D-011

Integrated Group Nearest Neighbor($IGNN$) Query Algorithm on Road Network Distance

TIN NILAR WIN* **HTOO HTOO*** **YUTAKA OHSAWA***

1.Introduction

This paper studies a new query type, named integrated group nearest neighbor ($IGNN$) query, which can be regarded as a kind of the group nearest neighbor (GNN) query [3]. The $IGNN$ query aims to find the best target point t ∈ T and a subset CS of data point set C, satisfying the condition that the aggregate distance from t to all query points and the points in CS is minimal. The $IGNN$ query applies to many real world problem settings. Suppose five medicine companies want to organize a product promotion fair in a city and wish to invite 30 local pharmacy stores closest to the promotion venue in that city. The $IGNN$ query can help to reserve the suitable venue that minimizes the total travel distance for all companies and a selection of stores. In this paper, we attempt $IGNN$ query using simple materialized path view (SMPV) structure with pre-computed distance data. Our approach visits only the qualified nodes that may contain data points comparing with the original pruning Dijkstra's method (OPD), which follows the fast pruning approach [2], so that our algorithm achieves the better performance and reduces the processing time drastically.

2.Related Work

The Nearest Neighbor (NN) query and its variant queries on road network have been studied since last few decades. One different variation of NN query is group nearest neighbor query (GNN) [3] which finds the k objects with the minimum total travel distance to a group of query points. These algorithms are based on R-tree index structure and progressively search the GNN by computing Euclidean distance. Another extended type of group query is the aggregate nearest neighbor query (ANN). Sun et al.[2] introduced a kind of ANN query called MANN query in road networks to find the optimal aggregate point for a merged set that consists of the given query point set and selected candidate set. The previous works are achieved with compute-on-demand approach. Such types of spatial query can perform efficiently using the materialized precomputed data (i.e., SMPV) [1] as an alternative approach eliminating the computation on the fly.

3.Preliminaries

IGNN query for road network is introduced in Definition 1.

Definition 1 ($IGNN$)**.** *Given a set of target points T, a set of query points Q and a set of data points C. The $IGNN$ query retrieves a target point $t \in T$ which is the nearest to the centroid point q_c and a set of k candidate points (CS) from C with minimum aggregate distances to target point t i.e.,* $q(T,Q,C,k)=(t,CS)|t \in T \wedge CS \in f(t,C,k)$ *such that* $dis_{agg} f(t, CS \cup Q) < dis_{agg} f(t', CS' \cup Q), \forall t', \forall CS'$

where $f(t,C,k)$ is a function which returns a subset of data point set $CS \subseteq C$ that satisfy the condition $dist_{agg}(t,t.CS) \leq dist_{agg}(t,C) \wedge |Q|>0 \wedge |CS| = k$.

Definition 2 (Pruning a target point)**.** *Let $T = \{t_1, t_2, .., t_n\}$, $Q = \{q_1, .., q_m\}$ and q_c be an arbitrary dummy point in space. If* $dist_E(t_1, q_c) \geq \frac{(dist_{agg}(t_2,t_2.CS))}{(|Q|)}$, *it confirms that t_1 cannot be the best target point, as can be seen from the comparison to t_2, where $dist_{agg}$(t,t.CS) is the minimum aggregate distance between the target point and its nearest data points.*

The same idea will apply for pruning intermediate nodes in the R-tree. The node N can be pruned if: $dist_{min}(N, q_c) \geq \frac{\sum_{q_i \in Q} dist(q_i,t)+dist_{agg}(t,t.CS)}{|Q|}$

4.IGNN query Processing on SMPV Structure

4.1.Processing centroid point

The materialized centroid point method answers the $IGNN$ query by a single dummy point approach. We assume that a query point q_i is located at $<x_i, y_i>$. We can thus compute a centroid dummy point q_c of Q which is located at $q_c = <\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}>$ as an approximate dummy point.

