EAR-Oracle: On Efficient Indexing for Distance Queries between Arbitrary Points on Terrain Surface

Bo Huang*, Victor Junqiu Wei†, Raymond Chi-Wing Wong^ and Bo Tang*

* Southern University of Science and Technology
† The Hong Kong Polytechnic University
^ The Hong Kong University of Science and Technology

Outline

- Introduction
- Proposed Solution
- Experimental Result
- Conclusion

Basics of Terrain Surface

- Terrain Surface in Real World:
  - Various *topographic features*:
    - Sand, rock, slope, etc.

Real terrain surfaces are complex.

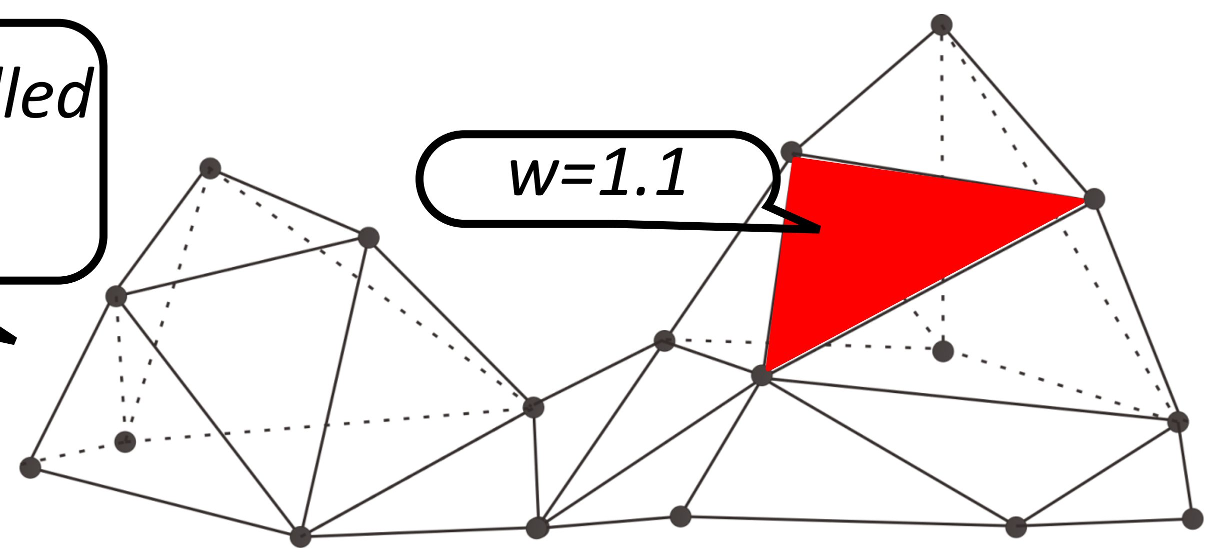
Mountains, hills and valleys in rural areas

Picture from: wallpapertag.com/mountains-wallpaper

Basics of Terrain Surface

- Terrain Surface in Digital World:
  - 3D geometric object:
    - Consists of *vertices* (V), *edges* (E) and *faces* (F):
      - 18 vertices, 39 edges and 23 faces in the example.
  - Each face is a triangle:
    - Assigned a *floating point value* to represent *topographic features*:
      - The face weight of the red face is 1.1 in the example.

This kind of terrain surface is called weighted terrain surfaces.

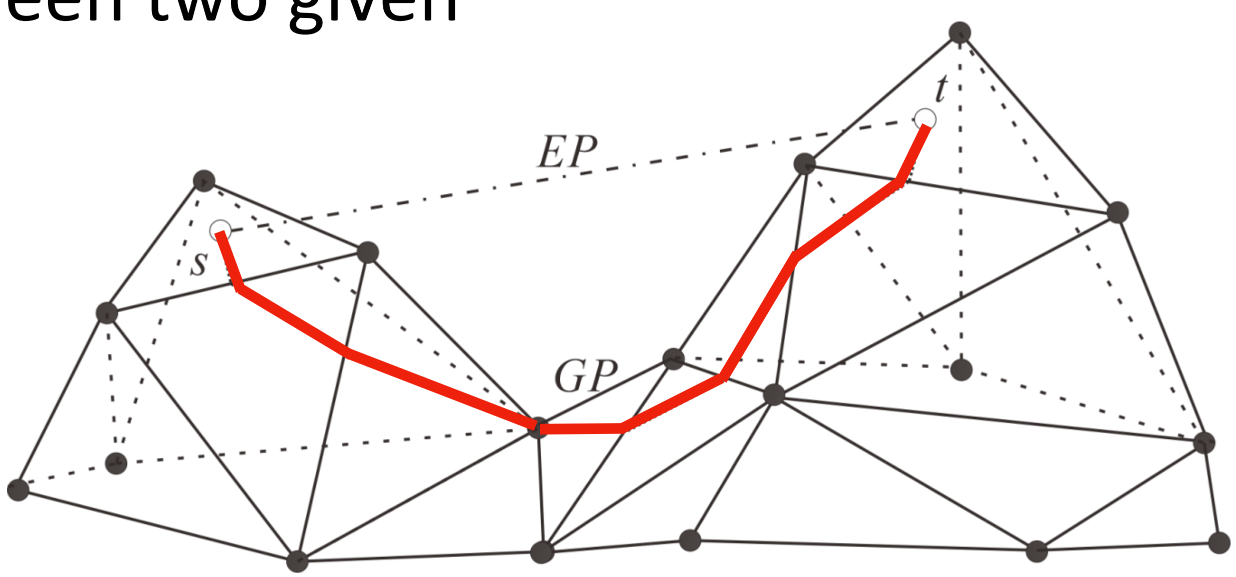


Digital Terrain Surface Example

Basics of Terrain Surface

- Geodesic Path/Distance:
  - The *geodesic path* between two given points is the shortest path *on the terrain surface*.
    - GP (*red* path) is the geodesic path between s and t.
  - The *geodesic distance* (denoted by $d_g(\cdot,\cdot)$) between two given points is the *length* of their geodesic path.

- *Arbitrary point-to-arbitrary point* distance queries (*A2A queries*):
  - The geodesic distance queries between two given *arbitrary* surface points.



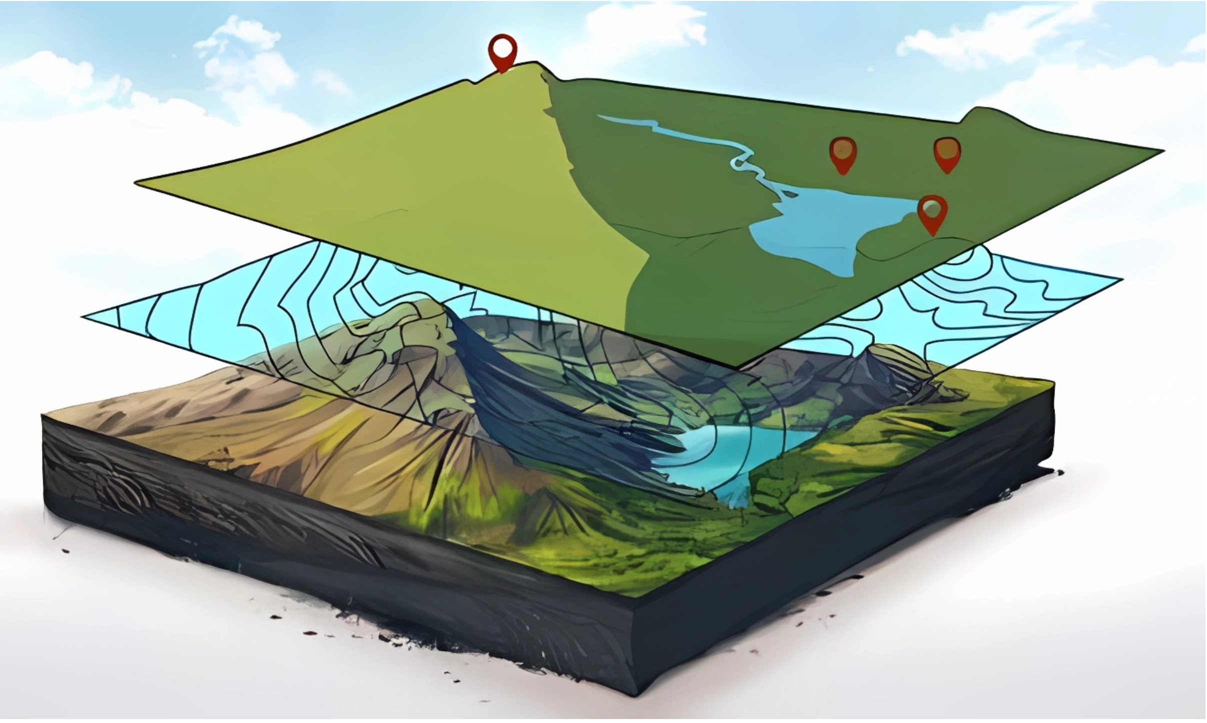
Geodesic Path/Distance Example

Motivation

- Geodesic distances are *essential* to many *high-level applications*:
  - *Geographical Information System (GIS):*

    - compute the *travel cost* between two places;

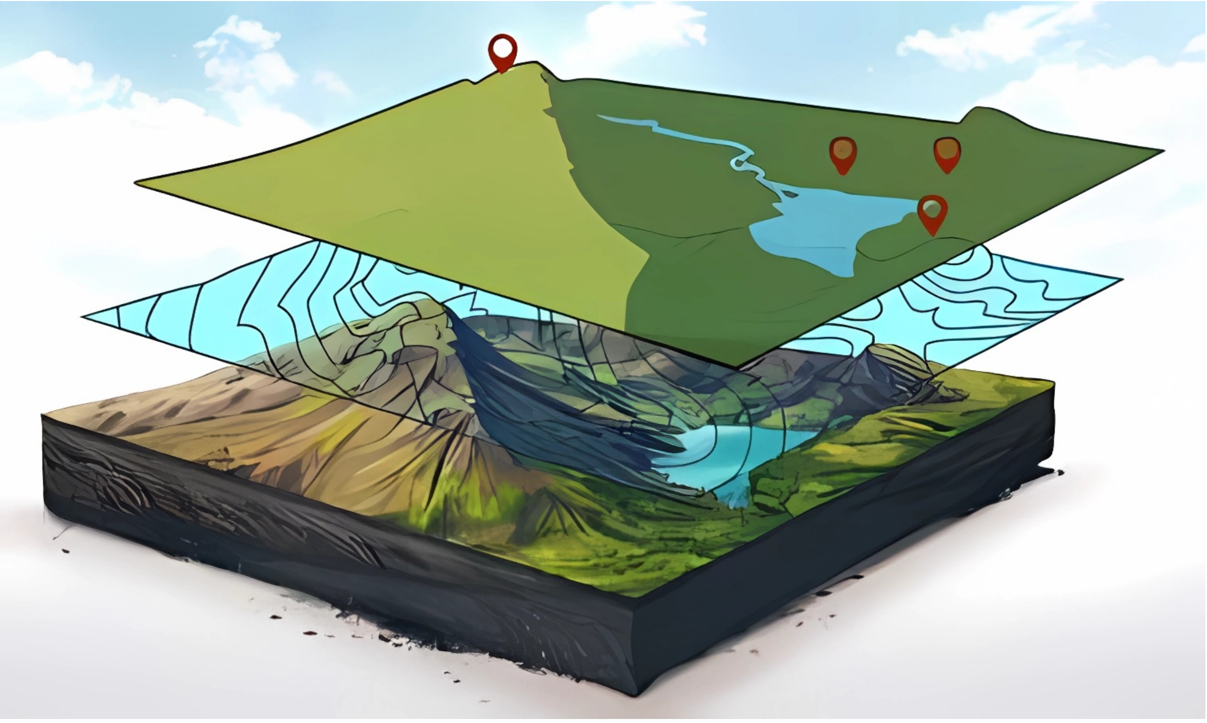
    - study *travel patterns* of animals based on residential sites.



