Query Processing in Geo-Social Networks

Nikos Armenatzoglou
OUTLINE

• Geo-Social Networks (GeoSNs)

• Query Processing Framework

• Geo-Social Queries

• Geo-Social Ranking

• Real-Time Multi Criteria Graph Partitioning
  (Geo-Social Graph Partitioning)

• Conclusion & Future Work
GEO-SOCIAL NETWORKS (GeoSNs)

Social network functionality + Location-based services = Geo-Social Query

My Friends in range
CHALLENGE

Social Relations (Friendships)

Online Task
Geo-Social Query Processing

Geographical Information (current check-ins)

Large 😞
Complex 😞
Relatively Static 😊

Small 😊
Simple 😊
Dynamic 😞
INDUSTRY

Nearby Friends

“Which social event is better?”

“One of your friends is here!”

“New friends!!”

10K geo-tagged tweets/min
[http://geosocialfootprint.com/]

No white papers documenting the processing of queries.
No unanimously accepted social and spatial storage implementation.

<table>
<thead>
<tr>
<th>Application</th>
<th>Storage System</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/50" alt="Facebook" /></td>
<td><img src="https://via.placeholder.com/50" alt="MongoDB" /></td>
<td>Adjacency lists in a Distributed Memory Hash Table</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/50" alt="Foursquare" /></td>
<td><img src="https://via.placeholder.com/50" alt="MongoDB" /></td>
<td>Adjacency lists in a Document-oriented database</td>
</tr>
<tr>
<td>Spatial</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/50" alt="Twitter" /></td>
<td><img src="https://via.placeholder.com/50" alt="JTS" /></td>
<td>R*-Tree</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/50" alt="Foursquare" /></td>
<td><img src="https://via.placeholder.com/50" alt="MongoDB" /></td>
<td>Grids &amp; Geohashes</td>
</tr>
</tbody>
</table>
ACADEMIA

Geo-Social Queries
- Nearby friends with common interests.
- Proximity detection among friends.
- k-Geo-Social Circle of Friend Query
- Socio-Spatial Group Query

Other tasks
- Metrics & Properties
- Link Prediction
- Recommendations
- Location Privacy

[Huang and Liu, Geoinformatics ’09]
[Yiu et al., PVLDB ‘10]
[Liu et al., DASFAA ‘12]
[Yang et al., SIGKDD ‘12]
[S. Scellato et al., ICWSM 2011]
[S. Scellato et al., WOSN 2010]
[Y. Mao et al., SIGIR 2011]
[A. Khoshgozaran et al., CSE 2009]
ACADEMIA

- No unanimously accepted social and spatial storage implementation.
- All data in a single machine.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Y. Doytsher et al., WWW 2012]</td>
<td>Adjacency lists in Neo4j</td>
</tr>
<tr>
<td>[W. Liu et al., DASFAA 2012]</td>
<td>Adjacency matrix</td>
</tr>
<tr>
<td>[Y. Doytsher et al., LSBN 2010]</td>
<td>Edge lists in a RDBMS</td>
</tr>
<tr>
<td>[J. Bao et al., ICDE 2012]</td>
<td>Grid</td>
</tr>
<tr>
<td>[A. Amir et al., PMC 2007]</td>
<td>Quad-Tree</td>
</tr>
<tr>
<td>[W. Liu et al., DASFAA 2012]</td>
<td>R*-Tree</td>
</tr>
</tbody>
</table>
OUR WORK ...

Data Management
A general framework for Geo-Social query processing.

Geo-Social Queries
Basic and advanced (novel) queries processed under the proposed framework.

Geo-Social Ranking
Rank the users based on their social and spatial attributes.

Graph Partitioning
Real-Time Multi-criteria Graph Partitioning task for partitioning the social graph of a GeoSN.
QUERY PROCESSING FRAMEWORK
SM and GM can be administrated by different entities.
- Implement GeoSN queries without owning geo-social data e.g., Agora app.

Fully **dynamic** geographical dataset vs. relatively **static** social structures
- Foursquare’s system downtime.

Easy **integration** of new, more efficient data structures without modifications.

**Novel GeoSN** query types = either a different combination of existing primitives or new ones.
Any primitive must be treated as an **atomic** operation.
- No *states*.
- *NextNearestUser* = multiple calls of *NearestUsers* – keep data locally.
- Find more!

