Cluster Computing (2022) 25:3249–3264
https://doi.org/10.1007/s10586-022-03559-6

Weighted spatial skyline queries with distributed dominance tests

Elaheh Gavagsaz[1]

Received: 16 November 2021 / Revised: 23 January 2022 / Accepted: 2 February 2022 / Published online: 17 February 2022

Abstract
In this paper, we study the retrieval of desired or relevant points of interest, a set of spatial skyline points, related to a set of query points to establish distance restrictions. Note that each point of interest has different importance, assigned to each of them as weight, and the weighted Euclidean distance is used. In order to efficiently handle the weighted spatial skyline queries, this research presents a novel MapReduce-based solution for the first time. The proposed method prevents the bottleneck of centrally finding the global skyline from the local skylines and reduces the dominance test by performing the necessary dominance tests in parallel. Finally, the experimental results show that the proposed method obtains significant performance improvement.

Keywords Decision-making support system · Spatial skyline query · Weighted Euclidean distance · Parallel processing · MapReduce

1 Introduction

Despite an increasing amount of data, the process of retrieving high-relevance information efficiently about a given query and separating irrelevant information is becoming increasingly important. The study of the problem of maximum vectors in the database context led to the proposal of several efficient algorithms for skyline queries [1]. Methods using indexing techniques [2–6] perform better because they can access the data through the index instead of the entire data set, although their applicability is limited due to collecting index data. Other approaches [1, 7–11] are more common because no specialized access structure is required to calculate the skyline set.

There are several types of skyline queries. Each of them maintains the main idea and notation of the skyline query and solves different aspects of the problem. Spatial skyline query is a novel type of skyline query that prioritizes both static and dynamic object attributes in multi-criteria decision-making applications [12]. In spatial skyline query, the distance between objects is a dynamic object attribute different from skyline query, which considers only static object attributes.

In particular, a set of data points of interest and a set of query points are the two main elements of a spatial skyline query. Query points define distance constraints on the set of points of interest. The spatial skyline query returns a subset of points of interest not spatially dominated by other points of interest.

Spatial skyline query applies to many applications such as facility location, crisis management, trips, or event planning. Take trip planning applications as an example. Spatial skyline queries help to select the desired set of hotels based on their distance from attractions of interest. Assume that some residential buildings must be evacuated because of several explosions or fires. The spatial skyline query helps to identify the first buildings to be evacuated.

The suitable hotels were selected by the traditional spatial skyline points, which only consider their distance to the query points. But this scenery is not realistic. When choosing a hotel, users also take into account several other aspects of the hotel, such as prices or opinions given by previous users. These aspects combine to obtain an attraction value to the hotel. Each hotel receives the obtained attraction value as a weight. The larger the weight associated with a hotel, the more attractive it is.

Notably, the Euclidean distance may not be able to model realistic scenery. Therefore, we assign a weight to

✉ Elaheh Gavagsaz
elaheh.gavagsaz@srbiau.ac.ir; egavagsaz@yahoo.com

1 Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

each point of interest (for example, hotels and residential buildings) to reflect their importance. The weighted Euclidean distance is the product of the Euclidean distance and the inverse of the facility weight [13]. It is notable that the higher the weight, the smaller the weighted distance for the Euclidean distance. Although there are practical applications in spatial skyline query considering weighted Euclidean distance, it has not received much attention at present.

Note that many of the features used to solve spatial skyline query become incorrect in the weighted case [13]. For instance, the convex hull and Voronoi diagram accelerate the calculation of spatial skylines. However, they are not practical for weighted spatial skylines. Some efficient index structures [14] relying on the triangular inequality are used in spatial skylines to reduce the dominance tests. When considering weights, they are not correct. Therefore, most spatial features that reduce computation are generally not applicable to weighted spatial skylines. Consequently, we have to use strategies that require several scans of the entire data set.

