Indexing Land Surface for Efficient kNN Query

Cyrus Shahabi, Lu-An Tang and Songhua Xing
InfoLab
University of Southern California
Los Angeles, CA 90089-0781
http://infolab.usc.edu

Outline

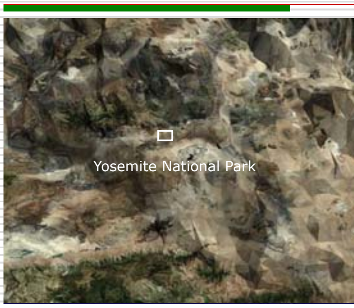
VLDB'08

- Motivation
- Related Work
- Background
- Indexing Land Surface
- Query Processing
- Performance Evaluation
- Conclusion and Future Work

2

Motivation

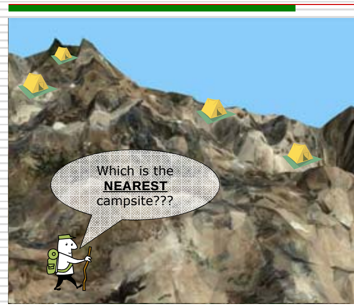
VLDB'08



3

Motivation

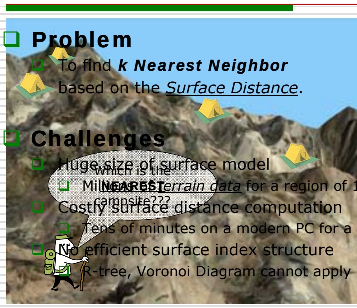
VLDB'08



4

Motivation

VLDB'08



- **Problem**
  - To find ***k* Nearest Neighbor** based on the *Surface Distance*.
- **Challenges**
  - Huge size of surface model
    - Millions of *terrain data* for a region of 10km×10km
  - Costly surface distance computation
    - Tens of minutes on a modern PC for a terrain of 10,000
  - No efficient surface index structure
    - R-tree, Voronoi Diagram cannot apply directly.

5

Outline

VLDB'08

- Motivation
- Related Work
- Background
- Indexing Land Surface
- Query Processing
- Performance Evaluation
- Conclusion and Future Work

7

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | Surface

- Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

8

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | Surface

- Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ NN Query: Roussopoulos et al., SIMGOD 1995

9

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | Surface

- Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ NN Query: Roussopoulos et al., SIMGOD 1995
✓ Influences Set: Korn et al., SIMGOD 2000
✓ FINCH Algorithm: Wu et al,. VLDB 2008

10

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | Surface

- Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ NN Query: Roussopoulos et al., SIMGOD 1995
✓ Influences Set: Korn et al., SIMGOD 2000
✓ FINCH Algorithm: Wu et al,. VLDB 2008
✓ Time-parameterized queries : Tao et al., SIMGOD 2002
✓ Continuous NN Search: Tao et al,. VLDB 2002

11

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | Surface

- Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ NN Query: Roussopoulos et al., SIMGOD 1995
✓ Influences Set: Korn et al., SIMGOD 2000
✓ FINCH Algorithm: Wu et al,. VLDB 2008
✓ Time-parameterized queries : Tao et al., SIMGOD 2002
✓ Continuous NN Search: Tao et al,. VLDB 2002
✓ VkNN Query: Nutanong et al., DASFAA 2007

12

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | **Road Networks** | Surface

- Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ **Query Processing in SNDB : Papadias et al., VLDB 2003**
✓ **V-based kNN in SNDB: Shahabi et al., VLDB 2004**
✓ **RNN in Large Graphs: Yiu et al., TKDE 2006**
✓ **CNN Monitoring in RN: Mouratidis et al., VLDB 2006**

13

Related Work

VLDB'08

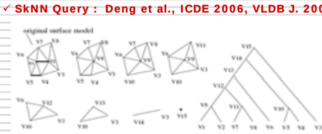
Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | **Surface**

- ☞ Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ **SkNN Query : Deng et al., ICDE 2006, VLDB J. 2008**