**Efficiency** depends on the underlying **storage scheme**.
- *AreFriends* - Adjacency matrix
- *GetFriends* - Adjacency Lists
- *GetUserLocation* – Hash Table
- *RangeUsers & NearestUsers* – Spatial Indices

They are supported by commercial GeoSNs’ APIs.
GEO-SOCIAL QUERIES
**QUERY PROCESSING**

**RANGE FRIENDS**

**Social Primitives**

- GetFriends($u$)
- AreFriends($u_i$, $u_j$)
- GetDegree($u$)

**Geographical Primitives**

- GetUserLocation($u$)
- RangeUsers($q$, $r$)
- NearestUsers($q$, $k$)

---

**Algorithm 1: $RF_1(u, r)$**

1. $F = \text{GetFriends}(u)$
2. For each user $u_i \in F$
3.   GetUserLocation($u_i$)
4.    If $||q, u|| \leq r$
5.     add $u_i$ into $R$
6. Return $R$

**Algorithm 2: $RF_2(u, r)$**

1. $R_1 = \text{GetFriends}(u)$
2. $R_2 = \text{RangeUsers}(q, r)$
3. $R = R_1 \cap R_2$
4. Return $R$

**Algorithm 3: $RF_3(u, r)$**

1. $U = \text{RangeUsers}(q, r)$
2. For each user $u_i \in U$
3.   If $\text{AreFriends}(u, u_i)$
4.     add $u_i$ into $R$
5. Return $R$

---

Friends of user $u$ within range $r$. 

**Spatial Index**

- Adjacency matrix
- Sparse check-ins
- # primitives

**Dense social network**

No Spatial Index

- Adjacency list
- Independent of check-ins

---

**14**
NEAREST STAR GROUP (NSG Query)

“the next group of five people who come to the restaurant will receive 20% discount”

Ideally:

Socially connected!  $\rightarrow$ Have a common friend (star).

Close to the restaurant  $\rightarrow$ Min. sum of distances to the restaurant

Output: $k$ nearest groups of $m$ users to $q$, such that the users in every group are connected through a common friend (star).
NEAREST STAR GROUP

Example \((k = 1, m = 3)\)

Result: \(\{u_5, u_1, u_3\}\)

Observation:
If user \(u\) is the center user, then his **best** group contains him and his \(m - 1\) closest friends to \(q\).

NSG is not an NP-Hard problem!
NSG QUERY PROCESSING

Basic Notation
\( b_s \): the current best aggregate distance achieved by the already examined users (seen).
\( b_{un} \): the lower aggregate distance that can be achieved by non-retrieved users (unseen).

Skeleton for NSG algorithms (Branch and Bound - BnB)
Input: Location \( q \), positive integers \( m, k \)
Output: Result set \( R \)

1. Initialize \( R, b_s, b_{un} \)
2. While \( b_{un} < b_s \)
3. Get the next nearest user to \( q \)
4. Construct his best group
5. Update result \( R \) and \( b_s, b_{un} \)
6. Refine \( R \)
7. Return \( R \)

<table>
<thead>
<tr>
<th>Eager</th>
<th>Lazy</th>
<th>Eager*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple ( b_{un} )</td>
<td>Simple ( b_{un} )</td>
<td>Aggressive ( b_{un} )</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Find the group</td>
<td>Construct the graph</td>
<td>Find the group</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
EXPERIMENTS

• Storage Schemes
  • Disk-based + Cache
    • Social:
      • Adjacency List: user → sorted list of friends’ ids. (document per user)
    • Geographical:
      • user → coordinates (document per user)
      • Index: Geohashes & Grids
    • Cache: Linux’s caching mechanism
  • Memory-based
    • Social:
      • (Hash Table) Adjacency List: user → sorted list of friends’ ids.
    • Geographical:
      • (Hash Table) user → coordinates
      • Index: Grid (CPM)

• Machine Architecture
  • Centralized: All modules at a single server.
  • Distributed: Separate server for each module (100 Mbps Ethernet)
EXPERIMENTS

- **Real Dataset (Foursquare & Twitter)**
  - **Check-ins**:
    - 12,652 users
    - *same* day (May 30th, 2012)
    - in New York City (1,112 km²).
  - **Social Graph**:
    - 12,652 + 2M (non checked-in friends) users
    - Avg. # of friends: 437.

- **Synthetic Dataset (1M, 2M, 3M, 4M, 5M)**
  - **Check-ins**
    - “The distribution of the distance between two friends follows a power law.”
    - BFS – assign locations: distance is randomly derived by the distribution in:
      - Area: 7,853 km²
  - **Social Graph**: Barabási-Albert preference model
    - Power-law degree distribution.
    - Small-world phenomenon.
    - Avg. # of friends: 100.
EXPERIMENTS

Friends of user \( u \) within range \( r \).

