Technion Israel Institute of Technology

Querying Geo-social Data by Bridging Spatial Networks and Social Networks

Yerach Doytsher

Ben Galon

Yaron Kanza

Motivation

- **Social networks** provide valuable information on social relationships among people (users)
- Associating users to a **spatial network** can provide geographical information on locations that users visit
- Combining social networks and spatial networks is required for answering queries whose constraints comprise spatial and social conditions

Life Patterns

- ***Life patterns* connect people and places**
- A life pattern is essentially a triple

  (user, geographic entity, time unit)
- For example,

  (Alice, Tower of London, Sundays)

  specifies that Alice visits the Tower of London, every Sunday
- Life patterns can be extracted from GPS logs. As shown in the work of Ye *et al*.

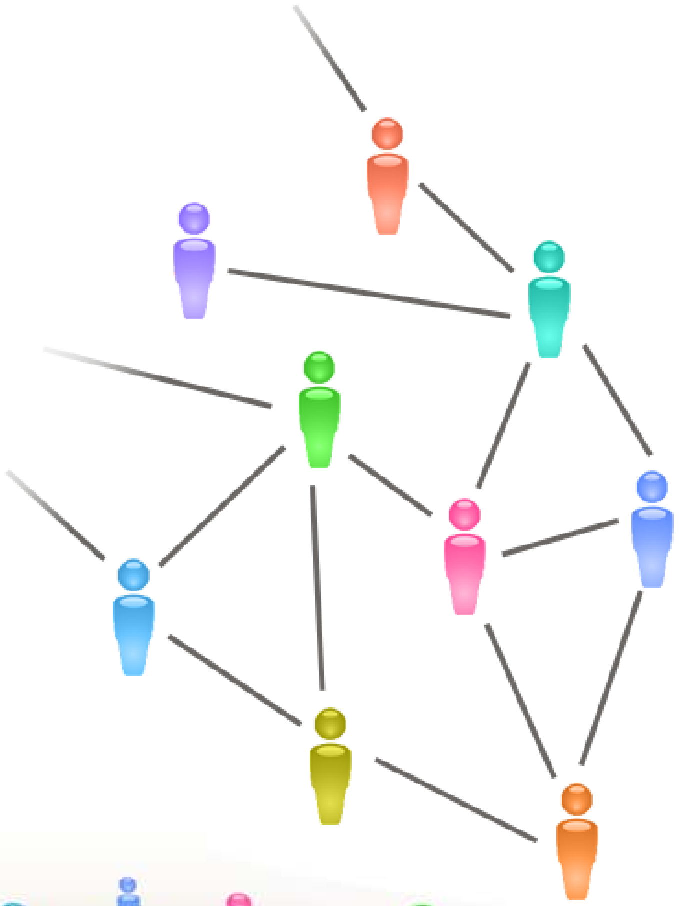
Example

Alice jogs every morning, and she wants to find a partner for jogging

- A potential partner will be someone who:
  1. Is a friend of Alice or a friend of a friend
  2. Frequently jogs in the same area where Alice jogs and at similar times as she does

The life patterns will indicate presence in the same parks at similar times

Proposed Model



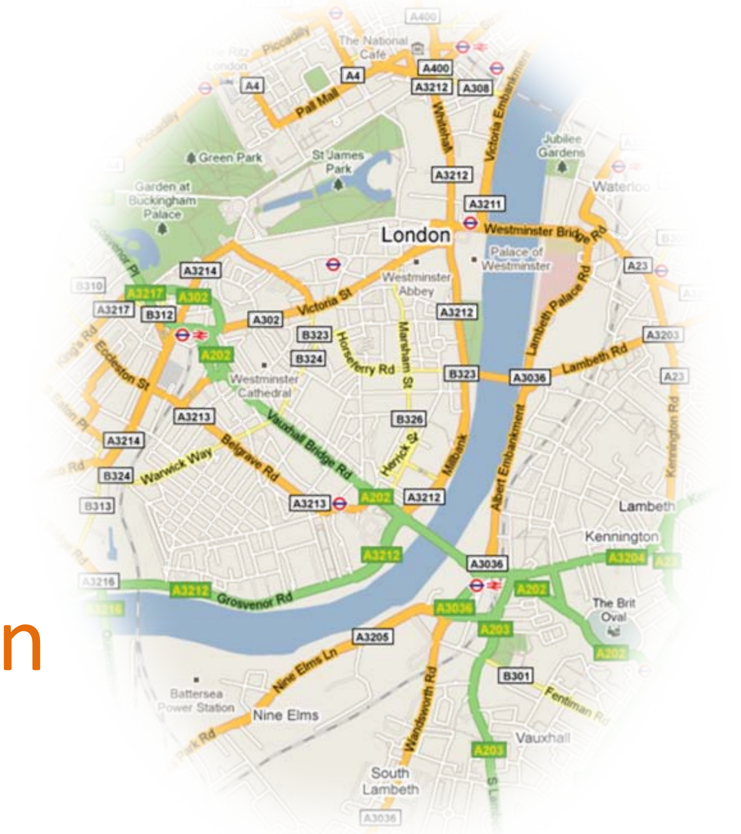
- A social network holds information about people and their relationships

Who are Alice’s friends?

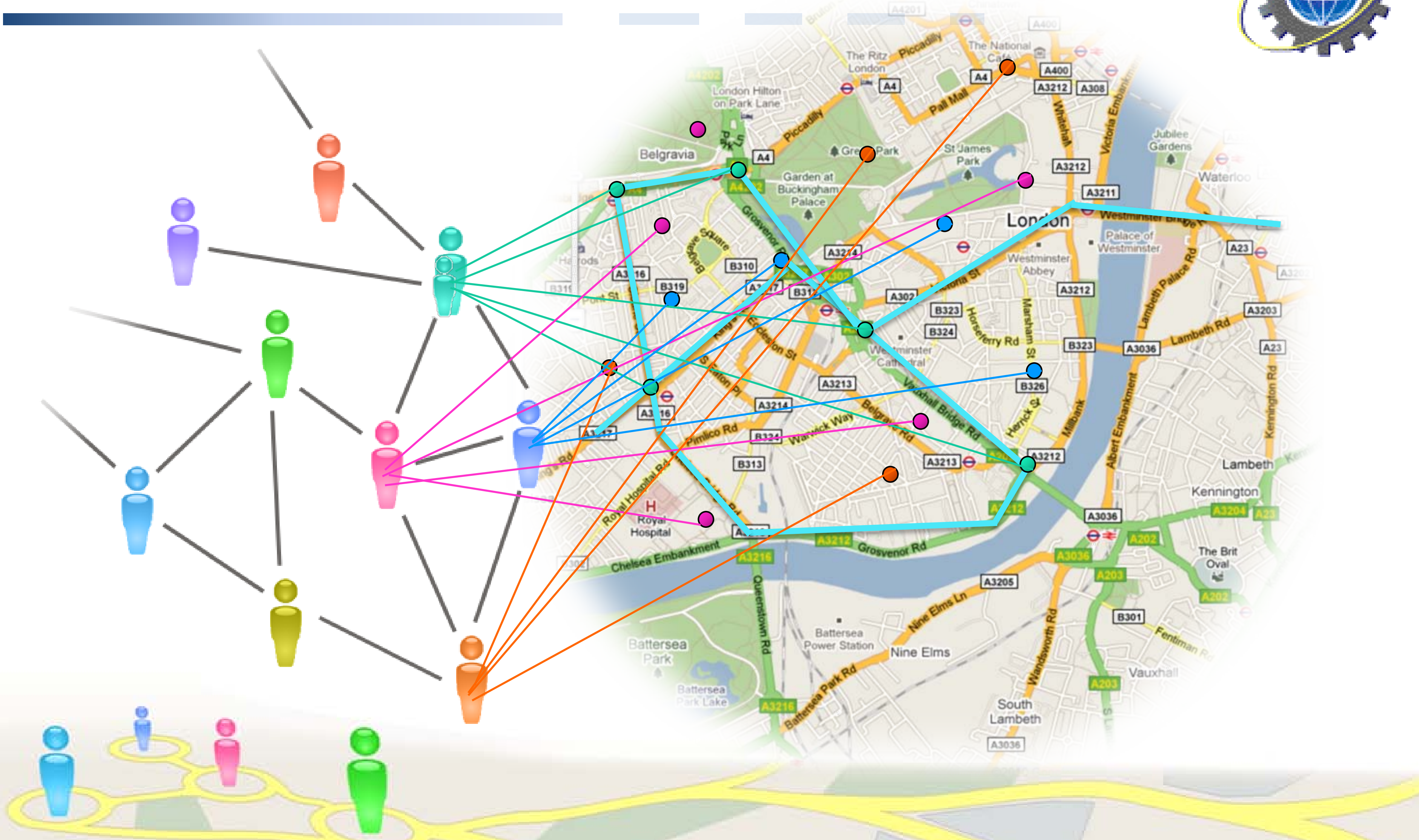
Proposed Model

- A spatial network holds information about spatial entities and their relationships

Where are the parks in Alice’s neighborhood?

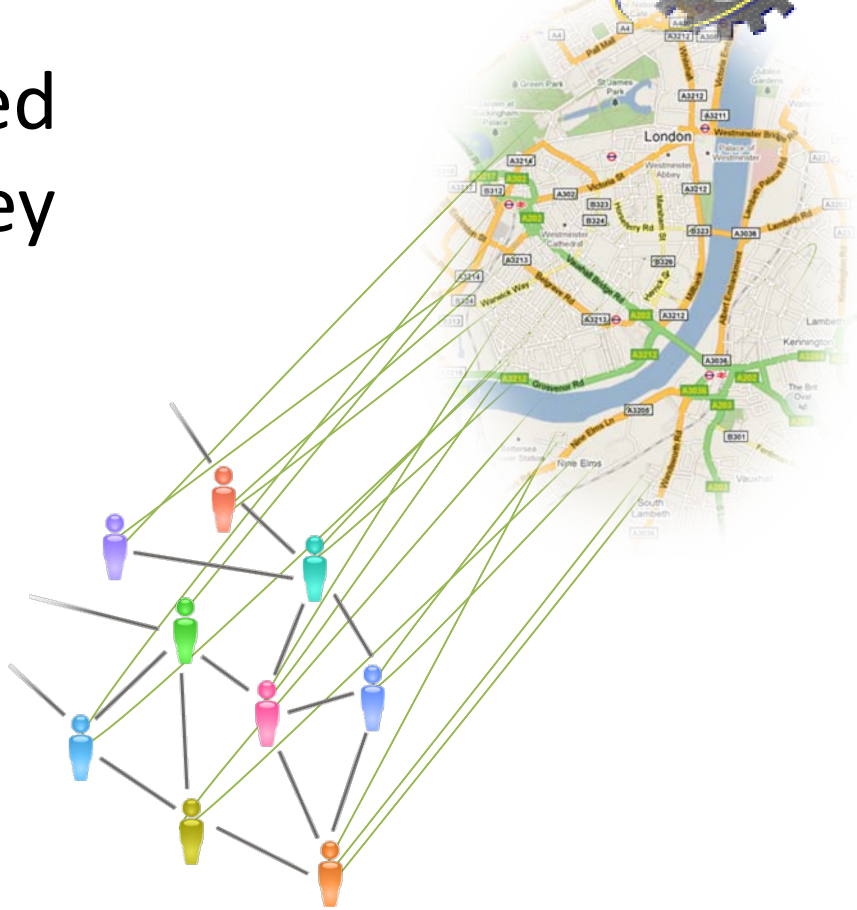


Proposed Model



Integrating the Networks

- Life patterns are generated from GPS log files and they connect people to places they frequently visit



When do these people visit the parks?

