Optimal Location Queries in Road Network Databases

Xiaokui Xiao\textsuperscript{1}, Bin Yao\textsuperscript{2}, Feifei Li\textsuperscript{2}

\textsuperscript{1}School of Computer Engineering
Nanyang Technological University, Singapore

\textsuperscript{2}Department of Computer Science
Florida State University, USA

April 8, 2011
An optimal location (OL) query:
An optimal location (OL) query:

\[ f_3 \]

- Facilities (F)

\[ f_1 \]

\[ f_2 \]
An optimal location (OL) query:

- Facilities (F)
- Clients (C)
An optimal location (OL) query:

- Facilities (F)
- Clients (C)
- Candidates (P)

Cabello et al. [1] and Wong et al. [2] deal with competitive location queries in the $L_2$ space. Du et al. [3] and Zhang et al. [4] investigate competitive and MinSum location queries in the $L_1$ space, respectively.

An optimal location (OL) query:

- Facilities (F)
- Clients (C)
- Candidates (P)
Introduction and Motivation

- An optimal location (OL) query:

\[ a(c_3) = d(c_3, f_2) \]

\[ \begin{align*}
&c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9 \\
f_1, f_2, f_3 \\
v_1, v_2, v_3, v_4, v_5, v_6 \\
\end{align*} \]

- Facilities (F)
- Clients (C)
- Candidates (P)

Cabello et al. [1] and Wong et al. [2] deal with competitive location queries in the \( L^2 \) space. Du et al. [3] and Zhang et al. [4] investigate competitive and MinSum location queries in the \( L^1 \) space, respectively.

An optimal location (OL) query:

- Facilities (F)
- Clients (C)
- Candidates (P)

Cabello et al. [1] and Wong et al. [2] deal with competitive location queries in the $L_2$ space. Du et al. [3] and Zhang et al. [4] investigate competitive and MinSum location queries in the $L_1$ space, respectively.

An optimal location (OL) query:

- Facilities (F)
- Clients (C)
- Candidates (P)

\[ a(c_3) = d(c_3, f_2) \]
An optimal location (OL) query:

Cabello et al. [1] and Wong et al. [2] deal with competitive location queries in the $L_2$ space.


An optimal location (OL) query:

Cabello et al. [1] and Wong et al. [2] deal with competitive location queries in the $L_2$ space.

Du et al. [3] and Zhang et al. [4] investigate competitive and MinSum location queries in the $L_1$ space, respectively.


Competitive location query:

\[ p = \arg\max_{p \in P} |C_p|, \]

where \( C_p \) is the set of clients attracted by \( p \).
Problem Formulation

Competitive location query:

\[ p = \arg\max_{p \in P} |C_p|, \]

where \( C_p \) is the set of clients attracted by \( p \).
Competitive location query:

\[ p = \arg\max_{p \in P} |C_p|, \]

where \( C_p \) is the set of clients attracted by \( p \).

Example 1: Given existing supermarkets \( F \) (residential locations \( C \)) in a city, Julie wants to open a new supermarket that can attract as many customers as possible.
MinSum location query:

\[ p = \arg\min_{p \in P} \sum_{c \in C} a(c). \]
MinSum location query:

\[ p = \arg\min_{p \in P} \sum_{c \in C} a(c). \]

*Example 2:* John owns a set \( F \) of pizza shops that deliver to a set \( C \) of places in a city. He looks for a location to add another pizza shop to minimize the average distance from the place in \( C \) to their respective nearest shops.
MinMax location query:

\[ p = \arg\min_{p \in P} \left( \max_{c \in C} a(c) \right). \]
MinMax location query:

\[ p = \arg\min_{p \in P} \left( \max_{c \in C} a(c) \right). \]

*Example 3:* Given the set \( F(C) \) of existing fire stations (buildings) in a city, the government may seek a candidate location that minimizes the maximum distance from any building to its nearest fire station.
Construct the road intervals.
Traverse the candidate road intervals in a certain order.
Identify the local optimal locations.
Return the global optimal locations.
Construct the road intervals.

Xiaokui Xiao, Bin Yao, Feifei Li
Optimal Location Queries in Road Network Databases
Construct the road intervals.

