Efficient Parallel Spatial Skyline Evaluation Using MapReduce

Wenlu Wang†, Ji Zhang†, Min-Te Sun‡, Wei-Shinn Ku†

†Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA
‡Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan, ROC

{wenluwang, jizhang, weishinn}@auburn.edu, msun@csie.ncu.edu.tw

ABSTRACT

This research presents an advanced MapReduce-based parallel solution to efficiently address spatial skyline queries on large datasets. In particular, given a set of data points and a set of query points, we first generate the convex hull of the query points in the first MapReduce phase. Then, we propose a novel concept called independent regions, for parallelizing the process of spatial skyline evaluation. Spatial skyline candidates in an independent region do not depend on any data point in other independent regions. Thus, we calculate the independent regions based on the input data points and the convex hull of the query points in the second phase. With the independent regions, spatial skylines are evaluated in parallel in the third phase, in which data points are partitioned by their associated independent regions in the map functions, and spatial skyline candidates are calculated by reduce functions. The results of the spatial skyline queries are the union of outputs from the reduce functions. Due to high cost of the spatial dominance test, which requires comparing the distance from data points to all convex points, we propose a concept of pruning regions in independent regions. All data points in pruning regions can be discarded without the dominance test. Our experimental results show the efficiency and effectiveness of the proposed parallel spatial skyline solution utilizing MapReduce on large-scale real-world and synthetic datasets.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Application—spatial databases

Keywords

Spatial Skyline Query, MapReduce, Parallel Computation

1. INTRODUCTION

Since the skyline operator was introduced into database research [4], a number of efficient algorithms have been proposed for the skyline evaluation. Bitmap [25], Index [25], NN (Nearest Neighbor) [16] and BBS (Branch-and-Bound Skyline) [19] rely on indices constructed before query processing; while BNL (Block Nested Loop) [4], D&C (Divide and Conquer) [4], SFS (Sort Filter Skyline) [7], and OSPS (Object-based Space Partitioning Skyline) [32] use non-index techniques. Moreover, several studies primarily focus on the skyline query in a variety of problem settings (data residing in a data stream [22] or on mobile devices [14]).

As a novel type of skyline queries, Spatial Skyline Query (SSQ) was proposed to consider the preference of both static and dynamic object attributes in multi-criteria decision-making applications [23]. Unlike skyline queries that only take static object attributes (e.g., rating and price of restaurants) into account, the distance between objects is also calculated as dynamic attributes in the spatial skyline queries. In particular, given a set of data points \( P \) and a set of query points \( Q \) in a \( d \)-dimensional space, spatial skyline queries return a subset of \( P \), in which data points are not spatially dominated by other data points in \( P \). The spatial dominance is defined by using the distance from data points to all query points.

Spatial skyline query is applicable to many applications. Take crisis management applications as an example, we assume that a number of waterborne infectious disease cases were confirmed at different locations, people who live at spatial skyline places with respect to those locations should be alerted and examined first, because there might be higher possibility that these people may have been exposed to contagious water. Travel planning applications are another type of example. People may prefer the spatial skyline hotels with respect to fixed locations of beaches and museums for their vacation. In this case, people would not like to choose a hotel, which is farther from all interesting attractions than other hotels. One more example of spatial skyline query is that people may plan to have dinner with their friends at weekends. They may consider the distance from their homes to the restaurant for the restaurant selection. The restaurants far from all of their homes would not be in the candidate list, because they may want to save time on the road. Thus, giving a list of spatial skyline restaurants is the first step of the restaurant selection.

Two index-based algorithms were proposed to efficiently address the spatial skyline queries [23]. Branch and Bound Spatial Skyline \( B^2S^2 \) algorithm searches spatial skyline candidates by visiting an R-tree from top to bottom. Once a spatial skyline is found, \( B^2S^2 \) expands the R-tree to access the node which has minimummindist value, and compares it with all spatial skyline candidates found so far in spatial dominance test. The other method, Voronoi-based Spatial Skyline \( VS^2 \) algorithm relies on a Voronoi diagram created over input data points. \( VS^2 \) starts with the closest data points to query points, searches in the space by visiting the neighbors of visited data points over the Voronoi diagram. Due to high cost of the spatial dominance test, \( VS^2 \) was improved by reducing the number of spatial dominance tests in [24]. In the method, seed skyline

© 2017, Copyright is with the authors. Published in Proc. 20th International Conference on Extending Database Technology (EDBT), March 21-24, 2017 - Venice, Italy. ISBN 978-3-8318-073-8, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0
points (a subset of spatial skyline points) can be identified with spatial dominance test.

However, the following problems motivate us to propose a novel parallel solution for spatial skyline evaluation. Firstly, as data grows rapidly, addressing skyline queries on large-scale datasets in a single-node environment becomes impractical. There are increasing number of approaches proposed for processing skyline queries in distributed and/or parallel environments [13]. But none of parallel spatial skyline solutions were observed in literature. Secondly, the distance between moving objects may keep changing. If indices are created at a preprocessing stage, the cost of index maintenance would be unacceptably high. Thirdly, MapReduce framework has been incorporated into parallel solutions for skyline computation [17] [20] [31] and other database applications [30] [26] [18].

Therefore, we propose a novel three-phase MapReduce-based solution, which is able to efficiently address spatial skyline queries on large-scale datasets in this paper. In particular, given a set of data points and a set of query points, we calculate the convex hull of the query points in the first phase. Initially, the query points are evenly partitioned. Each map function accepts a subset of query points, and outputs a local convex hull. Then, the reduce function produces the global convex hull by merging the intermediate results from the map functions. A filtering method can be applied to filter out unqualified data points before the convex hull computation. For example, CG_Hadoop uses skyline algorithms as a filtering method in convex hull evaluation [11]. Moreover, to parallelize the spatial skyline computation, we propose a novel concept, independent regions, in which each of which spatial skylines do not depend on any data point outside the independent region. If data points do not fall in any independent region, they can be discarded because they must be spatially dominated by other data points. Thus, our solution produces the independent regions at the second phase. Each map function receives a subset of data points and the convex hull of query points, and generates locally optimized independent regions. Then, the reduce functions output globally optimized independent regions.

With the independent regions in the third phase, map functions associate data points with their independent regions. By using the unique identifiers of independent regions as keys, all data points in an independent region are sent to a reducer after the shuffle phase, and reduce functions find the spatial skylines in independent regions in parallel. Due to high cost of the spatial dominance test, which requires comparing the distance from data points to all convex points, we propose a novel concept, pruning regions, in independent regions. The pruning regions are the areas in which all data points are dominated by other data points. Thus, since a pruning region is defined by a data point, a convex point, and its adjacent convex points, if a data point is in a pruning region, the data point can be discarded without accessing all convex points and calculating the distance from the data point to them. In addition, a data point may fall in more than one independent regions, and there may exist duplicates in spatial skyline candidates. We employ an elimination method in our solution to remove the duplicates with subtle overhead.

In short, the contributions of this study are summarized below:

1. We propose a parallel scheme to efficiently evaluate spatial skyline queries on large datasets using MapReduce.

2. We introduce a concept of independent regions in our solution. The spatial skyline candidates in an independent region do not depend on any data points in other independent regions. With the feature of the independence, spatial skyline queries can be addressed in parallel.

3. We present a concept of pruning regions in independent regions, in order to minimize the cost of the dominance test by avoiding the computation of distance from data points to all convex points.

4. We evaluate the performance of the proposed solution through extensive experiments with large-scale real-world and synthetic datasets.

The rest of this paper is organized as follows. Section 2 surveys related works. The spatial skyline queries and relevant techniques utilized in our solutions are formally defined in Section 3. In Section 4, our advanced solution is presented. The experimental validation of our design is presented in Section 5. We conclude the paper in Section 6.

2. RELATED WORK

In this section, we review previous works related to spatial skyline queries and parallel solutions for general skyline queries.