Picture from: https://www.edx.org/course/gis-foundations

Motivation

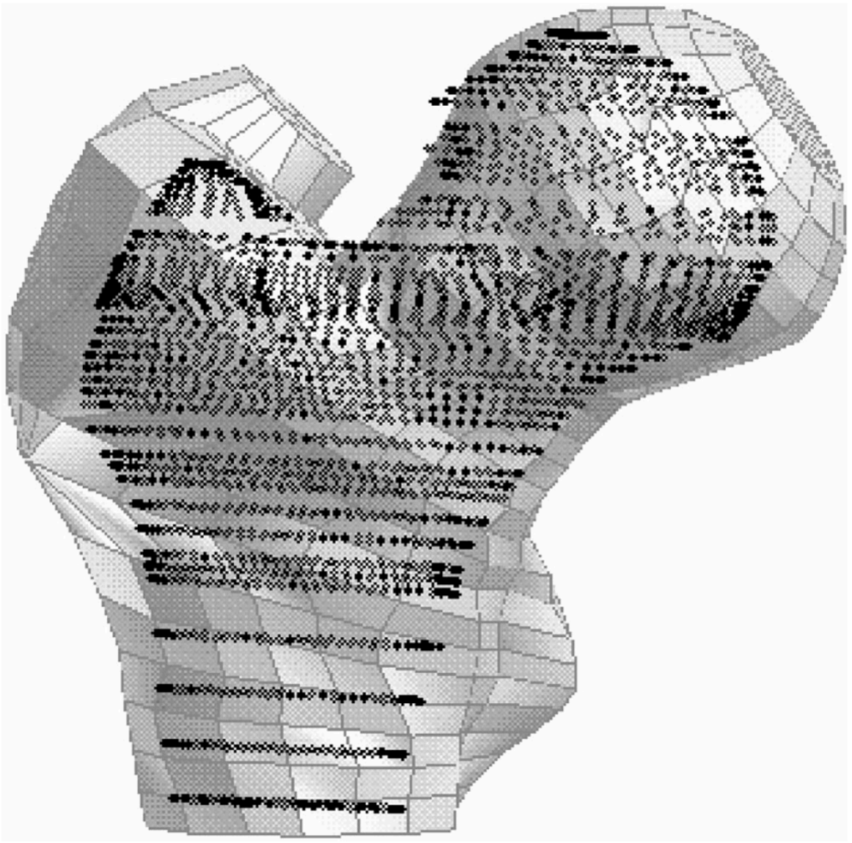
- Geodesic distances are *essential* to many *high-level applications*:
  - Geographical Information System (GIS):
    - compute the travel cost between two places;
    - study travel patterns of animals based on residential sites.
  - *Spatial data mining*:
    - check *spatial co-location patterns*;
    - *Clustering* objects on terrain sufaces.



Picture from: https://www.edx.org/course/gis-foundations

Motivation

- Geodesic distances are *essential* to many *high-lev*
  - Geographical Information System (GIS):
    - compute the travel cost between two places
    - study travel patterns of animals based on res
  - Spatial data mining:
    - check spatial co-location patterns;
    - Clustering objects on terrain sufaces.
  - *Scientific 3D modeling:*
    - analyse *key features* based on distances between reference points.
  - etc.



Picture from: *B. Couteau et al. Journal of Biomechanics' 2000*

Motivation

- Geodesic distances are *essential* to many *high-level applications*:
  - *Geographical Information System (GIS)*:
    - compute the travel cost between two places;
    - study travel patterns of animals based on residential sites.
  - *Spatial data mining*:
    - check spatial co-location patterns;
    - Clustering objects on terrain sufaces.
  - *Scientific 3D modeling*:
    - analyse key features based on distances between reference points.
  - etc.
- Many of them have *no restriction* on query points:
  - Any surface points can be regarded as query points.

There is a need to process A2A queries efficiently.

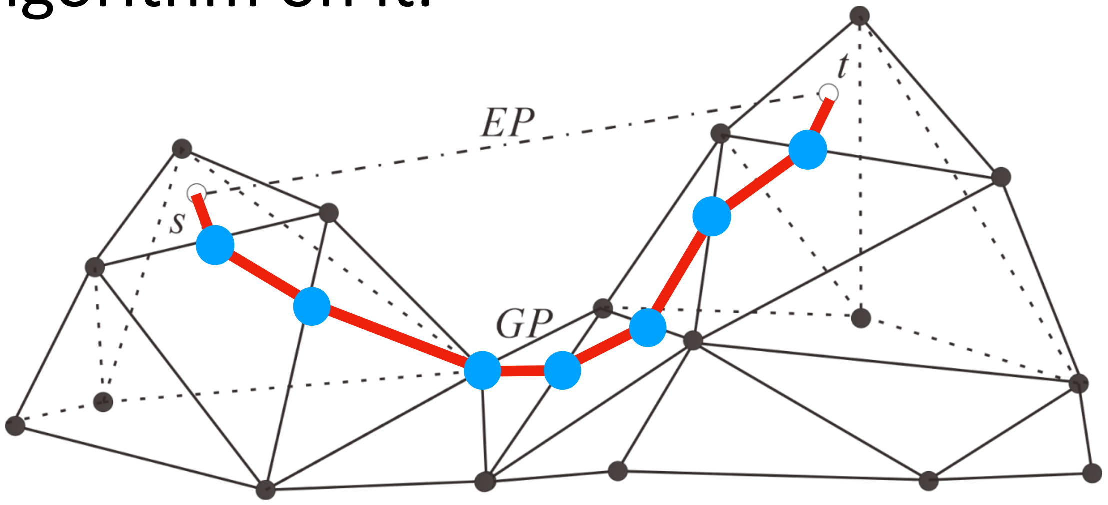
Existing Studies

- There is *no* efficient algorithm for calculating the *exact geodesic distance* on *weighted* terrain surfaces:
  - 3D *quadratic programming* model [*SIGSPATIAL*' 2021].

69.71 seconds for distance query passing only 5 faces.

- Follow the existing studies, we focus on finding *approximate geodesic distance* (denoted by $\tilde{d}_g(\cdot, \cdot)$) with *theoretical guarantees*:
  - Introduce *Steiner points* (blue auxiliary points) [*Algorithmica*' 2001]:
    - Obtain a *graph* and run shortest path algorithm on it.

Edge weights are calculated based on face weights.



Geodesic Path/Distance Example

Existing Studies

- Approximate Geodesic Distance Algorithms:
  - On-the-fly Algorithms:
    - *Fixed Scheme* (*FS*). [*Algorithmica'* 2001]
    - *Unfixed Scheme* (*US*). [*J. ACM'* 2005]
    - *K-Algorithm* (*K-Algo*). [*VLDB'* 2015]
  - Index-based Algorithms:
    - *Steiner-Point Oracle* (*SP-Oracle*). [*ESA'* 2011]
    - *Space-Efficient Oracle* (*SE-Oracle*). [*SIGMOD'* 2017]

Deficiency of Existing Studies

- Approximate Geodesic Distance Algorithms:
  - On-the-fly Algorithms:
    - *Fixed Scheme* (*FS*). [*Algorithmica'* 2001]
    - *Unfixed Scheme* (*US*). [*J. ACM'* 2005]
    - *K-Algorithm* (*K-Algo*). [*VLDB'* 2015]

Queries are processed online without any pre-computation.

  - Index-based Algorithms:
    - *Steiner-Point Oracle* (*SP-Oracle*). [*ESA'* 2011]
    - *Space-Efficient Oracle* (*SE-Oracle*). [*SIGMOD'* 2017]

Deficiency of Existing Studies

- Approximate Geodesic Distance Algorithms:
  - On-the-fly Algorithms:
    - *Fixed Scheme* (*FS*). [*Algorithmica'* 2001]
    - *Unfixed Scheme* (*US*). [*J. ACM'* 2005]
    - *K-Algorithm* (*K-Algo*). [*VLDB'* 2015]
  - Index-based Algorithms:
    - *Steiner-Point Oracle* (*SP-Oracle*). [*ESA'* 2011]
    - *Space-Efficient Oracle* (*SE-Oracle*). [*SIGMOD'*

Single query needs about 2.13 seconds for a 1-million-face dataset.

On a dataset with only 3,696 vertices (with skinny faces), about 37.48 seconds and 4.32 seconds are required for US and K-Algo, respectively.

Deficiency of Existing Studies

- Approximate Geodesic Distance Algorithms:
  - On-the-fly Algorithms:
    - *Fixed Scheme* (*FS*). [*Algorithmica'* 2001]
    - *Unfixed Scheme* (*US*). [*J. ACM'* 2005]
    - *K-Algorithm* (*K-Algo*). [*VLDB'* 2015]
  - Index-based Algorithms:
    - *Steiner-Point Oracle* (*SP-Oracle*). [*ESA'* 2011]
    - *Space-Efficient Oracle* (*SE-Oracle*). [*SIGMOD'* 2017]

Index too many points for A2A queries.

Deficiency of Existing Studies

- Approximate Geodesic Distance Algorithms:
  - On-the-fly Algorithms:
    - *Fixed Scheme* (*FS*). [*Algorithmica'* 2001]
    - *Unfixed Scheme* (*US*). [*J. ACM'* 2005]
    - *K-Algorithm* (*K-Algo*). [*VLDB'* 2015]
  - Index-based Algorithms:
    - *Steiner-Point Oracle* (*SP-Oracle*). [*ESA'* 2011]
    - *Space-Efficient Oracle* (*SE-Oracle*). [*SIGMOD'* 2017]

Only efficient for pre-defined query points.

More than 3 hours pre-processing time and 256 GB memory for a terrain with 10,243 vertices (for A2A queries).

Our Contribution

- Propose an index-based algorithm for *A2A distance queries*:
  - Called *Efficient Arbitrary Point-to-Arbitrary Point Oracle* (*EAR-Oracle*).
  - Outperforms the state-of-the-art *index-based* algorithm by *2 orders* of magnitude in terms of *building time* and *space consumption*;
  - Outperforms the fastest *on-the-fly* algorithm by *1 order* of magnitude in terms of *query time.*

Our Contribution

- Propose an index-based algorithm for *A2A distance queries*:
  - Called *Efficient Arbitrary Point-to-Arbitrary Point Oracle* (*EAR-Oracle*).
  - Outperforms the state-of-the-art *index-based* algorithm by *2 orders* of magnitude in terms of *building time* and *space consumption*;
  - Outperforms the fastest *on-the-fly* algorithm by *1 order* of magnitude in terms of *query time.*

- Thorough *theoretical analysis*:
  - *Building time*, *space consumption*, *query time* and *distance error.*

Our Contribution

- Propose an index-based algorithm for *A2A distance queries*:
  - Called *Efficient Arbitrary Point-to-Arbitrary Point Oracle* (*EAR-Oracle*).
  - Outperforms the state-of-the-art *index-based* algorithm by *2 orders* of magnitude in terms of *building time* and *space consumption*;
  - Outperforms the fastest *on-the-fly* algorithm by *1 order* of magnitude in terms of *query time.*

- Thorough *theoretical analysis*:
  - *Building time*, *space consumption*, *query time* and *distance error.*

- Extensive *experimental studies*:
  - On several *real* datasets with *different scales*;
  - On *factors influencing* the performance of *EAR-Oracle*.

