

Query Processing in Spatial Network Databases

written by D. Papadias, J. Zhang, N. Mamoulis and Y. Tao

Christoph Niemz
ETH Zurich
8th May 2010

Goal of this Presentation

After this presentation you should...

- Know what the purpose of a **Spatial Network Database (S NDB)** is
- understand the **necessity for new query processing methods** in spatial network databases
- have a basic idea of the **architecture** of the proposed Spatial Network Database
- understand two versions of one exemplary **algorithm** for the Spatial Network Database: the k-Nearest-Neighbor algorithm

Outline

- 0. Motivation
- 1. Definitions

- 2. Background: Network Representation
 - 2.1 R-Tree
 - 2.2 Adjacency list
- 3. SNDB architecture
- 4. *k*NN algorithm
 - 4.1 Incremental Euclidean Restriction (IER)
 - 4.2 Incremental Network Expansion (INE)

- 5. Performance
 - 5.1 IER vs. INE
- 6. Conclusion

Intro

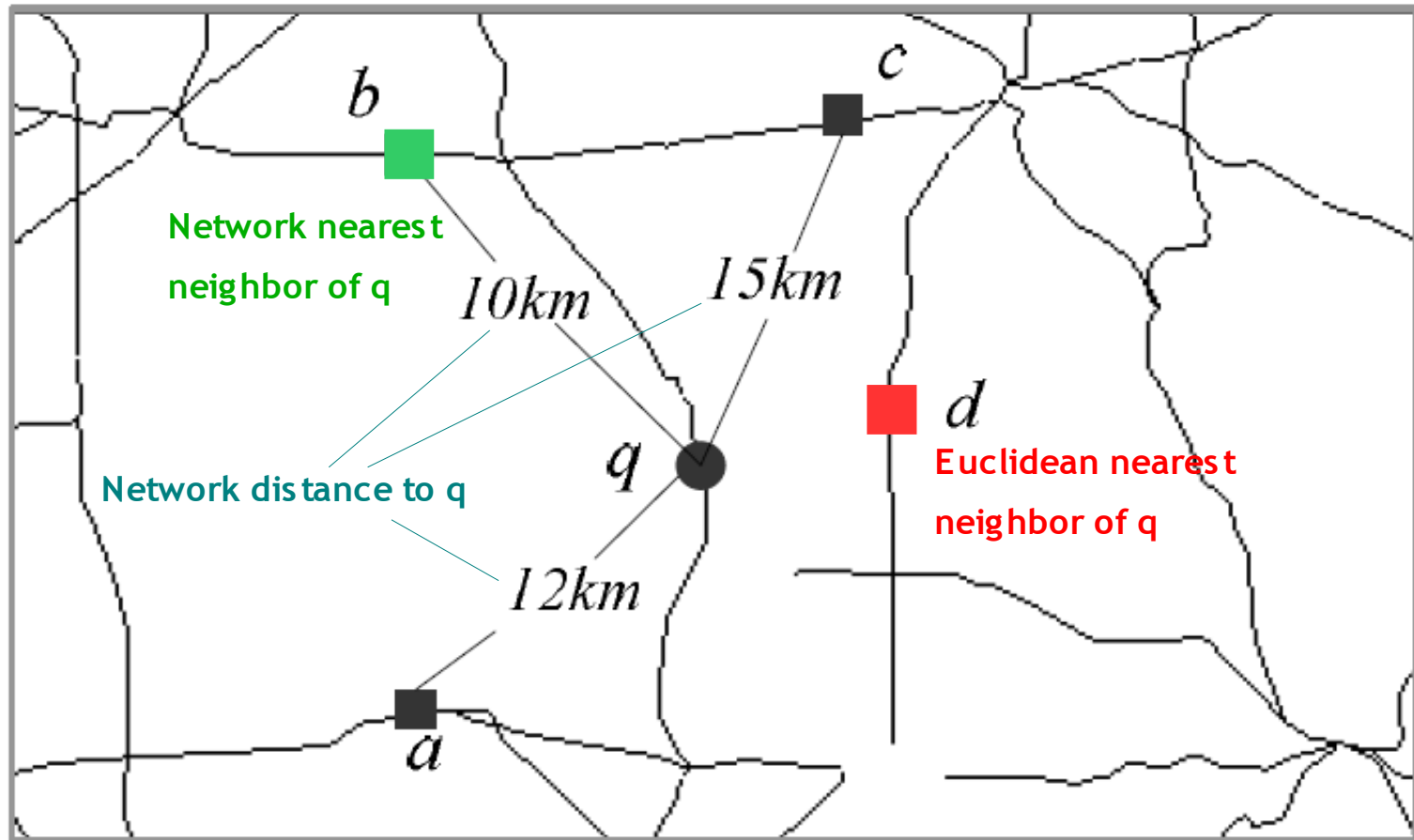
Technical Core

Discussion

0. Motivating Example

Example 1-Nearest-Neighbor-Query:

„Return the Hotel which is nearest to my current location q .“



[PZMT03]