2.1 Spatial Skyline Queries

As a special case of dynamic skyline queries, Spatial Skyline Queries (SSQ) can be addressed by Block Nested Loop (BNL) [4] and Branch-and-Bound Skyline algorithms (BBS) [19]. In a dynamic skyline query, each object is mapped to another search space by using pre-defined functions. All the objects that are not dominated by other objects in the search space after the mapping are returned from the dynamic skyline queries. BNL algorithm can address the dynamic skyline queries, because it compares every pair of objects in the input dataset, and eliminates the ones that are dominated by any other objects. BNL does not need indices and is efficient over small datasets. But it suffers from I/O access when the input datasets become large. If the size of skyline candidates exceeds the size of available memory space, all the candidates have to be written to a temporary data stream, and read back when they are needed in the next iteration of object comparison. BBS relies on an R-tree to evaluate the general skyline queries; it calculates the mindist of intermediate entries in the R-tree, and searches the space by expanding the entry with the smallest mindist. However, BBS does not consider the relation between the input query points and data points.

Motivated by the inefficiency of BNL and BBS, a Branch-and-Bound Spatial Skyline (BS2S) algorithm and a Voronoi-based Spatial Skyline (VS2) algorithm were proposed for spatial skyline evaluation [23]. In addition to considering the properties of the convex hull generated by input query points, BS2S expands the search space by visiting an R-tree from top to bottom. Once the first spatial skyline is found, BS2S expands the R-tree with the node which has the minimum mindist value, and checks the dominance between the visited node and all spatial skyline candidates found so far. The process continues until all intermediate nodes potentially containing spatial skylines have been visited. On the other hand, VS2 builds a Voronoi diagram over input data points. The input data points are organized by their Hilbert values in pages in order to preserve their locality. After completion of convex hull calculation, VS2 starts with the closest data points to the query points, and searches the space by visiting the neighbors of visited data points over the Voronoi diagram. For every visited data point, VS2 compares it with all spatial skylines found so far for spatial dominance test. The process continues until all Voronoi cells (or data points) that potentially contain spatial skylines have been visited. Inspired by high cost of the spatial dominance test, VS2 was improved by reducing the number of spatial dominance tests [24]. In addition to applying sorting techniques, the method is able to identify seed
skyline points (a subset of spatial skyline points) without dominance test. Given a set of query points \( Q \) and a set of data points \( P \), let \( V(p_i) \) be the Voronoi cell of data point \( p_i \in P \), the seed skyline points are the points \( p_i \) that \( V(p_i) \) intersect with the boundary of the convex hull of \( Q \) or is inside the convex hull. However, none of the aforementioned methods can address the spatial skyline query in parallel. \( B^n \times S^2 \) requires a pre-structured R-tree and \( V \times S^2 \) needs to build a Voronoi diagram over input data points. Extending their methods to a distributed and/or parallel environment is non-trivial.

2.2 Parallel Skyline Solution

Due to high cost of skyline evaluation, a number of advanced solutions have been proposed to evaluate the general skyline queries in a distributed and/or parallel environment. Balke et al. developed a parallel skyline solution over distributed environments [3]. Their method first vertically partitions input datasets in such a manner that each partition keeps object attributes in one dimension. Then, the skyline objects are calculated in parallel, and reported to a central point for a final dominance check. Wu et al. designed a parallel skyline method that leverages content-based data partitioning [28]. Their method can avoid unnecessary data access and can progressively produce skylines by using recursive region partitioning and dynamic region encoding mechanisms. Moreover, the incremental scalability is also provided in such a manner that workload can be automatically balanced by distributing objects to new nodes. In addition to random data partitioning methods that can generate similar data distribution in each partition [8] and grid-based data partitioning methods that consider object proximity [2] [21], Vlachou et al. proposed an angle-based data partitioning method that partitions objects by their angular coordinates [27]. The average pruning power of objects within a partition can be increased and the number of skyline objects in local skyline calculation can be minimized by applying the angle-based partitioning method. Köhler et al. designed a hyperplane-based data partitioning method in order to minimize the local skylines in a partition and achieve efficient local skyline merging [15]. Moreover, a variety of MapReduce-based parallel solutions have been proposed for skyline queries and other database applications. Han et al. proposed an advanced skyline algorithm that utilizes Sorted Positional Index Lists (SSPL) to reduce I/O cost [12]. Zhang et al. implemented BNL, SFS, and Bitmap algorithms using MapReduce framework [29]. Chen et al. applied an angular data partition in their MapReduce-based solution for skyline query evaluation [6]. Eldawy et al. developed CG_Hadoop, a suite of MapReduce algorithms, to solve fundamental computational geometry problems, which include convex hull computation [11]. Mullesgaard et al. investigated the general skyline queries by using the MapReduce framework. Their method uses bit strings to represent the dominance relation of attributes, and generates independent partition groups for calculating local skyline objects in parallel [17]. Zhang et al. proposed an efficient parallel skyline solution using MapReduce, in which a Partial-preserved Grid-based Partition Skyline (PGPS) algorithm was developed to significantly improve the merging skyline computation on large datasets [31]. More importantly, PGPS can be easily incorporated in the shuffle phase of the MapReduce framework with minor overhead. However, our proposed solution targets on spatial skyline queries, which are different from the general skyline queries. None of the partition schemes or computation algorithms above could address the spatial skyline problem. Therefore, we propose a novel partition method and a parallel algorithm which includes independent regions to parallelize the spatial skyline computation and pruning regions to reduce the cost of spatial dominance test.

3. PRELIMINARIES

3.1 Problem Statement

Given a dataset \( P \) in a \( d \)-dimensional space \( \mathbb{R}^d \), an object \( p \in P \) can be represented as \( p = \{x_1, x_2, \ldots, x_d\} \) where \( p \cdot e_i \) is the value of the object on the \( i \)-th dimension. \( D(\cdot, \cdot) \) denotes a distance metric that obeys the triangle inequality in \( \mathbb{R}^d \). The spatial domination relationship and the spatial skyline operator are defined as follows [23]. All notations used in this paper are summarized in Table 1.

**Definition (Spatial Domination)** Given a set of query points \( Q \), and two data points \( p \) and \( p' \) in \( \mathbb{R}^d \), \( p \) spatially dominates \( p' \) with respect to \( Q \), denoted by \( p \preceq_Q p' \), if \( \forall q \in Q, D(p, q) \leq D(p', q) \) and \( \exists q' \in Q, D(p, q') < D(p', q') \).

**Definition (Spatial Skyline)** Spatial skylines of a set of data points \( P \) with respect to a set of query points \( Q \) in \( \mathbb{R}^d \), denoted by \( SSKY(P, Q) \), are a set of data points in \( P \), which are not spatially dominated by any other data point in \( P \) with respect to \( Q \).

\[
SSKY(P, Q) = \{ p \in P \mid \exists p' \in P, p \neq p', p \preceq_Q p' \} \tag{1}
\]

**Property 1.** If any data point \( p \in P \) is a spatial skyline point with respect to a subset of query points \( Q' \subseteq Q \), then \( p \) is also a spatial skyline point with respect to \( Q \) [23].

**Property 2.** The set of spatial skyline points of data points \( P \) does not depend on any non-convex query points \( q \in Q, q \notin CH(Q) \), where \( CH(Q) \) indicates the convex hull of \( Q \) [23]. In other words,

\[
SSKY(P, Q) = SSKY(P, CH(Q)) \tag{2}
\]

**Definition (Dominator Region)** Given a data point \( p \in P \), a set of query points \( Q \), and hyper-spheres that center at \( q \) with radius \( D(p, q) \), \( q \in Q \), any data point inside the intersection of the hyper-spheres spatially dominates \( p \) with respect to \( Q \). The intersection area that potentially contains data points spatially dominating \( p \) with respect to \( Q \) is referred to as the dominator region of \( p \), denoted by \( DR(p, Q) \).