Related Studies Comparison

Algorithm	Type	Weighted Terrain	Index Time	Query Latency	Scalability	Result Quality
FS	On-the-fly	✓	-	✗	✓	✓
US	On-the-fly	✓	-	✗	✗	✓
K-Algo	On-the-fly	✗	-	✗	✗	✓
SP-Oracle	Index	✓	✗	✓	✗	✓
SE-Oracle	Index	✗	✗	✓	✗	✓
EAR-Oracle	Index	✓	✓	✓	✓	✓

Our proposed algorithm overcomes the drawbacks of existing studies and has the best overall performance.

Outline

- Introduction
- Proposed Solution
  - *Pre-processing phase*
  - *Query phase*
- Experimental Result
- Conclusion

EAR-Oracle Pre-processing Phase

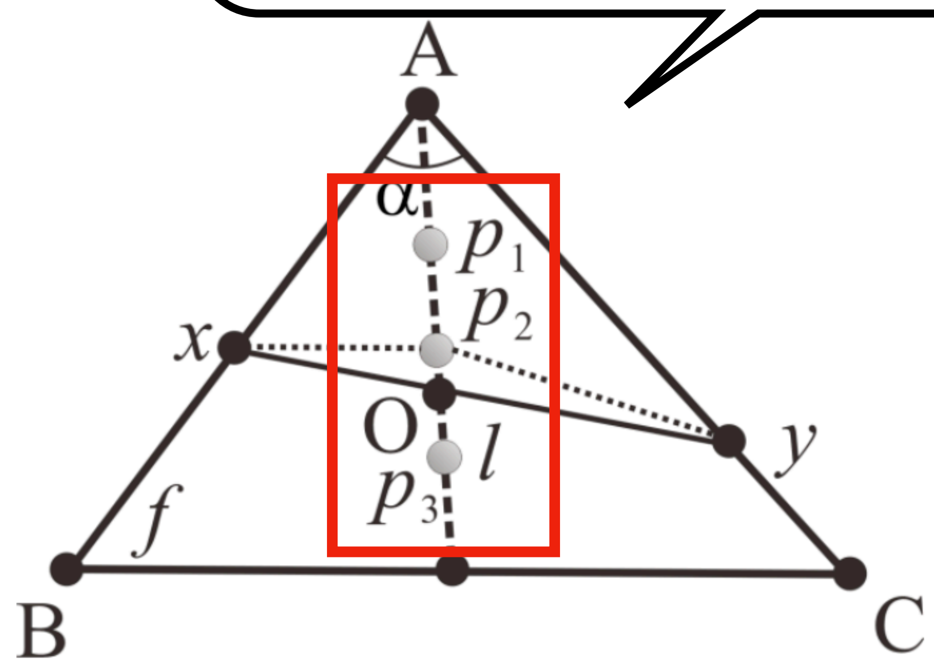
- Build a *base graph* (denoted by G_B) for *distance metric approximation*:

There is no efficient algorithm for exact solution on weighted terrain surfaces.

EAR-Oracle Pre-processing Phase

- Build a *base graph* (denoted by G_B) for *distance metric approximation*:
  - Place m *Steiner points uniformly* on each *angle-bisector of each face*:
    - Used to approximate the path inside *a single face.*

p_1,p_2,p_3 are 3 Steiner points on the bisector of $\angle BAC$.

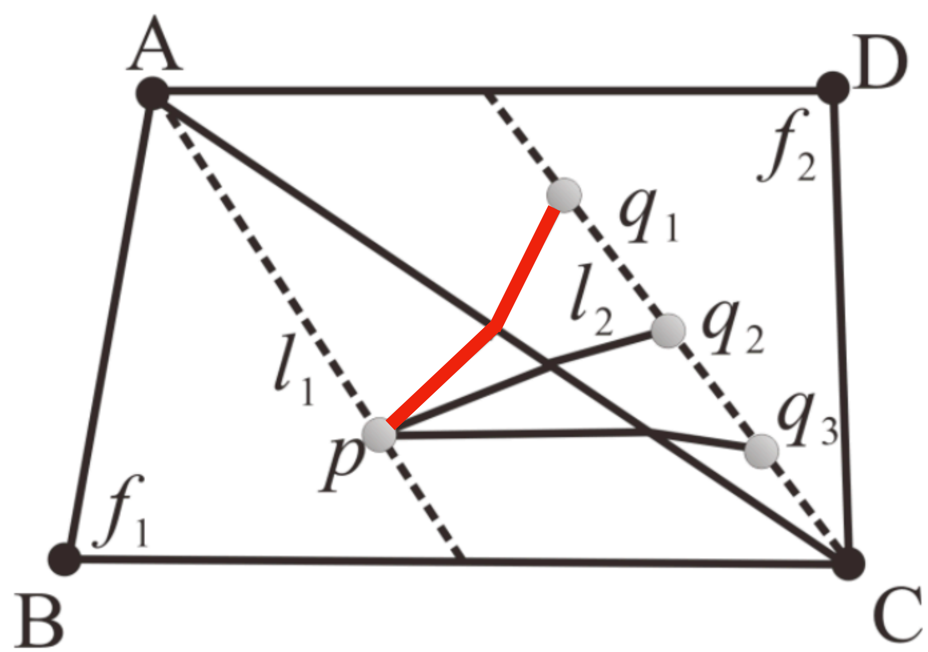


Example of base graph for m=3.

EAR-Oracle Pre-processing Phase

- Build a *base graph* (denoted by G_B) for *distance metric approximation*:
  - Connect *edges* between *Steiner points* on *adjacent faces*; The *weighted geodesic paths* are calculated based on the *Snell's Law*;

Also known as the law of reflection. It could be used to calculate the exact geodesic path for adjacent faces [J.ACM'2005].

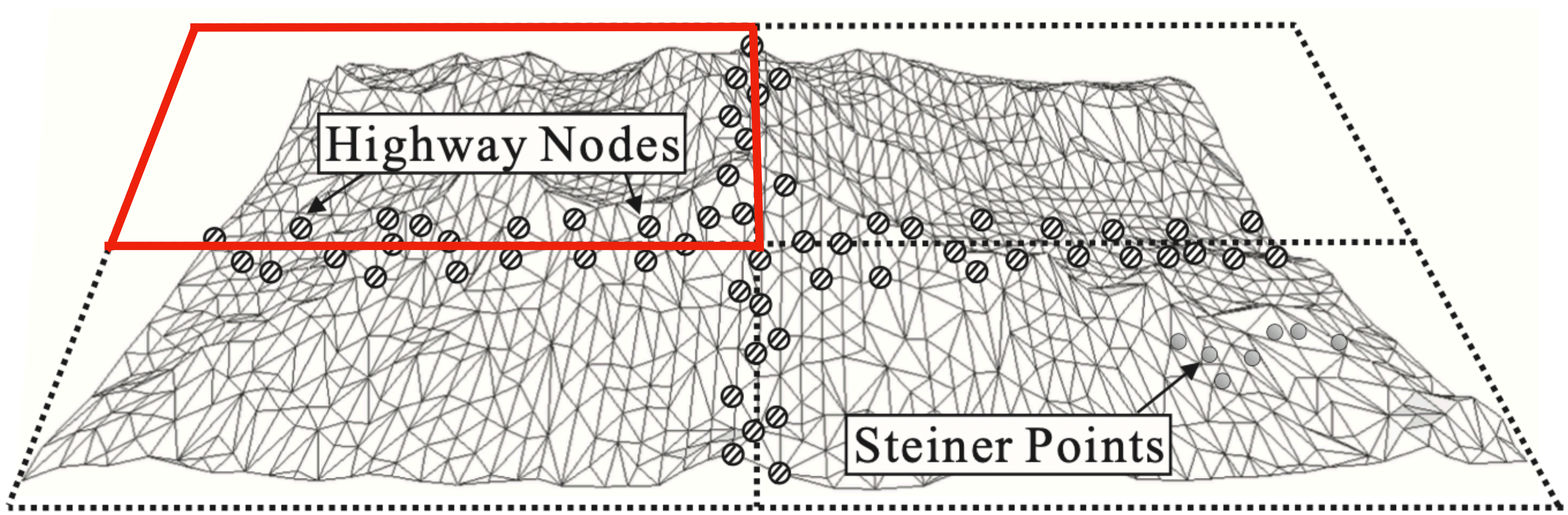


Example of base graph for m=3.

EAR-Oracle Pre-processing Phase

- *Partition* the terrain surface into several *boxes in 2D* (x-y plane):
  - The terrain surface is a *planar graph*;

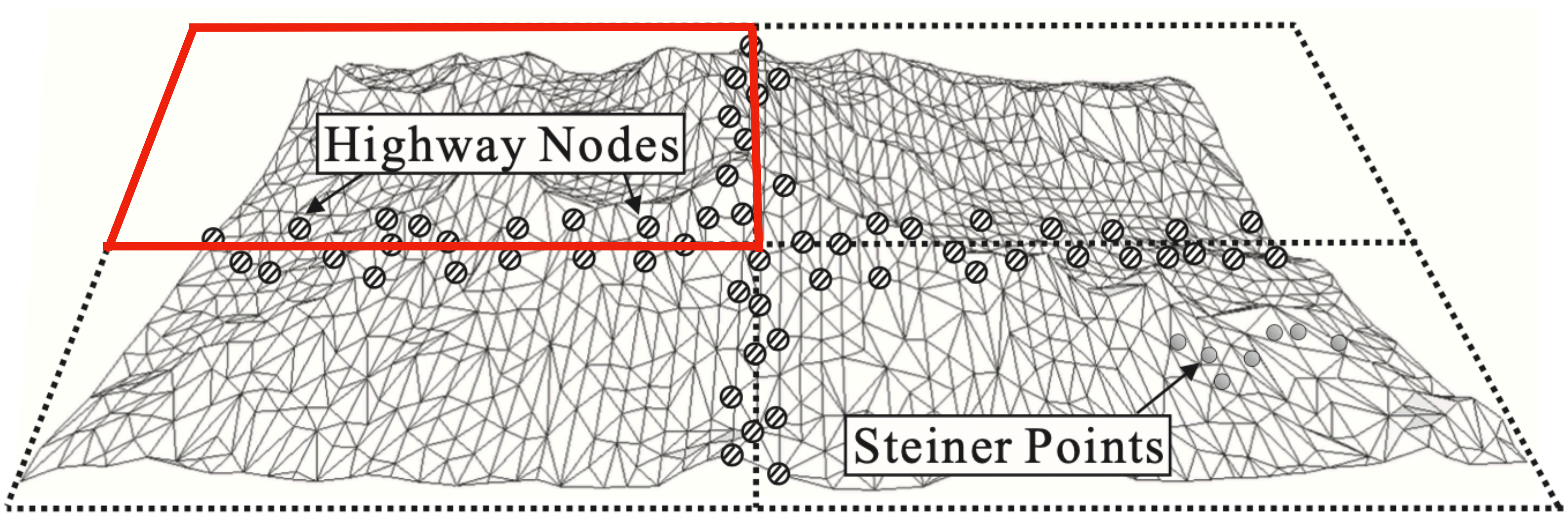
We could naturally partition the terrain on x-y plane.



