

Indexing Design for Correlation Queries on Spatial Time Series Databases: A Summary of Results

Pusheng Zhang *

Department of Computer Science and Engineering

University of Minnesota, Twin Cities

200 Union ST SE, 4-192, Minneapolis, MN, USA

Phone: 612-626-7515; Fax: 612-626-0572; Email: pusheng@cs.umn.edu

ABSTRACT: A spatial time series dataset is a collection of time series, each referencing a location in a common spatial framework. Correlation analysis is often used to identify pairs of potentially interacting elements from the cross product of two spatial time series datasets (the two datasets may be the same). However, the computational cost of correlation analysis is very high when the dimension of the time series and the number of locations in the spatial frameworks are large. In this paper, we use a spatial autocorrelation-based search tree structure to propose new processing strategies for correlation based similarity range queries and similarity joins. We provide a preliminary evaluation of proposed strategies using algebraic cost models and experimental studies with Earth science datasets.

¹This work was partially supported by Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under the Cooperative Agreement number DAAD19-01-2-0014. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

1 INTRODUCTION

A spatial time series dataset (Zhang et al. 2003a; Zhang et al. 2003b) is a collection of time series (Box et al. 1994), each referencing a location in a common spatial framework (Worboys 1995). Finding highly correlated time series from spatial time series datasets collected by satellites, sensor nets, retailers, mobile device servers, and medical instruments on a daily basis is important for many application domains such as epidemiology, ecology, climatology, and census statistics. For example, such queries were used to identify the land locations where the climate was often affected by El Nino (Taylor 1998). However, correlation queries are computationally expensive because large spatio-temporal frameworks contain many locations and time points. The design of efficient access methods to facilitate correlation-based query processing (Agrawal et al. 1993; Faloutsos 1996) on spatial time series data, the focus of this work, is crucial to organizations which make decisions based on large spatio-temporal datasets.

Problem Statement

The problem of designing an efficient indexing method for spatial time series data can be defined as follows.

Given:

- a spatial time series dataset
- a set of operations on the dataset, e.g., finding most correlated time series with a query time series, finding all time series with a correlation above a given threshold for a query time series, insert, delete, and bulk load.

Find: A disk-based data structure.

Objective: Efficiency – to minimize computational components, e.g., CPU and I/O costs.

Constraints:

- Correctness: no false admissions for the operations on the dataset if applicable
- Completeness: no false dismissals for the operations on the dataset if applicable
- Size of spatial framework is much larger than the length of time series

Due to the large amount of spatial time series data, a disk-based data structure is our

goal to efficiently facilitate a set of operations, such as finding most correlated time series with a query time series, finding all time series with a correlation above a given threshold for a query time series, insert, delete, and bulk load. The objective is to minimize computational components, e.g., CPU cost, I/O cost, etc. The computational costs may include search cost, update cost, and bulk load cost.

Related Work

Previous work (Agrawal et al. 1993; Chan and Fu 1999; Faloutsos 1996) on indexing time series data has focused on dimensionality reduction followed by the use of low dimensional indexing (Guttman 1984; Rigaux et al. 2001; Samet 1990) in the transformed space. Unfortunately, the efficiencies of these approaches deteriorates substantially when a small dimensions of subspace cannot represent enough information in the time series data. Many spatial time series datasets fall in this category. For example, finding anomalies is more desirable than finding well-known seasonality in the knowledge discovery process of spatial time series datasets. Therefore, data used in anomaly detection is usually data whose seasonality has been removed. After transformations are applied on deseasonalized data, the power spectrum spreads all over most dimensions. Furthermore, in most spatial time series datasets, the number of spatial locations is much greater than the length of time series. This makes it possible to improve the performance of query processing of spatial time series data by exploiting spatial proximity in the design of access methods.

Contributions and Potential Impacts

In this paper, we develop the spatial cone tree, an index structure for spatial time series data. The spatial cone tree groups similar time series together based on spatial proximity. Correlation-based similarity queries are facilitated using spatial cone trees. Our approach is orthogonal to dimensionality reduction solutions. The spatial cone tree structure preserves full length of time series, and therefore it is insensitive to the distribution of the power spectrum after data transformations. Algebraic analyses using cost models and experimental evaluations are carried out to show that the proposed access method saves a large portion of computational cost, ranging from 40% to 97%.

The proposed approaches have been successfully used in the investigation of teleconnections in NASA's Earth science data. Teleconnections[1] are simultaneous variations in climate and related processes over widely separated points on the Earth. For example, every three to seven years, an El Nino event, i.e., the anomalous warming of the eastern tropical region of the Pacific, may last for months, having significant economic and atmospheric consequences worldwide. El Nino has been linked to climate phenomena such as droughts in Australia and heavy rainfall along the Eastern coast of South America[2]. The investigation of such land-sea teleconnections helps climatologists better understand and predict the impacts of El Nino. In my study, the potentially influential ocean regions in the eastern tropical region of the Pacific were efficiently retrieved using the proposed algorithms for the query cities, states, and countries.