Dominator Region enables our solution to efficiently eliminate data points by reducing the search space of data points. For example, Figure 1 displays dominator region of a data point \( p \) and a set of query points \( Q \). \( Q \) has three query points \( q_1, q_2, \) and \( q_3 \),

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P, Q )</td>
<td>a set of data points and a set of query points</td>
</tr>
<tr>
<td>( p, q )</td>
<td>a data point and a query point</td>
</tr>
<tr>
<td>( p \cdot x_i )</td>
<td>the value of data point ( p ) in the ( i )-th dimension</td>
</tr>
<tr>
<td>( \mathbb{R}^d )</td>
<td>a ( d )-dimensional space</td>
</tr>
<tr>
<td>( h )</td>
<td>a hyper-plane</td>
</tr>
<tr>
<td>( S )</td>
<td>a half-space</td>
</tr>
<tr>
<td>( F )</td>
<td>a facet of a convex hull</td>
</tr>
<tr>
<td>( \Delta^C )</td>
<td>a set of adjacent convex points of ( p )</td>
</tr>
<tr>
<td>( p \preceq_Q p' )</td>
<td>( p ) spatially dominates ( p' ) with respect to ( Q )</td>
</tr>
<tr>
<td>( SSKY(P, Q) )</td>
<td>spatial skylines of ( P ) with respect to ( Q )</td>
</tr>
<tr>
<td>( CH(Q) )</td>
<td>the convex hull of ( Q )</td>
</tr>
<tr>
<td>( DR(p, Q) )</td>
<td>the dominator region generated by ( p ) and ( Q )</td>
</tr>
<tr>
<td>( IR(p, q) )</td>
<td>the independent region generated by ( p ) and ( q )</td>
</tr>
<tr>
<td>( IRP )</td>
<td>Independent Regions Pivot, the independent regions are generated by ( IRP )</td>
</tr>
<tr>
<td>( lsky )</td>
<td>a set of local spatial skyline candidates</td>
</tr>
<tr>
<td>( chsky )</td>
<td>a set of spatial skyline candidates in a convex hull</td>
</tr>
</tbody>
</table>
which represent a convex hull in a 2-dimensional space. Three circles centered at \( q_i \in Q \) with radius \( D(q_i, p) \) are created in order to highlight the dominance areas of \( p \) with respect to the query points. Any data point \( p' \) in the intersection of the three circles spatially dominates \( p \) with respect to \( Q \).

### 3.2 Convex Hull and Spatial Skyline Queries

Given a set of query points \( Q \) in a \( d \)-dimensional space \( \mathbb{R}^d \), the convex hull of \( Q \), denoted by \( CH(Q) \), is the smallest convex polytope that contains all query points in \( Q \). Theoretically, a convex hull can be represented as either a set of convex points or the intersection of a set of half-spaces. Each half-space contains all the query points in \( Q \). Moreover, a convex hull can also be abstracted by a set of facets and their adjacency relationships. Each facet can be defined by a number of convex points. For example, a facet (line) can be determined by two adjacent convex points in a 2-dimensional space. The facets become planes that can be represented by a convex point and its two adjacent convex points in a 3-dimensional space. Because the facets of \( CH(Q) \) separate the query points in \( Q \) from any point outside the convex hull, connecting a data point \( v \) outside \( CH(Q) \) with any data point in \( CH(Q) \) must intersect with at least one facet of the convex hull. Thus, the facet is referred to as a visible facet from \( v \).

The properties of convex hull provide opportunities to optimize the process of spatial skyline evaluation by reducing the search space of both data points and query points. Given a set of data points \( P \) and the convex hull of a set of query points \( Q \), all data points inside \( CH(Q) \) are spatial skylines of \( P \) with respect to \( Q \) [23]. Given two data points, if they are in the convex hull, the bisector hyper-plane of these two points partitions the space into two half-spaces, and there must exist convex points in either half-space. Thus, neither of the two data points can spatially dominate the other, and both of them are spatial skylines. If one point \( p_1 \) is in the convex hull and the other \( p_2 \) is not, then the bisector line of \( p_1 \) and \( p_2 \) partitions the space into two half-spaces, and there must exist a convex point in the same half-space with \( p_1 \). If the convex point does not exist, the convex hull cannot contain \( p_1 \), which contradicts with our assumption. Thus, \( p_1 \) is not spatially dominated by \( p_2 \). These two cases are summarized in Property 3.

**PROPERTY 3.** Given a set of data points \( P \) and a set of query points \( Q \), if any point \( p \in P \) is inside the convex hull of \( Q \), then \( p \) is a spatial skyline of \( P \) with respect to \( Q \) \((p \in SSKY(P;Q))\).

### 3.3 MapReduce Overview

MapReduce was proposed as a generic programming model for data-intensive applications in distributed environments [10]. The framework provides two simple primitives, map and reduce functions, and allows developers to mainly focus on their functionality. The task scheduling, load balancing, and other issues are encapsulated in the MapReduce framework, which significantly reduces the difficulty of the development of parallel applications. Driven by the MapReduce framework, map functions receive data in key/value pairs from input streams and output intermediate results in another type of key/value pairs. Then, reduce functions retrieve the intermediate results and write final results to an output stream. In the shuffle phase, the intermediate results are automatically grouped and sorted by the MapReduce framework. The two primitives can be represented as: map(\( K_1, V_1 \)) \( \rightarrow \) list(\( K_2, V_2 \)) and reduce(\( K_2 \), list(\( V_2 \))) \( \rightarrow \) list(\( K_3, V_3 \)).

### 4. DESIGN

In this section, we propose our advanced parallel spatial skyline solution using MapReduce. First of all, we briefly present the framework of the solution. Then, our spatial skyline algorithm is illustrated in detail in Section 4.2. The concepts of independent regions and pruning regions are introduced to optimize the process of spatial skyline evaluation. Finally, we discuss three critical implementation issues in our solution.

#### 4.1 Framework Overview

Our solution consists of three MapReduce phases, which receive a set of data points \( P \) and a set of query points \( Q \) as inputs, and output spatial skyline points of \( P \) with respect to \( Q \). As illustrated in Figure 3, we calculate the convex hull of \( Q \) in the first MapReduce phase. \( Q \) is initially partitioned into subsets of equal size, and each map function finds the local convex hull of query points in a subset. Then, a reduce function generates the global convex hull of \( Q \) by merging the local convex hulls. Convex hull algorithm like Graham scan could be employed in each map and reduce function [5]. Due to high complexity of convex hull computation, a filtering method can be used to filter out unqualified points with lower cost. For example, Eldawy et al. observed that the convex points must be at least one of four types of skyline points of \( Q \) (max-max, min-max, max-min, and min-min) in a 2-dimensional space, and applied skyline algorithms as a filtering step in their CG_Hadoop system [11].

An intuitive spatial skyline method requires to examine the spatial dominance between every pair of data points. Sharifzadeh and Shahabi utilized the R-tree and Voronoi diagram as indices in their \( B^2S^2 \) and \( V^2S^2 \) algorithms [23]. Son et al. enhanced \( V^2S^2 \) by reducing the number of dominance tests [24]. However, extending these methods to a parallel solution is non-trivial. Efficiently maintaining indices over data in a distributed and/or parallel environment requires expertise and extensive experience. To address this issue, we propose a novel concept, independent regions, in which each spatial skyline points do not depend on any data points outside the independent region. With the independence, the input data points can be partitioned by their independent regions, and spatial skyline points can be calculated in parallel. Therefore, after the completion of convex hull computation, we calculate the independent regions based on the convex hull and the input data points \( P \) in the second phase. Each map function takes a subset of \( P \) and the convex hull of \( Q \) as inputs, and outputs a locally optimal Independent Region Pivot (See Figure 2, the independent regions are determined by the independent region pivot and convex points). Then a reduce function produces a globally optimal independent region pivot by merging the intermediate results. More details of independent region pivot selection will be discussed in Section 4.3.1. In the third phase, \( P \) is initially partitioned, and each map function finds the independent regions of data points in a split. The output of the map functions can be represented as \( IR.id \, , \, p \) \( > \), where \( IR.id \) denotes the unique identifier of the independent region associated with a data point \( p \). There are three possible cases: (1) data points...
are eliminated if they are outside all independent regions; (2) data points are marked and output as spatial skylines by mappers and reducers if they are inside the convex hull of $Q$. These data points are needed in reduce functions, because they may spatially dominate data points in category 3; (3) data points are produced with their associated independent regions if they fall in at least one independent region. These data points will be processed by reducers to find spatial skylines in the independent regions. If a data point is inside two or more independent regions, the map function will produce a pair of $<IR.id, p>$ for every associated independent region. After the shuffle phase, data points in $P$ are grouped by independent regions, and sent to reduce functions for spatial skyline calculation in parallel. Finally, the global skyline points are the union of the output of reduce functions. A data point could be associated with two or more independent regions, which may introduce duplicates in the results. We design an elimination method to remove duplicates in our solution. The method will be presented in Section 4.3.3.

