Mining and Exploration of Multiple Intersecting Axis-aligned Objects

Master’s Thesis

Tilemachos Pechlivanoglou

Supervisor:
Manos Papagelis
Axis-aligned objects

1-D line segments/intervals

2-D rectangles

Regions

Multidimensional

3-D boxes/cuboids
Object intersection problem
Object intersection problem

Input:
- a set of axis aligned geometric objects

Output:
- which pairs of objects intersect
- how much
Sweep-line algorithm (1-D)
Sweep-line algorithm (2-D)

Interval tree:
Divide-and-conquer algorithm

Computationally equivalent to Sweep-Line
Multiple Intersecting Objects
Research questions

How to detect multiple intersecting objects?

What is the size of their overlap (common region)?

Where is that common region located?
The problem

Input:
- a set of regions in $\mathbb{R}^d$:

Output:
- enumeration of all intersecting sets of regions
- size and position of each common region

Sets:

<table>
<thead>
<tr>
<th>Sets</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B</td>
<td>A,B,C</td>
</tr>
<tr>
<td>A,C</td>
<td>A,B,D</td>
</tr>
<tr>
<td>A,D</td>
<td>B,C,D</td>
</tr>
<tr>
<td>B,C</td>
<td></td>
</tr>
<tr>
<td>B,D</td>
<td>A,B,C,D</td>
</tr>
<tr>
<td>C,D</td>
<td></td>
</tr>
<tr>
<td>D,E</td>
<td></td>
</tr>
</tbody>
</table>
Multiple Intersection Calculation
Common region

A set of 3 or more objects, all intersecting pair-wise with each other have a non-empty common region.

(Helly’s theorem, convex sets)
Common region size: 1-D

For a fully intersecting set I, the **common region length** $|Z|$ is:

$$|Z_I| = \text{max(start points)} - \text{min(end points)}$$

$$|Z_{ABC}| = \text{max}(a^0, b^0, c^0) - \text{min}(a^1, b^1, c^1) = a^1 - c^0$$
Common region size: 2-D, 3-D ...

For more dimensions, $|Z|$ is the **product** of the common region **lengths** in **each dimension** $|Z_d|$.
Intersection cardinality \((k)\)

The **number** of simultaneously overlapping objects in a set

\[
\begin{align*}
k_{ABCD} &= 4 \\
k_{DE} &= 2 \\
k_{AE} &= 0 \\
k_{ABCDE} &= 4
\end{align*}
\]
Sensible baseline algorithms
Naive approach

1. Compare each object with every other
2. If any 2 intersect, compare the pair’s common region with every other object
3. If any 3 intersect, compare the triplet’s common region with every other object
4. Repeat until no intersections found or no objects left

- many nested loops
- very high computational cost
Modified sweep-line approach

1. Execute **sweep-line** algorithm to find intersecting pairs
2. Get the common regions of all resulting **pairs**
3. Execute **sweep-line** on them to find **triplets, quadruplets**
4. **Repeat** until no intersections found

- better performance than naive
Limitations

- High computational cost
- Difficult to implement
- Lack of versatility
  - different implementations needed for different problems
  - hard to process/explore specific part of dataset
Our approach (SLIG)
A **graph** data structure where:

- Each **vertex** corresponds to an **object**
- An **edge** exists between two vertices if the corresponding objects **intersect**
Clique

Subset of vertices where every two are connected (i.e. a fully connected subgraph)

size-3 cliques: ABC, ABD, ACD, BCA
size-4 clique: ABCD (maximal clique)
Observation

On an intersection graph, a **clique** corresponds to a **fully intersecting set** with a **common region**.
Sweep-Line with Intersection Graph (SLIG)

1. Execute **sweep-line** algorithm to find intersecting pairs
2. Use **pairs** to construct the intersection graph
3. Execute a **clique enumeration** algorithm on graph

- best performance
- using established, efficient clique enumeration methods
- much easier to implement
The intersection graph provides **additional mining options**, such as exploration using **queries**:

- **Single Region Query**: given an object find all other objects intersecting with it
- **Multiple Region Query**: given a set of objects, find all intersections occurring in the set
Multiple Intersections Evaluation
Randomly generated objects

