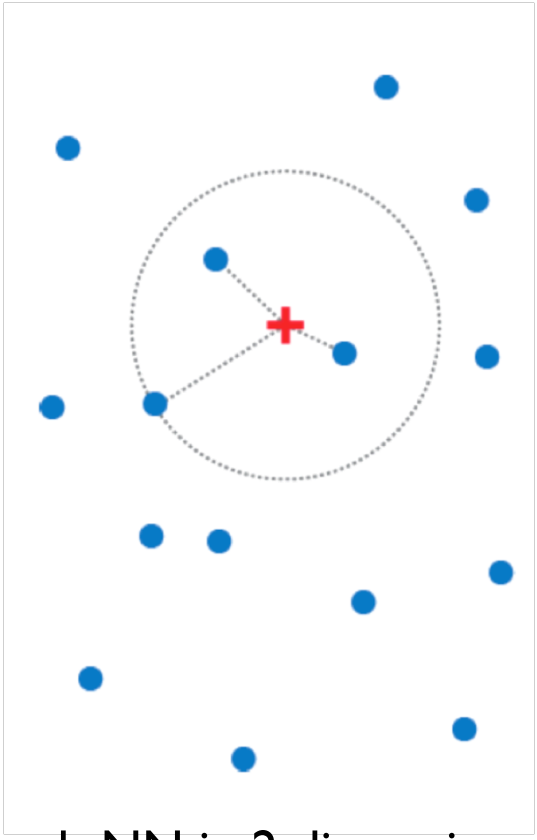
Concurrent Implementation of k-NN for WLAN Positioning

Eros Gulo
Department of Earth and Space Science, York University
October 13, 2015

K–NEAREST NEIGHBOURS

- Search a set of reference points for the k-nearest points to a query point and repeat for successive query points.
- Nearness or proximity can be arbitrarily defined, though usually a metric of distance (e.g. Euclidean distance) is used.
- The query point is classified or assigned a value by its nearest neighbours.



k-NN in 2 dimensions
(k = 3)

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY

K–NEAREST NEIGHBOURS

Reference Set: $R = \{r_1, r_2, \ldots, r_m\} \quad \mathbb{R}^d$

Query Set: $Q = \{q_1, q_2, \ldots, q_n\} \quad \mathbb{R}^d$

Distance computed for each possible pairing: $D(q_i, r_j) \quad j \in [1, m] \quad i \in [1, n]$

Complexity required: $O(nmd)$

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY

K–NEAREST NEIGHBOURS

Distances for each q_i are sorted and the r_j corresponding to the *k* smallest distances are selected as the nearest neighbours.

This portion of the algorithm requires an additional complexity of $O(nm \log m)$.

K-NN IN WLAN POSITIONING

WLAN Signal Fingerprint Matching (WSFM)

- Reference set is a database or map of the signal strength of all WLAN access points (APs) at all locations in the positioning area.
- Query point is a sample of the signal strengths of all APs (fingerprint) at user's particular location.
- k-NN used to match the fingerprint to the most similar locations (based on AP signal strength) in the database.

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY

K-NN IN WSFM

Reference Set: $L = \{\ell_1, \ell_2, \ldots, \ell_m\} \quad \mathbb{R}^d$

Query Set: $F = \{f_1, f_2, \ldots, f_n\} \quad \mathbb{R}^d$

Distance computed for each possible pairing: $D\left(f_i, \ell_j\right) \quad j \in [1, m] \quad i \in [1, n]$

Number of APs: d^*

* *d* may vary between different locations. AP visibility not guaranteed!

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY

PREVIOUS WORK

Reducing the Amount of Distance Calculations:

- Using a kd-tree to split up the reference points
- Locality Sensitive Hashing
- Other pre-processing of reference point data

These are all incompatible with the varying dimensions of the reference and query points. In addition, the method proposed in this paper was shown to greatly outperform them.

PROPOSED CONCURRENT K-NN ALGORITHM

The distance calculations between different r_j and q_i pairings are all independent of each other, therefore, computation time can be reduced if they are computed in parallel.

Sorting of all the distances for each q_i does not have to be fully completed, therefore, computation time can also be reduced by only sorting some of the distances. The distances are only sorted until the first k-items in the list are the k-smallest and they are in order.

PSEUDOCODE FOR THE INITIALIZATION OF THE DISTANCE CALCULATION AND SORTING THREADS

```
1  DistancesMatrix[] ← new m x n matrix
2  IndexMatrix[] ← new m x n matrix
3  SemaphoreArray[] ← new n length array
4  for i ← 1 to n
5       SemaphoreArray[i] ← new semaphore with initial value of 0
6
7       for j ← 1 to m
8            IndexMatrix[j,i] ← j
9            new Thread{
10                        DistanceMatrix[j,i] ← calculateDistance(j,i)
11                        V( SemaphoreArray[i] )
12                        terminate thread
13                   }
14
15      new Thread{
16                   for j ← 1 to m
17                        P( SemaphoreArray[i] )
18
19                   modifiedInsertionSort(DistanceMatrix[1 to m,i],
20                                         IndexMatrix[1 to m,i], k)
21                   terminate thread
22              }
```

LASSONDE SCHOOL OF ENGINEERING

YORK UNIVERSITÉ UNIVERSITY

PSEUDOCODE FOR THE MODIFIED INSERTION SORT

```
1  modifiedInsertionSort(D[],I[],k)
2  {
3        for i ← 2 to k
4             key ← D[i]
5             index ← I[i]
6             j ← i − 1
7             while (j > 0) and (D[j] > key) do
8                  D[j+1] ← D[j]
9                  I[j+1] ← I[j]
10                 j ← j − 1
11            D[j+1] ← key
12            I[j+1] ← index
13       for i ← k+1 to m
14            if (D[i] < D[k])
15                 key ← D[i]
16                 index ← I[i]
17                 j ← i − 1
18                 while (j > 0) and (D[j] > key) do
19                      D[j+1] ← D[j]
20                      I[j+1] ← I[j]
21                      j ← j − 1
22                 D[j+1] ← key
23                 I[j+1] ← index
24 }
```

LASSONDE SCHOOL OF ENGINEERING

YORK UNIVERSITÉ UNIVERSITY

CONCLUSION

Computation time for k-NN algorithm is cut down by parallelizing distance calculation and minimizing the number of distances sorted.