EAR-Oracle Pre-processing Phase

- *Partition* the terrain surface into several *boxes in 2D* (x-y plane):
  - When the query source and the query destination are *close* (in the same box), they have *spatial locality*;

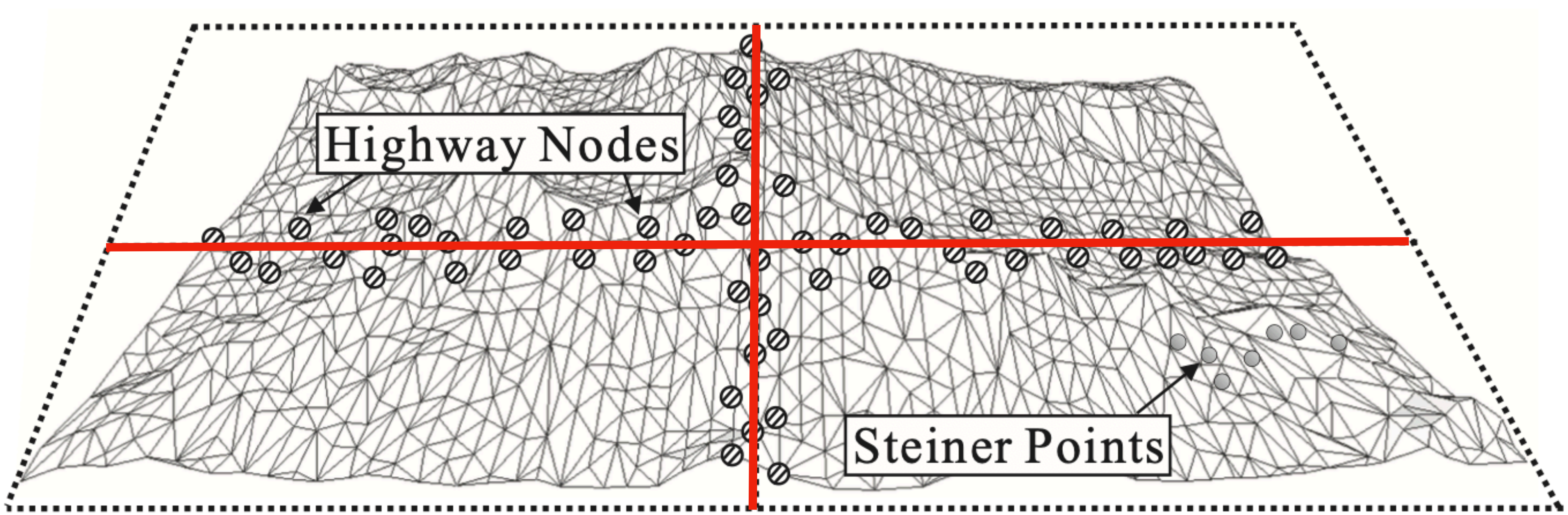
On-the-fly algorithms have good performance.



EAR-Oracle Pre-processing Phase

- *Partition* the terrain surface into several *boxes in 2D* (x-y plane):
  - When the query source and the query destination are *distant* (in different boxes), their *geodesic path* will *go through* certain *boundaries of some boxes*.

We only need to focus on a few points near boundaries.



EAR-Oracle Pre-processing Phase

- *Select* several terrain *vertices close to* the box *boundaries*:
  - *Previous* studies *index* a lot of *Steiner points* for theoretical guarantee;

On a small terrain with 1,440 vertices,
43,407 Steiner points are introduced.

EAR-Oracle Pre-processing Phase

- *Select* several terrain *vertices close to* the box *boundaries*:
  - *Previous* studies *index* a lot of *Steiner points* for theoretical guarantee;

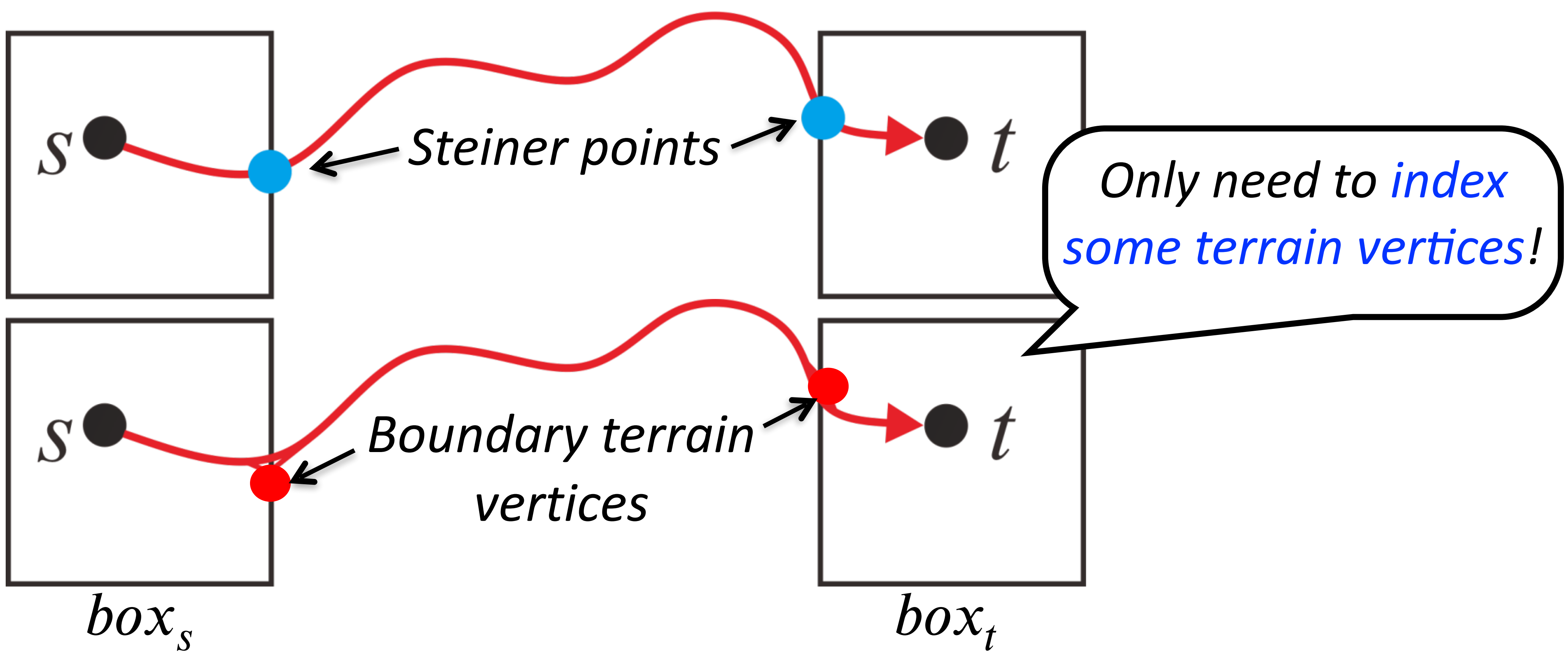
    On a small terrain with 1,440 vertices, 43,407 Steiner points are introduced.

  - If we index the Steiner points near the box boundaries, we still need a lot of *pre-processing time* and *space consumption*.

EAR-Oracle Pre-processing Phase

- *Select* several terrain *vertices close to* the box *boundaries*:
  - We *slightly move* the *Steiner points* to *terrain vertices* (on the same face) near the boundaries:
    - The two paths are very similar.

We derived distance error of the two paths.

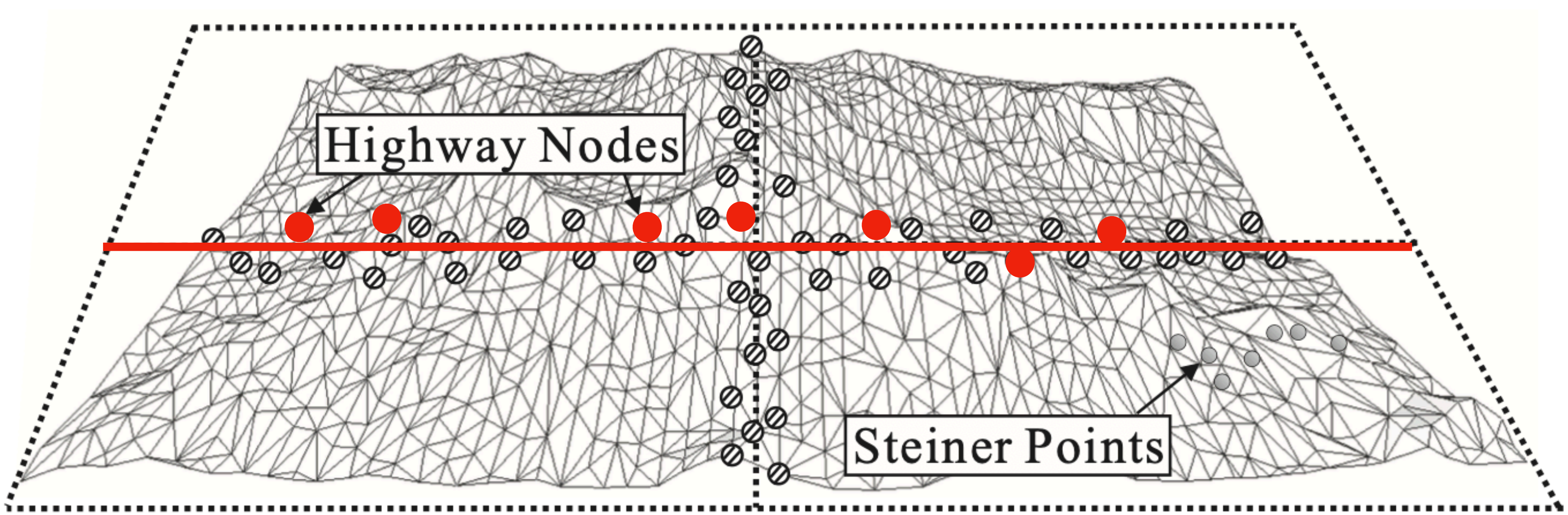


Example of moving Steiner points

EAR-Oracle Pre-processing Phase

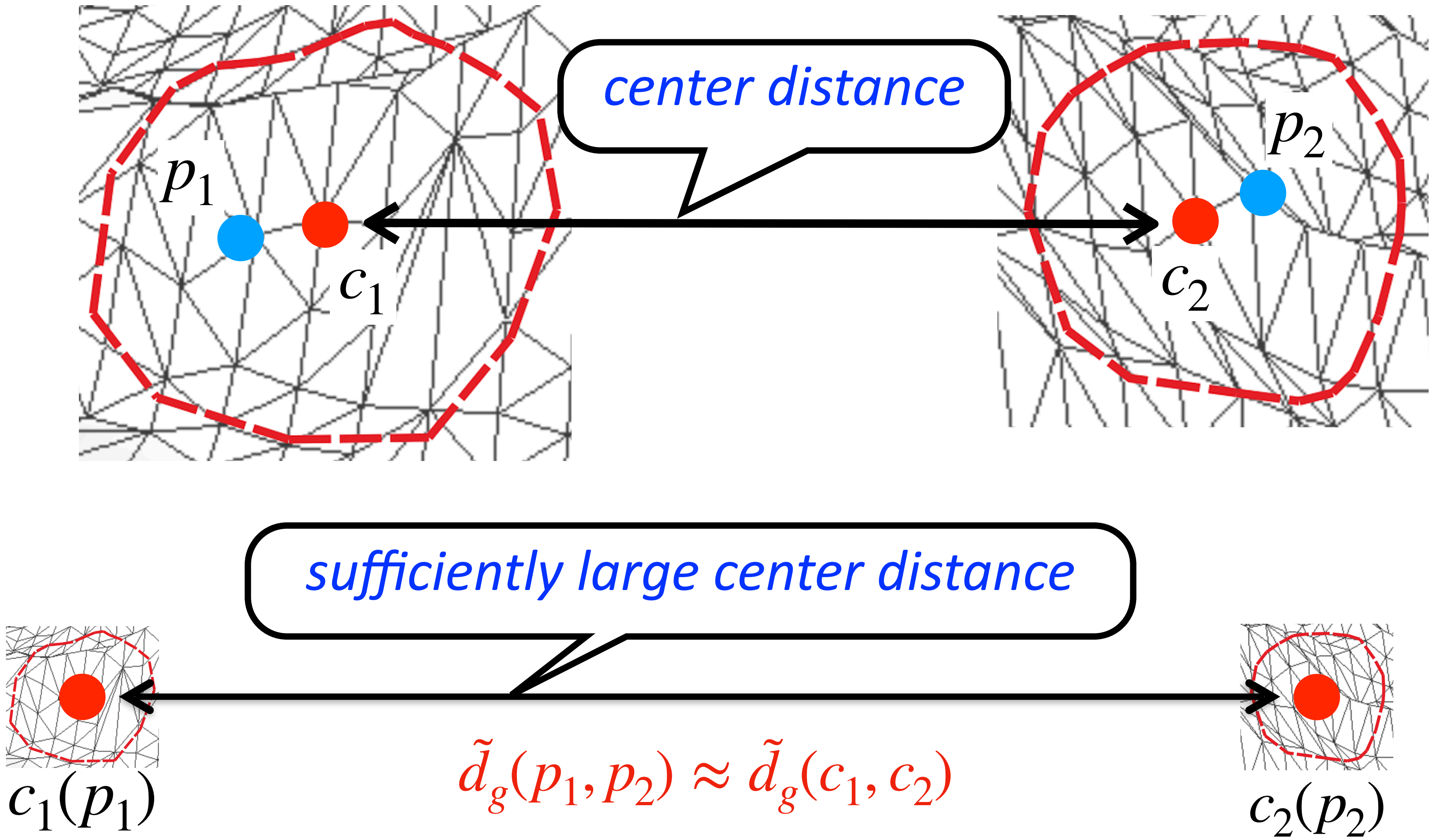
- *Select* several terrain *vertices close to* the box *boundaries*:
  - These terrain vertices near the boundaries are called *highway nodes*;

