Unified approach to detecting spatial outliers

Shashi Shekhar, Chang-Tien Lu And Pusheng Zhang

Pekka Maksimainen
University of Helsinki
2007

Spatial outlier

- Outlier
  - Inconsistent observation in data set
- Spatial outlier
  - Inconsistent attribute value in spatially referenced object
  - Spatial values (location, shape, ...) are not of importance
  - Local instability
    - Extreme attribute values compared to neighbors

Application domains

- Transportation, ecology, public safety, public health, climatology, location based services, ...
- Minnesota Department of Transportation Traffic Management Center Freeway Operations group traffic measurements
    - 900 sensor stations
    - Attributes
        - Volume of traffic on the roads
        - Occupancy
        - Sensor ID.

Traffic data

- Spatial attribute
  - Sensor location
    - $S = \{s_1, s_2, s_3, ..., s_n\}$
    - <Highway, milepoint>
    - Directed graph indicating road between two sensor locations (eg. $s_1 \rightarrow s_2$)
- Attribute data
  - Traffic volume, occupancy of road
- We are interested in finding locations which are different than their neighbors – that is outliers!

Measuring outlierness

- Set of definitions
  - *f(x)* – attribute value for location *x*
  - *N(x)* – neighborhood of location *x*
  - $E_{y \in N(x)}(f(y))$ – average attribute value for neighbors of *x*
  - $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$ – difference of *x*'s attribute value to it's neighbors
- For normally distributed *f(x)* we can measure outlierness by
  - $Z_{S(x)} = |(S(x) - \mu_S / \sigma_s)| > \theta$
    - μ_S is the mean value of S(x)
    - σ_S is the standard deviation of S(x)
  - Choice for θ specifies confidence level
    - Confidence level of 95% ~ θ = 2

Measuring outlierness (2)

- More general definition for outlierness
  - $f^N_{aggr} : \mathbb{R}^N \rightarrow \mathbb{R}$ – aggregate function for the values of *f* over neighborhood
  - $F_{diff} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ – difference function
  - $ST : \mathbb{R} \rightarrow \{True, False\}$ – statistical test for significance
- For finding outliers we can define above functions in different fashion to find outliers. In previous slide we defined aggregate function to ”average attribute value of node *x*'s neighbors”. Difference function was the arithmetic difference between *f(x)* and aggregate value. Statistical significance was defined with the help of mean and standard deviation.
- Object *O* is an *S*-outlier $(f, f^N_{aggr}, F_{diff}, ST)$ if $ST\{F_{diff}[f(x), f^N_{aggr}(f(x), N(x))]\}$ is true

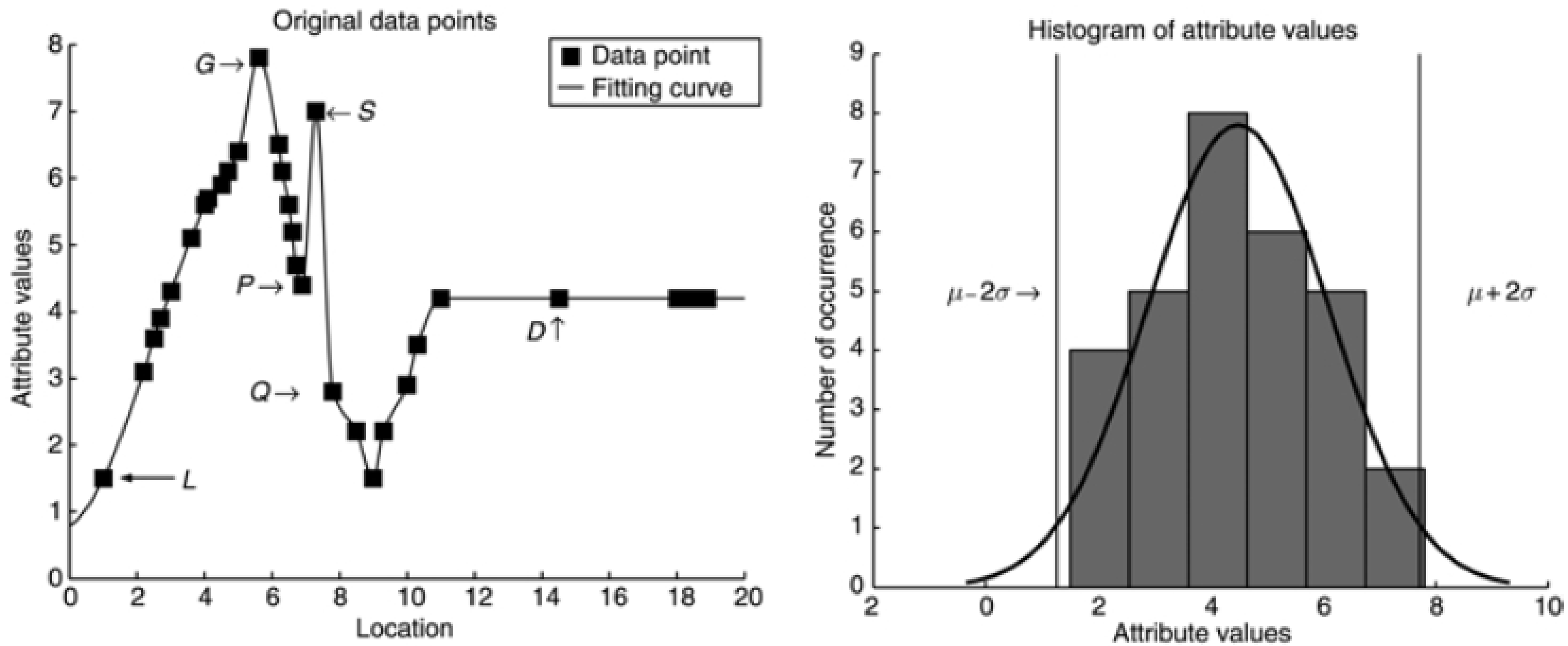
Measuring outlierness (3)