Figure 2 shows an example of spatial skyline query over three query points and eight data points ($Q = \{q_1, q_2, q_3, q_4\}, P = \{p_1, \ldots, p_8\}$). First of all, the convex hull of query points ($CH(Q)$) is generated in the first MapReduce phase. Then, the globally optimal independent region pivot is found by using $P$ and $CH(Q)$ in the second MapReduce phase. Each mapper takes a split of $P$ and $CH(Q)$ (a constant global variable), and selects a local optimal independent region pivot, and a reducer outputs the globally optimal pivot. In the third MapReduce phase, each mapper receives a split of $P$, and the pivot and $CH(Q)$ (as two constant global variables), and produces object points with their associated independent regions. In the example, there are three independent regions ($\{IR(p_1, q_1), IR(p_1, q_2), IR(p_1, q_3)\}$). All the independent regions can be calculated from the pivot and $CH(Q)$ in mappers. Moreover, object points are associated with the independent regions where they locate in. If we use $ir_1$, $ir_2$, and $ir_3$ to denote $IR(p_1, q_1)$, $IR(p_1, q_2)$, and $IR(p_1, q_3)$, then $p_1$ is associated with $ir_1$, $p_3$ is associated with $ir_2$ and etc. After the shuffle phase, $<ir_1, p_1>$, $<ir_2, p_2>$, $<ir_3, p_3>$, and $<ir_2, p_6>$ are grouped and sent to the first reducer; $<ir_2, p_1>$, $<ir_2, p_5>$, and $<ir_2, p_6>$ are passed to the second reducer, and $<ir_3, p_1>$, $<ir_3, p_4>$, $<ir_3, p_5>$ are processed in the third reducer. In this case, $p_1$ is a special object point, which is in all three independent regions. As we will discuss our elimination method in Section 4.3.3, $p_1$ will be only output by the first reducer. Thus, the first reducer outputs $p_1$, $p_2$ and $p_6$ as spatial skylines and discards $p_3$ because it is dominated by $p_6$. The second reducer produces $p_5$, $p_6$. The third reducer does not output any object because $p_4$ is eliminated in the spatial dominance test and $p_5$ has been produced in the second reducer. Finally, the result of the spatial skyline query is the union of the results of reducers, which are $\{p_1, p_2, p_5, p_6\}$.

4.2 Spatial Skyline Calculation

In the second and third MapReduce phases, we generate independent regions based on the convex hull of $Q$ and a set of data points $P$ for spatial skyline computation in parallel. In this subsection, we first provide a formal definition of an independent region. Due to high cost of the spatial dominance test that requires comparing the distance from data points to all convex points, we introduce pruning regions in independent regions. A pruning region can be defined by a data point inside $CH(Q)$, a convex point, and its adjacent convex points. If a data point is in a pruning region, the point can be discarded without the dominance test.

**Definition (Independent Region)** Given a data point $p$ and a set of query points $Q$ in a $d$-dimensional space, we define an Independent Region of $p$ and $q_i$, $q_i \in Q$ as a sphere centered at $q_i$ with radius $D(p, q_i)$. An Independent Region Group (IRG) of $p$ with respect to $Q$ is the union of the independent regions, as shown in Figure 2.

$$IRG(p, Q) = \bigcup_{q_i \in Q} IR(p, q_i)$$

We define data point $p$ as the **Independent Region Pivot** of $IRG(p, Q)$ as shown in Figure 2.

With this definition of the independent region, we provide the independence of spatial skylines as follows.

**THEOREM 4.1.** Given a data point $p$ and its independent regions $\{IR(p, q_i) \mid q_i \in CH(Q)\}$, where $CH(Q)$ is the convex hull of query points $Q$, $\forall q_i \in CH(Q)$, any data point $p'$ in $IR(p, q_i)$ is not dominated by any data point $p'' \notin IR(p, q_i)$.

**PROOF.** The proof is by contradiction. Assume that $\exists p' \in IR(p, q_i), p'' \notin IR(p, q_i), p'' \ngeq Q p'$. By the definition of spatial skyline, $p''$ is spatially closer to any query point $q_i (q_i \in Q)$ than $p$. Since $q_i \in CH(Q)$, so $q_i \in Q$ as well. But according to the definition of independent regions, $D(p'', q_i) \geq D(p', q_i)$ since $p''$ is outside of $IR(p, q_i)$, which leads to a contradiction. Thus, this concludes the proof. \(\square\)

The independent regions are determined by the independent region pivot and the convex hull of $Q$. The convex hull is uniquely determined by input query points $Q$; however, theoretically, the pivot can be arbitrarily selected. Since the independent regions specify the search region that contains spatial skyline candidates, an intuitive strategy of the data point selection is to select the data point that minimizes the total volume of its independent regions.

Figure 2 displays an example that utilizes independent regions in the spatial skyline evaluation in $\mathbb{R}^2$. The datasets $P$ and $Q$ consist of 8 data points and 4 query points, respectively. $q_1, q_2, q_3$ are the convex points of the convex hull of $Q$. Three dashed circles indicate three independent regions generated by $p_1$ and the convex points. In this example, $P$ is partitioned into three subsets, which are $P_1 = \{p_1, p_2, p_3, p_6\}$, $P_2 = \{p_4, p_5\}$, and $P_3 = \{p_1, p_5, p_6\}$. $p_1$ and $p_6$ are spatial skylines, because they are in the convex hull [23]. $p_7$ is outside all independent regions and...
can be discarded by mappers in the third phase. \( p_8 \) is in \( IR(p_1, q_2) \) and \( IR(p_1, q_3) \), thus \( p_8 \) is associated with both independent regions. Then, the spatial skylines in independent regions are calculated independently. Figure 4 shows an example of the pruning region in \( IR(p_1, q_2) \) (the formal definition of pruning regions will be presented in Section 4.2.1). \( p_4 \) is a data point that is closer to \( q_1 \) than \( p_1 \). Thus, we create a pruning region \( PR(p_4, q_1) \) (highlighted in gray) in \( IR(p_1, q_1) \) to filter out data points dominated by \( p_4 \). In the example, \( p_2 \) falls in \( PR(p_4, q_1) \) and can be discarded without being compared with \( p_2 \). Thus, \( p_2 \) is the only data point requiring spatial dominance test, comparing its distance to all convex points with the one of \( p_4 \). Our spatial skyline algorithm will be presented in Section 4.2.2.

4.2.1 Pruning Regions in Independent Regions

In the third MapReduce phase, a reduce function calculates spatial skylines of a set of data points in an independent region. In particular, the data points are comparing their distances to all convex points of \( CH(Q) \) with all other data points (spatial skylines do not depend on non-convex points [23]), and the ones are discarded if they are spatially dominated in the same independent region. The data point comparison would be expensive when the number of convex points of \( CH(Q) \) becomes large. Thus, to minimize the cost of the dominance test, we propose a pruning method that is able to efficiently filter out unqualified data points without accessing all convex points of \( CH(Q) \). This method defines a pruning region in each independent region: if data points fall in the pruning region, they can be discarded because there must exist a data point dominating these data points. We will first illustrate the pruning regions in a 2-dimensional space, and then provide a formal definition and proof of the pruning regions in high-dimensional spaces.