A subset of terrain vertices (The amount of highway nodes is small).



EAR-Oracle Pre-processing Phase

- Construct a *highway network* to index distances between highway nodes:
  - *Generate edges* between highway nodes according to *geometric property*:
    - Use *center distance* as *approximation*.



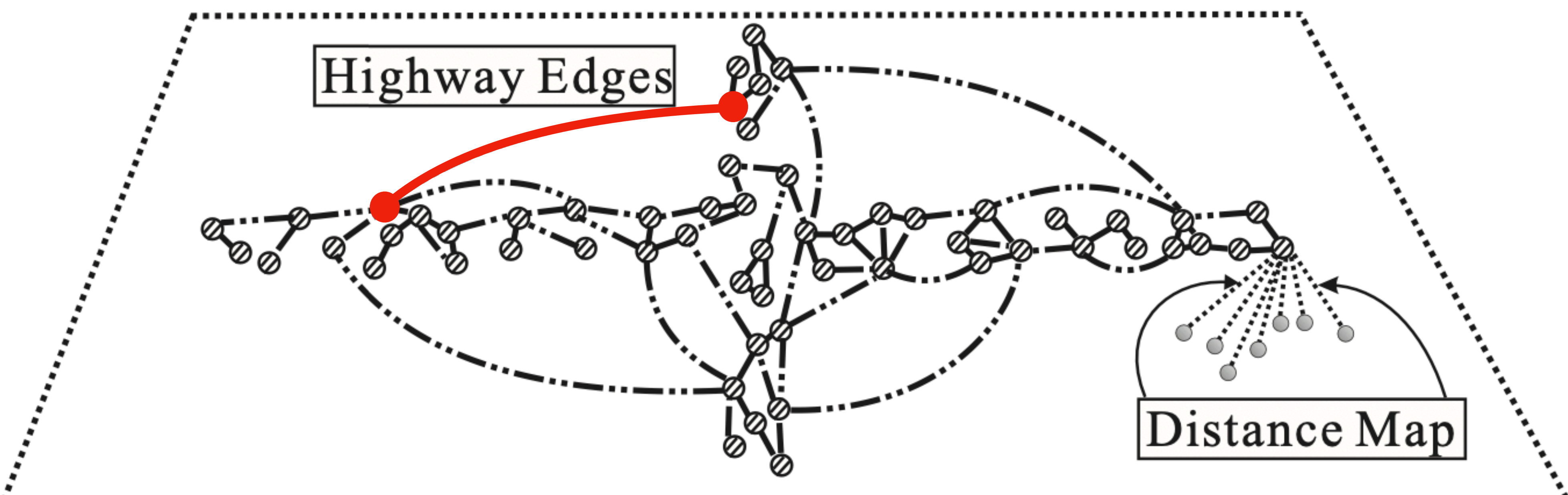
c_1,c_2 are two *highway nodes* and they are *centers* of two surface disks.

p_1 and p_2 are two arbitrary *points* in the two disks, respectively.

EAR-Oracle Pre-processing Phase

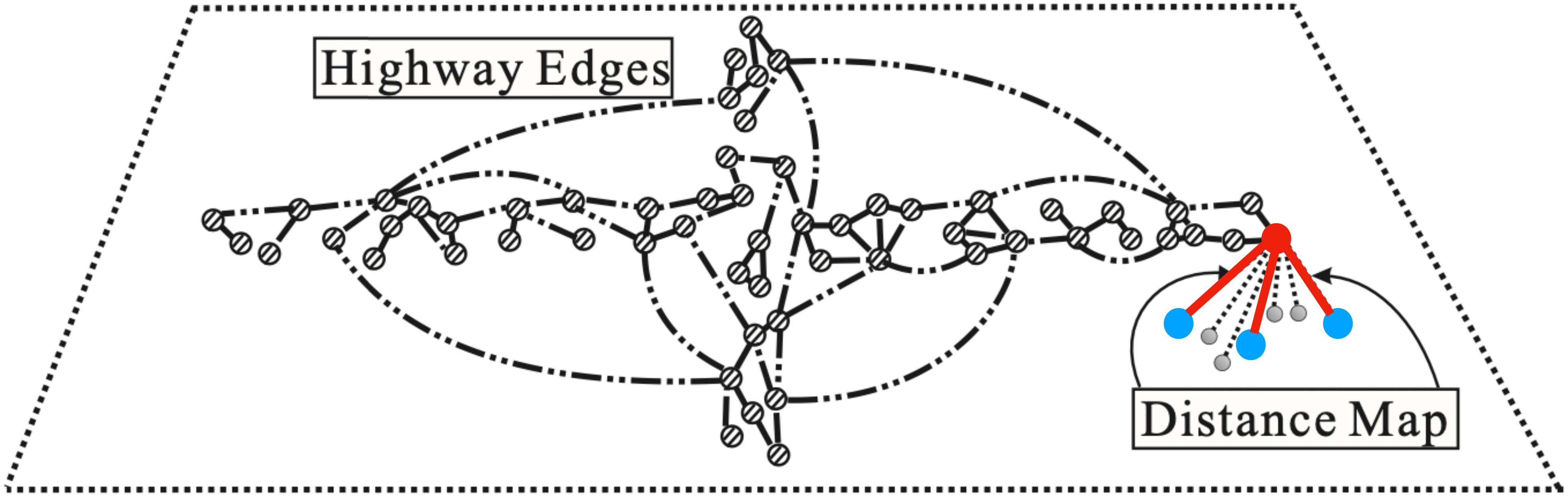
- Construct a *highway network* to index distances between highway nodes:
  - Obtain a *lightweight* highway *network* with *distance guarantee*.

Avoid all-pair distances computation.



EAR-Oracle Pre-processing Phase

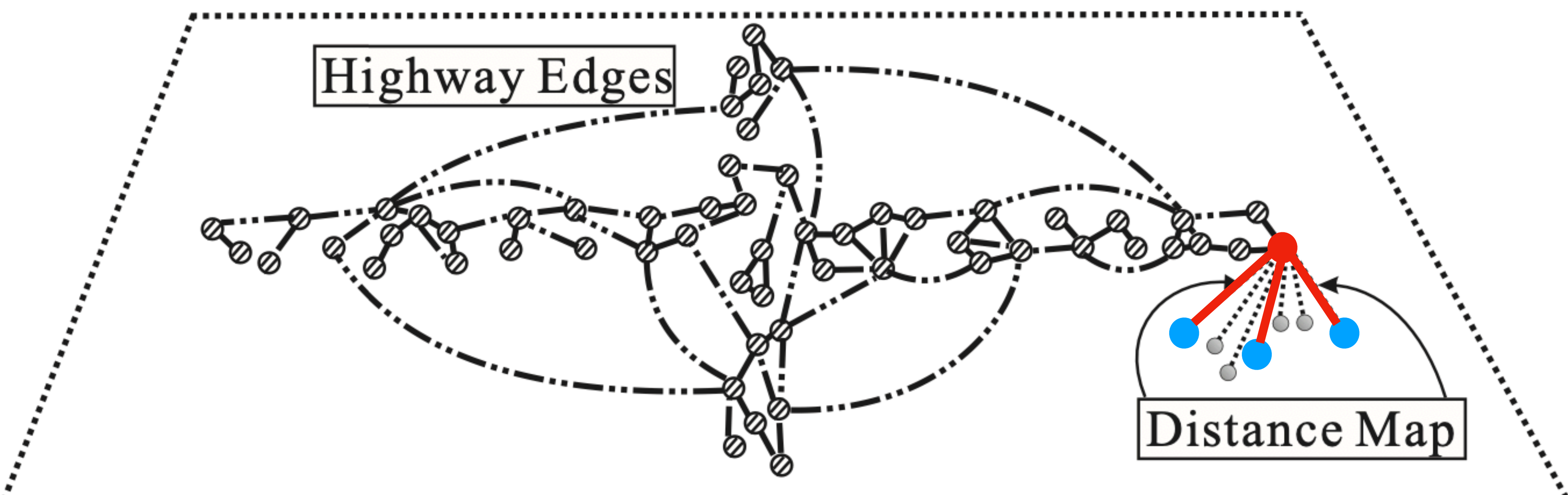
- Build a *distance map* to index distances between *highway nodes* and *Steiner points*:
  - For each box, *index* the distance between each *highway node* on its *boundaries* and *Steiner points* on the faces inside it;



EAR-Oracle Pre-processing Phase

- Build a *distance map* to index distances between *highway nodes* and *Steiner points*:
  - For each box, *index* the distance between each *highway node* on its *boundaries* and *Steiner points* on the faces inside it;

Any surface point can reach the highway network via a single Steiner point.