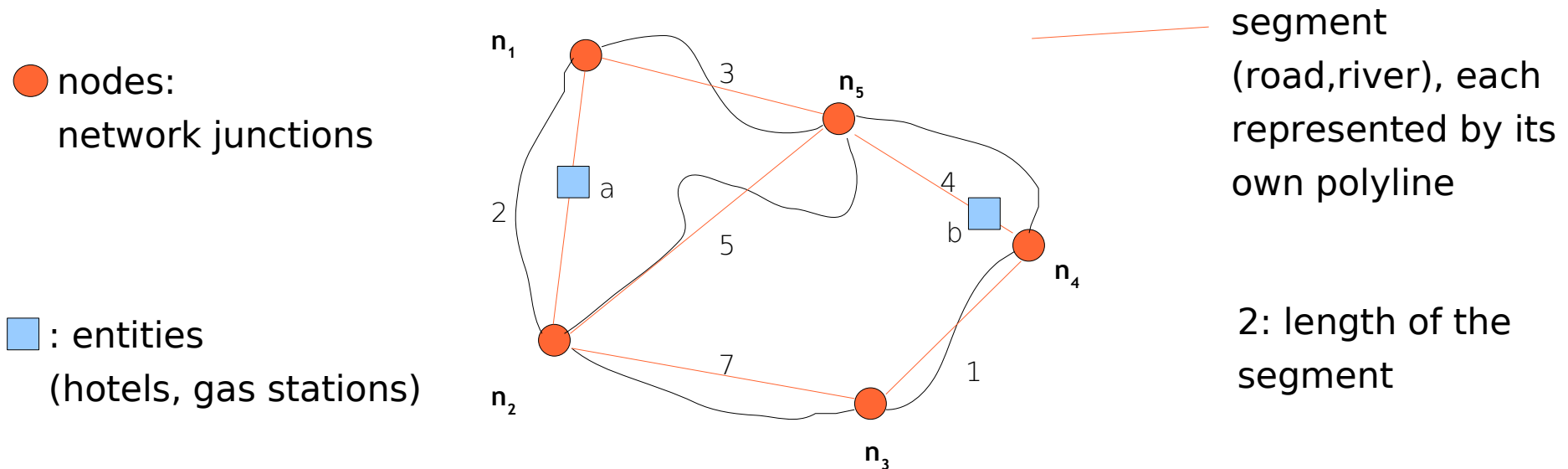
0. What is a Spatial **Network** Database (SNDB) and what do we need it for?

- SNDB processes spatial queries on objects whose movement is constrained by an underlying network
- SNDB must be able to process the „network version“ of the most popular spatial queries:
 - k -Nearest Neighbor
 - Range Search
 - Closest Pairs
 - e-Distance Joins
- e.g. „network version“ of k NN: Given a query point q , return the k objects with smallest network distance from q .

1. Modeling Graph

Assumptions:

1. There is a digitization process that generates a **Modeling graph** out of a **Real-Life Network**
2. Segments (edges) are bidirectional
3. Length of a segment is given by the Euclidean length of its polyline



This graph will serve as an example throughout the presentation.

1. Definitions

- The Euclidean distance $d_E(p,q)$ of two distinct entities p,q is $d_E(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$.
- The network distance $d_N(p,q)$ of two distinct nodes p,q is the length of the shortest path between them.

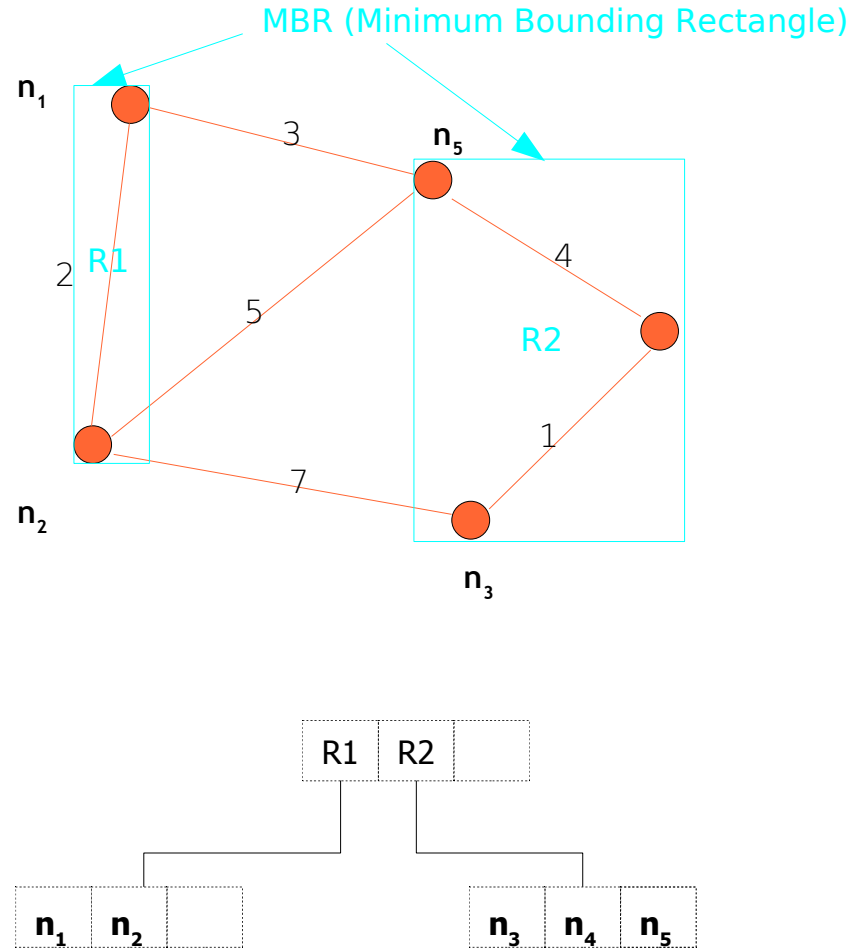
1. Implication:

The Euclidean lower-bound property

- How can we use Euclidean/location information to restrict the search space of spatial network queries?
- Idea: $d_E(p,q) \leq d_N(p,q)$, i.e., the Euclidean distance between two points is equal or smaller than their network distance.