- DB(p, D)-outlier (distance based)
  - Statistical significance measure *p* (fraction of nodes)
  - *N* objects in set *T*
  - Object *O* is a DB(p, D)-outlier if atleast fraction *p* of the objects in *T* lie greater than distance *D* from *O*
  - Let f_{aggr}^{N} be the number of objects within the distance *D* from object *O*
  - Statistical test function can be defined as

$$(N - f_{aggr}^{N}(x))/(N) > p$$

Related methods

- Non-spatial methods are not fit for detecting spatial outliers

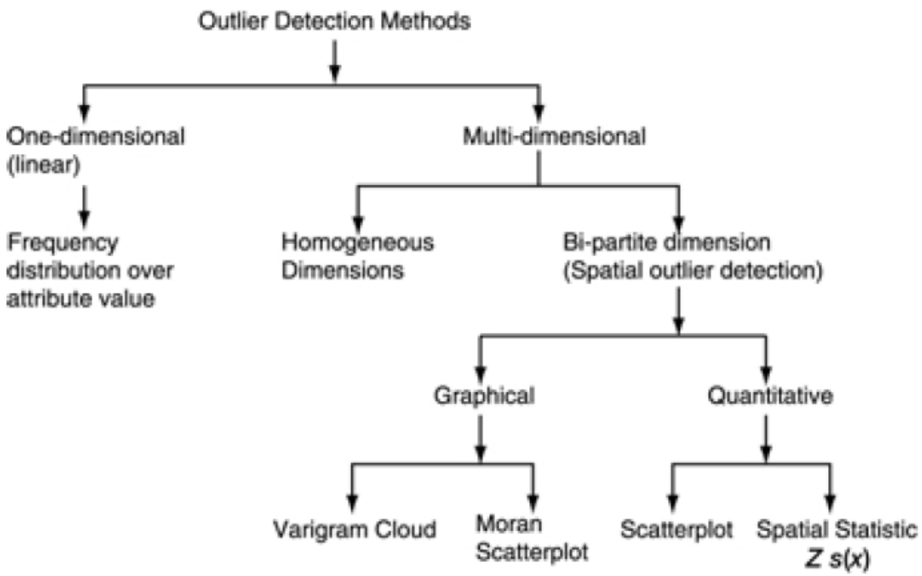


(a) An example data set (b) Histogram

- Node G is an outlier because it's attribute value exceeds the threshold on normal distribution limits
  - Spatial location is not considered

Related methods (2)

- Outlier detection method categories



(a) Classification

	One-dimensional (linear)	Multi-dimensional	
		Homogeneous	Spatial (Spatial method)
Neighbor Definition	N/A	location and attribute	location
Comparison	with population distribution	location and attribute	attribute values of neighbors