Figures 6 and 7 show an example of a pruning region in \( \mathbb{R}^2 \). Given a query point \( q \), two data points \( p \) and \( v \), let \( L_{qv} \) be a line connecting \( q \) to any point \( x \in \mathbb{R}^2 \), and \( L_{qv} \) be the line of \( q \) and \( v \). We build a 2-dimensional Cartesian coordinate system, in which \( q \) is the origin and \( L_{qv} \) is \( x \) axis. \( L_{qv} \) and \( L_{pq} \) partition \( \mathbb{R}^2 \) into two half-spaces, denoted by \( S^+_{qv} \) and \( S^-_{qv} \), and \( S^-_{qv} \) and \( S^+_{qv} \), respectively.

**Theorem 4.2.** If \( p \) and \( v \) satisfy

1. \( v \in S^+_{qv} \) and \( p \in S^-_{qv} \)
2. \( v.x \leq p.x \)
3. \( D(v, q) > D(p, q) \)

then \( p \) spatially dominates \( v \) with respect to any point \( q^* \). \( q^* \in S^+_{qv} \setminus S^-_{qv} \).

**Proof.** As a case of \( p.x \geq 0 \) shown in Figure 6, the three conditions indicate that (1) \( L_{qv} \) partitions \( v \) and \( p \) into two half-spaces, \( v.y > 0, p.y < 0 \), (2) if we create a line \( L_{pq} \) perpendicular to \( L_{qv} \), then \( L_{pq} \) results in the case of \( p.x < 0 \). Then, \( p \) spatially dominates \( v \) with respect to \( q^* \).

**Figure 4:** An example of Pruning Regions in \( \mathbb{R}^2 \).

**Figure 5:** An example of merged Independent Regions in \( \mathbb{R}^2 \).

Figure 6: A pruning region Figure 7: An example of using \( p \) and a visible facet \( L_{qv}, \) \( PR(p, q) \).

then \( v \) is at the left side of \( L_{qv} \); (3) \( v \) is outside of the circle centered at \( q \) with radius \( D(p, q) \). The circle intersects with \( L_{qv} \) at \( p \) and \( b \).

Given any data point \( q^* \) or \( q^{**} \) in \( S^+_{qv} \setminus S^-_{qv} \) (highlighted in gray), \( L_{qv} \) must intersect with either an arc from \( a \) to \( b \) (\( \text{arc}_{ab} \)) or a line from \( b \) to \( c \) (\( \text{arc}_{cb} \)). If \( L_{qv} \) intersects with \( \text{arc}_{ab} \) at \( e \), then we can get \( D(p, q) = D(e, q) \) and \( v.x > e.x \). Given a query point \( q_1 = \{q^*, x, 0\} \) on \( L_{qv} \), then

\[
D(e, q_1) = \sqrt{(e.x - q^*.x)^2 + (e.y)^2} \\
= \sqrt{D(e, q)^2 - 2 \cdot (e.x) \cdot (q^*.x) + (q^*.x)^2} \\
> \sqrt{D(p, q)^2 - 2 \cdot (p.x) \cdot (q^*.x) + (q^*.x)^2} \\
= \sqrt{(p.x - q^*.x)^2 + (p.y)^2} \\
= D(p, q)
\]

Thus, \( D(p, q_1) < D(v, q) \). If \( q_1 \) is moved to \( q^* \), \( q^*.y < 0 \), then \( D(p, q^*) < D(v, q^*) \) is also held. On the other hand, if \( L_{qv} \) intersects with \( L_{bc} \) at \( e' \), then \( D(p, q^*') < D(e', q^*') \), because \( L_{qv} \) is the bisection line of \( p \) and \( b \), and both \( p \) and \( q^* \) are in \( S^+_{qv} \). Thus, \( D(p, q^*') < D(e', q^*') \leq D(v, q^*) \). We can get the similar result in the case of \( p.x < 0 \). Therefore, \( p \) spatially dominates \( v \) with respect to any query point in \( S^+_{qv} \setminus S^-_{qv} \).

In a 2-dimensional space, a convex hull is a convex polygon, in which each convex point has two adjacent convex points. Figure 7 shows three convex points \( q_1, q_2, \) and \( q_3 \) of a convex hull \( CH(Q) \). \( q_1 \), \( q_2 \), and \( q_3 \) are adjacent to \( q_1 \). Line segments \( L_{q_1 q_2} \) and \( L_{q_2 q_3} \) are two visible facets from a data point \( v \) outside \( CH(Q) \).

**Theorem 4.3.** In a 2-dimensional space, given a query point \( q_i \in CH(Q) \) and a data point \( v \) outside \( CH(Q) \), let \( A^\infty_{q_i} \) be a set of \( \text{adjacent convex points of } q_i \), and \( p \) be an invisible data point from \( v \). Each of the lines from \( p \) perpendicular to \( L_{q_i q_{j+1}} \) \( q_j \in A^\infty_{q_i} \) partitions the space into two closed half-spaces. Let \( S^-_{q_{j+1} q_i} \) be the half space containing \( q_i \). Then, any data point \( v \) outside \( CH(Q) \) satisfies

1. \( v \in S^-_{q_{j+1} q_i} \) \( q_j \in A^\infty_{q_i} \)
2. \( D(v, q_i) > D(p, q_i) \)

is spatially dominated by \( p \) with respect to \( Q \).

**Proof.** In Figure 7, \( L_{ab} \) and \( L_{cd} \) are two lines from \( p \) perpendicular to \( L_{q_1 q_2} \) and \( L_{q_2 q_3} \), respectively. \( L_{ab} \) and \( L_{cd} \) separate \( v \) from the convex hull \( CH(Q) \). \( L_{pq} \) partitions \( CH(Q) \) into two closed half regions, \( G^- \) and \( G^+ \), all convex points are in either \( G^- \) or \( G^+ \). If a convex point \( q^* \) is in \( G^- \), we can easily get \( D(p, q^*) < D(v, q^*) \) by using Theorem 4.2. The similar result can be obtained in the case that any convex point \( q^{**} \) is in \( G^+ \). Thus, \( v \) is spatially dominated by \( p \) with respect to \( Q \).
After an illustration of pruning regions in $\mathbb{R}^2$, we extend the concept of pruning regions to high-dimensional spaces. In a d-dimensional space, a convex hull of query points $Q$ can be represented by a set of half-spaces $H$, where $CH(Q) = \bigcap_{h \in H} h^\circ$. The bisector hyper-plane of each half space contains a (d-1)-dimensional facet of the convex hull, which can be determined by a convex point and a subset of its adjacent convex points. The formal definition of the pruning regions is provided as follows.

**Definition (Pruning Regions)** In a d-dimensional space, given a convex hull of query points $CH(Q)$, a data point $v$ outside $CH(Q)$, a visible convex point $q$, and an invisible data point $p$ from $v$, let $A^\circ_q$ be a set of adjacent convex points of $q$ in facets, $h^\perp_{pq}$ be the (d-1)-dimensional hyper-plane that contains $p$ and is perpendicular to $L_{pq}^\perp$, $q_j \in A^\circ_q \cup h^\perp_{pq}$ partitions the space into two closed half-spaces; the one containing $q$ is denoted by $S^-_{h^\perp_{pq}}$. Then any data point $v$ outside $CH(Q)$ satisfying

\begin{align}
(1) & \quad v \in S^-_{h^\perp_{pq}}, \quad q_j \in A^\circ_q \\
(2) & \quad D(v, q) > D(p, q)
\end{align}

(7)

is spatially dominated by $p$ with respect to $Q$. The region containing all possible $v$ is called a Pruning Region of $p$ and $q$, denoted by $PR(p, q)$.