The mentioned problems motivate us to use a parallel solution for weighted spatial skyline queries. An increasing number of methods have been proposed for skyline queries in parallel and distributed environments [2, 15–23]. There is only one parallel solution for spatial skyline queries using the MapReduce framework [24]. It uses the convex hull of the query points to define the independent regions. We know that the properties of the convex hull cannot be used for weighted spatial skyline queries. The only existing parallel solution for weighted spatial skyline queries is [13] using the GPU. It is notable that GPU hardware has limited memory, and the MapReduce scheme can efficiently handle large-scale data sets.

For the first time, this paper proposes a novel one-phase MapReduce-based solution for weighted spatial skyline queries called MR-WSS considering the weighted Euclidean distance. Our proposed method divides a MapReduce job into two phases: Phase 1: mapping data points, computing local weighted skyline points. Phase 2: verifying and reducing local weighted skyline points. As we all know, MapReduce is a divide-and-conquer model. Furthermore, obtaining a weighted spatial skyline set is not a decomposable problem. That means that the union of the results of reducers is not the final result for the weighted spatial skyline set. Hence, it is difficult to process a weighted spatial skyline query on large-scale data points in MapReduce. To address this deficiency, each reducer receives a dominatee set and several dominator sets in the second phase from mappers. Then, it verifies the data points in the dominatee set to be dominated by at least a data point in dominator sets. Finally, the results of all reducers create the final weighted spatial skyline points set.

The main contributions of this paper can be summarized as follows:

- We propose an effective one-phase MapReduce algorithm to evaluate the weighted spatial skyline queries on large data sets. To the best of our knowledge, it is possibly the first of this work.
- We introduce an adaptation of traditional methods to compute the local set of weighted spatial skyline points in the mappers for decreasing the computation overhead.
- We propose a distributed dominance test to calculate the final set of weighted spatial skyline points in the reducers.
- We theoretically and experimentally analyze our method and show the efficiency and effectiveness of our proposed algorithm.

The rest of this paper is organized as follows. Section 2 introduces related works, including the traditional skyline problems. We study the basic concepts of general spatial skyline query in Sect. 3. We also describe weighted Euclidean distance and the properties of the spatial skyline query under this distance. Section 4 presents the proposed solution based on MapReduce to handle weighted spatial skyline queries and theoretical analysis. The experimental validation of the proposed algorithm is shown in Sect. 5. Section 6 concludes this paper.

2 Related works

There are many studies on skyline computation. Skyline queries were first studied as maximal vectors and then introduced in database context [1]. Then, different algorithms were proposed to compute the skyline [5, 9, 25, 26]. Some studies solve the dimensionality problem of skyline queries due to the dimensional growth of data [27–29]. Some papers have introduced the dynamic skylines queries [5, 12, 30, 31]. They survey a moving point of interest that changes the distance to the candidate points. The reverse skylines queries are another research topic studying the changes in the skyline points when finding a new candidate. The approaches proposed to this problem are based on the dynamic skylines [3, 32].

As a specific case of dynamic skyline query, in [12], spatial skyline query was first proposed. Two efficient algorithms, Branch and Bound Spatial Skyline (B^2S^2) and Voronoi based Spatial Skyline (VS^2), are introduced for the static query points. The set of data points of interest uses a data partitioning method, such as R-tree in the B^2S^2 algorithm for indexing. This method constructs a convex hull on the query points Then, it uses the properties of the convex hull to reduce the check of all pairs of data points in the interest points and the query points. The B^2S^2 algorithm scans space by traversing the R-tree from its root to leaves. In this algorithm, the *mindist* is the sum of distances between a point and the set of data points. When discovering the first spatial skyline, the R-tree expands with the node with the least *mindist*. Next, it performs the dominance test between the found node and all the candidates of the spatial skyline identified so far. This process continues until all intermediate nodes, including spatial skylines, have been visited. Another method, VS^2, generates a Voronoi diagram of the points of interest. Then, it considers the closest data points to the query points and traverses the space to access the neighborhood of the data points visited on the Voronoi diagram. Each visited data point is compared to all spatial skyline points identified earlier for the spatial dominance test. This procedure repeats until the algorithm finds all Voronoi cells involving spatial skyline points. The experiments in [12] show that the VS^2 algorithm performs better than the B^2S^2 algorithm, but [33] demonstrates that the proposed VS^2 algorithm is not correct and accurate. The improvement algorithm identifies some spatial skyline points without a dominance test. The basic idea is that a spatial skyline point is each point of interest whose Voronoi cell intersects the interior of the convex hull of query points. Although error correction decreased the performance, it was still the best algorithm at that moment.