(b) Comparison

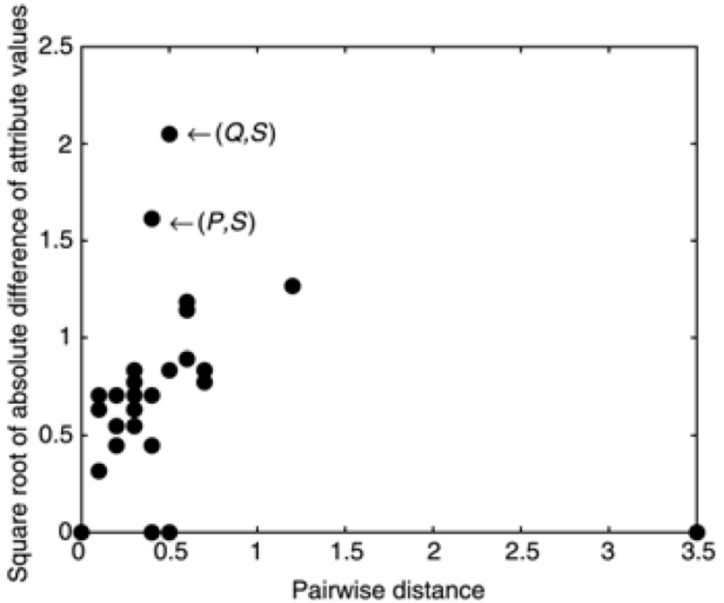
- Homogeneous methods don't differentiate between attribute dimensions and spatial dimensions
- Homogeneous methods use all dimensions for defining neighborhood as well as for comparison

Related methods (3)

- Bi-partite multi-dimensional tests are designed to detect spatial outliers
  - Spatial attributes characterize location, neighborhood and distance
  - Non-spatial attributes are used to compare object to its neighbors
- Two kinds of bi-partite multi-dimensional tests
  - Graphical tests
    - Visualization of spatial data which highlights spatial outliers
  - Quantitative tests
    - Precise test to distinguish spatial outliers

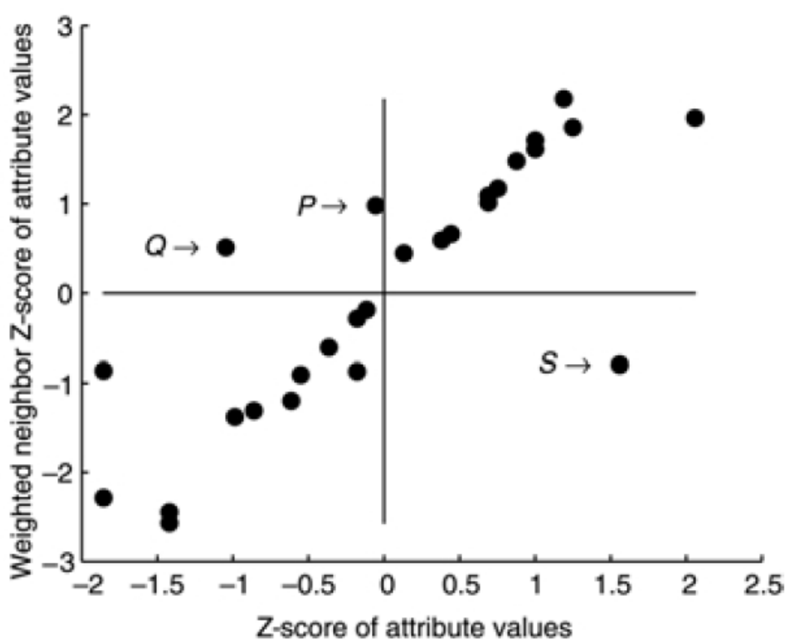
Variogram-cloud

- Variogram-cloud displays objects related by neighborhood relationships
- For each pair of locations plot the following values
  - Square root of the absolute difference between attribute values
  - Distance between the locations
- Locations that are near to eachother but large attribute differences might indicate spatial outlier
- Point *S* can be identified as spatial outlier