2.1 Review: R-Tree

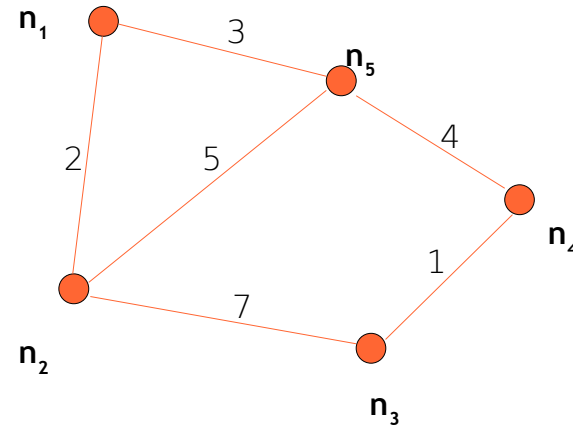
- stores **Euclidean** point and region information
- balanced
- efficiently computes:
 - range queries
 - k NN queries



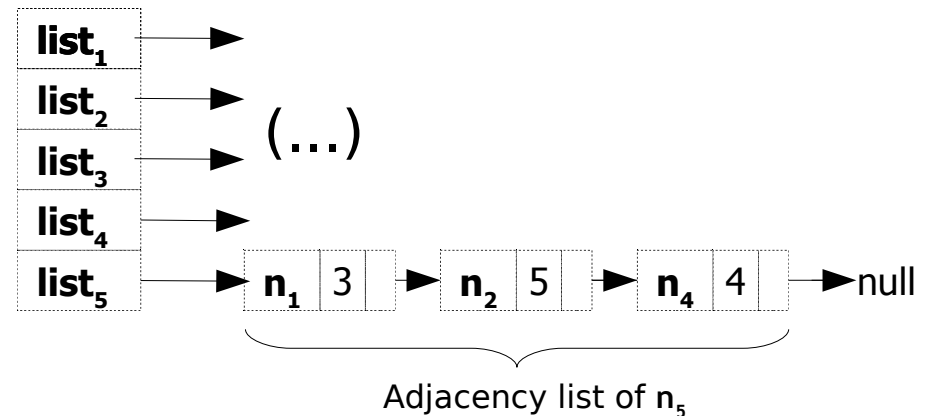
R-Tree Representation of the locations of the above nodes

2.2 Review: Adjacency List

- Space-efficient way to represent a (weighted) graph



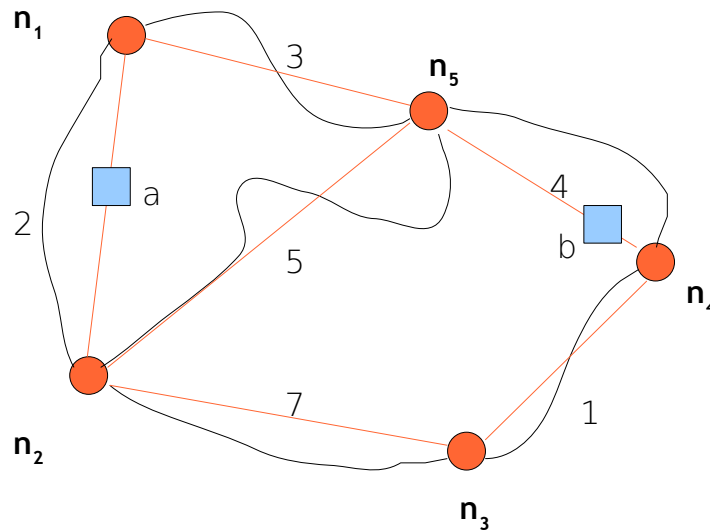
List of adjacency lists



3. Idea: Combine R-Tree and Adjacency List → suitable S NDB Architecture

	Network	Euclidean Space
Modeled by	Graph	Points/Regions
Information stored	Connectivity	Location
Data Structure	Adjacency list	R-Tree
Computes	Graph Traversal Algorithms: ▪ Network Expansion ▪ Shortest-Path (Dijkstra)	Euclidean Queries: ▪ Range ▪ Nearest Neighbor ▪ Closest-Pair ▪ Intersection Join