Integrating the Networks

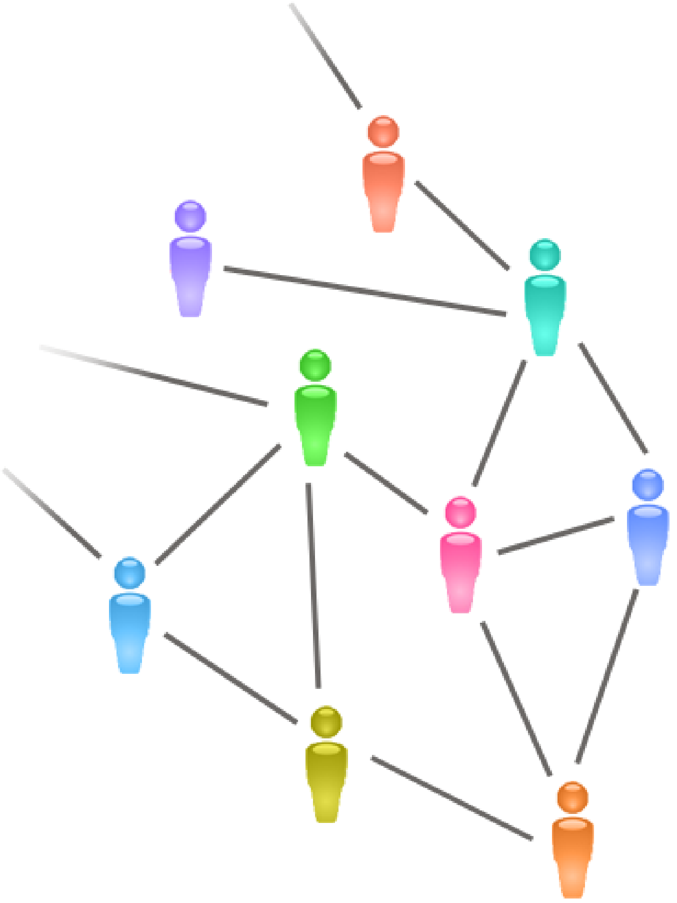
- A spatio-social network (SSN) comprises both networks and the life patterns that connect them

Who has been Where and When

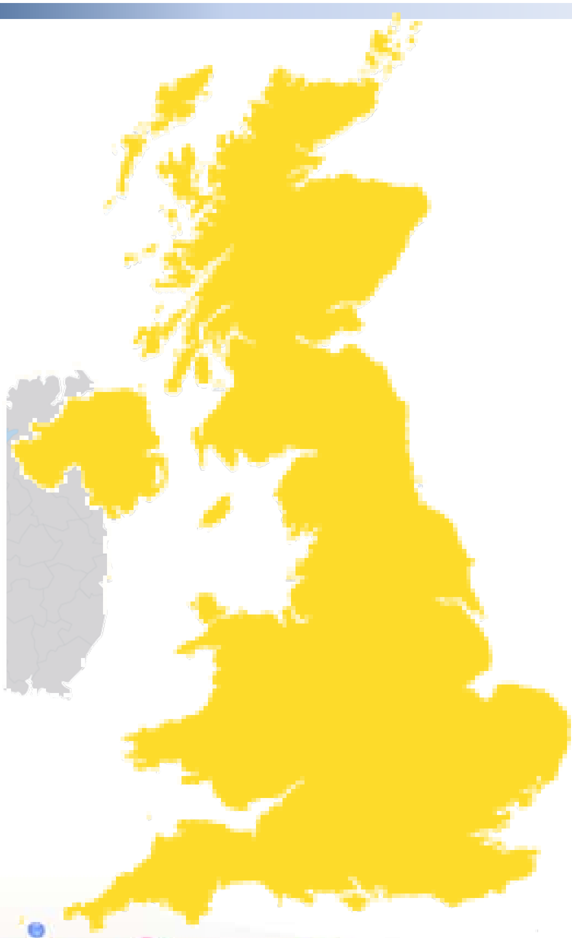
Social Network

The social network is a graph where:

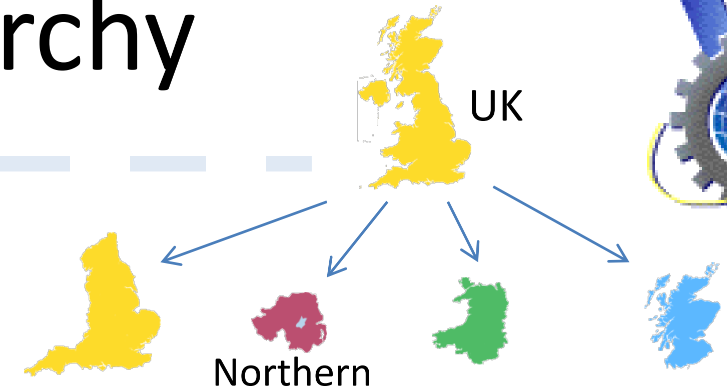
- The nodes represents real-world people, namely **users**, with their attributes
- The edges represent relationships, typically friendship relationships, between users