Algorithm 1: \( RF_1(u, r) \)
One GetFriends\((u)\)
Multiple GetUserLocation\((u_i)\)

Algorithm 2: \( RF_2(u, r) \)
GetFriends\((u)\) \( \cap \) RangeUsers\((q, r)\)

Algorithm 3: \( RF_3(u, r) \)
One RangeUsers\((q, r)\)
Multiple AreFriends\((u, u_i)\)

(Average over 100 random queries)

Real Dataset

Memory - Centralized

Real Dataset

Memory - Decentralized

Synthetic Dataset

Memory - Distributed

\( r = 2.5 \text{ km} \)
EXPERIMENTS
NEAREST STAR GROUP (NSG)

**NSG\textsubscript{eager}**
For each newly retrieved user compute his best group eagerly.

**NSG\textsubscript{lazy}**
Construct the social graph around \( q \) iteratively.

**NSG\textsubscript{*eager}**
Similar to NSG\textsubscript{eager} but more aggressive bounds. Refinement step.

(Average over 100 random queries + warm up)

- In the most of the cases \( \text{NSG}^{*}\text{eager} \) is the best.
- Performance scales well with the dataset size.
GEO-SOCIAL RANKING
**GEO-SOCIAL RANKING (GSR)**

**Motivation:** Given a query location \( q \), rank the users based on their social and spatial attributes.

**Possible Solution:** Use NSG or any other existing Geo-Social Query.

**Shortcomings:**
- How many friends \( m \)? Do not restrict it.
- The top-1 NSG maybe really far from \( q \).
- Query issuer does not know the check-ins distribution.

**OK, but:**
- How can I assign a score?
- How far from \( q \) should I search?
- How many friends?

**Key Application:**
- Advertisement: A merchant posts an advertisement using a GeoSN: promising targets \( \rightarrow \) users with high scores \( \rightarrow \) nearby + influence their friends.
CHALLENGE
WHO IS THE TOP-1?

$v_3$ has 3 friends ($v_4$, $v_6$, $v_7$), reasonably close to $q$, and tightly connected to each other.

$v_1$ is the closest to $q$, and has two friends ($v_2$, $v_4$) that are very near $q$.

$v_4$ could influence 5 friends ($v_1$, $v_3$, $v_5$, $v_7$, $v_8$) in the area around $q$. 
THE PROBLEM
FORMALLY

• Relevant Set

Given a query location \( q \), we denote with \( V_i \) the set of friends of user \( v_i \) who are *relevant* for \( q \) (e.g., the friends near \( q \)).

- Ranking score depends on \( V_i \).
- \( V_i \) should be the set of friends who maximize \( v_i \)'s score.
- We can also assume that \( V_i \) contains \( v_i \).

• Geo-Social ranking function \( f(q, v_i) \) assigns to each user \( v_i \) a score that considers:
  - the distance \( ||q, v_i|| \) between the query and \( v_i \),
  - the distance between \( q \) and the users in \( V_i \),
  - the cardinality of \( V_i \), and
  - possibly the social connectivity of \( V_i \).

Given a query location \( q \), a Geo-Social ranking function \( f \) and a positive integer \( k \), **GSR** returns the top-\( k \) users according to \( f \).

• Result: \{\{v_i, V_i, score\}, \ldots\}
OUR CONTRIBUTIONS

• Introduce Geo-Social Ranking problem.

• Propose four ranking functions that cover several practical scenarios.
  • Linear Combination (LC)
  • Ratio Combination (RC)
  • H-Geo-Social (HGS)
  • Geo-Social Triangles (GST)

• For each ranking function, we design a top-k query processing technique.

• Qualitative Evaluation
  • Visualization (Real-World Dataset)
  • Rank Correlation

• Performance Evaluation
RANKING FUNCTIONS
LINEAR COMBINATION (LC)

Ranking function

\[
f_{LC}(q, v_i) = w \cdot \frac{|V_i|}{F} + (1 - w) \cdot (1 - \frac{\sum_{u \in V_i} \|q, u\|}{F \cdot C})
\]

\(w \in (0, 1), F, C: \) normalization factors

Goal: Find the relevant set \(V_i\) that maximizes \(f_{LC}(q, v_i)\).