3. The Architecture: Separation of network and entities

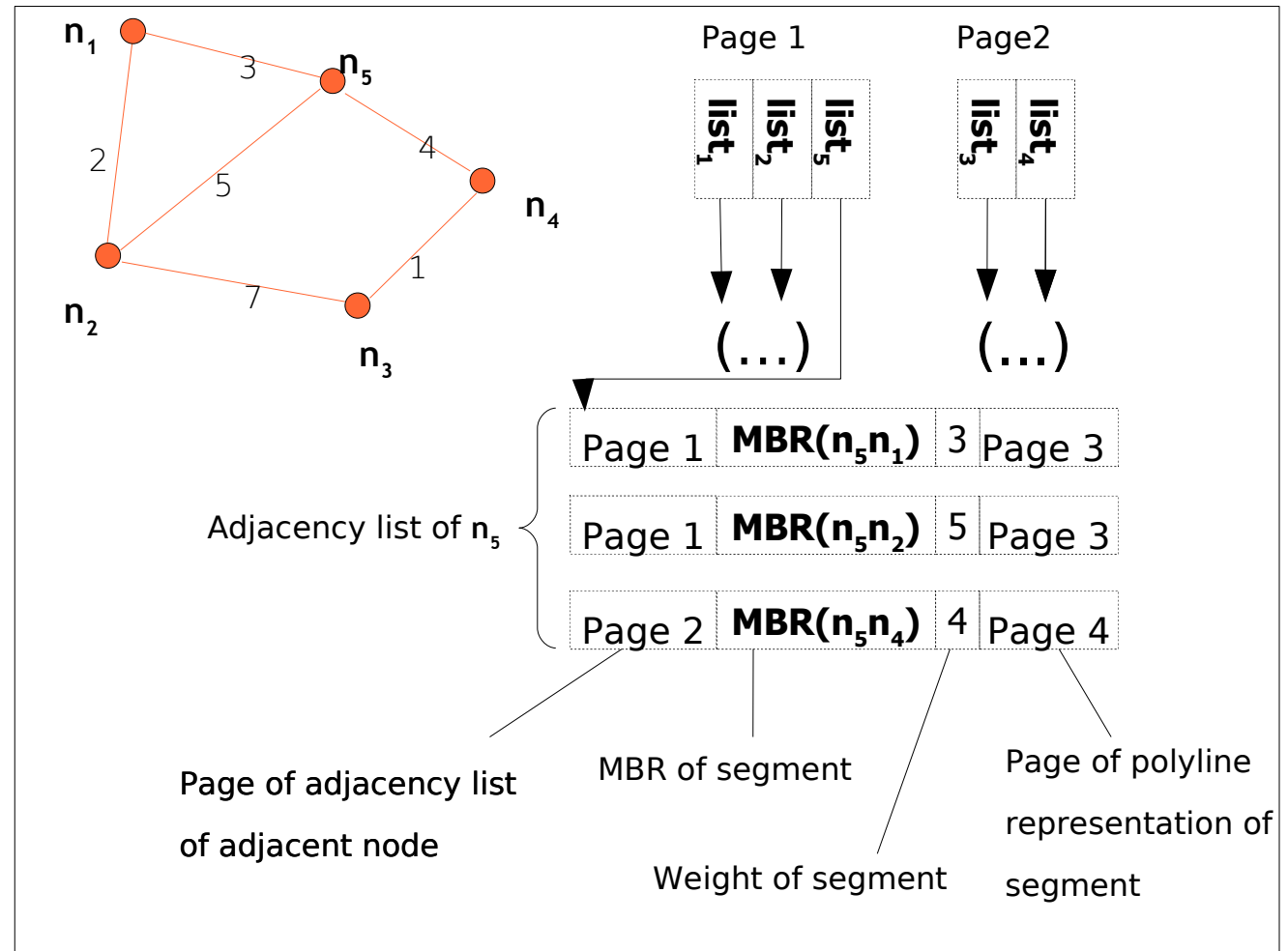


The architecture presented on the next slides will deal only with the **network**.

Each **entity dataset** will be stored in a separate R-Tree.

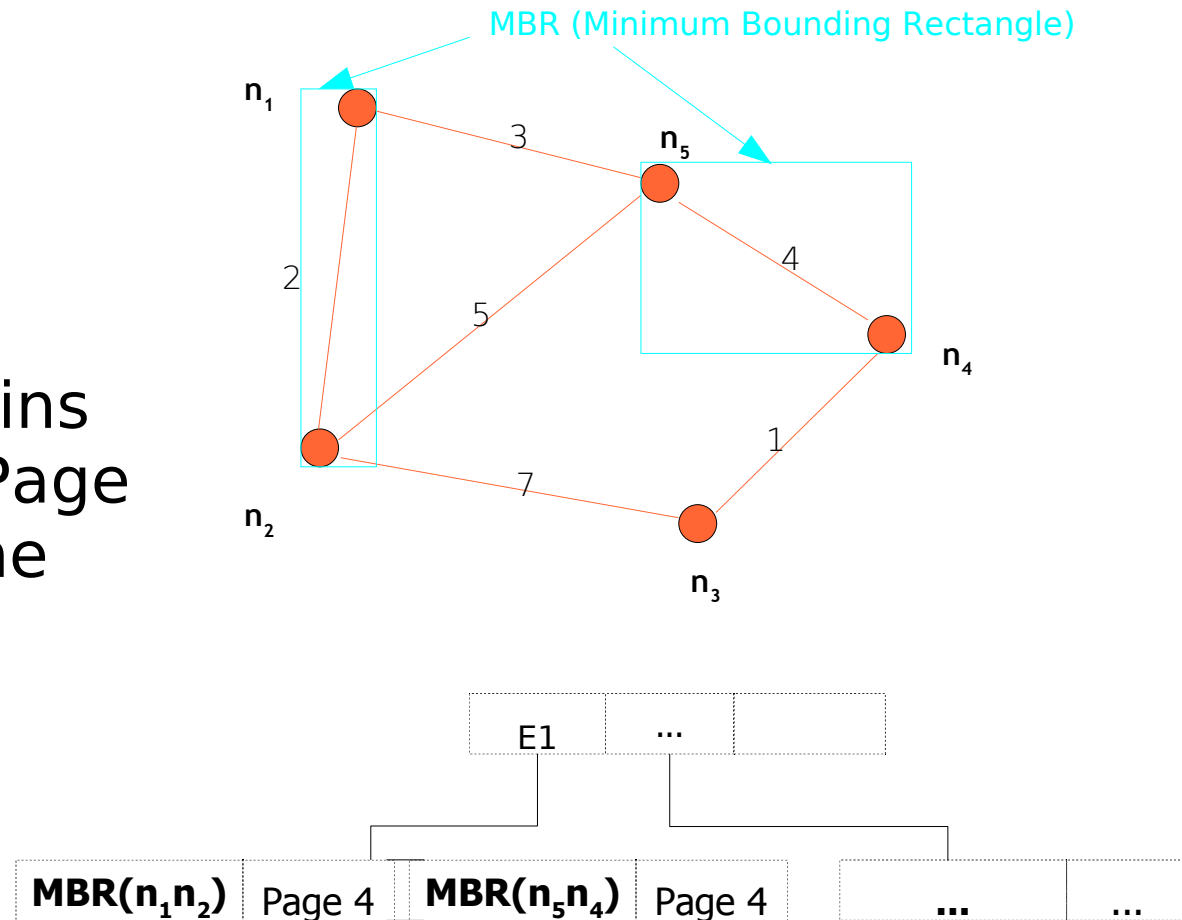
3. The Architecture: The adjacency component

- stores adjacency list of each node
- additional information in each adjacency list entry