14

Related Work

VLDB'08

Spatial Database

kNN Query Processing

Euclidean Space | Road Networks | **Surface**

- ☞ Conventional kNN
- Reverse kNN
- Time-aware kNN
- Visible kNN

✓ **SkNN Query : Deng et al., ICDE 2006, VLDB J. 2008**
- ❑ **Not an incremental approach**
- ❑ **Not an exact approach**

15

Outline

VLDB'08

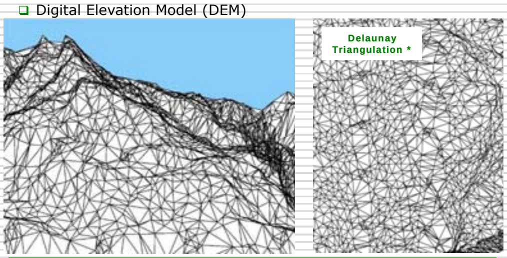
- ❑ Motivation
- ❑ Related Work
- ☑ Background
- ❑ Indexing Land Surface
- ❑ Query Processing
- ❑ Performance Evaluation
- ❑ Conclusion and Future Work

16

Background

VLDB'08

- ❑ **Triangular Irregular Network (TIN) Model**
  - ❑ Triangular Mesh
  - ❑ Digital Elevation Model (DEM)



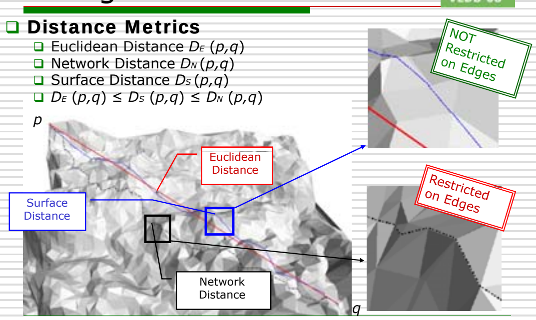
* **Computational Geometry:** Algorithms and Applications *(BERG, M., KREVELD, M., OVRMARS, M., SCHWARZKOPF, O.)*

17

Background

VLDB'08

- ❑ **Distance Metrics**
  - ❑ Euclidean Distance $D_E(p,q)$
  - ❑ Network Distance $D_N(p,q)$
  - ❑ Surface Distance $D_S(p,q)$
  - ❑ $D_E(p,q) \le D_S(p,q) \le D_N(p,q)$

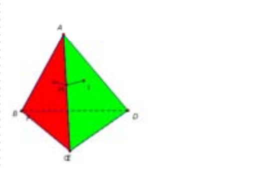


18

Background

VLDB'08

- **Shortest Surface Path Computation**
  - Chen-Han (CH) Algorithm * : *unfold* all the faces of a polyhedron to one plane
  - Time Complexity: $O(n^2)$, *n* is the total number of the vertices on the surface



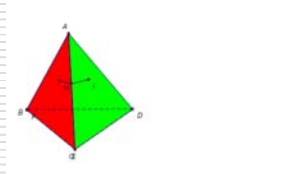
* **Shortest paths on a polyhedron**: CHEN, J., HAN, Y., Computational Geometry 1990

19

Background

VLDB'08

- **Shortest Surface Path Computation**
  - Chen-Han (CH) Algorithm * : *unfold* all the faces of a polyhedron to one plane
  - Time Complexity: $O(n^2)$, *n* is the total number of the vertices on the surface



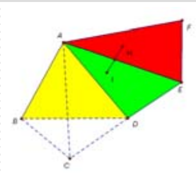
* **Shortest paths on a polyhedron**: CHEN, J., HAN, Y., Computational Geometry 1990

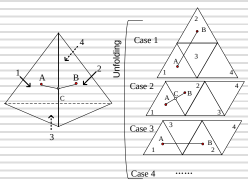
20

Background

VLDB'08

- **Shortest Surface Path Computation**
  - Chen-Han (CH) Algorithm * : *unfold* all the faces of a polyhedron to one plane
  - Time Complexity: $O(n^2)$, *n* is the total number of the vertices on the surface





* **Shortest paths on a polyhedron**: CHEN, J., HAN, Y., Computational Geometry 1990