The proposed approaches could be applied in many application domains to efficiently process correlation-based queries in spatial time series datasets. Correlation-based similarity queries are frequently used to retrieve interesting relationships among observations in spatial time series data. Efficient tools for retrieving information from spatial time series data are crucial to organizations which make decisions based on large spatial time series datasets, including NASA, the National Imagery and Mapping Agency (NIMA), the National Cancer Institute (NCI), and the United States Department of Transportation (USDOT). These organizations are spread across many application domains including geographical information systems (GIS), Earth science, epidemiology, ecology, and climatology.

An Illustrative Application Domain

NASA Earth observation systems currently generate a large sequence of global snapshots of the Earth, including various atmospheric, land, and ocean measurements such as sea surface temperature (SST), pressure, precipitation, and Net Primary Production (NPP). NPP is the net photosynthetic accumulation of carbon by plants. Keeping track of NPP is important because it includes the food source of humans and all other organisms and thus, sudden changes in the NPP of a region can have a direct impact on the regional ecology. These data are spatial time series data in nature.

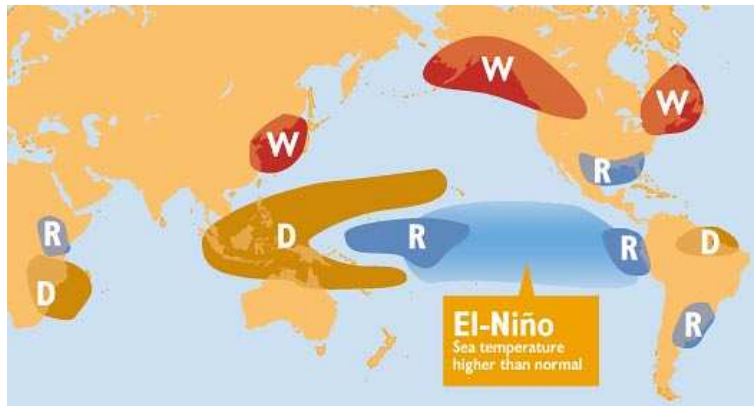


Figure 1: Worldwide climatic impacts of warm El Niño events during the northern hemisphere winter.

The climate of the Earth's land surface is strongly influenced by the behavior of the oceans. Simultaneous variations in climate and related processes over widely separated points on the Earth are called teleconnections. For example, every three to seven years, an El Niño event (NOAA 2004), i.e., the anomalous warming of the eastern tropical region of the Pacific Ocean, may last for months, having significant economic and atmospheric consequences worldwide. El Niño has been linked to climate phenomena such as droughts in Australia and heavy rainfall along the eastern coast of South America, as shown in Figure 1 (reproduced from (Food and Organization)). D indicates drought, R indicates unusually high rainfall (not necessarily unusually intense rainfall) and W indicates abnormally warm periods. To investigate such land-sea teleconnections, time series correlation analysis across the land and ocean is often used to reveal the relationship of measurements of observations.

For example, the identification of teleconnections between Minneapolis and the eastern tropical region of the Pacific Ocean would help Earth scientists to better understand and predict the influence of El Niño in Minneapolis. In our example, the query time series is the monthly NPP data in Minneapolis from 1982 to 1993, denoted as T_q . The minimal correlation threshold is denoted as θ . This is a correlation-based similarity range query to retrieve all highly correlated SST time series in the eastern tropical region of the Pacific Ocean with the NPP time series in Minneapolis. We carry out the range query to retrieve all time series which correlate with T_q over θ in the spatial time series data S , which contain

all the SST time series data in the eastern tropical region of the Pacific Ocean from 1982 to 1993. The query can be represented using *SQL*-like syntax as follows:

```
select SST from S where correlation(SST,  $T_q$ )  $\geq \theta$ 
```

Due to large amount of data available, the performance of linear searching algorithms is not sufficient to satisfy the increasing demands to efficiently process correlation-based similarity queries in large spatial time series datasets. We propose the spatial cone tree, an index structure to facilitate the correlation-based similarity query processing in spatial time series data.

2 PROPOSED ACCESS METHOD

A normalized time series with m time points is located on the surface of an m -dimensional unit sphere (Zhang et al. 2003a). The correlation of two time series is directly related to the angle between the two normalized time series vectors in the multi-dimensional unit sphere. A cone (Zhang et al. 2003a) is a set of normalized time series in a multi-dimensional unit sphere. Every cone c has the following characteristic variables:

- ◇ $axis(c)$, the mean vector of all normalized time series vectors in cone c
- ◇ $span(c)$, the maximal angle between any normalized time series vector in cone c and $axis(c)$ vector

2.1 Spatial Cone Tree Structure

A *spatial cone tree* is a auxiliary search structure for correlation-based queries on spatial time series data. The spatial cone tree uses a tree data structure, and it is formed of nodes. Each node in the spatial cone tree, except for the root has one parent node and several-zero or more-child nodes. The root node has no parent. A node that does not have any child node is called a leaf node and a non-leaf nodes is called an internal node.