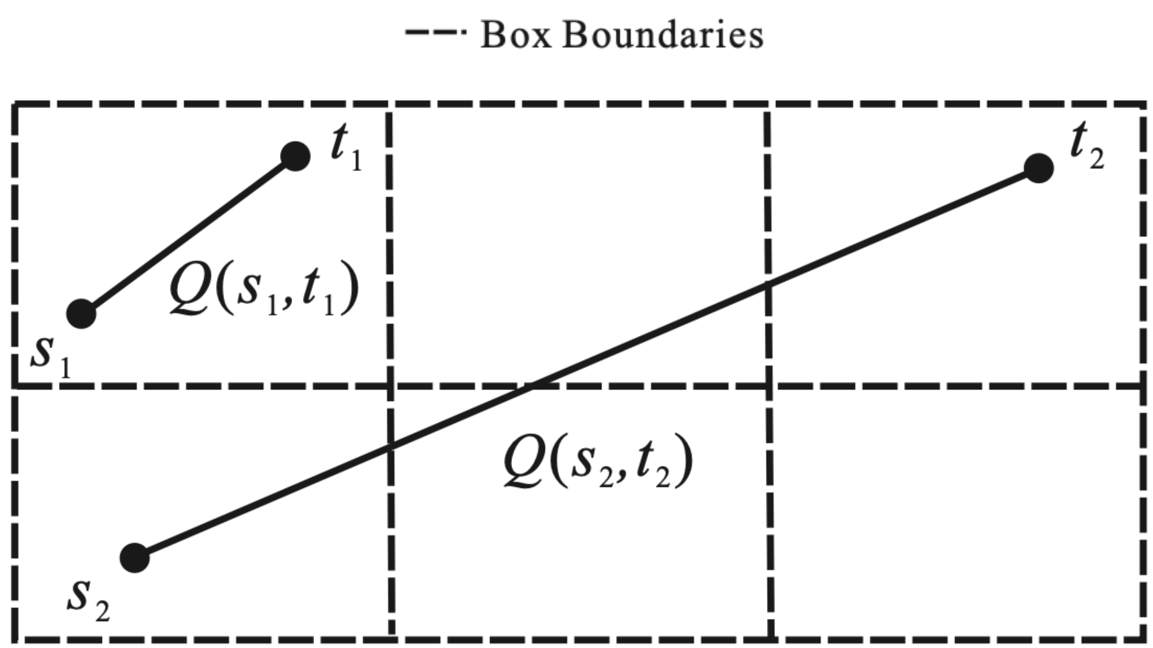


EAR-Oracle Query Phase

- We are given two *arbitrary* surface points s and t. The geodesic distance query $Q(s, t)$ taken s as the *source* and t as the *destination* is called *A2A query*.

- Based on the partition, the queries could be divided into *two types*:
  - The *inner-box* query ($Q(s_1, t_1)$ in the example);
  - The *inter-box* query ($Q(s_2, t_2)$ in the example).

Determine two different query processing routines.

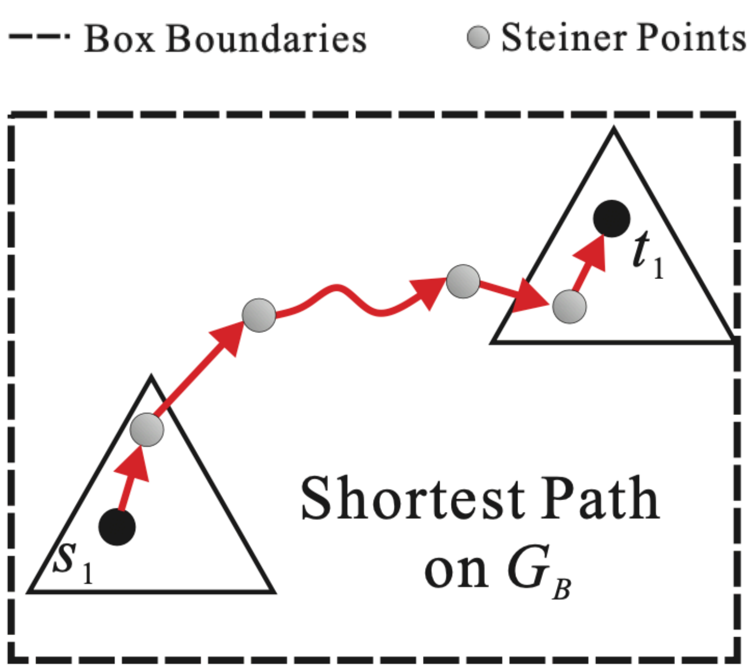


Two types of distance queries

EAR-Oracle Query Phase

- The *inner-box* query ($Q(s_1, t_1)$):
  - Adopt *Dijkstra's algorithm* on base graph G_B.

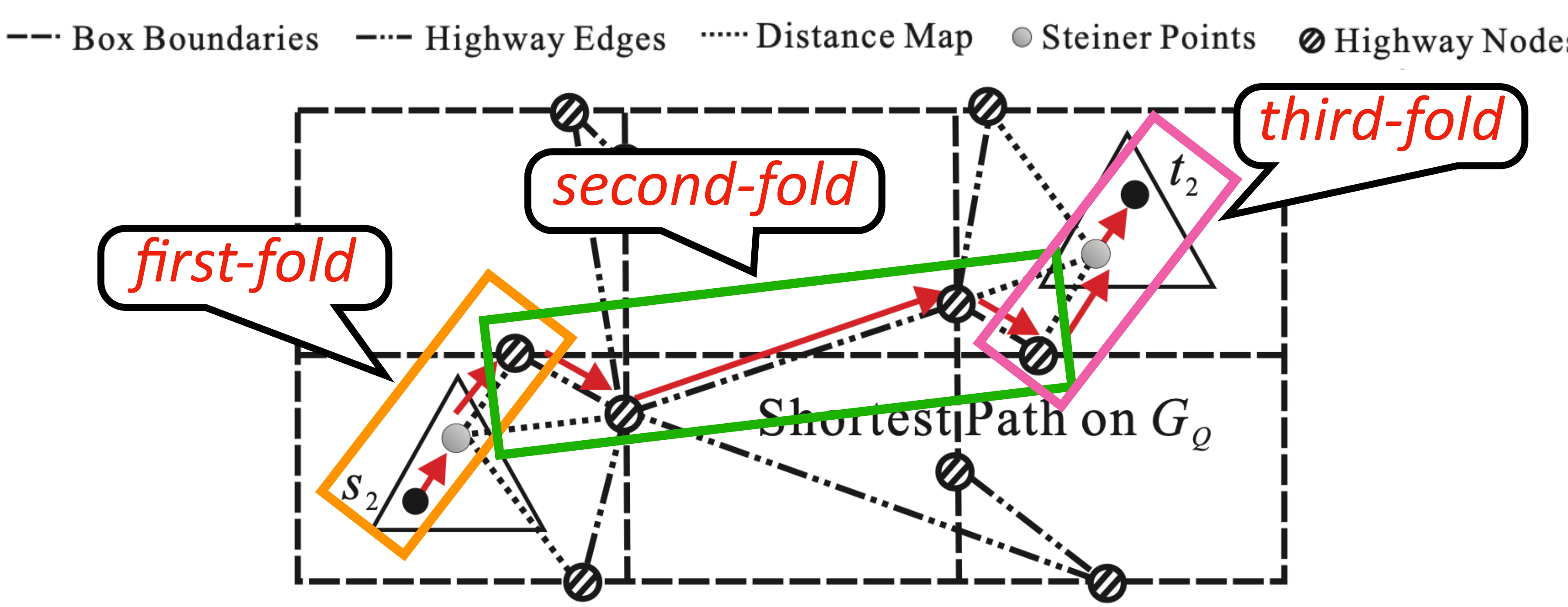
Efficient due to spatial locality.



Inner-box query example

EAR-Oracle Query Phase

- The *inter-box* query ($Q(s_2, t_2)$):
  - it is *three-fold*:
    - From s_2 to *highway node* (distance map);
    - From *highway node* to *highway node* (highway network);
    - From *highway node* to t_2 (distance map).

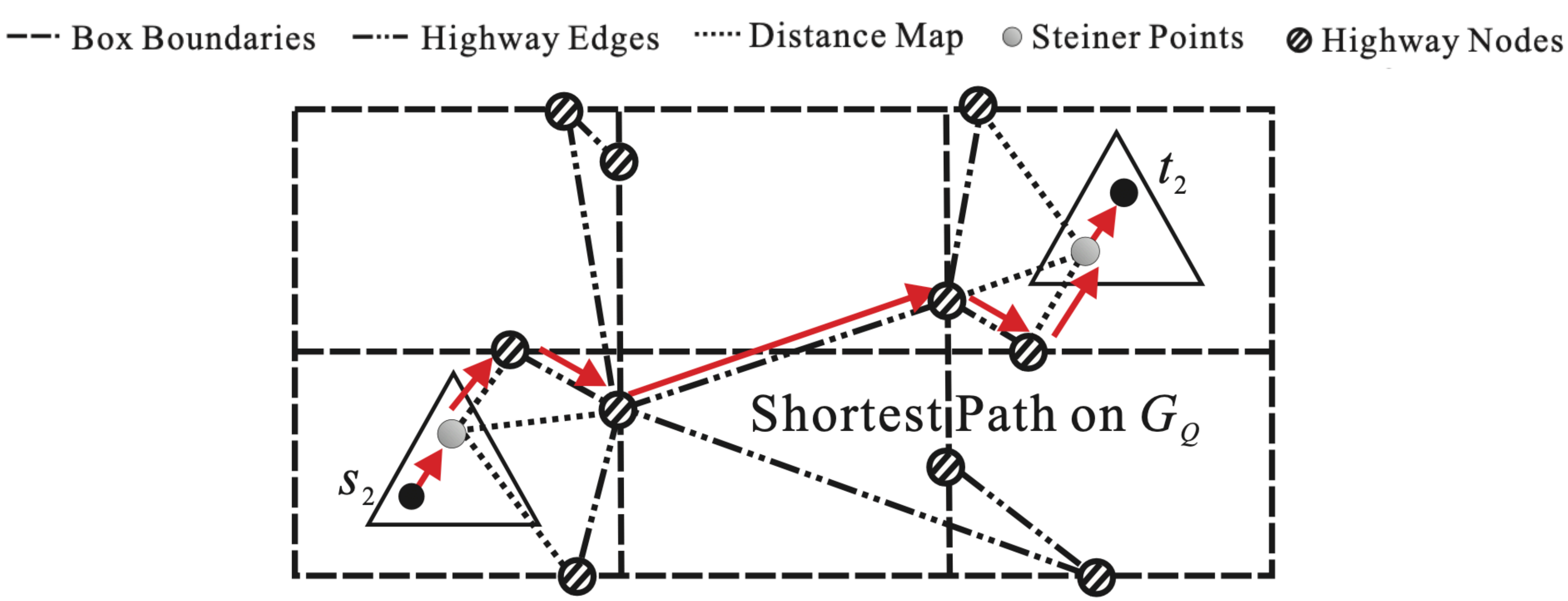


Inter-box query example

EAR-Oracle Query Phase

- The *inter-box* query ($Q(s_2, t_2)$):
  - Construct a *query graph* G_Q by adding edges (from the *distance map*) to the *highway network*;
  - Perform *Dijkstra's algorithm* on query graph G_Q.

Efficient since G_Q is *lightweight*.



