MEMO-F-524 Masters Thesis (MA-INFO)

Manipulation of GTFS periodic trajectories in MobilityDB

Student

Szymon Swirydowicz (000477108)

Supervisor

Esteban Zimányi

Year

2023 - 2024

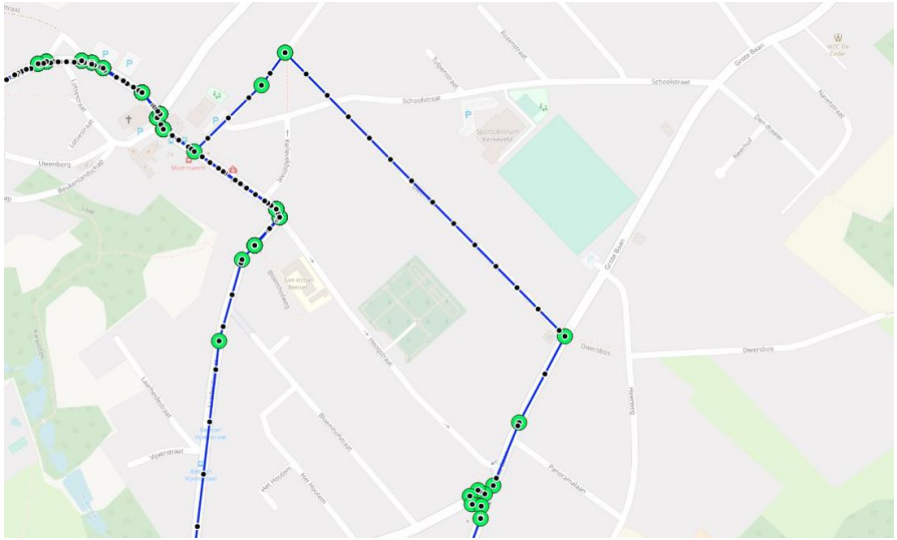
Outline

- Background
  - MobilityDB
  - GTFS
- Problem and Objectives
- Literature Overview
  - Periodic Motion
  - In Database Management Systems
  - In Data Mining
- Implementation
  - Periodic Sequences in MobilityDB
  - Operations
- Use Case: GTFS of STIB/MIVB in MobilityDB
- Conclusion and Future Work

Background

MobilityDB (1/2)

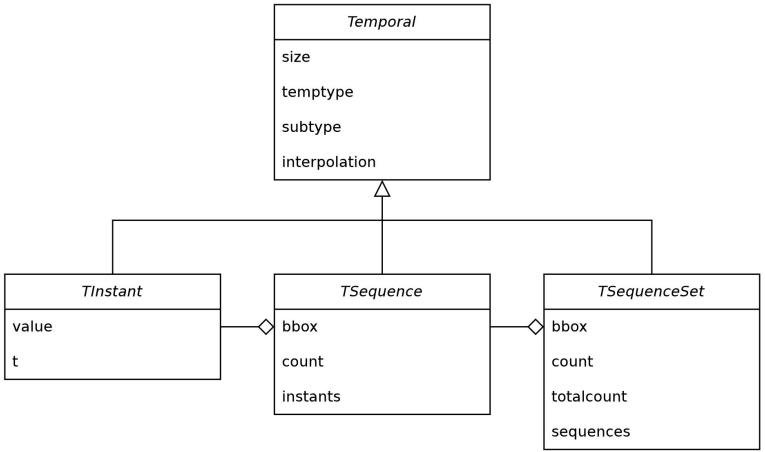
- Moving Object Database 

- PostgreSQL  + PostGIS 
- 

MobilityDB (2/2)

SELECT ttype '[v1@t1, ..., vn@tn]'

SELECT ttext '[Buyl@2024-09-04 14:00:00, ULB@2024-09-04 14:10:00]'
SELECT tgeompoint '[Point(0 0)@2024-09-04 14:00:00, Point(2 3)@2024-09-04 14:10:00]'



GTFS

- **G**eneral **T**ransit **F**eed **S**pecification
- Open Standard for Transit Information