A leaf node contains a cone and a data pointer p_d to a disk page containing data entries, and is of the form $\langle (cone.span, cone.axis), p_d \rangle$. The cone contains multiple-one or more-normalized time series, which are contained in the disk page referred by the pointer p_d . The $cone.span$ and $cone.axis$ are made up of the characteristic parameters for the cone.

The data pointer is a block address. An internal node contains a cone and a pointer p_i to an index page containing the pointers to children nodes, and is of the form $\langle (cone.span, cone.axis), p_i \rangle$. The *cone.span* and *cone.axis* are the characteristic parameters for the cone, which contains all normalized time series in the subtree rooted at this internal node. In a balanced tree, all leaf nodes are on the same level. The balancing is desirable to facilitate searching in the spatial cone tree, however, the costs in updates and insertions to maintain balancing are high. Therefore, we don't preserve balancing property in the spatial cone tree in this paper. We will investigate balanced spatial cone trees in future work.

Multiple nodes are organized in a disk page, and the number of nodes per disk page is defined as the blocking factor for a spatial cone tree. Notice that the blocking factor, the number of nodes per disk page, depends on the sizes of cone span, cone axis, and the pointer. The sizes for cone span and the pointer are fixed for different spatial time series data. However, the cone axis is the length of time series, which may be different in different spatial time series data. Therefore when the time series length is too long, the blocking factor drops down. Here we discuss a few options to potentially improve blocking factor. First, dimensionality reduction techniques (Agrawal et al. 1993; Chan and Fu 1999), e.g., discrete Fourier transformation and discrete wavelet transformation, can be applied to time series data before indexing, and indexing is carried out on the transformed space with reduced dimensions. Second, subdividing a long time series into smaller fixed length time series chunks (Moon et al. 2001). We will investigate these options for indexing using spatial cone tree in future work.

Let M denote the maximum number of data entries that fit in one disk page. and *max_span* denote the maximum span threshold for the leaf node. The spatial cone tree satisfies the following properties:

- ◇ The root has at least two children unless it is a leaf;
- ◇ The number of data entries in the cone for a leaf node is no more than M ;
- ◇ The span in a the cone for a leaf node is no more than *max_span*;

Figure 2 (a) shows a spatial cone tree with $M = 4$ and *max_span* = 300. Figure 2 (b) illustrates the normalized time series vectors in a multi-dimensional sphere. Since the

dimension of time series here is set as 2 for simplicity, the sphere in Figure 2 (b) is reduced to a circle.

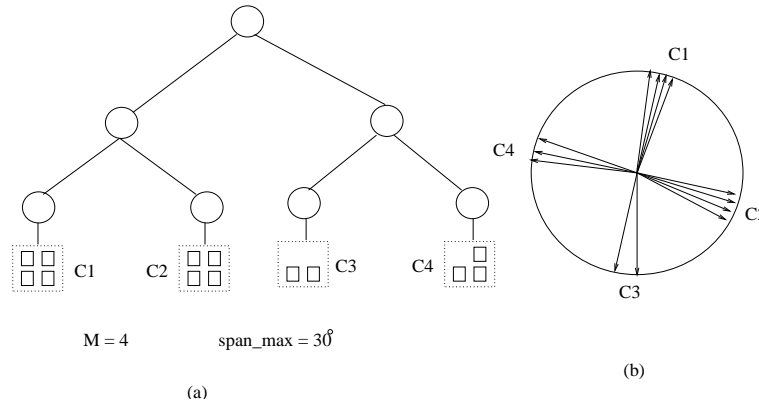


Figure 2: (a)Illustration of a Spatial Cone Tree (b)Normalized Time Series Vectors in a Circle

The spatial cone tree structure allows cone overlapping in multi-dimensional unit spheres. Thus it cannot guarantee that only one search path is required for an exact match query. However, the overlapping-cones technique does not hurt the performance of correlation-based similarity query processing. We will show the proposed access method saves a large portion of computational cost in Section 4.

2.2 Operations

The spatial cone tree can efficiently support point search, query processing operations, and maintenance operations as follow:

- ◇ point search : identify the leafs which may contain the given time series;
- ◇ range query : find all time series with a correlation above a given threshold for a query time series;
- ◇ nearest-neighbor query: find most correlated time series with a query time series;
- ◇ join query: find all pairs of time series with a correlation above a given threshold between two spatial cone trees;
- ◇ insert : add a new time series into a spatial cone tree;
- ◇ delete : remove a new time series entry from a spatial cone tree;

◇ bulk load : construct a spatial cone tree on a large amount of time series.

