

Continuous Nearest Neighbour Monitoring in Road Networks



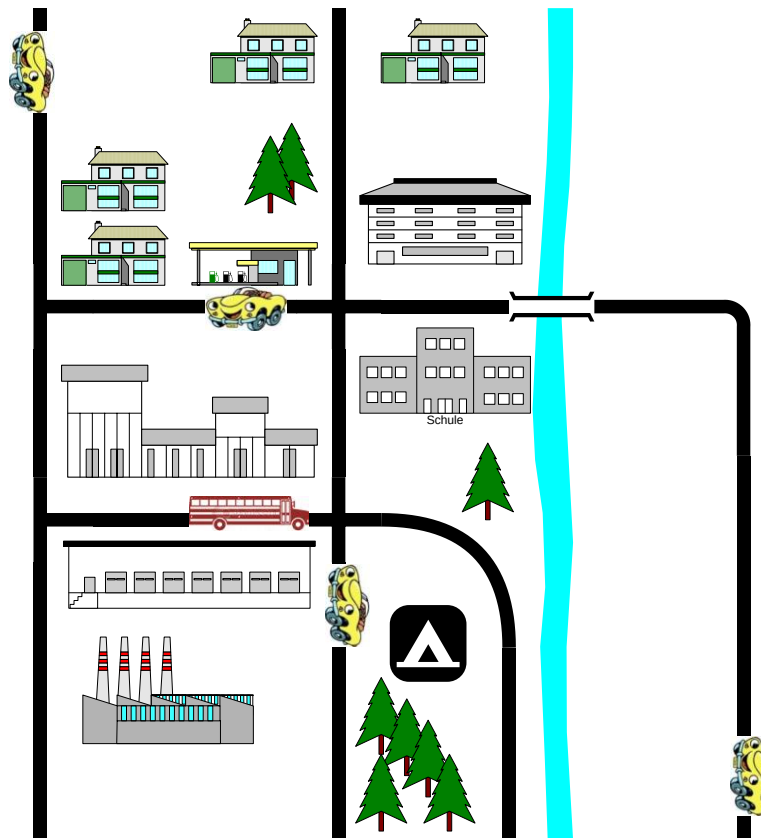
Overview

1. Introduction
2. Background
3. Data Structures
4. Incremental Monitoring Algorithm (IMA)
5. Conclusion

Introduction



Example: List of the 2 closest taxis displayed in a bus



Introduction



Challenges:

- Moving Objects and queries
- Road with variable conditions
- “Real-Time” Monitoring

→ Nearest neighbour resolution

Introduction



Goals

- Central server that monitors the positions of *CkNN queries and objects, as well as the* current edge weights, which may fluctuate over time.
- Fast algorithms to perform continuously computed results of each query.

Overview

1. Introduction
2. Background
3. Data Structures
4. Incremental Monitoring Algorithm (IMA)
5. Conclusion

Background



Finding out the k nearest neighbours is a fundamental problem in the field of spatial databases.

Previous work:

- Snapshot kNN-queries over static data
 - Restricted techniques for CkNN monitoring in Euclidean Spaces
 - Inapplicable to road networks
- First work on continuous kNN monitoring with high dynamism in edge weights, object and query movement

Overview

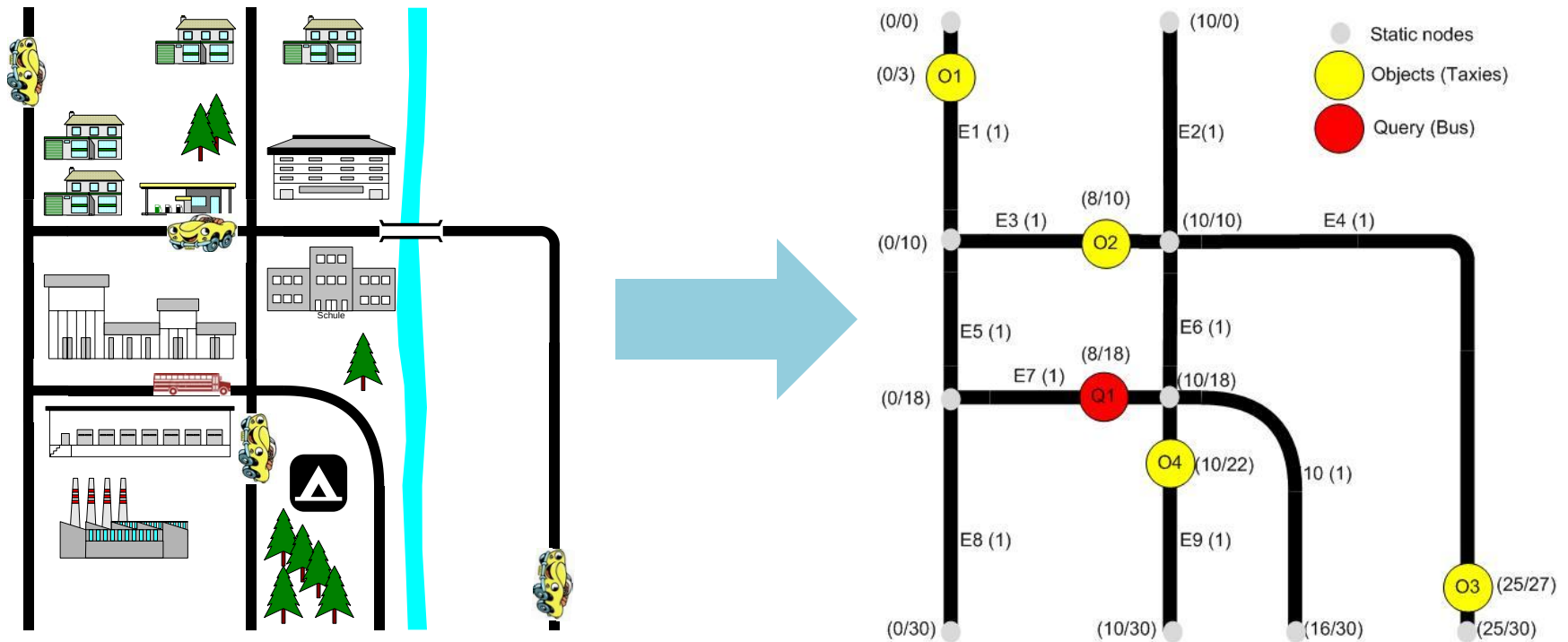
1. Introduction
2. Background
3. Data Structures
4. Incremental Monitoring Algorithm (IMA)
5. Conclusion