21

Outline

VLDB'08

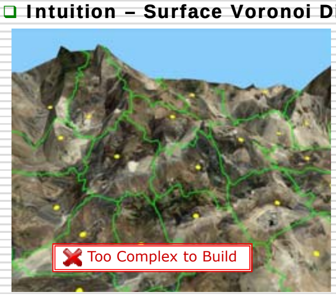
- Motivation
- Related Work
- Background
- Indexing Land Surface
- Query Processing
- Performance Evaluation
- Conclusion and Future Work

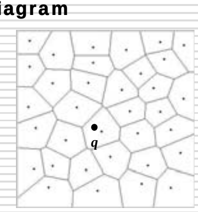
22

Indexing Land Surface

VLDB'08

- **Intuition – Surface Voronoi Diagram**





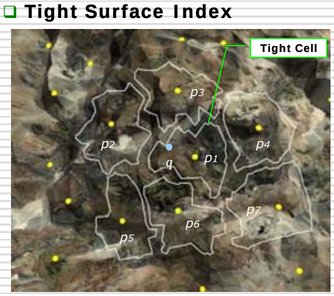
Voronoi Diagram

23

Indexing Land Surface

VLDB'08

- **Tight Surface Index**



$TC(p_i)=\{q: q \in T$ and $D_N(p_i , q) < D_E(p_j, q)\ (\forall p_j \in P, p_j \neq p_i)\}$

For any query point $q \in TC(p_i)$, the nearest neighbor of *q* in surface distance is p_i.

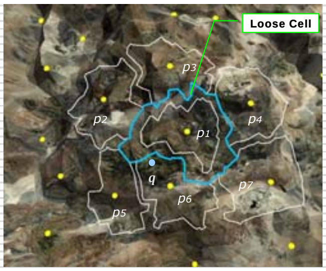
$D_S (p_i , q) \leq D_N (p_i , q) < D_E(p_j, q) \leq D_S (p_j , q)$ $(\forall p_j \in P, p_j \neq p_i)\}$

24

Indexing Land Surface

VLDB'08

- **Loose Surface Index**



LC(pi)={*q*: *q* ∈*T* and DE (*pi* , *q*) < DN(*pj*, *q*) (∀*pj* ∈*P*, *pj* ≠ *pi*)}

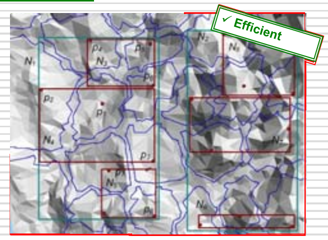
Site *pi* is guaranteed not to be the nearest neighbor of *q* if *q* is outside *LC(pi)*.

∃*pj* ∈*P* (*pj* ≠ *pi*) such that DS(*pi*, *q*)≥ DE(*pi*, *q*) > DN(*pj*, *q*) ≥ DS(*pj*, *q*)

Indexing Land Surface

VLDB'08

- **Storage Scheme**
  - **R-Tree?**
    - **Unlike the Voronoi diagram, tight/loose cell are concave polygons in most cases and much more irregular**
    - **All cells are adjacent to each other, causing too much overlapping in R-Tree**
    - **Index both on TC/LC**
  - **Solution: *SIR-tree***
    - **An R-tree that is generated on site set *P***
    - **Leaf node stores: sites inside the corresponding MBR, the pointer to the vertices list of the tight / loose cell and its neighbor list**

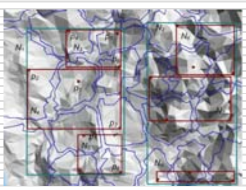


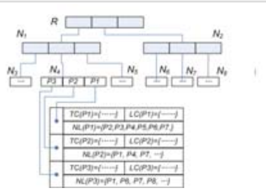
* For the purpose of clarity, textures on terrain are removed.