Geographical Hierarchy



Geographical Hierarchy



UK

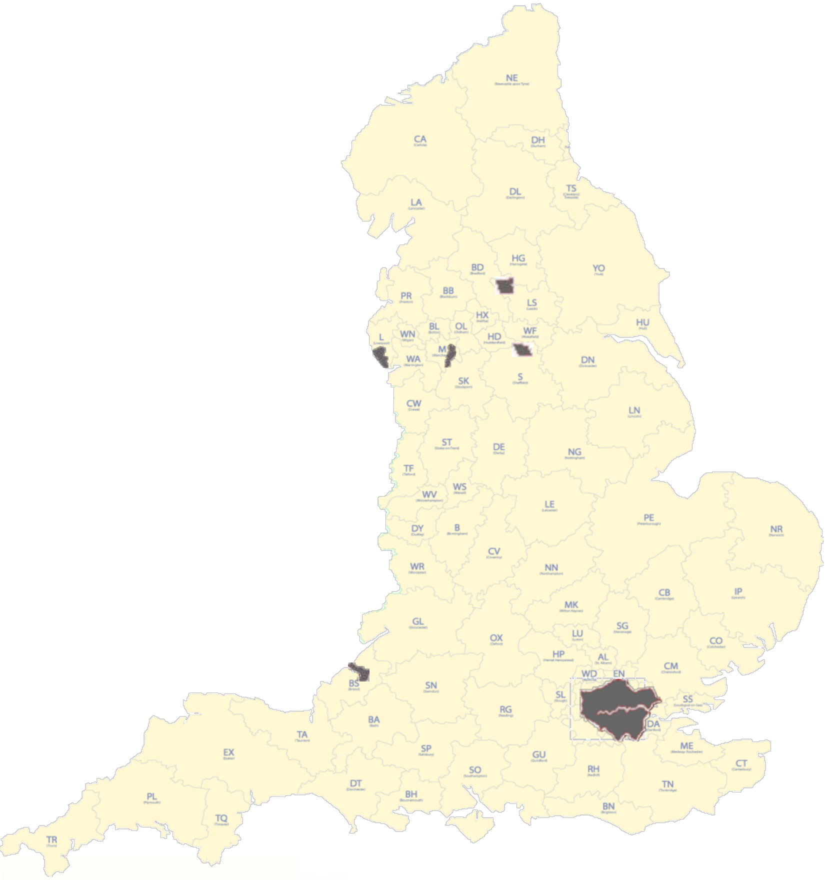
England

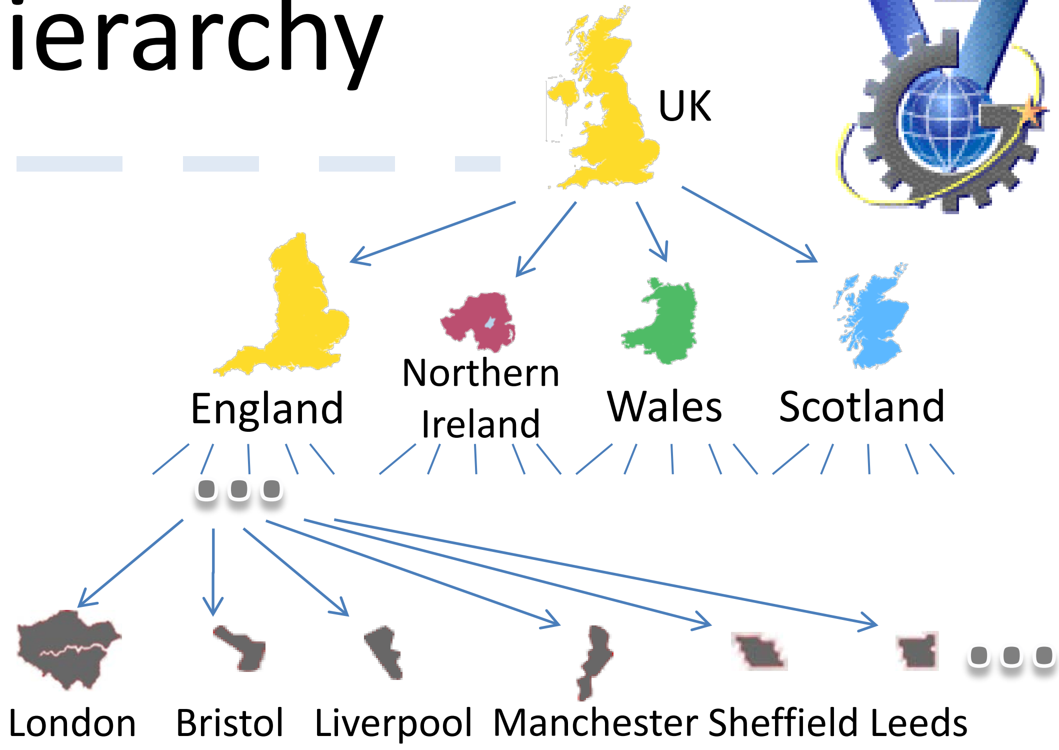
Northern Ireland

Wales

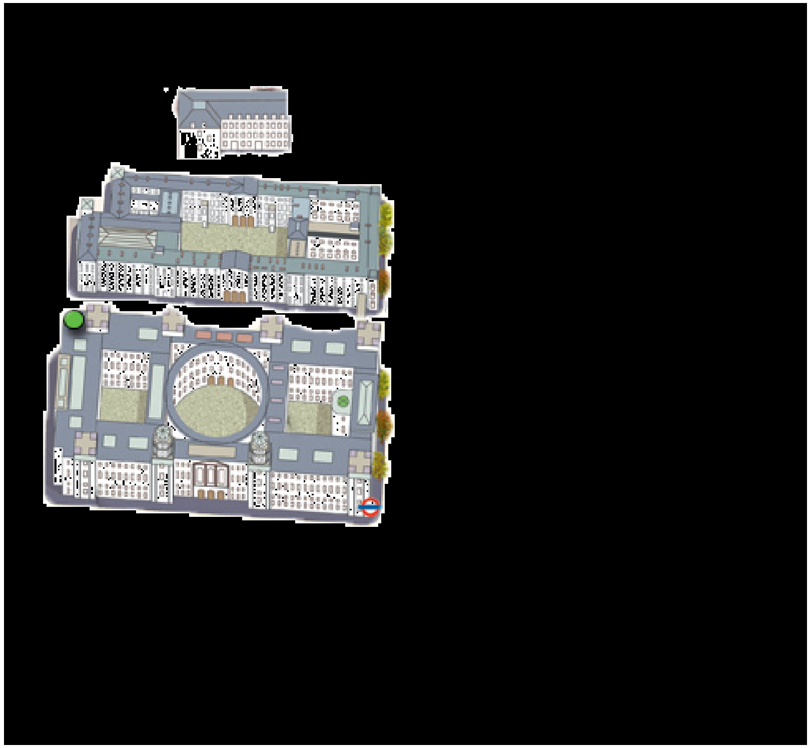
Scotland

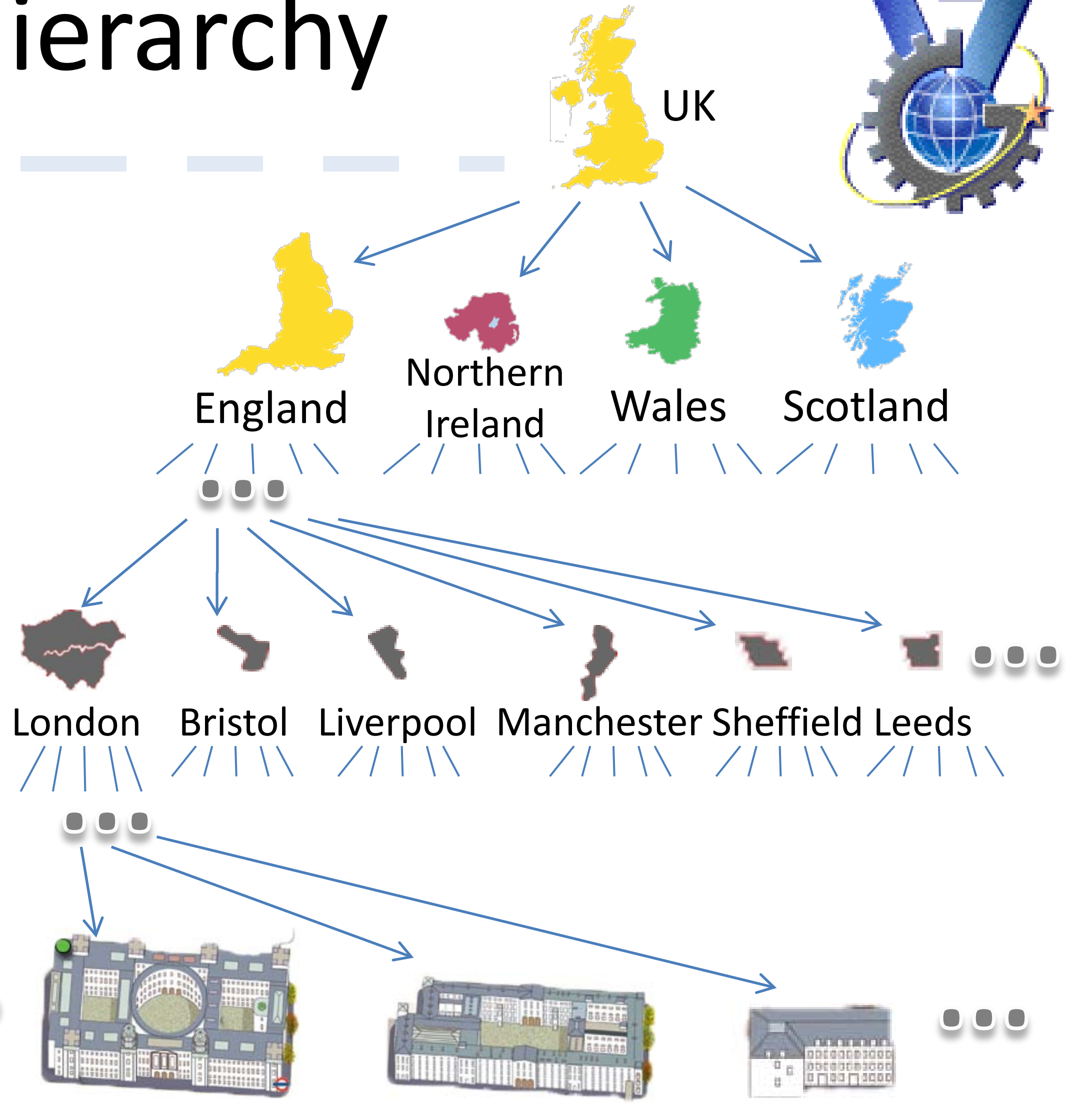
Geographical Hierarchy





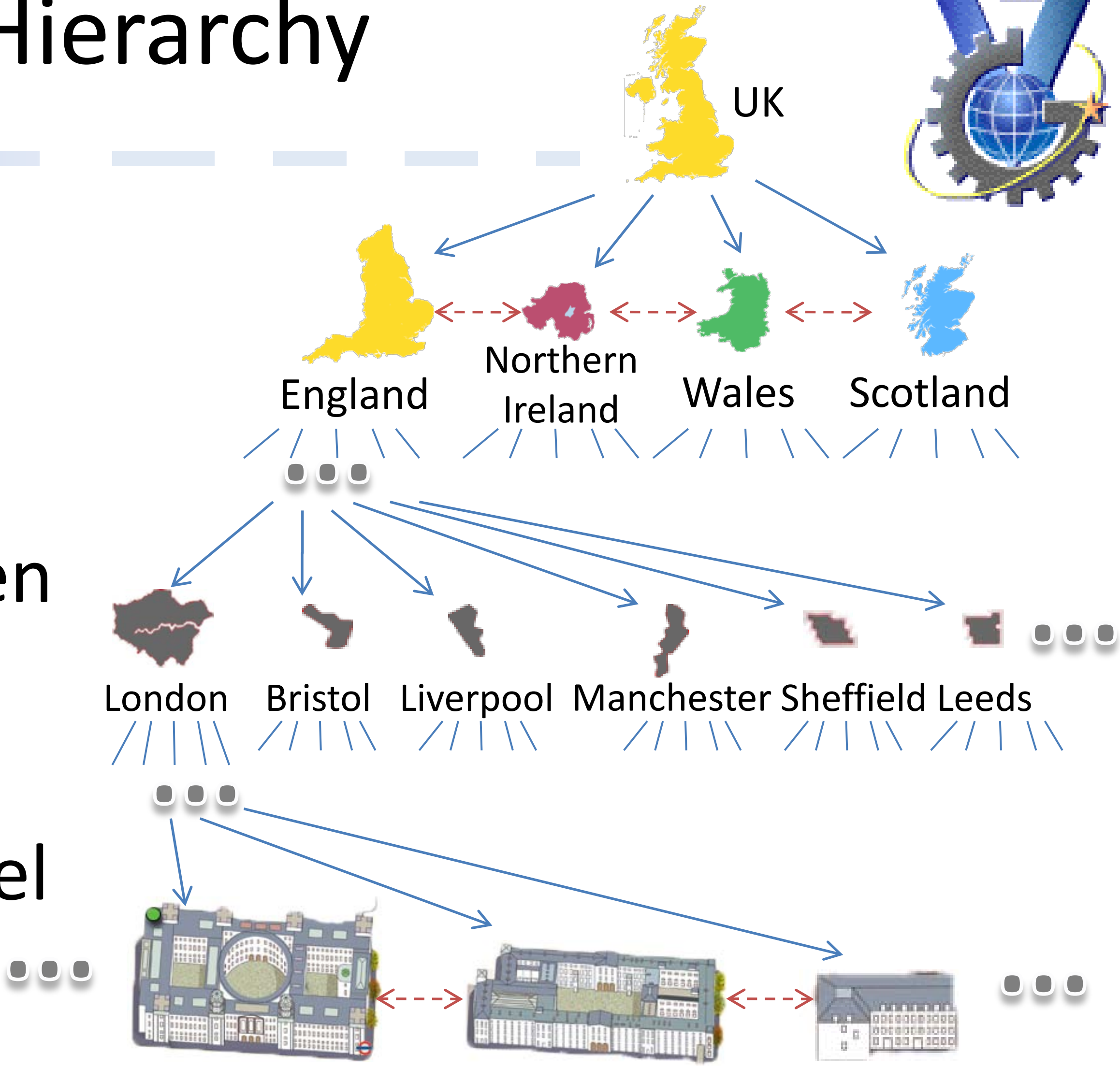
Geographical Hierarchy





Geographical Hierarchy

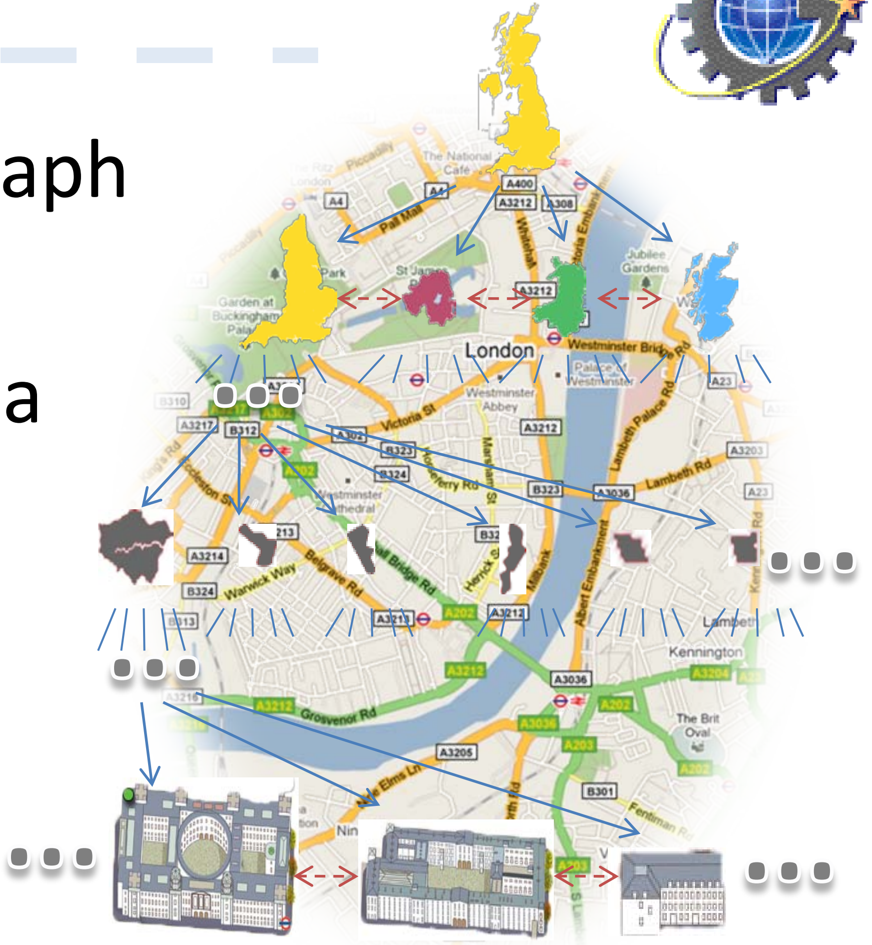
Adjacency edges represent a direct real-world connection between two geographical entities from the same hierarchy level