4.2.Materialized centroid point (MCP) algorithm

There are two solutions to process the $IGNN$ query, namely OPD and MCP. The OPD algorithm is mainly based on fast pruning approach [2] and processed by traditional compute on demand approach. The MCP algorithm is designed to compute using pre-computation approach. The target point set is indexed by an R-tree in both approaches. The data point set is constructed as a cell-points relationship table in advance that is utilized only for MCP algorithm. Generally, both algorithms mainly have two phases. In first phase, the algorithm computes the minimum pruning distance d which propagates the aggregate distance corresponding to the current result of the subset of data point set. The second step is to prune away the target points based on Definition 2 and updates the pruning distance d.

The pseudo code and the detailed explanation of MCP algorithm in Algorithm1 are depicted below. In first phase, the Algorithm1 is initialized with searching the centroid dummy point q_c of Q. This dummy point retrieves its nearest target point $q_c.NN$ by applying an incremental NN query. We then retrieve the candidate result (t.CS) corresponding to $q_c.NN$ based on the Algorithm 2, and assign the pruning distance d_N and the current result.

Next, the Algorithm1 figures out the target points. The target points are visited based on the ascending order of their mindist to q_c. To be more specific, it visits the nodes of R-tree that index all the target points with the best first order, with the help of the min-heap PQ. The intermediate node of R-tree will be pruned based on Definition 2. Note that e is the entry within PQ that has the smallest mindist to q_c. The algorithm will evaluate e. If e is leaf node and then the candidate result of e will be retrieved. If the result of e is better than the current result, the pruning distance d_N and the result GP will be updated. When the mindist of e is greater than $\frac{d_N}{|Q|}$, the algorithm will terminate.

4.3.Searching candidate result on SMPV

Algorithm 2 describes the searching phase of the candidate result of the target t from C set on the SMPV structure which can retrieve proficiently compare with the on-fly approach (OPD). Firstly, a cell (subgraph) in the SMPV structure to which target t belongs is determined and the data points belong to the cell are searched. Next, the algorithm will enlarge the neighbor cell until the t.CS contains k candidate points and all the query points in Q have been visited according to the original strategy [2]. In the expanding neighbor cell, the algorithm looks for the

*Saitama University

nearest border point from the current node by referring to the inner to border node distance table (IBDT) of the respective subgraph. The candidate search algorithm will terminate when the set of t.CS contains k candidate points with the minimum aggregate distance to the target point.

Algorithm 1 IGNN

```
1: function MCP((T, Q, C, k, rtp))
2:     PQ ← ∅;
3:     q_c = SearchCentroidQ(Q);
4:     NNRecord t_NN ← rtp.NN(q_c);
5:     PQ.enque(rtp.root, d_min(root, q_c));
6:     IGNNPoint GP = FindCandidateResult(t_NN, Q, C, k);
7:     d_N = GP.dist_agg;
8:     while PQ not empty do
9:         v ← PQ.deQueue()
10:        e = v.entry();
11:        d_E = v.dist();
12:        if dist_E = d_N/|Q| then then
13:            break;
14:        else
15:            if e ≠ null ∧ e = leafnode then then
16:                PQ.enque(e.child, d_min(q_c, child));
17:            else ▷ retrieve the current point's candidate result
18:                p = Point(e.x, e.y);
19:                eGP= FindCandidateResult(p, Q, C, k);
20:                if eGP.dist_agg ≤ dist_N then
21:                    d_N = eGP.dist_agg
22:                    Update(GP);
23:                end if
24:            end if
25:        end if
26:    end while
27: end function
```