First, the point search is to identify the leafs which may contain the given time series. Secondly, the query processing operations consist of range query, nearest-neighbor query, and join query. Thirdly, the maintainable operations consist insert, delete, and bulk load operations. In the following sections, we discuss the operations on the spatial cone tree in details.

Point Search

Given a time series t , point search is to find the leaf cone L which contains t . The search begins at the root of the spatial cone tree, then it traverses to the subcone which has the smallest angle with t . The traversal is proceeded until a leaf cone is reached.

2.3 Query Processing Algorithms

Each cone node in a spatial cone tree contains multiple time series, thus the query processing can be carried out on the cone level, instead of on the time series level. First we introduce some lemmas which are used in the query processing algorithms. Due to space consideration, we focus on the query processing algorithms for range query.

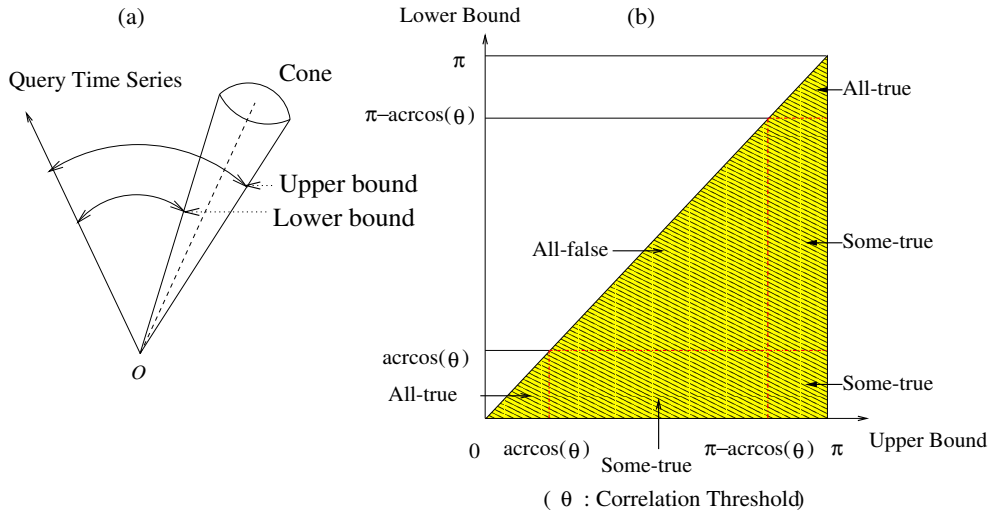


Figure 3: (a)Upper Bound and Lower Bound (b)Filtering Lemmas

Filtering Lemmas:

Given a minimal correlation threshold $\theta(0 < \theta < 1)$, the possible relationships between a cone C and the query time series, T_q , consist of all-true, all-false, or some-true. All-true means that all time series with a correlation over the correlation threshold; all-false means all time series with a correlation less than the correlation threshold; some-true means only part of time series with a correlation over the correlation threshold. The largest angle between the query time series and a cone is denoted as γ_{max} and the smallest angle is denoted as γ_{min} , as illustrated in Figure 3 (a). Let T is any normalized time series in the cone C and $\angle(\vec{T}_q, \vec{T})$ is denoted for the angle between the query time series vector \vec{T}_q and the time series vector \vec{T} in the multi-dimensional sphere.

we have the following properties(please refer to (Zhang et al. 2003a) for proof details):

1. If $\gamma_{max} \in (0, \arccos(\theta))$, then $\angle(\vec{T}_q, \vec{T}) \in (0, \arccos(\theta))$;
2. If $\gamma_{min} \in (180^\circ - \arccos(\theta), 180^\circ)$, then $\angle(\vec{T}_q, \vec{T}) \in (180^\circ - \arccos(\theta), 180^\circ)$;
3. If $\gamma_{min} \in (\arccos(\theta), 180^\circ)$ and $\gamma_{max} \in (\gamma_{min}, 180^\circ - \arccos(\theta))$, then $\angle(\vec{T}_q, \vec{T}) \in (\arccos(\theta), 180^\circ - \arccos(\theta))$.

If either of the first two conditions is satisfied, the cone C is called an all-true cone (all-true lemma). If the third condition is satisfied, the cone C is called an all-false cone (all-false lemma). If none of condition is satisfied, the cone C is called a some-true cone (some-true lemma). These lemma are developed to eliminate cones with all times series satisfying/dissatisfying the correlation threshold in query processing.

Range Query Processing Algorithm: As shown in Algorithm 1, the key idea of range query processing is to process a range query in a filter-and-refine style on the cone level, instead on the individual time series level. The *filtering* step traverses the spatial cone tree, applying the all-true and all-false lemmas on the cones. Therefore, the cones satisfying all-true or all-false conditions are filtered out. The cones satisfying some-true are traversed recursively until all-true or all-false is satisfied or a leaf cone is reached. The *refinement* step manually checks the some-true leaf cones.

