Sequencing Geographical Data
For Efficient Query Processing On Air In Mobile Computing

Jianting Zhang
School of Computer Science
The University of Oklahoma
Chapter 1: Geographical Data Broadcasting
  • Spatial Range Query for Point Data
  • Network Path Query for graph Data

Chapter 2: Literature Review

Chapter 3: Cost Models of Access Time (DBW, AT_{Data}^{Sep}, AT_{Data}^{Mul})

Chapter 4: Hyper-Graph Representation

Chapter 5: Ordering Heuristics

Chapter 6: Optimization Methods

Chapter 7: Experiments and Evaluations

Chapter 8: Conclusions and Future Work


Outline

- Introduction
- Related Work
- Access Time Cost Models
- Hyper-graph Representation
- Ordering Heuristics
- Optimization Methods
- Experiments and Evaluations
- Conclusions and Future Work
Introduction

Different Types of Data Communication Networks

Technology

• Wired/Wireless
• Fixed/Mobile
• Point-to-Point/Broadcast
• Single-hop/Multiple hop

Data Broadcast/Broadcast: Single-Hop Digital Wireless
Introduction

Why Broadcasting

Key problems in Mobile Computing

- Bandwidth
  - Independent of number of users
  - Excellent scalability

- Power Consumption:
  - Listen/Sleep mode consumes less power than in send mode

- Mobility
  - No mobility management is required at neither server side nor client side
Introduction
Types of Broadcasting

- **Pull based**
  - Explicit client request
  - No unwanted data
  - Need scheduling

- **Push based**
  - Need prior knowledge
  - Schedule broadcast sequence without requests
  - Suitable for large number of users
Introduction
What is Geographical Information

• Spatial Range Query: Retrieve all the gas stations within 2 miles of my current location

• Network Path Query: Find the shortest path from OKC to Houston
Introduction

Application Areas

• Location Dependent Services
  – Potentially large number of users
  – Position Technologies: GPS-based, cellular, hybrid
  – Location dependent queries (LDQ)

• Unusual Event Monitoring
• Disaster Rescue
• Military Operations
Introduction
Disk Access vs. Air Access
The objectives of this study are to develop cost models and optimization algorithms in placing geographical data items onto a broadcast channel based on their spatial semantics to reduce the response time and energy consumption for processing spatial queries over the broadcast channel.
Outline

- Introduction
  - Related Work
- Access Time Cost Models
- Hyper-graph Representation
- Ordering Heuristics
- Optimization Methods
- Experiments and Evaluations
- Conclusions and Future Work
Related Work

• General Data Broadcasting:
  – Proposed in (Imielinski, 1993 SIGMOD Record)
  – Signature (Lee, 1996), Hybrid (Hu, 2001)
  – Broadcast Disk techniques based (Achrya, 1995; Shivakuma, 1996; Peng, 2000; Hsu, 2002)
  – Suitable for One-dimensional and/or categorical data
  – Allows only one data item per access
  – Focus on trade off between Tune-in Time and Access Time using replication
Related Work

• Object-Orientated (Chehadeh, 1999) / Relational Database Broadcasting (Si, 1999; Lee, 2002; Lee 2003)
  – Allows multiple data items per access
  – Assumes data access has predefined orders

• (Chunag 2001):
  – Most similar to our cost model under SEP scheme
  – Provide QEM scheduling method based on the concept of Query Distance (QD) and is greedy
  – Proof of QD and the average data access time is NOT correct
Related Work

• Geographical Data is multi-dimensional and continuous data.

• There may be multiple data items in a spatial query result set and they may not have a pre-defined order.

• Existing broadcast techniques cannot be applied to geographical data broadcast for efficient spatial query processing.
Outline

- Introduction
- Related Work
  - Access Time Cost Models
- Hyper-graph Representation
- Ordering Heuristics
- Optimization Methods
- Experiments and Evaluations
- Conclusions and Future Work
Cost Models

The four components

**IPW**: Index Probe Wait. (Time to meet the first index)

**IBW**: Index Bcast Wait. (Time to retrieve indexes)

**DPW**: Data Probe Wait. (Time to meet the first data item)

**DBW**: Data Bcast Wait. (Time to retrieve all data items)

IPW=Probe Wait  \( \text{IBW+DPW+DBW=Bcast Wait} \)

• Using the four components allows us to study the access time to data and index separately.

• We are interested in access time to data \((\text{DPW+DBW})\)

• DBW is more important than the other components
Cost Models

Using Separate Channel Scenario (SEP)

Index Broadcast Cycle

Data Broadcast Cycle

Pointers

Access Sequence

DPW

DBW

Data Broadcast Cycle

T1

T2

T3

T4

T5

IPW

IBW

Pointers

Access Sequence
Cost Models
Multiplexing Scenario (MUL)
Cost Models

Processing a Single Complex Query

Complex Query:

- Has more than one data items in a query result set
- Spatial Range Query and Network Path Query
Cost Models

Processing a Single Complex Query

\[ AT_{Data} = DPW + DBW \]

- Multiplexing Scheme:
  \[ AT_{Data}^{Mul} = L_1 + L_2 \]

- Using Separate Channels: three cases
  - During L_1
    \[ \text{Cost } 1 = \sum_{i=0}^{L_1-1} (L_1 - i + L_2) = \frac{L_1(L_1 + 1)}{2} + L_1 \times L_2 \]
  - During L_2
    \[ \text{Cost } 2 = \sum_{i=0}^{L_2-1} L = L \times L_2 \]
  - During L_3
    \[ \text{Cost } 3 = \sum_{i=0}^{L_3-1} (L_3 - i + L_1 + L_2) = \sum_{i=0}^{L_3-1} (L - i) = L_3 \times L - \frac{L_3(L_3 - 1)}{2} \]
Cost Models

Processing a Single Complex Query

\[ ATData_{\text{Sep}} = \frac{1}{L} (\text{Cost 1} + \text{Cost 2} + \text{Cost 3}) \]

\[ = \frac{1}{L} \left[ \frac{L_1(L_1 + 1)}{2} + L_1 \cdot L_2 + L \cdot L_2 + L_3 \cdot L - \frac{L_3(L_3 - 1)}{2} \right] \]

\[ = \frac{1}{L} \left[ \frac{(L_1 + L_3)(L_1 - L_3 + 1)}{2} + L_1 \cdot L_2 + L \cdot (L_2 + L_3) \right] \]

\[ = \frac{1}{L} \left[ \frac{(L - L_2)(L_1 - L_3 + 1)}{2} + L_1 \cdot L_2 + L \cdot (L_2 - L_1) \right] \]

\[ = \frac{1}{L} \left[ L^2 - L_1 \cdot (L - L_2) \right] + \left( \frac{L - L_2}{2} \right) \left( \frac{L_1 - L_3 + 1}{2} \right) \]

\[ = \frac{1}{L} \left[ L^2 - \frac{(L - L_2)(L_1 - L_3 + 1)}{2} \right] \]

\[ = \frac{1}{L} \left[ L^2 - \frac{(L - L_2)(L_1 + L_3 - 1)}{2} \right] \]

\[ = \frac{1}{L} \left[ L^2 - \frac{(L - L_2)(L - L_2 - 1)}{2} \right] \]

\[ \text{L}_2 < L, \text{ the average cost decreases monotonically as } L_2 \text{ decreases.} \]

\[ g(L_2) = \frac{1}{L} \left[ L^2 - \frac{(L - L_2)(L - L_2 - 1)}{2} \right] \]
Cost Models
Computing Access Frequencies of Point Data