- GTFS Schedule
  - Static Data
  - Routes & Stops
  - Schedules & Frequencies

- GTFS Realtime
  - Live Updates
  - Vehicle Locations
  - Service Alerts

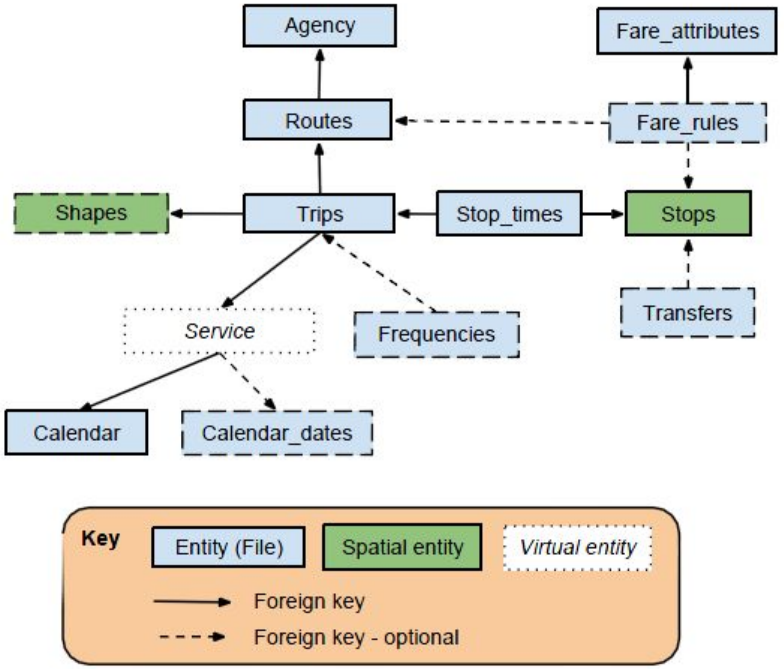


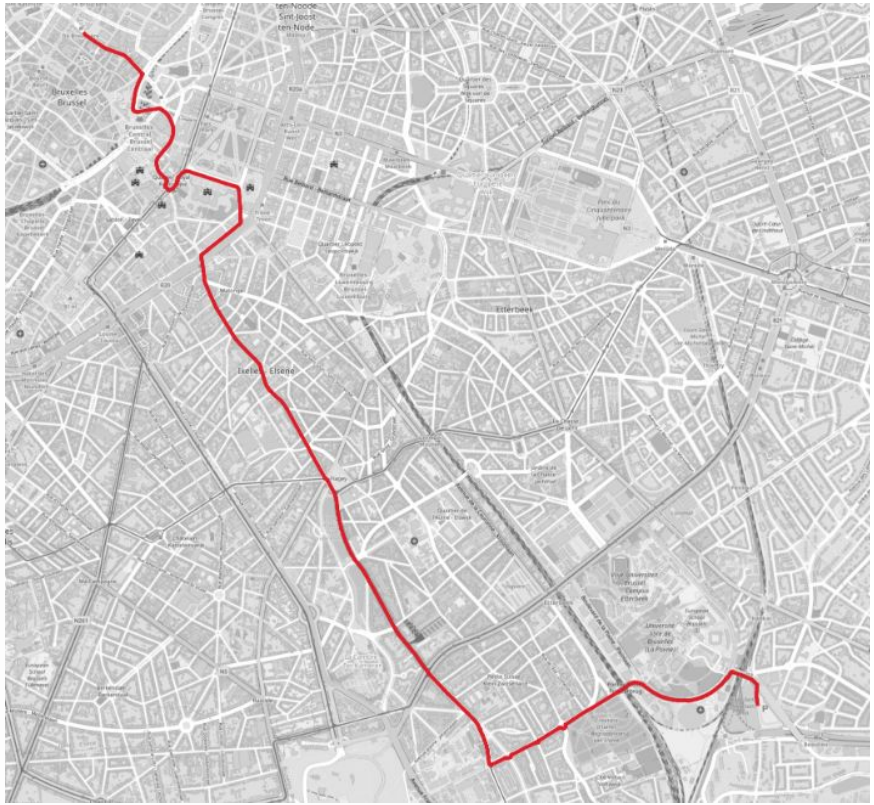
Diagram source:
http://lin-ear-th-inking.blogspot.com.au/2011/09/data-model-diagrams-for-gtfs.html

MobilityDB + GTFS

- GTFS → MobiltyDB trajectories

- service_id = 250654060
  monday = 1
  tuesday = 1
  wednesday = 1
  thursday = 1
  friday = 1
  saturday = 0
  sunday = 0
  start = 2023-03-23
  end = 2023-04-13