3. The Architecture: The network R-Tree

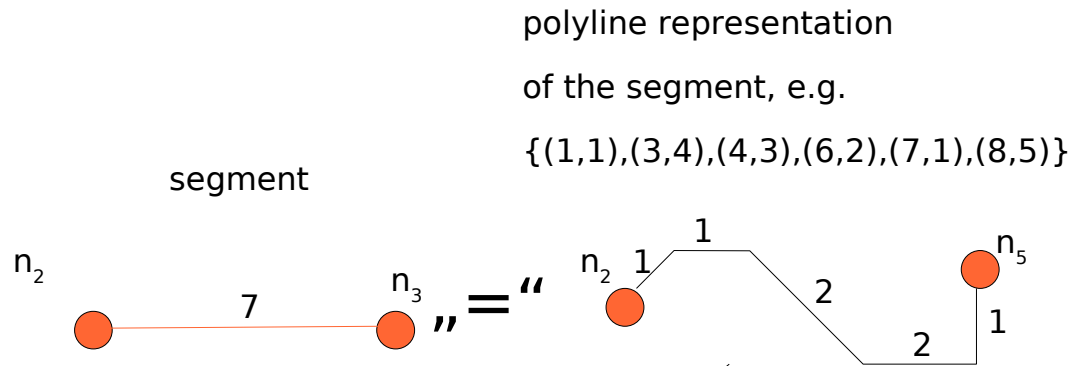
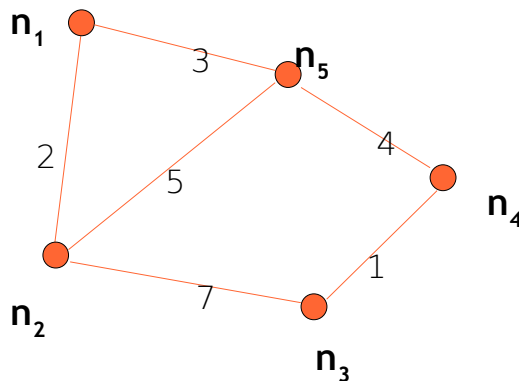
- Indexes MBR of each segment
- Each entry contains a pointer to the Page where the polyline of the segment is stored



R-Tree representation of the locations of the highlighted segments

3. The Architecture: The polyline component

- Stores for each segment its polyline representation
- Connects network R-tree and adjacency component

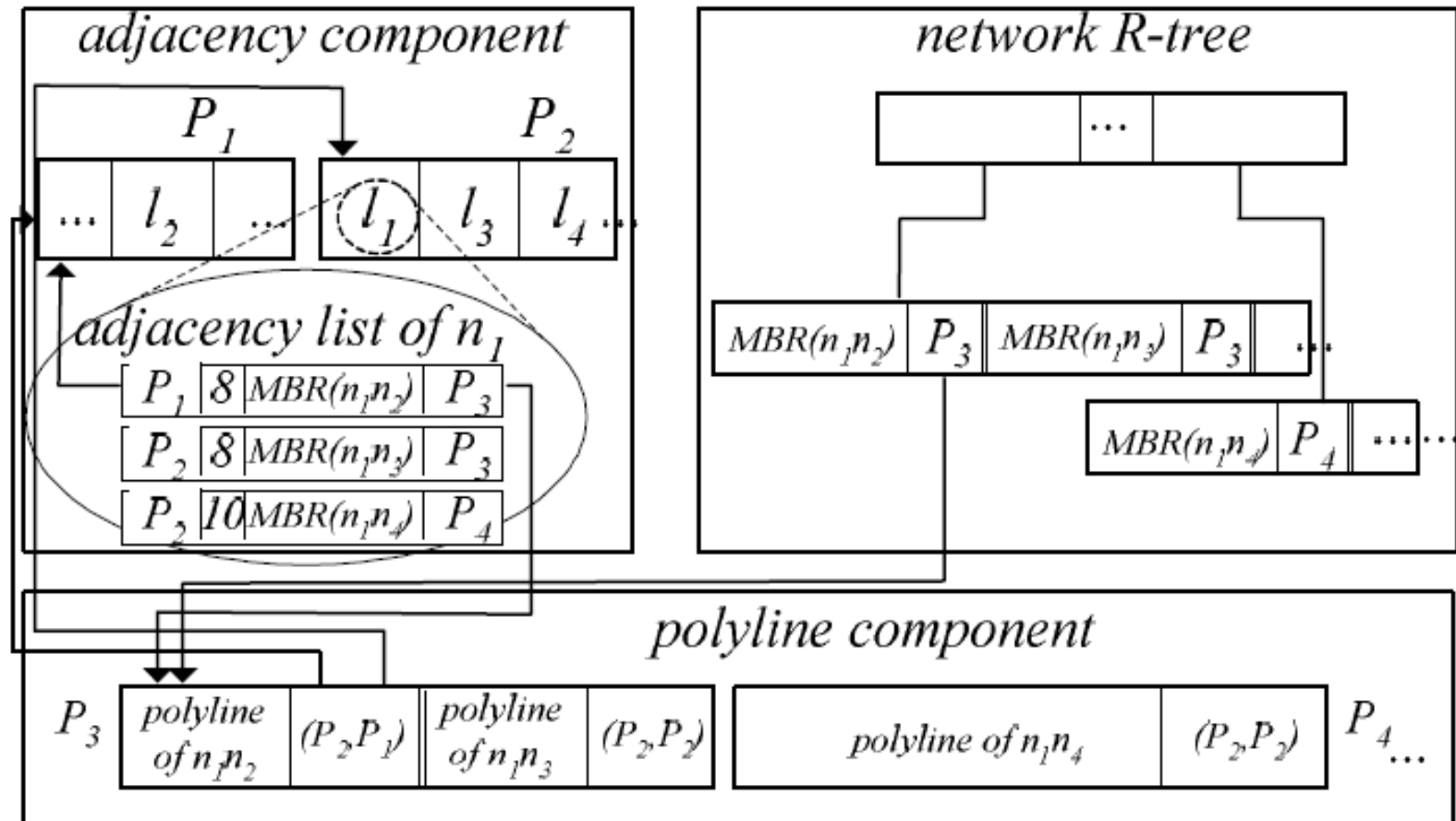


Pages of the adjacency lists of n_1 and n_2

polyline of n_1n_2	(Page 1, Page 1)	polyline of n_5n_4	(Page 1, Page 2)
--	------------------	--	------------------

Example Polyline component entries.