Spatial Network

The spatial network is a graph where:

- Each node represents a geographic entity
- Two types of edges:
  1. Hierarchical edges
  2. Adjacency edges



Time Patterns

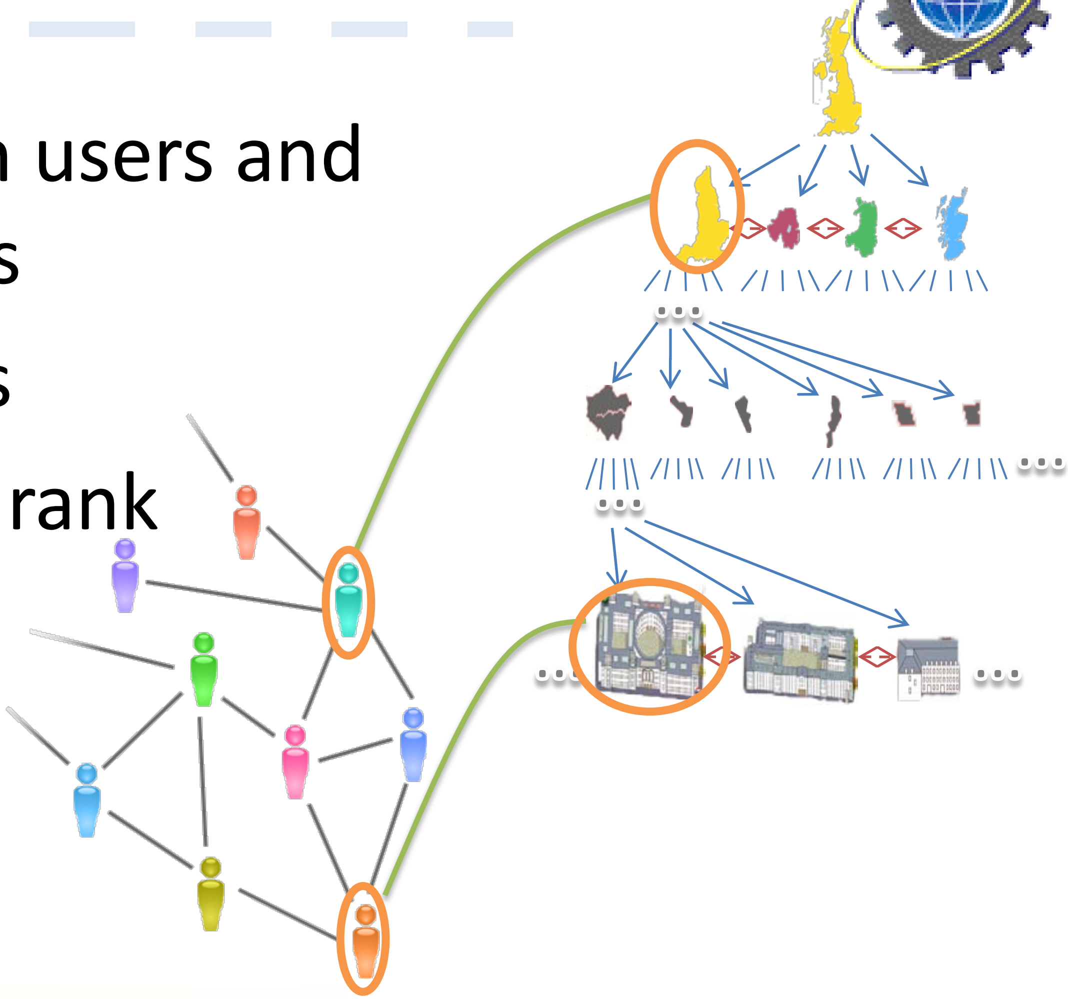
- Time patterns represent repeated events:

  “every week”, “every day”, "every workday”, etc.

- There is a hierarchy of time patterns:
  - If an event happens at some level in the hierarchy, it also occurs in the higher levels
  - If Alice visits 10 Downing St. every workday, then Alice visits 10 Downing St. every week, every month, etc.

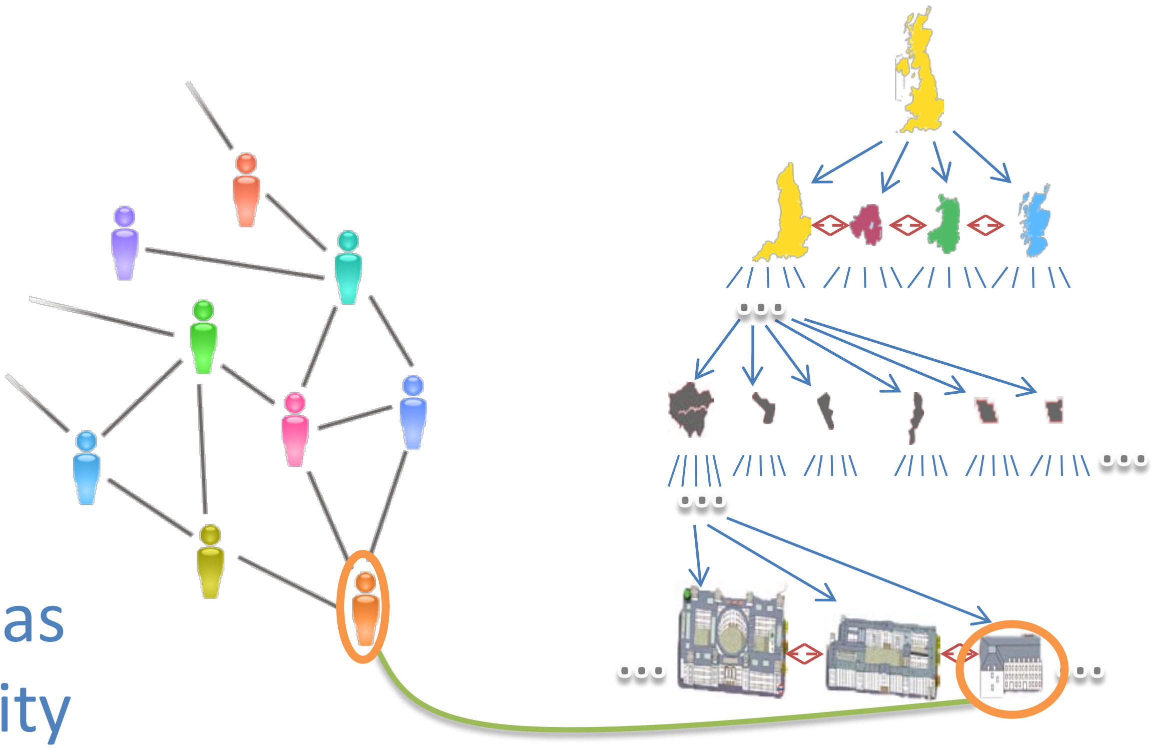
Life Patterns

- Associate between users and geographic entities
- Hold time patterns
- Have a *confidence* rank



Life Patterns – Example

- Alice visits 10 Downing St. every workday from 10 A.M to 12 P.M



Confidence value was omitted, for simplicity

The Query Language

- We developed a query language that has the form of an algebra with seven operators:

1. Select
2. Extend
3. Union
4. Intersect
5. Difference
6. Bridge
7. Multi-Bridge

Each operator returns a collection of nodes **of a single network** (either users or geographic entities)

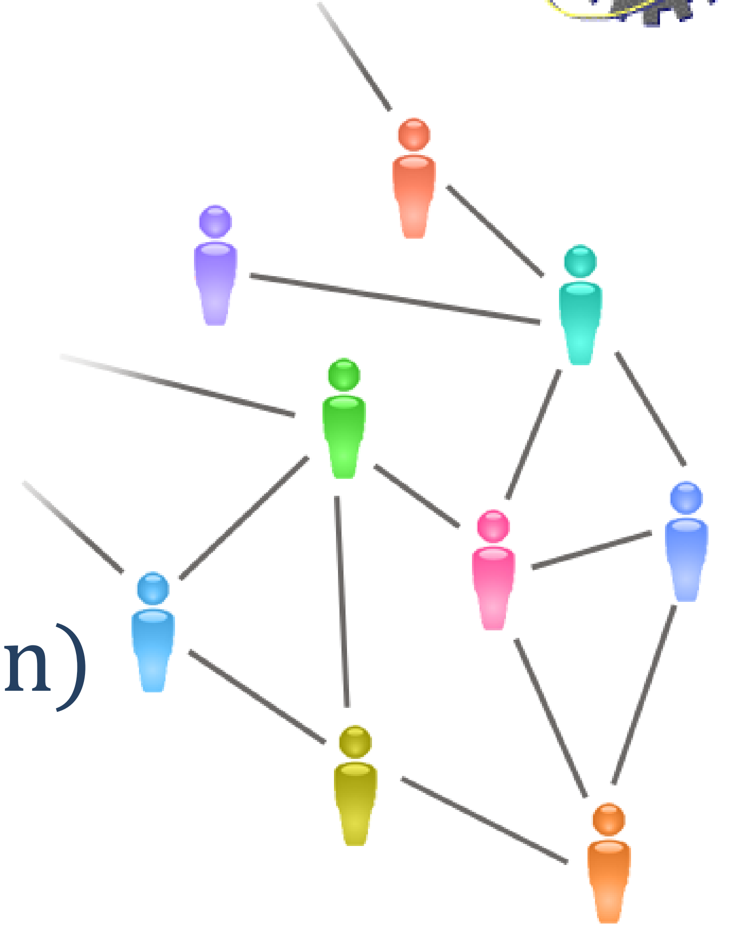
The Algebra

- The proposed algebra was designed to be
  - Expressive
  - Yet, efficient – e.g., no Cartesian product