- Problem: **repetitions**
- Periodic Movement Data



Literature

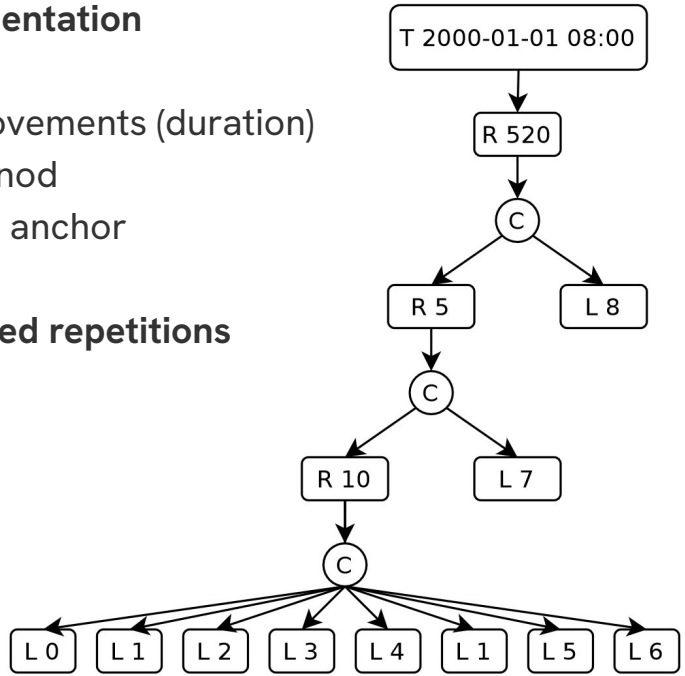
Periodic Trajectories

- **Periodic movement** = repeated in equal time intervals
- **Period** = length of a cycle

- # Periodicities
  - Strong
    - Each Monday at 08:00
  - Near
    - Once a week
  - Weak
    - Eventually in the future

Literature – In Databases (1/3)

- **Periodic tree representation**
  - **(L)** Relative movements (duration)
  - **(R)** Repetition nod
  - **(T)** Timestamp anchor
  - Supports **nested repetitions**

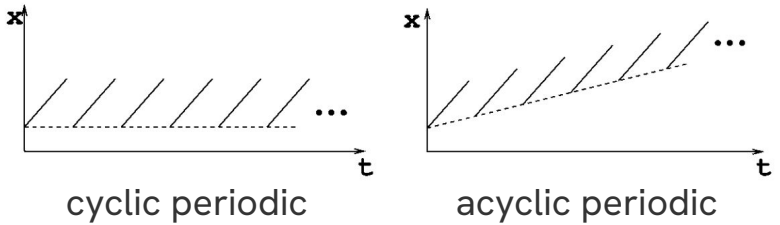


Literature – In Databases (2/3)

- **Multislices / Mixed recurrences**
  - Split in 2 parts:
    - Relative trip (spatial)
    - Repeating trip *λ* (start time rule set)
  - *λ = (yea{24}, wee{0 – 25}, day{0 – 4}, hou{7}, min{0, 25, 55})*
  - Mixed recurrence = Rule multislices \ Exceptions multislices
  - Supports **exceptions** (e.g. holidays)

Literature – In Databases (3/3)

- **Periodic Parametric Rectangles**
  - *r* = < *[5t + 9, 5t + 19], [10, 20], [0, 10]* > with *r* = < *X1, . . . , Xd, T* >
    - Speed = 5 east (positive x)
    - Initial x between 9 and 19
    - Initial y between 10 and 20
    - Between time [0, 10]
  - Periodicity with *f(t mod p)*
  - Supports **acyclic periodic movements**
    - cyclic(t) + linear(t)



