Multi-Approximate-Keyword Routing Query

Bin Yao[1], Mingwang Tang[2], Feifei Li[2]

[1]Department of Computer Science and Engineering
Shanghai Jiao Tong University, P. R. China

[2]School of Computing
University of Utah, USA

Outline

Introduction and motivation

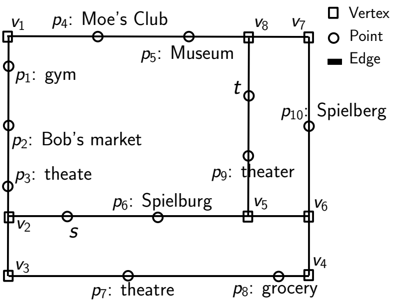
- Approximate keyword search is important:
  - GIS data has errors and uncertainty with it.
  - GIS data is keeping evolving, routinely data cleaning and data integration is expensive
  - People may make mistakes in query input (typos)

Introduction and motivation

- Approximate keyword search is important:
  - GIS data has errors and uncertainty with it.
  - GIS data is keeping evolving, routinely data cleaning and data integration is expensive
  - People may make mistakes in query input (typos)
- Shortest path search has many applications:
  - map service.
  - strategic planning of resources

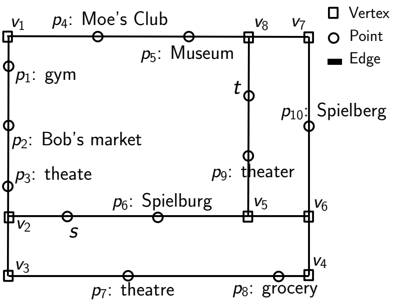
Introduction and motivation

- Approximate keyword search is important:
  - GIS data has errors and uncertainty with it.
  - GIS data is keeping evolving, routinely data cleaning and data integration is expensive
  - People may make mistakes in query input (typos)
- Shortest path search has many applications:
  - map service.
  - strategic planning of resources
- Our work: Multi-Approximate-Keyword Routing (MAKR) query.
  - A combination of shortest path search and approximate keyword search
  - Given a source and destination pair (s, t) and a query keyword set ψ on a road network, the goal is to find the shortest path that passes through at least one matching object per keyword.



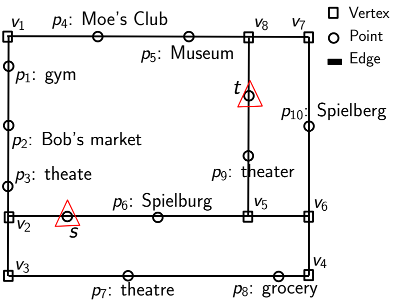
v_1
p_4: Moe's Club
v_8
v_7
p_5: Museum
p_1: gym
t
p_{10}: Spielberg
p_2: Bob's market
p_3: theate
p_9: theater
p_6: Spielburg
v_5
v_6
v_2
s
v_3
v_4
p_7: theatre
p_8: grocery
Vertex
Point
Edge

Multi-Approximate-Keyword Routing (MAKR) query



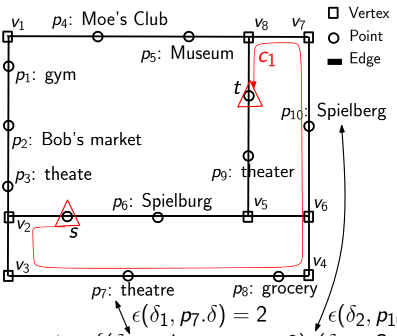
Approximate string similarity:
edit distance $\epsilon(\delta_1, \delta_2) = \tau$.

Multi-Approximate-Keyword Routing (MAKR) query



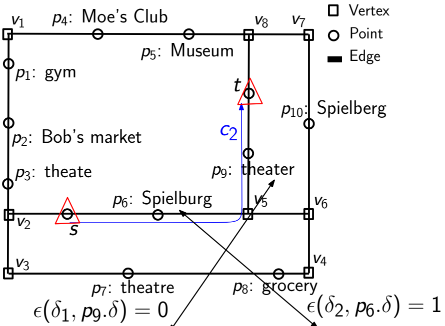
Example: $s, t, \psi = \{(\delta_1 = \text{theater}, \tau_1 = 2), (\delta_2 = \text{Spielberg}, \tau_2 = 1)\}$.

Multi-Approximate-Keyword Routing (MAKR) query



Example: $s, t, \psi = \{(\delta_1 = \text{theater}, \tau_1 = 2), (\delta_2 = \text{Spielberg}, \tau_2 = 1)\}$.