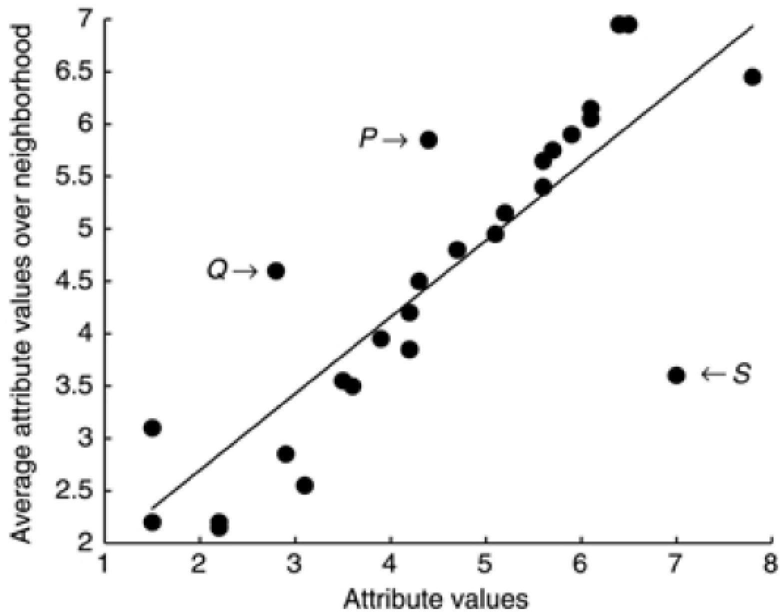
Moran scatterplot

- Moran scatterplot is a plot of normalized attribute value against the neighborhood average of normalized attribute values
  - $Z[f(i)=(f(i)-\mu_f)/\sigma_f]$
- Upper left and lower right quadrants indicate spatial association of dissimilar values
- Points P and Q are surrounded by high value neighbors
- Point S is surrounded by low value neighbors
- → Spatial outliers



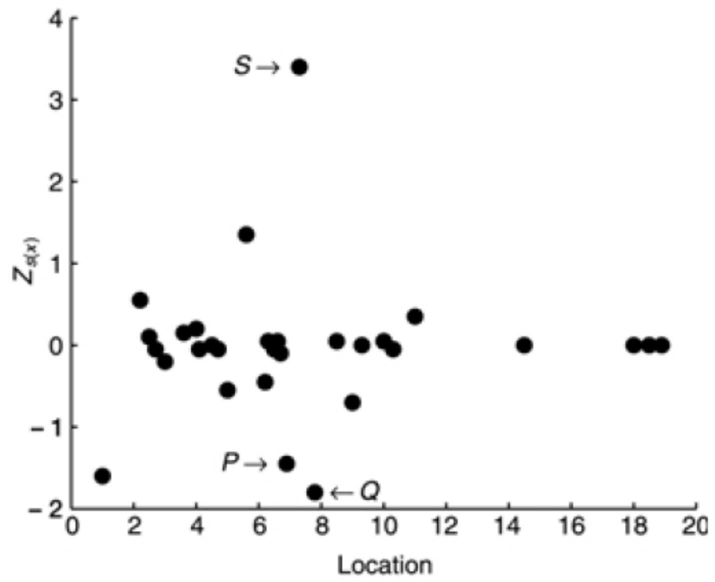
Scatterplot

- Scatterplot shows attribute values on the X-axis and the average of the attribute values in the neighborhood on the Y-axis
- Best fit regression line is used to identify spatial outliers
- Positive autocorrelation
  - Scatter regression slopes to the right
- Negative autocorrelation
  - Scatter regression slopes to the left
- Vertical difference of a data point tells about outlierness



Spatial statistic $Z_{S(x)}$ test

- Spatial statistic test shows the location of data points in 1-D space on X-axis and statistic test values for each data point on Y-axis
- Point *S* has $Z_{S(x)}$ value exceeding 3 and will be detected as spatial outlier
- Because *S* is an outlier the neighboring data points *P* and *Q* have values close to -2



Spatial outlier detection problem

- Objective is to design a computationally efficient algorithm to detect *S*-outliers
- Previously introduced functions and measumerements are used (aggregates, difference functions, neighborhoods, ...)
- Constraints
  - The size of data is greater than main memory size
  - Computation time is determined by I/O time

Model building

- Model building: *”efficient computation method to compute the global statistical parameters using a spatial join”*
- Distributive aggregate functions
  - min, max, sum, count, ...
- Algebraic aggregate functions
  - mean, standard deviation, ...
- These values can be computed by single scanning of the data set
  - I/O reads
- Algebraic aggregate functions can be used by difference function F_{diff} and statistical test function *ST*

