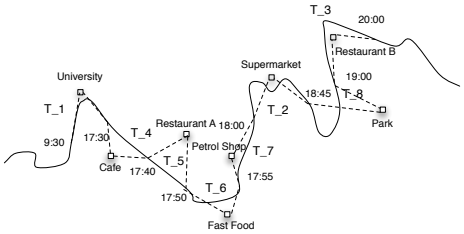
From Trajectories to Activities: A Spatio-Temporal Join Approach*



PAMS

...			
...			
...			
Restaurant	Delivery	15 minutes	45 minutes
Restaurant	Dining	30 minutes	2 hours
Restaurant	Working	3 hours	8 hours
University	Studying	2 hours	12 hours
University	Working	4 hours	12 hours
Petrol Shop	Fueling	10 minutes	30 minutes
Supermarket	Shopping	30 minutes	2 hours
Fast Food	Takeaway	10 minutes	45 minutes
Park	Playing	15 minutes	1 hour
...			
...			
...			

Kexin Xie, Ke Deng, Xiaofang Zhou

School of Information Technology & Electrical Engineering
University of Queensland
Brisbane, Australia
{*kexin, dengke, zxf*}*@itee.uq.edu.au*

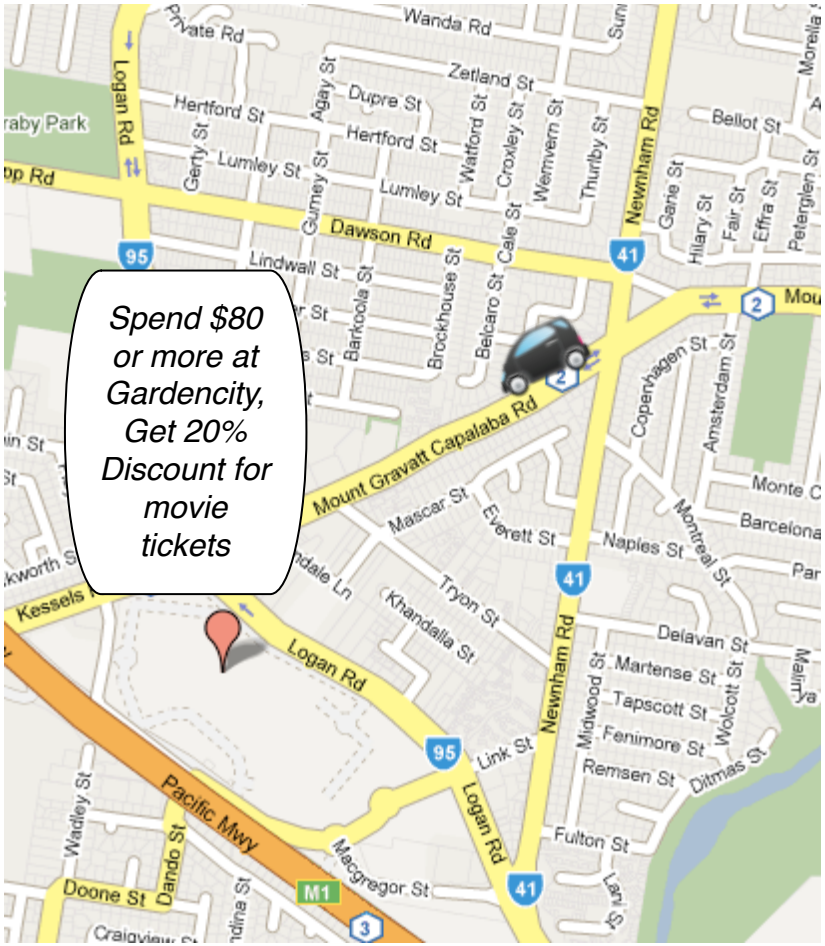
* This work is partially supported by Microsoft Research Asia Internet Services Theme Research Program

Activity Sequences

- What do we do each day?
- E.g.
  - Drink after work?
  - Drunk after semester ends?
  - Jogging for 1 hour every Saturday?
  - Watching movie after 4 hours of shopping?
  - Only have one meal everyday during semester breaks?
  - Sleep when the sun rises?