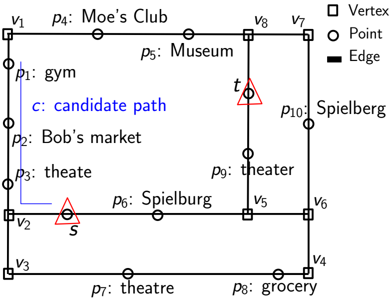
Multi-Approximate-Keyword Routing (MAKR) query



Example: $s, t, \psi = \{(\delta_1 = \text{theater}, \tau_1 = 2), (\delta_2 = \text{Spielberg}, \tau_2 = 1)\}$.

path length: $d(c_2) < d(c_1)$

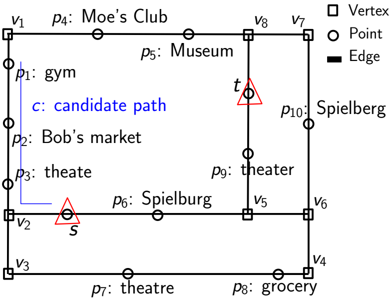
Multi-Approximate-Keyword Routing (MAKR) query



Example: $s, t, \psi = \{(\delta_1 = \text{theater}, \tau_1 = 2), (\delta_2 = \text{Spielberg}, \tau_2 = 1)\}$.

$\psi(c) = \{\delta_1 = \mathit{theater}\}$

Multi-Approximate-Keyword Routing (MAKR) query



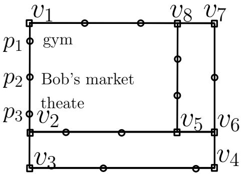
Example: $s, t, \psi = \{(\delta_1 = \text{theater}, \tau_1 = 2), (\delta_2 = \text{Spielberg}, \tau_2 = 1)\}$.

$\psi(c) = \{\delta_1 = theater\}$

$|\psi| = \kappa$, when $\psi(c) = \psi$, c becomes a qualified path

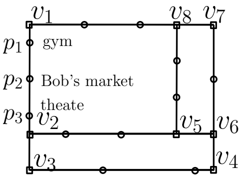
Outline

Data structure: Disk-based storage of the road network

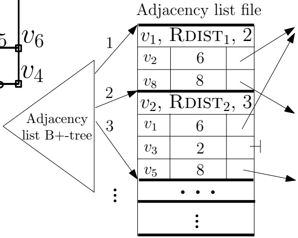


$d(v_1, v_2) = 6$
$d(v_1, v_8) = 8$
$d(v_1, p_1) = 1$
$d(v_1, p_2) = 3$
$d(v_1, p_3) = 5$

v_i: network vertex.



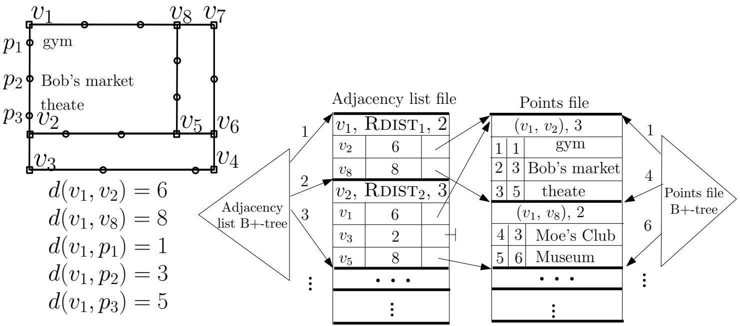
$$d(v_1, v_2) = 6$$
$$d(v_1, v_8) = 8$$
$$d(v_1, p_1) = 1$$
$$d(v_1, p_2) = 3$$
$$d(v_1, p_3) = 5$$



v_i: network vertex.

$RDIST_i$: distances to the landmarks.

Data structure: Disk-based storage of the road network

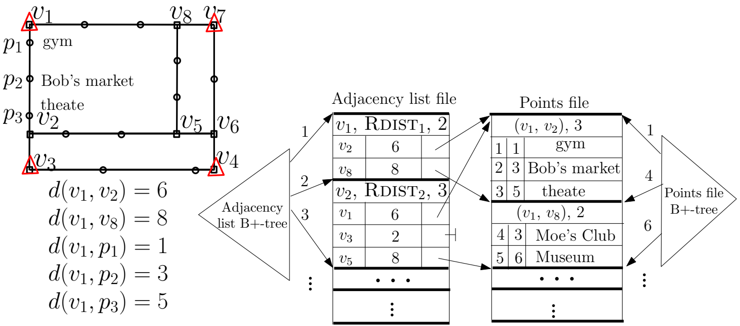


v_i: network vertex.

$RDIST_i$: distances to the landmarks.

- [sl97]: CCAM: A connectivity-clustered access method for networks and network computations. In IEEE TKDE, 1997.