Solution: Consider the inclusion of a friend \(v_j\) of \(v_i\) in \(V_i\).

\[\text{Social: } \Delta S_i = \frac{w}{F} \text{ vs. Spatial: } \Delta G_i = (1 - w) \frac{\|q, v_j\|}{F \cdot C}\]

Include \(v_j\) iff \(\Delta S_i > \Delta G_i\)

Relevant Set

\[V_i = \{v_i\} \cup \{v_j : v_j \text{ friend of } v_i \land \|q, v_j\| < \frac{w \cdot C}{1 - w}\}\]
RANKING FUNCTIONS
LINEAR COMBINATION (LC)

Example 1: \( w = 0.15, \ C = 30, \ F = 5, \ k = 3. \)
\[
\| q, v_j \| < \frac{w \cdot C}{1 - w} = 5.3
\]
Top-3 results:
1. \( v_1, V_1 = \{ v_1, v_2, v_4 \} \),
2. \( v_2, V_2 = \{ v_2, v_1 \} \), and
3. \( v_4, V_4 = \{ v_4, v_1, v_3 \} \).

Example 2: \( w = 0.8 \)
Issue: \( \| q, v_j \| < \frac{w \cdot C}{1 - w} = 200 \)
Solution: Define a range
i) User-defined
ii) Data-dependent
RANKING FUNCTIONS
RATIO COMBINATION (RC)

Ranking function

\[ f_{RC}(q, v_i) = \frac{|V_i| - w}{\sum_{u \in V_i} ||q, u||} \quad w \in (0, 1) \]

Goal: Find the relevant set \( V_i \) that maximizes \( f_{L^C}(q, v_i) \).

Solution: Add a friend \( v_j \) of \( v_i \) in \( V_i \) iff:

\[ \frac{|V_i| - w}{\sum_{u \in V_i} ||q, u||} \leq \frac{|V_i| - w + 1}{\sum_{u \in V_i} ||q, u|| + ||q, v_j||} \]

Relevant Set

Starting from the nearest friend, keep adding friends to \( V_i \) until:

\[ ||q, v_j|| < \frac{|V_i|}{|V_i| - w} \cdot \frac{\sum_{u \in V_i} ||q, u||}{|V_i|} \]
RANKING FUNCTIONS
RATIO COMBINATION (RC)

Example: $w = 0.001$

Compute $V_2$
Initially, $V_2 = \{v_2\}, f_{RC}(v_2, V_2) = \frac{2}{1} = 2$.
Include $v_4$? ($||q, v_4|| > 2$)

Compute $V_4$
Initially, $V_4 = \{v_4\}, f_{RC}(v_4, V_4) = \frac{4}{1} = 4$.
Include $v_1$? ($||q, v_1|| < 4$) : $V_4 = \{v_4, v_1\}, f_{RC}(v_4, V_4) = \frac{5}{2} = 2.5$
Include $v_2$? ($||q, v_2|| < 2.5$) : $V_4 = \{v_4, v_1, v_2\}, f_{RC}(v_4, V_4) = \frac{7}{3} = 2.33$

Algorithmic tip:
For each user, consider only his friends who are closer to $q$ than him.
RANKING FUNCTIONS
H-GEO-SOCIAL (HGS)

The *h*-index of an author corresponds to the maximum number *h* of his papers that have at least *h* citations.

- Let $D_1, D_2, \ldots, D_l, \ldots$ be an increasing sequence of positive numbers.

- **HGS index $h_i$ of user $v_i$:** the largest value $l$ such that:
  - $||q, v_i|| \leq D_l$ and,
  - $v_i$ has at least $m$ friends within distance $D_m$ from $q$, $\forall m \in [1, l]$. 
RANKING FUNCTIONS
H-GEO-SOCIAL (HGS)

Examples

\[ D_l = \sum_{b=1}^{l} \frac{w+(b-1)w}{2^{b-1}}, \ w > 0 \]

- \( h_1 = 2, \ V_1 = \{v_2, v_4\} \)
- \( h_2 = 1, \ V_2 = \{v_1\} \)
- \( h_4 = 5, \ V_4 = \{v_1, v_3, v_5, v_7, v_8\} \)
- \( h_3 = 0 \)

Algorithmic tip: Candidate Results:
Users within \( D_1 \) and their friends.
RANKING FUNCTIONS
GEO-SOCIAL TRIANGLES (GST)

Ranking function
\[ f_{GST}(q, v_i) = \sum_{\text{triangles } (v_i, u_j, u_p)} e^{-\frac{||q,v_i||+||q,u_j||+||q,u_p||}{w}} \quad w > 0 \]

- Comparable scores to triangles close to \( q \), and exponentially lower scores to triangles with large total distances.