3. The Architecture: Big Picture



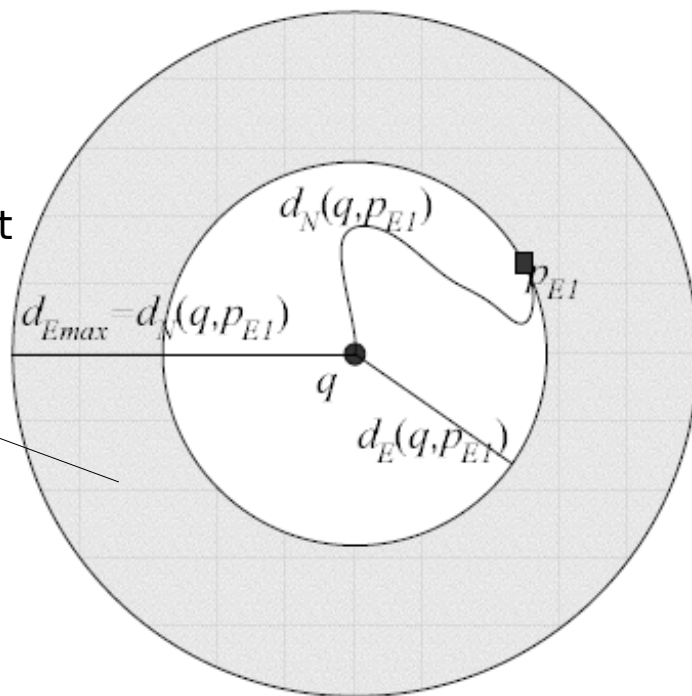
3. Operations on the architecture

With this architecture we now can get useful information from the network graph, e.g.:

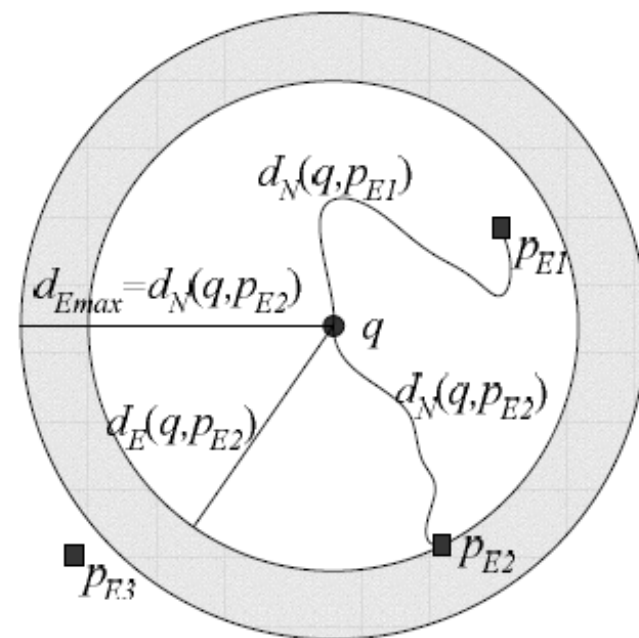
- Which entities lie on a segment?
- Is entity p on segment seg ?
- On which segment is entity p ?
- What is the network distance between two points p_1 and p_2 ?
 - Dijkstra's algorithm is used to compute the shortest path
- **We need these primitive operations for the construction of algorithms on the architecture**