Indexing Land Surface

VLDB'08

- **SIR-Tree**
  - **An R-tree that is generated on site set *P***
  - **Leaf node stores: sites inside the corresponding MBR, the pointer to the vertices list of the tight / loose cell and its neighbor list**



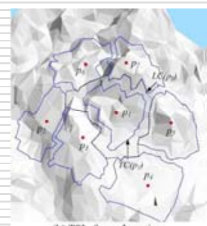


Indexing Land Surface

VLDB'08

- **SIR-Tree Insertion**
  - **Algorithm**

1. **locate *p* in *I*, find out the loose cell *LC(r)* containing *p*;**
2. ***p*.neighbor ← *LC(r)*'s neighbor;**
3. **compute *TC(p)* and *LC (p)*;**
4. **for each site p_i in *p*.neighbor**
5. **update *LC*(p_j)'s edges according to *TC(p)*;**
6. **update *TC*(p_j)'s edges according to *LC(p)*;**
7. **insert *p* into *I*;**
8. **return *I*;**



(b) TSI after p_1 Insertion

Indexing Land Surface

VLDB'08

- **More about TSI and LSI**
  - Definitions:
    - **TSI**, **LSI** and **Neighbor**
    - Please refer to *Section 4.1, 4.2* in the paper.
  - Observation:
    - Given that TSI and LSI are generated for the same site set *P*, the tight and loose cells have common edges; more specifically, all the tight cell's edges are also the edges of loose cells.
    - Please refer to *Section 4.2 Property 3* in the paper.
  - TSI and LSI Construction
    - Naïve Index Construction
    - Fast Index Construction
    - Please refer to *Section 4.3* in the paper.

Outline

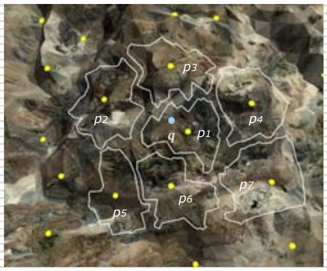
VLDB'08

- Motivation
- Related Work
- Background
- Indexing Land Surface
- ✓ Query Processing
- Performance Evaluation
- Conclusion and Future Work

Query Processing

VLDB'08

- **Nearest Neighbor Query**
  - If the query point falls into one tight cell, its nearest neighbor could be identified immediately without any surface distance computation.
  - Our experiment shows about 75% queries fall into one of these tight cells.

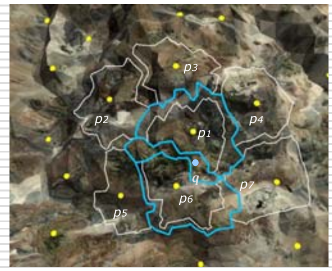


31

Query Processing

VLDB'08

- **Nearest Neighbor Query**
  - If the query point falls out of all tight cells, we need to unfold all loose cells that contain the query point to compute its surface distance to the candidates.
  - Search (i.e., number of candidates we need compute distance to) is *localized* in loose cells.

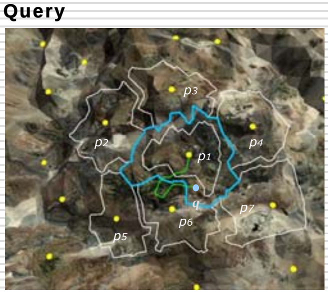


32

Query Processing

VLDB'08

- **Nearest Neighbor Query**
  - If p_i is the nearest neighbor of q, then the shortest surface path from q to p_i is inside the loose cell $LC(p_i)$. *
  - Computation (i.e., unfolding: invocation of CH algorithm) is *localized* in loose cells.



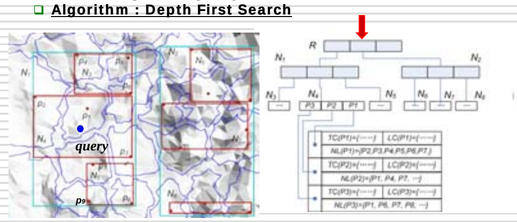
* Please refer to *Section 4.2 Property 4* of the paper for proof.

