Optimal Location Queries in Road Network Databases

Xiaokui Xiao[1], Bin Yao[2], Feifei Li[2]

[1]School of Computer Engineering
Nanyang Technological University, Singapore

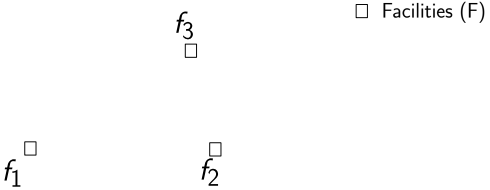
[2]Department of Computer Science
Florida State University, USA

April 8, 2011

Introduction and Motivation

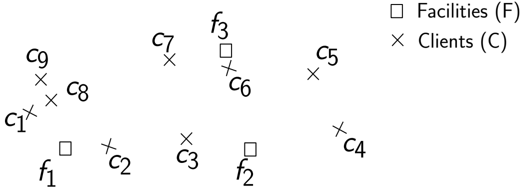
- An optimal location (OL) query:

- An optimal location (OL) query:



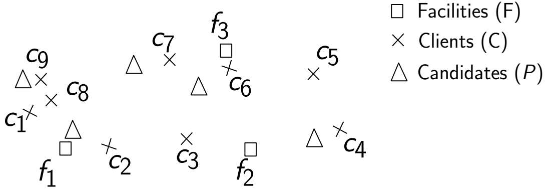
Introduction and Motivation

- An optimal location (OL) query:

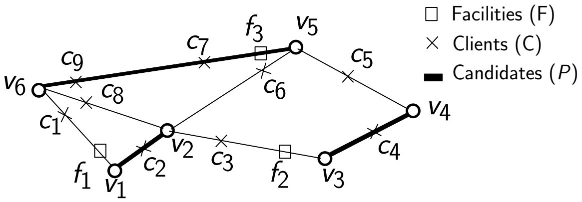


Introduction and Motivation

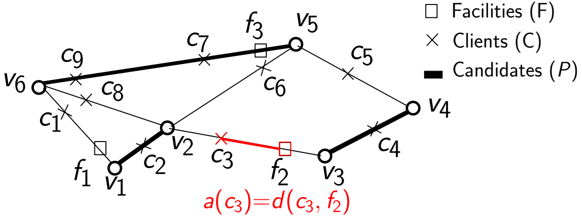
- An optimal location (OL) query:



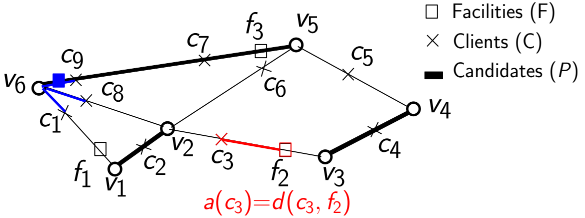
- An optimal location (OL) query:



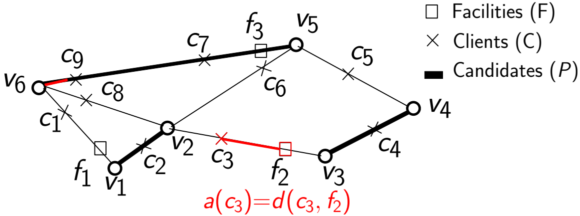
- An optimal location (OL) query:



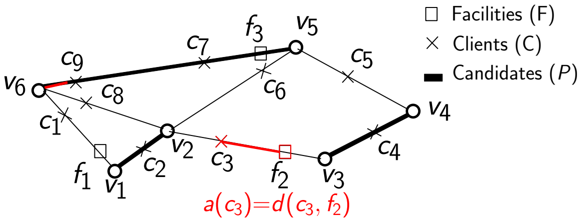
- An optimal location (OL) query:



- An optimal location (OL) query:



- An optimal location (OL) query:

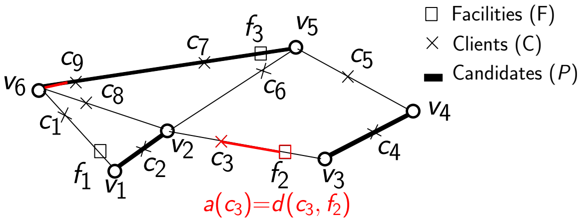


- Cabello et al. [1] and Wong et al. [2] deal with competitive location queries in the L_2 space

- [1] S. Cabello, et al. Reverse facility location problems. In CCCG, 2005
- [2] R. Wong, et al. Efficient method for maximizing bichromatic reverse nearest neighbor. In PVLDB, 2009

Introduction and Motivation

- 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.