Data Structures



1. Spatial index SI: Efficient retrieval of edges
 - Quadtree, top-down retrieval
2. Edge table ET: Maintenance of network and data object information
 - Hash-table on edge id
 - Endpoints, sets of adjacent edges to each endpoint
 - Weights
 - List of objects on edge
 - influence list with affected queries
3. Query table QT: Maintenance of query information
 - Coordinates
 - #required NNs
 - current result of kNN set
 - Expansion tree: shortest path to k nearest neighbours

Data Structures

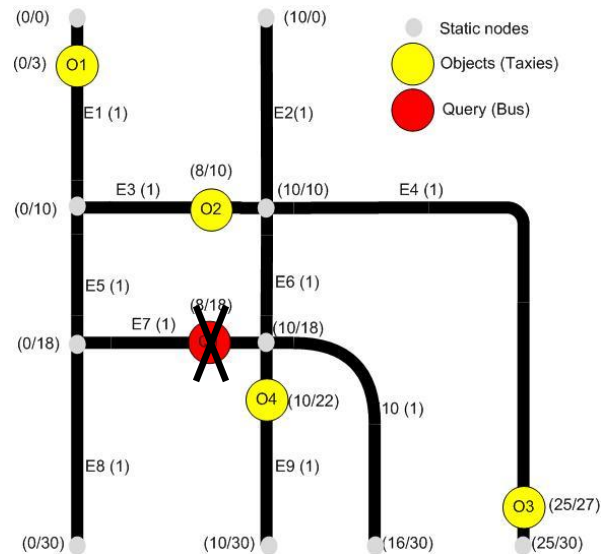


Data Structures



Edge table examples before initial result computation of Q1:

Edge ID	Start	End	AdjEdgesStart	AdjEdgesEnd	Weight	Objects	Influence List
E1	(0/0)	(0/10)	{}	{E3,E5}	1	{O1(0/3)}	{}
...	...	...	...	...	...	...	...
E5	(0/10)	(0/18)	{E1,E3}	{E7,E8}	1	{}	{}
E6	(10/10)	(10/18)	{E2, E3, E4},	{E7, E9, E10}	1	{}	{}
...	...	...	...	...	...	...	...



Overview

1. Introduction
2. Background
3. Data Structures
4. Incremental Monitoring Algorithm (IMA)
5. Conclusion

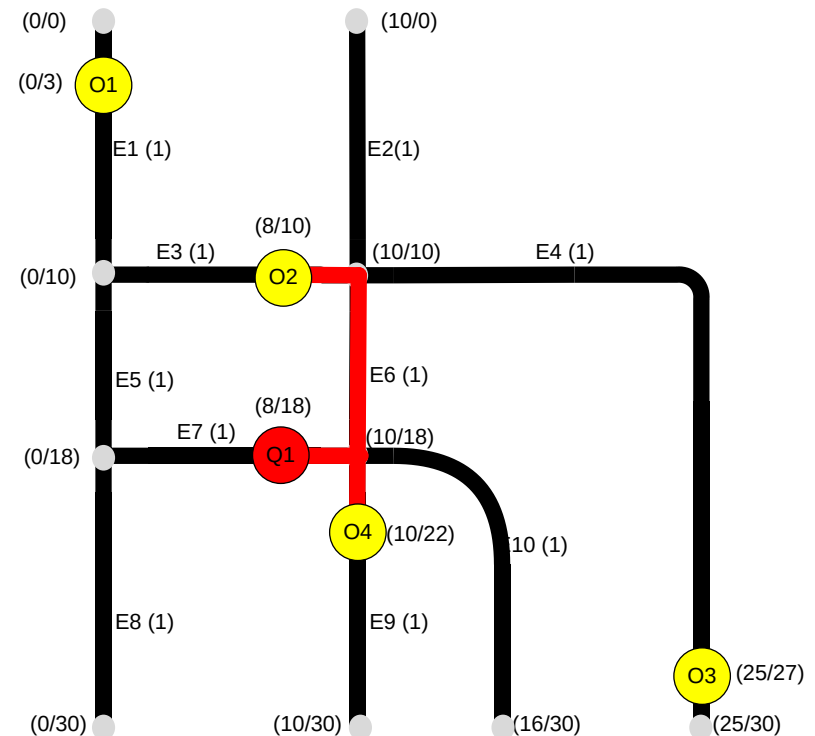
Incremental Monitoring Algorithm



Algorithm consists of:

- Initial result computation
- Monitoring of query results that can be changed due to:
 - Object movements
 - Query movements
 - Edge weight changes

In example: $k = 2$



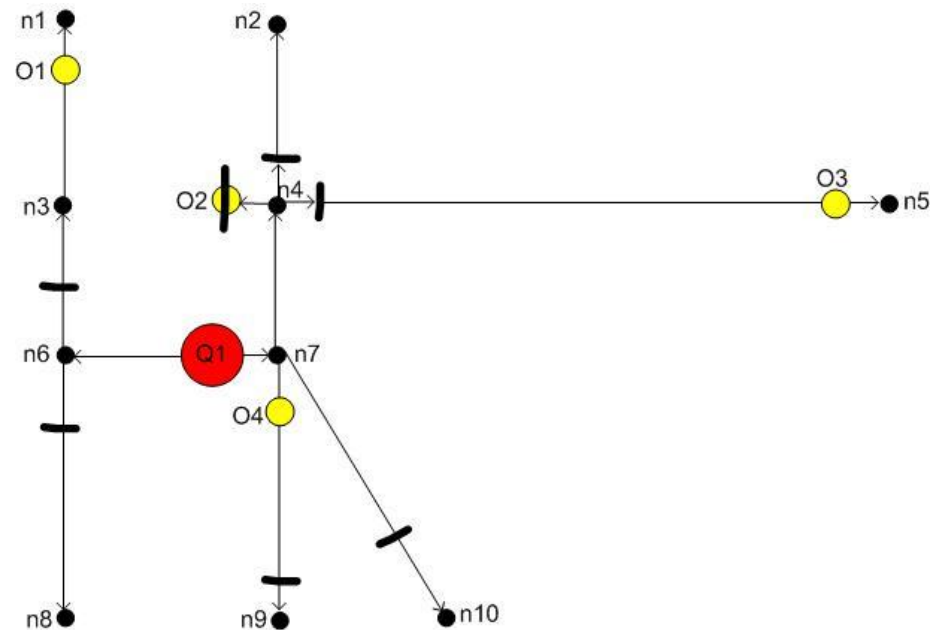
Incremental Monitoring Algorithm

Initial result computation



Initial Result computation

- Initial computation based on Dijkstra algorithm
- Expanding network around query q
- “From scratch”



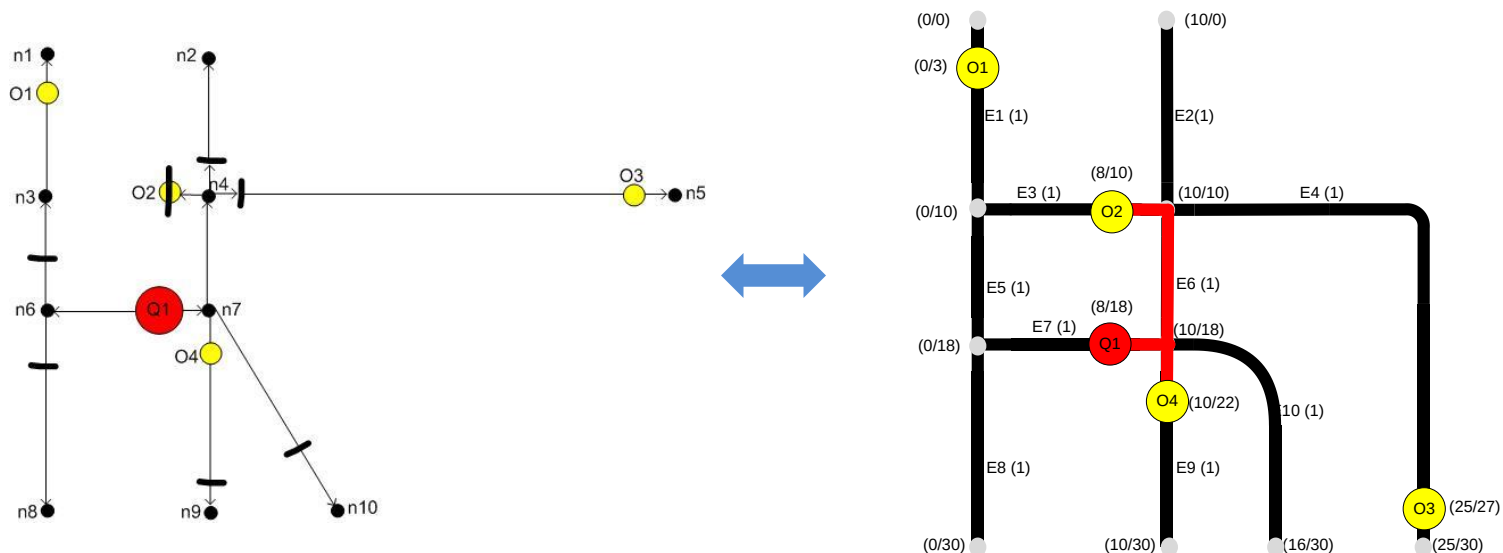
Incremental Monitoring Algorithm

Initial result computation



Edge table examples after initial result computation of Q1:

Edge ID	Start	End	AdjEdgesStart	AdjEdgesEnd	Weight	Objects	Influence List
E1	(0/0)	(0/10)	{}	{E3,E5}	1	{O1(0/3)}	{}
...	...	...	...	...	...	...	...
E5	(0/10)	(0/18)	{E1,E3}	{E7,E8}	1	{}	{Q1}
E6	(10/10)	(10/18)	{E2, E3, E4},	{E7, E9, E10}	1	{}	{Q1}
...	...	...	...	...	...	...	...

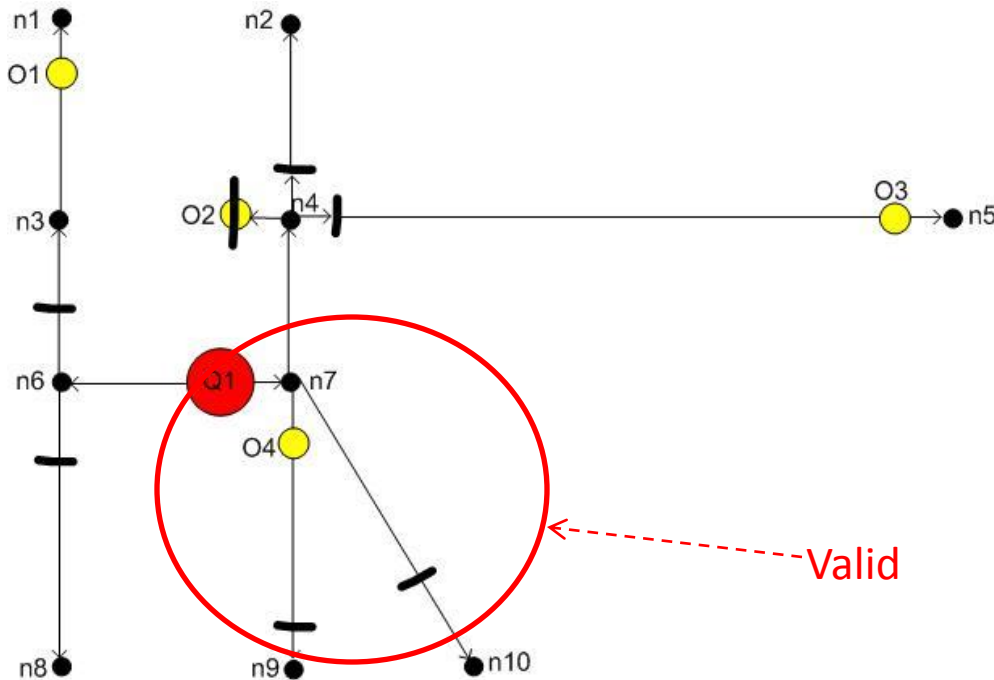


Incremental Monitoring Algorithm

Invalidation



Example of valid/ invalid parts of expansion tree:



Incremental Monitoring Algorithm

Object Update

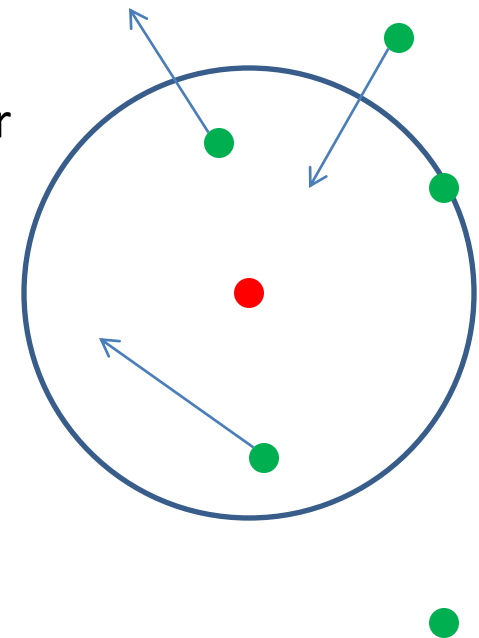


Object updates

- Moving Objects influencing a query:
 - In between the furthest distance of a neighbour
 - Incoming objects
 - Outgoing object

Query re-computation:

- Outgoing > Incoming $\rightarrow$ Expand from marks
- Incoming > Outgoing $\rightarrow$ k best of current set