Traverse the candidate road intervals in a certain order.
Construct the road intervals.

Traverse the candidate road intervals in a certain order.

Identify the local optimal locations.
Solution Overview

- Construct the road intervals.
- Traverse the candidate road intervals in a certain order.
- Identify the local optimal locations.
- Return the global optimal locations.
Local optimal locations: competitive location queries

Xiaokui Xiao, Bin Yao, Feifei Li

Optimal Location Queries in Road Network Databases
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$. 
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.

$$a(c) - d(c, v_l) = 1$$

$d(c, v_l) = 4$

$a(c) = 5$
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.

\[ \mathcal{A}(v_l) = \{ \langle c_1, 4 \rangle, \langle c_3, 1 \rangle, \langle c_4, 3 \rangle \} \]
\[ \mathcal{A}(v_r) = \{ \langle c_2, 3 \rangle, \langle c_3, 2 \rangle, \langle c_4, 4 \rangle \} \]

$e$ (length = 5)

$a(c_i) = 5$
Local optimal locations: competitive location queries

Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.

- $\mathcal{A}(v_L)$:
  - $\langle c_1, 4 \rangle$
  - $\langle c_3, 1 \rangle$
  - $\langle c_4, 3 \rangle$

- $\mathcal{A}(v_R)$:
  - $\langle c_2, 3 \rangle$
  - $\langle c_3, 2 \rangle$
  - $\langle c_4, 4 \rangle$

$e$ (length = 5)

$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad R$

$a(c_i) = 5$
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$. 

\[
\begin{align*}
\mathcal{A}(v_l) & \quad \mathcal{A}(v_r) \\
\langle c_1, 4 \rangle & \quad \langle c_2, 3 \rangle \\
\langle c_3, 1 \rangle & \quad \langle c_3, 2 \rangle \\
\langle c_4, 3 \rangle & \quad \langle c_4, 4 \rangle \\
\end{align*}
\]

$e$ (length = 5) $a(c_i) = 5$
Lemma

A client \( c \) is attracted by a point \( p \) on an edge \( e \in E_c \), iff there exists an entry \( \langle c, d(c, v) \rangle \) in the attraction set of an endpoint \( v \) of \( e \), such that \( d(c, v) + d(v, p) \leq a(c) \).
**Lemma**

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.
Lemma

A client $c$ is attracted by a point $p$ on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint $v$ of $e$, such that $d(c, v) + d(v, p) \leq a(c)$.

\[ \mathcal{A}(v_l) \]

\[
\begin{align*}
&\langle c_1, 4 \rangle \\
&\langle c_3, 1 \rangle \\
&\langle c_4, 3 \rangle
\end{align*}
\]

\[ \mathcal{A}(v_r) \]

\[
\begin{align*}
&\langle c_2, 3 \rangle \\
&\langle c_3, 2 \rangle \\
&\langle c_4, 4 \rangle
\end{align*}
\]

$v_l \quad e(\text{length} = 5) \quad v_r$

$a(c_i) = 5$
Computing attractor distances and attraction sets

Computing attractor distances: Erwig and Hagen’s algorithm.
Computing attraction sets: The Blossom and OTF algorithms.

The Blossom algorithm, Time: $O(n^2 \log n)$, space: $O(n^2)$.

The OTF algorithm

Xiaokui Xiao, Bin Yao, Feifei Li

Optimal Location Queries in Road Network Databases
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.

The Blossom algorithm, Time: $O(n^2 \log n)$, space: $O(n^2)$.

The OTF algorithm.

Xiaokui Xiao, Bin Yao, Feifei Li

Optimal Location Queries in Road Network Databases
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.

\[ v_1, v_5, v_4 \]
\[ f_1, f_3, f_2 \]
\[ c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9 \]
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.

- Computing attraction sets: The Blossom and OTF algorithms.
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.

- Computing attraction sets: The Blossom and OTF algorithms.
  - The Blossom algorithm, Time: $O(n^2 \log n)$, space: $O(n^2)$. 

Xiaokui Xiao, Bin Yao, Feifei Li 
Optimal Location Queries in Road Network Databases
Computing attractor distances and attraction sets

- Computing attractor distances: Erwig and Hagen’s algorithm.