- Relevant friends: All friends of \( v_i \) who participate in triangles with \( v_i \).

Algorithmic Tip:
BnB that generates a candidate set by only considering triangles near \( q \).
VISUALIZATION

- Real dataset: Gowalla
- 6K users
- Checked-in on the same day in Austin (Texas, US).
- Avg. degree 7.6
- Max. degree 390
- Max distance: 32km
**VISUALIZATION**

**SPARSE AREA**

**LC:** Set Constraints
A restaurant sending lunch promotions to potential customers within 1km.

**RC:** Locality is crucial
A cinema has empty seats for a film starting soon, and sends coupons to users in close proximity.

**HGS:** Far but many
A concert promotion targeting users with many friends in the wider area of the concert.

**GST:** Connectivity is essential
Similar to concert, but this time for an event (party) that involves social interaction among the various users.
**CORRELATION**

**KENDALL’S TAU-B (τ_b) RANK CORRELATION COEFFICIENT**

Statistical dependence between **two** ranking functions.

No positive correlation! ➔ Uniqueness of each ranking function ➔ Need for different functions to accommodate various application requirements.
EXPERIMENTS

Centralized Architecture, Main Memory, C++, Real Dataset (Visualization) ($k = 32$)

(a) Sparse

(b) Dense
GRAPH PARTITIONING
THE PROBLEM

EXAMPLE

A GeoSN wishes to **promote** (recommend) upcoming events. Assign each user to an event that minimizes

- the distance/travel time between the user and the event, and
- the social connectivity between users assigned to different events.

- Another criterion: Textual (dis)similarity
- Combination of criteria: Euclidean distance + Textual dissimilarity
THE PROBLEM
REAL-TIME MULTI-CRITERIA GRAPH PARTITIONING

Input:
- A weighted undirected graph $G = (V, E, W)$
- A set of classes $P$.
- Function $c: V \times P \rightarrow \mathbb{R}^+$: cost of assigning a user to a class,
  i.e., $c(v, s_v)$ is the cost of assigning $v \in V$ to $s_v \in P$.

Goal: Assign each user to a class such that the following equation is minimized.

$$RMGP(G, P, a) = c_n \cdot a \cdot \sum_{v \in V} c(v, s_v) + (1 - a) \cdot \sum_{e = (u, f) \in E \land s_u \neq s_f} w_e$$

$a \in (0,1)$
$c_n$: normalization factor
CHALLENGES

NP-Hard

Scalability

Recommendations

Real-Time

Decentralized Environment
RELATED WORK
GRAPH PARTITIONING

Attribute-based
Partition based on similarity of node attributes
[J. Sun et al., SIGKDD ’07]

Connectivity-based
Partition based on connectivity
• Normalized cut [J. Shi et al., TPAMI ’00]
• Modularity.
[M. E. Newman et al., Physical Review ‘04]

Attribute & Connectivity-based
Partition based on both connectivity and attribute similarity
• Connectivity + Eucl. Distance [Y. van Gennip et al. SIAM JAP ‘13]

Platforms for offline processing
RELATED WORK

GRAPH PARTITIONING

Uniform Metric Labeling (UML)

**Input:** (i) Undirected $G(V, E, W)$, (ii) a set $L$ of $k$ labels, (iii) assignment cost function $c: V \times L \to \mathbb{R}^+$, and (iv) a uniform function $d(l, l')$, where $l, l' \in L$, that returns 1 if $l \neq l'$; 0 otherwise.

**Goal:** Minimize

$$\sum_{v \in V} c(v, v_l) + \sum_{e=(u,v)\in E} w_e \cdot d(u_l, v_l)$$

NP-Hard

Same objective function, **but:**

- No preferences and normalization.
- Existing solutions focus only on theory and they are not scalable
  - [J. Kleinberg and E. Tardos, JACM ‘02]: ILP 2-approx. ratio. $O((|E| + k|V|)^{3.5})$
  - [C. Chekuri et al., SODA ‘01]: ILP 2-approx. ratio. $O((k|V| + k|E|)^{3.5}))$
  - [E. C. Bracht et al., JEA ‘04]: Greedy $8logV$-approx. ratio. $O(k|V|^{3.6})$
- No Real-time and decentralized solution.
BASELINE SOLUTION

GAME THEORY – BEST RESPONSE DYNAMICS

• Each node/user is a player who has a cost function that depends on the event \( s_v \) that he will attend and his friends’ decisions.
• His goal is to attend the event \( s_v \) that minimizes his cost function.

