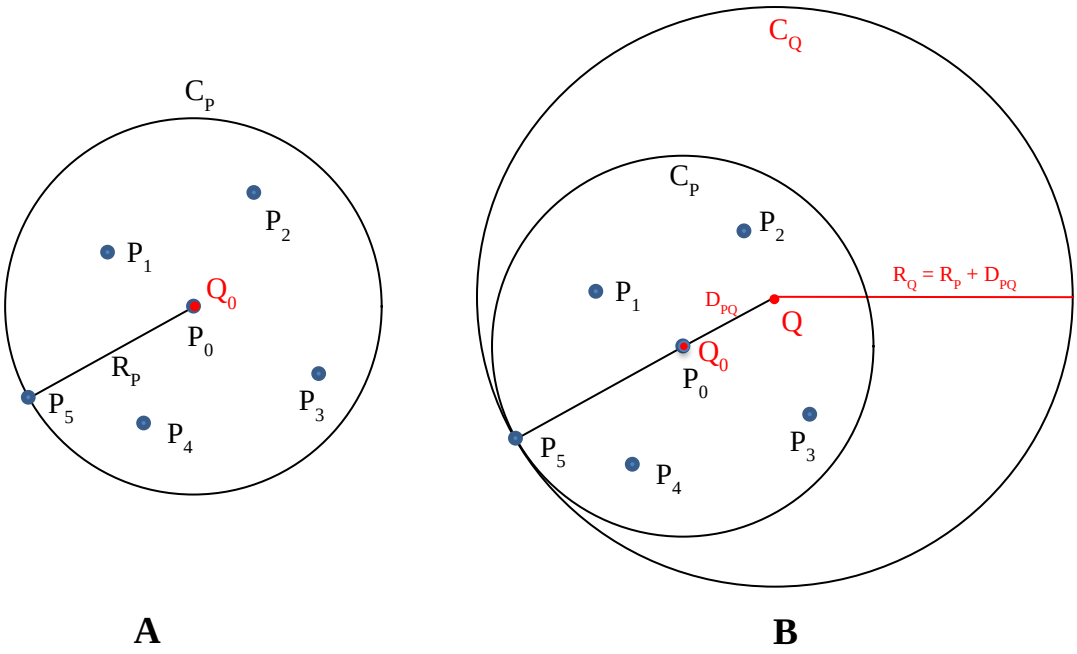

Figure 1 K-NN of query locations in proximity

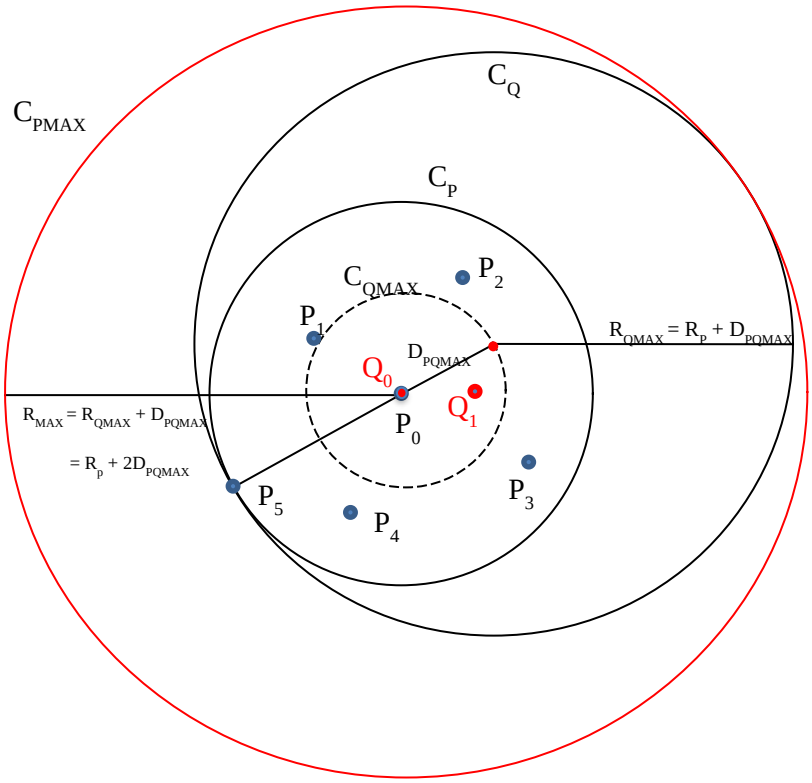

Figure 2 Q_0 with known K-NN and adjacent query point Q_1 in general position

Query space may be partitioned into regions in some way (e.g. using Voronoi diagram). Each region is associated with a query pivot Q_0 (conveniently but not necessarily coincident to a data point) and multiple R_P for various K. All query points falling into a region forms the QS of that region as described above, and the K-NN of these query points may then be computed accordingly. We refer to this approach as the 2DMAX method.