**Purpose:** Computing Access Frequencies of all possible spatial range query results sets based on spatial semantics of point data sets to compute total access time

**Parameters:** $DS = [x_1, x_2] \times [y_1, y_2]$ Query Window $(q_x, q_y)$

**Assumption:** All the locations inside the DS have the same possibilities for users to request a location dependent spatial range query. The number of range queries that are requested within a region is proportional to its area and we use the area as the access frequency.
Cost Models

Computing Access Frequencies of Point Data

- **Extended Region** $R_u$ of point $P_u$: the rectangle of size $(q_x, q_y)$ centered at $P_u$.

- The distribution of the locations, where a user issues location dependent range queries of size $(q_x, q_y)$ and the results contain data item $P_u$, is the extended region $R_u$ of $P_u$.
Cost Models
Computing Access Frequencies of Point Data

• The distribution of the locations, where a user issues location dependent range queries of size \((q_x, q_y)\) and the results contain both data items \(P_u\) and \(P_v\), is the intersection of the extended regions \(R_u\) and \(R_v\) of \(P_u\) and \(P_v\) respectively.

• …

• Up to the intersection among all \(n\) extended regions
Cost Models
Computing Access Frequencies of Point Data

Let

\( A_i \) be the area of \( R_i \),
\( A_{i,j} \) be the intersection area of \( R_i \) and \( R_j \),
\( \ldots \),
\( A_{1,2,\ldots,n} \) be the intersection area of \( R_1, R_2 \ldots R_n \).

Let

\( \tilde{A}_i \) be the part of \( A_i \) that solely contains point \( P_i \),
\( \tilde{A}_{i,j} \) be the part of \( A_i \) that solely contains points \( P_i \) and \( P_j \),
\( \ldots \)
\( \tilde{A}_{1,2,\ldots,n} \) be the part of the intersection area of \( R_i, R_2 \ldots R_n \).
Let function $\pi(u)$ maps point $u$ to its position in a broadcast sequence.

The DBW for a single query result set that contains $k$ data items $n_1, n_2 \ldots n_k$.

$L_1 = \min\{\pi(n_1), \pi(n_2), \ldots \pi(n_k)\}$

$L_2 = \max\{\pi(n_1), \pi(n_2), \ldots \pi(n_k)\} - \min\{\pi(n_1), \pi(n_2), \ldots \pi(n_k)\}$
Cost Models

Spatial Range Query for Point Data: DBW

The total DBW cost for a query window \((q_x, q_y)\) is the summation of the weighted DBW for all possible query result sets, i.e.

\[
\text{Cost}^{(q_x, q_y)} = \sum_{1 \leq i < j \leq n} \tilde{A}_{ij}^{(q_x, q_y)} \times |\pi(i) - \pi(j)| + \sum_{1 \leq i < j \leq k \leq n} \tilde{A}_{i,j,k}^{(q_x, q_y)} \times [\max(\pi(i), \pi(j), \pi(k)) - \min(\pi(i), \pi(j), \pi(k))] \\
+ ... \\
+ \tilde{A}_{1,2,...,n}^{(q_x, q_y)} \times [\max(\pi(1), \pi(2)...\pi(n)) - \min(\pi(1), \pi(2)...\pi(n))]}
\]
Cost Models

Spatial Range Query for Point Data: DBW

The final total cost of DBW will be the summation of DBW over all possible query windows Q, i.e.,

\[
\text{Cost}_{DBW} = \sum_{(qx, qy) \in Q} \text{Cost}^{(q_x, q_y)}(q_x, q_y)
\]

\[
= \sum_{(qx, qy) \in Q \leq i < j \leq n} \sum_{i,j,k} \tilde{A}_{ij}^{(q_x, q_y)} \cdot |\pi(i) - \pi(j)|
\]

\[
+ \sum_{(qx, qy) \in Q \leq i < j \leq k \leq n} \sum_{i,j,k} \tilde{A}_{i,j,k}^{(q_x, q_y)} \cdot [\max(\pi(i), \pi(j), \pi(k)) - \min(\pi(i), \pi(j), \pi(k))]
\]

\[
+ \ldots
\]

\[
+ \sum_{(qx, qy) \in Q} \tilde{A}_{1,2,...n}^{(q_x, q_y)} \cdot [\max(\pi(1), \pi(2)...\pi(n)) - \min(\pi(1), \pi(2)...\pi(n))]
\]
Cost Models

Spatial Range Query for Point Data: DBW

\[ w_{i,j} = \sum_{(q_x,q_y) \in Q} \bar{A}_{i,j}(q_x,q_y) \]
\[ w_{i,j,k} = \sum_{(q_x,q_y) \in Q} \bar{A}_i(q_x,q_y) \]
\[ \ldots \]
\[ w_{1,2,\ldots,n} = \sum_{(q_x,q_y) \in Q} \bar{A}_{1,2,\ldots,n}(q_x,q_y) \]

\[ \text{Cost}^{DBW} = \]
\[ \sum_{1 \leq i < j \leq n} w_{i,j} \cdot |\pi(i) - \pi(j)| \]
\[ + \sum_{1 \leq i < j \leq k \leq n} w_{i,j,k} \cdot [\max(\pi(i),\pi(j),\pi(k)) - \min(\pi(i),\pi(j),\pi(k))] \]
\[ + \ldots \]
\[ + w_{1,2,\ldots,n} \cdot [\max(\pi(1),\pi(2)\ldots\pi(n)) - \min(\pi(1),\pi(2)\ldots\pi(n))] \]
Cost Models

Spatial Range Query for Point Data: $AT_{Data}^{Mul}$

\[ Cost^{AT_{Data}^{Mul}} = \]

\[ = \sum_{1 \leq i \leq n} w_i \ast \pi(i) \]

\[ + \sum_{1 \leq i < j \leq n} w_{i,j} \ast \max(\pi(i), \pi(j)) \]

\[ + \sum_{1 \leq i < j \leq k \leq n} w_{i,j,k} \ast \max(\pi(i), \pi(j), \pi(k)) \]

\[ + \ldots \]

\[ + w_{1,2,\ldots,n} \ast \max(\pi(1), \pi(2)\ldots\pi(n)) \]
Cost Models
Spatial Range Query for Point Data: $AT_{Data}^{Sep}$

\[
\begin{align*}
\text{Cost} \quad & = \\
& \sum_{1 \leq i < j \leq n} w_{i,j} \cdot g(|\pi(i) - \pi(j)|) \\
& + \sum_{1 \leq i < j \leq k \leq n} w_{i,j,k} \cdot g(\max\{\pi(i), \pi(j), \pi(k)\} - \min\{\pi(i), \pi(j), \pi(k)\}) \\
& + \ldots \\
& + w_{1,2,...,n} \cdot g(\max\{\pi(1), \pi(2),...\pi(n)\} - \min\{\pi(1), \pi(2),...\pi(n)\})
\end{align*}
\]
Cost Models