- [1] S. Cabello, et al. Reverse facility location problems. In CCCG, 2005
- [2] R. Wong, et al. Efficient method for maximizing bichromatic reverse nearest neighbor. In PVLDB, 2009
- [3] Y. Du, et al. The optimal-location query. In SSTD, 2005
- [4] D. Zhang, et al. Progressive computation of the min-dist optimal-location query. In VLDB, 2006

Problem Formulation

Competitive location query:

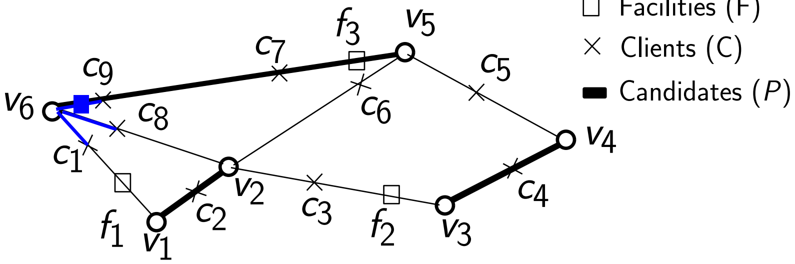
$$p = \operatorname*{argmax}_{p \in P} |C_p|,$$

where C_p is the set of clients attracted by p.

Competitive location query:

$$p = \operatorname*{argmax}_{p \in P} |C_p|,$$

where C_p is the set of clients attracted by p.



Problem Formulation

Competitive location query:

$$p = \operatorname*{argmax}_{p \in P} |C_p|,$$

where C_p is the set of clients attracted by p.

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

Problem Formulation

MinSum location query:

$$p = \operatorname*{argmin}_{p \in P} \sum_{c \in C} a(c).$$

Problem Formulation

MinSum location query:

$$p = \underset{p \in P}{\operatorname{argmin}} \sum_{c \in C} a(c).$$

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.

Problem Formulation

MinMax location query:

$$p = \operatorname*{argmin}_{p \in P} \left(\max_{c \in C} a(c) \right).$$

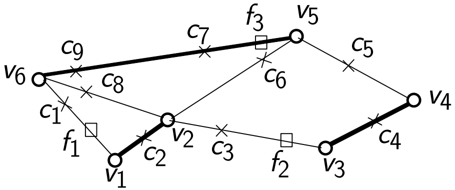
Problem Formulation

MinMax location query:

$$p = \operatorname*{argmin}_{p \in P} \left(\max_{c \in C} a(c) \right).$$

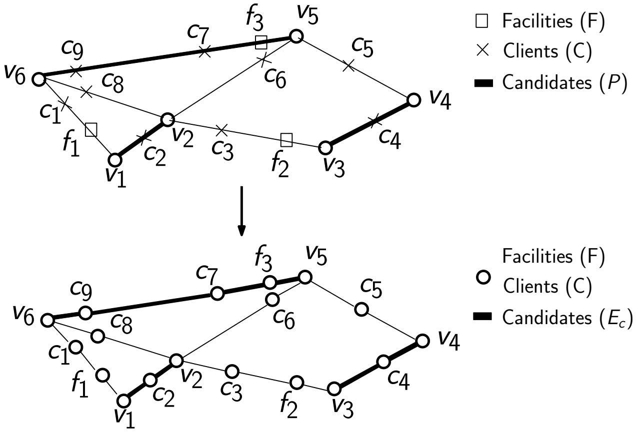
Example 3: Given the set F (C) of existing fire stations (buildings) in a city, the government may seek a candidate location that minimizes the maximum distance from any building to its nearest fire station.

Solution Overview



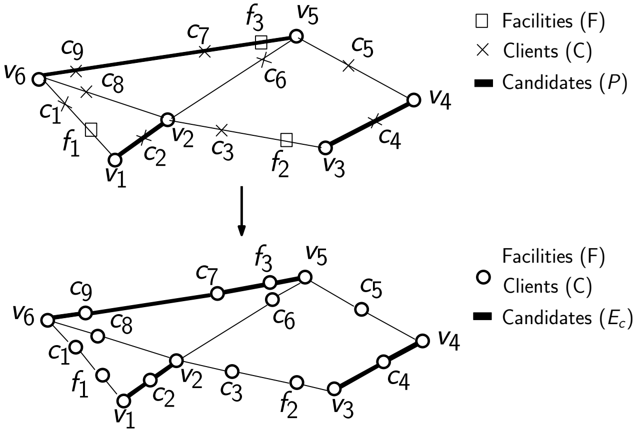
□ Facilities (F)
× Clients (C)
▬ Candidates (P)

