Cairo University

Faculty of Engineering Credit Hours System

CMPS401

Advanced Database Systems

Project Proposal

Prepared by:

● Abdelrahman Mohamed Ezzat	1190158
● Abdelrahman Ibrahim Mohamed	1190493
● Amr Ahmed Abdelzaher	1190074
● Joseph Ameer Aziz	1190052

DiskANN-based index

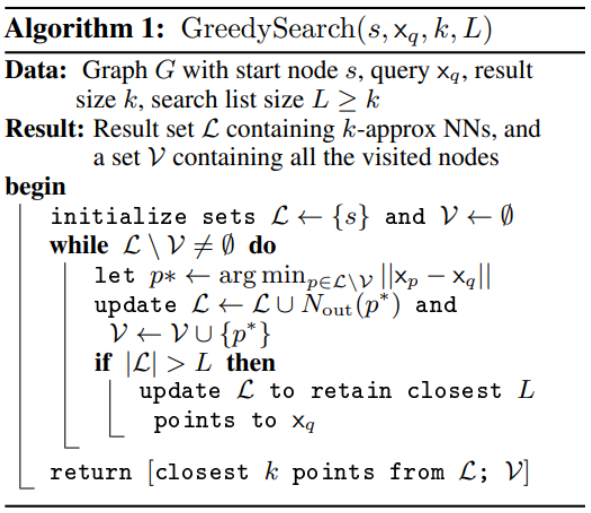
We are going to implement an approximate nearest neighbour algorithm based on DiskANN[1], which uses Vamana data structure. This is a graph-based index that uses Greedy Search for retrieving the top **“k”** approximate nearest neighbours to a given vector embedding. The algorithm was modified in multiple papers but we are referencing the original paper published in 2019.

The algorithm optimizes over brute force comparison that it compares the query with much fewer nodes.

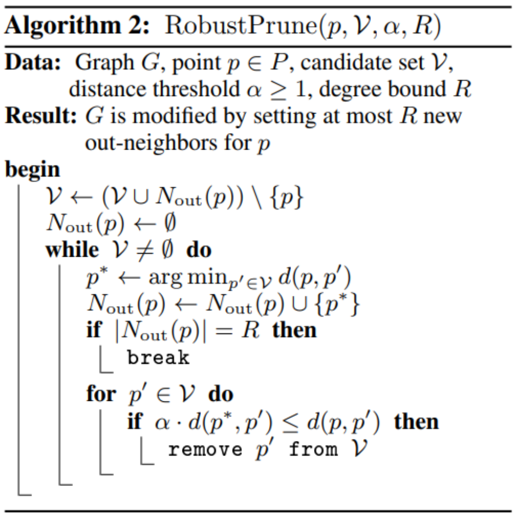
The algorithm at the very beginning constructs a random R-regular directed graph, which means that the out-degree of each node is at most R. Each node represents an embedding and the edges represent how close the node is to its neighbours.

Graph Construction

1. We initialize a graph where each node has R randomly chosen out-neighbours.
2. We chose the vertex s (the medoid of the dataset) which will be the starting node for the search algorithm.
3. We iterate on all the points in a random order and then update the graph to make the greedy search converge faster.
4. We run Greedy Search from every point with (k = 1) and then we apply a pruning algorithm to maintain the R out-degree property.



```
Algorithm 1: GreedySearch($s, x_q, k, L$)
Data: Graph $G$ with start node $s$, query $x_q$, result
      size $k$, search list size $L \geq k$
Result: Result set $\mathcal{L}$ containing $k$-approx NNs, and
        a set $\mathcal{V}$ containing all the visited nodes
begin
    initialize sets $\mathcal{L} \leftarrow \{s\}$ and $\mathcal{V} \leftarrow \emptyset$
    while $\mathcal{L} \setminus \mathcal{V} \neq \emptyset$ do
        let $p* \leftarrow \arg\min_{p \in \mathcal{L} \setminus \mathcal{V}} ||x_p - x_q||$
        update $\mathcal{L} \leftarrow \mathcal{L} \cup N_{\text{out}}(p^*)$ and
         $\mathcal{V} \leftarrow \mathcal{V} \cup \{p^*\}$
        if $|\mathcal{L}| > L$ then
            update $\mathcal{L}$ to retain closest $L$
             points to $x_q$
    return [closest $k$ points from $\mathcal{L}$; $\mathcal{V}$]
```



```
Algorithm 2: RobustPrune($p, \mathcal{V}, \alpha, R$)
Data: Graph $G$, point $p \in P$, candidate set $\mathcal{V}$,
      distance threshold $\alpha \geq 1$, degree bound $R$
Result: $G$ is modified by setting at most $R$ new
        out-neighbors for $p$
begin
    $\mathcal{V} \leftarrow (\mathcal{V} \cup N_{\text{out}}(p)) \setminus \{p\}$
    $N_{\text{out}}(p) \leftarrow \emptyset$
    while $\mathcal{V} \neq \emptyset$ do
        $p^* \leftarrow \arg\min_{p' \in \mathcal{V}} d(p, p')$
        $N_{\text{out}}(p) \leftarrow N_{\text{out}}(p) \cup \{p^*\}$
        if $|N_{\text{out}}(p)| = R$ then
            break
        for $p' \in \mathcal{V}$ do
            if $\alpha \cdot d(p^*, p') \leq d(p, p')$ then
                remove $p'$ from $\mathcal{V}$
```

K Top ANNs Retrieval

We use Greedy Search with the given k after constructing the graph correctly. We also use cosine similarity for distance comparison.

References

[1] DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node: https://proceedings.neurips.cc/paper_files/paper/2019/file/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Paper.pdf