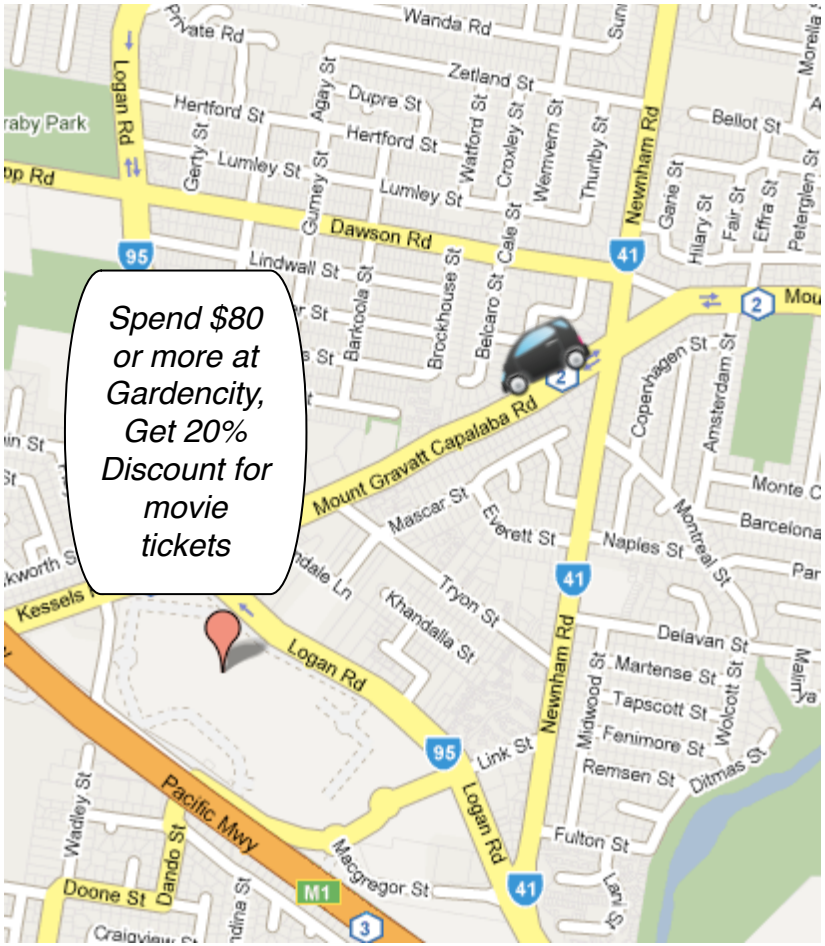
Data Mining with Activity Sequences

- Technique => Knowledge
  - Frequent Sequential Pattern => Common Behavior (e.g. Collaborative promotions)
  - Periodic Pattern => Periodic Behavior (e.g. POI suggestion)
  - Outlier Detection => Odd Behavior (e.g. Criminal Investigation)
  - Clustering => Similar Behavior (e.g. Friend suggestion)
- The Problem?



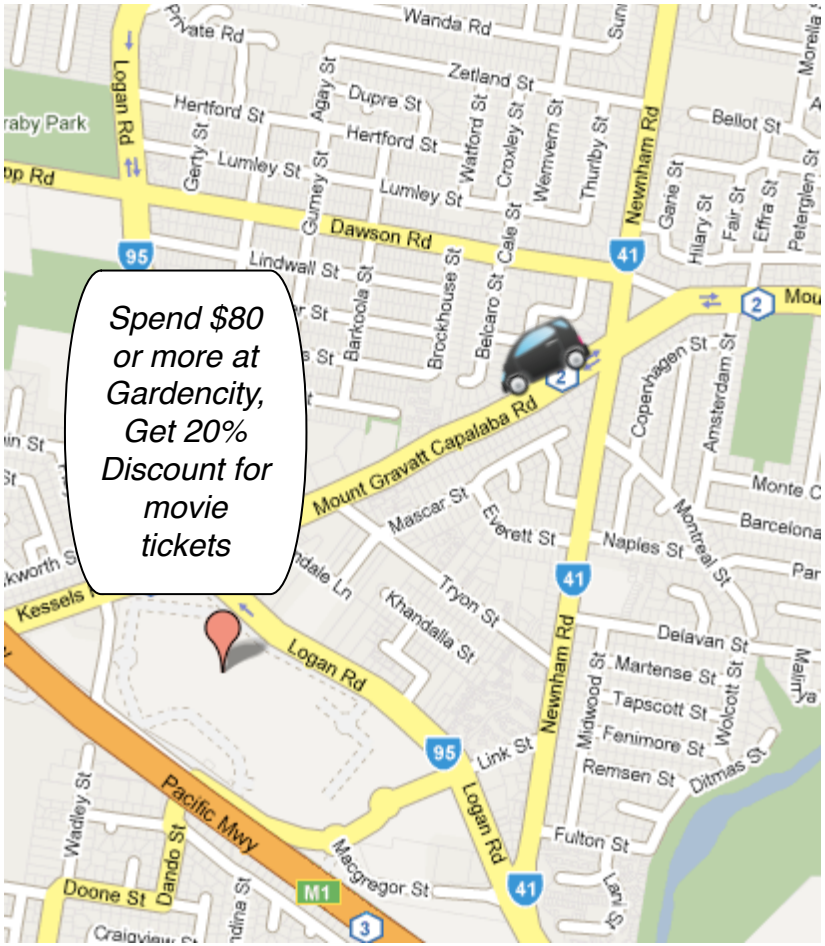
Data Mining with Activity Sequences

- Technique => Knowledge
  - Frequent Sequential Pattern => Common Behavior (e.g. Collaborative promotions)
  - Periodic Pattern => Periodic Behavior (e.g. POI suggestion)
  - Outlier Detection => Odd Behavior (e.g. Criminal Investigation)
  - Clustering => Similar Behavior (e.g. Friend suggestion)
- The Problem?
  - Facebook vs Twitter



Data Mining with Activity Sequences

- Technique => Knowledge
  - Frequent Sequential Pattern => Common Behavior (e.g. Collaborative promotions)
  - Periodic Pattern => Periodic Behavior (e.g. POI suggestion)
  - Outlier Detection => Odd Behavior (e.g. Criminal Investigation)
  - Clustering => Similar Behavior (e.g. Friend suggestion)
- The Problem?
  - Facebook vs Twitter
  - How to obtain the activity sequences?



Travel Sequences

- Guess activity from travel sequences
  - From office to a bar every workday.
  - Stay at night club for 5 hours twice a year.
  - Moving slightly faster than working speed in the park every Saturday.
  - Moving slowly in shopping center for 4 hours then stays in a cinema.
- Where can we get such travel sequences?
  - GPS-enabled mobile devices