Path Query for Graph Data

V: the vertex set of the graph
S_{ij}: a path sequence from source vertex i to destination vertex j
f(i,j): access frequency of S_{ij}

\[
\text{Cost }^{DBW} = \sum_{i \in V, j \in V} f(i, j) \cdot [\max_k \left( \bigwedge_{m=0}^k \pi(S_{ij}^m) \right) - \min_k \left( \bigwedge_{m=0}^k \pi(S_{ij}^m) \right)]
\]

\[
\text{Cost }^{AT_{Data} \text{ Sel}} = \sum_{i \in V, j \in V} f(i, j) \cdot g \left( \max_k \left( \bigwedge_{m=0}^k \pi(S_{ij}^m) \right) - \min_k \left( \bigwedge_{m=0}^k \pi(S_{ij}^m) \right) \right)
\]

\[
\text{Cost }^{AT_{Data} \text{ Mul}} = \sum_{i \in V, j \in V} f(i, j) \cdot [\max_k \left( \bigwedge_{m=0}^k \pi(S_{ij}^m) \right)]
\]
Outline

- Introduction
- Related Work
- Access Time Cost Models
  - Hyper-graph Representation
- Ordering Heuristics
- Optimization Methods
- Experiments and Evaluations
- Conclusions and Future Work
Hyper-graph Representation

Point Data
Hyper-graph Representation

Graph Data
Hyper-Graph Representation

Relationship With MinLA

Graph Minimum Linear Arrangement problem

$$la(G) = \sum_{(u,v) \in E} w(u, v) | \pi(u) - \pi(v) |$$

Problems:

- Extending to hyper-graph

$$\max\{\pi(n_1), \pi(n_2), \ldots, \pi(n_k)\} - \min\{\pi(n_1), \pi(n_2), \ldots, \pi(n_k)\}$$

- Non-Linearity

$$g(L_2) = \frac{1}{L} \left[ L^2 - \frac{(L-L_2)(L-L_2-1)}{2} \right]$$
Outline

- Introduction
- Related Work
- Access Time Cost Models
- Hyper-graph Representation
  - Ordering Heuristics
- Optimization Methods
- Experiments and Evaluations
- Conclusions and Future Work
Ordering Heuristics

Overview

• Purposes:
  – Generating final orderings for broadcast if efficiency is the primary concern
  – Serving as the initial ordering for further optimization if quality is the primary concern

• Requirements
  – Low-computation complexity
  – Exploring existed structures in spatial databases if there are any
Ordering Heuristics

Classification Structure

Ordering Heuristics

Geometry-Based
- Hierarchical
  - Traversal of Spatial Clustering Trees
    - Quad-Tree
    - R-Trees
  - Traversal of Spatial Index Trees
    - Z-Ordering
    - Hilbert-Ordering
- Non-Hierarchical
  - Space Filling Curves

Graph-Based
- Hierarchical
  - Traversal of Graph Partition Trees
    - Spanning Tree
  - Graph Traversal
    - DFS
    - BFS
- Non-Hierarchical
  - Node Degree
  - Node Weight
  - Edge Weight
Outline

- Introduction
- Related Work
- Access Time Cost Models
- Hyper-graph Representation
- Ordering Heuristics
  - Optimization Methods
- Experiments and Evaluations
- Conclusions and Future Work
**Optimization**

**Overview**

Requirements:

- Low-Cost: near real time scheduling (e.g., sequencing 100-10000 nodes in 1-5 minutes)

- Trade off between Greedy and Non-greedy methods

Algorithm of (Bar-Yehuda, 2001):

- Divide-and-conquer strategy
- Space Complexity: $O(2^{\text{depth}(T)}) \rightarrow O(n)$

- Time Complexity: $\sum_{t \in T} 2^{\text{depth}(t)} \rightarrow O(n^2)$
• Propose to use (Bar-Yehuda, 2001) for DBW optimization
  – Prove the correctness of hyper-graph case
  – Revise the implementation to allow having weights
  – Propose to use R-Tree to generate BDT

• Propose to use (Bar-Yehuda, 2001) for AT\textsubscript{Data\,SEP} Optimization
  – Based on our proof of the monotonic relationship between DBW and AT\textsubscript{Data\,SEP} for a single complex query

• Present a novel approach for AT\textsubscript{Data\,MUL} Optimization
  – Adopts the same divide-and-conquer strategy and has low computation cost
  – Compute the access time directly (weight*position)
  – The position is computed efficiently by exploring the hyper-graph data structures
Optimization

The BDT

1-Orientation

0-Orientation

Number of Possible Orderings $2^{n-1}$ a BDT is full and balanced

Orientation tree (or_tree): The orientations at each intermediate node of the BDT form a tree that has the same structure as the BDT
Optimization

The BDT

(a)  (b)  {0,1,2}: 13  
    {0,2,1}: 11  
    {1,2,0}: 11  
    {2,1,0}: 13  

(c)  (d)  {2,1,0}: 13  
    {2,0,1}: 12  
    {1,0,2}: 12  
    {0,1,2}: 13  

\[
\begin{align*}
\{0,1,2\} & : 13 \\
\{0,2,1\} & : 11 \\
\{1,2,0\} & : 11 \\
\{2,1,0\} & : 13 \\
\end{align*}
\]
Optimization
Generating BDT form R-Tree

Spatially adjacent points are likely to be queried together ➔
The hyper-edges contain such points have larger weight ➔
Putting them close to each other will reduce access time ➔
Generating BDT based on R-Tree
Optimization

DBW

• (Bar-Yehuda, 2001) recursively tests the two possible orientations of a BDT and choose the better one.

• It uses position implicitly in computing the cost (access time in our applications) which is very efficient.

\[
\text{cost}_0 = \text{cost}(\text{left}(0)) + \text{cost}(\text{right}(0)) \\
+ |V(t_2)| \cdot \text{cost}((\text{right cut}(\text{or tree}(t_1)))) \\
+ |V(t_1)| \cdot \text{cost}((\text{left cut}(\text{or tree}(t_2))))
\]

\[
\text{cost}_1 = \text{cost}(\text{left}(1)) + \text{cost}(\text{right}(1)) \\
+ |V(t_1)| \cdot \text{cost}((\text{right cut}(\text{or tree}(t_2)))) \\
+ |V(t_2)| \cdot \text{cost}((\text{left cut}(\text{or tree}(t_1))))
\]
Optimization

DBW

cost(left_cut(or_tree(t))) = cost(left_cut(left(or_tree(t)))) + cost(left_cut(right(or_tree(t)))) - cost(in_cut(t))

cost(right_cut(or_tree(t))) = cost(right_cut(left(or_tree(t)))) + cost(right_cut(right(or_tree(t)))) - cost(in_cut(t))