Solution Overview



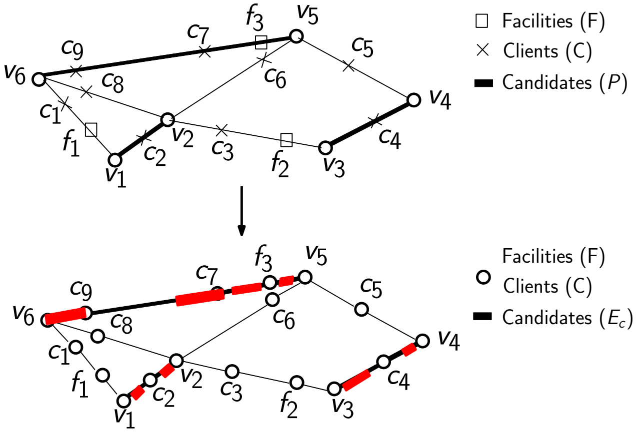
- Construct the road intervals.

Solution Overview



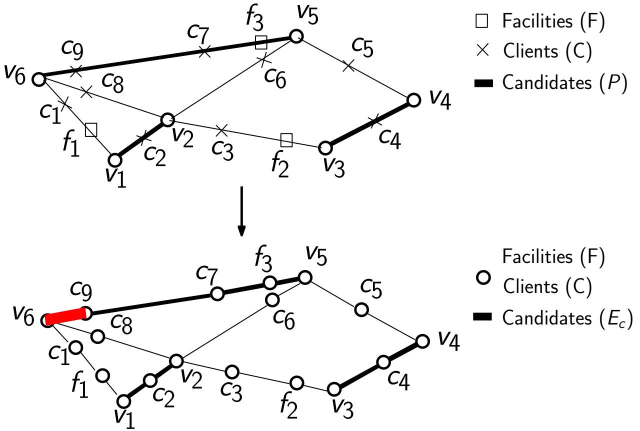
- Construct the road intervals.
- Traverse the candidate road intervals in a certain order.

Solution Overview



- 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

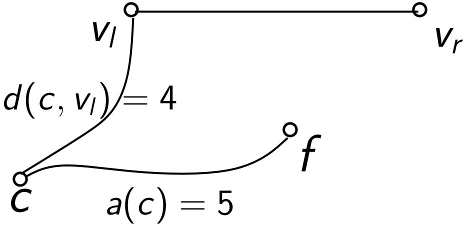
Lemma

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 $d(c, v) + d(v, p) \leq a(c)$.

Local optimal locations: competitive location queries

Lemma

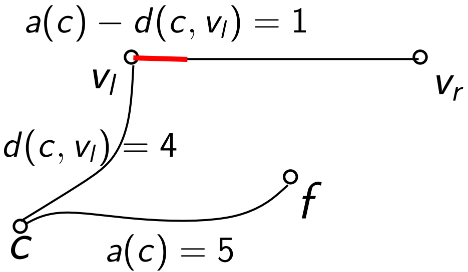
A client c is attracted by a point p on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v)\rangle$ in the attraction set of an endpoint v of e, such that $d(c, v) + d(v, p) \leq a(c)$.



Local optimal locations: competitive location queries

Lemma

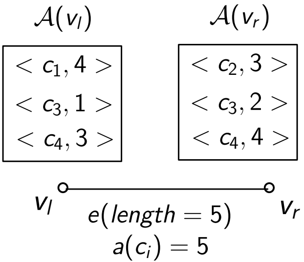
A client c is attracted by a point p on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c,v)\rangle$ in the attraction set of an endpoint v of e, such that $d(c,v) + d(v,p) \leq a(c)$.



Local optimal locations: competitive location queries

Lemma

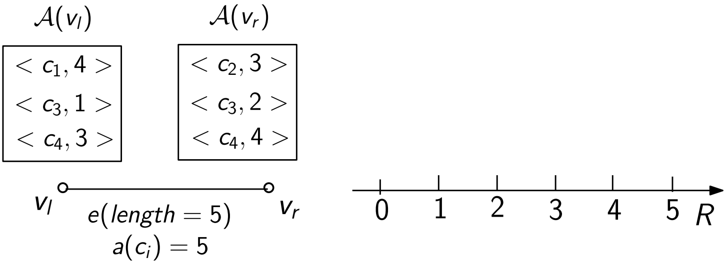
A client c is attracted by a point p on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v) \rangle$ in the attraction set of an endpoint v of e, such that $d(c, v) + d(v, p) \leq a(c)$.