For example, a range query is carried out on the spatial cone tree shown in Figure 4. The search begins with the root of the spatial cone tree. The root cone is a some-true cone,

Algorithm 1 Range_Query

Input: 1) SCT : a spatial cone tree
2) T_q : the query time series denote;
3) a user defined correlation threshold θ ;

Output: all time series each from SCT whose correlations with T_q are above θ ;

Method:

```
traverse  $SCT$ ; (1)
for each cone  $c$  on the route do (2)
   $Filter\_Flag = Cone\_level\_Join(T_q, c, \theta)$ ; (3)
  if ( $Filter\_Flag == all\text{-}true$ ) (4)
    output all time series in the cone  $c$  (5)
  else if ( $Filter\_Flag == some\text{-}true$ ) (6)
    if  $c$  is a leaf cone node (7)
      for all pair  $T_q$  and  $s$  from  $c$  do (8)
         $Refine\_Flag = Instance\_level\_Join(T_q, s, \theta)$ ; (9)
        if ( $Refine\_Flag == true$ ) output  $s$ ; (10)
      else for each  $c'$  of  $c$ 's children do (11)
        Range_Query( $c', T_q, \theta$ ) (12)
```

and therefore its children are traversed. Cones 2 and 4 are all-true cones, and cone 3 is a all-false cone. Thus only cone 1 is traversed further. All time series in cones 2 and 4 and time series 12 and 13 are identified to be highly correlated with the query time series.

The traversal strategies in the filtering step can use any traversal strategy (Brinkhoff et al. 1993; Huang et al. 1997) used in tree-based spatial index structures (Rigaux et al. 2001; Samet 1990; Shekhar and Chawla 2003). In this paper, we use a depth-first traversal strategy in the experimental evaluations.

The range query processing algorithm is correct and complete, i.e., there are no false admissions and false dismissals in the query processing. Please refer to (Zhang et al. 2003b) for more details on the proofs.

Join Query Processing Algorithm: Similar to the range query processing algorithm, the join processing algorithm on spatial cone trees is also processing joins in a filter-and-refine style on the cone level. The key difference of the join processing algorithms with the range

query processing algorithm is to filter out the all-true and all-false cone pairs using the filtering lemmas. The join strategies in tree index structures (Rigaux et al. 2001; Samet 1990; Shekhar and Chawla 2003) are applicable in spatial cone trees. A nested-loop join algorithm (Zhang et al. 2003b) was proposed to facilitate the join queries in spatial time series data, please refer to it for more details.

Maintenance Operations

Insert The insertion operation is to insert a new time series into a spatial cone tree. Inserting new time series into a spatial cone tree is similar to B-tree in that new entries are added to the leaf cones, leaf cones that overflow are split, and splits propagate up the tree. As shown in Algorithm 2, a new time series t is inserted into a spatial cone tree. First, we traverse the tree down to find the leaf cone to be inserted for the new time series. Second, we add the new time series into the leaf cone. Finally we update the leaf cone and propagate changes upward. In practice, we could keep a counter for the number of insert occurrences in each cone. The recalculation for the axis and span of this cone are triggered only if the counter reaches a upper limit, e.g., 5, to avoid tedious updates. The Adjust_Tree step updates the leaf cone and invoke splitting on the leaf as necessary. Ascend from a leaf cone node to the root, propagate changes upward in a similar way as B-tree.

Algorithm 2 Insert(t , root)

```

L = Point Search( $t$ , root)
add  $t$  into leaf cone L
Adjust_Tree(L, root)

```

Delete Due to space consideration, we provide only a high-level description of the delete operation. The deletion operation of a time series entry t from a spatial cone tree is carried out in three steps: (1)find the leaf cone L that contains t , (2)remove t from L , and (3)reorganize the tree if needed.

Bulk Load Both space-partitioning methods (Samet 1990) and data-partitioning methods (Guttman 1984; Berchtold et al. 1996) can be applied to the spatial cone tree construction. For simplicity, a top-down quad-tree-like (Samet 1990) spatial cone tree construction method is

used.