In_cut: summation of the weights of edges the beginning and ending nodes of which are within sub-tree t

Left_cut: summation of the weights of edges the beginning node of which is not within sub-tree t while the ending node of which is within sub-tree t

Right_cut: summation of the weights of edges the beginning node of which is within sub-tree t while the ending node of which is not within sub-tree t
Optimization

**DBW: proof correctness for hyper-graph case**

(Bar-Yehuda, 2001):

- The algorithm presented in the paper is only applicable for regular graph
- We need to prove the applicability of the algorithm to a hyper-graph

Proof: To prove the value computed by the formula equals the hyper-graph version of $\lambda(G)$ it is sufficient to prove that for any hyper-edge $e$, the cost computed by the formula equals

$$w(e)^* \mid \pi(u) - \pi(v) \mid$$
Optimization

DBW: proof correctness for hyper-graph case

t₀: root node of the least common ancestor of all nodes of e
u: left-most node of e
v: right-most node of e
x: right-most node of left sub-tree t₁
y: left-most node of right-sub-tree t₂
π(y) - π(x) = 1

Fig. 6-4. The BDT Structure in an Ordering Sequence for an Edge
Optimization

DBW: proof correctness for hyper-graph case

Suppose the nodes on the path from $t_1$ to $u$ are $L_1, \ldots L_{k-2}, L_{k-1}, L_k$, and the sizes of the right sub-trees of the trees having $L_1, \ldots L_{k-2}, L_{k-1}, L_k$ as the root nodes are $p_1, \ldots p_{k-2}, p_{k-1}, p_k$, respectively.

$$p_1 + p_2 + \ldots + p_k = \pi(x) - \pi(u)$$

$$\text{cost}_0 = \text{cost}(\text{left}(0)) + \text{cost}(\text{right}(0)) + |V(t_2)| \cdot \text{cost}(	ext{right_cut}(	ext{or_tree}(t_1))) + |V(t_1)| \cdot \text{cost}(	ext{left_cut}(	ext{or_tree}(t_2))))$$

Expand completely and examine which terms involve $w(e)$

• appear once as the tree cost when the sub-tree is a leaf node
• appear in the right outer cuts from or_tree($t_1$) to or_tree($t_2$)
• appear as the left outer cuts from or_tree($t_2$) to or_tree($t_1$)
Optimization

DBW: proof correctness for hyper-graph case

Examine left outer cuts:

• Observe that only sub-trees that contain node $u$ can contribute costs in terms of $w(e)$ to the left outer cuts of node $v$.

• In formula $\text{cost}_0 = \text{cost}(\text{left}(0)) + \text{cost}(\text{right}(0)) + |V(t_2)| \cdot \text{cost}(\text{right_cut(or_tree(t_1)))} + |V(t_1)| \cdot \text{cost}(\text{left_cut(or_tree(t_2)))}$ only $\text{cost}(\text{left}(0))$ and $\text{left_cut(or_tree(t_2))}$ can contribute to the left outer cuts of node $v$.

• Cost($\text{left}(0)$) can contribute to the left outer cuts of node $v$ because when it is computed recursively another level down, the left outer cuts of node $v$ with regard to $e$ will appear.
Optimization

**DBW: proof correctness for hyper-graph case**

- We are only concerning costs with regard to $e$, thus $\text{left\_cut(\text{or\_tree}(t_2))}$ with regard to $e$ is $w(e)$

- At $t_1$, $|V(t_1)| \cdot \text{cost(\text{left\_cut(\text{or\_tree}(t_2)))} = |L_1| \cdot w(e)$

- Continue the recursion process till leaf node $u$ is reached, the total left outer cuts with regard to $e$ is

$$|L_k| \cdot w(e) + |L_{k-1}| \cdot w(e) + \ldots + |L_1| \cdot w(e)$$

$$= p_k \cdot w(e) + p_{k-1} \cdot w(e) + \ldots + p_1 \cdot w(e)$$

$$= (p_1 + p_2 + \ldots + p_k) \cdot w(e)$$

$$= [\pi(x) - \pi(u)] \cdot w(e)$$
Optimization

**DBW: proof correctness for hyper-graph case**

• Similarly, the total right outer cuts with regard to e is
  \[ \pi(v) - \pi(y) \] \cdot w(e) \]

• w(e) will appear once as the tree (leaf node) cost

• The total cost with respect to w(e)

\[ \pi(x) - \pi(u) \] \cdot w(e) + \pi(v) - \pi(y) \] \cdot w(e) + w(e) 
\[= \pi(v) - \pi(u) \] \cdot w(e) + \pi(x) - \pi(y) \] \cdot w(e) + w(e) 
\[= \pi(v) - \pi(u) \] \cdot w(e) 

#
Optimization

Approximating $\text{AT}_{\text{Data}}^{\text{Sep}}$

- The monotonic relationship between $L_2$ and $g(L_2)$.

$$g(L_2) = \frac{1}{L} \left[ L^2 - \frac{(L - L_2)(L - L_2 - 1)}{2} \right]$$

- $L_2$ is a good linear approximation of $g(L_2)$.

- Use the optimized ordering for DBW as the ordering for $\text{AT}_{\text{Data}}^{\text{Sep}}$

- Expect that the optimized ordering where the optimization is based on the definition of $la(G)$ which is linear with respect to $L_2$, is also a good ordering according to quadratic cost model respect to $L_2$. 
Optimization
Approximating $AT_{Data}^{Sep}$

How good is the approximation?

Answer from the example:
The algorithm:

1. Set the positions of all nodes to the specified initial order, or the natural order of \{1,2,…,n\} if no initial order is available. Set the starting BDT node \( t \) to the root of BDT. Do step 2 and step 3 recursively.

2. If \( t \) is an intermediate node of BDT:
   - a) Test the two orientations of the sub-trees \( t_1 \) and \( t_2 \) and add the access times of \( t_1 \) and \( t_2 \) under the orientations.
   - b) Set the orientation of \( t \) to the one that has less access time.
3 If $t$ is a leaf node of BDT:

a) Set the access time associated with the node to zero.

b) Compute the position of the node in the current sequence.

c) Retrieve all the queries that contain this node and their corresponding weights using the hyper-graph data structures.

d) For each query that has the node as the ending node in the broadcast sequence, add position*weight to the access time associated with the node.
Augment each of the BDT node with a pointer pointing to its parent.
Find the path from a leaf node all the way to the root by using the pointers.
Compute the position of a node under a BDT: counting the number of all the nodes that are to the left of the path,

\[ 1 + 3 = 4 \]
Determine whether a node is the ending node of a hyper-edge

- For each hyper-edge we pre-build a Least Common Ancestor (LCA) tree.
- Each LCA tree node is a pointer pointing to a BDT node, which stores the orientation information.
- Start with the root of the LCA tree of the node, follow the right children until a leaf node is reached.
- Compare the leaf node’s ID with the ID of the BDT node.
Let \( S(n) \) be the total computation cost for an \( n \)-nodes hyper-graph.

Assuming the corresponding BDT \( T \) is balanced.