33

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



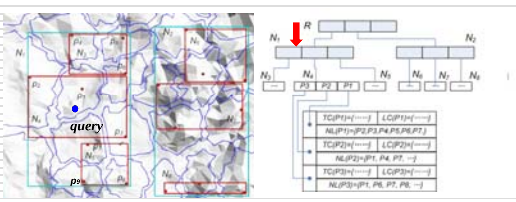
Current Node	Stack: Nodelist
Root	

34

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



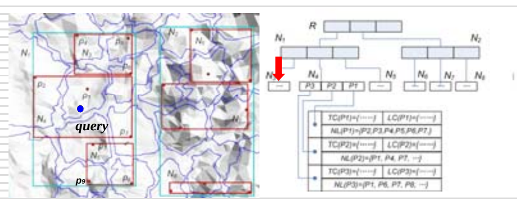
Current Node	Stack: Nodelist
N_1	N_1

35

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



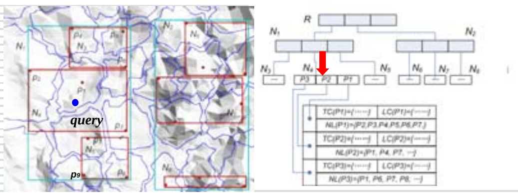
Current Node	Stack: Nodelist	
N_3	N_1	N_3

36

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search



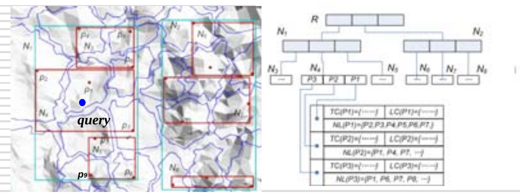
Current Node	Stack: Nodelist						
N4	*N1*	*N4*					

37

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search



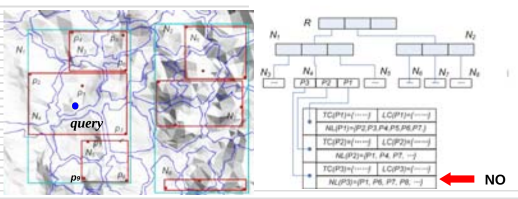
Current Node	Stack: Nodelist						
N4	*N1*	*N4*					

38

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search



NO

Does *TC(P3)* or *LC(P3)* contain *q*?

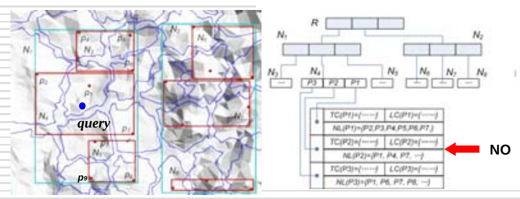
Current Node	Stack: Nodelist						
N4	*N1*	*N4*					

39

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search



NO

Does *TC(P2)* or *LC(P2)* contain *q*?

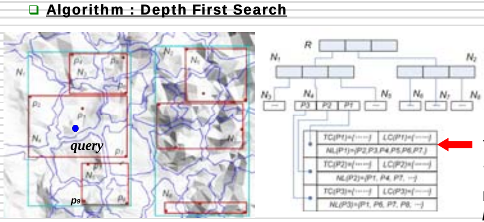
Current Node	Stack: Nodelist						
N4	*N1*	*N4*					

40

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search



YES,

TC(P1)

Return *p1* as NN

Does *TC(P1)* or *LC(P1)* contain *q*?

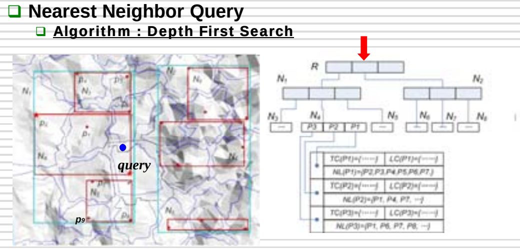
Current Node	Stack: Nodelist						
N4	*N1*	*N4*					

41

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search



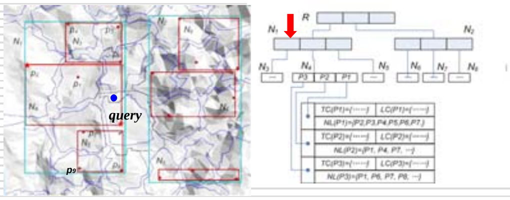
Current Node	Stack: Nodelist						
Root							

42

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



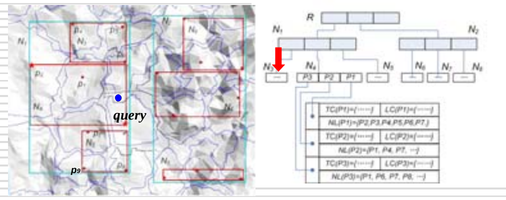
Current Node	Stack: Nodelist						
N_1	N_1						