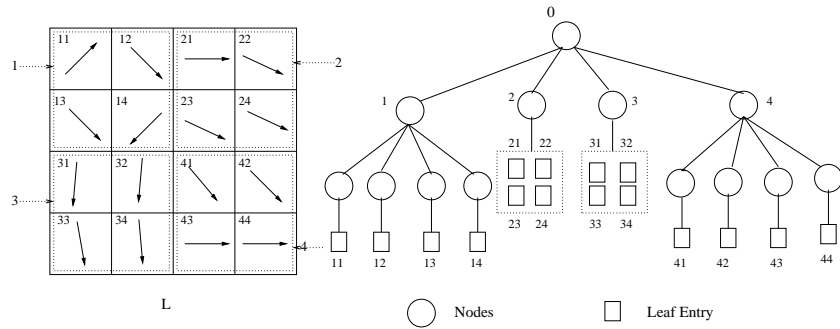


Figure 4: Spatial Cone Tree Construction

We begin with the whole space as the root of the cone tree. When the cone span exceeds max_span or the number of time series exceeds M , a cone is split into four sub-cones based on spatial proximity. The time series are re-distributed into the sub-cones, and the axis and span are calculated for each sub-cone. Each sub-cone is checked and split recursively until its cone span is no more than max_span and its number of time series is no more than M . Figure 4 illustrates the spatial cone tree construction for a small spatial time series dataset. The spatial framework consists of 16 locations, and each location contains a time series of length 2. Each arrow in a location represents the normalized time series vector. The whole space L was divided into four disjoint quadrants 1, 2, 3, and 4, and each quadrant corresponds to a cone node in the spatial cone tree. $M = 4$ and $max_span = 10^\circ$. Cones 1 and 4 are further split because their spans exceed max_span .

3 COST MODELS

In this section, we provide simple algebraic cost models for the operations on spatial cone trees, especially for range query processing and join query processing. The correlation analysis of spatial time series is a CPU intensive task, and the CPU cost is at least as important as the I/O cost for datasets with a long sequence of time series. Furthermore, the number of correlation computations could also be used to measure the computational cost of correlation analyses in different system configurations. Therefore, the number of

correlation computations is used as the unit of cost in the cost models. We will investigate a cost model that includes the I/O cost of query processing in spatial time series data in future work.

As we discussed in Section 2, the query processing algorithms are of filter-and-refine style. Therefore the cost of query processing consists of the costs in the filtering step and refinement step. There are three possible relationships between a cone and the query time series T_q : all-true, all-false, or some-true. Since some-true is the only case to be checked in the refinement step, here we define the leaf cone selectivity ratio j to capture the portion of leaf cones which can not filtered out in the traversal through the spatial cone tree, e.g., 30 out of 100 leaf cones are some-true cones in a range query processing, $j = 0.3$.

The objective of the correlation-based similarity range queries be to retrieve all highly correlated time series with a query time series, T_q , from a spatial time series data S . Let L denote the leaf cones in a spatial cone tree and $|L|$ denote the number of the leaf cones in L . Due to the definition of the leaf cone selectivity ratio, the number of the leaf cones in the refinement step is $|L| \times j$. Let c_r denote the average cost of manual checking between a leaf cone and T_q in the refinement step. Therefore the cost of the refinement step is $|L| \times j \times c_r$.

Let p denote the number of children in a spatial cone tree. Assume a spatial cone tree is a full complete tree, the numbers of leaf nodes and non-leaf nodes in the spatial cone tree are p^n and $1 + p + p^2 + \dots + p^{n-1}$ respectively. Since the number of the leaf cones in the refinement step is $|L| \times j$, the number of non-leaf nodes visited in the traversal can be estimated by $|L| \times j \times \frac{1+p+p^2+\dots+p^{n-1}}{p^n} = |L| \times j \times \frac{1}{p-1}$. Let c_f denote the cost of the checking between a non-leaf cone and T_q in the filtering step, and $c_f = 1$ for our approach since only one correlation calculation is carried out between T_q and the axis vector. Therefore the cost of the filtering step is $|L| \times (1 + \frac{1}{p-1}) \times j \times c_f = |L| \times \frac{p}{p-1} \times j$.

The objective of the correlation based similarity join is to retrieve all highly correlated time series pairs between the two datasets. Let T_1 and T_2 denote the search trees for the dataset S_1 and S_2 respectively. Let $|T_1|$ and $|T_2|$ denote the number of nodes in T_1 and T_2 respectively. Let L_1 and L_2 be the leaf cone sets for T_1 and T_2 respectively, and $|L_1|$ and $|L_2|$ be the number of leaf cones in L_1 and L_2 respectively. Similarly to range query, the cost of the refinement step is $|L_1| \times |L_2| \times j \times c_{jr}$, where c_{jr} denote the average cost of

manual checking between two leaf cones in the refinement step. Assume both the numbers of children in the two spatial cone trees are the same to p for simplicity. Let c_{jf} denote the cost of the checking between two leaf cones in the filtering step, and $c_{jf} = 1$ for our approach since only one correlation calculation is carried out between the two axis vectors. The cost of the filtering step is $|L_1| \times |L_2| \times \frac{p}{p-1} \times j$,

In summary, for a fixed number of the children in a spatial cone tree, a lower leaf cone selectivity ratio implies better computational efficiency for both range query processing and join processing.

4 PERFORMANCE EVALUATION

We evaluated the performance of the proposed query processing strategies using spatial cone trees with a dataset from NASA Earth science data (Potter et al. 1999). In this experiment, correlation-based similarity queries were carried out between the Sea Surface Temperature (SST) in the eastern tropical region of the Pacific Ocean (80W - 180W, 15N - 15S) and Net Primary Production (NPP) in the United States. The NPP time series from 2901 land cells of the United States and the SST time series from 11556 ocean cells of the eastern tropical region of the Pacific Ocean were obtained under a 0.5 degree by 0.5 degree resolution. The records of NPP and SST were monthly data from 1982 to 1993.