At each \( t \in T \) having \( n \) nodes

- Calculate the costs of its two children under the two orientations: \( 4*S(n/2) \)
- One addition for each orientation
- One comparison

\[
S(n) = 4 * S\left(\frac{n}{2}\right) + 3
\]
\[
= 4 * [4 * S\left(\frac{n}{2^2}\right) + 3] + n + 3
\]
\[
= 4^2 * S\left(\frac{n}{2^2}\right) + 4 * 3 + 3
\]
\[
= ... 
\]
\[
= 4^k * S\left(\frac{n}{2^k}\right) + ... + 4 * 3 + 3 
\]
\[
= 4^k * S(1) + \left(4^k + 4^{k-1} + ... 1\right) * 3 
\]
\[
= \left(2^k\right)^2 * S(1) + \frac{4^{k+1} - 1}{4 - 1} * 3 
\]
\[
= n^2 * S(1) + 4 * n^2 - 1 
\]
What’s in $S(1)$

- $O(\log n)$ to compute the position of a node in Step 3.b.
  (For practical $n$ values, e.g., 100-10000, $\log(n)$ can be treated as a constant).
- Constant time to retrieve all the hyper-edges that contain the node by using the inverse hyper-graph in Step 3.c
- Constant time to determine whether a node is the ending node of a hyper-edge in Step 3.d

$\Rightarrow S(1)$ is approximately constant

$\Rightarrow$ The algorithm is approximately $O(n^2)$
Optimization

$AT_{Data Mul}$ Example

\[ AT(2) = 2 \times (4 + 38 + 19 + 3) = 128 \]
\[ AT(1) = 3 \times (2 + 14 + 22 + 62) = 300 \]
\[ AT(T_{11}) = 128 + 300 = 428 \]
Optimization

**AT\_Data\_Mul** Example

\[
\begin{align*}
AT(1) &= 2 \times (2 + 62) = 128 \\
AT(2) &= 3 \times (4 + 38 + 14 + 22 + 19 + 3) = 300 \\
AT(T_{11}^0) &= 128 + 300 = 428
\end{align*}
\]
Optimization

$AT_{Data}^{Mul}$ Example

$AT(3) = 1 \times (8 + 34) = 42$

$AT(4) = 0$

$AT(T_{12}^{-1}) = 42 + 0 = 42$

$AT(T_{12}^0) = 93 + 0 = 93$

$AT(T_0^{-1}) = 428 + 42 = 470$
Optimization

\[ AT_{Data}^{Mul} \] Example

\( T_{11} \): 1-orientation 428, **0-orientation 428**
\( T_{12} \): **1-orientation 42**, 0-orientation 93
\( T_0 = 1\)-orientation 470, [4,3,1,2]

\( T_{11} \): 1-orientation 84, **0-orientation 41**
\( T_{12} \): 1-orientation 476, **0-orientation 476**
\( T_0 = 0\)-orientation 517 [1,2,3,4]

Initial ordering [1,2,3,4] **517**
Optimized ordering [4,3,1,2] **470**
Experiments and Evaluations

Software Modules

Diagram showing software modules and their relationships:
- Point Data Set
- Graph Data Set
- Hyper-graph generation
- Hilbert SPC Ordering
- R-Tree
- R-Tree to BDT conversion
- Traversal Ordering
- Hyper-Graph Partition
- Graph-Partition
- BDT
- Evaluation
- Optimization
- Floyd-Warshall AP-SP
- EAFN Generation
- NODE/EDGE WEIGHT *
- BFS/DFS
- MAX/MAX-LD *
- Prim/Kruskal Spanning Tree *
- * MUL Only
Experiments and Evaluations

Data Sets and Performance Metrics

- Five random point data sets
- 51 real point data sets: the centers of the zip codes among the 50 states and the District of Columbia of the United States
- Real graph data set: the transportation network of the State of Texas
- All experiments are performed on a Dell Dimension 4100 personal computer with 866 MHZ Intel processor and 512M memory under the Windows 2000 professional operating system
- Normalized Access Time (NAT, for DBW, AT^{MUL}_{Data}, AT^{SEP}_{Data})

\[
NAT = \frac{AT_{Data}}{W}
\]
Experiments and Evaluations

Synthetic Data Sets

- Number of points: 100, 200, 300, 400, 500
- Data Space \([0, 1) \times [0, 1)\)
- Query window: of 0.1 by 0.1
- Reports access time of the three cost models under six measurements
  - the minimum of 1000 random orderings
  - the maximum of 1000 random orderings
  - the average of 1000 random orderings
  - the Hilbert SFC ordering
  - the R-Tree traversal ordering
  - the optimized R-Tree traversal ordering
Experiments and Evaluations

Synthetic Data Sets

Table 7-2. Results of 1000 Random Orderings Under DBW Cost Model for Synthetic Data Sets

| Data Set | Minimum AT (Rand _ Min) | Maximum AT (Rand _ Max) | Average AT (Rand _ Ave) | Improvement
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31.67</td>
<td>47.8</td>
<td>40.19</td>
<td>40.13%</td>
</tr>
<tr>
<td>2</td>
<td>81.02</td>
<td>102.24</td>
<td>92.86</td>
<td>22.85%</td>
</tr>
<tr>
<td>3</td>
<td>142.27</td>
<td>171.42</td>
<td>154.95</td>
<td>18.81%</td>
</tr>
<tr>
<td>4</td>
<td>200.15</td>
<td>239.73</td>
<td>222.11</td>
<td>17.82%</td>
</tr>
<tr>
<td>5</td>
<td>272.65</td>
<td>317.27</td>
<td>297.09</td>
<td>15.02%</td>
</tr>
</tbody>
</table>

- The improvements of 1000 random orderings drop from 40% to 15%
- Finding a good ordering by examining a fixed number of random orderings is not a feasible solution
Experiments and Evaluations

Synthetic Data Sets

Table 7-3. Comparisons of Hilbert and R-Tree Traversal Ordering With 1000 Random Orderings Average Under DBW Cost Model for Synthetic Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Rand-Ave</th>
<th>Hilbert Ordering (HO)</th>
<th>R-Tree Ordering (RO)</th>
<th>Hilbert Ordering Improvement $\frac{\text{Rand} - \text{Ave}}{\text{HO}}$</th>
<th>R-Tree Ordering Improvement $\frac{\text{Rand} - \text{Ave}}{\text{RO}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40.19</td>
<td>41.47</td>
<td>27.06</td>
<td>-3.09%</td>
<td>48.52%</td>
</tr>
<tr>
<td>2</td>
<td>92.86</td>
<td>94.15</td>
<td>63.04</td>
<td>-1.37%</td>
<td>47.30%</td>
</tr>
<tr>
<td>3</td>
<td>154.95</td>
<td>152.31</td>
<td>94.87</td>
<td>1.73%</td>
<td>63.33%</td>
</tr>
<tr>
<td>4</td>
<td>222.11</td>
<td>211.08</td>
<td>135.93</td>
<td>5.23%</td>
<td>63.40%</td>
</tr>
<tr>
<td>5</td>
<td>297.09</td>
<td>294.84</td>
<td>178.86</td>
<td>0.76%</td>
<td>66.10%</td>
</tr>
</tbody>
</table>

• Hilbert SFC ordering may be better or worse than the 1000 random orderings average.