A faster method is represented in [34], which reduces the number of tests done. First, it reduces the query points to the set of points that constructs the smallest convex hull. Then, the algorithm selects an arbitrary query point q and sorts the set of interest points according to the increasing distance up to that point. It chooses the closest data point to q as the spatial skyline point and evaluates the remaining data points by increasing the distance to q. Furthermore, this method uses the transitivity of the dominance property to reduce the number of tests. A fast approximate algorithm is also introduced in [34] using the Voronoi diagram of points of interest and the convex hull of query points. This algorithm approximates the spatial skyline points by the points of interest whose Voronoi cell intersects the interior of the convex hull.

Notably, none of the mentioned approaches can solve the problem of the spatial skyline in parallel. It is nontrivial to extend them to a distributed and parallel system. Due to the high cost of skyline evaluation, various advanced solutions have been introduced in [2, 15–23] to analyze general skyline queries in distributed and parallel environments. None of the computation methods mentioned could address the spatial skyline problem. There is just a parallel method for calculating the spatial skyline introduced in [24] using the MapReduce technique. The basic idea is to define several independent regions into the search region. The spatial skyline points of each independent region do not depend on external data points. Therefore, the problem is divided into several subproblems and solved in parallel. This method consists of three MapReduce phases. The first phase partitions query points and constructs several local convex hulls in the map step. Then, it produces the global convex hull by merging the local convex hulls in the reduce step. The second phase determines the independent regions based on points of interest and the global convex hull. It calculates the pivots of the local independent region in the map step and then estimates the optimal pivot of the independent region in the reduce step. It is a point with equal distance from all convex points that can subdivide data points of the same size. The third phase selects an independent region for each point of interest in the map step. Then, it groups the points of interest by independent regions and sends them to reducers. Each reducer computes the spatial skyline points, and the global spatial skyline points are the union of the output of reducers. This method removes the repeated skyline points using an elimination algorithm. The reason is that some data points may be sent to two or more independent regions and produced duplicates into the result set.

As mentioned before, many properties such as convex hull and Voronoi diagram are not practical for calculating weighted spatial skylines. The only existent method that evaluates the spatial skyline queries under the weighted Euclidean distances is in [13]. Their authors introduce a sequential algorithm called the weighted distance sorting algorithm, WDS.

The WDS algorithm selects an arbitrary query point at first. Then, it sorts the points of interest according to their weighted distance and chooses the closest point of interest as the weighted spatial skyline query point. In the next step, the algorithm evaluates other points of interest in increasing distance order. It checks whether currently detected spatial skyline points dominate the data point. If the data point is not dominated, it is a point on the spatial skyline and added to the set of spatial skylines. Another algorithm is proposed in [13], which is called PWBF. This algorithm takes advantage of the capabilities of the GPU to accelerate the process and compute the weighted spatial skylines in parallel.

Our proposed solution targets weighted spatial skyline queries, which are different from the other skyline queries. None of the above computational algorithms except one could address the weighted spatial skyline problem. The only existing parallel solution for weighted spatial skyline queries cannot efficiently deal with large-scale data sets. Therefore, a novel MapReduce-based solution is proposed, including a parallel approach for computing the global weighted spatial skyline points from the local weighted spatial skyline points into multiple reducers.

3 Preliminaries

This section presents some fundamental concepts of general spatial skyline queries. Then, the weighted Euclidean distance is defined, and the properties of the spatial skyline queries under this distance are studied. Finally, the problem tackled in this paper is formalized which deals with the weighted spatial skyline queries. All notations used in this paper are summerized in Table 1.