Inter-box query example

Theoretical Analysis

- Let N be the amount of terrain faces and ϵ be the user-defined error bound:
  - The *building time* of *EAR-Oracle* is *linearithmic* to N;
  - The *space consumption* of *EAR-Oracle* is *linear* to N;
  - The *query time* of *EAR-Oracle* is *linearithmic* to the amount of highway nodes;

    The amount of highway nodes is much less than N.
  - The *relative distance error* of *EAR-Oracle* is very *close to ϵ*.

$$\frac{|\tilde{d}_g(s,t) - d_g(s,t)|}{d_g(s,t)} \approx \epsilon$$

Outline

- Introduction
- Proposed Solution
- Experimental Result
- Conclusion

Experimental Result

- Tested Algorithms:
  - *On-the-fly* algorithms:
    - *FS* [Algorithmica' 2001]
      - *Fastest* on-the-fly algorithm.
    - *US* [J. ACM' 2005]
      - *Snell's law* applied, ϵ-*bounded* distance error.
    - *K-Algo* [VLDB' 2015]
      - ϵ-*bounded* distance error.
  - *Index-based* algorithms:
    - *SE-Oracle* [SIGMOD' 2017, TODS' 2022]
      - *State-of-the-art* index-based algorithm.
    - *EAR-Oracle* [*Proposed*]

Experimental Result

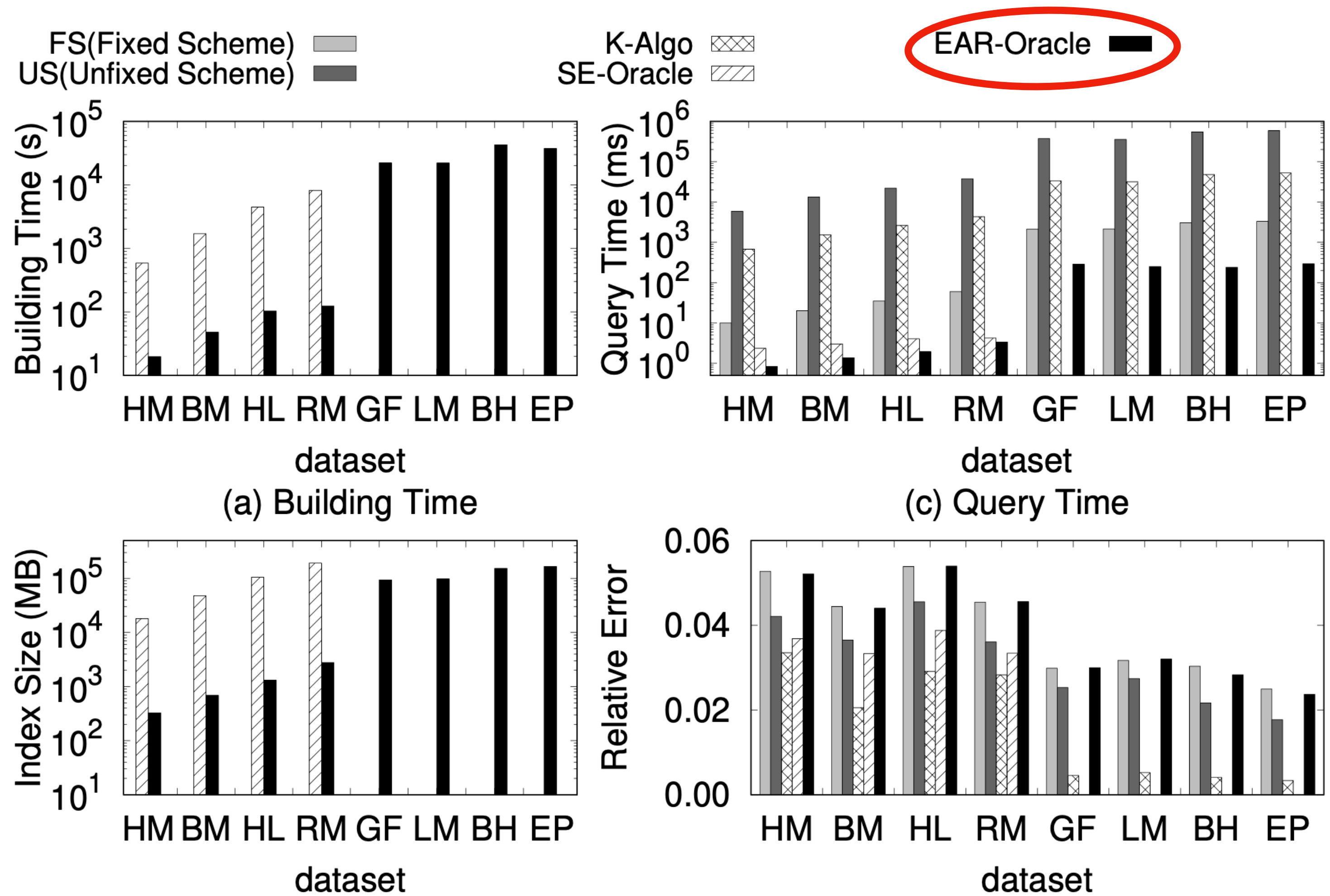
- Datasets:
  - We adopt several *real* terrain surfaces:

Dataset	No. of Faces	Region Covered
HorseMountain (HM)	1,488	15 km^2
BigMountain (BM)	2,772	29 km^2
HeadLightMountain (HL)	4,771	49 km^2
RobinsonMountain (RM)	7,200	71 km^2
GunnisonForest (GF)	199,998	10,038 km^2
LaramieMountain (LM)	199,996	12,400 km^2
BearHead (BH)	292,914	140 km^2
EaglePeak (EP)	325,713	150 km^2

- Measures:
  - *Building Time*, *Space Consumption*, *Query Time* and *Relative Error*.

Experimental Result

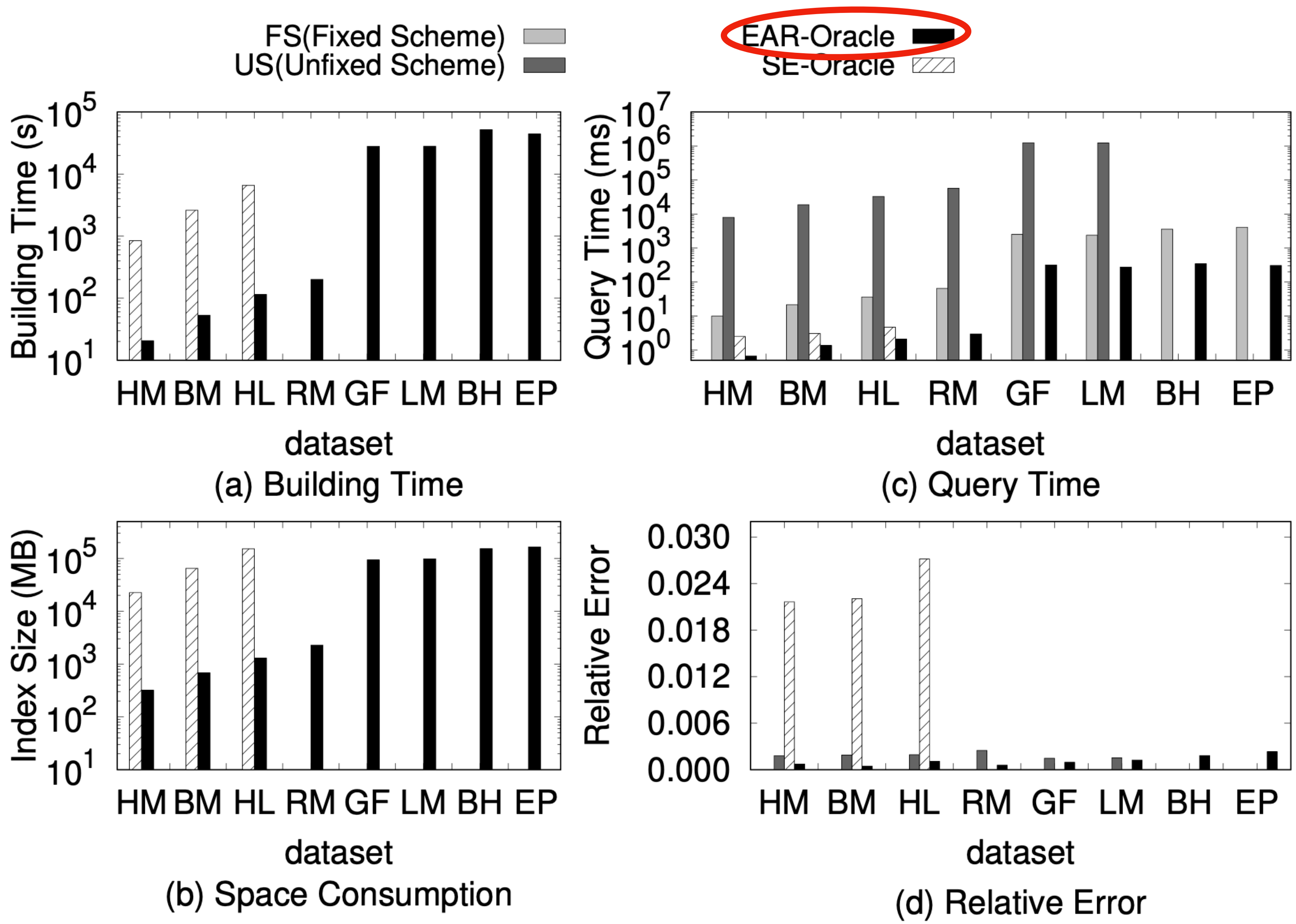
- Result on *unweighted* terrain datasets (under default parameter setting):
  - *EAR-Oracle* outperforms *SE-Oracle* by *2 orders* of magnitude in terms of *building time* and *space consumption*.
  - *EAR-Oracle* outperforms *other tested algorithms* by *more than 1 order* of magnitude in terms of *query time*.
  - All tested algorithms have *small relative error*.



(a) Building Time
(c) Query Time
(b) Space Consumption
(d) Relative Error

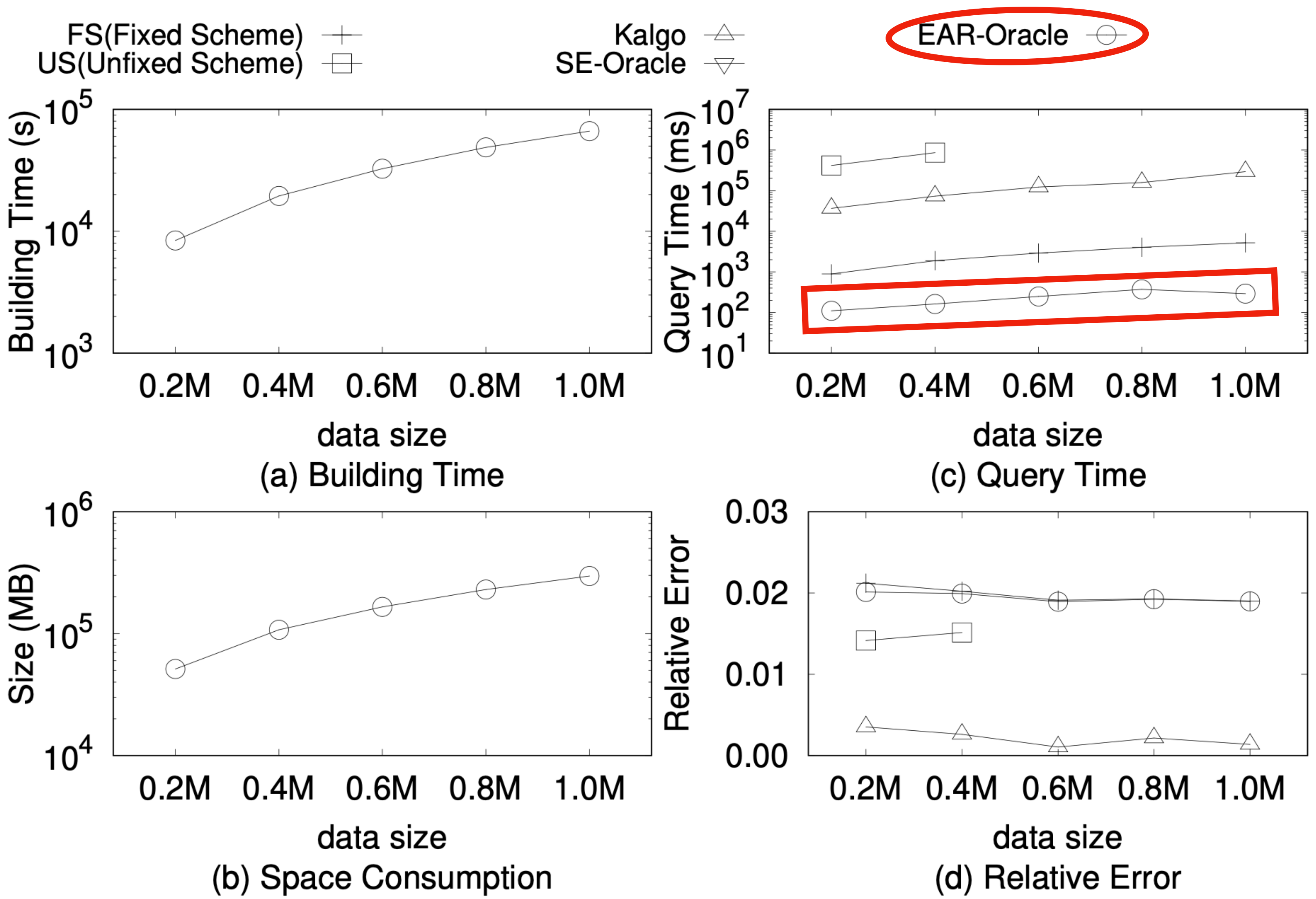
Experimental Result

- Result on *weighted* terrain datasets (under default parameter setting):
  - *Fixed Scheme (FS) is selected as the pivot* for error comparison (exact distance is expensive to compute);
  - *Similar results* as the unweighted datasets.