43

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



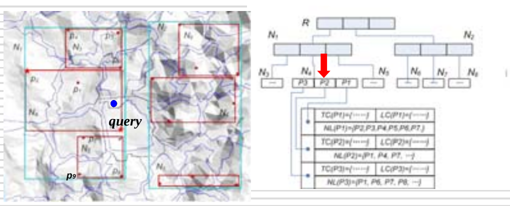
Current Node	Stack: Nodelist						
N_3	N_1	N_3					

44

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



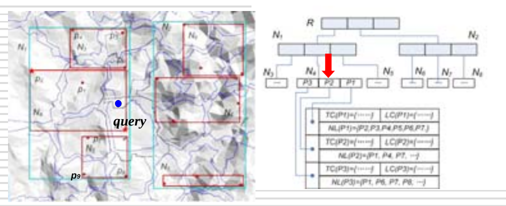
Current Node	Stack: Nodelist						
N_4	N_1	N_4					

45

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



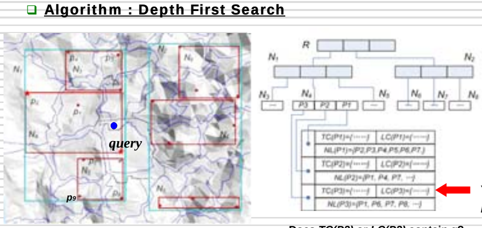
Current Node	Stack: Nodelist						
N_4	N_1	N_4					

46

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



YES, *LC(P3)*

Does *TC(P3)* or *LC(P3)* contain *q*?

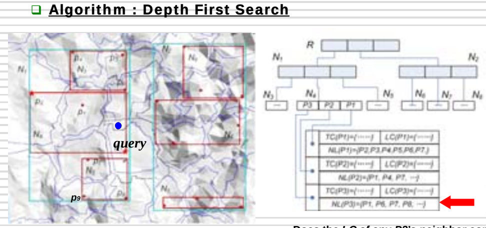
Current Node	Stack: Nodelist / Candidate Set C						
N_4	N_1 / P_3	N_4					

47

Query Processing

VLDB'08

- Nearest Neighbor Query
  - **Algorithm : Depth First Search**



YES, *LC(P6)*

Does the *LC* of any *P3*'s neighbor contain *q*?,

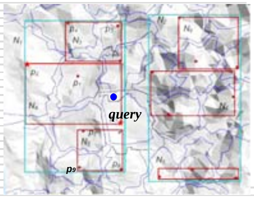
Candidate Set C						
P_3	P_6					

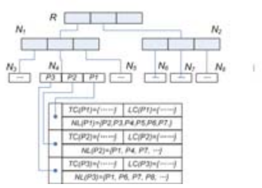
48

Query Processing

VLDB'08

- Nearest Neighbor Query
  - Algorithm : Depth First Search





Candidate Set C

P_3	P_6					

***Unfold** the area covered by LC(p3) and LC(p6) and compute their surface distance to q by **CH algorithm** and return the p3 as NN .*

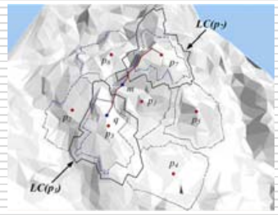
49

Query Processing

VLDB'08

- k Nearest Neighbor Query
  - Property 4

The next nearest site is the generator of one of the neighbors of the NNs found so far.



Therefore, The shortest surface path from *q* to the *k*-th NN p_k will lie in the area of *LC(G)* U $LC(p_k)$ = $LC(p_1)$ U $LC(p_2)$ U … U $LC(p_k)$.