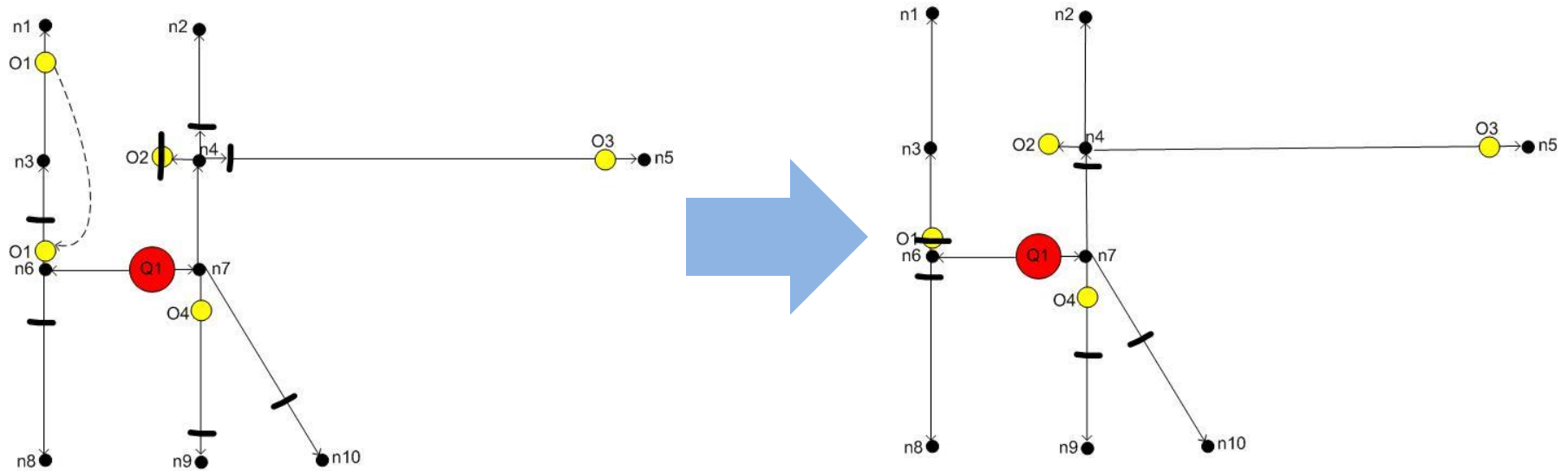


Incremental Monitoring Algorithm

Object Update



Object update



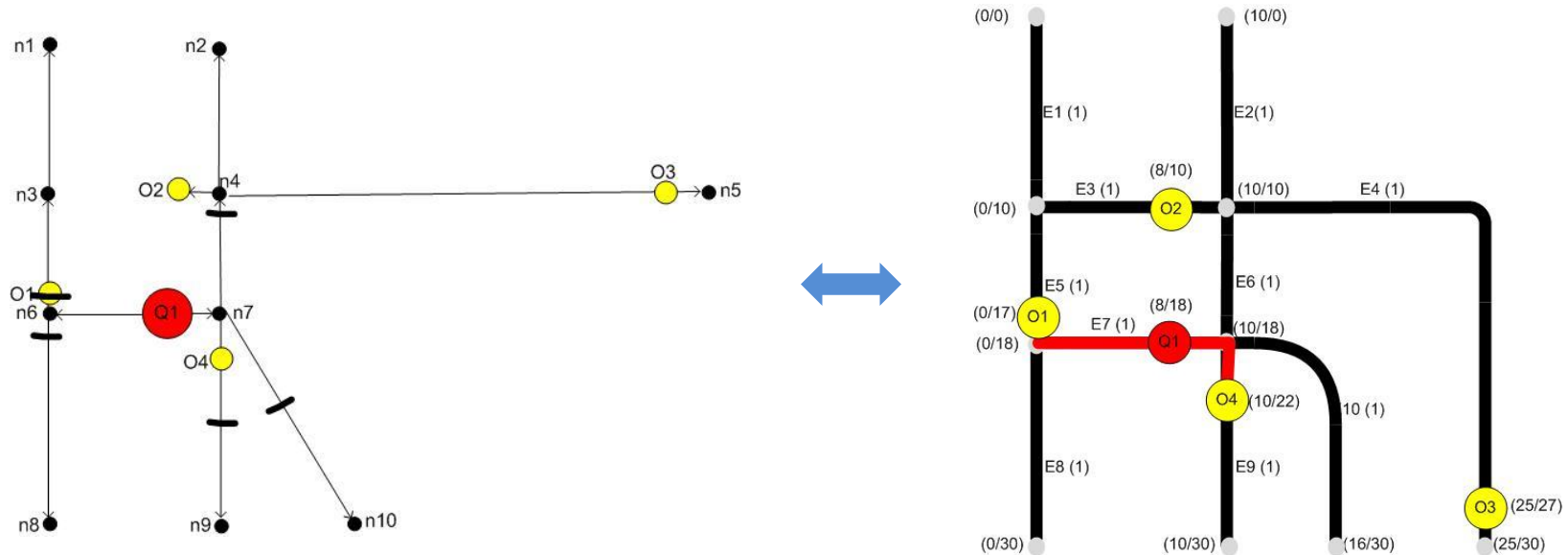
Incremental Monitoring Algorithm

Object Update



Edge table examples after object update:

Edge ID	Start	End	AdjEdgesStart	AdjEdgesEnd	Weight	Objects	Influence List
E1	(0/0)	(0/10)	{}	{E3,E5}	1	{}	{}
...	...	...	...	...	...	...	...
E5	(0/10)	(0/18)	{E1,E3}	{E7,E8}	1	{O1(0/3)}	{Q1}
E6	(10/10)	(10/18)	{E2, E3, E4},	{E7, E9, E10}	1	{}	{Q1}
...	...	...	...	...	...	...	...