- Computing attraction sets: The Blossom and OTF algorithms.
  - The Blossom algorithm, Time: $O(n^2 \log n)$, space: $O(n^2)$.
  - The OTF algorithm
The OTF algorithm: construct the $A(v)$ on the fly

A straightforward solution: apply Dijkstra's algorithm to scan all vertices starting at $v$. If $d(v, c) < a(c)$, add $c$ into $A(v)$. 

Lemma
Given two vertices $v$ and $v'$ in $G$, such that $d(v, v')$ is larger than the distance from $v'$ to its nearest facility $f'$. Then, $\forall c \in A(v)$, the shortest path from $v$ to $c$ must not go through $v'$. 

Xiaokui Xiao, Bin Yao, Feifei Li
Optimal Location Queries in Road Network Databases
The OTF algorithm: construct the $A(v)$ on the fly

- A straightforward solution:
  - apply Dijkstra’s algorithm to scan all vertices starting at $v$.
  - If $d(v, c) < a(c)$, add $c$ into $A(v)$.

Lemma
Given two vertices $v$ and $v'$ in $G$, such that $d(v, v')$ is larger than the distance from $v'$ to its nearest facility $f'$. Then, $\forall c \in A(v)$, the shortest path from $v$ to $c$ must not go through $v'$. 
The OTF algorithm: construct the \( A(v) \) on the fly

- A straightforward solution:
  - apply Dijkstra’s algorithm to scan all vertices starting at \( v \).
  - If \( d(v, c) < a(c) \), add \( c \) into \( A(v) \).

Lemma

Given two vertices \( v \) and \( v' \) in \( G \), such that \( d(v, v') \) is larger than the distance from \( v' \) to its nearest facility \( f' \). Then, \( \forall c \in A(v), \) the shortest path from \( v \) to \( c \) must not go through \( v' \).
The OTF algorithm: construct the $A(v)$ on the fly

- A straightforward solution:
  - apply Dijkstra’s algorithm to scan all vertices starting at $v$.
  - If $d(v, c) < a(c)$, add $c$ into $A(v)$.

**Lemma**

*Given two vertices $v$ and $v'$ in $G$, such that $d(v, v')$ is larger than the distance from $v'$ to its nearest facility $f'$. Then, $\forall c \in A(v)$, the shortest path from $v$ to $c$ must not go through $v'$.*
The OTF algorithm: construct the $A(v)$ on the fly

- A straightforward solution:
  - apply Dijkstra’s algorithm to scan all vertices starting at $v$.
  - If $d(v, c) < a(c)$, add $c$ into $A(v)$.

**Lemma**

Given two vertices $v$ and $v'$ in $G$, such that $d(v, v')$ is larger than the distance from $v'$ to its nearest facility $f'$. Then, $\forall c \in A(v)$, the shortest path from $v$ to $c$ must not go through $v'$. 

Xiaokui Xiao, Bin Yao, Feifei Li
Optimal Location Queries in Road Network Databases
The OTF algorithm: construct the $\mathcal{A}(v)$ on the fly

- A straightforward solution:
  - apply Dijkstra’s algorithm to scan all vertices starting at $v$.
  - If $d(v, c) < a(c)$, add $c$ into $\mathcal{A}(v)$.

**Lemma**

Given two vertices $v$ and $v'$ in $G$, such that $d(v, v')$ is larger than the distance from $v'$ to its nearest facility $f'$. Then, $\forall c \in \mathcal{A}(v)$, the shortest path from $v$ to $c$ must not go through $v'$.
The OTF algorithm: construct the $\mathcal{A}(v)$ on the fly

Time: $O(n^2 \log n)$, space: $O(n)$
The OTF algorithm: construct the $\mathcal{A}(v)$ on the fly

Time: $O(n^2 \log n)$, space: $O(n)$
The OTF algorithm: construct the $A(v)$ on the fly

$$d(v_2, c_2) \leq a(c_2)$$
The OTF algorithm: construct the $A(v)$ on the fly

\[ d(v_2, c_2) \leq a(c_2) \]

$A(v_2)$

\[ < c_2, 1 > \]
The OTF algorithm: construct the $A(v)$ on the fly

\[ d(v_2, v_1) > d(v_1, f_1) \]
The OTF algorithm: construct the $\mathcal{A}(v)$ on the fly