3.1 Skyline query

Let P be a data set in a d-dimensional space R^d. We represent a point $p \in P$ as $p = \{x_1, x_2, \ldots, x_d\}$ where $p.x_i$ is the value of the point on the ith dimension. Without loss of generality in this paper, assume that smaller values are preferable to larger ones in all dimensions. Given two points $p, p' \in P$, p dominates p' ($p \prec p'$) if and only if for every dimension $i \in \{1, \ldots, d\}$, $p.x_i \leq p'.x_i$ and there is at least one dimension j such that $p.x_j < p'.x_j$, i.e.

$$\forall i \in \{1, \ldots, d\}, p.x_i \leq p'.x_i \wedge \exists j \in \{1, \ldots, d\}, p.x_j < p'.x_j \tag{1}$$

A point $p \in P$ is a skyline point if and only if no other point $p' \in P$ dominates p in the data set. A skyline query retrieves the data points of P, called skyline points ($S_{skyline}$), over which no other data point dominates, i.e.

$$S_{skyline}(P) = \{\, p \in P | \nexists\, p' \in P - \{p\}, p' \prec p\} \tag{2}$$

The skyline query returns the points in the data set that are better than other points in the data set. Based on the situation, "better" means lower or higher coordinate values.

Table 1 Overview of the symbols used in the paper

Notation	Description
P, Q	A set of data points and a set of query points
p, q	A data point and a query point
$p.x_i$	The value of the data point p in the ith dimension
R^d	A d-dimensional space
$p \prec p'$	The data point p dominates the data point p'
$S_{skyline}(P)$	The skyline points of data set P
$d_E(p, q)$	The Euclidean distance between points p and q
$p \prec_Q p'$	The data point p spatially dominates the data point p' with respect to Q
$S_{spatialskyline}(P, Q)$	The spatial skyline points of P with respect to Q
$PB(p, p')$	The perpendicular bisector of two points
$CR(p, p')$	The closest region to p with respect to p$'$
$CH\ (Q)$	The convex hull of Q
$CP(Q)$	The convex points of the convex hull
$VD\ (P)$	The voronoi diagram of P
$IR\ (p, q)$	The independent region generated by p and q
$DR(p, Q)$	The dominator region of p with respect to Q
$w(p)$	A positive real number as the weight of p
$d_{WE}(p, q)$	The weighted Euclidean distance between points p and q
$B(p, p')$	The bisector of p and p' with weighted Euclidean distance
$WR(p, p')$	The weight ratio of two points
$p \prec_Q^W p'$	The data point p spatially dominates the data point p' with respect to Q in weighted Euclidean distance
$S_{WSS}(P, Q)$	The weighted spatial skyline points of P with respect to Q
$d\_test(p, p', Q)$	A function to evaluate whether p spatially dominates p' with respect to Q
$DS_Q(P1, P2)$	The dominated subset operation, the subset of data points in $P2$ which are dominated by at least one point in $P1$
$local\_S_{WSS}(P, Q)$	$\bigcup_{i=1}^{m} S_{WSS}(Pi, Q)$
$non\mathrm{S}_{WSS}\ (Pi, Q)$	The set of weighted spatial non-skyline points of the set Pi with respect to Q

3.2 Spatial skyline query

Assume that P is a set of n points of interest and Q is a set of m query points in a d-dimensional space. Assume $p.x_i$ is the value of the point in the ith dimension. The distance function $d_E(p, q)$ computes the Euclidean distance between points p and q, and Eq. (3) represents it:

$$d_E(p,q) = \sqrt{\sum_{i=1}^{d}(p.x_i - q.x_i)^2} \tag{3}$$

Given two points $p, p' \in P$, p spatially dominates p' with respect to Q, denoted $p \prec_Q p'$, if and only if $d_E(p,q) \le d_E(p',q)$ for every $q \in Q$ and there is at least one query point $q' \in Q$ such that $d_E(p,q') < d_E(p',q')$, i.e.

$$p \prec_Q p' \Leftrightarrow \forall q \in Q, d_E(p,q) \le d_E(p',q) \wedge \exists q' \in Q, d_E(p,q') < d_E(p',q') \tag{4}$$