Local optimal locations: competitive location queries

Lemma

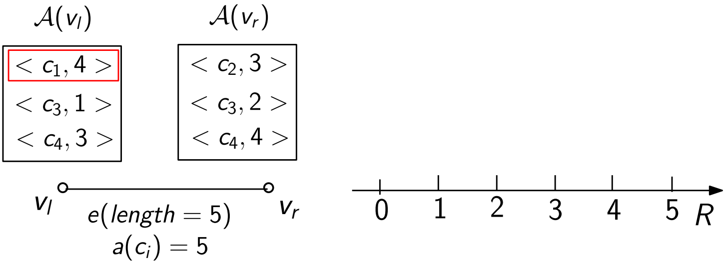
A client c is attracted by a point p on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v)\rangle$ in the attraction set of an endpoint v of e, such that $d(c, v) + d(v, p) \leq a(c)$.



Local optimal locations: competitive location queries

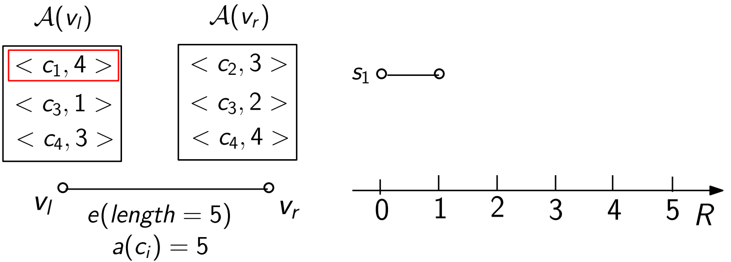
Lemma

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 $d(c, v) + d(v, p) \leq a(c)$.



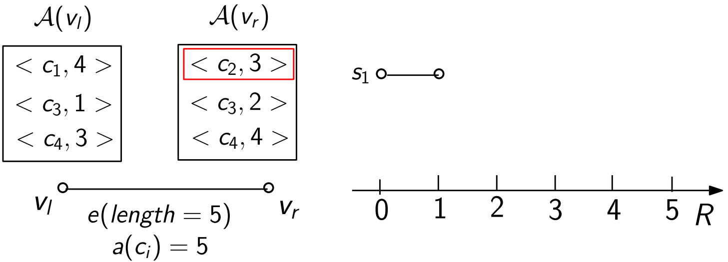
Lemma

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 $d(c,v) + d(v,p) \le a(c)$.



Lemma

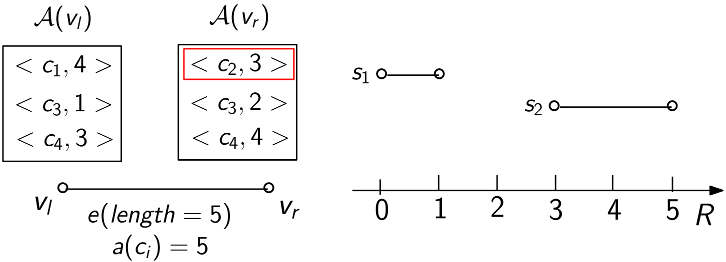
A client c is attracted by a point p on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c,v)\rangle$ in the attraction set of an endpoint v of e, such that $d(c,v) + d(v,p) \leq a(c)$.



Local optimal locations: competitive location queries

Lemma

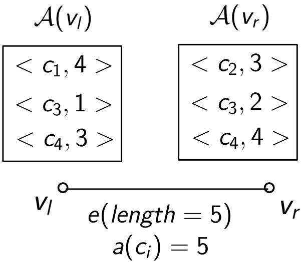
A client c is attracted by a point p on an edge $e \in E_c$, iff there exists an entry $\langle c, d(c, v)\rangle$ in the attraction set of an endpoint v of e, such that $d(c, v) + d(v, p) \leq a(c)$.

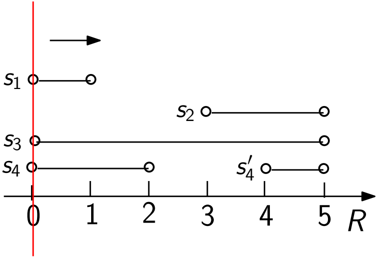


Local optimal locations: competitive location queries

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 $d(c,v) + d(v,p) \leq a(c)$.

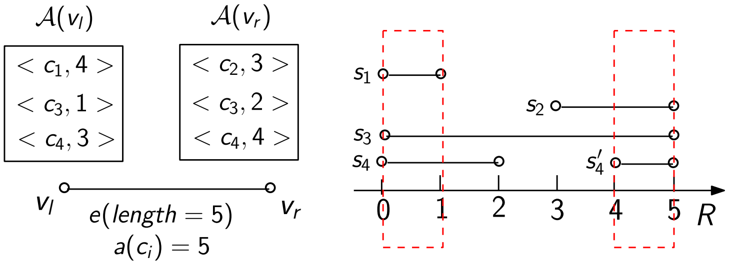




Local optimal locations: competitive location queries

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 $d(c, v) + d(v, p) \leq a(c)$.