\[
C_v(s_v, s_v) = c_n \cdot a \cdot c(v, s_v) + (1 - a) \cdot \sum_{f \in \text{adj}(v) \land s_v \neq s_f} w(v, f)
\]

Algorithm (Best-Responses)
1. Assign a random strategy (event) to each player
2. Repeat
3. For each player \( v \in V \)
4. compute \( v \)'s best strategy (event) wrt the other players’ strategies
5. let \( v \) follow his best strategy
6. Until no player has incentive to change his strategy (Nash equilibrium)
7. Return the strategy of each player
**BASELINE SOLUTION**

**EXAMPLE**

$$a = 0.5$$

Simple but effective for recommendations and gives space for optimizations

<table>
<thead>
<tr>
<th>Steps</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization</strong></td>
<td>$v_1(0.29), v_4(0.735)$</td>
<td>$v_2(0.27), v_5(0.15), v_6(0.385)$</td>
<td>$v_3(0.735)$</td>
</tr>
<tr>
<td>$v_1(0.29, 0.35, 0.235)$</td>
<td>$v_4(0.735)$</td>
<td>$v_2(0.27), v_5(0.15), v_6(0.385)$</td>
<td>$v_1(0.235), v_3(0.735)$</td>
</tr>
<tr>
<td>$v_2(0.65, 0.27, 0.47)$</td>
<td>$v_4(0.735)$</td>
<td>$v_2(0.27), v_5(0.15), v_6(0.385)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_3(0.05, 0.67, 0.735)$</td>
<td>$v_3(0.05), v_4(0.385)$</td>
<td>$v_2(0.27), v_5(0.15), v_6(0.385)$</td>
<td>$v_1(0.235), v_3(0.735)$</td>
</tr>
<tr>
<td>$v_4(0.385, 0.55, 0.77)$</td>
<td>$v_3(0.05), v_4(0.385)$</td>
<td>$v_2(0.27), v_5(0.15), v_6(0.385)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_5(0.77, 0.15, 0.7)$</td>
<td>$v_3(0.05), v_4(0.385)$</td>
<td>$v_2(0.27), v_5(0.15), v_6(0.385)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_6(0.27, 0.385, 0.645)$</td>
<td>$v_3(0.05), v_4(0.385), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.15)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td><strong>Round 2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_1(0.29, 0.35, 0.235)$</td>
<td>$v_3(0.05), v_4(0.385), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.15)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_2(0.65, 0.27, 0.47)$</td>
<td>$v_3(0.05), v_4(0.385), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.15)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_3(0.05, 0.67, 0.735)$</td>
<td>$v_3(0.05), v_4(0.385), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.15)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_4(0.335, 0.6, 0.77)$</td>
<td>$v_3(0.05), v_4(0.335), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.15)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_5(0.67, 0.25, 0.7)$</td>
<td>$v_3(0.05), v_4(0.335), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.25)$</td>
<td>$v_1(0.235)$</td>
</tr>
<tr>
<td>$v_6(0.27, 0.385, 0.645)$</td>
<td>$v_3(0.05), v_4(0.335), v_6(0.27)$</td>
<td>$v_2(0.27), v_5(0.25)$</td>
<td>$v_1(0.235)$</td>
</tr>
</tbody>
</table>
Baseline solution always terminates

- Our game is an exact potential game.
  - i.e., \( \exists \) function \( \Phi \) that expresses the objective functions of all the players.
  - \( C_v(s_v, \bar{s}_v) - C_v(s'_v, \bar{s}_v) = \Phi(s_v, \bar{s}_v) - \Phi(s'_v, \bar{s}_v) \)
- Potential games **always converge**.
- Solution = Local minimum of \( \Phi \).

Time Complexity

- Number of rounds \( \times \) Complexity of each round
- \( O(\max\{|V|, |E|\} \cdot (k \cdot |V| + |E|)) \)

Price of Stability (Best Possible Solution)

- \( PoS = \frac{\text{best equilibrium}}{OPT} = 2 \)
  - best equilibrium = Global minimum of \( \Phi \).

Price of Anarchy (Worst Possible Solution)

- \( PoA = \frac{\text{worst equilibrium}}{OPT} = \text{unbounded} \)

[D. Monderer et al., GEB 1996]
Observation: There are events that a user $v$ will never attend.