**Proof.** The proof is by induction. The pruning region in a 2-dimensional space has been proven in Theorem 4.3. We assume that the pruning region is held in an (i-1)-dimensional space ($i \geq 3$), then, in an i-dimensional space, since $v$ is outside $CH(Q)$, the line connecting $v$ with any convex point $q^*$ must intersect with a visible closed (i-1)-dimensional facet $F$ of $CH(Q)$. The hyper-sphere centered at $q$ with radius $D(p, q)$ and $h^\perp_{pq}$ ($q_j \in A^\circ_q$) separate $v$ from $CH(Q)$. If a ray from $v$ to $q^*$ intersects with the hyper-sphere at $e$ earlier than any $h^\perp_{pq}$ ($q_j \in A^\circ_q$), then given any convex point $q_k \in A^\circ_q$, we can build an i-dimensional Cartesian coordinate system, in which $F$ is a hyper-plane ($x_i = 0$, $v.x_i > 0$, $p.x_i < 0$, and $L_{q_k} \in F$) is $x$ axis. Any query point on $L_{q_k}$ can be represented by $[x_1, 0, ..., 0]$. Since $D(e, q_j) = D(p, q)$ and $e.x_i < p.x_i$, given any query point $q_k$ on $L_{q_k}$, we get

\[ D(e, q_k) = \sqrt{(e.x_i - q_k.x_i)^2 + (e.q_i)^2} \]

\[ = \sqrt{D(e, q)^2 - 2 \cdot (e.x_i) \cdot (q_k.x_i) + (q_k.x_i)^2} > \sqrt{D(p, q)^2 - 2 \cdot (p.x_i) \cdot (q_k.x_i) + (q_k.x_i)^2} \]

(8)

\[ = \sqrt{(p.x_i - q_k.x_i)^2 + D(p, q_k)^2} = D(p, q_k) \]

where $D(e, L_{q_k})$ denotes the distance from $e$ to the line $L_{q_k}$. Thus, we can get that, given any query point $q^*$ satisfying $D(p, q^*) \leq D(e, q^*)$, if $q^*$ is moved to $q''$ on any of the directions from $q$ to its adjacent convex points in $F$, $D(p, q'') < D(e, q'')$ is also held. So, $p$ is closer to any query point in $F$ than $e$. Since, $v.x_i > 0$, $p.x_i < 0$, and $q''_i.x_i < 0$, so $D(p, q'') < D(v, q'')$. On the other hand, if a ray from $e$ to $q''$ first intersects with $h^\perp_{pq}$, then $q''$ is the center of the intersection of $h^\perp_{pq}$ and the hyper-sphere centered at $q$ with radius $D(p, q)$. $h^\perp_{pq}$ is an (i-1)-dimensional hyper-plane, in which $D(e, q'') > D(p, q'')$, and $e'' \in S^-_{h^\perp_{pq}}$, $q_\neq q_k, q_\neq q_k, q_\neq q_k, q_\neq q_k$, which satisfies the conditions in an (i-1)-dimensional space. Thus, $D(p, q'') < D(v, q'')$; then $D(p, q'') < D(e, q'')$. Therefore, this concludes the proof. \( \square \)

Figure 8 shows an example of the convex hull of query points $Q$ in a 3-dimensional space. $v$ is a data point outside the convex hull. The line $L_{q_{i-1}}$ intersects with a visible facet $F$ at $e$. $F$ can be determined by three convex points $q_{i-1}, q_i, q_{i+1}$. After a coordinate transformation, $F$ is transformed to be on plane $Z$ ($z = 0$), $v.z > 0$, and $p.z < 0$, as displayed in Figure 9. $L_{q_{i+1}}$ is the $x$ axis in plane $Z$. The area invisible from $v$, including $p$, is highlighted in gray. Two hyper-planes $h^\perp_{q_{i+1}q_i}$ and $h^\perp_{q_{i+1}q_i}$ are highlighted in red. $q_i$ and $q_{i-1}$ are two elements of $A^\circ_q$. If a ray from $v$ to $q''$ first intersects with the sphere centered at $q_i$ with radius $D(p, q_i)$ at $e$, then according to Equation 8, we can get that any point $q''$ on $L_{q_{i+1}q_i}$, $q''_i \geq q_i, q''_i < q_i, q''_i > D(p, q_i) \leq D(e, q''_i)$, and moving the point on the direction from $q_{i-1}$ to $q_{i+1}$ with distance $\Delta z (\Delta z > 0)$ makes the point closer to $p$ than $e$. The similar result can be obtained on the direction from $q_i$ to $q_{i-1}$.

4.2.2 Spatial Skyline Algorithm

Ams the concept of pruning regions, we present our spatial skyline algorithm used in reduce functions of the third phase. The input data points are grouped by their independent regions through the shuffle phase. With the concept of pruning regions, the input data points are grouped by their independent regions through the shuffle phase. With the concept of pruning regions, the input data points are grouped by their independent regions through the shuffle phase. With the concept of pruning regions, the input data points are grouped by their independent regions through the shuffle phase.

Figure 8: An example of visible facets of a convex hull in a 2-dimensional space.

Figure 9: An example of a visible facet after a coordinate transformation.
Algorithm 1 Spatial Skyline Algorithm

Input: \( P_i, CH(Q) \)
Output: \( lssky \cup chsky \)

1: \( lssky = \emptyset; \)
2: \( chsky = \emptyset; \)
3: \( PR = \emptyset; \)
4: for \( \forall p \in P_i \) do
5:     if \( p \) is inside \( CH(Q) \) then
6:         \( chsky = chsky \cup \{p\}; \)
7:         \( PR = PR \cup PR(p, q_i); \)
8:     else
9:         \( lssky = lssky \cup \{p\}; \)
10: end if
11: end for
12: for \( \forall p \in lssky \) do
13:     if \( p \) is in \( PR \) then
14:         \( lssky = lssky - \{p\}; \)
15:     Continue;
16: end if
17: if \( \exists p' \in (chsky \cup lssky), p' \neq p, p' \prec Q p \) then
18:     \( lssky = lssky - \{p\}; \)
19: end if
20: end for
21: return \( lssky \cup chsky; \)

Figure 10: An example of \( Grid(lssky \cup chsky) \).
Figure 11: An example of \( Grid(DR(lssky \cup chsky)) \).

In this subsection, we discuss three implementation issues in our solutions.

4.3.1 Independent Region Pivot Selection

In the second MapReduce phase, the search space is partitioned into a number of independent regions. The spatial skylines are calculated in parallel by reducers in the third MapReduce phase. The execution time of a parallel program is determined by the slowest process, and the spatial skyline algorithm takes longer on larger inputs. Thus, distributing the data points to reducers in a balanced manner is critical to our approach.

If the data points are uniformly distributed in the search space, the number of data points in an independent region is proportional to the volume of the independent region, which depends on the distance between the independent region pivot and the convex point. Theoretically, the point with equal distance to all convex points is the optimal independent region pivot, which could split data points in equal size. But the optimal pivot may not exist in irregular convex hull. Moreover, the point that minimizes the total volume of independent regions would be an alternative optimal pivot. However, the cost of finding the point is expensive. Thus, we turn to an approximation method in our implementation. After the convex hull is calculated, we choose the center of the Minimum Bounding Rectangle (MBR) of the convex hull as the independent region pivot. The experimental results of varying independent region pivots can be found in Section 5.6.

4.3.2 Independent Region Merging

In the third phase of our solution, a reducer processes data points in an independent region. The number of reducers needed in the spatial skyline calculation depends on the number of independent regions or the number of convex points in the convex hull of query points \( Q \). Since the size of the convex hull would be large, the task maintenance and communication overhead in MapReduce framework would be unacceptably high.

Thus, there are two merging strategies that can be applied to our proposed solution if the number of independent regions is much greater than the number of available computing resources. In the strategies, we assume that objects are uniformly distributed in the search space. The smaller the total volume of independent regions is, the less the number of objects is processed in spatial skyline computation.

**Shortest distance merging.** In this method, we merge the closest pair of two neighboring independent regions. The distance of two independent regions is evaluated by the distance between the centers of the independent regions. We assume that there is higher possibility that two independent regions overlap with each other if they are close. Merging two overlapped independent regions may reduce the cost of spatial skyline computation for the following two reasons: (1) the objects in the overlapping region are fed to one reducer instead of two, which minimizes the total number of objects in the spatial dominance test; (2) the pruning regions of the independent regions are also merged; more objects could be eliminated without the dominance test. Take Figure 5 for example, \( q_1 \) and \( q_2 \) are the closest pair of convex points in the figure. \( IR(p_1, q_1) \) and \( IR(p_1, q_2) \) are merged, and the new independent region is denoted by \( IR(p_1, \{q_1, q_2\}) \). So, \( p_1 \) and \( p_2 \) are only processed by the reducer, which receives \( IR(p_1, \{q_1, q_2\}) \). The pruning region \( IR(p_1, \{q_1, q_2\}) \) is \( PR(p_1, q_1) \cup PR(p_1, q_2) \).