Model building algorithm

Model building algorithm

Input: S is a spatial framework;

f is an attribute function;

N is the neighborhood relationship;

f^N_{aggr} is the neighborhood aggregate function;

$D^{G1}_{aggr}, D^{G2}_{aggr}, \ldots, f^{Gk}_{aggr}$ are the distributive aggregate functions;

Output: Algebraic aggregate functions $A^{G1}_{aggr}, A^{G2}_{aggr}, \ldots, A^{Gk}_{aggr}$

for(i $=$ 1;i $\leq$ $|S|$;i++){

$O_i =$ Get_One_Object(i,S); /* *Select each object from* S */

NNS $=$ Find_Neighbor_Nodes_Set(O_i, N, S); /* *Find neighbor nodes of* O_i *from* S */

for(j $=$ 1;j$\leq$ |NSS|;j++){

$O_j =$ Get_One_Object(j,NNS); /* *Select each neighbor of* O_i */

$f^N_{aggr} =$ Compute_and_Aggregate($f(O_i)$,$f(O_j)$);

}

/* *Add the element to global aggregate functions* */

Aggregate_Element($D^{G1}_{aggr}, D^{G2}_{aggr}, \ldots, D^{Gk}_{aggr}$,$f^N_{aggr}$,i);

}

/* *Compute the algebraic aggregate functions*/*

$\langle A^{G1}_{aggr}, A^{G2}_{aggr}, \ldots, A^{Gk}_{aggr} \rangle$ = Compute_Algebraic_Aggregate($D^{G1}_{aggr}, D^{G2}_{aggr}, \ldots, D^{Gk}_{aggr}$);

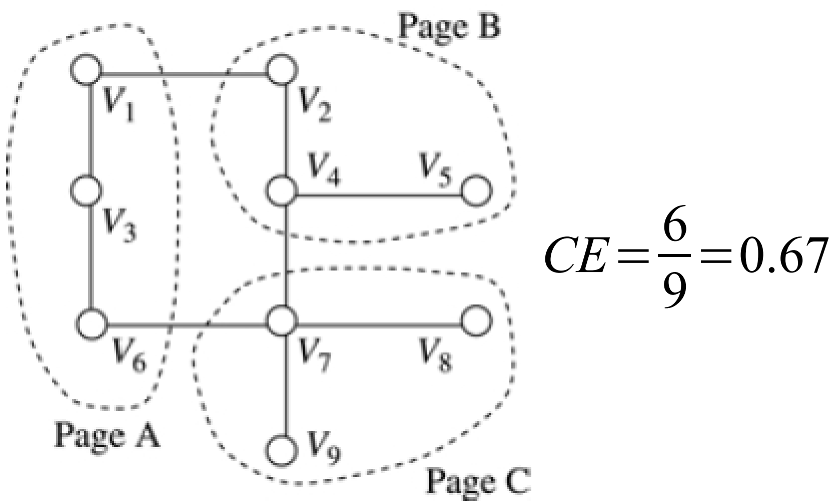
return ($A^{G1}_{aggr}, A^{G2}_{aggr}, \ldots, A^{Gk}_{aggr}$).

Effectiveness of model building

- Efficiency depends greatly on I/O
  - Most time consuming process in model building algorithm is the method *Find_Neighbor_Nodes_Set()*
    - If neighboring nodes are not in memory then extra I/O read must be done
    - Idea: try to cluster each node with its neighbors to same disk page
  - Clustering efficiency
    - Practically CE defines the execution time

Clustering efficiency

- To get neighbors of node v_1 pages A and B must be read. Page A however is already in memory because v_1 was read from there.
- For node v_3 no extra reads are needed



$$CE = \frac{Total\ number\ of\ unsplit\ edges}{Total\ number\ of\ edges}$$

Route outlier detection

- Route outlier detection (ROD) detects outliers on the user given route
  - ROD retrieves the neighboring nodes for each node in given route *RN*
  - Compute neighborhood aggregate function F^{N}_{aggr}
  - Difference function F_{diff} is computed using the attribute function *f(x)*, neighborhood aggregate function and the algebraic aggregate functions computed in the model building algorithm
  - Test node *x* using the statistical test function *ST*