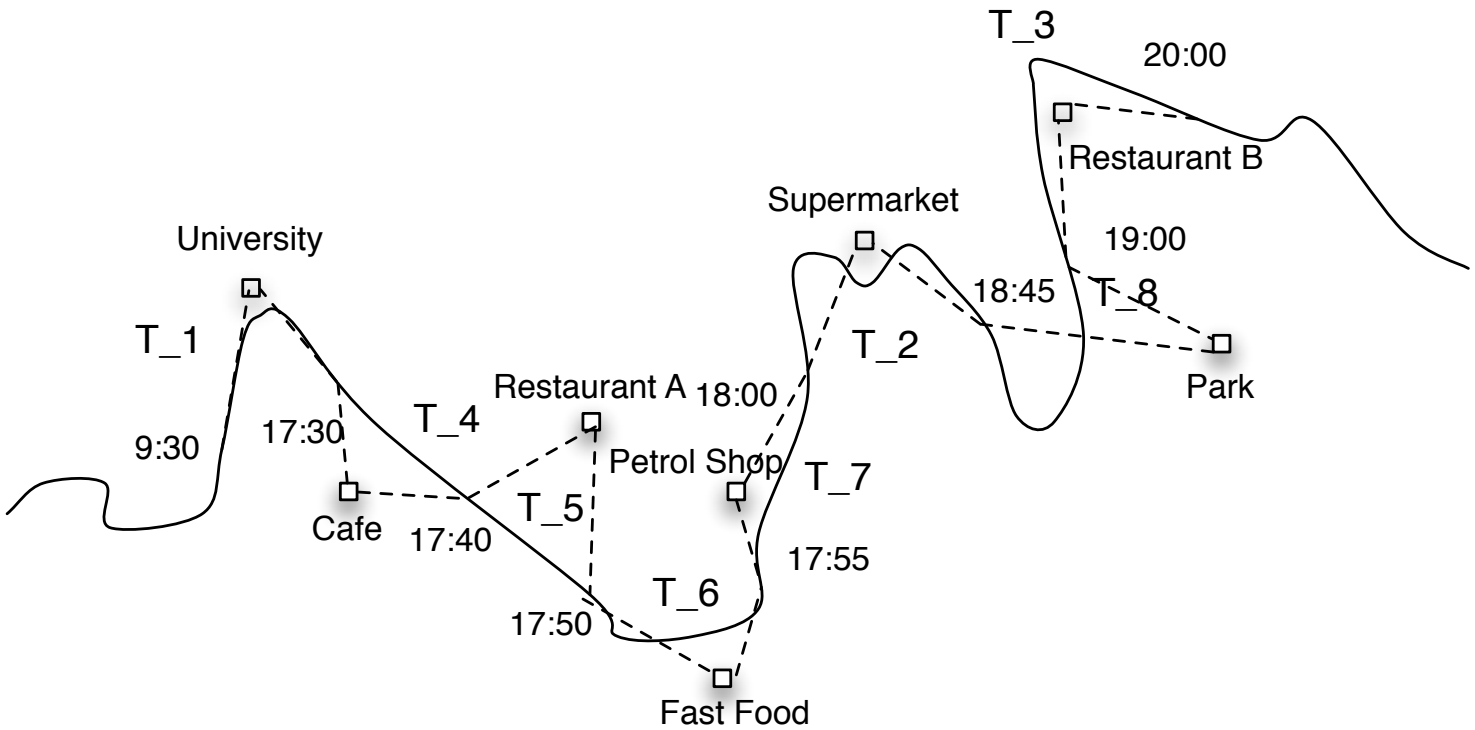
From Trajectory to Sequence of Activities

- Problem
  - Given a set of trajectories and a set of POIs, find the sequence of activities that might be performed during those set of trajectories
- Rationale
  - If the user stays at a POI for long enough time, them some activity may take place.
- Question
  - Which POIs did the user stay?
  - What activities the user performed?

An Influence Model

- Given a set of POIs $\mathcal{P}$, a segment of trajectory T is *influenced* by a $p \in \mathcal{P}$ if for every point *pt* on T, there does not exists a point $p' \in \mathcal{P}$, $dist(pt, p') < dist(pt, p)$.
- The *influence duration* of a POI *p* given a segment of trajectory T influenced by *p*, is the timestamp difference of T.

Influence Example



Study@University ⇨ Shopping@Supermarket ⇨ Dining@Restaurant_B

Trajectories to Activities using Influence Duration

- How long is long enough?
  - Different POI have different requirement
  - e.g. Fast food restaurant vs Luxury restaurant
- Which activity took place?
  - Different activity requires different time length
  - e.g. Working in restaurant vs Dining in restaurant

POI-Activity Mapping Set

A POI-Activity Mapping Set (PAMS) is a set of quadruple $M = \{(p, e, t_{min}, t_{max})\}$, where p is some POI, e is some activity, and t_{min}, t_{max} are the minimum and maximum elapsed time for user activity e happened at p.

PAMS			
	...		
	...		
	...		
Restaurant	Delivery	15 minutes	45 minutes
Restaurant	Dinning	30 minutes	2 hours
Restaurant	Working	3 hours	8 hours
University	Studying	2 hours	12 hours
University	Working	4 hours	12 hours
Petrol Shop	Fueling	10 minutes	30 minutes
Supermarket	Shopping	30 minutes	2 hours
Fast Food	Takeaway	10 minutes	45 minutes
Park	Playing	15 minutes	1 hour
	...		
	...		
	...		

Assign User Activities

- Influence duration are checked agains t_{min} and t_{max} in PAMS
- Multiple activities may happen.

E.g. Given a subtrajectory T_s influenced by POI p_i with influence duration 50, the PAMS is given as follow:

POI	Event	t_{min}	t_{max}
p_i	a	15	45
p_i	b	30	60
p_i	c	40	90

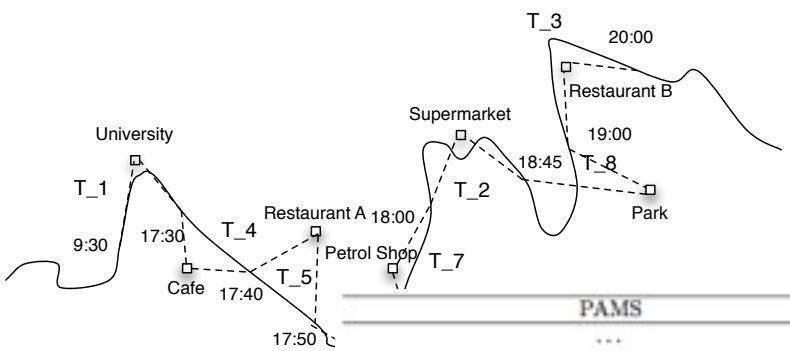
The possible activity are:

- a and b
- b
- c

Trajectory Semantic Join

- Input: A set of trajectories $\mathcal{ST}$, A set of POIs $\mathcal{P}$ and a PAMS $\mathcal{M}$
- Output: All subtrajectory-activity doubles *(T_s, e)*, such that *e* may happen at T_s

$$\left(\quad \exists_{T \in \mathcal{ST}}.T_s \in T, \text{ and } \exists_{p \in P}.E = EA(p, idr(p, T_s), M) \quad \right)$$