2. Efficient Large-scale K-NN Computation Using 2DMAX method

In this section we present several strategies for computing K-NN (K > 1) in data point set **S** for a query set **Q** under different circumstances.

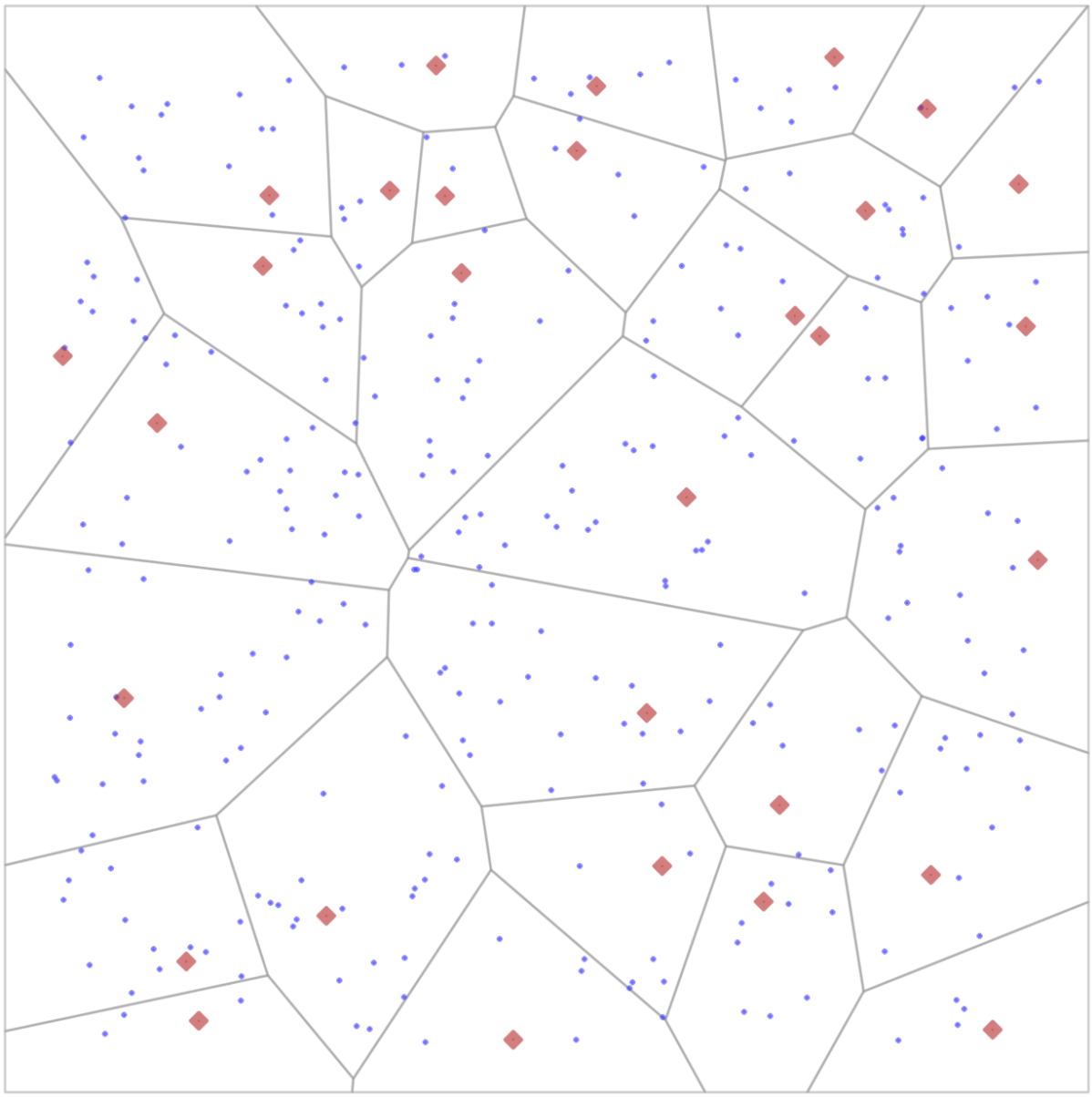

Figure 3 Query points (blue square), source points (large red diamond) and their Voronoi polygons

2.1 Small S, Large Q, Computed K-NN

This is the scenario in the 6-nearest school use case. The source dataset (about 24k schools) is small compared to query points (near 29 millions addresses).