In our implementation, we iterate the convex hull in counter clockwise order, and calculate the distance between every pair of two consecutive independent regions. Let \( n \) be the number of computing resources available to the spatial skyline evaluation and \( m \).
be the number of convex points, we will merge the top \( m - n \) (\( m \geq n \)) closest pairs of the independent regions (the number of pairs of independent regions is equal to the number of convex points).

Threshold-based merging. An alternative strategy merges independent regions by considering the volume of the overlapping region of two independent regions. In this method, we visit the independent regions in counter clockwise order. Given two consecutive independent regions, we calculate the ratio of volume of the overlapping region of the two independent regions to the volume of the smaller independent region. If the ratio is higher than a specific threshold, the two independent regions will be merged. Another difference from the first method is that two or more independent regions may be merged if they are close to each other. The ratio can be defined as follows.

\[
\text{ratio}(q_1, q_2) = \frac{\text{Vol}^d(\text{IR}(p_1, q_1)) \cap \text{Vol}^d(\text{IR}(p_1, q_2))}{\text{Vol}^d(\text{IR}(p_1, q_2))} \tag{9}
\]

where \( \text{IR}(p_1, q_1) \) and \( \text{IR}(p_1, q_2) \) are two consecutive independent regions, \( \text{Vol}^d(\text{IR}(p_1, q_1)) \) denotes the volume of \( \text{IR}(p_1, q_1) \) in a \( d \)-dimensional space, and \( \text{Vol}^d(\text{IR}(p_1, q_1)) \geq \text{Vol}^d(\text{IR}(p_1, q_2)) \).

Moreover, the volume of the overlapping region of two independent regions (two spheres) can be calculated as follows (See Figures 12 and 13).

\[
\text{Vol}^d(\text{IR}(p_1, q_1) \cap \text{IR}(p_1, q_2)) = \int_{u_0}^{r_1} \text{Vol}^{d-1}(h) du + \int_{t_0}^{r_2} \text{Vol}^{d-1}(h) dt \tag{10}
\]

where \( \text{Vol}^{d-1}(h) \) is the volume of the sphere with radius \( h \) in a \( (d-1) \)-dimensional space, \( h = (r_1^2 - t^2)^{1/2} \) and \( t \) is the lower bound of the integrals, where \( u_0 = \frac{r_1^2 - r_2^2 + D(q_1, q_2)^2}{2D(q_1, q_2)} \) and \( t_0 = \frac{r_2^2 - r_1^2 + D(q_1, q_2)^2}{2D(q_1, q_2)} \). \( D(q_1, q_2) \) denotes the distance between \( q_1 \) and \( q_2 \).

Figure 12 shows an example of the independent region merging in a 2-dimensional space. Line \( l_{p} \) decomposes the overlapping region into two sub-regions. The length of \( l_{p} \) is denoted by \( V^1(h) = 2h \). If we move \( l_{p} \) towards \( q_1 \) and \( q_2 \), respectively, then, the volume of the two sub-regions is the sum of integral of \( V^{d-1}(h) \) in the overlapping area. In a \( d \)-dimensional space, \( l_{p} \) becomes a sphere in a \( (d-1) \)-dimensional hyper-plane, and \( h \) is the radius of the sphere.

In a 2-dimensional space, \( \text{ratio}(q_1, q_2) \) can be calculated as follows,

\[
\text{ratio}(q_1, q_2) = \frac{\text{Vol}^2(\text{IR}(p_1, q_1)) \cap \text{Vol}^2(\text{IR}(p_1, q_2))}{\text{Vol}^2(\text{IR}(p_1, q_2))} = \frac{\int_{u_0}^{r_1} \text{Vol}^{1}(h) du + \int_{t_0}^{r_2} \text{Vol}^{1}(h) dt}{\text{Vol}^2(\text{IR}(p_1, q_2))} = \frac{\frac{r_1^2 \cos^{-1} \left( \frac{r_1^2 - r_2^2 + D(q_1, q_2)^2}{2r_1(r_1 - r_2)} \right)}{r_1^2} + \frac{r_2^2 \cos^{-1} \left( \frac{r_2^2 - r_1^2 + D(q_1, q_2)^2}{2r_2(r_2 - r_1)} \right)}{r_2^2}}{\pi r_1^2} \tag{11}
\]

### 4.3.3 Duplicate Elimination

The third issue is that our solution may produce duplicates since a data point may locate in two or more independent regions. If the data point is a spatial skyline, it will be written to the results of the query by multiple reducers. To eliminate the duplicates, we associate a unique independent region identifier to each data point, which indicates that the data point will be output as a spatial skyline by the reducer which processes data points in the independent region. Reducers processing data points in other independent regions will not output the data point even if it is a spatial skyline. Take Figure 4 as an example, \( p_5 \) is a data point in \( \text{IR}(p_1, q_2) \) and \( \text{IR}(p_1, q_4) \). If the identifier of \( \text{IR}(p_1, q_2) \) is associated with \( p_5 \), and \( p_5 \) is a spatial skyline, \( p_5 \) is output only by the reducer processing data points in \( \text{IR}(p_1, q_2) \).

### 5. EXPERIMENTAL VALIDATION

In this section, we evaluate the performance of the proposed MapReduce-based solution over synthetic and real-word datasets. Our proposed algorithm is denoted by \( \text{PSSKY-G-IR-PR} \), which combines the concepts of independent regions, pruning regions, and multi-level grid data structure for efficient query evaluation.

Since none of the existing solutions can be easily extended to address spatial skyline queries in parallel, we developed two single-phase MapReduce-based solutions as baselines, \( \text{PSSKY} \) and \( \text{PSSKY-G} \). \( \text{PSSKY} \) applies a random data partitioning method to split data points. Each mapper uses BNL to produce local spatial skylines by comparing every pair of data points, and a reducer merges the local results and outputs the global spatial skylines. \( \text{PSSKY-G} \) works similarly to \( \text{PSSKY} \) except that \( \text{PSSKY-G} \) utilizes multi-level grid data structure for efficient spatial dominance test. Since all three solutions use the same algorithm in convex hull computation, we will focus primarily on the investigation of the overall performance of solutions and the effect of independent regions and pruning regions on spatial skyline computation in the second and third MapReduce phases. All solutions were implemented in Java on Hadoop 2.6, which is an open source implementation of the MapReduce framework [1].

In the experiments, the real-word datasets were downloaded from Geonames1. We retrieved 11 million objects (streams, schools, etc.) in the United States, and used them as data points and query points. The data points in our synthetic datasets are randomly generated under uniform distribution in a 2-dimensional space. Similarly with [23], the query points were also generated in a specified region at the center of the search space. By default, there are 10 convex points in the convex hull of query points. The area covered by the Minimum Bounding Rectangle (MBR) of query points is fixed at 1% of the search space. The experiments were conducted on a 12-node shared-nothing cluster. Each node is equipped with 19 Intel Xeon 2.2 GHz processors and 128 GBytes of memory. All nodes are connected by Gigabit Ethernet network. All results were recorded after the system model reached a steady state.

1http://www.geonames.org/
5.1 Scalability with Cardinality

First of all, we evaluate the effect of data cardinality on all three solutions. We vary the cardinality of both synthetic and real-world datasets from 100 to 500 million and 2 to 10 million data points, respectively. As Figure 14 displays, the execution time of all solutions increase when the datasets grow. The growth rate of PSSKY-G-IR-PR over synthetic datasets is lower than PSSKY and PSSKY-G. In addition, on average, PSSKY-G-IR-PR executes around 90% faster than PSSKY and 32% faster than PSSKY-G, respectively. The reason is that PSSKY-G-IR-PR is able to parallelize the spatial skyline evaluation by applying the concept of independent regions and efficiently filter out unqualified data points in pruning regions. Moreover, a performance improvement was observed when comparing PSSKY-G with PSSKY, because the multi-level grid data structure is employed to efficiently access the proximity information of data points for the dominance test.