1-D intervals

1-D intersection graph

2-D rectangles

2-D intersection graph
Intersection graph size

The diagram illustrates the relationship between the number of regions and the number of edges in intersection graphs of 1-D, 2-D, and 3-D objects. The x-axis represents the number of regions, ranging from $10^1$ to $10^5$. The y-axis represents the number of edges, ranging from $10^2$ to $10^8$. The graph shows that the number of edges increases significantly with the number of regions, with 1-D objects having the highest number of edges, followed by 2-D and then 3-D objects.
Performance of SLIG

SLIG scales much better than baseline
Effect of graph topology

- smaller/sparser objects -> sparser graphs -> faster execution
SLIG query performance

![Graph showing SLIG query performance](image)
Real-world data

Overlapping areas of extreme weather in CA & NV, USA
Node Importance in Trajectory Networks
Trajectories of moving objects
Trajectory Mining

Trajectory similarity

Trajectory clustering

Trajectory anomaly detection

Trajectory pattern mining

Trajectory classification

...more
Node Importance
Node importance (or centrality)

Degree centrality

Betweenness centrality

Closeness centrality

Eigenvector centrality
Over time

Node degree over time

Triangles over time

Connected components over time
(connectedness)
Applications

- Infection spreading
- Wireless signal security
- Rich dynamic network analytics
Proximity networks
Distance can represent

- line of sight
- Wifi signal range
- travel distance in a day
Trajectory networks
Problem difficulty

Node importance algorithms for static networks

Sequence of static networks (snapshots)

One large network per discrete time unit!
Node Importance in Trajectory Networks
Naive approach
For *every* discrete time unit:

1. get static **snapshot** of network

2. run static node importance **algorithms** on snapshot

**Aggregate** results at the end
Streaming approach

Similar to naive, but:

- no final aggregation
- results calculated iteratively at every step

Still every time unit
Every discrete time unit
Sweep Line Over Trajectories (SLOT)

(algorithm sketch)

represent TN edges as time intervals

apply variation of sweep line algorithm

simultaneously compute node degree, triangle membership, connected components in one pass
Edges as time intervals...

e_1: (n_1, n_2)

e_n

time
Sweep Line Over Trajectories (SLOT)
At every edge start

- **Degree**
  - nodes $u, v$ now connected
  - increment $u, v$ degree

- **Triangles**
  - did a triangle just form?
  - look for $u, v$ common neighbors
  - increment triangle $(u,v,\text{common})$

- **Components**
  - did two previously unconnected components connect?
  - compare old components of $u, v$
  - if not same, merge them
At every edge stop

- **Degree**
  - nodes $u, v$ now disconnected
  - decrement $u, v$ degree

- **Triangles**
  - did a triangle just break?
  - look for $u, v$ common neighbors
  - decrement triangle $(u,v,\text{common})$

- **Components**
  - did a component separate?
  - BFS to see if $u, v$ still connected
  - if not, split component to two
**SLOT:** At the end of the algorithm...

**Rich analytics**
- node degrees: start/end time, duration
- triangles: start/end time, duration
- connected components: start/end time, duration

**Exact** results (not approximations)
Evaluation of SLOT
Simulating trajectories

constant velocity

random velocity
Degree

![Chart showing running time vs. number of objects for Naive, Streaming, and SLOT methods. The Naive method has the highest running time, followed by Streaming, and SLOT has the lowest running time.]
SLOT performance (triangles, connectedness)
with max=0.15, min=0
Seagull migration trajectories
Summary
Multiple Intersections

Axis-aligned object intersections

Sweep-line algorithm

SLIG properties:
- Fast & efficient
- Exact
- Query capabilities

Region intersection graph
Node importance in TNs

Trajectory networks

Network Importance over time

SLOT properties:
- Fast
- Exact
- Scalable

SLOT algorithm
Contributions

- Fast and Accurate Mining of Node Importance in Trajectory Networks
  - IEEE International Conference on Big Data, 2018

- Efficient Mining and Exploration of Multiple Axis-aligned Intersecting Objects
  - Pending review in IEEE International Conference on Data Mining, 2019

- Working on extensions/applications of shown concepts
  - Data visualization
  - Location-aware computation offloading
  - Distributed versions of algorithms

- Industry collaboration project with Fortran Traffic
Thank you!