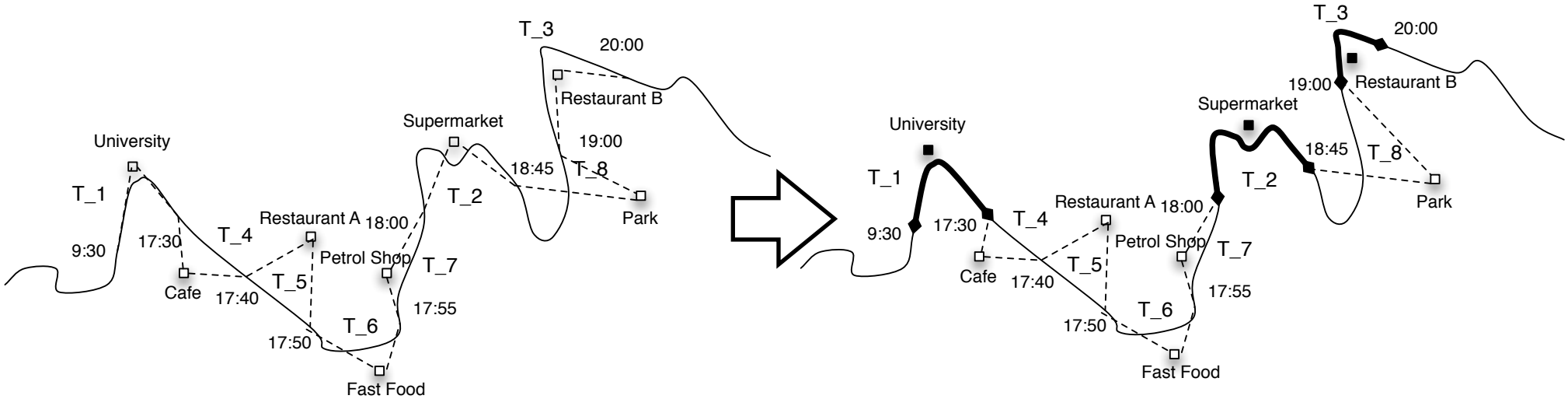
PAMS

...			
...			
...			
Restaurant	Delivery	15 minutes	45 minutes
Restaurant	Dinning	30 minutes	2 hours
Restaurant	Working	3 hours	8 hours
University	Studying	2 hours	12 hours
University	Working	4 hours	12 hours
Petrol Shop	Fueling	10 minutes	30 minutes
Supermarket	Shopping	30 minutes	2 hours
Fast Food	Takeaway	10 minutes	45 minutes
Park	Playing	15 minutes	1 hour
...			
...			
...			

(T_1, (Studying@University OR Working@University)),
(T_2, Shopping@Supermarket),
(T_3, Dining@Restaurant)

Filter and Refine Approach

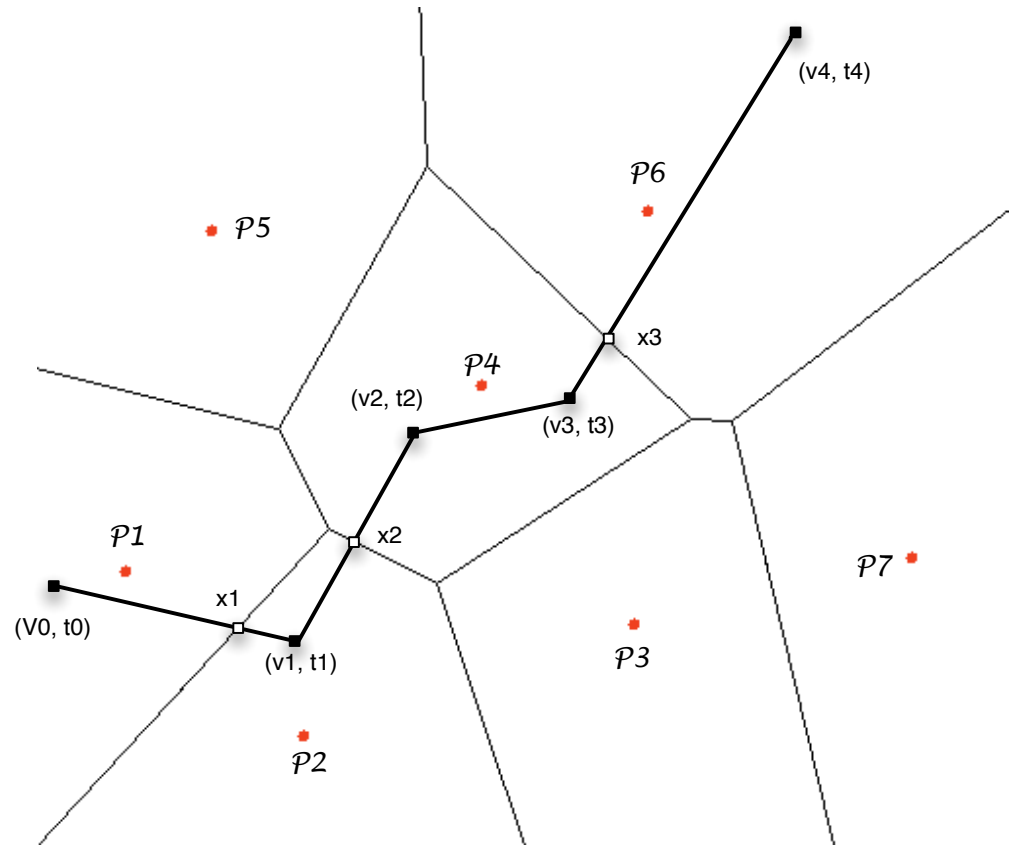
- Filter: *Trajectory-POI join* - find all subtrajectory-POI pairs that activities may took place