Data structure: Disk-based storage of the road network

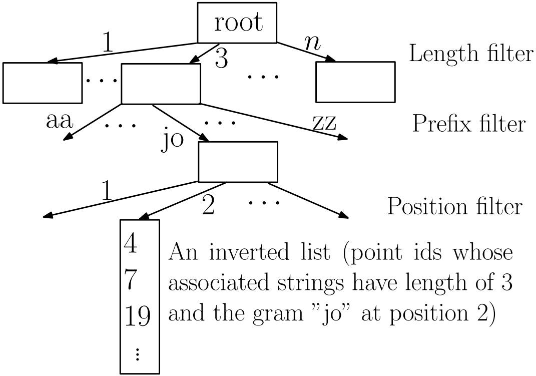


v_i: network vertex.

$RDIST_i$: distances to the landmarks.

- [sl97]: CCAM: A connectivity-clustered access method for networks and network computations. In IEEE TKDE, 1997.
- [gh05]: Computing the shortest path: A* search meets graph theory. In SODA, 2005.

Data structure: FilterTree for Approximate Keywords-Matching



- [lll08]: Efficient merging and filtering algorithms for approximate string searches. In ICDE, 2008.

Exact solution overview

- Intuition: **PER–P**ath **E**xpansion and **R**efinement.

Exact solution overview

- Intuition: **PER–P**ath **E**xpansion and **R**efinement.

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

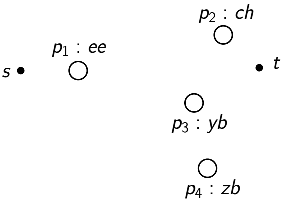
For each keyword $w \in \psi - \psi(c)$, add a point p from $P(w)$ into current shortest candidate path, s.t. $\forall p \in P(w), \epsilon(p.\delta, w) \leq \tau_w$, to minimize the impact to $d(c)$

Exact solution overview

- Intuition: **PER–P**ath **E**xpansion and **R**efinement.

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

For each keyword $w \in \psi - \psi(c)$, add a point p from $P(w)$ into current shortest candidate path, s.t. $\forall p \in P(w), \epsilon(p.\delta, w) \leq \tau_w$, to minimize the impact to $d(c)$



L

$\{s, p_1, t\}$
$\{s, p_3, t\}$
$\{s, p_2, t\}$

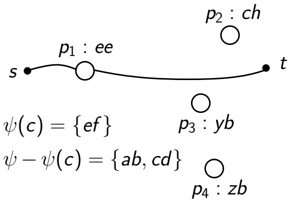
IO efficient priority queue of candidate paths: initialized with c's tha each covers a din- stinct, single $w \in \psi$

Exact solution overview

- Intuition: **PER–P**ath **E**xpansion and **R**efinement.

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

For each keyword $w \in \psi - \psi(c)$, add a point p from $P(w)$ into current shortest candidate path, s.t. $\forall p \in P(w), \epsilon(p.\delta, w) \leq \tau_w$, to minimize the impact to $d(c)$

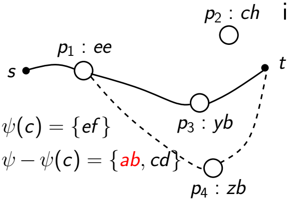


Exact solution overview

- Intuition: **PER–P**ath **E**xpansion and **R**efinement.

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

For each keyword $w \in \psi - \psi(c)$, add a point p from $P(w)$ into current shortest candidate path, s.t. $\forall p \in P(w), \epsilon(p.\delta, w) \leq \tau_w$, to minimize the impact to $d(c)$



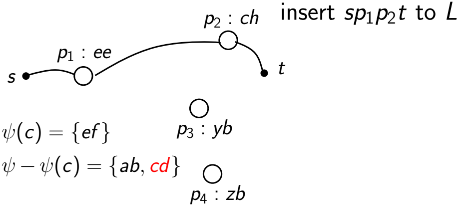
insert sp_1p_3t to L

Exact solution overview

- Intuition: **PER–P**ath **E**xpansion and **R**efinement.

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

For each keyword $w \in \psi - \psi(c)$, add a point p from $P(w)$ into current shortest candidate path, s.t. $\forall p \in P(w), \epsilon(p.\delta, w) \leq \tau_w$, to minimize the impact to $d(c)$



$\psi(c) = \{ef\}$

$\psi - \psi(c) = \{ab, cd\}$

Exact solution overview

- Improvement.
  - use Landmarks to estimate distances when finding points;
  - modify and then combine with FilterTree to find $p \in P(w)$ incrementally;
  - refine $d(c)$ when c becomes a qualified path.
    - two methods to refine $d(c)$: PER-full and PER-partial

Outline

Approximate solutions for MAKR query

- Problem with the exact solution:
  Theorem 1: The MAKR problem is NP-hard.

Approximate solutions for MAKR query

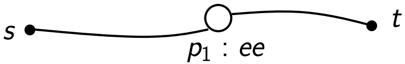
- Problem with the exact solution:
  Theorem 1: The MAKR problem is NP-hard.
- Approximate solutions:
  - The local minimum path algorithms: A_{LMP1} and A_{LMP2}.
  - The global minimum path algorithm: A_{GMP}.

