Efficient Shortest Path Queries on Triangular Irregular Networks and Point Clouds

Yinzhao Yan
The Hong Kong University of Science and Technology
yanyinzhao0329@gmail.com

Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
raywong@cse.ust.hk

2026 27th IEEE International Conference on Mobile Data Management (MDM) | 979-8-3195-3310-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/MDM71479.2026.00065

***Abstract*—Performing shortest path queries on a 3D surface is a topic of widespread interest in both industry and academia. Among different representations of a 3D surface, the most popular representations are *Triangular Irregular Network* (*TIN*), and *point cloud*. However, all existing shortest path query algorithms on a *TIN* are inefficient, and there is no existing shortest path query algorithm on a point cloud. In this thesis, we study how to effectively calculate the shortest path passing on a *TIN* and a point cloud in three aspects. (1) We propose an efficient on-the-fly shortest path algorithm answering the shortest path passing different regions on a weighted *TIN*, where different regions are assigned with different weights. (2) We propose an efficient updatable shortest path oracle answering the shortest path query for a set of *Points-Of-Interests* (*POIs*) on an updated *TIN*. (3) We propose an efficient shortest path oracle answering the shortest path query for a set of *POIs* on a point cloud, and an efficient proximity query algorithm using our oracle. Our experimental results show that they outperform the best-known algorithms or oracles concerning time and memory. The dissertation is available at https://yanyinzhao.github.io/files/Thesis.pdf.**

I. Introduction

This extended abstract summarizes Dr. Yinzhao Yan's PhD dissertation, titled *"Shortest Path Queries on Triangular Irregular Networks and Point Clouds"* completed at The Hong Kong University of Science and Technology (HKUST) in August 2025. The dissertation was supervised by Prof. Raymond Chi-Wing Wong from HKUST. One of Yinzhao's papers was awarded as "MDM 2024 Best Paper Award".

Performing *shortest path queries* on 3D surfaces is important in both industry and academia. In industry, Google Earth and Metaverse utilize shortest paths passing on 3D surfaces (such as Earth and virtual reality) for route planning. In academia, this is a prevalent database research topic [1], [2], [3], [4], [5], [6], [7]. 3D surface representations include *Triangular Irregular Network* (*TIN*), and *point cloud*. A *TIN* contains a set of vertices, *edges* and (triangle) *faces*. A point cloud contains a set of *points*. Figure 1 (a) shows a 3D surface of a mountain. Figures 1 (b) and (c) show the shortest path passing on a *TIN* and a point cloud of this surface, respectively.

A. Motivations

1) ***On-the-fly algorithm for shortest path queries on weighted TINs****:* Given a *TIN*, we can calculate the *unweighted* shortest path passing on an *unweighted TIN* (where all faces of the *TIN* have the same weights). We also aim to calculate the *weighted* shortest path passing on a *weighted TIN* (where each face of the *TIN* has a *weight*) using on-the-fly algorithm. Figures 1 (d) and (e) show weighted and unweighted shortest paths in the purple and blue lines. For example, in wildfire, we set *TIN* faces of the destroyed regions with a larger weight, while the *TIN* faces of the non-destroyed regions with a smaller weight. Then, we can calculate the blue path that does not pass the destroyed region as the evacuation path.

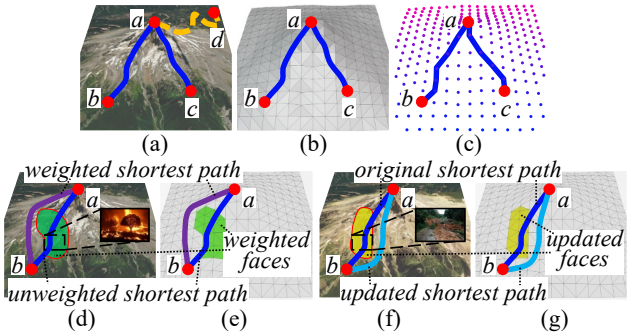

Fig. 1. Shortest paths passing on (a) a 3D surface, (b) a *TIN*, (c) a point cloud, weight and unweighted shortest path passing on (d) a 3D surface and (e) a *TIN*, original and updated shortest path passing on (f) an updated 3D surface and (g) an updated *TIN*.