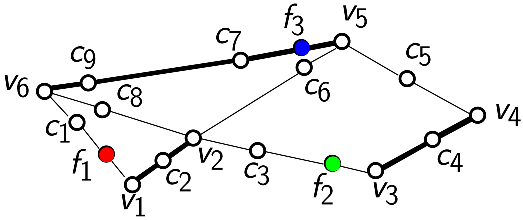


Computing attractor distances and attraction sets

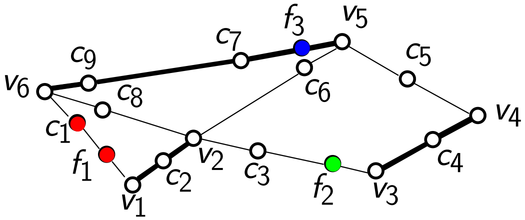
Computing attractor distances and attraction sets

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

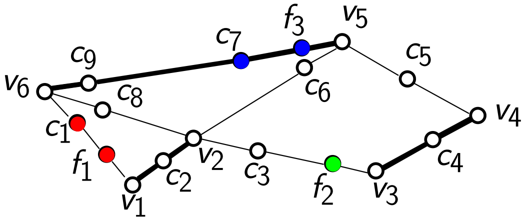
- Computing attractor distances: Erwig and Hagen's algorithm.



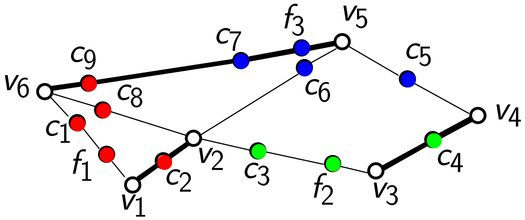
- Computing attractor distances: Erwig and Hagen's algorithm.



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

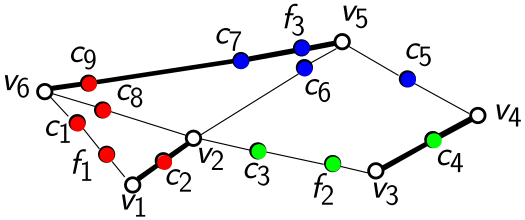


- Computing attractor distances: Erwig and Hagen's algorithm.



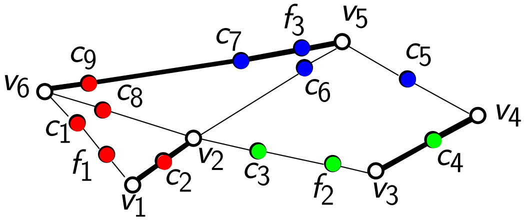
Computing attractor distances and attraction sets

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



- Computing attraction sets: The Blossom and OTF algorithms.

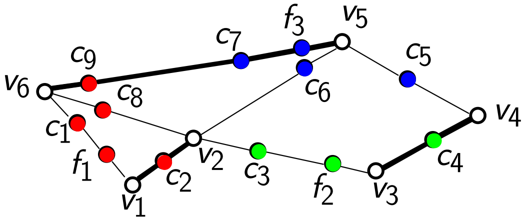
- Computing attractor distances: Erwig and Hagen's algorithm.



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

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 $\mathcal{A}(v)$ *on the fly*

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 $\mathcal{A}(v)$.

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

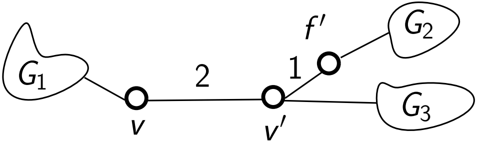
Given two vertices v and v' in G, such that $d(v, v')$ is larger than the distance from v' to its nearest facility f'. Then, $\forall c \in \mathcal{A}(v)$, the shortest path from v to c must not go through v'.

The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*

- 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'.

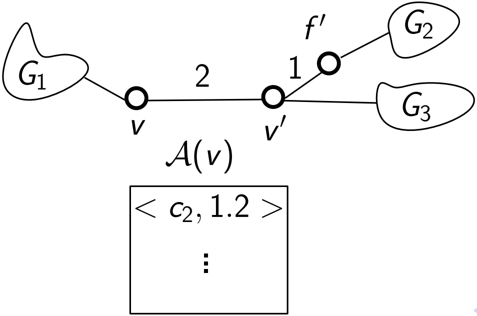


The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*

- 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

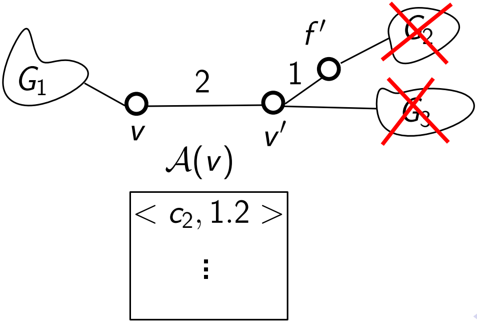
Given two vertices v and v' in G, such that $d(v, v')$ is larger than the distance from v' to its nearest facility f'. Then, $\forall c \in \mathcal{A}(v)$, the shortest path from v to c must not go through v'.