The local minimum distance algorithms: A_{LMP1} and A_{LMP2}

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

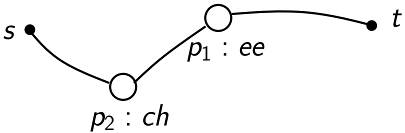
For each segment (p_i, p_j), find a point p, $p.\delta$ similar to keywords in $\psi - \psi(c)$, to minimize sum of d$(p_i,\ p)$ and d$(p,\ p_j)$.



The local minimum distance algorithms: A_{LMP1} and A_{LMP2}

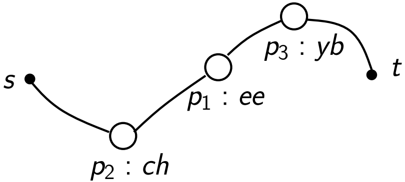
$$Q: s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

For each segment (p_i, p_j), find a point p, $p.\delta$ similar to keywords in $\psi - \psi(c)$, to minimize sum of d(p_i, p) and d(p, p_j).



$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

For each segment (p_i, p_j), find a point p, $p.\delta$ similar to keywords in $\psi - \psi(c)$, to minimize sum of d(p_i, p) and d(p, p_j).



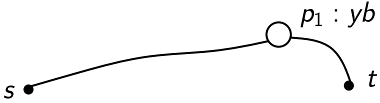
The local minimum distance algorithms: A_{LMP1} and A_{LMP2}

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

The local minimum distance algorithms: A_{LMP1} and A_{LMP2}

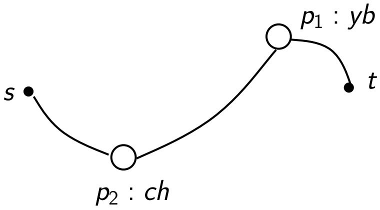
$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$

For each keyword $w \in \psi - \psi(c)$, we iterate through the segments in c and add the point $p \in P(w)$, which minimizes $d(c)$, to one segment (p_i, p_j) of c.



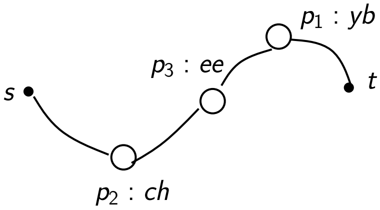
$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$

For each keyword $w \in \psi - \psi(c)$, we iterate through the segments in c and add the point $p \in P(w)$, which minimizes $d(c)$, to one segment (p_i, p_j) of c.



$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$

For each keyword $w \in \psi - \psi(c)$, we iterate through the segments in c and add the point $p \in P(w)$, which minimizes $d(c)$, to one segment (p_i, p_j) of c.

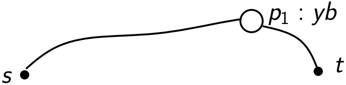


The minimum distance algorithm: A_{GMP}

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$

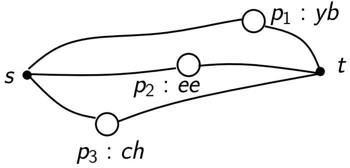
For each keyword $w \in \psi - \psi(c)$, find a point $p \in P(w)$ to minimize sum of d(s, p) and d(p, t).



The minimum distance algorithm: A_{GMP}

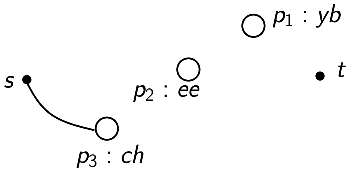
$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$

For each keyword $w \in \psi - \psi(c)$, find a point $p \in P(w)$ to minimize sum of d(s, p) and d(p, t).

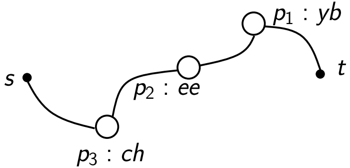


The minimum distance algorithm: A_{GMP}

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$

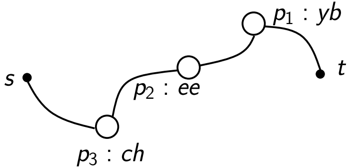


$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$



The minimum distance algorithm: A_{GMP}

$$Q : s, t, \psi = \{(ab, 1), (cd, 1), (ef, 1)\}$$



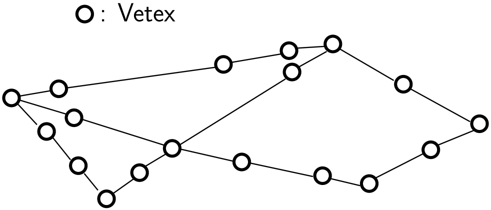
- Theorem 2: The A_{GMP} algorithm gives a κ-approximate path. This bound is tight.