Route outlier detection (ROD) algorithm

Route outlier detection (ROD) algorithm

Input: S is a spatial framework;
f is an attribute function;
N is the neighborhood relationship;
f^N_{aggr} is a neighborhood aggregate function;
F_{diff} is a difference function;
$A^{G1}_{aggr}, A^{G2}_{aggr}, \ldots, A^{Gk}_{aggr}$ are algebraic aggregate functions;
ST is the spatial outlier test function;
RN is the set of node in a route;

Output: Outlier_Set.

```
for(i = 1;i ≤ |RN| ;i++){
   O_i = Get_One_Object(i,RN); /* Select each object from RN */
   NNS = Find_Neighbor_Nodes_Set(O_i, N, S);
   /* Find neighbor nodes of O_i from S */
   for( j = 1;j ≤ |NSS|;j++){
      O_j = Get_One_Object( j,NNS); /* Select each neighbor of O_i */
      f^N_aggr = Compute_and_Aggregate(f(O_i),f(O_j));
   };
   F_diff = Compute_Difference(f, f^N_aggr, A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr);
   if(ST(F_diff, A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr) = = True){
      Add_Element(Outlier_Set,i); /* Add the element to Outlier_Set */
   }
}
return Outlier_Set.
```

Clustering methods

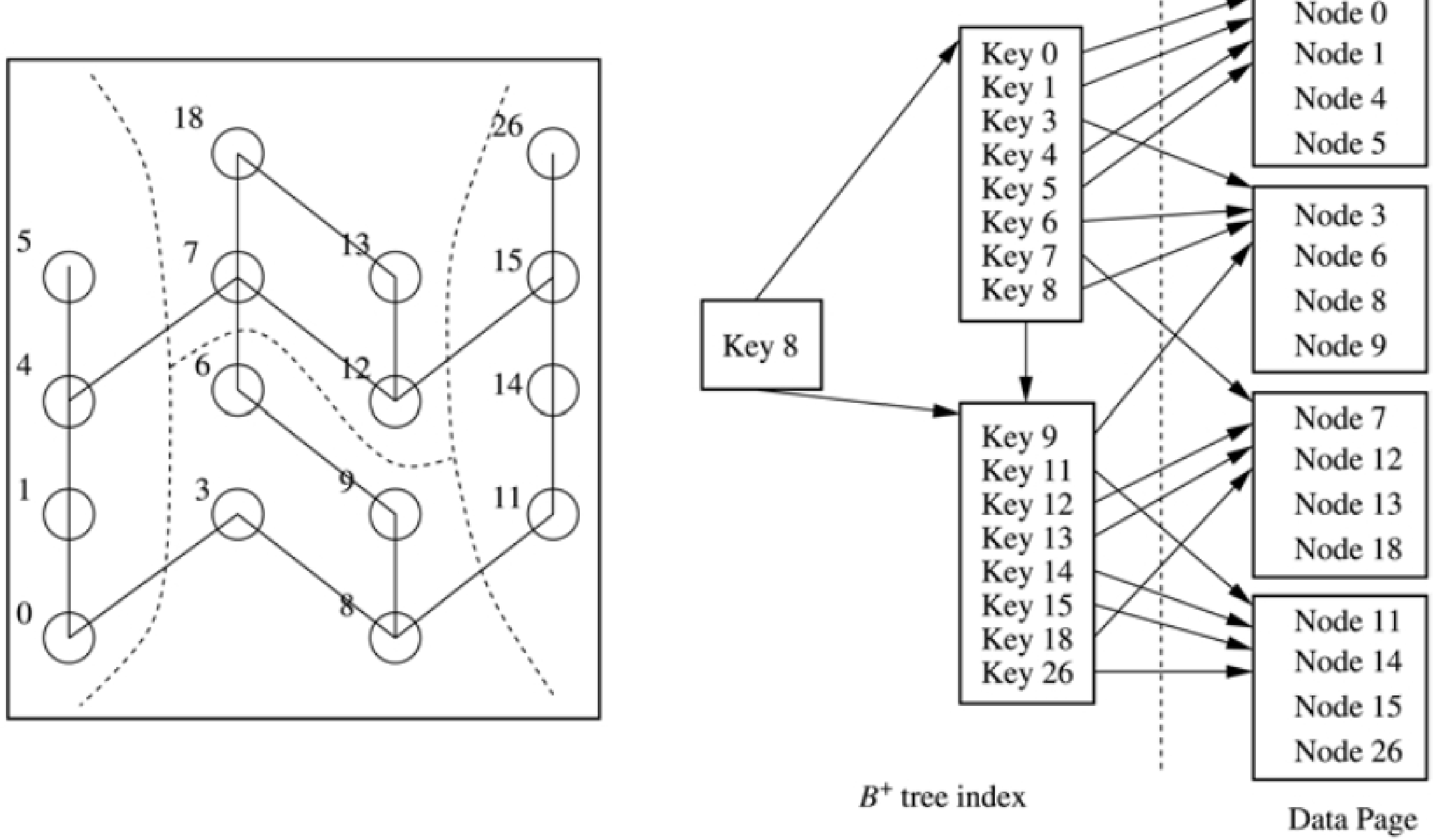
- Connectivity-clustered access method (CCAM)
  - Cluster the nodes via graph partitioning
    - Graph partitioning methods
  - Secondary index to support query operations
    - B^+-tree with Z-order
    - Grid File, R-tree, ...
- Linear clustering by Z-order
- Cell-tree