- Refine: assign possible events to these subtrajectory-POI pairs

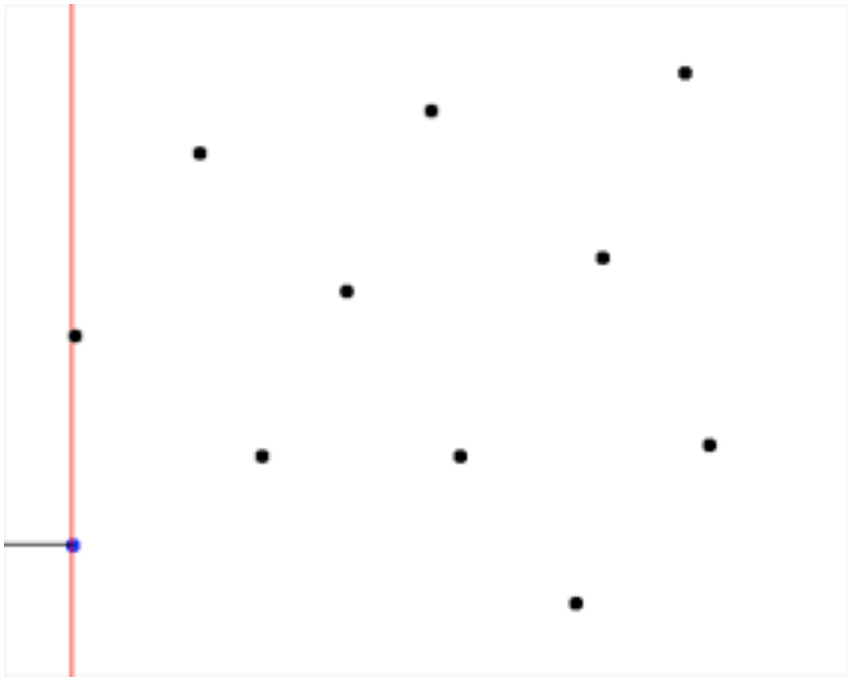
Trajectory Semantic Join using Voronoi Diagram

- Voronoi Diagram
  - A Voronoi diagram of a set of points P is the subdivision of the plain into *n* cells, one of each site in $\mathcal{P}$, with the property that a point *q* lies in the cell corresponding to a site pi if and only if $dist(p, p_i) < dist(q, p_j)$ for each $p_j \in \mathcal{P}$, with $j \neq i$.
- Computes Voronoi Diagram
  - Fortune's Algorithm to Compute Voronoi diagram in $O(n \log n)$ time.



Trajectory Semantic Join using Voronoi Diagram

- Voronoi Diagram
  - A Voronoi diagram of a set of points P is the subdivision of the plain into *n* cells, one of each site in $\mathcal{P}$, with the property that a point *q* lies in the cell corresponding to a site pi if and only if $dist(p, p_i) < dist(q, p_j)$ for each $p_j \in \mathcal{P}$, with $j \neq i$.
- Computes Voronoi Diagram
  - Fortune's Algorithm to Compute Voronoi diagram in $O(n \log n)$ time.



Trajectory POI Join Algorithm

Algorithm 1: Trajectory_POI_Join

Input: A set of trajectories ST, a set of POIs P, and its Voronoi diagram V.

Output: A set of triples (subtrajectory, POI, actual influence duration)

```
/* Step 1 - Compute line intersections                    */
```

Compute line intersection points of all trajectories in ST and all edges in V using plain-sweep technique;

```
/* Step 2 - Organize and Output                           */
```

Group the intersection points by trajectory and sort them by their timestamp;

foreach T *in* ST **do**

  foreach *consecutive intersection points* p_i *and* p_j *in* T **do**

    Find the subtrajectory T_s of T defined enclosed by p_i and p_j;

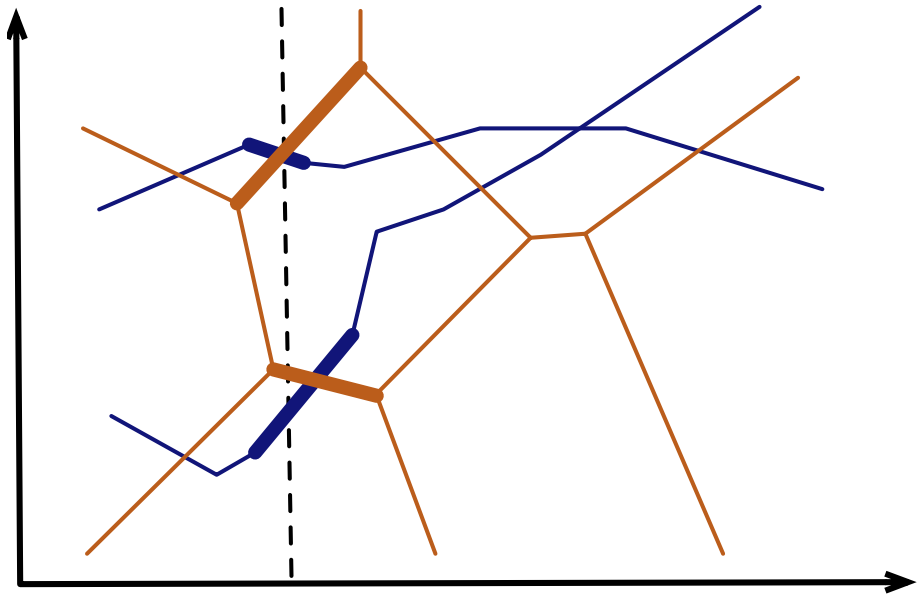
    Find common POI p that influence both p_i and p_j;

    Find the time difference d of p_i and p_j;

    Output the triple (T_s, p, d);