Incremental Monitoring Algorithm

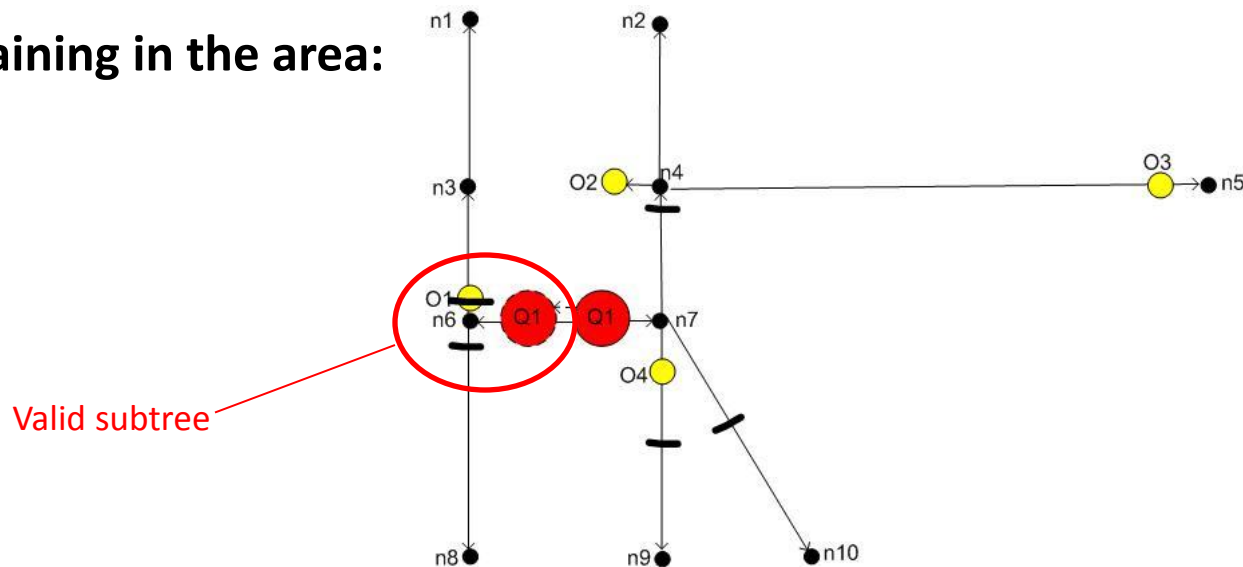
Query Update



Query updates

- Movement of query = Re-computation of the result
 - Out of the area → Re-computation from scratch
 - Remaining in the area → Sub-Tree remains valid
 - Re-computation with the use of the valid part (expanding)

Case of remaining in the area:

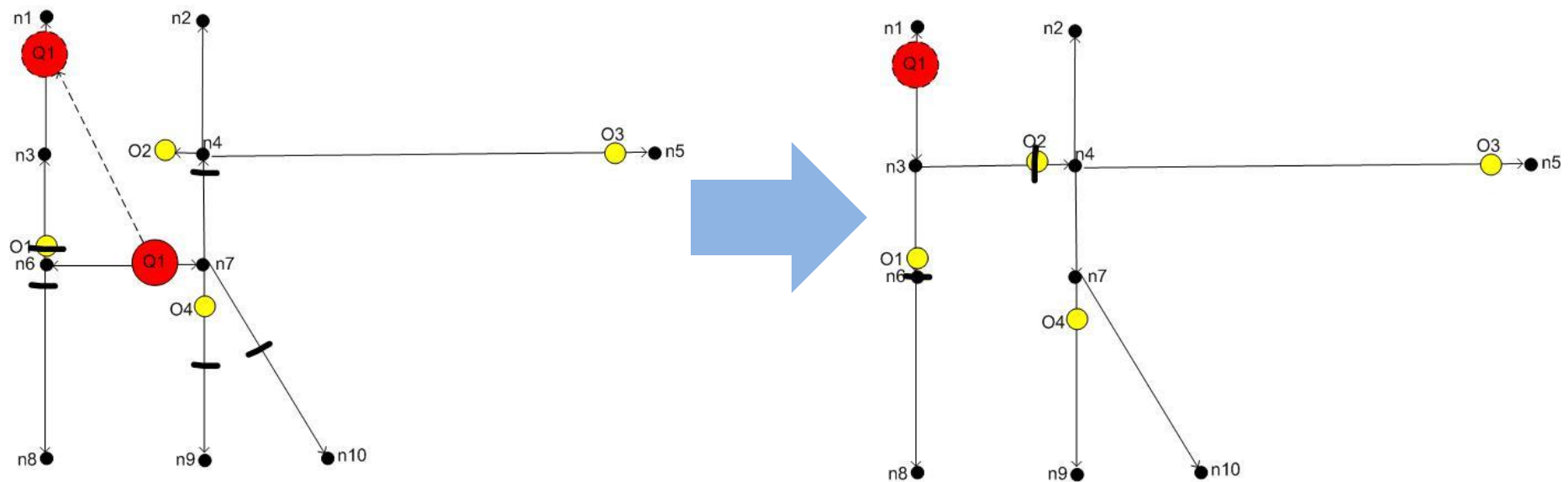


Incremental Monitoring Algorithm

Query Update



Query update:



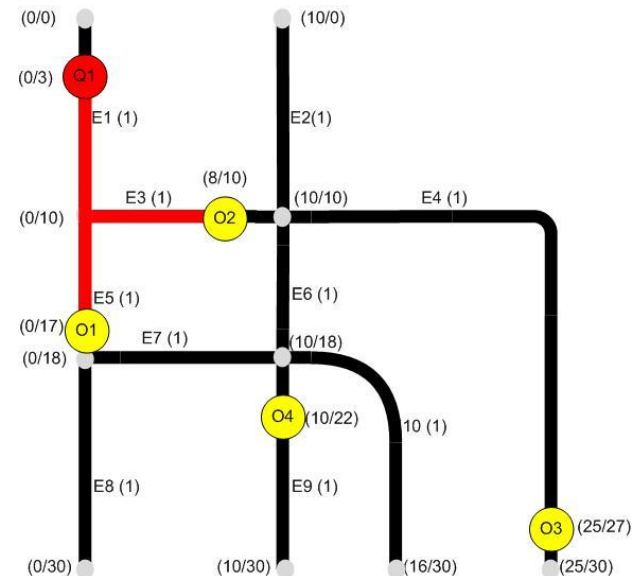
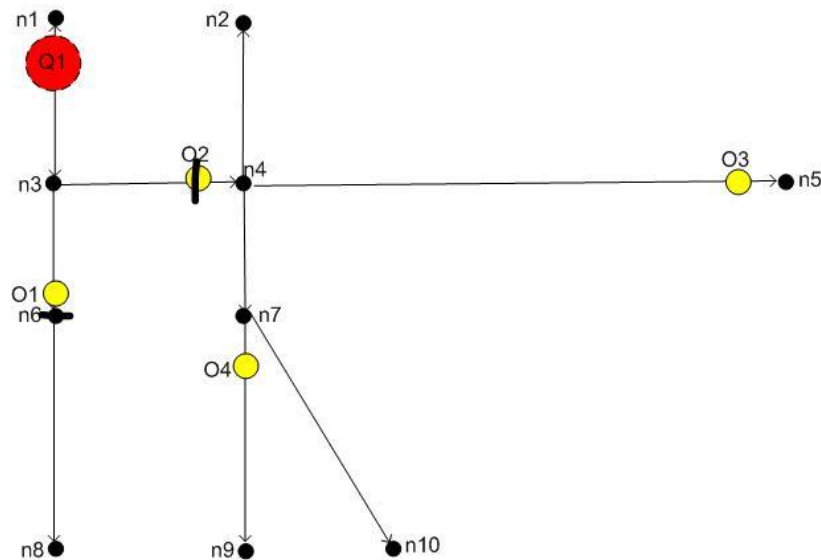
Incremental Monitoring Algorithm

Query Update



Edge table examples after query update:

Edge ID	Start	End	AdjEdgesStart	AdjEdgesEnd	Weight	Objects	Influence List
E1	(0/0)	(0/10)	{}	{E3,E5}	1	{}	{Q1}
...	...	...	...	...	...	...	...
E5	(0/10)	(0/18)	{E1,E3}	{E7,E8}	1	{O1(0/3)}	{Q1}
E6	(10/10)	(10/18)	{E2, E3, E4},	{E7, E9, E10}	1	{}	{}
...	...	...	...	...	...	...	...



Incremental Monitoring Algorithm

Edge Update



Edge updates

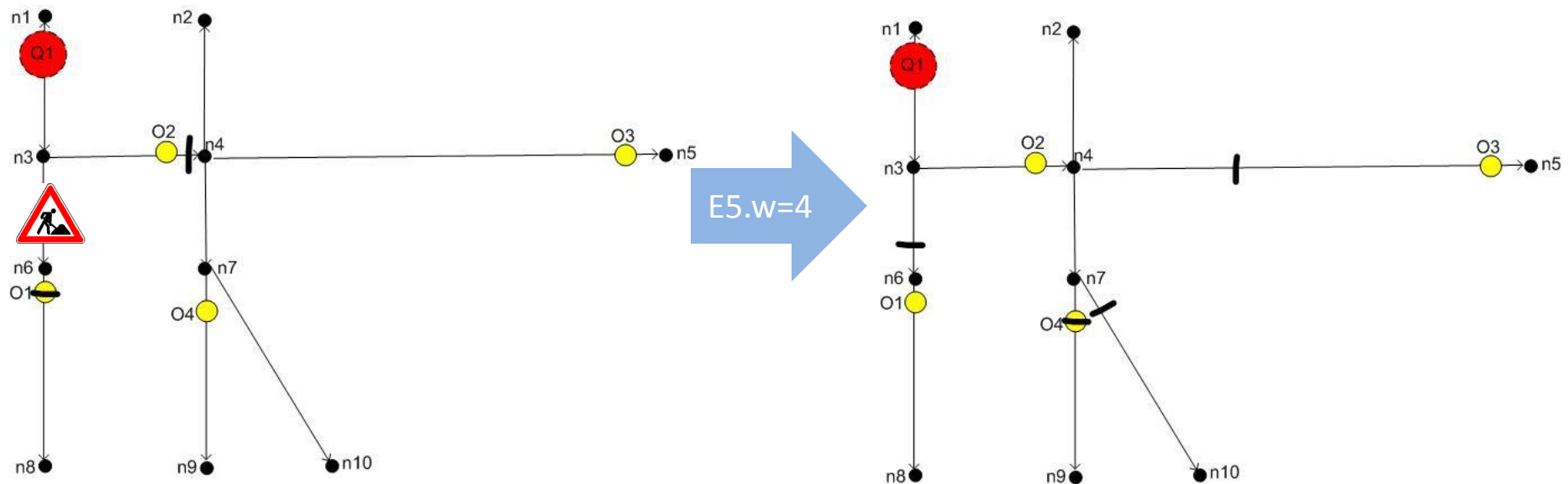
- Weights may be updated due to traffic condition changes
 - Increasing weight = subtree invalid
 - Decreasing weight = subtree only valid
- Again: re-computation of expansion tree uses the knowledge of the remaining valid part of the 'old' expansion tree