The Select Operator

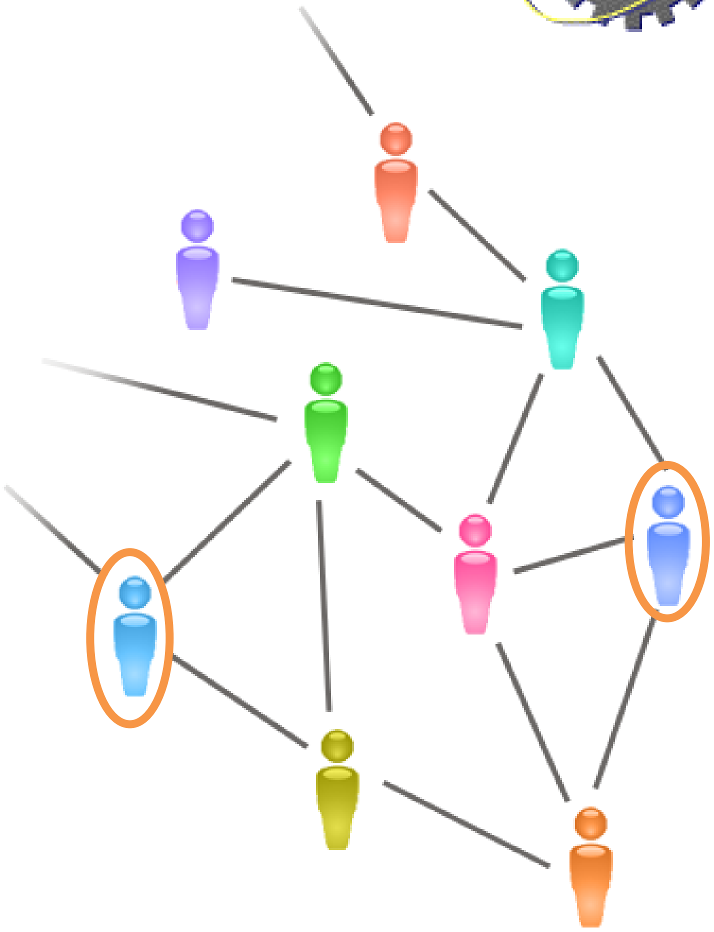
- Receives a set of nodes from a network (social or spatial) and a condition
- Returns the nodes that satisfy the condition

select(nodes_set, condition)



Select – Example

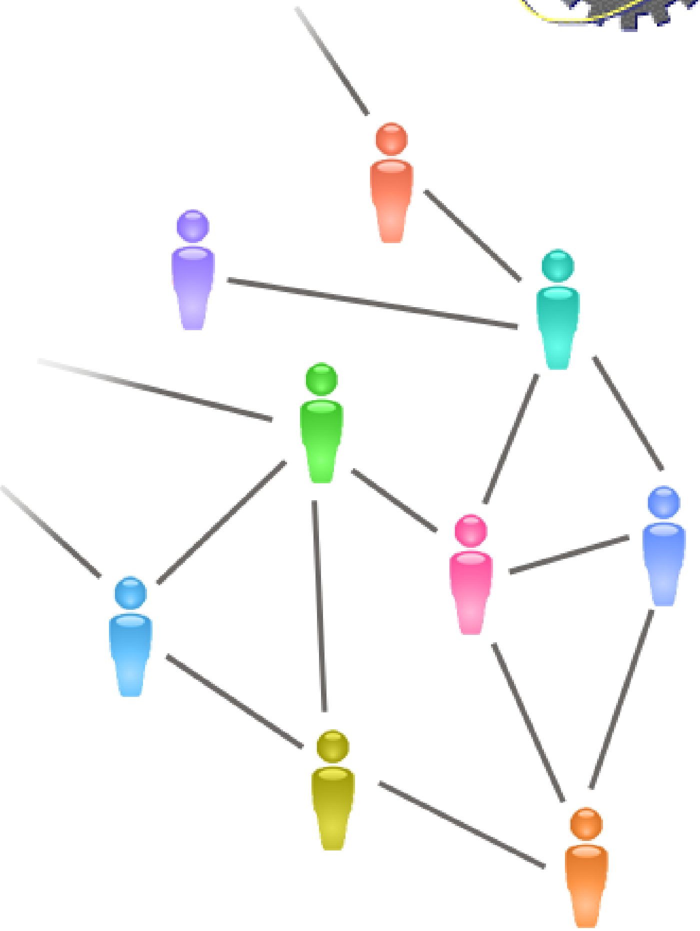
$\text{select}(N_{\text{social}},\ \text{color} = \text{blue}\ )$



The Extend Operator

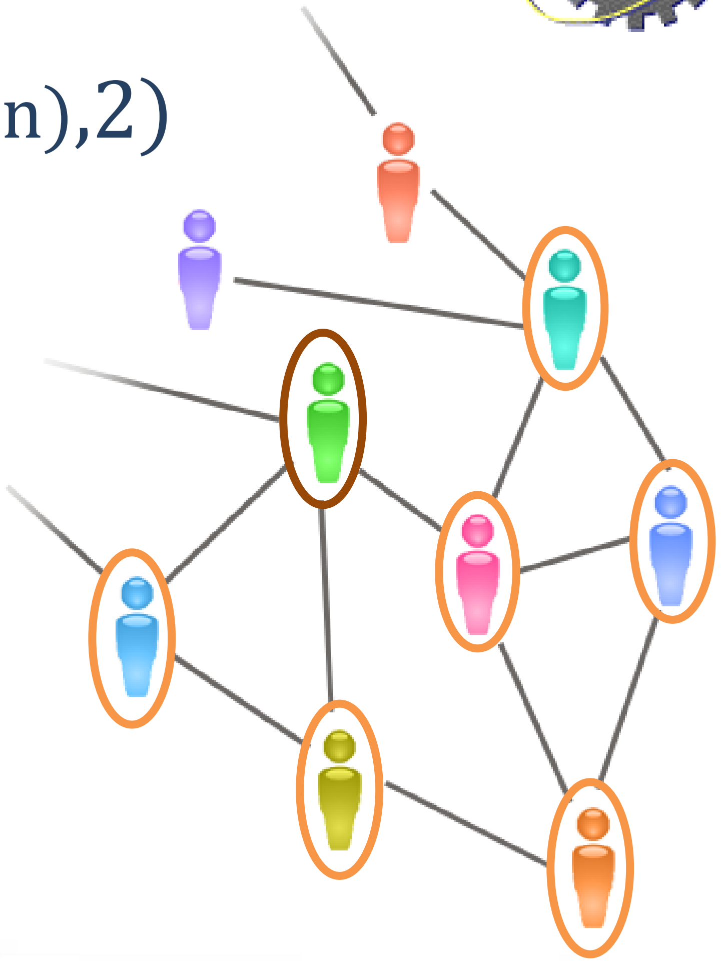
- Receives a set of nodes from a network (social or spatial) and a parameter *n*
- Returns the set of nodes that are reachable by paths with maximum length of *n* from the given nodes

extend(nodes_set, n)



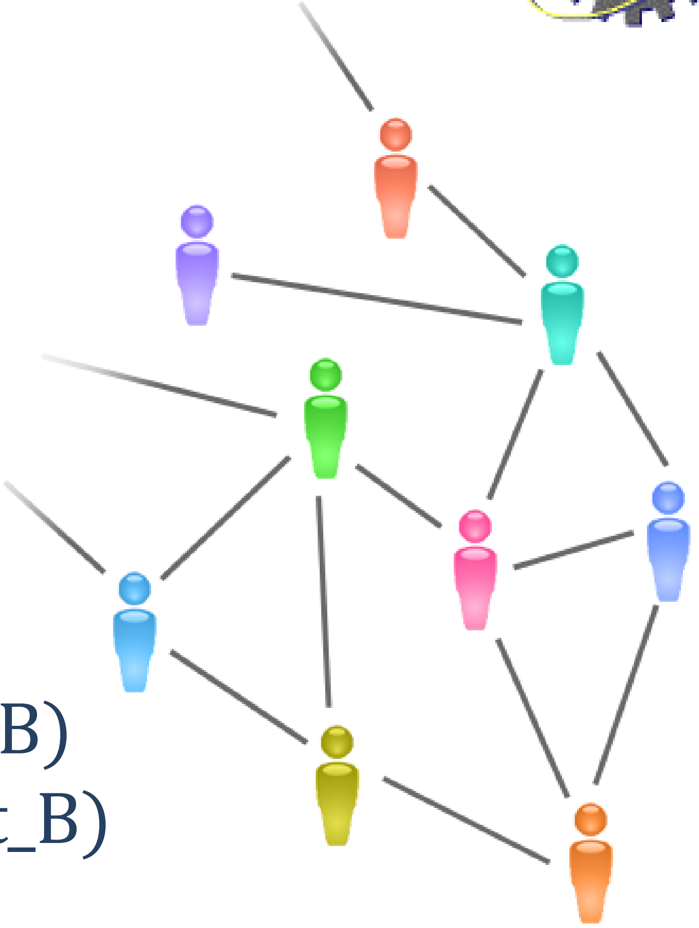
Extend – Example

$\text{extend}(\text{select}(N_{social},\ \text{color} = \text{green}),2)$



Union, Intersect and Difference

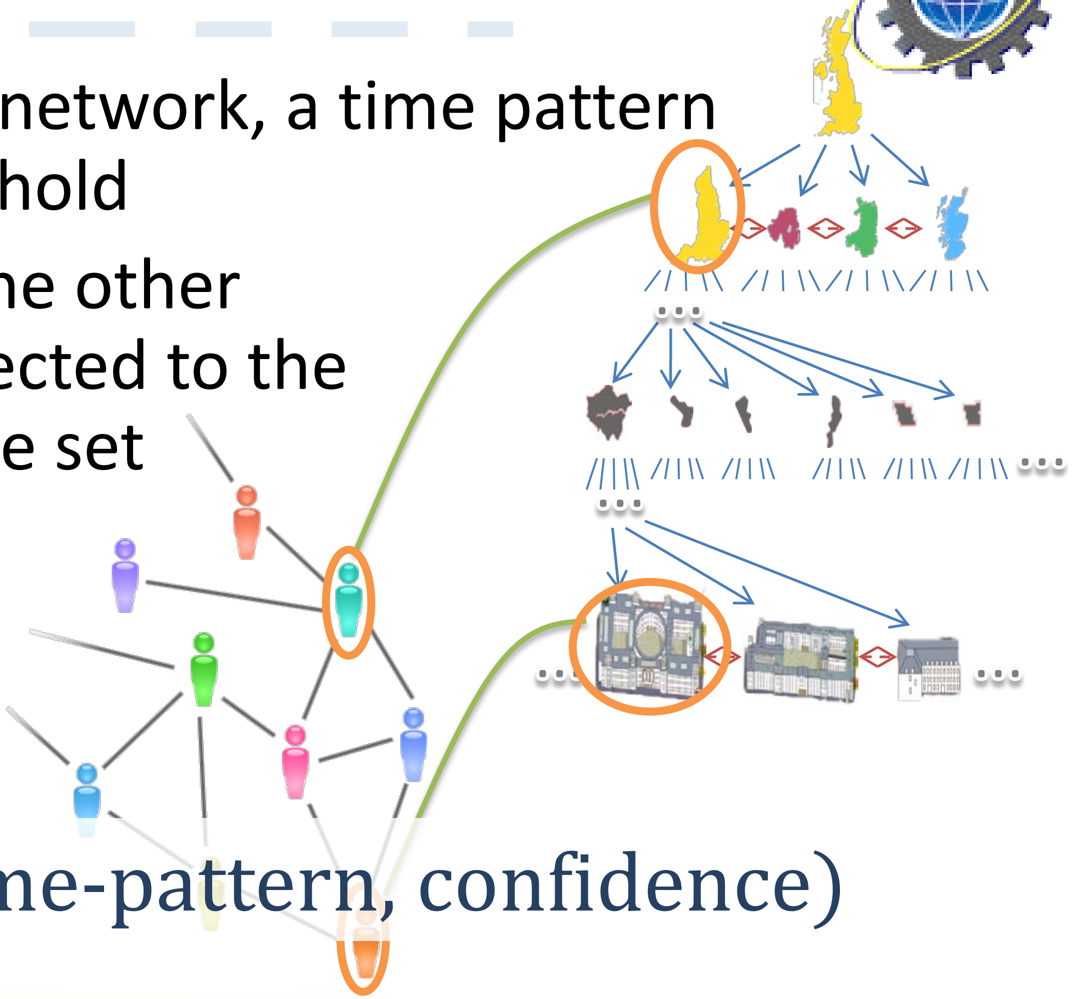
- Receive two sets of nodes – all the nodes from the same network
- Have the same semantics as in set theory



union(nodes_set_A, nodes_set_B)
intersect(nodes_set_A, nodes_set_B)
difference(nodes_set_A, nodes_set_B)