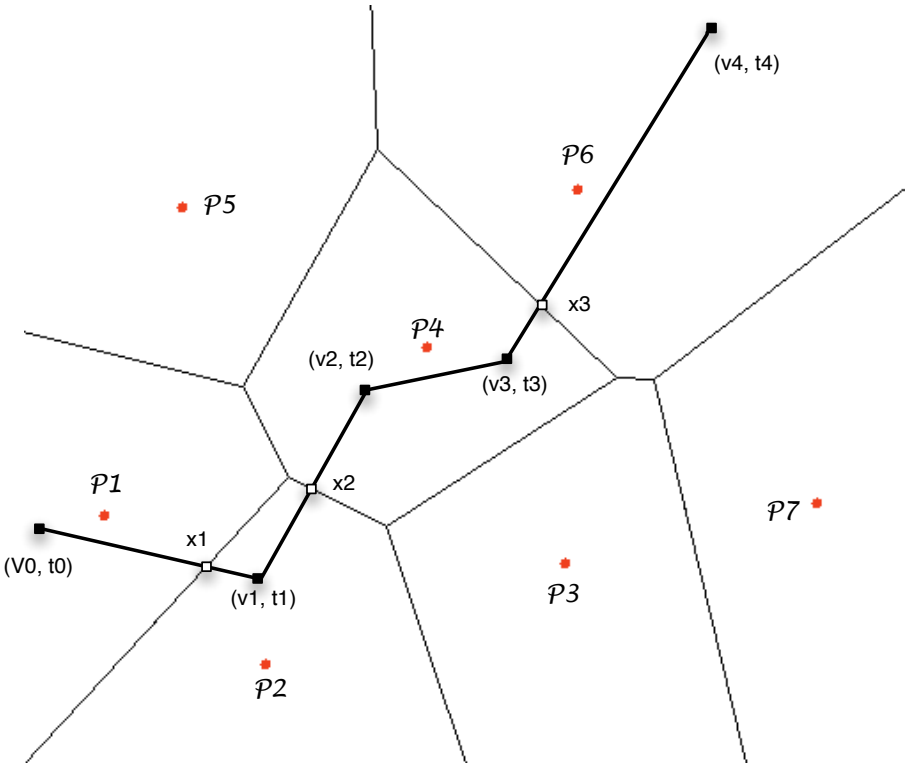
TP Join (Compute Line Intersections)

- Plane-sweep for Line Intersections
  - Line segment becomes active when intersect with the sweep line
  - Only “active” line segments can intersect with each other
  - Perform intersection check between “active” line segments



TP Join - Organize and Output

- Observation: Given a trajectory intersects Voronoi edges on a sequence of points (p_0, p_1, ..., p_n), where $\forall_{0 \le i < n}\ timestamp(p_i, T) \le timestamp(p_{i+1}, T)$. All consecutive points p_i and p_{i+1} $(0 \le i < n)$ have at least one common influenced POI influencing POI p_c, and the subtrajectory T_i of T, defined by p_i and p_{i+1} is influenced by p_c.



- Temporally sorted intersection points enclosed subtrajectories are fully influenced by some POI.

Partition Based Trajectory Semantic Join (PTSJ)

- Problem with TP Join
  - Memory based techniques
  - Need sort Voronoi edges and segment of trajectories globally
- Partition Based TSVID Join (PTSJ)
  - A modified algorithm from PBSM (Patel SIGMOD 96)
  - Partition the space into smaller size grids that spatial object can be fit into memory, insert the Key-Pointer-Element (KPE) of spatial objects into each partition if their MBR overlap with the partition.
  - Estimate number of partitions:

$$pp = \lceil \frac{(||T|| + ||V||) \times sizeof(KPE)}{M} \rceil$$

  - Perform in memory technique for each partition.
  - Repartition may be needed if not enough memory.

Partition Based Trajectory Semantic Join (PTSJ)

Algorithm 2: PTSJ

Input: A set of trajectories ST, a set of POIs P with their influence regions VC, and a PAMS L

Output: A set of TPT and event pairs

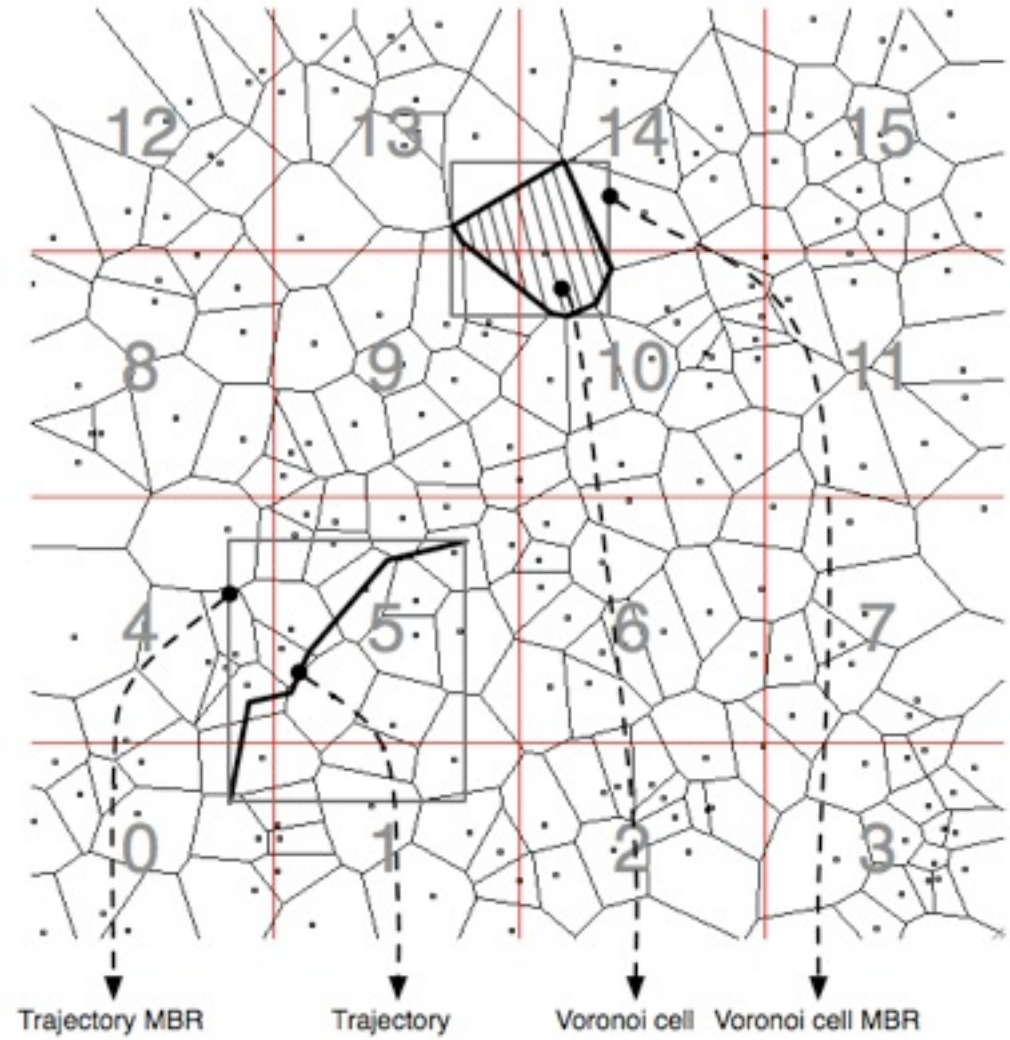
```
p ← Estimate number of partitions, if not specified in the input;
Partition table G ← Partition the space into p partitions;
foreach T ∈ ST do
    G' ← The set of partitions that T.mbr overlap with;
    Insert KPE of T into partitions in G';
foreach vc ∈ VC do
    G' ← The set of partitions that V.mbr overlap with;
    Insert KPE of vc into partitions in G';
foreach partition g ∈ G do
    Load All Voronoi cells VC' and Trajectories ST' in g into internal memory;
    TPTS ← Trajectory_POI_Join(ST', VC');
    Assign events for each TPT in TPTS and output the result;
```