Algorithm 2 FindCandidateResult

```
1: procedure FindCandidateResult(t_i, Q, C, k)
2:     IGNNPoint GP ← null
3:     PQ ← ∅, GP.CS ← ∅,GP.dist_agg ← 0.0
4:     GP.point ← t_i
5:     SearchInStartCell(PQ, GP, t_i, k)
6:     while PQ is not empty and |Q| > 0 or |GP.CS|<k do
7:         e ← PQ.dequeue();
8:         if e ∈ Q then then
9:             GP.dist_agg+ = e.dist;
10:            Remove e from Q;
11:        else[e ∈ C and |GP.CS|<k]
12:            GP.dist_agg+ = e.dist;
13:            GP.CS.add(e);
14:        end if
15:        if |GP.CS| < k then
16:            ExpandNeighborCell(PQ, GP, t_i, k)
17:        end if
18:    end while
19:    return GP
20: end procedure
```

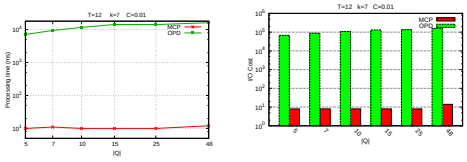
5.Experiments

To evaluate the performance of proposed algorithm, we carried out several experiments on the same PC with Intel Core i7-3770 CPU (3.4 GHz) and 16GB of memory for both algorithms. A road map of Saitama city with 16,284 network nodes and 24,914 network links was used in the experiments. We used two types of precomputed distance tables called: border to border node distance table and inner to border node distance table which require 5.2MB memory space altogether. In addition, the memory requirement of SMPV structure is much less compared to the Hierarchical Encoded Path View (HEPV) [1].

We conducted experiments to evaluate the effect of the number of query points on the search performance. The experimental results confirm that the SMPV approach represents an better compromise between the compute-on-demand and precomputation approaches for processing $IGNN$ query as shown in Fig.1(a) and (b).

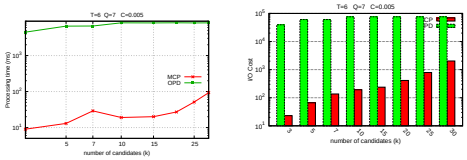
Fig.2(a) and (b) depict the overall performance results of the effect of the k numbers. The outcomes show that both of approaches are directly impact on the numbers of k. Nevertheless, in the proposed algorithm, MCP requires less processing time and less I/O cost compare to the OPD. The reason is that while the search area expands, k value increases and hence the I/O cost leads to increase in the nodes visit.

Fig.3(a) and (b) describe the experimental results with various setting of the target set and query set while the data points' set density is varying and the value of k is fixed to seven. According to the results, the processing time and I/O cost of the MCP gradually decrease when the density increases. This is due to the search area and the node expansion decrease with the increasing density.



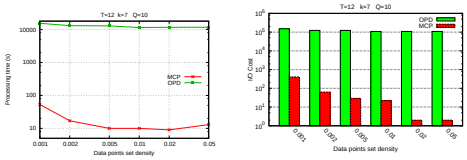
(a) Processing time (b) I/O cost

Fig. 1: The effect of the varying number of query points



(a) Processing time (b) I/O cost

Fig. 2: The effect of the varying number of k



(a) Processing time (b) I/O cost

Fig. 3: The effect of the varying candidate set densities

6.Conclusion

This paper studied the integrated group nearest neighbor ($IGNN$) query using a simple distance materialization approach suitable for LBS. We conducted experiments to compare our simple materialization approach against traditional compute on demand approach. We found that MCP algorithm effectively prunes unnecessary target points and locates the nearest candidate points from C. According to the experimental results, we can conclude that the proposed algorithm is efficient in both processing time and I/O cost.

References

[1] Hlaing, A. T., Win, T. N, Htoo, H., Ohsawa, Y.: RkNN Query on Road Network Distances, Journal of Information Processing Vol.23 No2, pp.163-170(March. 2015)

[2] W. W. Sun, C. Chen, B. Zheng, C. Chen, L. Zhu and W. Lui. Merged aggregate nearest neighbor query processing in road networks. ISBN: 978-1-4503-8, pages 2243-2248, 2013.

[3] D. Papadias, Q.Shen, Y.Tao, and K.Mouratidis. Group nearest neighbor queries. In ICDE, pages 301-312, 2004.