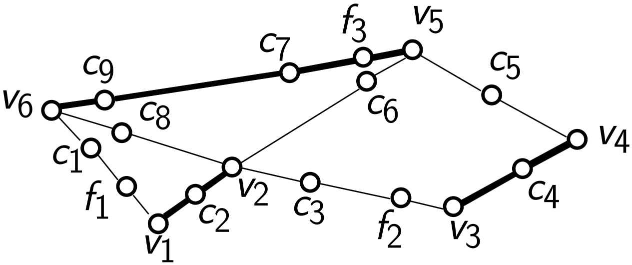
- A straightforward solution:
  - apply Dijkstra's algorithm to scan all vertices starting at v.
  - If $d(v, c) < a(c)$, add c into $\mathcal{A}(v)$.

Lemma

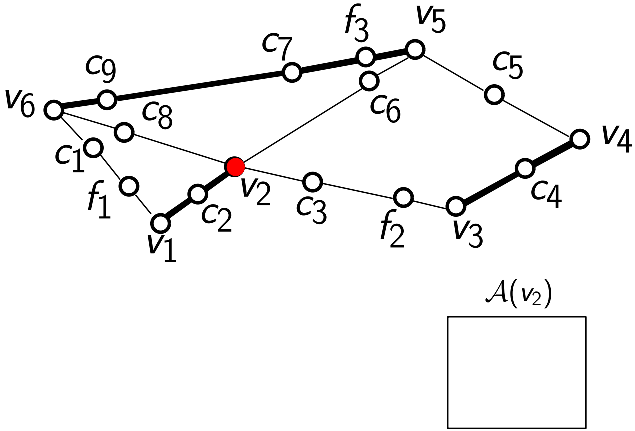
Given two vertices v and v' in G, such that $d(v, v')$ is larger than the distance from v' to its nearest facility f'. Then, $\forall c \in \mathcal{A}(v)$, the shortest path from v to c must not go through v'.



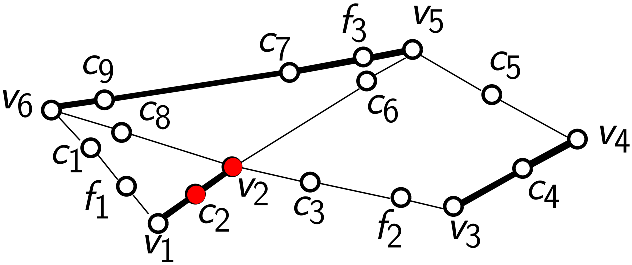
The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



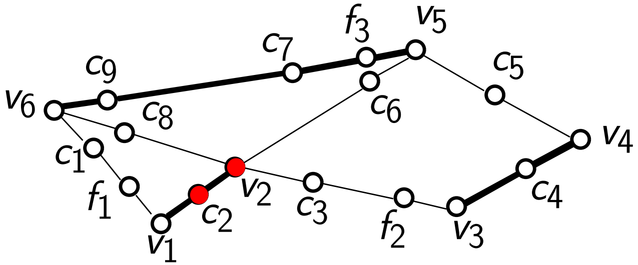
The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