The Bridge Operator

- Receives nodes of one network, a time pattern and a confidence threshold
- Returns the nodes of the other network that are connected to the nodes of the given node set by those life patterns that satisfy the given time pattern and confidence threshold

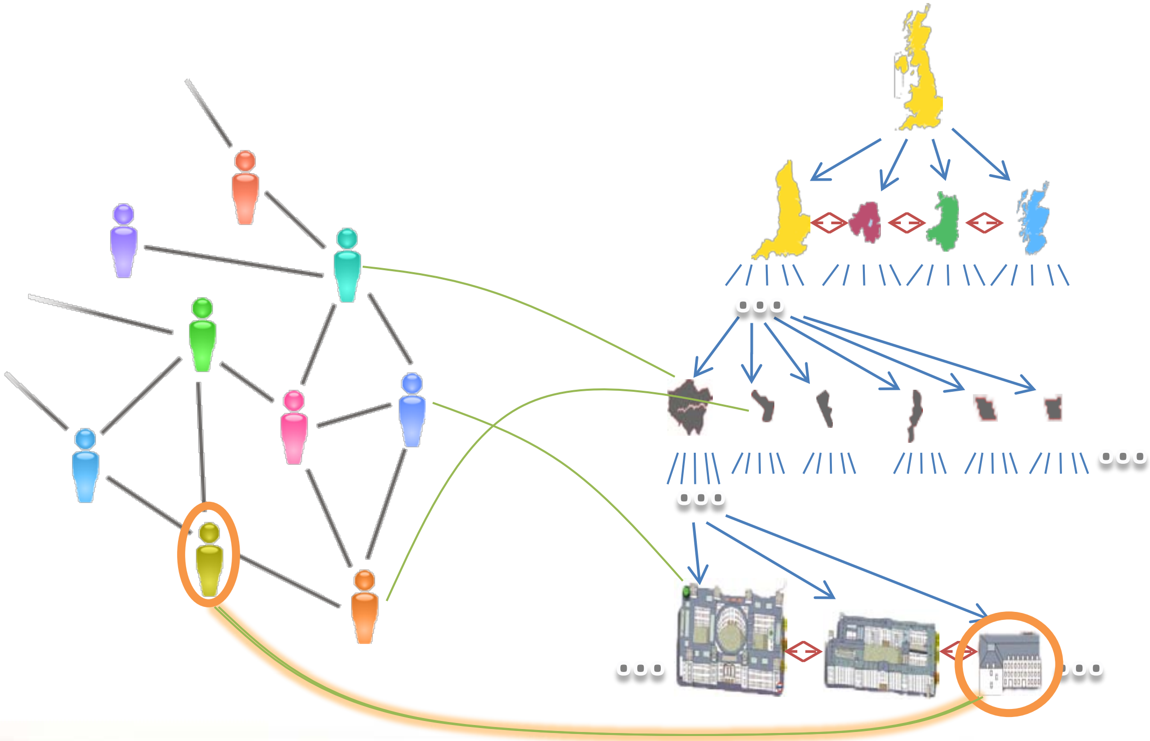


bridge(nodes_set, time-pattern, confidence)

Bridge – Example I

A = select($N_{spatial}$, address like '% 10 Downing St%')

bridge(A, 'every day', 0.8)

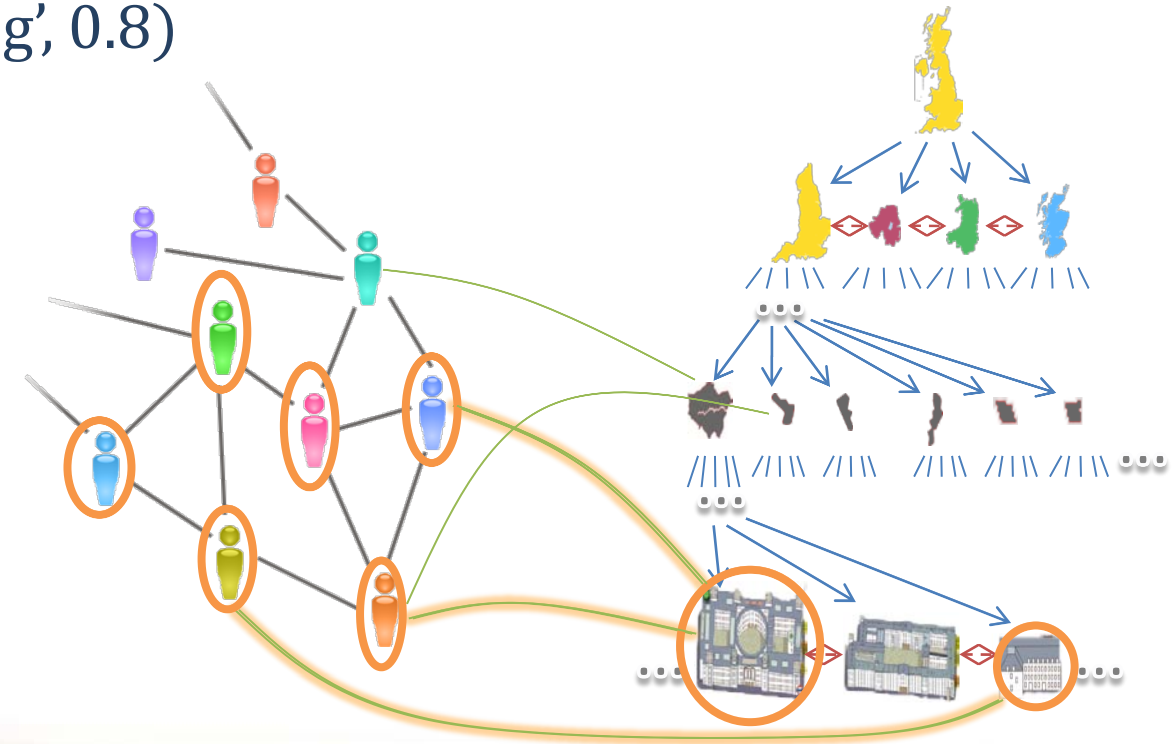


Bridge – Example II

A = select(N_{social}, color = yellow)

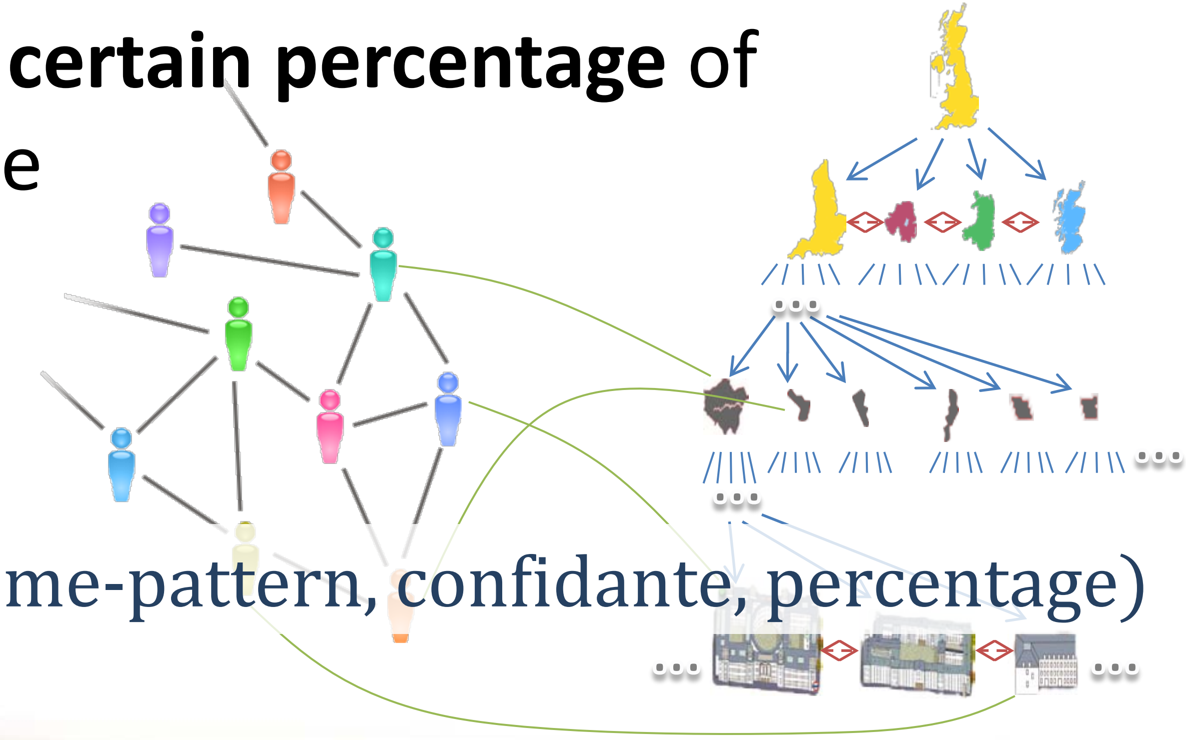
B = extend(A, 2)

bridge(B, ‘every morning’, 0.8)



The Multi-Bridge Operator

- Similar to Bridge, except that the returned nodes are only those that are connected to a **certain percentage** of the nodes of the given set

