Graph-Decomposed *k*-NN Searching Algorithm on Road Network

Wei JIANG, Bo NING, Guanyu LI, Mei BAI, Xiao JIA, and Fangliang WEI

Frontiers of Computer Science, DOI: 10.1007/s11704-023-3626-3

k-NN Searching Algorithm

Definition 1 (The k-NN Searching problem on Road Network). *Given a road network $G(V, E)$, a query point $q \in V$, an integer $k \in \mathbb{N}^+$ and a set of moving objects M ($|M| > k$), the k nearest neighbor (kNN) searching problem on road network aims to find a result set of objects R, satisfies that (1) $|R| = k$; (2) $R \subseteq M$; (3) $\forall v \in R$, $v' \in M-R$, such that $dist(q,v) \leq dist(q,v')$.*

The tree nodes are connected into a tree based on the weighted cost model (Formula (1)).

$$Cost(v,v') = W_{min} dist(v,v') + \frac{W_{ed}(v')}{\sum_{x \in (X(v)-v)} W_{ed}(x)} \cdot \frac{W_{ed}(v')}{|N_{ed}(v')|} \cdot |N_{ea}(v')| \quad (1)$$

Algorithm 1: Graph-Decomposed k-NN Searching Algorithm (GDSA-Query)

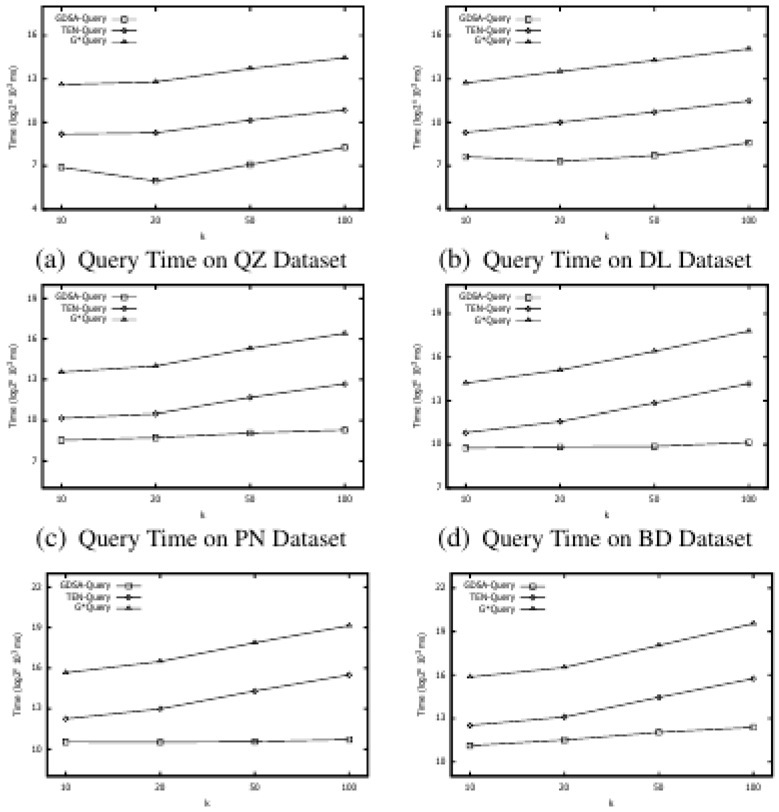
Input: road network $G(V,E)$, query point q, integer k
Output: result set R

```
1  Λ(X(V), E(X(V))) ← construct a tree of G(V,E);
2  CS ← (q, 0);
3  for v ∈ V−q do
4      if q and v hasnot a minimum common ancestor v′ then
5          CS ← (v, dist(q,v));
6      else
7          if c=(v′, dist(q,v′)) ∈ CS then
8              CS ← (v, c+dist(v′,v)))
9          CS ← (v, dist(q,v′)), CS ← (v, dist(q,v′)+dist(v′,v))
10 R = Top-k sort in Hash(V, M) of CS;
11 Return R
```

Algorithm Description

The pruning strategy can improve the time-efficiency of k-NN searching algorithm, displayed in Algorithm 1, that can be divided into three subroutines. The first subroutine is used to construct a graph-decomposed tree of road network (Section 3.1 and Line 1). The second subroutine is the key step of kNN searching to find all the shortest path from nodes to query node q (Lines 2-9). The query node weighted by 0 is initially added to the candidate set CS (Line 2), then the verification of minimum common ancestor between query node q and data node v is determined for different processing strategies to calculate the shortest path. If q and v has not a minimum common ancestor v', $dist(q,v)$ is calculated into candidate set CS. Otherwise, $dist(q,v)$ is calculated as the sum of $dist(q,v')$ and $dist(v',v)$ by **Lemma 1**. If candidate set contains the local result $dist(q,v')$, the local result is extracted and combined with $dist(v',v)$ for the shortest path $dist(q,v)$(Line 8). To improve the time-efficiency of shortest path calculation, a structure of hashmap is employed to store all the adjacent weighted edge of tree nodes. The third subroutine is used to conduct the k moving objects bound to the nearest nodes of q, where the moving objects have been transposed by an offset and desployed on a hash structure (Line 10).

Experimental Result and Conclusions



(a) Query Time on QZ Dataset (b) Query Time on DL Dataset
(c) Query Time on PN Dataset (d) Query Time on BD Dataset
(e) Query Time on TR Dataset (f) Query Time on BK Dataset

Fig.3: Query Performance Varying k-values

Table 1: Statistics of road network data

DataSets	|*V*|	|*E*|	*h*	*w*
Quanzhou(QZ)	5672	7521	210	79
Dalian(DL)	13605	17984	209	65
Pune(PN)	28649	36925	359	144
Baghdad(BD)	60108	88876	560	201
Tehran(TR)	110580	147339	721	356
Bangkok(BK)	154352	187798	806	357

Experimental Evaluation: The query performance varying different *k* values is evaluated in Fig.3. A more stationary linearity of our algorithm is measured than G*-Query and TEN-Query algorithms as the incremental changes of *k* values. The graph-decomposed tree abstracts all nodes of graph as the tree nodes, and the size of tree nodes depends on the quantity of neighbors. Therefore, this tree is not related to the selection of *k* value, which transcend the *k* value limitation by storing all potential nodes.

Experiments conduct that our GDSA-Query algoirthm has a stable performance evaluated in the wild range of tree-depth and maximum quantity of tree nodes on multiple real datasets, counted in Table 1.

Conclusions: This paper designs a novel algorithm of kNN searching on road network. A graph-decomposed tree is constructed from road network based on one abstracted rule and a cost model, then a graph-decomposed kNN searching algorithm is proposed to improve the query time-efficiency. Extensive empirical studies on real and synthetic graphs demonstrate that our techniques outperform the state-of-the-art algorithms on six datasets.