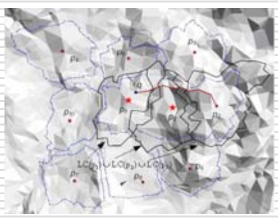
50

Query Processing

VLDB'08

- k Nearest Neighbor Query
  - ***Algorithm***

```
kNN Query (SIR-tree I, point q, surface T)
    p ← Nearest Neighbor Query(I, q, T);
    add p to kNN set G;
    initialize minimum heap H;
    while(G.size < k)
    for each neighbor site p_i of G;
    unfold LC(G) U LC(p_i) to compute surface distance;
    add p_i to H;
    end for
   p← deheap H;
   add p to G;
 end while;
 return G;
```



51

Query Processing

VLDB'08

- **More about Query Processing**
  - Surface Index R-Tree (<u>SIR-tree</u>)
    - How an R-tree is built on TSI and LSI?
    - SIR-tree insertion
    - Please refer to *Section 4.4* in the paper.
  - NN Query Algorithm
    - Please refer to *Section 5.1 Algorithm 3* in the paper.
  - kNN Query Processing
    - Property of next nearest neighbor
    - Incremental algorithm for kNN Query
    - Please refer to *Section 5.2* in the paper.

52

Outline

VLDB'08

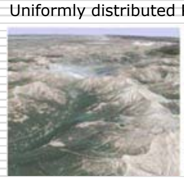
- Motivation
- Related Work
- Background
- Indexing Land Surface
- Query Processing
- Performance Evaluation
- Conclusion and Future Work

53

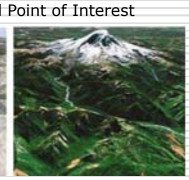
Performance Evaluation

VLDB'08

- **Dataset** *
  - Eagle Peak (EP) at Wyoming State, USA
    - 10.7km×14km, 1.4M sampled points.
  - Bearhead (BH) at Washington State, USA
    - Similar size as above, 1.3M sampled points.
  - Uniformly distributed Point of Interest



Eagle Peak (EP)



Bearhead (BH)

* http://data.geocomm.com/

54

Performance Evaluation

VLDB'08

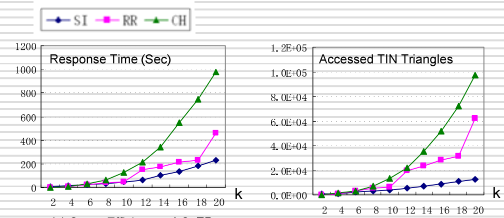
- **Competing Approaches**
  - Surface Index (SI)
    - Exact and quick answer
  - Range Ranking (RR)
    - Approximate and quick answer
  - Chen Han Algorithm (CH)
    - Exact and slow answer

55

Performance Evaluation

VLDB'08

- **Query Efficiency, I/O cost vs. Value of k**
  - The difference in improvement of SI over CH increases for larger k.



(a) Query Efficiency, d=2, EP

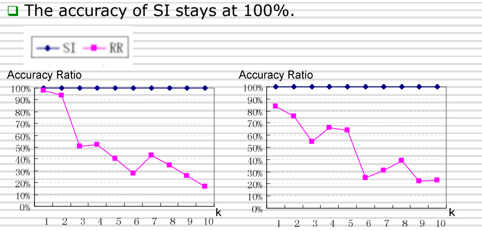
(e) I/O Cost, d=2, EP

56

Performance Evaluation

VLDB'08

- **Accuracy vs. Value of k**
  - The accuracy of RR drops dramatically when the value of k increases.
  - The accuracy of SI stays at 100%.



(a) Query Accuracy on EP

(b) Query Accuracy on BH

57

Outline

VLDB'08

- Motivation
- Related Work
- Background
- Indexing Land Surface
- Query Processing
- Performance Evaluation
- Conclusion and Future Work

58

Conclusion and Future Work

VLDB'08

- **Conclusion**
  - We extend the traditional kNN Query to the space constrained with the third dimension.
  - We construct two complementary indexing schemes, namely Tight Surface Index (TSI) and Loose Surface Index (LSI) to reduce the invocation of the costly surface distance computation.
  - SI significantly outperforms its competitors in accuracy and efficiency.
- **Future Work**
  - Further evaluate its performance with synthetic datasets.
  - Study variations of *skNN* such as the *continuous skNN query*, *dynamic skNN query* and *visible skNN query*.

59

Thanks!

VLDB'08

Email: Songhua Xing
sxing@usc.edu

InfoLab

USC IMSC

60