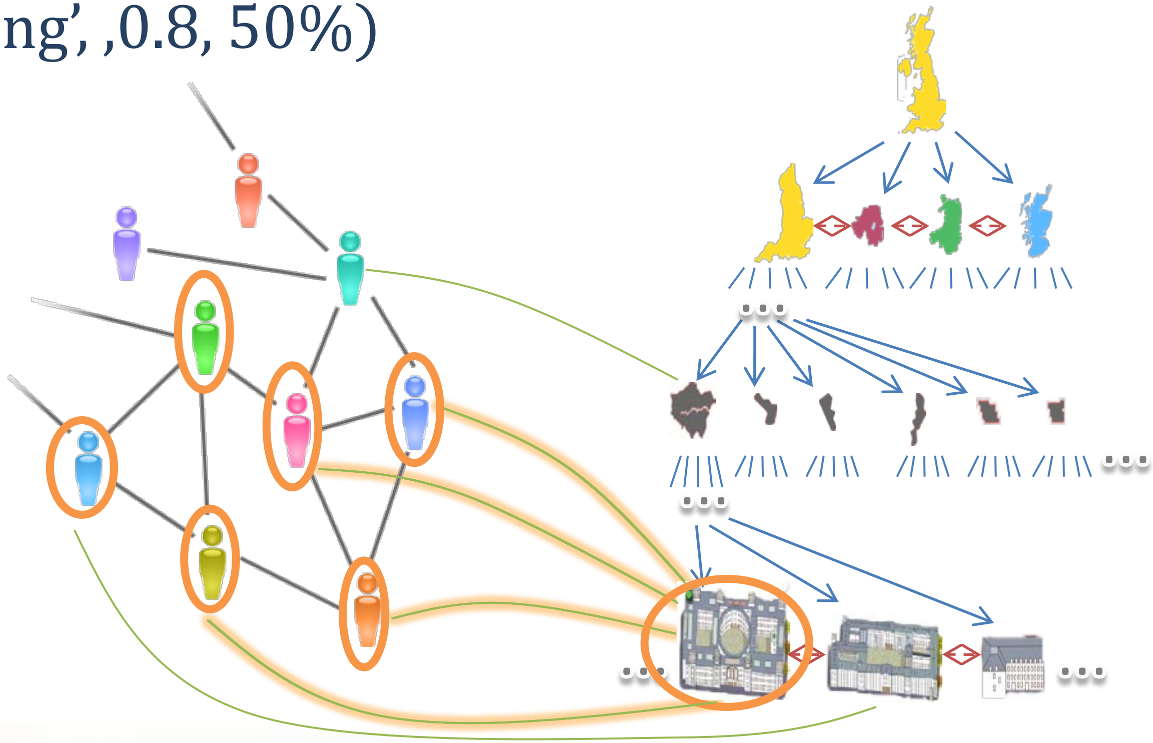


Mbridge(nodes_set, time-pattern, confidante, percentage)

Multi Bridge – Example I

A = select(N_{social}, color = yellow)
B = extend(A, 2)
Mbridge(B, ‘every morning’, ,0.8, 50%)

The operator can be used to discover groups with socio-spatial similarity



Multi Bridge – Example II

John is searching for new friends to go out with

FriendsOfJohn = extend(select(N_{social}, name='John'), 1)

Returns John's friends

Entertainment = select(Mbridge(FriendsOfJohn, 'every week', 0.8, 60%), category='entertainment')

Returns entertainment place where John's friends frequently visit

PotentialNewFriends = Mbridge(Entertainment , 'every week', 0.8, 80%)

Returns people that frequently visit these places

Queries – Example I

Find partners for a carpool

- John lives in Downing St. and works in Heathrow airport
- He wants to find co-workers for a carpool

Neighborhood = extend(select($N_{spatial}$, address like '%Donwning%), 100)

Returns the geographic entities near John's home

Neighbors = bridge(Neighborhood , 'every morning', 0.8)

Returns people who stay every morning in John's neighborhood

Co-workers = bridge(select($N_{spatial}$, address like '% Heathrow airport %) , 'every workday', 0.8)

Returns people that are in Heathrow airport every workday

Queries – Example I

Find partners for a carpool

- John lives in Downing St. and works in Heathrow airport
- He wants to find co-workers for a carpool

Neighbors = bridge(Neighborhood , ‘every morning’, 0.8)

Returns people who stay every morning in John’s neighborhood

Co-workers = bridge(select($N_{spatial}$, address like ‘% Heathrow airport %) , ‘every workday’, 0.8)

Returns people that are in Heathrow airport every workday

Potential = intersect(Neighbors , Co-workers)

Returns potential users for John’s carpool

Queries – Example II

Find a jogging partner for Alice

ParksInAliceHood= select(extend(select($N_{spatial}$, address = Alice_address), 1000), type = park)

Returns the parks in Alice's neighborhood

UsersInParks = brigde(ParksInAliceHood, 'mornings in the week' , 0.6)

Returns people that spend time during the mornings in parks at Alice's neighborhood

Queries – Example II

Find a jogging partner for Alice

UsersInParks = brigde(ParksInAliceHood, ‘mornings in the week’ , 0.6)

Returns people that spend time during the mornings in parks at Alice’s neighborhood

FriendsOfAlice = extend(select(N_{social}, name=‘Alice’), 2)

Returns Alice’s friends

PotentialPartner = intersect(FriendsOfAlice , UsersInParks)

Returns potential jogging partners

Implementation

Goals:

- Demonstrate the feasibility of the model
- Show that a socio-spatial network can be built effectively upon common data-storage tools

Two implementations

1. Relational based
2. Graph based

- Experimentally compare the two implementations

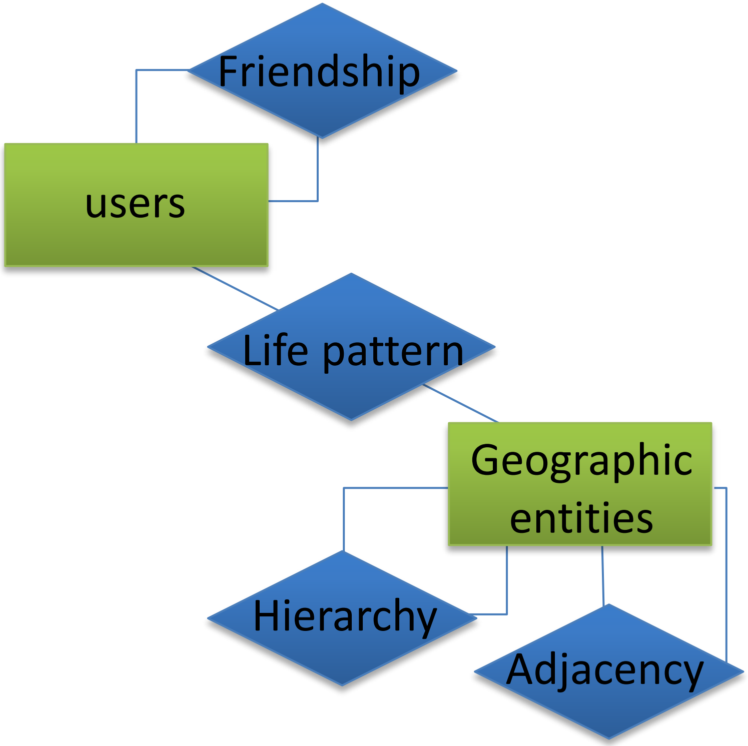
Graph-Based Implementation

- Graph database management system provides a natural storage for the SSN
- The implementation uses Neo4j – an open source graph database management system, in Java
- The SSN network is stored as a graph with attributes on the spatial and social nodes
- Life patterns are edges with the time pattern and confidence as attributes

Relational Implementation

The Relations

- **Users**
- Friendship
- **Geographic entities**
- Hierarchy
- Adjacency
- **Life pattern**

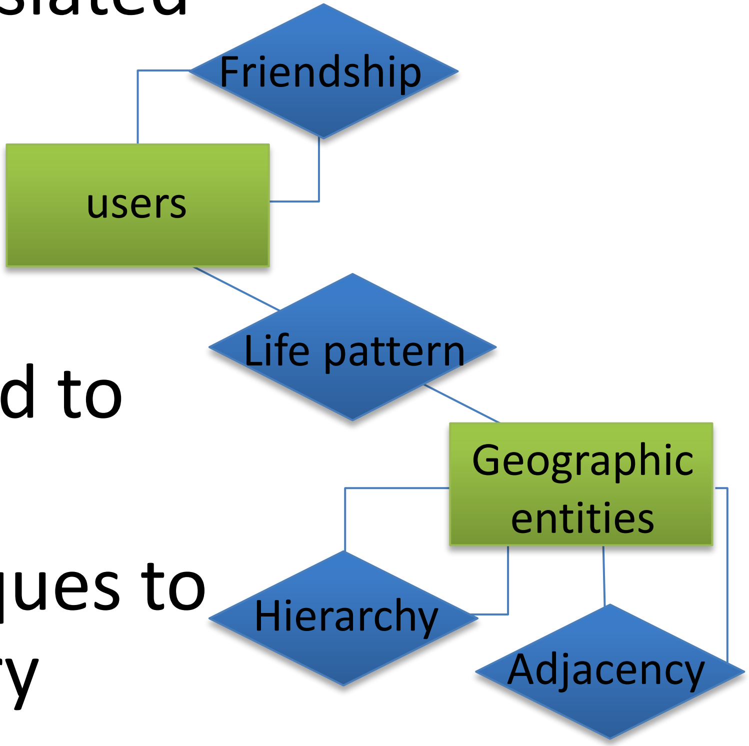


Relational Implementation

- The query operations are translated to SQL queries

```
SELECT user_id
FROM users
WHERE name = ‘John Smith’
```

- Complex queries are translated to nested SQL queries
- We used **optimization** techniques to improve the efficiency of query evaluation

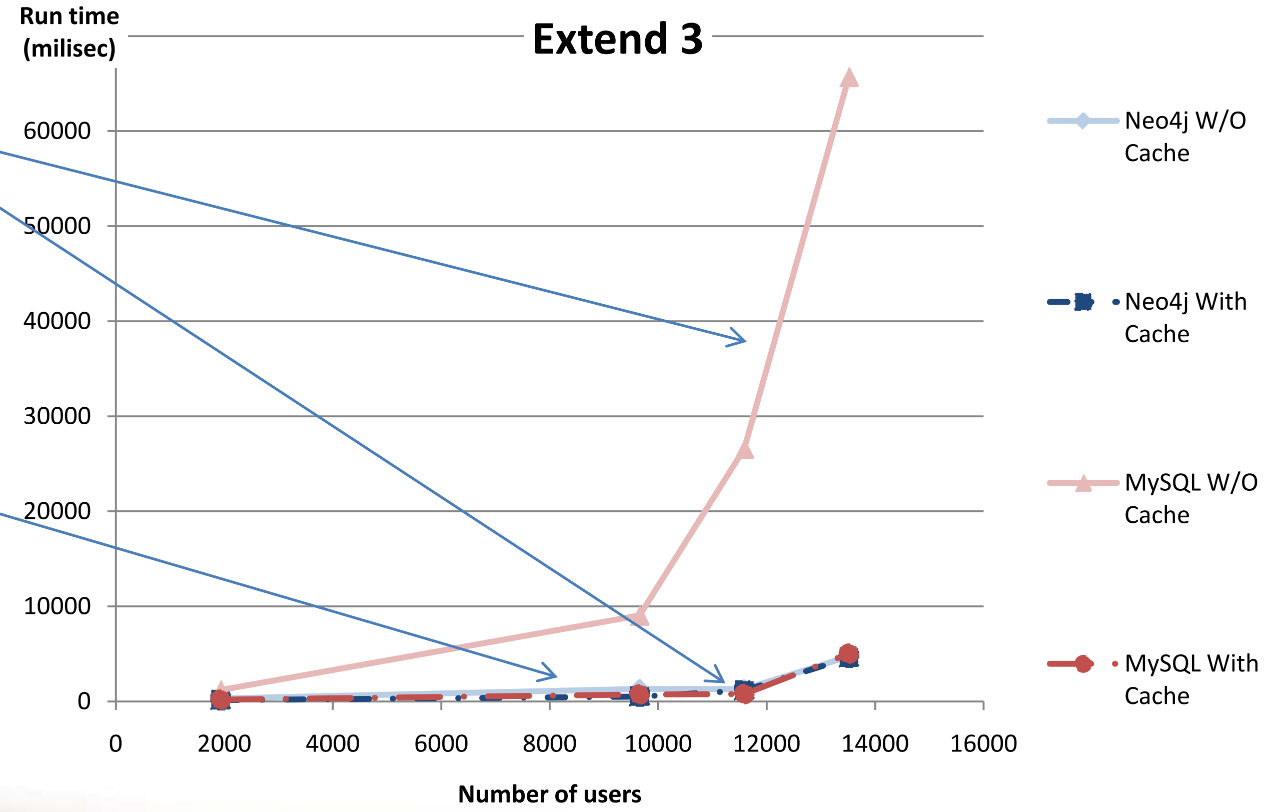


Experiments

extend(N, name = 'john', 3)