• The R-Tree traversal orderings improve the 1000 random orderings significantly, from 47% to 66%
Experiments and Evaluations

Synthetic Data Sets

Table 7-4. Comparison of Optimized Ordering, R-Tree Ordering and 1000 Random Orderings Average Under DBW Cost Model for Synthetic Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Rand-Ave</th>
<th>R-Tree Ordering (RO)</th>
<th>Optimized R-Tree Ordering (OO)</th>
<th>R-Tree Improvement ( \frac{\text{Rand Ave}}{\text{RO}} - 1 )</th>
<th>Opt-Improvement ( \frac{\text{RO}}{\text{OO}} - 1 )</th>
<th>Overall Improvement ( \frac{\text{Rand Ave}}{\text{OO}} - 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40.19</td>
<td>27.06</td>
<td>22.7</td>
<td>48.52%</td>
<td>19.29%</td>
<td>77.05%</td>
</tr>
<tr>
<td>2</td>
<td>92.86</td>
<td>63.04</td>
<td>52.77</td>
<td>47.30%</td>
<td>19.45%</td>
<td>75.97%</td>
</tr>
<tr>
<td>3</td>
<td>154.95</td>
<td>94.87</td>
<td>63.07</td>
<td>63.33%</td>
<td>50.43%</td>
<td>145.68%</td>
</tr>
<tr>
<td>4</td>
<td>222.11</td>
<td>135.93</td>
<td>101.21</td>
<td>63.40%</td>
<td>34.30%</td>
<td>119.45%</td>
</tr>
<tr>
<td>5</td>
<td>297.09</td>
<td>178.86</td>
<td>121.9</td>
<td>66.10%</td>
<td>46.72%</td>
<td>143.72%</td>
</tr>
</tbody>
</table>

- Overall improvement of the optimized ordering to the 1000 random ordering average varies from 76% to 146%.
- Both R-Tree traversal ordering heuristic and the optimization method are effective.
Experiments and Evaluations

Synthetic Data Sets

SEP

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Rand-Ave</th>
<th>R-Tree Ordering (RO)</th>
<th>Optimized R-Tree Ordering (OO)</th>
<th>R-Tree Improvement $\frac{\text{Rand} - \text{Ave}}{\text{RO}}$</th>
<th>Opt-Improvement $\frac{\text{RO}}{\text{OO}}$</th>
<th>Overall Improvement $\frac{\text{Rand} - \text{Ave}}{\text{OO}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>77.09</td>
<td>69.03</td>
<td>65.64</td>
<td>11.68%</td>
<td>5.16%</td>
<td>17.44%</td>
</tr>
<tr>
<td>2</td>
<td>165.28</td>
<td>145.76</td>
<td>141.40</td>
<td>13.39%</td>
<td>3.08%</td>
<td>16.89%</td>
</tr>
<tr>
<td>3</td>
<td>252.63</td>
<td>214.79</td>
<td>197.47</td>
<td>17.62%</td>
<td>8.77%</td>
<td>27.93%</td>
</tr>
<tr>
<td>4</td>
<td>338.24</td>
<td>287.20</td>
<td>269.45</td>
<td>17.77%</td>
<td>6.59%</td>
<td>25.53%</td>
</tr>
<tr>
<td>5</td>
<td>415.65</td>
<td>352.06</td>
<td>319.79</td>
<td>18.06%</td>
<td>10.09%</td>
<td>29.98%</td>
</tr>
</tbody>
</table>

MUL

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Rand-Ave</th>
<th>R-Tree Ordering (RO)</th>
<th>Optimized R-Tree Ordering (OO)</th>
<th>R-Tree Improvement $\frac{\text{Rand} - \text{Ave}}{\text{RO}}$</th>
<th>Opt-Improvement $\frac{\text{RO}}{\text{OO}}$</th>
<th>Overall Improvement $\frac{\text{Rand} - \text{Ave}}{\text{OO}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>67.62</td>
<td>60.65</td>
<td>58.3</td>
<td>11.49%</td>
<td>4.03%</td>
<td>15.99%</td>
</tr>
<tr>
<td>2</td>
<td>145.89</td>
<td>131.84</td>
<td>128.15</td>
<td>10.66%</td>
<td>2.68%</td>
<td>13.84%</td>
</tr>
<tr>
<td>3</td>
<td>223.80</td>
<td>191.01</td>
<td>167.86</td>
<td>17.17%</td>
<td>13.79%</td>
<td>33.33%</td>
</tr>
<tr>
<td>4</td>
<td>301.38</td>
<td>278.96</td>
<td>248.11</td>
<td>8.04%</td>
<td>12.43%</td>
<td>21.47%</td>
</tr>
<tr>
<td>5</td>
<td>373.93</td>
<td>302.26</td>
<td>269.69</td>
<td>23.71%</td>
<td>12.08%</td>
<td>38.65%</td>
</tr>
</tbody>
</table>

- Similar patterns as in DBW
- R-Tree heuristic: 2/3
- Optimization method: 1/3
Experiments and Evaluations

Zip code Data Sets
# Experiments and Evaluations

## Zip code Data Sets: Access Time

<table>
<thead>
<tr>
<th>Cost Model</th>
<th>Window Size</th>
<th>Rand _Ave</th>
<th>HO</th>
<th>RO</th>
<th>OO</th>
<th>( \frac{Rand_Ave}{RO} )</th>
<th>( \frac{RO}{OO} )</th>
<th>( \frac{Rand_Ave}{OO} )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBW</strong></td>
<td>0.05</td>
<td>158.56</td>
<td>157.93</td>
<td>61.02</td>
<td>42.07</td>
<td>160.19%</td>
<td>51.81%</td>
<td>290.33%</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>178.07</td>
<td>177.30</td>
<td>77.84</td>
<td>57.58</td>
<td>144.75%</td>
<td>44.76%</td>
<td>254.83%</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>330.23</td>
<td>326.75</td>
<td>153.78</td>
<td>114.38</td>
<td>130.03%</td>
<td>42.62%</td>
<td>229.57%</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>419.02</td>
<td>418.95</td>
<td>224.95</td>
<td>177.96</td>
<td>89.10%</td>
<td>30.03%</td>
<td>144.89%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>141.78</td>
<td>140.84</td>
<td>95.35</td>
<td>67.26</td>
<td>46.92%</td>
<td>43.32%</td>
<td>109.26%</td>
</tr>
<tr>
<td><strong>SEP</strong></td>
<td>0.05</td>
<td>282.15</td>
<td>282.14</td>
<td>218.02</td>
<td>206.12</td>
<td>27.66%</td>
<td>6.65%</td>
<td>36.05%</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>309.92</td>
<td>309.48</td>
<td>245.46</td>
<td>233.22</td>
<td>26.52%</td>
<td>6.09%</td>
<td>34.20%</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>548.01</td>
<td>546.07</td>
<td>438.52</td>
<td>414.86</td>
<td>25.50%</td>
<td>6.24%</td>
<td>33.31%</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>637.31</td>
<td>637.32</td>
<td>526.39</td>
<td>499.88</td>
<td>20.52%</td>
<td>5.82%</td>
<td>27.49%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>175.59</td>
<td>175.37</td>
<td>155.09</td>
<td>141.87</td>
<td>11.47%</td>
<td>9.11%</td>
<td>21.56%</td>
</tr>
<tr>
<td><strong>MUL</strong></td>
<td>0.05</td>
<td>250.65</td>
<td>250.31</td>
<td>201.66</td>
<td>192.59</td>
<td>23.50%</td>
<td>6.07%</td>
<td>30.89%</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>276.09</td>
<td>275.45</td>
<td>224.71</td>
<td>216.07</td>
<td>23.43%</td>
<td>5.46%</td>
<td>30.14%</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>489.8</td>
<td>488.09</td>
<td>398.75</td>
<td>377.27</td>
<td>22.55%</td>
<td>7.43%</td>
<td>31.62%</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>572.85</td>
<td>572.04</td>
<td>473.57</td>
<td>453.62</td>
<td>20.11%</td>
<td>7.35%</td>
<td>28.84%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>162.37</td>
<td>161.29</td>
<td>134.85</td>
<td>124.27</td>
<td>20.27%</td>
<td>14.49%</td>
<td>38.28%</td>
</tr>
</tbody>
</table>
Experiments and Evaluations