Let $S_{\text{spatialskyline}}$ be the set of spatial skyline points of P with respect to Q in the Euclidean plane. The point $p \in P$ is a spatial skyline point with respect to Q ($p \in S_{spatialskyline}$) if and only if there is no other point in P that spatially dominates it, i.e.

$$S_{spatialskyline}(P,Q) = \{\, p \in P | \nexists\, p' \in P - \{p\}, p' \prec_Q p\} \tag{5}$$

To obtain spatial skyline points, we require investigating each point in P whether it is spatially dominated by any other point in P with respect to Q. However, several geometric features of the spatial skyline can reduce the region where the spatial skyline point can be found. Some algorithms use these features to decrease dominance tests. We summarize the most important definitions and properties of the spatial skyline points in the following.

- *The perpendicular bisector of two points* $PB(p,p')$ is the locus of the points of equal distance of p and p' in space. It is a straight line perpendicular to the line passing through the midpoint of the points p and p'.
- *Closest region to p with respect to* p' $CR(p,p')$ is the locus of points closer to p than to p' in Euclidean space. The perpendicular bisector of two points p and p' is $PB(p,p')$ and the half-plane including p.
- *The convex hull of the query points CH* (Q) is the smallest convex polytope that contains all points in Q. Its vertices are called convex points and define the set CP (Q).
- *The voronoi diagram of the points of interest VD* (P) divides the space into $|P|$ regions. Each region contains a point of $p \in P$ and the points of the plane that are closer to p than any other point in P. Each partition is called the Voronoi cell.
- *Independent Region IR* (p, q) is a sphere centered at q with radius d (p, q). The union of the independent regions is *IRG* (*Independent Region Group*) of p with respect to Q [24].
- *Dominator Region DR*(p, Q) is the region where p is spatially dominated by any point within it [24]. Defined as Eq. (6):

$$\forall p \in P, \quad DR(p,Q) = \bigcap_{q_i \in Q} IR(p, q_i) \tag{6}$$

We can use these definitions to limit the search region to find the spatial skyline points. Based on these definitions, some properties are extracted in [12, 13, 24, 33–35], as shown in the following.

Property 1 If $p \in P$ is a spatial skyline point for $Q' \subset Q$, then p is also a spatial skyline point for Q [12].

Property 2 The set of spatial skyline points of P with respect to Q depends only on the convex points which define the convex hull of Q [12], i.e.

$$S_{spatialskyline}(P,Q) = S_{spatialskyline}(P, CP(Q)) \tag{7}$$

Property 3 Each point of P inside the convex hull of Q is a spatial skyline point of P with respect to Q [12].

Property 4 Each point of P whose Voronoi cell intersects the inside of the convex hull of Q is a spatial skyline point [33, 35].

Property 5 Let p, p', and p'' be three data points of P. If p spatially dominates p' with respect to Q and p' spatially dominates p'' with respect to Q, then p spatially dominates p'' with respect to Q [34].

Property 6 Let $p, p' \in \mathrm{P}$ and $PB(p,p')$ be the perpendicular bisector of two points in the plane, and CH (Q) is the convex hull of Q. Two points do not spatially dominate each other if and only if $PB(p,p')$ intersects the interior of $CH(Q)$ [34].

Property 7 Assume that p is the data point of P and DR(p, Q) is the dominator region of p with respect to Q. The p is a spatial skyline if and only if it is the only point of P in its dominator region [24]. Equation (8) represents it:

$$\forall p \in P, p \in S_{spatialskyline}(P,Q) \Leftrightarrow P \cap DR(p,Q) = \{p\} \tag{8}$$

Property 8 Given $p \in \mathrm{P}$ and $q \in Q$, each point $p' \in P \cap DR(p,q)$ can only be dominated by a data point of $P \cap DR(p,q)$ [24].

3.3 Weighted Euclidean distance

Let P be a set of points of interest and Q be a set of query points in the space. The $d_E(p, q)$ denotes the Euclidean distance between points $p \in P$, and $q \in Q$. Assume that each point $p \in P$ is assigned a positive real number as the weight $w(p) > 0$ to consider its importance. The weighted Euclidean distance between two points $p \in P$ and $q \in Q$ is defined as:

$$d_{WE}(p,q) = \frac{1}{w(p)} d_E(p,q) \tag{9}$$

The weighted Euclidean distance is a non-metric parameter. Because the weights are point-dependent, triangle inequality is not maintained.