- Assume that none of his friends will attend his closest event $s_{v,min}$.
- Cost: $c_{v,min} = a \cdot c(v, s_{v,min}) + (1 - a) \cdot \sum_{f \in \text{adj}(v)} w(v,f)$
- How far he can go in order to meet them?
- User $v$ will attend an event $p$ iff $a \cdot c(v, p) + 0 \leq c_{v,min}$

Valid region of user $v \leq \frac{c_{v,min}}{a}$

Example: $a = 0.5$
OPTIMIZATIONS
PARALLELISM WITH INDEPENDENT STRATEGIES

Observation: if two users are not socially connected, the strategic deviations of one will not affect the best-response of the other.

- \( v_1 \)'s decision influences \( v_4 \).
- But, \( v_1 \)'s decision does not influence \( v_3 \).
- Consequently, \( v_1 \) and \( v_3 \) can select their best response simultaneously.

Main idea:
- Partition the users in groups such that no two users in the same group share an edge. [Graph Coloring]
- The best responses of the users in the same group are computed in parallel.
- Until examined all colors and no strategic deviations happened.


OPTIMIZATIONS
SCHEDULING WITH GLOBAL TABLE

Observation: Numerous redundant computations can be eliminated.

- If the set of $v$'s friends, who follow event $p$ does not change, then the cost of assigning $v$ to $p$ remains the same.
- Even if some of his friends have switched events, it is possible that $v$ is not affected.

Solution: Global Table (GT) index:

Randomly assign each user to an event.

For all users and events, compute their cost and store them to GT.

If a user attends the minimum cost event then he is happy 😊; unhappy ☹ otherwise.

When a user deviates, it updates GT (his and his friend’s records)

During the update, users may become unhappy 😞

Examine only the unhappy users.
**DECENTRALIZED GAME**

1. Initialize, $Q$
2. Create a table $T$ with assignments and store users’ colors.
3. Compute blue users.
4. Update $T$.
5. Compute red users.
6. Deviations

Query $Q$

1. Assignments, & users’ colors
2. Store $T$. 
4. Update $T$. 
2. ACK 
4. ACK 
3. Deviations

Master/Coordinator

Slave/Player

Slave/Player
**EXPERIMENTS**

RMGP VS. UML

Centralized – Main Memory – Real Dataset (Gowalla - Sampling) - C++

$k = 7$, average degree $= 7$
EXPERIMENTS
RMGP CENTRALIZED

Real Dataset (Gowalla): 13K users, 50K edges

Centralized Setting – Main Memory - C++

![Graphs showing performance comparison for different RMGP variants with varying $k$ values and number of rounds. The graphs illustrate the time (sec) and time (ms) for each variant.]
EXPERIMENTS
RMGP: DECENTRALIZED

Real Dataset (Foursquare): 2.2 M users, 27M edges
3 Machines (100Mbps Ethernet)
CONCLUSION AND FUTURE WORK
CONCLUSION

Data Management
A general framework for Geo-Social query processing.

Geo-Social Queries
Basic and advanced (novel) queries processed under the proposed framework.

Geo-Social Ranking
Rank the users based on their social and spatial attributes.

Graph Partitioning
Real-Time Multi-criteria Graph Partitioning task for partitioning the social graph of a GeoSN.
FUTURE WORK

Geo-Social Query Optimization

Translate a GeoSN task $T$ (e.g., range friends) into a structured (language) query $Q_T$, and apply multiple query optimizations techniques over $Q_T$ automatically.

Real Dataset

Memory - Centralized

Real Dataset

Memory - Decentralized

Synthetic Dataset

Memory - Decentralized
THANK YOU
APPENDIX
LC: DATA DEPENDENT
RC: VALUE OF $W$
HGS DISTANCES

The graph illustrates the relationship between the distance (d/i) and the index (i). The graph shows three different types of distances:

- **d/i** (grey line)
- **geometric** (yellow line)
- **arith o geom** (blue line)
- **arithmetic** (purple line)

The x-axis represents the index (i), and the y-axis represents the distance. The graph demonstrates how the distance changes as the index increases.
EXAMPLE GST $W$

- Consider two users, $v_1$ and $v_2$, where
  - $v_1$ participates in exactly one triangle with total distance 2, and
  - $v_2$ is a member of two triangles each having total distance 2.1.

- If $w = 0.1$, then the scores of
  - $v_1$ is $2.2 \cdot 10^{-9}$ and
  - $v_2$ is $1.6 \cdot 10^{-9}$.