Clustering methods - CCAM

Graph partitioning

B$^+$-tree index

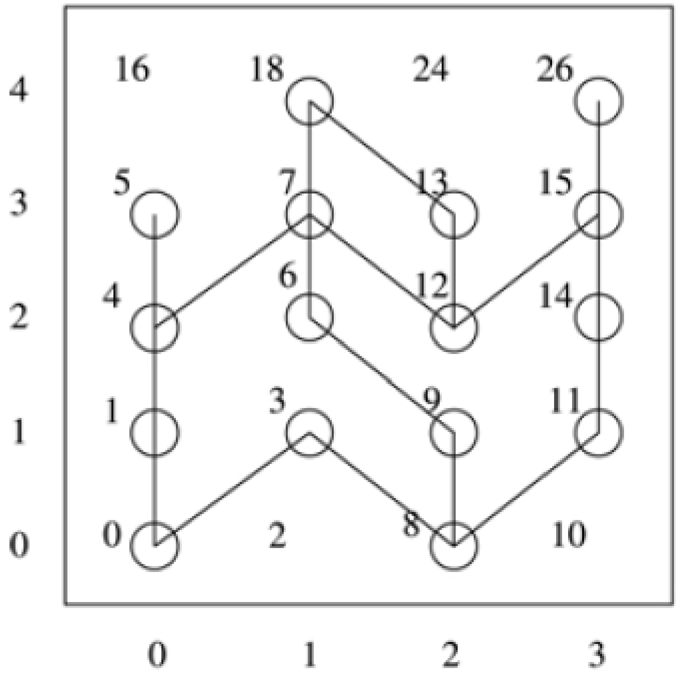
Data page



Clustering methods - Z-order

- Nodes in different partitions are
  - (0, 1, 3, 4)
  - (5, 6, 7, 8)
  - (9, 11, 12, 13)
  - (14, 15, 18 26)