For instance, Fig. 1 shows a set of eight points $P = \{p1, p2, \ldots, p8\}$ and a query point $Q = \{q\}$. The Euclidean distance is considered for each point of P with respect to the query point q in Fig. 1a. Based on the point weight, the weighted Euclidean distance is computed for each point of P concerning q in Fig. 1b, while the weight of each point is shown in parentheses. It is notable that some points place near point q in the Euclidean distance (e.g., *p1*, *p2*, and *p3* in Fig. 1a), containing a greater weighted Euclidean distance (in Fig. 1b). Since these points weigh less than other points, their weighted Euclidean distance is more. On the other hand, the more weight the points are than the other points (e.g., *p5*, *p6*, *p7*, and *p8*), the less their weighted Euclidean distance.

3.4 Bisector between two points with weighted Euclidean distance

Assume that P is a set of points, and each point $p \in P$ is also given a weight $w(p) > 0$. For each pair $p, p' \in$ P, the set of all points that have equal weighted Euclidean distance between two points p and p', with weights $w(p)$ and $w(p')$ is called the bisector of p and p' and is denoted $B(p,p')$ where

$$\begin{aligned} B(p,p') &= \{\, x | d_{WE}(p,x) = d_{WE}(p',x)\} \\ &= \{x | d_E(p,x)/d_E(p',x) = w(p)/w(p')\} \end{aligned} \tag{10}$$

Bisectors generate two different types. If $w(p) = w(p')$ then, $B(p, p')$ is the hyperplane being orthogonal to the line between p and p' and bisecting it. In the special case ($x \in R^2$ in Eq. 10), the hyperplane is a straight line. If $w(p) \neq w(p')$ then $B(p, p')$ is a sphere that has the following center and radius:

$$c = \frac{w(p)^2 p' - w(p')^2 p}{w(p)^2 - w(p')^2} \text{ and } r = \frac{w(p) w(p') d_E(p,p')}{\left|w(p)^2 - w(p')^2\right|} \tag{11}$$

In the special case ($x \in R^2$ in Eq. 10), the sphere is a circle. If $w(p) < w(p')$, $B(p, p')$ is a circle containing p in its interior [36]. Let WR be the weight ratio of two points, $WR(p, p') = w(p)/w(p')$, Fig. 2 represents three examples of bisector $B(p, p')$ for fixed points p and p' corresponding to $WR(p, p') = 1, 2, 1/3$ values. The bisector becomes a straight line in the particular situation $WR(p, p') = 1$ ($w(p) = w(p')$). The bisector is a circle surrounding point p' in the special case of $WR(p, p') = 2$, and it is a circle surrounding point p in the case of $WR(p, p') = 1/3$. Notably, the circle radius decreases as WR increases, and the center becomes closer to the surrounded point.

The definition of the closest region to p with respect to p', $CR(p,p')$ should be re-evaluated according to the denotation of $B(p,p')$. The closest region to the point p may be the interior or the exterior of the circle specified by $B(p,p')$. For instance, in the case of $WR(p, p') = 2$ in Fig. 2, the closest region to the point p with respect to p' is the outer region of the circle.