Partition

TP Join for each Partition

Problems with PTSJ

- Trajectories and Voronoi cells may be inserted into multiple partition entries.
- Many duplicated entries



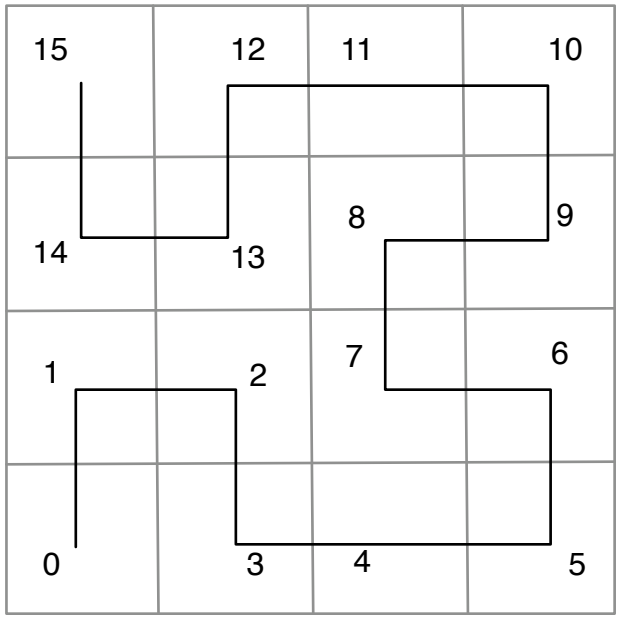
Reuse duplicated entries

- Duplication Reuse
  - Each KPE stores the partitions the spatial object is inserted in, when retrieve the spatial objects of the next partition, the spatial objects with duplicated KPE do not need to be loaded in to memory (save I/O cost).
  - Order of performing computations for the partitions are important.

Number of duplicated entries

- If KPE_o is duplicated with partition P and one of its opposite partitions P_{opp}, then it must have duplicated entires in one of its adjacent partitions.
- If KPE_o is duplicated with partition P and one of its disjoint partitions P_{dist}, then it must have duplicated entires in one of its adjacent partitions.
- Hence $N_{dup}(P, adj(P)) \geq N_{dup}(P, opp(P)) \geq N_{dup}(P, disj(P))$
- Hillbert order always traverse adjacent partitions one after the other.

Disjoint	Disjoint	Disjoint	Disjoint
Disjoint	Opposite	Adjacent	Opposite
Disjoint	Adjacent		Adjacent
Disjoint	Opposite	Adjacent	Opposite



PTSJ+ - PTSJ with duplication reuse

Algorithm 3: PTSJ+

Input: A set of trajectories ST, a set of POIs P with their influence regions VC, and a PAMS L

Output: A set of TPT and event pairs

```
p ← Estimate number of partitions if not specified in input;
Partition table G ← Partition the space into p partitions;
foreach T ∈ ST do
    G' ← The set of partitions that T.mbr overlap with;
    Insert KPE of T into partitions in G';
    Annotate each KPE with all the partitions they are inserted to;
foreach vc ∈ VC do
    G' ← The set of partitions that V.mbr overlap with;
    Insert KPE of vc into partitions in G';
    Annotate each KPE with all the partitions they are inserted to;
Sort the partitions in G using Hillbert order;
Shared Voronoi cells SSV ← ∅;
Shared Trajectories SST ← ∅;
foreach partition g ∈ G do
    VC' ← SSV ∪ load Voronoi Cells in g but not in SSV;
    SSV ← Voronoi cells in both g.Contents and g.next.Contents;
    ST' ← ∅;
    foreach KPE k ∈ g.TrajectoryContents do
        if k.id ∈ SST.ids then
            ST' ← ST' ∪ {SST.get(k.id)});
        else
            ST' ← ST' ∪ {loadFromDisk(k.id)};
    TPTS ← Trajectory_POI_Join(ST', VC');
    Assign events for each TPT in TPTS and output result;
```

- Insert KPEs of each spatial object in to each partition.
- Each KPEs include the IDs of all partitions it is inserted into.
- Partitions need to be sorted in Hillbert order.