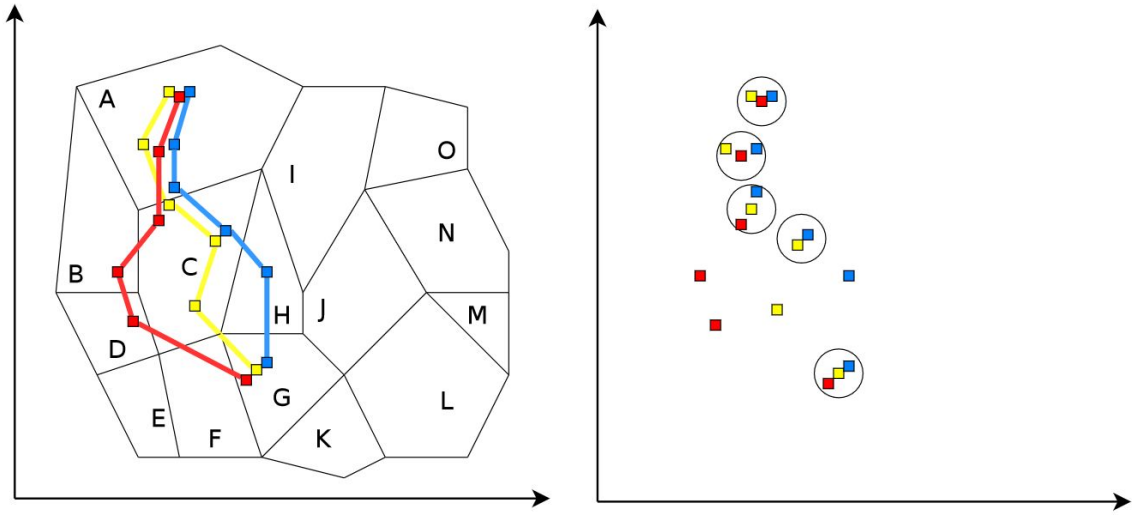
cyclic periodic acyclic periodic

Literature – In Data Mining (1/2)

- Sequence → Periodic Pattern
- Periodic Pattern Mining (PPM)
- Difficulties with GPS data:
  - Inaccuracies
  - Irregular sampling
- Detecting repetition period
  - Fourier transform
- Finding repeating patterns

Literature – In Data Mining (2/2)

- Basic idea: simplify into regions or clusters



- AACCCG AACBDG AAACHG → AA***G (3) AAC**G (2) AA*C*G (2)

Implementation

Relative Sequences

Temporal

- SELECT ttext '[A@2024-09-04 14:00:00, B@2024-09-05 14:30:00, C@2024-09-06 15:00:00]'

Relative

- SELECT ttext '[A#0, B#1 day 30 minutes, C#2 days 1 hour]'

Semi-relative

- SELECT ttext '[A#14 hours, B#1 day 14 hours 30 minutes, C#2 days 15 hours]'

Anchor

- SELECT ttext '[A#14 hours, B#1 day 14 hours 30 minutes, C#2 days 15 hours]'

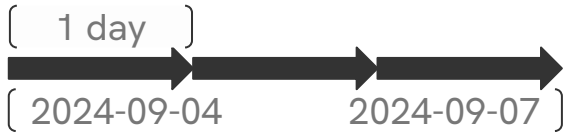
⬇ **2024-09-04**

- SELECT ttext '[A@2024-09-04 14:00:00, B@2024-09-05 14:30:00, C@2024-09-06 15:00:00]'

Periodicity

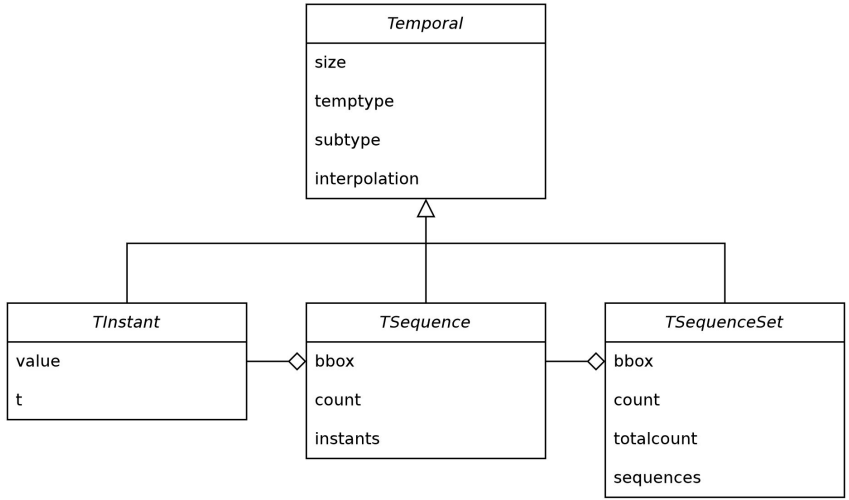
- Parameters
  - Period **:=** Interval
  - Anchor span **:=** TimestampSpan

- Others
  - Number of repetitions **:=** Integer
  - Strict pattern **:=** Boolean
  - ...



Implementation.. attempts

- How to implement periodicity?
- Inheritance?
  - Temporal = **variable-length** struct
  - Inefficient + casting problems
- New Temporal sub-type?
  - Highly increased code complexity



Implementation (1/2)

- **Temporal**
  - 2000-01-01 00:00:00 UTC
  - Flags
    - Normal = 0
    - Periodic = 1
    - ...