2) ***Oracle for shortest path queries on updated TINs****:* Given a *TIN* and a set of *Points-Of-Interests* (*POIs*), we can pre-compute the shortest paths among these POIs using an *oracle* on a *TIN*, then we can use the oracle to answer the shortest path query more efficiently. When the *TIN* updates, we also aim to construct the oracle on *TIN*s prone to update, and efficiently update it after the update, then return shortest paths efficiently. Figures 1 (f) and (g) show the original and updated shortest path in blue and light blue lines. For example, in earthquake, we pre-compute shortest paths (among POIs including viewing platforms, shelters, hotels) using an oracle on *TIN*s prone to earthquake, and efficiently update the oracle after the earthquake, then we can use it to efficiently return shortest paths for evacuation.

3) ***Oracle for proximity queries on point clouds****:* Given a point cloud, although we can convert the point cloud to a *TIN* for the shortest path calculation, there are some advantages of point clouds compared with *TIN*s, such that we aim to directly calculate the shortest path passing on a point cloud. These advantages include (1) more direct access to point cloud

2375-0324/26/$31.00 ©2026 IEEE
DOI 10.1109/MDM71479.2026.00065

data, (2) lower hard disk usage of a point cloud, and (3) faster shortest path query time on a point cloud. We also aim to precompute the shortest paths among POIs using an oracle on a point cloud, to answer the proximity query (inlcuding shortest path, *kNN* and *range* queries) more efficiently. For example, in wildfire, we build an oracle on the point cloud and use it to efficiently answer proximity queries for evacuation.

B. Contributations

There are three studies in this extended abstract.

1) ***On-the-fly algorithm for shortest path queries on weighted TINs****:* In MDM 2024 [3], we propose an efficient two-step $(1+\epsilon)$-approximate on-the-fly shortest path algorithm that answers the shortest path query on a weighted *TIN* called *Rough-Refine*, i.e., *Roug-Ref*, where $\epsilon > 0$ is called the *error parameter*. Our experimental results show that its shortest path query time is 73s for a *TIN* with 50k faces, which is up to 1,630 times faster than the best-known algorithm. This paper was awarded as "MDM 2024 Best Paper Award".

2) ***Oracle for shortest path queries on updated TINs****:* In TKDE 2024 [7], we propose an efficiently updatable $(1+\epsilon)$-approximate shortest path oracle that answers the shortest path query on an updated *TIN* called *Updatable Path Oracle*, i.e., *UP-Oracle*. Our experimental results show that its oracle update time and shortest path query time are 400s $\approx$ 6.7 min and 0.1s, respectively, for a *TIN* with 0.5M faces and 250 POIs, which are up to 88 times and 3 times faster than the best-known oracle, respectively.

3) ***Oracle for proximity queries on point clouds****:* In SIGMOD 2024 [4], we propose an efficient $(1+\epsilon)$-approximate shortest path oracle that answers the shortest path query on a point cloud called *Rapid Construction path Oracle*, i.e., *RC-Oracle*. Our experimental results show that its oracle construction time and proximity (e.g., *kNN*) query time are 200s $\approx$ 3.2 min and 0.12.5s, respectively, for a point cloud with 2.5M points and 500 POIs, which are up to 390 times and 12 times faster than the best-known oracle, respectively.

II. Algorithms

A. Algorithm Roug-Ref

1) ***Preliminary****:* Consider a weighted *TIN* T with N vertices, and a set of edges and faces. Each vertex has three coordinate values, and each face has a weight. Given a source s and a destination t on T, let $\Pi^*(s,t)$ be the exact weighted shortest path passing on T, and $\Pi(s,t)$ be the weighted shortest path passing on T calculated by algorithm *Roug-Ref* (the sum of each face's weight multiplied by the length of the path segment passing on it). Let $|\cdot|$ be a path's weighted distance, e.g., $|\Pi^*(s,t)|$ is $\Pi^*(s,t)$'s weighted distance.

2) ***Major idea****:* Algorithm *Roug-Ref* achieves outstanding performance concerning the shortest path query time due to the rough-refine concept.

(i) *Algorithm Roug*: Given a source s, a destination t and a weighted *TIN* T, we use a *novel* pruning step by placing a small nunber of Steiner points on edges of T and calculate a rough path between s and t. We can transfer the pruned-out information to a later step, and after conducting necessary checks in the later step, there is no need to perform calculations on the pruned-out information anymore.

(i) *Algorithm Ref*: Given the rough path, we use *Snell's law* to efficiently refine it to get the refined path. In Physics, when a light ray passes the boundary between two different media (e.g., air and glass), it bends at the boundary since light seeks the path with the minimum time. The incidence angle and refraction angle for the light satisfy *Snell's law*. This avoid placing many Steiner points, which can reduce the shortest path query time of the algorithm.