Time: $O(n^2 \log n)$, space: $O(n)$
The OTF algorithm: construct the $A(v)$ on the fly

$A(v_2) = \langle c_2, 1 \rangle, \langle c_3, 2 \rangle, \langle c_8, 3 \rangle$

Time: $O(n^2 \log n)$, space: $O(n)$
The OTF algorithm: construct the $\mathcal{A}(v)$ on the fly

$\mathcal{A}(v_2)$

$< c_2, 1 >$

$< c_3, 2 >$

$< c_8, 3 >$
The OTF algorithm: construct the $A(v)$ on the fly

Time: $O(n^2 \log n)$, space: $O(n)$
Enumerating the local optima will incur significant overhead when $E_c$ is large.
Fine-grained Partitioning (FGP)

- Enumerating the local optima will incur significant overhead when $E_c$ is large.
For each type of OL queries, we examine two approaches:
- the basic approach
- the Fine-grained partitioning (FGP) approach
For each type of OL queries, we examine two approaches:

- the basic approach
- the Fine-grained partitioning (FGP) approach

For each approach, we test two techniques for deriving attraction sets: the Blossom and OTF.
For each type of OL queries, we examine two approaches:
- the basic approach
- the Fine-grained partitioning (FGP) approach

For each approach, we test two techniques for deriving attraction sets: the Blossom and OTF.

C++, Linux, Intel Xeon 2GHz CPU and 4GB memory
For each type of OL queries, we examine two approaches:
- the basic approach
- the Fine-grained partitioning (FGP) approach
For each approach, we test two techniques for deriving attraction sets: the Blossom and OTF.

- C++, Linux, Intel Xeon 2GHz CPU and 4GB memory
- Data sets
  - San Francisco(SF) and California(CA) road networks from the Digital Chart of the World Server.
  - building locations in SF and CA from the OpenStreetMap project.
For each type of OL queries, we examine two approaches:
- the basic approach
- the Fine-grained partitioning (FGP) approach

For each approach, we test two techniques for deriving attraction sets: the Blossom and OTF.

C++, Linux, Intel Xeon 2GHz CPU and 4GB memory

Data sets
- San Francisco (SF) and California (CA) road networks from the Digital Chart of the World Server.
- building locations in SF and CA from the OpenStreetMap project.

Default settings.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>$\tau$</td>
<td>the percentage of candidate edges</td>
<td>100%</td>
</tr>
</tbody>
</table>
Vary $|F|$ (CLQ on SF)

![Graph showing memory consumption vs number of facilities]
Vary $|F|$ (CLQ on SF)

The graph shows the running time (in seconds) on the y-axis and the number of facilities (×10^3) on the x-axis. The graph compares three algorithms:

- **Blossom** (dashed blue line) starts with lower running time but runs out of memory at higher facility counts.
- **OTF** (solid black line) maintains a lower running time throughout.
- **Basic** (blue asterisks) starts with the highest running time, followed by **FGP** (blue squares), which shows the least variation in running time.

The x-axis denotes the number of facilities, with markers at 0, 0.5, 1, 2, 3, and 4, scaled by 10^3.
Vary $|C|$ (CLQ on SF)

Number of clients \times 10^5

Memory consumption (MB)

- Blossom
- OTF
- Basic
- FGP

Out of memory

Xiaokui Xiao, Bin Yao, Feifei Li

Optimal Location Queries in Road Network Databases
Vary $|C|$ (CLQ on SF)

- **Blossom**
- **OTF**

- **Basic**
- **FGP**

- *out of memory*

- **Xiaokui Xiao**, Bin Yao, Feifei Li

**Optimal Location Queries in Road Network Databases**
Define three variants of OL queries on the road networks.
Define three variants of OL queries on the road networks.

Introduce a unified framework that addresses all three query types efficiently.
Conclusion

- Define three variants of OL queries on the road networks.

- Introduce a unified framework that addresses all three query types efficiently.

Future work

- the incremental monitoring of the optimal locations when the facility or client sets have been updated.
- the optimal location queries for moving objects in road networks.
Thank You

Q and A