Zip code Data Sets: Access Time

• The results from the zip code data sets are pretty much similar to that from the five synthetic data sets

• For the 51 data sets, on average, the optimized orderings are better than 1000 times random ordering average 1.09 to 2.90 times under DBW, 22% to 36% under SEP and 30% to 38% under MUL, respectively

• Both the R-Tree traversal heuristic and the optimization method are effective which makes the orderings based on them query efficient
Experiments and Evaluations

Zip code Data Sets: Computation Time

Computation Time for DBW/AT\textsubscript{Data}\textsuperscript{Sep} Optimization Method of Zip Code Data Sets
Experiments and Evaluations

Zip code Data Sets : Computation Time

Computation Time for \( AT_{Data}^{Mul} \) Optimization Method For Zip Code Data Sets
Experiments and Evaluations

Zip code Data Sets: Computation Time

• The computation times are generally quadratic with respect to the number of nodes in a hyper-graph. They support the theoretical result of DBW\/ AT_{Data}^{Sep} optimization method given in (Bar-Yehuda, 2001) and our AT_{Data}^{Mul} optimization method very well.

• The computation time for our AT_{Data}^{Mul} optimization method is about 2/3 of that of the DBW/AT_{Data}^{Sep} optimization method.

• The computation time is not only related to the number of nodes, but also the number of hyper-edges. The hidden const factor behind the big O notation is important in practical applications.
Experiments and Evaluations

Zip code Data Sets: Computation Time

• Geographical data changes relatively slow in practice. For many of them, the proposed optimization methods are fast enough to meet practical needs. By using more powerful processors, the computation time for optimizations can be further reduced.

• Due to the divide-and-conquer nature of the optimization methods, it is also possible to explore parallelism to further reduce the computation time.
Experiments and Evaluations

Texas Road Network Data Set

- Manually Input from an AAA map
- Original network: 62 nodes, 120 edges
- PQ: 1891 AP-SP (hyper-edges)
- RQ: Query window size: 100 by 100 miles, 420 hyper-edges
Experiments and Evaluations

Texas Road Network: DBW

- Heuristics: random, graph-based, geometry-based
- Optimization: graph partition tree is used as BDT for optimizing network path query while R-tree decomposition is used as BDT for optimizing spatial range query
- Graph-based heuristic and optimized orderings are then evaluated on spatial range queries, and, geometry-based heuristic and optimized orderings are then evaluated on network path query
# Experiments and Evaluations

## Texas Road Network: DBW

<table>
<thead>
<tr>
<th>Orderings</th>
<th>Path Query</th>
<th>Range Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-Rand-min</td>
<td>35.55</td>
<td>26.68</td>
</tr>
<tr>
<td>N-Rand-max</td>
<td>45.10</td>
<td>34.25</td>
</tr>
<tr>
<td>N-Rand-Avg</td>
<td>40.79</td>
<td>30.46</td>
</tr>
<tr>
<td>N-EAFN-BFS-min</td>
<td>35.16</td>
<td>21.58</td>
</tr>
<tr>
<td>N-EAFN-BFS-max</td>
<td>42.57</td>
<td>32.24</td>
</tr>
<tr>
<td>N-EAFN-BFS-avg</td>
<td>39.35</td>
<td>27.97</td>
</tr>
<tr>
<td>N-EAFN-DFS-min</td>
<td>32.54</td>
<td>18.58</td>
</tr>
<tr>
<td>N-EAFN-DFS-max</td>
<td>40.23</td>
<td>27.93</td>
</tr>
<tr>
<td>N-EAFN-DFS-avg</td>
<td>37.31</td>
<td>23.50</td>
</tr>
<tr>
<td>Traversal of original graph partition tree</td>
<td>30.69</td>
<td>21.33</td>
</tr>
<tr>
<td>Optimization based on original graph partition tree</td>
<td>22.56</td>
<td>20.04</td>
</tr>
<tr>
<td>Traversal of EAFN partition tree</td>
<td>26.37</td>
<td>20.61</td>
</tr>
<tr>
<td>Optimization based on EAFN partition tree</td>
<td>22.26</td>
<td>21.16</td>
</tr>
<tr>
<td>Traversal of hyper-graph partition</td>
<td>24.25</td>
<td>25.19</td>
</tr>
<tr>
<td>Optimization based on hyper-graph partition tree</td>
<td>22.74</td>
<td>18.69</td>
</tr>
<tr>
<td>Hilbert</td>
<td>38.63</td>
<td>25.61</td>
</tr>
<tr>
<td>Traversal of R-Tree –min</td>
<td>35.73</td>
<td>17.76</td>
</tr>
<tr>
<td>Traversal of R-Tree –max</td>
<td>39.97</td>
<td>25.92</td>
</tr>
<tr>
<td>Traversal of R-Tree –avg</td>
<td>37.99</td>
<td>21.24</td>
</tr>
<tr>
<td>Optimization R-Tree –min</td>
<td>33.53</td>
<td>13.31</td>
</tr>
<tr>
<td>Optimization R-Tree –max</td>
<td>35.87</td>
<td>22.41</td>
</tr>
<tr>
<td>Optimization R-Tree –avg</td>
<td>34.30</td>
<td>13.49</td>
</tr>
</tbody>
</table>
Experiments and Evaluations

Texas Road Network: DBW

• For network path queries, traversal of the graph (original network, EAFN and the hyper-graph) partition tree orderings and their optimized orderings achieve much better results than both the graph traversal orderings and geometric based heuristic orderings.

• Among these orderings, traversal of the hyper-graph partition tree ordering as an ordering heuristic is the best. The optimized ordering based on EAFN partition is the best among the three optimized orderings although they are pretty close.