3) ***Theoretical analysis****:* The shortest path query time for algorithm *Roug-Ref* is $O(N \log N)$. For any pairs of vertices s and t on T, it has $|\Pi(s,t)| \leq (1+\epsilon)|\Pi^*(s,t)|$.

B. UP-Oracle

1) ***Preliminary****:* Consider a *TIN* T_{bef} with N vertices, and a set of edges and faces. Each vertex has three coordinate values, if the vertices have position update, we obtain a new *TIN*, T_{aft}. Let P be a set of n POIs on T_{bef}. Given a source s and a destination t in P, and a *TIN* T (which can be T_{bef} or T_{aft}), let $\Pi^*(s,t|T)$ be the exact shortest path passing on T, and $\Pi(s,t|T)$ be the shortest path passing on T calculated by *UP-Oracle*.

2) ***Major idea****: UP-Oracle* achieves outstanding performance concerning the oracle update time due to the useful information on T_{bef} and a novel property. It has three phases, i.e., construction phase, update phase and shortest path query phase.

(i) *Construction phase*: Given an original *TIN* T_{bef} and a set of POIs P, we construct *UP-Oracle* by calculating the exact shortest paths among P. This is the useful information on T_{bef} to reduce the likelihood of updating paths when T_{bef} udpates, and to reduce the oracle update time.

(ii) *Update phase*: Given an updated *TIN* T_{aft}, we compare T_{bef} and T_{aft} to detect updated faces and update *UP-Oracle* by only calculating the updated exact shortest paths using utilize the novel *non-updated TIN shortest path intact* property. It implies that given a path between two POIs passing on T_{bef}, if the distances from both POIs to the updated faces are large enough, then the path between them passing on T_{aft} remains unchanged and does not need updating. This reduces the oracle update time. Then, we generate a sub-graph using the exact shortest paths to reduce the output size.

(iii) *Shortest path query phase*: Given a source s and a destination t in P, we return the shortest path between them by performing Dijkstra's algorithm on the sub-graph.

3) ***Theoretical analysis****:* The oracle update time and shortest path query time for *UP-Oracle* are $O(N^2 + n\log^2 n)$ and $O(\log n)$. For any pairs of POIs s and t in P, it has $|\Pi(s,t|T_{aft})| \leq (1+\epsilon)|\Pi^*(s,t|T_{aft})|$.

C. RC-Oracle

1) ***Preliminary****:* Given a set of N points with three coordinate values, we let C be a point cloud of these points. Let P be a set of n POIs, each of which is a point on the point

cloud. Given a source s and a destination t in P, let $\Pi^*(s,t|C)$ be the exact shortest path passing on C, and $\Pi(s,t|C)$ be the shortest path passing on C calculated by *RC-Oracle*.

*2) **Major idea**: RC-Oracle* achieves outstanding performance concerning the oracle construction time due to the *tight* earlier termination criterion. It has two phases, i.e., construction phase and shortest path query phase.

(i) *Construction phase*: Given a *point cloud* C and a set of POIs P, we construct *RC-Oracle* by calculating the shortest paths among P. We regard each POI as a source and use a variation of Dijkstra's algorithm for n times for oracle construction, and we assign a *tight* earlier termination criteria for each POI to terminate the *SSAD* algorithm earlier to reduce the oracle construction time.

(ii) *Shortest path query phase*: Given a source s and a destination t in P, we return the shortest path between them by concating the precomputed paths stored in *RC-Oracle*.

*3) **Theoretical analysis**:* The oracle construction time and shortest path query time for *RC-Oracle* are $O(N \log N + n \log n)$ and $O(1)$. For any pairs of POIs s and t in P, it has $|\Pi(s,t|C)| \leq (1+\epsilon)|\Pi^*(s,t|C)|$.

III. Empirical Studies

A. Algorithm Roug-Ref

We conducted our experiment on 27 real *TIN* datasets [3] with multi-resolutions (e.g., *EP-small* and *EP*), and compared algorithm *Roug-Ref* with four baselines. Figure 2 shows the effect of ϵ on *EP-small* dataset (with 3k faces). The shortest path query time of algorithm *Roug-Ref* is smaller than other baselines, and its distance error ratio is close to 0.