In the following sections, we provide experiments on real data to evaluate computational performance on processing algorithms using spatial cone trees for range queries and join queries.

4.1 Range Query Processing

This section describes a group of queries carried out to show the savings of the range query processing strategy using spatial cone trees. We chose a depth-first top-down traversal strategies for the range query processing algorithm. The SST data for the eastern tropical region of the Pacific ocean was chosen as the inner loop to construct a spatial cone tree. The query time series were from the NPP data in the United States. We carried out the range queries in the spatial cone tree for SST. All time series in SST, which correlates with the query NPP time series over the given minimal correlation threshold θ , are retrieved.

The brute force strategy scans all the time series in SST linearly. The cost of the brute force range queries is equal to $|S|$, where $|S|$ denotes the number of time series in the SST data. Here we define the saving ratio as the percentage of cost savings of a range query processing compared to the cost of a range query using the brute force strategy measured in the unit of number of correlation computations. And we define the average saving ratio for multiple range queries as the mean saving ratio for these range queries. We define the selectivity ratio for a range query as the fraction of query results of time series among all the time series in the dataset. And we define the average selectivity ratio for multiple range queries as the mean selectivity ratio for these range queries.

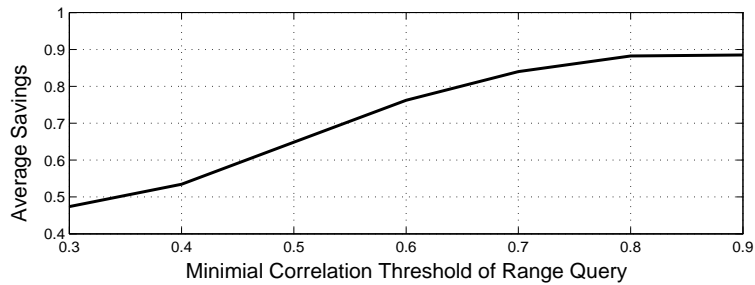


Figure 5: Savings of Range Queries

We randomly chose 10 NPP time series from the United States and carried out the correlation-based similarity range queries using the two different strategies respectively with the SST data from the eastern tropical region of the Pacific Ocean. The geographical locations of the 10 query time series were widely spread in the United States. As shown in Figure 5, the solid line represents the average saving ratios for the top-down traversal strategy, and the saving ratios range from 0.48 to 0.89.

As the minimal correlation thresholds increases, more and more non-leaf nodes(cones) in the spatial cone tree are identified as all-true or all-false cones in the query processing. Therefore the leaf cone selectivity ratio drops down, and the computational efficiency is improved.

4.2 Join Processing

This section describes a group of experiments carried out to show the net savings of the proposed query processing algorithm using spatial cone trees for the correlation-based sim-

ilarity joins. The NPP time series dataset for the United State was chosen as the outer loop. The SST time series data for the eastern tropical region of the Pacific Ocean was chosen as the inner loop. A spatial cone tree was constructed for the SST data. (Assume that we have built the spatial cone trees before we carried out the similarity join operations.)

The cost of a brute force strategy is $|N| \times |S|$, where $|N|$ and $|S|$ are the number of the time series in NPP and SST respectively. Here we define the saving ratio as the percentage of cost savings of a join processing compared to the cost of a join using a brute force (nested-loop without index) strategy measured in the unit of number of correlation computations. We define the selectivity ratio for a join as the fraction of join results of time series among the cross product of the two spatial time series datasets.

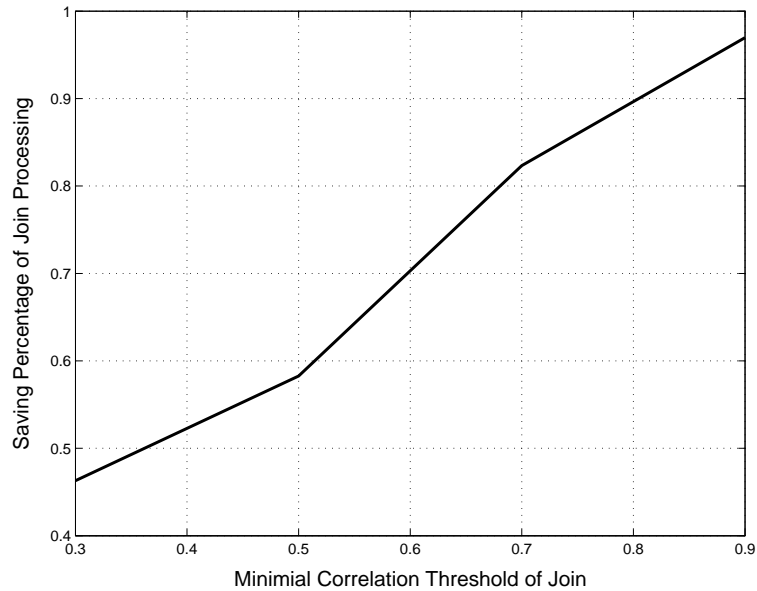


Figure 6: Savings of Join Queries

As shown in Figure 6, the saving ratios of the join processing using spatial cone trees increases ranging from 46% to 97%, as the minimal correlation threshold of the joins increases. When the minimal correlation threshold increases, less leaf nodes(cones) are possibly traversed in the join processing using spatial cone tree. The leaf cone selectivity ratio drops down and therefore the computational efficiency is improved.