Challenges in approximate solutions

- Challenges in all approximate methods:
  - how to find $p \in P(w)$ incrementally for each type of objective function (instead of finding $P(w)$ all at once and iterate through points in $P(w)$ one by one)?
  - how to avoid exact distance computation as much as possible?

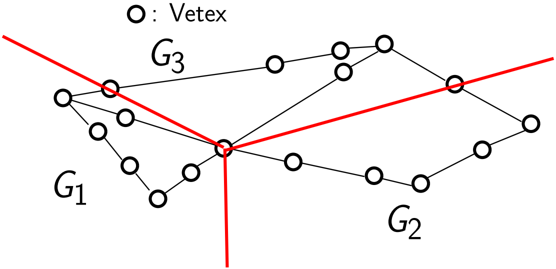
Improvement on approximate solutions by network partitioning

- Voronoi-diagram-like partition (by Erwig and Hagen's algorithm).



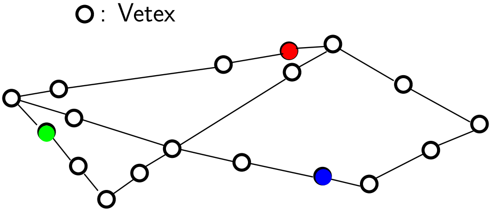
Improvement on approximate solutions by network partitioning

- Voronoi-diagram-like partition (by Erwig and Hagen's algorithm).



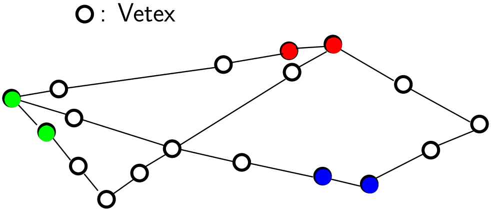
Improvement on approximate solutions by network partitioning

- Voronoi-diagram-like partition (by Erwig and Hagen's algorithm).



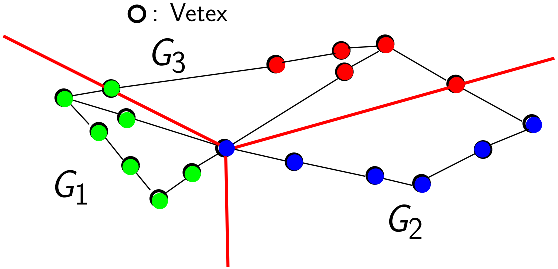
Improvement on approximate solutions by network partitioning

- Voronoi-diagram-like partition (by Erwig and Hagen's algorithm).



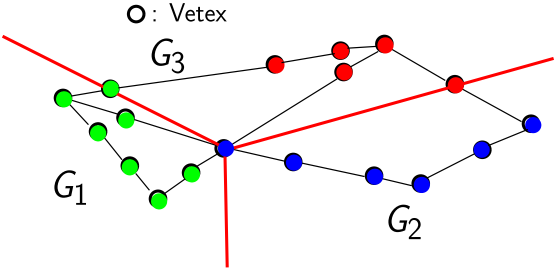
Improvement on approximate solutions by network partitioning

- Voronoi-diagram-like partition (by Erwig and Hagen's algorithm).



Improvement on approximate solutions by network partitioning

- Voronoi-diagram-like partition (by Erwig and Hagen's algorithm).



$d^-(p, G_i)$: lower bound distance from p to the boundary of G_i, computed using the landmarks.

$$d^-(s, G_i) + d^-(G_i, t) \leq d^-(s, p) + d^-(p, t), \forall p \in G_i.$$

Extensions

- Top-k MAKR query:
  - Exact methods.
  - Approximate methods.

Extensions

- Top-k MAKR query:
  - Exact methods.
  - Approximate methods.
- Multiple strings.

Extensions

- Top-k MAKR query:
  - Exact methods.
  - Approximate methods.
- Multiple strings.
- Updates.

Outline

Experiment setup

- All experiments were executed on a Linux machine with an Intel Xeon CPU at 2.13GHz and 6GB of memory.