Incremental Monitoring Algorithm

Edge Update



Edge weight change



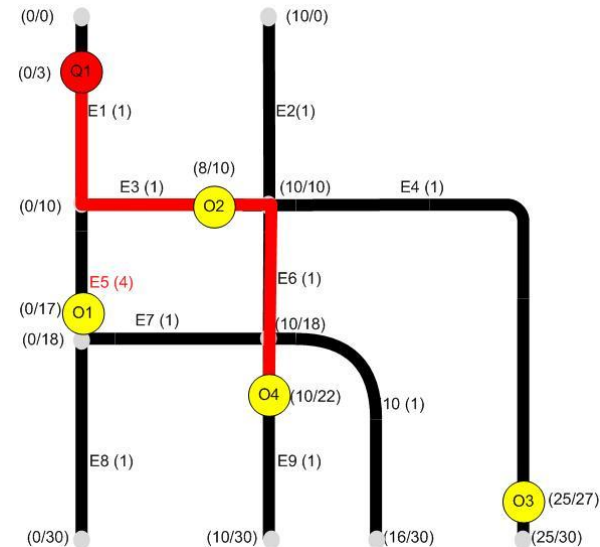
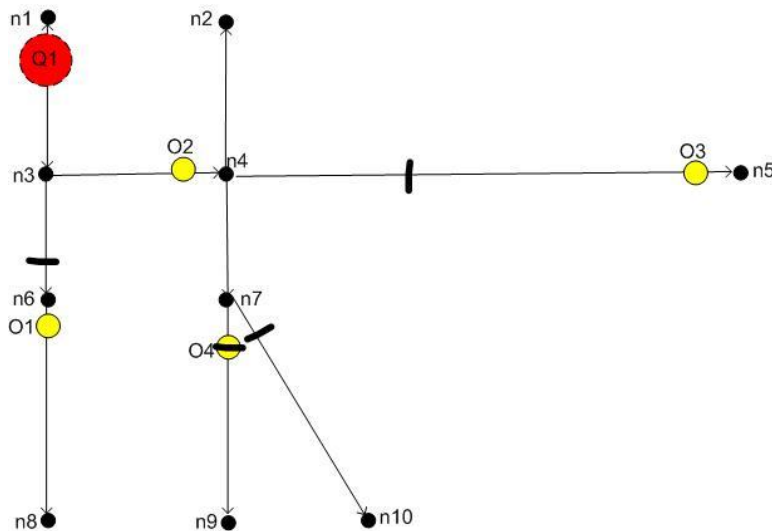
Incremental Monitoring Algorithm

Edge Update



Edge table examples after edge update:

Edge ID	Start	End	AdjEdgesStart	AdjEdgesEnd	Weight	Objects	Influence List
E1	(0/0)	(0/10)	{}	{E3,E5}	1	{}	{Q1}
...	...	...	...	...	...	...	...
E5	(0/10)	(0/18)	{E1,E3}	{E7,E8}	4	{O1(0/3)}	{Q1}
E6	(10/10)	(10/18)	{E2, E3, E4},	{E7, E9, E10}	1	{}	{Q1}
...	...	...	...	...	...	...	...



Incremental Monitoring Algorithm

Complete Algorithm



Complete Algorithm

1. Query moves further then the furthest current NN → “From scratch”
2. Weight updates
3. Query updates
4. Object updates
5. Re-computation with the remaining valid part of the tree

Overview

1. Introduction
2. Background
3. Data Structures
4. Incremental Monitoring Algorithm (IMA)
5. Conclusion

Conclusion



- Benefits of IMA:
 - Avoids re-computation “from scratch” and only re-computes invalidated results
 - Can handle high dynamics of road networks
 - Continuous result

IMA was the first work on that important problem of CkNN in road networks.

- Since then, continuous nearest neighbour monitoring has been studied in several paper in different research fields.

Thank you for your attention.

Questions?