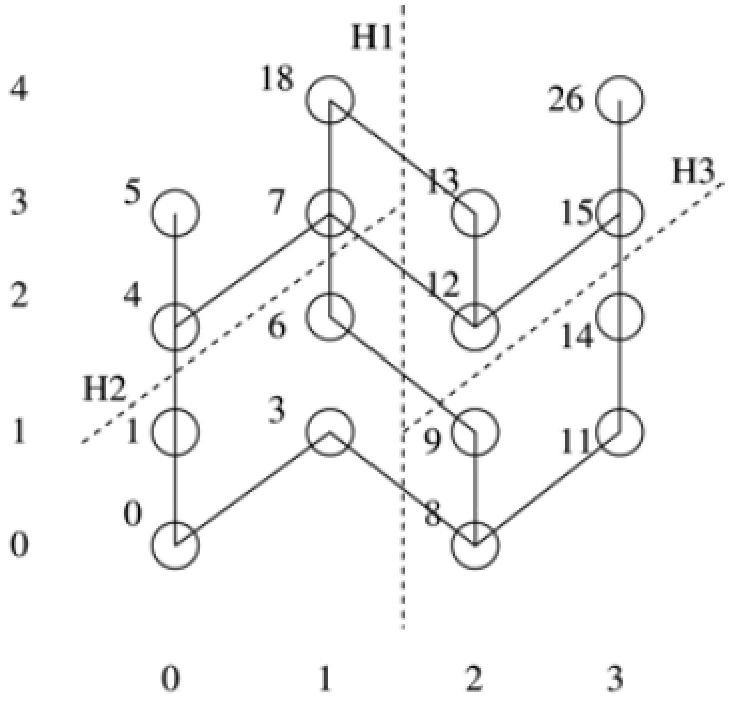
Z-order



Clustering methods - cell-tree

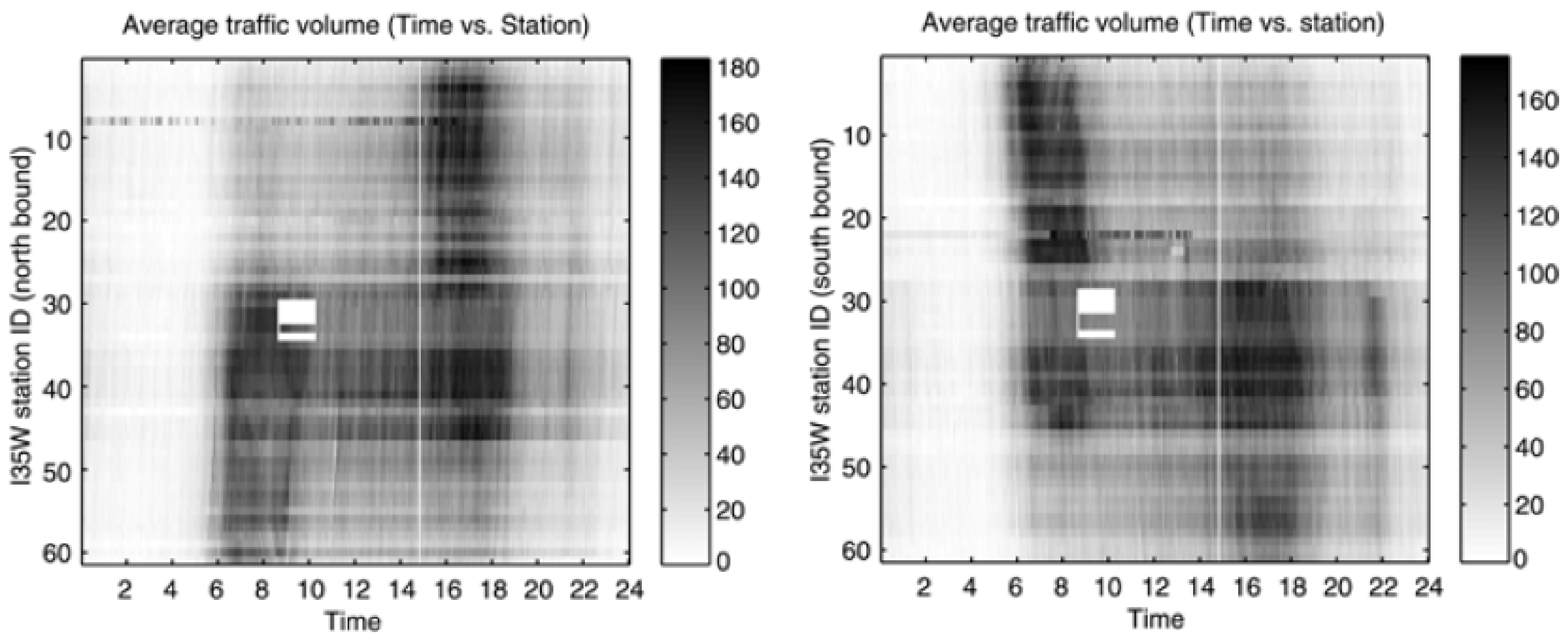
- Nodes in different partitions are
  - (0, 1, 3, 6)
  - (4, 5, 7, 18)
  - (8, 9, 11, 14)
  - (12, 13, 15, 26)

BSP partitioning



Buffering methods

- Of course, buffers matter too
  - FIFO
  - MRU
  - LRU
- Buffer size (page size)
  - 1k, 4k, 8k, ...
- Effect to model building
- Effect to ROD algorithm



(a) I-35W north bound

(b) I-35W south bound

- Outliers detected
  - White vertical line (14.45) – temporal outlier
  - White square (8.20-10.00) – spatial-temporal outlier
  - Station 9 – inconsistent traffic flow