• For spatial range queries, R-Tree traversal orderings and their optimized orderings are significantly better than graph-based heuristics and the corresponding optimized orderings.
Experiments and Evaluations

Texas Road Network: $A T_{Data}^{Mul}$

• MAX, MAX-LD, NODE-WEIGHT and EDGE-WEIGHT heuristics are applicable only under $A T_{Data}^{Mul}$ cost model

• Three types of graphs: original network, hyper-graph from all-pair-shortest paths and the derived EAFN.

• While MAX and MAX-LD heuristics cannot be extended to hyper-graph, the NODE-WEIGHT and EDGE-WEIGHT heuristics can be used for both a regular graph and a hyper-graph.

• When performing optimizations, the original network and the EAFN are used to generate the BDTs and then used for optimization, while the hyper-graph is still used for computing the access time in all the three optimizations
# Experiments and Evaluations

**Texas Road Network: $AT_{Data}^{Mul}$**

## Table 7-13. Summary of Results Under $AT_{Data}^{Mul}$ Cost Model

<table>
<thead>
<tr>
<th>Orderings</th>
<th>ORGN</th>
<th>EAFN</th>
<th>Hyper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000-Rand-min</td>
<td>47.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000-Rand-max</td>
<td>53.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000-Rand-Avg</td>
<td>51.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-BFS-min</td>
<td>47.78</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>N-BFS-max</td>
<td>53.49</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>N-BFS-avg</td>
<td>50.41</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>N-DFS-min</td>
<td>48.67</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>N-DFS-max</td>
<td>53.07</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>N-DFS-avg</td>
<td>51.35</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>N-Prim-MST-Min</td>
<td>47.89</td>
<td>48.67</td>
<td>N/A</td>
</tr>
<tr>
<td>N-Prim-MST-Max</td>
<td>53.09</td>
<td>52.18</td>
<td>N/A</td>
</tr>
<tr>
<td>N-Prim-MST-Avg</td>
<td>49.56</td>
<td>50.47</td>
<td>N/A</td>
</tr>
<tr>
<td>Kruskal-MST</td>
<td>49.61</td>
<td>47.01</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX</td>
<td>47.79</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>MAX-LD</td>
<td>47.64</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>NODE-WEIGHT</td>
<td>52.28</td>
<td>52.00</td>
<td>52.43</td>
</tr>
<tr>
<td>EDGE-WEIGHT</td>
<td>50.91</td>
<td>53.15</td>
<td>51.05</td>
</tr>
<tr>
<td>Traversal of partition tree</td>
<td>44.38</td>
<td>47.79</td>
<td>45.01</td>
</tr>
<tr>
<td>Optimization based on partition tree</td>
<td>41.76</td>
<td>41.31</td>
<td>41.20</td>
</tr>
<tr>
<td>Hilbert</td>
<td>50.71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traversal of R-Tree –min</td>
<td>48.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traversal of R-Tree –max</td>
<td>51.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traversal of R-Tree –avg</td>
<td>50.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization R-Tree –min</td>
<td>46.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization R-Tree –max</td>
<td>48.79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization R-Tree –avg</td>
<td>47.56</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experiments and Evaluations

Texas Road Network: $A T_{Data}^{Mul}$

- Graph partition based heuristic orderings and optimized orderings remain among the best orderings under $A T_{Data}^{Mul}$ cost model.

- The Kruskal-MST heuristic on the EAFN, the MAX and the MAX-LD heuristics on the original network/EAFN are slightly better than the rest of the ordering heuristics.

- Although they are slightly worse than 1000-rand-min, are better than 1000-rand-average. Since generating a large number of random orderings could be expensive, these heuristics may be preferable.
Experiments and Evaluations

Texas Road Network: AT_{Data}^{Mul}

• Geometry-based heuristics and optimizations do not show favorable results when applied to network path query under AT_{Data}^{Mul} cost model.

• The results could be data-dependent and more experiments are needed.
Conclusions

(1)

• Geographical information over air is an attractive solution for emerging location dependent services, in terms of scalability, mobility management at the server side and power consumption at the client side.

• We developed the cost models for computing the data access time for processing spatial queries over broadcast geographical data, including DBW, AT_{Data}^{Mul} and AT_{Data}^{Sep}. The derived simple quadratic form of AT_{Data}^{Sep} for processing a single complex query is not only easy to use but also theoretically meaningful, which is the base for us to propose using DBW to approximate AT_{Data}^{Sep}.

• We provided three optimization methods for reducing data access time under DBW, AT_{Data}^{Mul} and AT_{Data}^{Sep}, respectively. They can be applied to spatial range queries, network path queries or any other types of complex queries.
Conclusions (2)

– We first proposed to use an efficient graph MinLA algorithm to optimize DBW after proving its applicability to hyper-graph case.

– We proposed to use DBW to approximate $AT_{Data}^{Sep}$ and use the same algorithm to optimize $AT_{Data}^{Sep}$ based on the proven monotonic relationship between them.

– Our most significant contribution related to optimization is the novel method to optimize $AT_{Data}^{Mul}$. By exploring efficient hyper-graph data structures, we are able to compute the positions of the ending node of a hyper-graph efficiently and provide an effective and efficient yet simple optimization method to optimize $AT_{Data}^{Mul}$. 
Conclusions

(3)

• We performed experiments on five synthetic point data sets, 51 zip code point data sets of 50 states and the District of Columbia of US, and the Texas road network graph data set. The results show that the three proposed optimization methods are very effective.
  – For the 51 zip data sets, on average, the data access time based on the optimized ordering is only about 1/3 of that of the 1000 time random orderings average under the DBW cost model, and, improved by about 30% under the $AT_{Data}^{Mul}$ and the $AT_{Data}^{Sep}$ cost models with acceptable computation overheads.
Conclusions

(4)

–The results from the geometric and graph-based heuristics and their optimizations under the DBW and the $A_{\text{Data} \cdot \text{Mul}}$ cost models applied to the Texas road network data set show that geometric heuristics should be applied to optimizations of spatial range queries for point data sets and graph heuristics should be applied to optimizations of network path queries for graph data sets.
Future Work

• Take access time to the index channel into consideration.
• Follow the multi-scale paradigm to further reduce computation time for optimizations
• Investigate on more efficient methods to compute access frequencies of point data
• Explore more ordering heuristics
• Perform more experiments using both synthetic and real data sets with different sizes and distributions
Publications

Publications Derived From the Dissertation Research


Publications

Publications in Mobile Computing & GI


Publications

Other Publications During the Ph.D. Studies

1. Jianting Zhang, Le Gruenwald, An Efficient Optimal Leaf Ordering for Hierarchical Clustering in Microarray Gene Expression Data Analysis, Accepted by DEXA 2nd International Workshop on Biological Data Management (BIDM04)


3. Candler, C., L. Gruenwald, and J. Zhang, Developing an Information Model: A Lesson in Future Proofing, Slice of Life 2002 for Medical Multimedia Developers and Educators, June 2002
Thanks!

Questions?