Stage 1: Data point pre-process

- Construct Delaunay triangulation DT(S) on points in S;
- For point $\mathbf{s_0} \in S$, find its K-1 nearest neighbour $\mathbf{s_{i=2,\,K}}$ and store $\mathbf{R_{p\_i}}$ = Distance($\mathbf{s_0}$, $\mathbf{s_i}$)
  - given $\boldsymbol{k}$ and point $\mathbf{s_0}$ in DT
    - Find points $\mathbf{s_{1j}}$ connected to $\mathbf{s_0}$ by DT edges, with maximum distance to $\mathbf{s_0}$ as L_{max}
      - Search all $\mathbf{s_{1j}}$ to find connected points, count the number of points within L_{max} from $\mathbf{s_0}$
      - New points within $\mathbf{L_{max}}$ from s_0 will be searched recursively for connected points
    - If the number of current within L_{max} point $\geq \boldsymbol{k}$, return and compute R; otherwise, use the maximum distance of all current points to $\mathbf{s_0}$ as new L_{max} and repeat the previous two steps.
  - Multiple $\mathbf{R_p}$ are stored to support queries of various $\boldsymbol{k} \leq K$
- Construct Voronoi diagram VD(S) of S from DT(S) where VDC(**s**) is the Voronoi polygon associated with point $s \in S$ (Figure 3). Here VDC(s) is the partition and **s** is the query pivot.

Stage 2: Query point set partition and K-NN computation:

- Given $K_q \leq K$, for each point $\mathbf{s} \in S$, retrieve polygon VDC(s) and select query point set Q_s = {$\mathbf{q}|\mathbf{q} \in Q$ and VDC(**s**).Contains(**q**) = True}
- If $K_q = 1$ (1-NN, the nearest neighbour), **s** is the nearest neighbour for all points in Q_s.
- if $K_q > 1$, retrieve the corresponding R_p stored in stage 1, compute D_{PQMAX} = Max(Distance(**q**, **s**)) and perform a spatial query to retrieve candidate result set S_q={$\mathbf{s_i}|\mathbf{s_i} \in S$ and Distance($\mathbf{s_i}$, **s**) $\leq R_p + 2D_{PQMAX}$}.
  - For each query point **q** in Q_s, compute its distances to all points in S_q to find its nearest K_q-NN. Note that the two-point K-NN result may be used here if necessary to (potentially) further speed up the process (i.e. any P for Distance(Q, P) > $R_p + 2D_{PQ}$ may be disregarded without comparison).
  - Max(Distance(**s**, $\mathbf{v_{vdc}}$)) may be used as D_{PQMAX} where $\mathbf{v_{vdc}}$ are vertices of the Voronoi polygon VDC(**s**). This will eliminate a lot of distance computation (make sense if it is expensive) but may result in an increased query range (more points in S_q).

Note that the second stage of process is inherently parallel and suitable for distributed computation.

2.2 Small S, Multiple Real Time Queries

In this scenario of moving query point, the query points are unknown prior to query so we have to use Max(Distance(**s**, $\mathbf{v_{vdc}}$)) as D_{PQMAX}. Subsequently the candidate result set S_q may be selected in the same manner as stage 2 in 2.1.

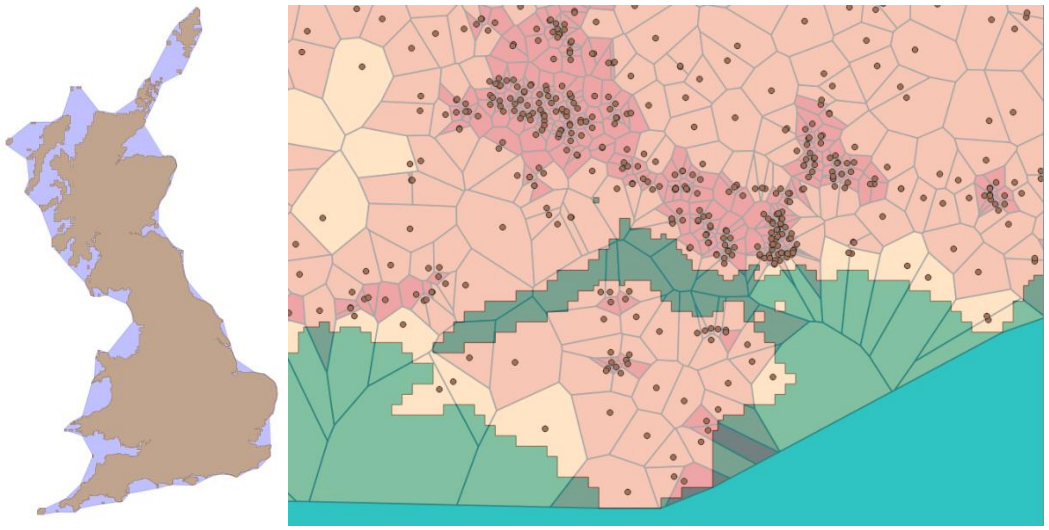

Figure 4 Voronoi polygons clipped by the concave hull (overlaying gridded coastline)

For the entire duration while the query point (e.g. a vehicle) is inside a VDC(**s**), the candidate result set S_q will be retrieved once only to compute multiple K-NN in real time.