$d(v_2, c_2) \leq a(c_2)$

$\mathcal{A}(v_2)$

The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*

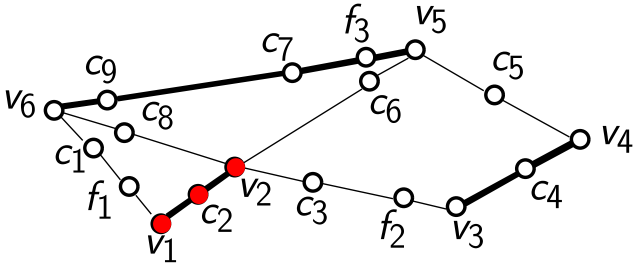


$d(v_2, c_2) \leq a(c_2)$

$\mathcal{A}(v_2)$

$< c_2, 1 >$

The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*

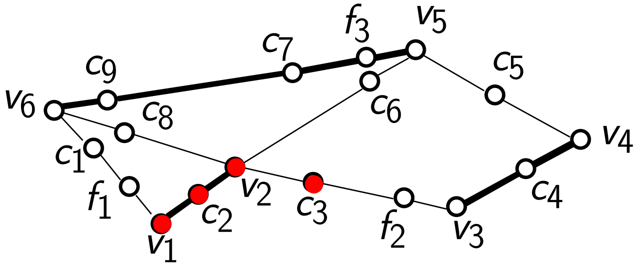


$d(v_2, v_1) > d(v_1, f_1)$

$\mathcal{A}(v_2)$

$< c_2, 1 >$

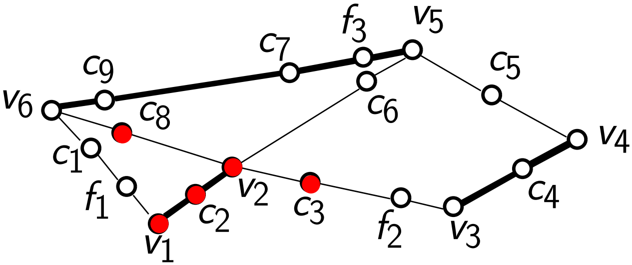
The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



$\mathcal{A}(v_2)$

$< c_2, 1 >$
$< c_3, 2 >$

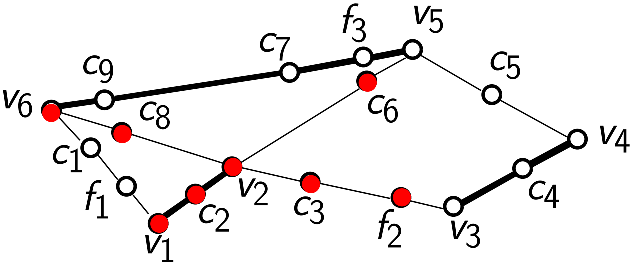
The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



$\mathcal{A}(v_2)$

$< c_2, 1 >$
$< c_3, 2 >$
$< c_8, 3 >$

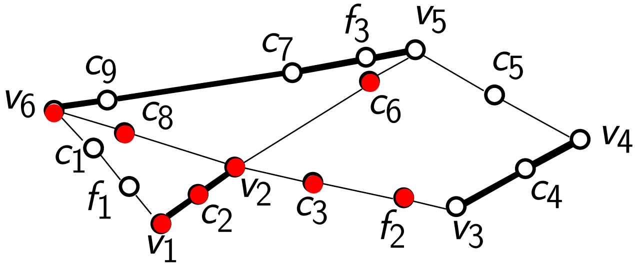
The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



$\mathcal{A}(v_2)$

$< c_2, 1 >$
$< c_3, 2 >$
$< c_8, 3 >$

The OTF algorithm: construct the $\mathcal{A}(v)$ *on the fly*



$\mathcal{A}(v_2)$

$< c_2, 1 >$
$< c_3, 2 >$
$< c_8, 3 >$

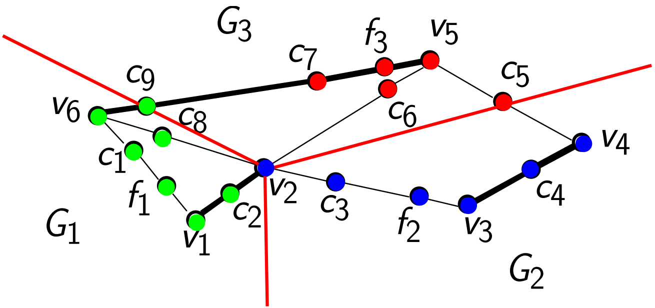
- Time: $O(n^2 \log n)$, space: $O(n)$

Fine-grained Partitioning (FGP)

- 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.



Experiment

- For each type of OL queries, we examine two approaches:
  - the basic approach
  - the Fine-grained partitioning (FGP) approach

Experiment

- 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.

Experiment

- 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

Experiment

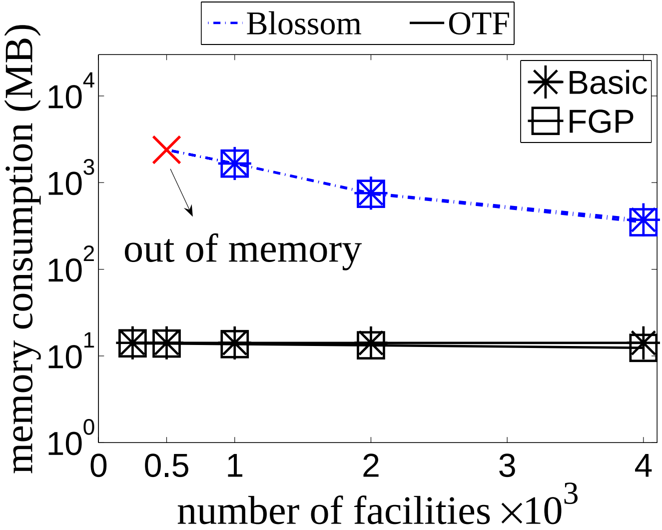
- For each type of OL queries, we examine two approaches:
  - the basic approach
  - the Fine-grained partitioning (FGP) approach
- For each approach, we test two techniques for deriving attraction sets: the Blossom and OTF.
- C++, Linux, Intel Xeon 2GHz CPU and 4GB memory
- Data sets
  - San Francisco(SF)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