- Do TP-Join in for each partition in Hillbert order
- Keep track of shared spatial objects
- Avoid reloading already loaded spatial objects

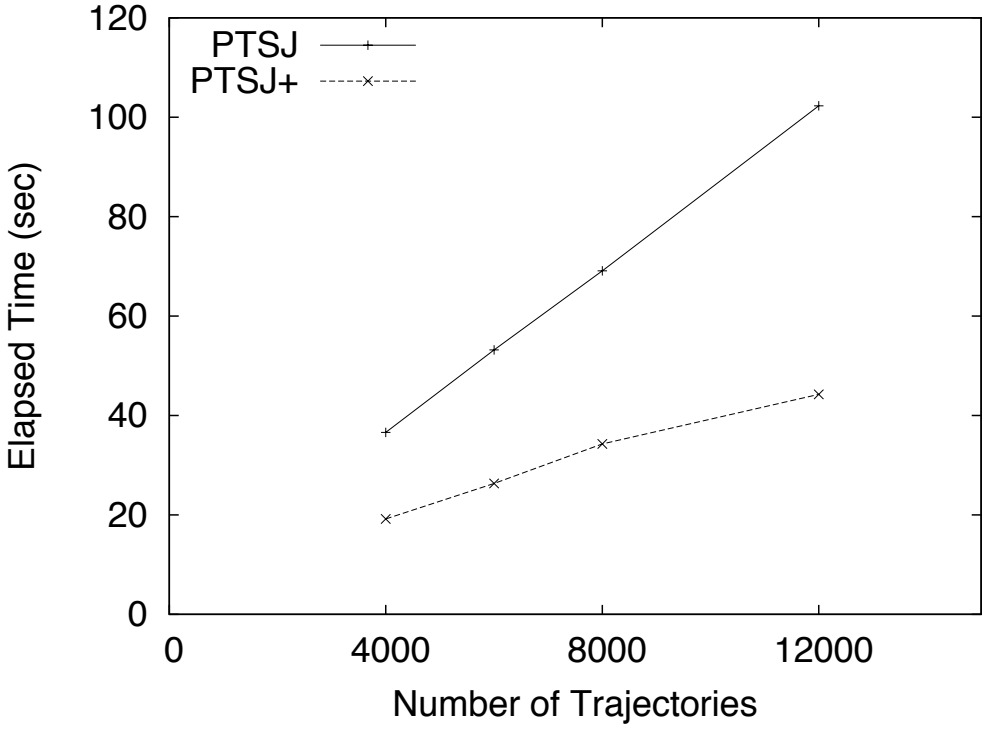
Experiments

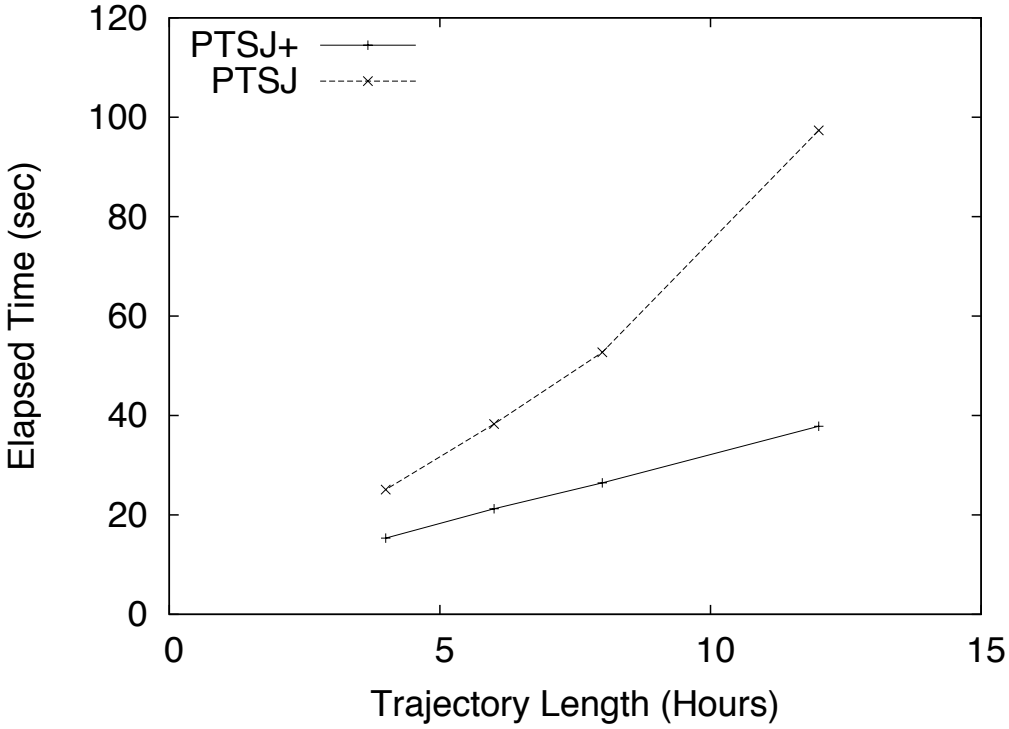
- Experimental Setting
  - POI: 6487 POIs cropped from 105725 California POIs
  - Trajectories: simulated from California road networks

	Drive	Walk	$Stop_s$	$Stop_m$	$Stop_l$
Average Speed (Km/Hour)	50	5	0.3	0.3	0.3
Switching Mode (probability)	$\frac{1}{50}$	$\frac{1}{15}$	$\frac{1}{20}$	$\frac{1}{120}$	$\frac{1}{240}$
Can Switch To (Mode)	Walk	All	Walk	Walk	Walk

  - Time consumed by building index, retrieving spatial objects (I/O) is measured.

Experimental Result





Conclusion

- Trajectory and Semantic Issues:
  - Preprocessing for Data Mining
  - Understand User Behaviours
- Future works
  - Further Optimisations
  - Privacy Issues

blummi!.com beta
... save your day.

carticipate

Buddy Beacon™

bliin YourLIVE!

bw

dodgeball.com
get started > sign in join dodgeball

Citysense™ ALPHA
Live San Francisco Nightlife Activity

brightkite®
people. places. friends.

loopt

belysio
my social navigator

buzzd

Google latitude

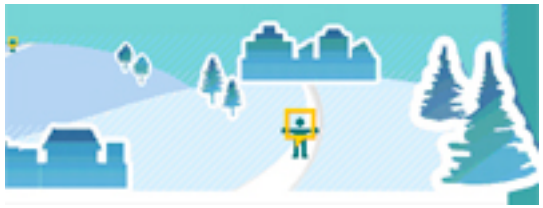
bluemapia
BETA

Microsoft GeoLife

aka'aki
the discovery of a lifetime

BuddyMob

this is where i belong
cometogethr!



Questions?