4.1 *k*NN in SNDB - Incremental Euclidean Restriction: Example for $k=1$

All remaining nearest neighbor candidates have to be in this area



(a) 1st Euclidean NN

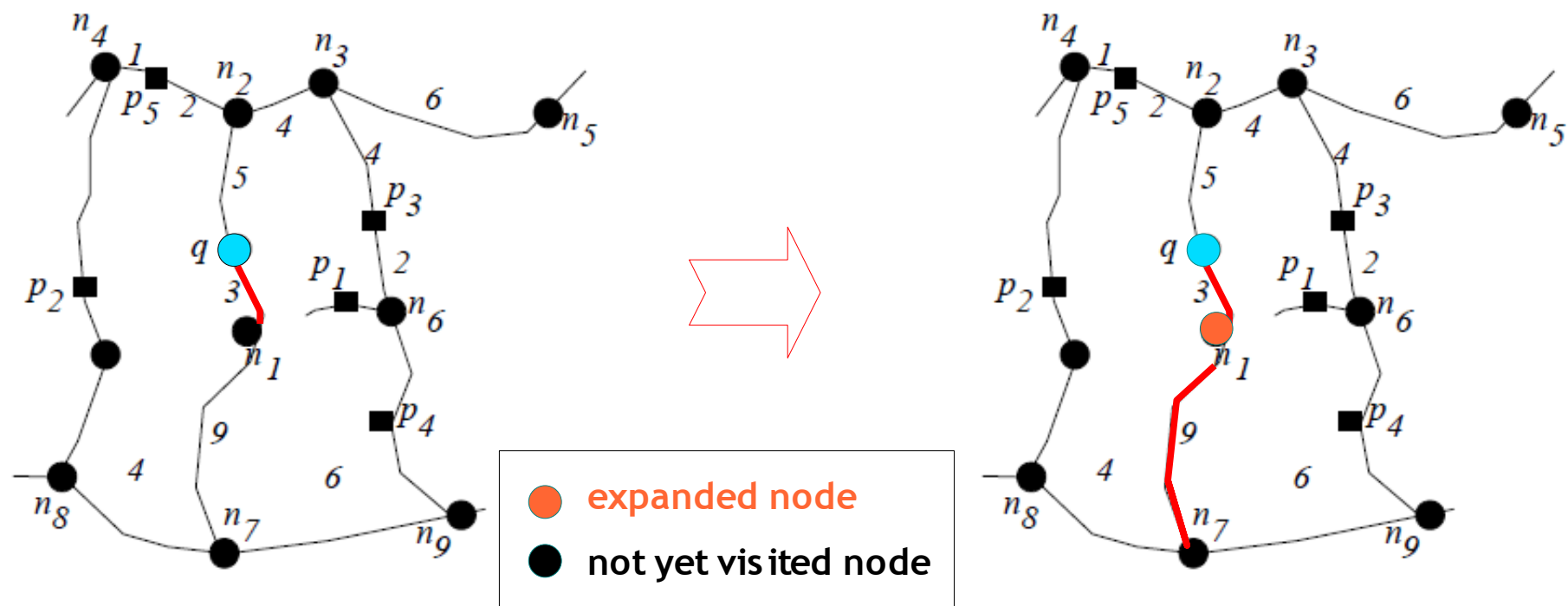


(a) 2nd Euclidean NN

Figure 4.1: Finding the NN p_{E2}

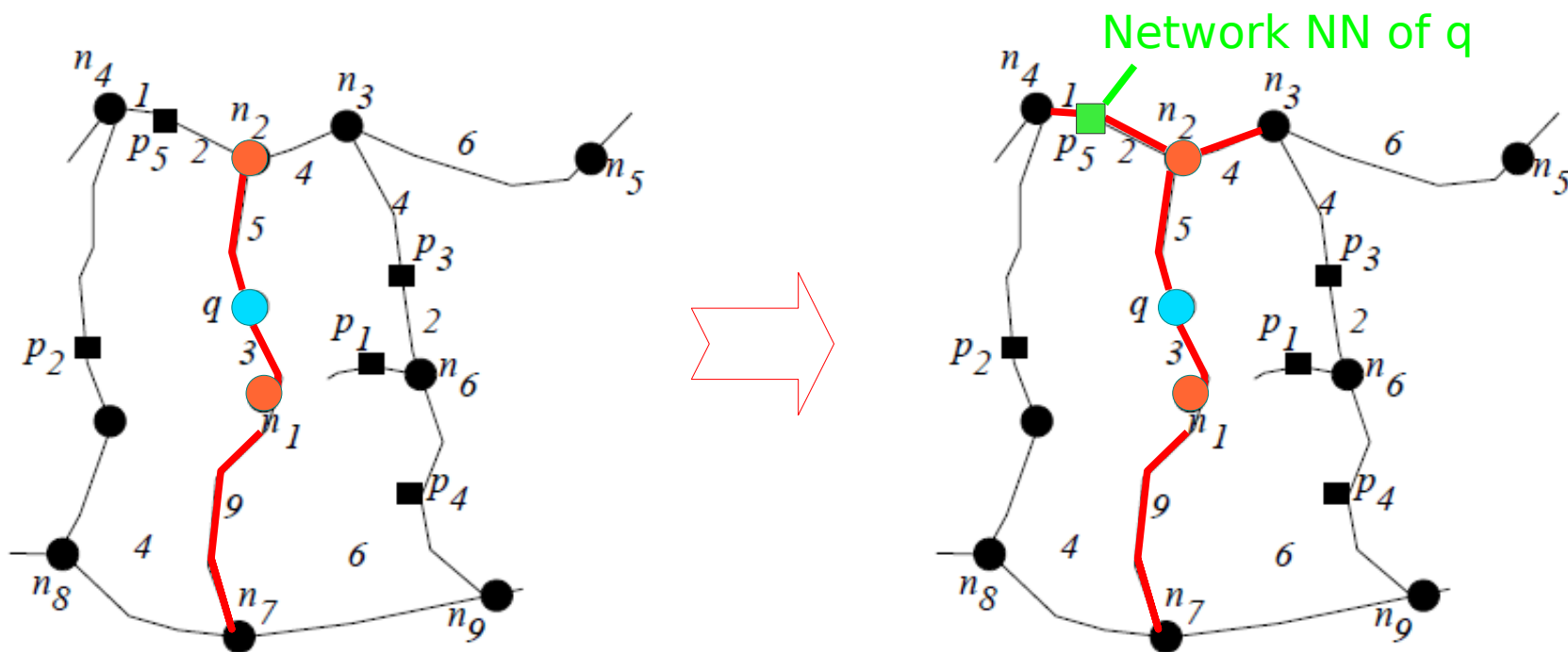
- Use the Euclidean lower-bound property to limit the search space

4.2 kNN in SNDB - Incremental Network Expansion: Example for k=1



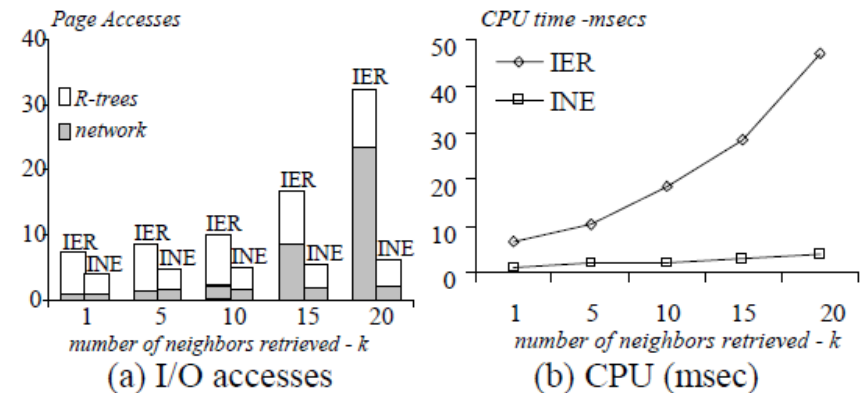
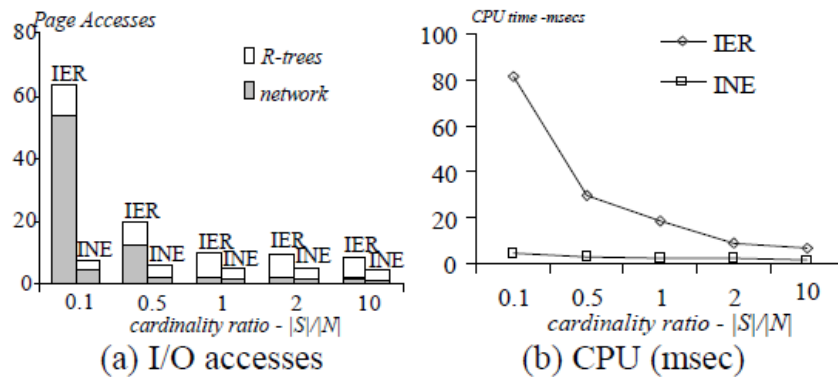
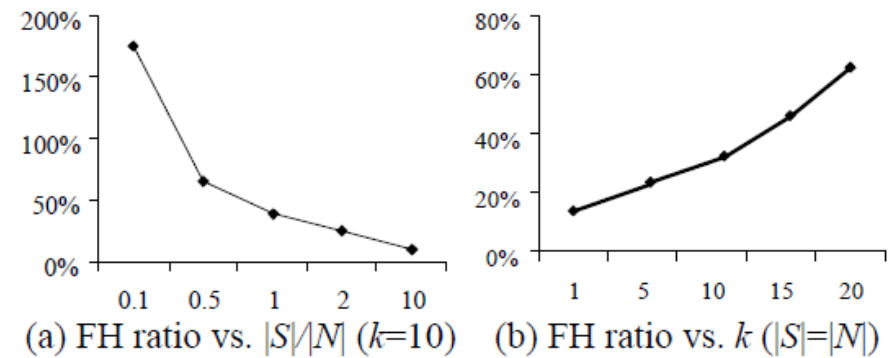
- Procedure: Always expand the node which has smallest network distance to query point q
- Expand node „=“ inspect all its outgoing segments