Experimental Result

- *Scalability* test on high resolution EP dataset (w.r.t number of faces):
  - *SE-Oracle* *exceeds memory budget* for a dataset with only 200,000 faces;
  - *EAR-Oracle* can *scale up* to dataset with 1 million faces;
  - *EAR-Oracle* outperforms *all on-the-fly* algorithms by more than 1 order of magnitude in terms of *query time*.



Outline

- Introduction
- Proposed Solution
- Experimental Result
- Conclusion

Conclusion

- The geodesic distance problem is both *fundamental* and *important* for many high-level applications;

- We propose *EAR-Oracle*:
  - *No assumption* on query points;
  - *Outperforms* the state-of-the-art algorithms;
  - *Can scale up* to terrain surfaces with millions of faces;
  - *Quality guarantee* on result.

Thanks for your attention!

Support materials

Theoretical Analysis

- Let O^* be the O notation hiding terrain related constants:
  - The *building time* of *EAR-Oracle* is:
    - $O^*(\zeta mN \log(mN) + \frac{N \log N}{\epsilon^2} + N \log N + \frac{N}{\epsilon^2})$:
      - For N: *larger* terrain dataset yields *longer* building time;
      - For ϵ: *tighter* (*smaller*) error bound yields *longer* building time;
      - For m: *more* auxiliary points yields *longer* building time;
      - For ζ: *more* highway nodes yields *longer* building time;

Theoretical Analysis

- Let O^* be the O notation hiding terrain related constants:
  - The *space consumption* of *EAR-Oracle* is:
    - $O^*(\frac{mN}{\zeta} + \frac{N}{\epsilon^2})$:
      - For N: *larger* terrain dataset yields *more* space;
      - For ϵ: *tighter* (*smaller*) error bound yields *more* space;
      - For m: *more* auxiliary points yields *more* space;
      - For ζ: *more* highway nodes yields *less* space;

Theoretical Analysis

- Let O^* be the O notation hiding terrain related constants:
  - The *query time* of *EAR-Oracle* is: $O^*(\zeta \log \zeta)$

    \- *Only* related to number of highway nodes;

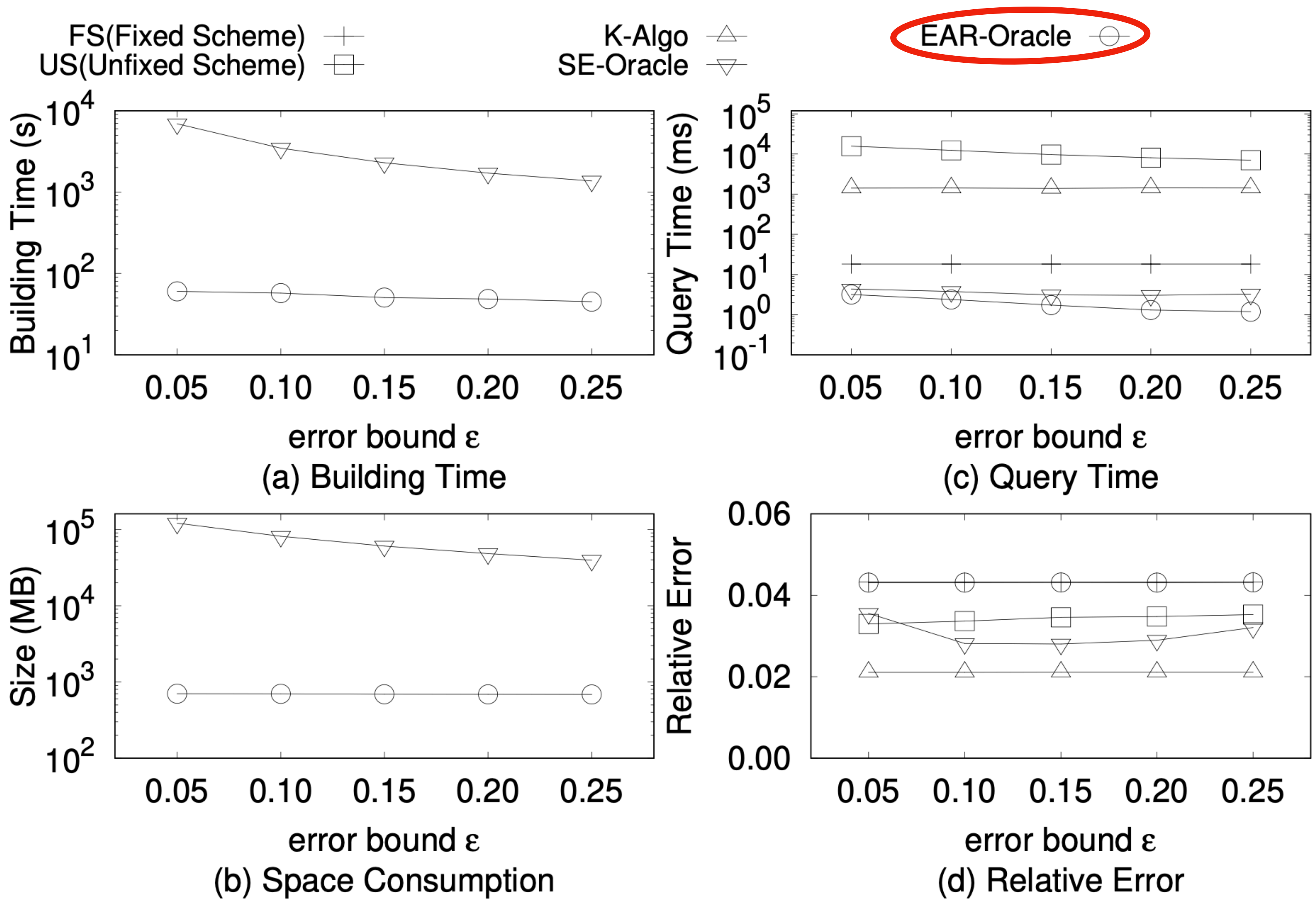
    \- ζ: *more* highway nodes yields *more* query time.

Theoretical Analysis

- Let δ be the distance error of *FS*:
  - The *distance error* of *EAR-Oracle*:
    - $\tilde{d}_g(s,t) \leq (1+\epsilon)(d_g(s,t) + 2\delta)$

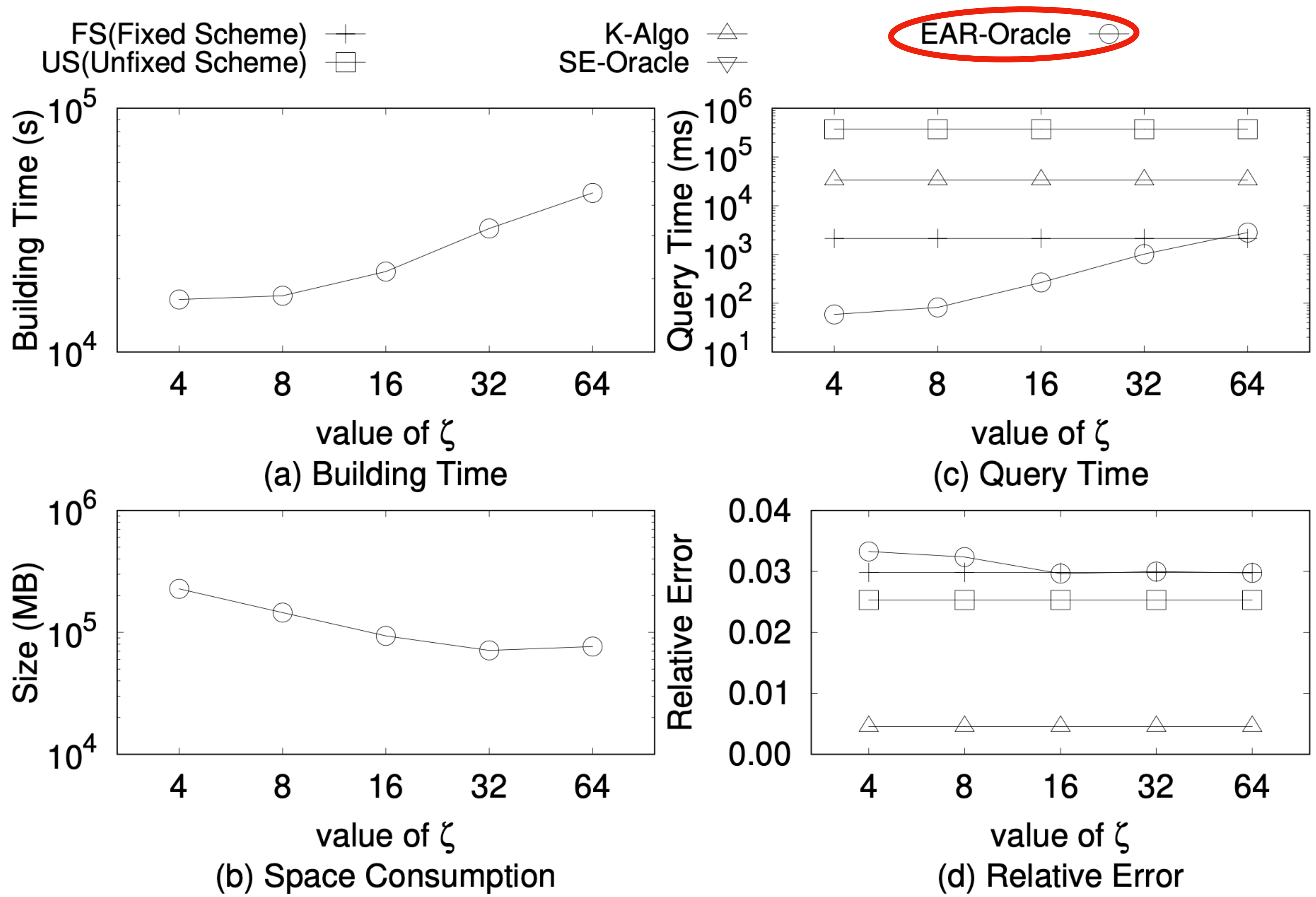
Experimental Result

- *Effect of ϵ* on BM dataset:
  - *Larger ϵ* (looser error bound) yields *better performance* of EAR-Oracle.



Experimental Result

- *Effect of ζ* on BM dataset:
  - *Larger ζ* (more boundary vertices) yields *more building time*, *query time* and *less space* of EAR-Oracle.



Experimental Result

- *Effect of m* on BM dataset:
  - *Larger m* (more Steiner points) yields *more building time*, *query time*, *space consumption* and *higher result quality* of *EAR-Oracle*.