Experiment setup

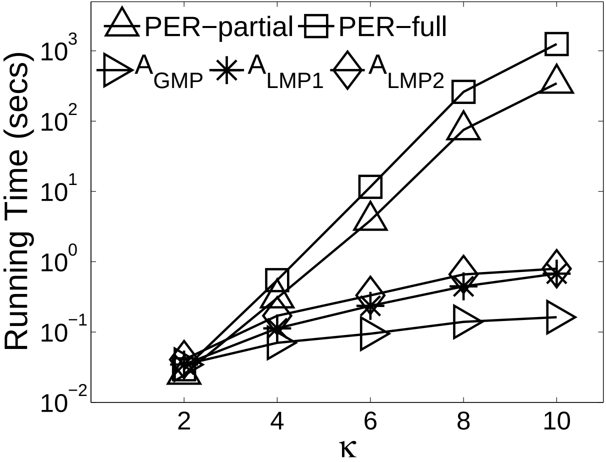
- All experiments were executed on a Linux machine with an Intel Xeon CPU at 2.13GHz and 6GB of memory.
- Data sets:
  - road networks from the *Digital Chart of the World Server*:
    City of Oldenburg (OL,6105 vertices, 7029 edges)
    California(CA,21048 vertices, 21693 edges)
    North America (NA,175813 vertices, 179179 edges)
  - building locations in OL, CA and NA from the *OpenStreetMap* project.

Experiment setup

- All experiments were executed on a Linux machine with an Intel Xeon CPU at 2.13GHz and 6GB of memory.
- Data sets:
  - road networks from the *Digital Chart of the World Server*:
    City of Oldenburg (OL,6105 vertices, 7029 edges)
    California(CA,21048 vertices, 21693 edges)
    North America (NA,175813 vertices, 179179 edges)
  - building locations in OL, CA and NA from the *OpenStreetMap* project.
- The default experimental parameters:

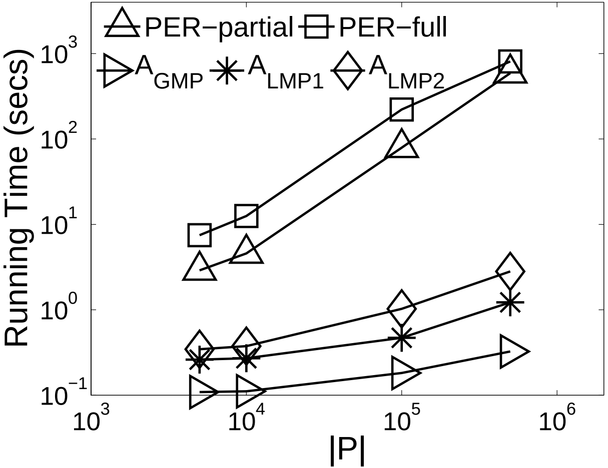
Symbol	Definition	Default Value
$\|P\|$	number of points for exact solution	$10,000$
$\|P\|$	number of points for approximate solution	$1,000,000$
κ	number of query strings	6
τ	edit distance threshold	2
	road network	CA

Query time:



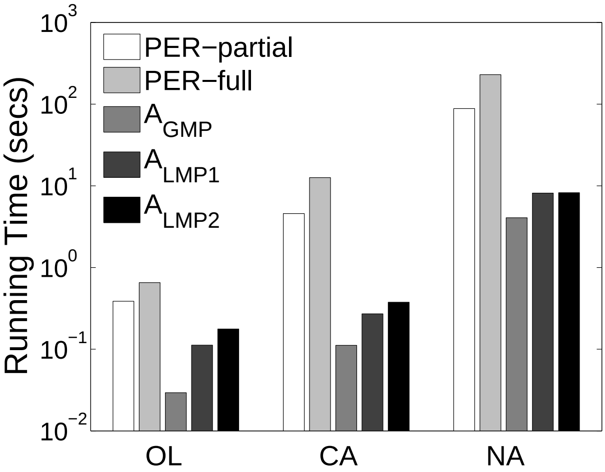
$|P| = 10,000$

Query time:



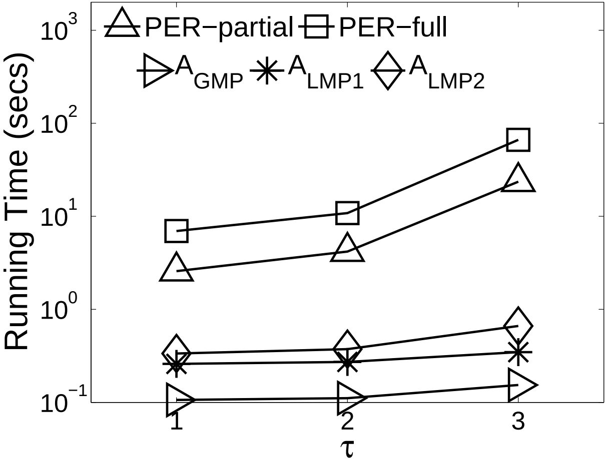
$|P| = 10,000$

Query time:



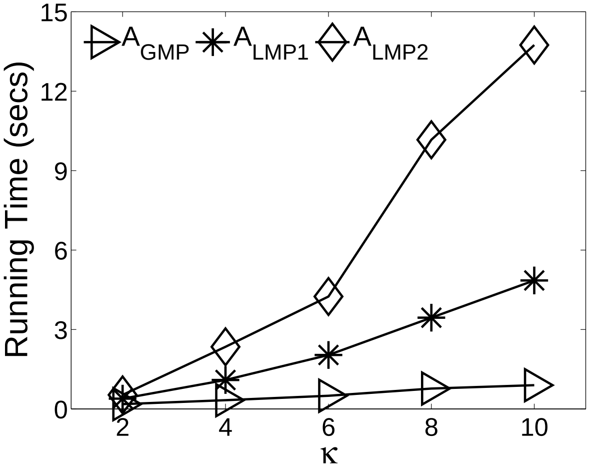
$|P| = 10,000$

Query time:



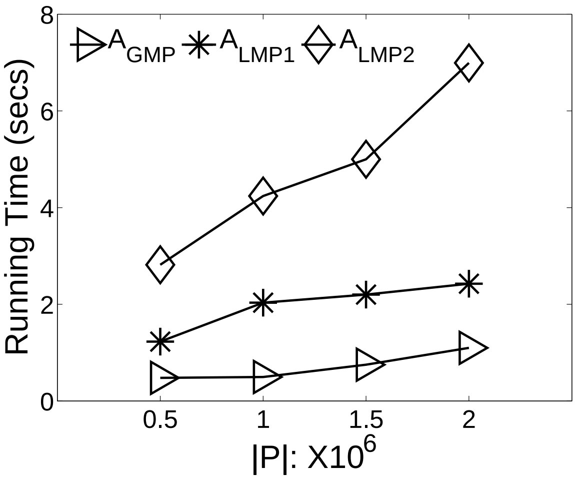
$|P| = 10,000$

Scalability of approximate solutions:



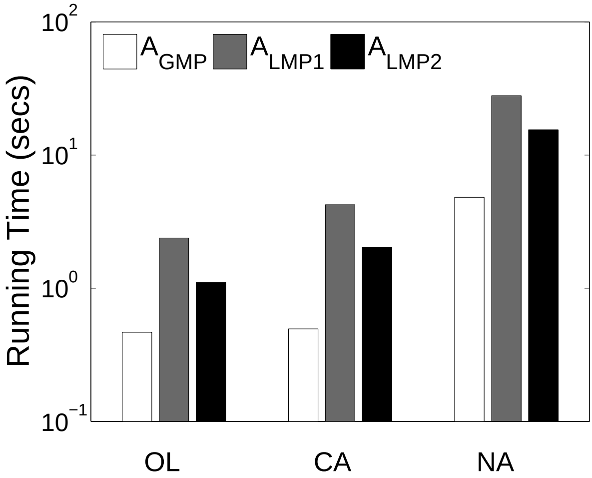
$|P| = 1,000,000$

Scalability of approximate solutions:



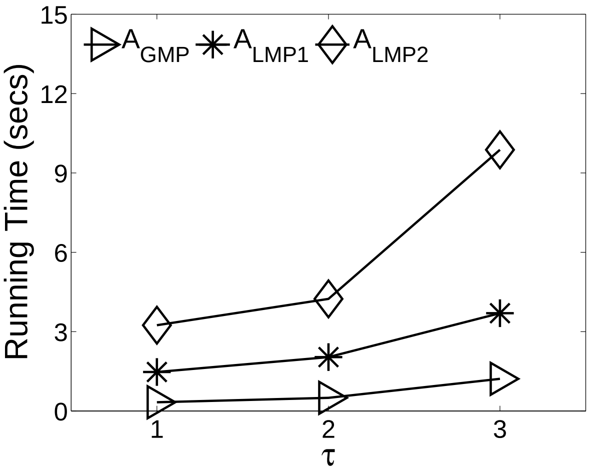
$|P| = 1,000,000$

Scalability of approximate solutions:



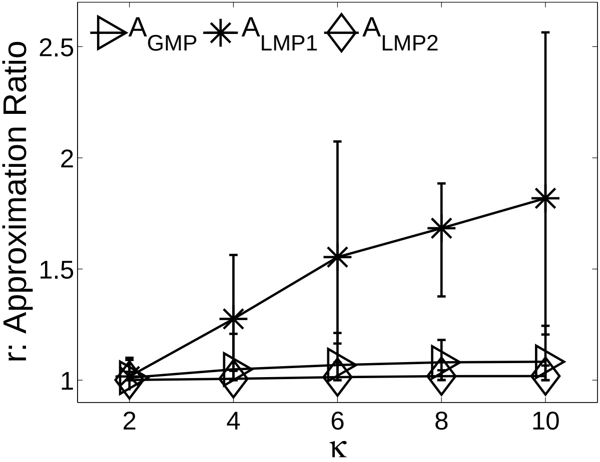
$|P| = 1,000,000$

Scalability of approximate solutions:

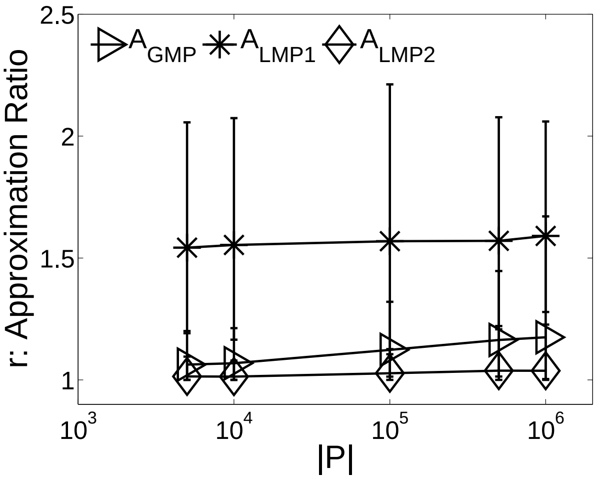


$|P| = 1,000,000$

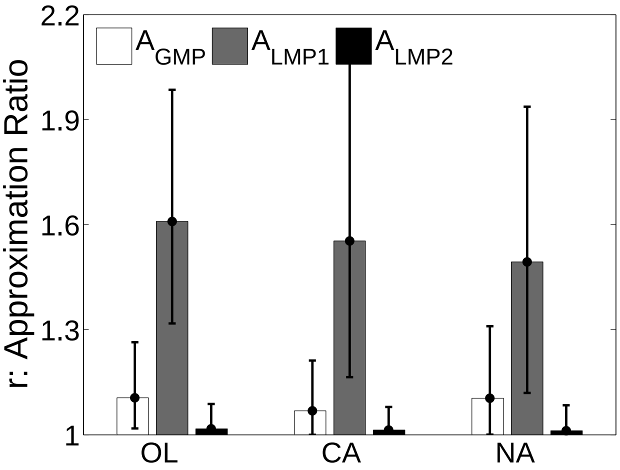
Approximation quality:



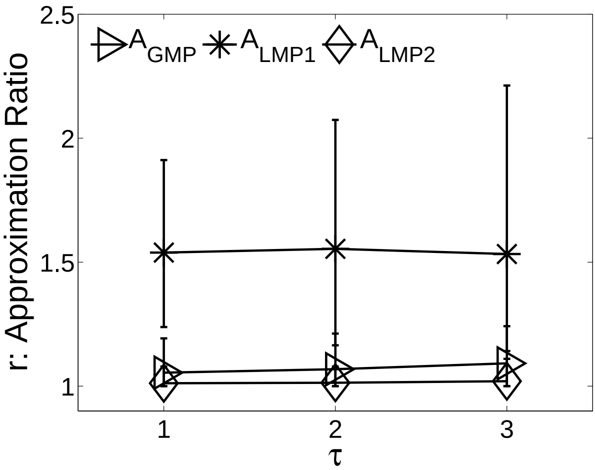
Approximation quality:



Approximation quality:



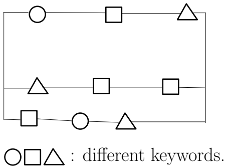
Approximation quality:



Outline

Related work

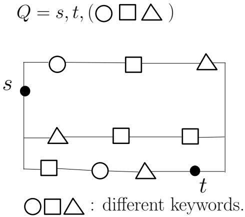
- The optimal sequenced route (OSR) query [sks07].



- [sks07]: The Optimal Sequenced Route Query. In VLDBJ, 2007.

Related work

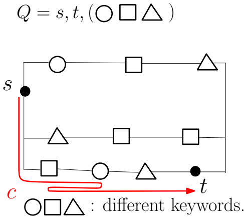
- The optimal sequenced route (OSR) query [sks07].



- [sks07]: The Optimal Sequenced Route Query. In VLDBJ, 2007.

Related work

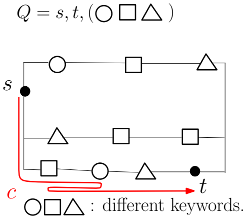
- The optimal sequenced route (OSR) query [sks07].



- [sks07]: The Optimal Sequenced Route Query. In VLDBJ, 2007.

Related work

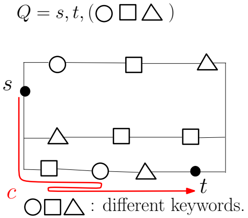
- The optimal sequenced route (OSR) query [sks07].
- Exact keyword query and only handles the query keywords sequentially.



- [sks07]: The Optimal Sequenced Route Query. In VLDBJ, 2007.

Related work

- The optimal sequenced route (OSR) query [sks07].
- Exact keyword query and only handles the query keywords sequentially.
- In MAKR queries, "categories" are dynamically decided only at the query time.



- [sks07]: The Optimal Sequenced Route Query. In VLDBJ, 2007.

Thank You

Q and A