- **PMode** data structure
  - Additional periodic parameters

```
double tpoint_length(const Temporal *temp)
double periodic_point_length(const Temporal *temp, const PMode *pmode)
double periodic_point_length(const Temporal *temp, const Interval *period, const Span *anchor)
```

Implementation (2/2)

- **Timestamp** (Internal)
  - ’[A#2000-01-01 00:00:00, B#2000-01-01 10:00:00, ...]’
- **Interval** (Default)
  - ’[A#0 days 00:00:00, B#0 days 10:00:00, ...]’
- **Daily**
  - ’[A#00:00:00, B#10:00:00, ...]’
- **Weekly**
  - ’[A#Monday 00:00:00, B#Monday 10:00:00, ...]’

Some common issues...

- **Irregular months**
  - 28, 29, 30, 31 days

- **Time Zones**
  - Daylight saving time (**DST**) changes (Summer ⇔ Winter)
  - Use Coordinated Universal Time (**UTC**)

- ...

Operations - Categories

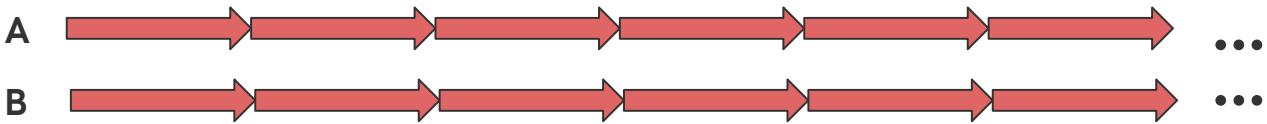
- **Basic**
  - no periodicity
  - e.g. minValue( ), trajectory( )

- **Extended**
  - modified output
  - e.g. periodic_length = length( ) x nb_of_repetitions

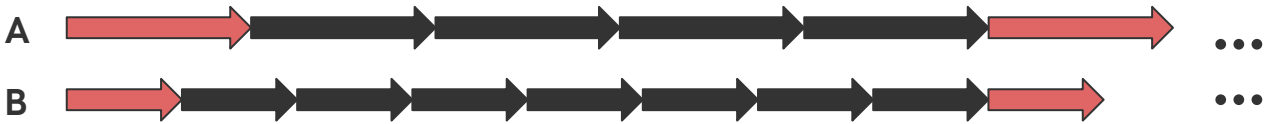
- **Advanced**
  - modified code
  - e.g. valueAt( ), tdistance( )

Operations - Periodic_TDistance

- Compute distance between trajectory points
- Same periods:



- Different periods:
  - Can be stopped when synchronised (Least Common Multiple of periods)



Operations - Anchor

- Relative sequence → Classic temporal

  [A#14 hours, B#1 day 14 hours 30 minutes, C#2 days 15 hours]
  Anchor(seq, 2024-09-04)
  [A@2024-09-04 14:00:00, B@2024-09-05 14:30:00, C@2024-09-06 15:00:00]

- Also repeats the sequence in time using
  - Period
  - Timestamp Span

Operations - PeriodicValueAt

- Value of periodic sequence at **any** chosen timestamp
- *f (timestamp modulo period)*
- As Anchor( ) it requires
  - Period
  - Timestamp Span

Case Study
STIB/MIVB GTFS

Import

- GTFS → MobilityDB
  - Imported GTFS.csv as PostgreSQL tables
  - Processed into MobilityDB trajectories
    - Group GPS coordinates into MobilityDB sequences
  - Reduced to relative trajectories

Days of the week: Monday → Friday
Start and end dates: 2023-03-24, 2023-04-16
[POINT(4.352683 50.849673)#Monday 16:14:00,
POINT(4.352794 50.849594)#Monday 16:14:04.593401,
...
POINT(4.405075 50.816745)#Monday 16:52:00]

Weekly Repetition (1/2)

2000-01-01
[POINT(4.352683 50.849673)#Monday 16:14:00,
  POINT(4.352794 50.849594)#Monday 16:14:04.593401,
  ...
  POINT(4.405075 50.816745)#Monday 16:52:00]
...
2000-01-05
[POINT(4.352683 50.849673)#Friday 16:14:00,
  POINT(4.352794 50.849594)#Friday 16:14:04.593401,
  ...
  POINT(4.405075 50.816745)#Friday 16:52:00]

Weekly Repetition (2/2)

```
SELECT anchor(
        trip, --relative_sequence
        span(c.start_date, c.end_date + '1 day'), --anchor
        '1 week', --period
        true --strict_pattern
) as anchor_trip

 FROM trips_mdb_week_shifted t
 INNER JOIN calendar c ON t.service_id = c.service_id
 WHERE trip_id = '1166219082506540 60';
```