This modified k-NN algorithm will be implemented in the context of my WSFM positioning algorithm.

Other sorting algorithms will be studied, along with an alternate method of parallelization of distance calculations and sorting.

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY

END OF PRESENTATION

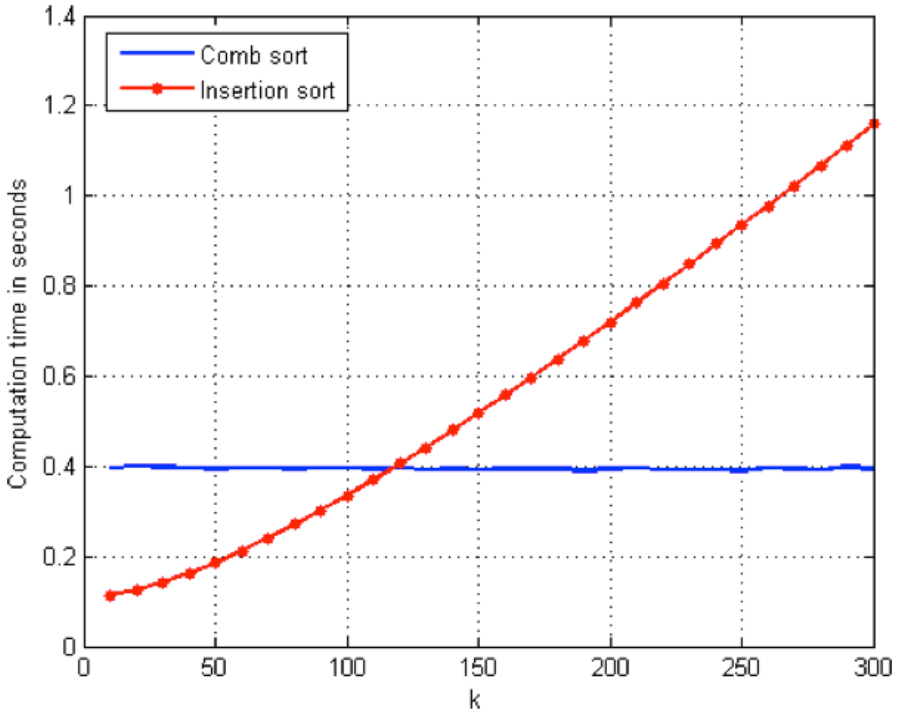
Thank you for your attention.

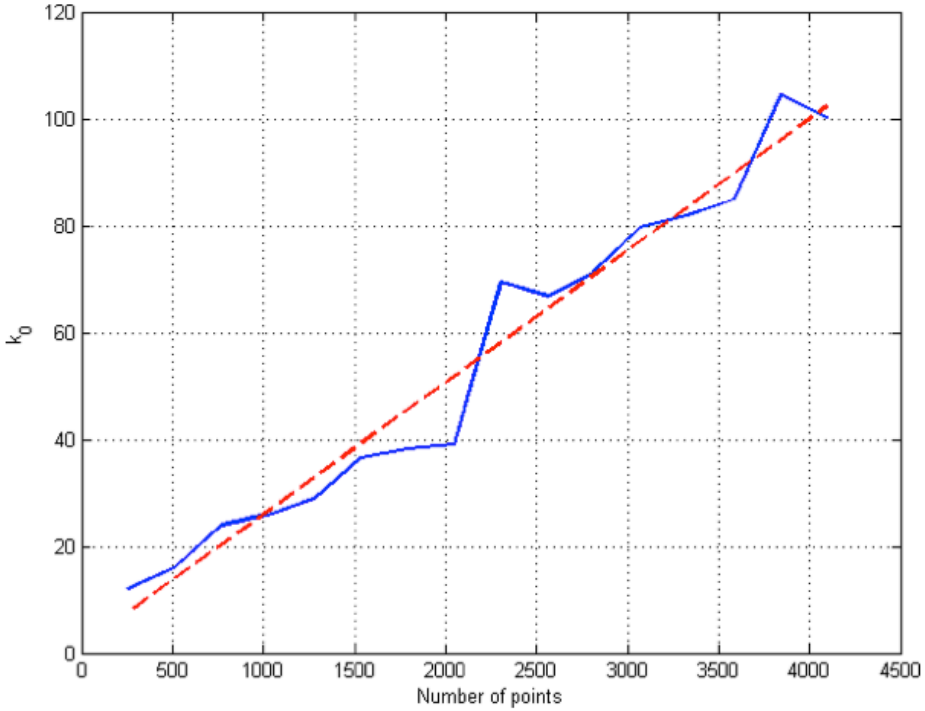
Feel free to ask any questions you may have.

V. Garcia, E. Debreuve, and M. Barlaud. Fast k Nearest Neighbor Search Using GPU. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, Hong Kong, June 2008. IEEE.

LASSONDE SCHOOL OF ENGINEERING | YORK UNIVERSITÉ UNIVERSITY

COMB SORT VS INSERTION SORT





Modified version of insertion sort is faster than comb sort for small values of k.

LASSONDE SCHOOL OF ENGINEERING

YORK UNIVERSITÉ UNIVERSITY