- On the other hand, when $w = 1$, the score of
  - $v_1$ (0.13) is lower than that of
  - $v_2$ (0.24).
NSG LAZY
NEAREST STAR GROUP
EAGER SOLUTION

Iteration 1
31.3
31.3
u1
u5
q
u6
u2

b_{un} = 15 < b_s = 31.3

Iteration 2
u4
u2

b_{un} = 17.8 < b_s = 29

Iteration 3
u3
u2
u1
u5

b_{un} = 21.4 < b_s = 29

Iteration 4
u4
u2
u1
u5

b_{un} = 21.8 < b_s = 26.8

Iteration 5
u3
u5
u4
u1
u7

b_{un} = 22.7 < b_s = 26.3

Iteration 6
u4
u2
u1
u5

b_{un} = 23.6 < b_s = 26.3

Iteration 7
u3
u1
u5
u7

b_{un} = 26.4 > b_s = 26.3
NSG EAGER
NORMALIZATION ISSUES

- In several applications, the assignment and social costs may not be comparable.
  - Distance vs. Edge weights
- Direct application maybe meaningless.

**Goal:** When $a = 0.5$,

$$
\sum_{v \in V} c(v, s_v) = \sum_{e=(u,f) \in E \land s_u \neq s_f} w_e
$$

$$
|V| \cdot AC_v = \frac{1}{2} \cdot |V| \cdot SC_v
$$

$$
c_n = \frac{SC_v}{2 \cdot AC_v}
$$
CONCLUSION

• General framework for GeoSN Query Processing
  • Segregate social, geographical and query processing modes.
  • GeoSN query is processed via a combination of primitive operations.
  • Flexible data management and algorithmic design.

• Novel GeoSN queries
  • Nearest Star Group: Social Location-based Advertisement
  • Various solutions based on different sets of primitives.

• Exhaustive Experimental Evaluation
  • Real and Synthetic Datasets.
  • Viability of our framework.
  • Practicality of our GeoSN queries and algorithms.
ACADEMIA

**Geo-Social Queries**

- Nearby friends with common interests.
  - No concrete processing algorithms.

- Proximity detection among friends.
  - Goal: to minimize the communication cost.

- **k-Geo-Social Circle of Friend Query (k-gCoFQ (u, k))**:
  - Returns a strong socially connected and spatially close group of $k+1$ users that contains a given user $u$.

- **Socio-Spatial Group Query (SSGQ(n, k, q))**:
  - Returns a group of $n$ users, such that each member is socially connected with at least $(n - k)$ members, and the sum of distances of all members to $q$ is minimized.

[Huang and Liu, *Geoinformatics* ‘09]

[Amir et al., *Pervasive and Mobile Computing* ‘07]

[Yiu et al., *PVLDB* ‘10]

[Liu et al., *DASFAA* ‘12]

[Yang et al., *SIGKDD* ‘12]
ACADEMIA

Metrics & Properties

• Geo-Social Influence [C. Zhang et al., CIKM 2012]
• Node Locality & Geographic Clustering Coefficient [S. Scellato et al., WOSN 2010]

Other Tasks

• Link Prediction [S. Scellato et al., WOSN 2010]
• Recommendations Y. Mao et al., SIGIR 2011]
• Location Privacy [A. Khoshgozaran et al., CSE 2009]
**QUERY PROCESSING**

**NEAREST FRIENDS**

---

**Social Primitives**
- GetFriends(u)
- AreFriends(u_i, u_j)
- GetDegree(u)

**Geographical Primitives**
- GetUserLocation(u)
- RangeUsers(q, r)
- NearestUsers(q, k)

---

1. \( F = \text{GetFriends}(u) \)
2. For each user \( u_i \in F \)
3. \( \text{GetUserLocation}(u_i) \)
4. Sort \( F \) (asc. ||q, u_i||)
5. \( R = \) top-k of \( F \)
6. Return \( R \)

**Algorithm 1:** \( NF_1(u, q, k) \)

---

**Spatial Index**
- Adjacency list
- Independent of check-ins
- Dense social network

**Algorithm 2:** \( NF_2(u, q, k) \)

1. \( F = \text{GetFriends}(u) \)
2. While \( |R| < k \)
3. \( u_i = \text{NextNearestUser}(q) \)
4. If \( u_i \in F \), add \( u_i \) into \( R \)
5. Return \( R \)

**Algorithm 3:** \( NF_3(u, q, k) \)

1. While \( |R| < k \)
2. \( u_i = \text{NextNearestUser}(q) \)
3. If \( \text{AreFriends}(u, u_i) \)
4. add \( u_i \) into \( R \)
5. Return \( R \)

**k nearest friends of user u to location q.**