“Daily” Repetition

```
SELECT anchor_array(
        trip, --relative_sequence
        span(c.start_date, c.end_date + ‘1 day’), --anchor
        ’1 day’, --period
        true, --strict_pattern
        ARRAY[monday, tuesday, wednesday, thursday, friday, saturday, sunday], --exception_array
        EXTRACT(DOW FROM c.start_date::timestamptz)::int + 6 % 7 --array_shift
) as anchor_trip

 FROM trips_mdb_day t
 INNER JOIN calendar c ON t.service_id = c.service_id
 WHERE trip_id = ‘1166219082506540 60’;

-- note: reduces nb of stored trips, BUT limits periodic functions
```

Identical Trips (Group)

- Same
  - transportation line
  - spatial trajectory
  - day

- … but different
  - start times + overlap
  - delays between stops
- “*duplicate trips*”

- Delays often grouped per day period (e.g. morning)
  - Compare as relative sequences + group similar
  - Store start times separately
  - Purely for storage optimization

Performances (1/2)

- Storage size reduction

Relation	Classic	Week	Daily	Group
Sequence count	506,689	197,401	68,309	13,837
Disk Size	4792 MB	1873 MB	1012 MB	146 MB
		-61%	-79%	-97%

Performances (2/2)

- Query: Quickest travel between A and B points using public transport

Query time	**Avg**	**Std**	**Min**	**Max**
periodic classic	**3.7s** 15.6s	**0.4s** 0.6s	**3.4s** 15.0s	**4.9s** 17.5s
periodic (seqscan=off) **classic** (seqscan=off)	4.1s **1.3s**	0.2s **0.05s**	3.7s **1.2s**	4.5s **1.4s**

Limitations

- Efficiency depends on repetition span (GTFS trip date coverage week buckets)

Range	1W	2W	3W	4W	≥ 5W	Total
March 2023	20.95%	5.99%	7.01%	66.05%	0%	68,309
July 2024	70.73%	17.85%	11.4%	0%	0%	123,480

- Exception dates
- GPS inaccuracies in real-data
- Assumptions + data abstraction + simplification
- ...

Future work

- Better implementation and other periodic concepts
  - Nested, Acyclic, more functions, ...
- Locale support
- NPoints
- Scheduling tools
- Periodic Pattern Mining
- Prediction and anomalies
- Study other use cases
- ...

References (1/2)

1. **[MobilityDB]** Zimanyi, E., Sakr, M., and Lesuisse, A.
MobilityDB: A mobility database based on PostgreSQL and PostGIS.
ACM Trans. Database Syst. 45, 4 (Dec. 2020), 1–42.

2. **[GTFS in MobilityDB]** Godfrid, J., Radnic, P., Vaisman, A. A., and Zimanyi, E.
Analyzing public transport in the city of Buenos Aires with MobilityDB.
Public Transport 14, 2 (June 2022), 287–321.

3. **[Strong/Near/Weak definitions]** Tuzhilin, A., and Clifford, J.
On periodicity in temporal databases.
Information Systems 20, 8 (Jan. 1995), 619–639.

4. **[Periodic Tree]** Behr, T., de Almeida, V. T., and Guting, R. H.
Representation of periodic moving objects in databases.
In Proceedings of the 14th Annual ACM International Symposium
on Advances in Geographic Information Systems (GIS '06) (New York, NY, USA, 2006),
Association for Computing Machinery, pp. 43–50.

References (2/2)

5. [**Parametric Rectangles**] Revesz, P., and Cai, M.
Efficient querying and animation of periodic spatiotemporal databases.
Annals of Mathematics and Artificial Intelligence 36, 4 (Dec. 2002), 437–457.

6. [**Multislices**] Kasperovics, R., Bohlen, M. H., and Gamper, J.
Representing public transport schedules as repeating trips.
In 2008 15th International Symposium on Temporal Representation and Reasoning (TIME ’08)
(Washington, DC, USA, 2008), IEEE Computer Society, pp. 54–58.

7. [**PPM Regions/Clusters**] Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., and Cheung, D. W.
Mining, indexing, and querying historical spatiotemporal data.
In Proceedings of the Tenth ACM SIGKDD International Conference
on Knowledge Discovery and Data Mining (KDD ’04) (New York, NY, USA, 2004),
Association for Computing Machinery, pp. 236–245.

Thank you for your attention

Q&A