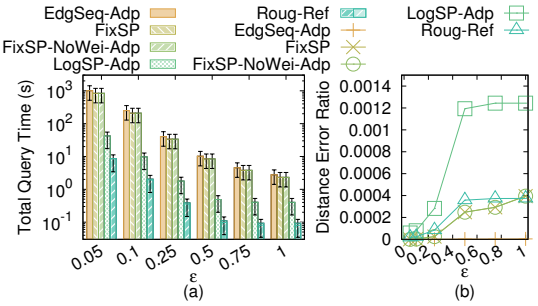

Fig. 2. Effect of ϵ on *EP-small* dataset for algorithm *Roug-Ref*

B. UP-Oracle

We conducted our experiment on 30 real before and after earthquake *TIN* datasets [7] (e.g., *GI*), and compared *UP-Oracle* with eight baselines. Figure 3 shows the effect of ϵ on *GI* dataset (with 0.5M faces). The oracle update time and shortest path query time of *UP-Oracle* are smaller than other baselines, and its distance error ratio is close to 0.

C. RC-Oracle

We conducted our experiment on 34 real point cloud datasets with multi-resolutions [4] (e.g., *EP-small* and *EP*), and compared *RC-Oracle* with seven baselines. Figure 4 shows

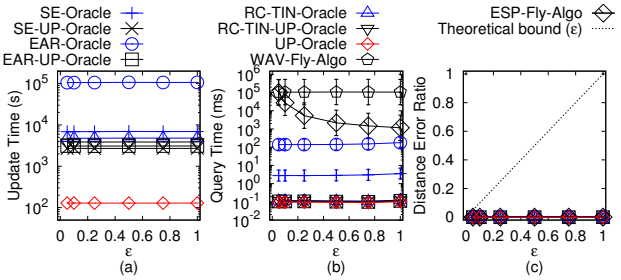

Fig. 3. Effect of ϵ on *GI* dataset for *UP-Oracle*

the effect of ϵ on *EP-small* dataset (with 10k points). The oracle construction time and shortest path query time of *RC-Oracle* are smaller than other baselines, and its distance error ratio is close to 0.

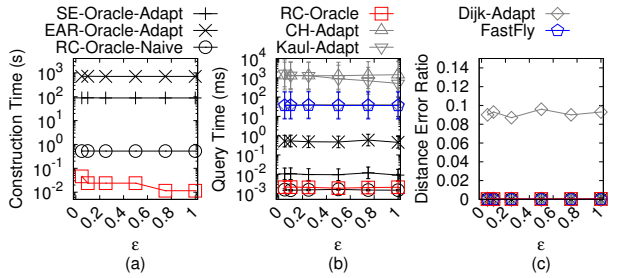

Fig. 4. Effect of ϵ on *EP-small* point cloud dataset for *RC-Oracle*

IV. Conclusion

We proposed several shortest path query algorithms on *TIN*s and point clouds in the dissertation. Apart from these, we deployed Path Advisor (VLDB 2021 [2]) for HKUST, proposed advanced point cloud algorithms (TODS 2025 [5]), and studied shortest path queries on height maps (SIGMOD 2026 [6]). For future work, we will integrate Large Language Models in these algorithms for more complicated tasks.

References

[1] Y. Yan, "Shortest path queries on triangular irregular networks and point clouds," Ph.D. dissertation, The Hong Kong University of Science and Technology, 2025.

[2] Y. Yan and R. C.-W. Wong, "Path advisor: a multi-functional campus map tool for shortest path," in *International Conference on Very Large Data Bases (VLDB)*, vol. 14, no. 12, 2021, pp. 2683–2686.

[3] Y. Yan and R. C.-W. Wong, "Efficient shortest path queries on 3D weighted terrain surfaces for moving objects," in *IEEE International Conference on Mobile Data Management (MDM)*, 2024.

[4] Y. Yan and R. C.-W. Wong, "Proximity queries on point clouds using rapid construction path oracle," in *ACM Conference on Management of Data (SIGMOD)*, vol. 2, no. 1, 2024, pp. 1–26.

[5] Y. Yan and R. C.-W. Wong, "Efficient path oracles for proximity queries on point clouds," *ACM Transactions on Databases Systems (TODS)*, vol. 51, no. 1, pp. 1–42, 2026.

[6] Y. Yan and R. C.-W. Wong, "Efficient proximity queries on simplified height maps," in *ACM Conference on Management of Data (SIGMOD)*, vol. 3, no. 4, 2026, pp. 1–26.

[7] Y. Yan, R. C.-W. Wong, and C. S. Jensen, "An efficiently updatable path oracle for terrain surfaces," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, no. 1, pp. 1–14, 2024.