5.2 Effect of Independent Regions and Pruning Regions on Spatial Skyline Algorithms

To evaluate the effectiveness of independent regions and pruning regions on the query evaluation, we compare the execution time of spatial skyline computation in PSSKY-G-IR-PR (the execution time of reducers in the third MapReduce phase) with the ones in PSSKY and PSSKY-G. The cardinality of synthetic and real-world datasets varies from 100 to 500 million and 2 to 10 million data points. As Figure 15 shows, the execution time of all solutions increase when the datasets grow. The execution time of PSSKY increases rapidly due to high complexity of spatial skyline computation. The growth rate of PSSKY-G-IR-PR is the lowest, because all data points can be processed in parallel and a significant portion of data points can be discarded without dominance test. Moreover, the reducer that merges spatial skylines becomes a bottleneck in PSSKY and PSSKY-G, which consumes 50% to 90% of the total execution time over large synthetic and real-world datasets.

5.3 Effect of Number of Nodes

We evaluate the speedup of proposed solutions by scaling up the size of our cluster. The real and uniform datasets are fixed at 10 million and 100 million objects. The cardinality of cluster nodes varies from 2 to 12.

In Figure 17, the execution time of all solutions drops as the size of the cluster increases. As expected, PSSKY always consumes more execution time than PSSKY-G and PSSKY-G-IR-PR while scaling up the cluster. On average, PSSKY-G-IR-PR enjoys the highest dropping rate. Take experiments on real world data for example, PSSKY-G-IR-PR drops 34.35% when scaling up to 8 nodes, PSSKY-G only drops 27%; the dropping rate of PSSKY is constantly lower than 20% in all experiments. The reason is that more map or reduce tasks can be executed in parallel with more computing resources. However, even all three methods will take advantage of mapper parallelism, only reducers of PSSKY-G-IR-PR run in parallel, because the global region is partitioned into independent regions; the skyline results in each independent region do not depend on the ones in other independent regions.

For both synthetic and real data, PSSKY-G-IR-PR performs approximately 50% better than PSSKY-G and 80% better than PSSKY in terms of execution time.

5.4 Effect of Pruning Regions

We also evaluate the effect of pruning regions by comparing the number of dominance tests among three solutions. The cardinality of synthetic and real-world datasets varies from 100 to 500 million and 2 to 10 million data points. Figure 16 displays the number of dominance test in the three solutions over the datasets. As expected, PSSKY always suffers from more dominance tests than PSSKY-G and PSSKY-G-IR-PR. Using multi-level grid data structure can reduce the cost of the dominance test, because, instead of access all data points, PSSKY-G only needs to visit data points in the cells that intersect with dominator regions of other data points. Moreover, the effect of pruning regions can be observed by comparing the results of PSSKY-G and PSSKY-G-IR-PR. Although data points locating at two or more independent regions may introduce subtle overhead in data point comparison, PSSKY-G-IR-PR can save more time in the dominance test by utilizing the concept of pruning regions. According to our experiments, there are a small number of duplicate data points generated.
by PSSKY-G-IR-PR. Combining the experiments shown in Figure 15, PSSKY-G-IR-PR only takes a few minutes longer if there are additional 100 million dominance test are performed by PSSKY-G-IR-PR. Note that these dominance tests may be conducted in parallel in PSSKY-G-IR-PR.

Table 2 shows the power of pruning regions in terms of data point reduction rate by varying cardinality of datasets. The reduction rate is defined by the average percentage of data points eliminated by pruning regions in independent regions. We find that around 27% of data points in synthetic datasets fall in pruning regions, and can be discarded by PSSKY-G-IR-PR while there are around 9% of real-world data points that can be pruned without the dominance test. Moreover, the object elimination rate is slightly changed for large real-world datasets. The reason is that the number of data points varies in the experiments, but the number of query points and its convex hull are fixed. Theoretically, if data points are uniformly distributed, the effectiveness of the Pruning Region only depends on the volume of Pruning Regions. Increasing the density of data points over a large data point datasets does not help much to generate larger pruning regions. Thus, the reduction rate over synthetic datasets remains unchanged, and there is slight change in the reduction rate over real-world datasets due to the non-uniform distribution of data points.

Table 3 shows the power of pruning regions in terms of data point reduction rate by varying the distribution of data points. We replace 5%, 10%, 15%, and 20% of uniform data points with anti-correlated data points. For example, the experiments performed over datasets with 20% anti-correlated and 80% uniform data points are displayed in the first row of the results in Table 3. We find that the reduction rate remains the same over datasets under the same distribution of data points. Moreover, when 20% of anti-correlated data points are generated in datasets, only 2% difference is observed in the experiments, which tells us that the ratio of the volume of independent regions and pruning regions to that of the search space is small. In other words, if 20% data points are moved to the central area of the search space, there are only 2% of data points moved outside the pruning regions.

5.5 Effect of Query Points

We investigate the effect of the area covered by the convex hull of query points on the solutions in this subsection. We fix the size of data points at 100 million. The ratio of the area covered by the MBR of query points to the search space ranges from 1% to 2.5%. The number of Convex Hull query points selected for real-world datasets is 10, 14, 17, and 23, and that for synthetic datasets are 10, 12, 14, and 16, respectively. Figure 18 displays the overall execution time of solutions over synthetic and real-world datasets. The ratios of the MBR of the convex hull of query points are indicated by $x$ axis. Intuitively, a larger convex hull may help to reduce the cost of the dominance test because more data points would locate in the convex hull, and can be output as spatial skylines without dominance test. However, our experimental results show that the entire process of query evaluation takes longer. The reason is that there are more data points in the search region, and the number of data points requiring dominance test becomes larger. Take Figure 2 for example, a convex hull is represented by $q_1$, $q_2$, and $q_3$. $p_1$ is used to generate three independent regions. If the convex hull grows larger, the area covered by the three independent regions will become larger accordingly. Thus, more data will be located in the independent regions, and be processed by reducers in the third MapReduce phase.

The evidence is also displayed in Figure 19 and 20. Figure 19 shows the execution time of spatial skyline computation and the number of dominance tests grow rapidly when the MBRs of the convex hull of query points cover larger areas. A similar results are observed in terms of the number of dominance test in Figure 20.

5.6 Effect of Independent Region Pivot Selection

We investigate the effect of independent region pivot selection on the query evaluation by varying the pivot on real world datasets.
randomly choose three query dataset in such a way that the convex hull of the query points is in random shape. Figure 20 shows the execution time of the query when the pivot locates at the query points with minimum and maximum y-coordinate values (p3 and p1), and the center of MBR(CH(Q)) (p0). Two additional pivots are selected at the midpoint of the line of p1 and p3, and the line of p0 and p3. The two pivots are denoted by p2 and p4, respectively. In general, p3 is closer to the optimal pivot than other four pivots, and PSSKY-G-IR-PR runs faster when p3 is selected as the independent region pivot.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose an advanced parallel spatial skyline solution utilizing MapReduce framework. Given a set of data points and a set of query points, our approach first calculates the convex hull of the query points. Then, we propose a novel concept of independent regions; the input data points are partitioned by their associated independent regions. Unqualified data points outside independent regions can be eliminated without the dominance test. Finally, all spatial skylines in independent regions are calculated in parallel, and the global spatial skylines are the union of local spatial skylines. Moreover, to avoid high cost of data point comparison, we propose a concept of pruning regions, in which objects can be discarded without comparing their distance to all convex hull query points. We demonstrate the efficiency and effectiveness of the proposed solution through extensive experiments on large-scale real-world, and synthetic datasets.

We plan to extend the proposed parallel solution to address spatial skyline queries on road networks. Theoretically, the concepts of independent regions and pruning regions can be applied in the space of road networks. However, more investigation is needed to evaluate the cost of calculating the independent regions and pruning regions. Due to variety of data distribution, it is also interesting to study the pruning power of pruning regions on road networks.

Acknowledgments

This research has been funded in part by the National Science Foundation grants IIS-1618669 (III) and ACI-1642133 (CICI).

References