Due to the manner the Voronoi polygons are generated, on the boundary of the dataset the D_{PQMAX} distance may become (unnecessarily) very large, resulting in many irrelevant data points being retrieved. A simple solution is to use concave hull to clip the boundary Voronoi polygons to reduce their size and subsequently D_{PQMAX} (**Figure 4**).

2.3 Large S, very large Q

In this scenario similar to Lu et al (2012), S is also very large (e.g. find 6 nearest lamp posts for all households in GB). It is not feasible to create VD for the whole source dataset S and a sampling process is required.

- Assuming datasets are stored in an indexed spatial database, given a sample rate $\mathbf{r} \in (0.0, 1.0)$, n = S.size()***r** points will be randomly selected from S to form a sample set S_1.
- For each point **s** in S_1,
    - its K-1 nearest neighbours are discovered via conventional K-NN query on S and R_{p_i} (**i** = 2, K) are calculated;
    - Select query points falling into VDC(s) to generate query subset Q_s and compute D_{PQMAX};
    - Select source points into a sub source set $S_s = \{\mathbf{s_i} \mid \mathbf{s_i} \in S \text{ and } Distance(\mathbf{s_i}, \mathbf{s}) \leq R_{p\_K} + 2D_{PQMAX}\}$.
    - Now the K-NN of every point in Q_s are contained in S_s. We may then apply the algorithm in 2.1 to Q_s and S_s. If S_s is still too large, additional levels of the same sampling process may be performed on S_s.

If size of S is comparable to Q, improvement may not be significant. Nevertheless, this method may be useful for data partition in a parallel computing environment.

3. Experiment results

We performed experiments of 2DMAX-based KNN on the 6-nearest school problem. The number of addresses is 28,597,000 and the number of schools is 23,855. The programme is written in Java connected to a local PostGIS server via JDBC.

Total process time is 25.9 hours on a ThinkPad W530 laptop with 16GB RAM and Intel Core i7-3820QM processor (in addition to about a minute to pre-process school data, compute distances from

each school to 5 nearest other schools and generate Voronoi diagram for all schools). This equals to about 3.22 seconds per 1000 addresses.

In comparison, single query making use of pre-process results (as shown in **Figure 1-B**) on 100 random selected Voronoi diagram cells performs at 19.43 seconds per 1000 addresses. It is 28.37 seconds for PostGIS' own KNN (using 'order by' and 'limit').

We also tested a pure SQL solution on one region with 681 addresses, the result equals to 52.81 seconds per 1000 addresses (which is a surprise, because the process runs on the server side only).

4. Summary

For application scenarios described in this paper, the 2DMAX approach significantly improves the performance of batched process of large query set (our main concern) by near one magnitude. Clear (but small) improvements is observed on single query over built-in query optimiser. This method may be extended to higher dimensional and non-spatial data.

Remarks

A patent application (19183722.8 - 1222) based on this research has been submitted to European Patent Office.

5. Biography

Sheng Zhou is a Senior Data Scientist at Ordnance Survey. He obtained his PhD in GIS/Urban Planning from Cardiff University. Prior to joining OS Research, he participated in research projects on multiscale spatial database and multi-agent system for active maps at Glamorgan and Cardiff universities. His research interests include machine learning algorithms and applications on spatial data; computational geometry; spatial databases; spatial analysis; map generalisation.

Jonathan Simmons is the Head of Data Science and Analytics at Ordnance Survey Data Office.

References

Lu W, Shen Y, Chen S and Ooi BC (2012). Efficient Processing of k Nearest Neghbor Joins using MapReduce. *Proceedings of the VLDB Endowment*, 5(10).

Roussopoulos N, Kelley S, Vincent, F D R. (1995). Nearest neighbor queries. *Proceedings of SIGMOD '95*. p.71

Sankaranarayanan J, Samet H and Varshney A (2007). A fast all nearest neighbour algorithms for applications involving large point-clouds. *Computers & Graphics*, 31, 157-174.