In summary, the experimental results show that the query processing using spatial cone tree saves a large fraction of the computational cost.

5 CONCLUSION AND FUTURE DIRECTION

In this paper, we developed the spatial cone tree, an index structure for correlation-based similarity queries. Correlation-based query processing was efficiently facilitated using the spatial cone tree. Analytical and experimental evaluations were carried out to show the efficiency of proposed query processing algorithms. In future work, we will study more design issues, e.g., blocking factor and balancing. Other similarity measures, such as time lagged correlation, will be incorporated into spatial cone trees. We would like to investigate the generalization of spatial cone trees to non-spatial index structures using spherical k-means (Dhillon et al. 2001) to construct cone trees. Moreover, we will investigate further nearest-neighbor search and incremental insert/delete operations on spatial cone trees.

6 ACKNOWLEDGMENTS

I am particularly grateful to my adviser, Prof. Shashi Shekhar for his review and helpful comments. This work was partially supported by NASA grant No. NCC 2 1231 and by Army High Performance Computing Research Center contract number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

REFERENCES

- Agrawal, R., C. Faloutsos, and A. Swami 1993. Efficient Similarity Search In Sequence Databases. In *Proc. of the 4th Int'l Conference of Foundations of Data Organization and Algorithms*.
- Berchtold, S., D. Keim, and H. Kriegel 1996. The X-tree: An Index Structure for High-Dimensional Data. In *The Proc. of 22nd VLDB Conference*.
- Box, G., G. Jenkins, and G. Reinsel 1994. *Time Series Analysis: Forecasting and Control*. Prentice Hall.
- Brinkhoff, T., H. Kriegel, and B. Seeger 1993. Efficient Processing of Spatial Join Using R-trees. In *ACM SIGMOD*.
- Chan, K. and A. W. Fu 1999. Efficient Time Series Matching by Wavelets. In *Proc. of the 15th ICDE*.

- Dhillon, I., J. Fan, and Y. Guan 2001. Efficient Clustering of Very Large Document Collections. In R. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu (Eds.), *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers.
- Faloutsos, C. 1996. *Searching Multimedia Databases By Content*. Kluwer Academic Publishers.
- Food and A. Organization. Farmers brace for extreme weather conditions as El Nino effect hits Latin America and Australia . <http://www.fao.org/NEWS/1997/970904-e.htm>.
- Guttman, A. 1984. R-Trees: A Dynamic Index Structure For Spatial Searching. In *ACM SIGMOD*.
- Huang, Y., N. Jing, and E. Rundensteiner 1997. Spatial Joins using R-trees: Breadth-First Traversal with Global Optimizations. In *The Proc. of 23rd VLDB Conference*.
- Moon, Y., K. Whang, and W. Loh 2001. Efficient Time-Series Subsequence Matching using Duality in Constructing Windows. *Information Systems* 26(4).
- NOAA 2004. El Nino Page. <http://www.elnino.noaa.gov/>.
- Potter, C., S. Klooster, and V. Brooks 1999. Inter-annual Variability in Terrestrial Net Primary Production: Exploration of Trends and Controls on Regional to Global Scales. *Ecosystems* 2(1), 36–48.
- Rigaux, P., M. Scholl, and A. Voisard 2001. *Spatial Databases: With Application to GIS*. Morgan Kaufmann Publishers.
- Samet, H. 1990. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Company, Inc.
- Shekhar, S. and S. Chawla 2003. *Spatial Databases: A Tour*. Prentice Hall, ISBN:0130174807.
- Taylor, G. H. 1998. Impacts of the El Nio/Southern Oscillation on the Pacific Northwest. http://www.ocs.orst.edu/reports/enso_pnw.html.
- Worboys, M. F. 1995. *GIS - A Computing Perspective*. Taylor and Francis.
- Zhang, P., Y. Huang, S. Shekhar, and V. Kumar 2003a. Correlation Analysis of Spatial Time Series Datasets: A Filter-and-Refine Approach. In *the Proc. of the 7th Pacific-Asia Conference on Data Mining and Knowledge Discovery*.
- Zhang, P., Y. Huang, S. Shekhar, and V. Kumar 2003b. Exploiting Spatial Autocorrelation to Efficiently Process Correlation-Based Similarity Queries. In *the Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases*.