Experiment

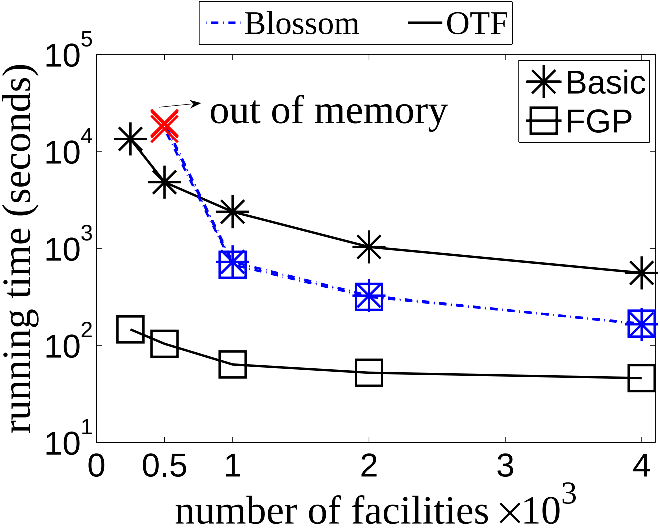
- For each type of OL queries, we examine two approaches:
  - the basic approach
  - the Fine-grained partitioning (FGP) approach
- For each approach, we test two techniques for deriving attraction sets: the Blossom and OTF.
- C++, Linux, Intel Xeon 2GHz CPU and 4GB memory
- Data sets
  - San Francisco(SF)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

Symbol	Definition	Default Value
$\lvert F\rvert$	number of facilities	1000
$\lvert C\rvert$	number of clients	$300,000$
τ	the percentage of candidate edges	100%

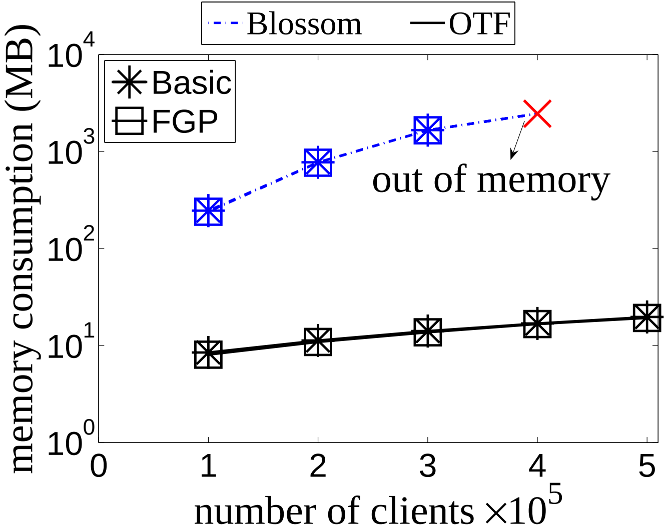
Vary $|F|$ (CLQ on SF)

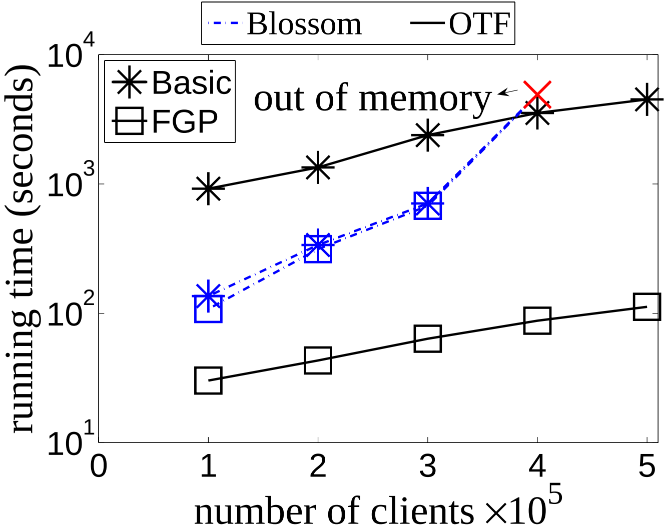


Vary $|F|$ (CLQ on SF)



Vary |C| (CLQ on SF)





Blossom
OTF
Basic
FGP
out of memory
running time (seconds)
10^4
10^3
10^2
10^1
0
1
2
3
4
5
number of clients ×10^5

Conclusion

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

Conclusion

- 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