4.2 kNN in SNDB - Incremental Network Expansion: Example for $k=1$



- p_5 has to be q 's NN, since the network distance $d_N(q, p_5)$ provides a bound $d_{N_{\max}}$ that restricts the search space. All other entities are more far away.

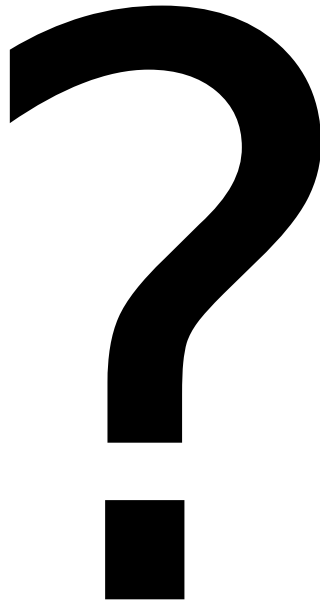
5. Performance: IER vs. INE



- IER performs very bad when cardinality ratio ($= \# \text{entities} / \# \text{netw. nodes}$) is low and when k is large
 - This is because for high cardinality ratios (and for large k as well) the entities we are interested in might be far away in the Euclidean space.
 - We find a high number of Euclidean NNs that are no candidates for network NNs (i.e. False hits) before encountering the network NNs we are actually interested in.

6. Conclusion

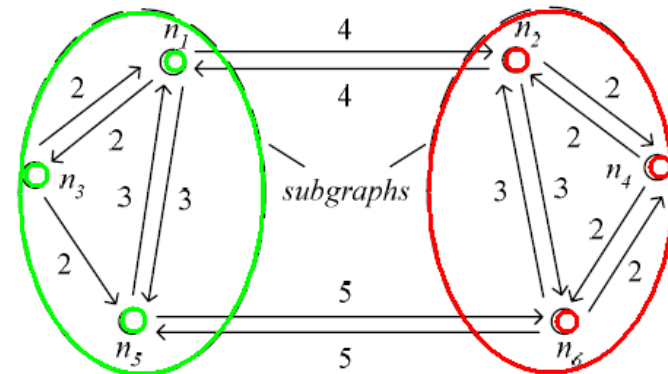
- + separation of entities from network
 - allows to handle queries on the entities separately
- + algorithms provided for 4 basic query forms
 - However, mostly only unoptimized basic algorithms
- architecture is not very advanced
 - possible improvements: partitioning the network, storing additional information (e.g. paths, subpaths)
- Euclidean Restriction is based on an assumption that might not always hold in reality
 - The cost of each segment might e.g. be defined as travel time and not as the Euclidean length of its polyline.
- Only basic optimizations provided:
 - e.g. Dijkstra's algorithm might visit a lot of nodes and edges (worst case: $O(n\log(n)+m)$) until we finally reach a node we are interested in.



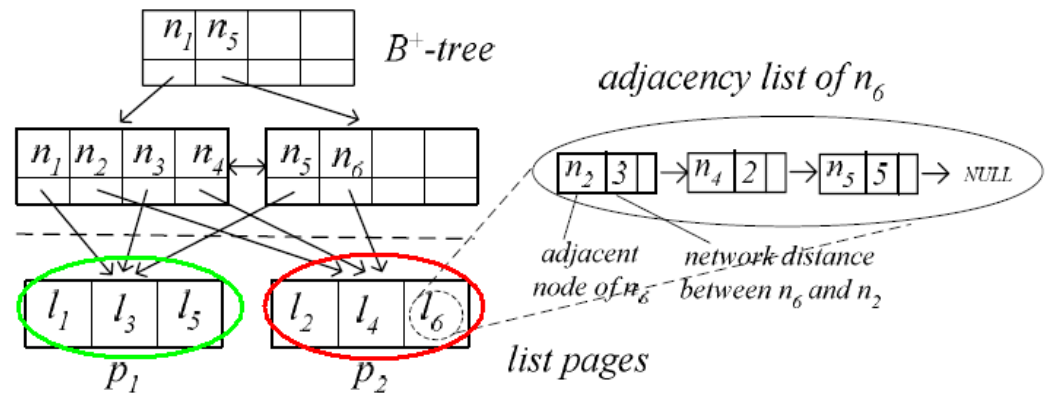
Backup

2.2 Adjacency List with CCAM (Connectivity-Clustered Access Method)

- CCAM: Adjacency lists of neighbored nodes are stored in such a way that proximity is preserved for most of the nodes
- We use an R-Tree to index the nodes, instead of a B⁺-Tree



(a) A graph example



(b) The CCAM structure

2.2 CCAM (Connectivity-Clustered Access Method)

- Minimizes the Connectivity-Residue-Ratio (CRR):

$$CRR = \frac{\text{total number of unsplit segments}}{\text{total number of segments}}$$

- Provides operations to:
 - Insert node (dynamically)
 - Delete node(dynamically)
 - Find node

...while maintaining high CRR

- For more information look at paper:

Shashi Shekhar, Duen-Ren Liu: CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations, IEEE Transactions on Knowledge and Data Engineering 1997