Large effect of a cache on the running time, in the relational-based implementation

Almost no effect of the cache on the running time, in the graph-based implementation

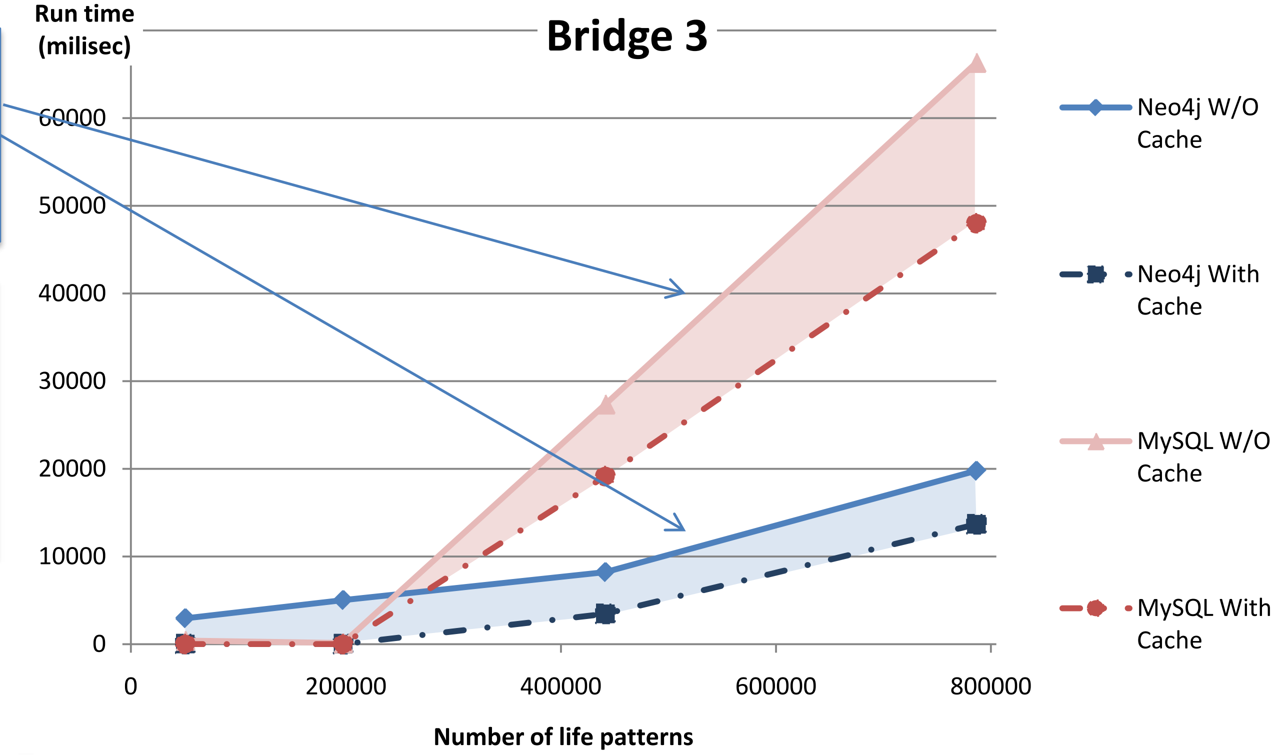


Experiments

bridge(bridge(bridge(select(N, name='John'),*,0.8),*,0.8),*,0.8)

The graph model shows better result for large datasets

In both models the cache significantly improves the efficiency



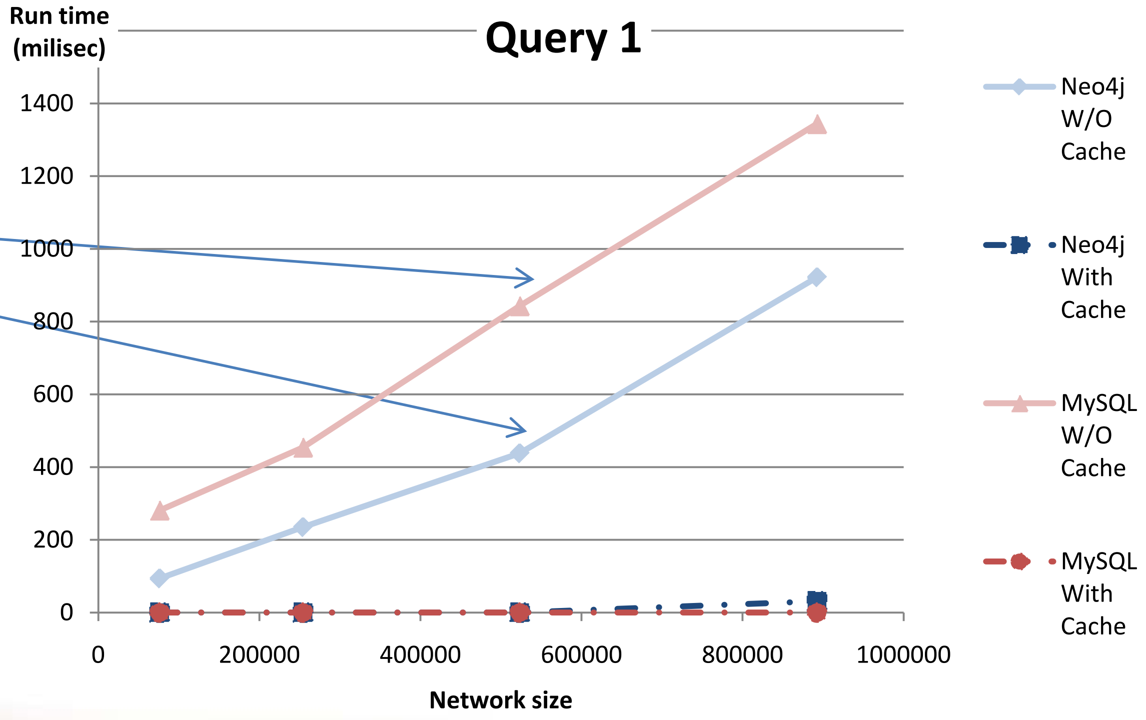
Experiments

Paramedics = Select(N_{social}, occupation=`Paramedics')

Query_1 = Bridge(Paramedics, `some_night_of_the_week',0.85)

Find where paramedics might live

Queries with bridge are evaluated more efficiently over the graph DBMS than over the relational DBMS



Experiments

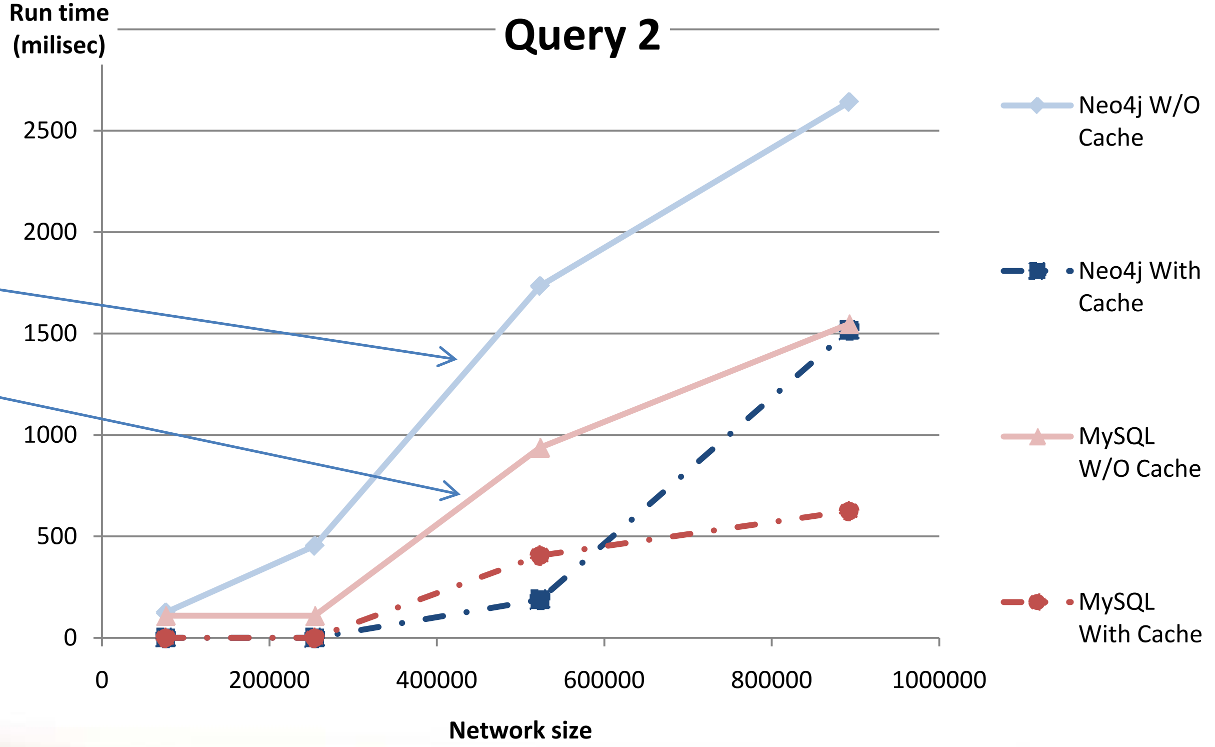
John = Select(N_{social}, *name=`John')*

Places = Bridge(John, *all, 0.5)*

Query_2 = MBridge(Places, *all, 0.5, 20%)*

Find people that visit in 20% or more at the same places as John

The evaluation of Mbridge is more efficient over RDBMS than over graph DBMS



Conclusions

- We presented a model for representing the integration of social networks with spatial networks
- We provided an algebra that allows querying the combined networks, effectively and efficiently
- We compared an implementation of the proposed model over graph DBMS and RDBMS, and we illustrated the superiority of the graph DBMS over the RDBMS, for all the operators except the multi-bridge

Thank You

Questions?