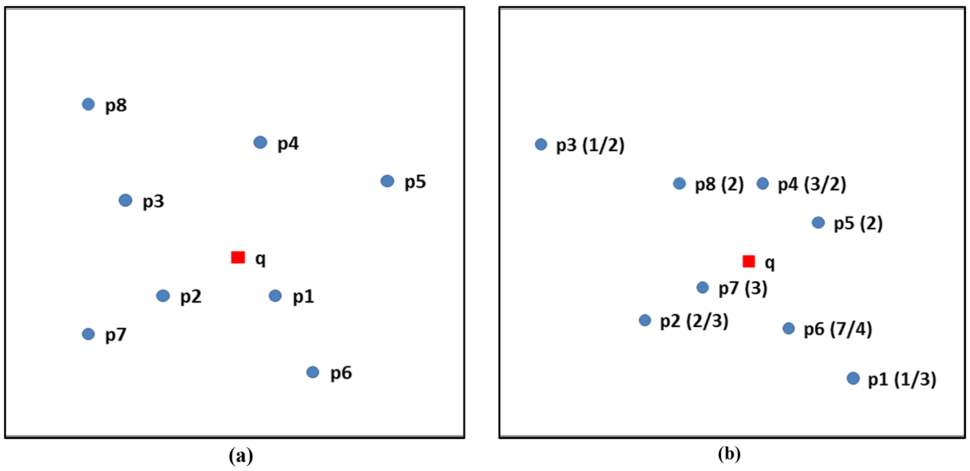


Fig. 1 An example of a set of points of interest and a query point: **a** Euclidean distances **b** weighted Euclidean distances

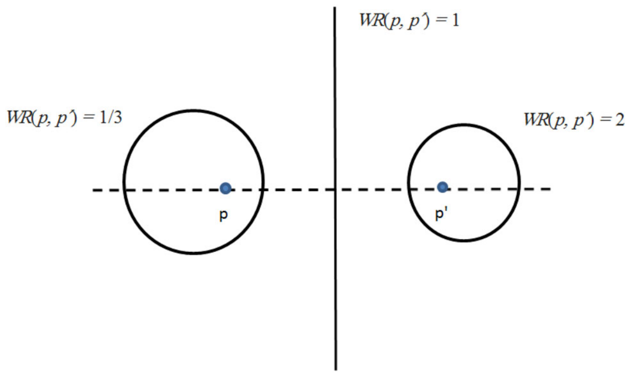


Fig. 2 The bisectors of p and p' corresponding to three ratios $WR(p, p')$

3.5 Weighted spatial skyline queries

Let P be a set of n weighted points and Q be a set of m query points in a d-dimensional space. Assume that p and p' are two weighted points of P, p spatially dominates p' with respect to Q in weighted Euclidean distance, denoted $p \prec_Q^W p'$, if and only if $d_{WE}(p,q) \le d_{WE}(p',q)$ for every $q \in Q$ and there is at least one query point $q' \in Q$ such that $d_{WE}(p,q') < d_{WE}(p',q')$, i.e.

$$p \prec_Q^W p' \Leftrightarrow \forall q \in Q, d_{WE}(p,q) \le d_{WE}(p',q) \wedge \exists q' \in Q,\ d_{WE}(p,q') < d_{WE}(p',q') \tag{12}$$

Assume that S_{WSS} is the set of weighted spatial skyline points of P with respect to Q in weighted Euclidean distance. The weighted point $p \in P$ is a weighted spatial skyline point with respect to Q ($p \in S_{WSS}$) if and only if there is no other weighted point in P that spatially dominates it, i.e.

$$S_{WSS}(P,Q) = \{\, p \in P | \nexists\, p' \in P - \{p\}, p' \prec_Q^W p\} \tag{13}$$

In other words, $S_{WSS}(P,Q)$ contains the weighted points of the set P with respect to Q:

$$S_{WSS}(P,Q) = \{\, p \in P | \forall p' \in P - \{p\}, \exists q \in Q, q \in CR(p,p')\} \tag{14}$$

We obtain different results in the spatial skyline query with the weights for the points of interest.

Some of the well-known properties of the spatial skyline points presented in Sect. 3.2 still hold in the weighted spatial skyline points, but most of them turn to be not true. Several statements related to the geometric properties of the weighted spatial skyline queries have been provided to demonstrate the difference between the weighted and the unweighted spatial skyline queries.

1. Assume that $Q' \subset Q$ and $p \in P$ for Q' is a weighted spatial skyline point. It may happen that p for Q is not a weighted spatial skyline point. For example, in Fig. 3, $Q' = \{q_2, q_3, q_4\}$ and p is a weighted spatial skyline point. The point p is not a weighted spatial skyline point for Q because p' spatially dominates it with respect to q_1.
2. Let CP (Q) denote the convex points of Q, the points defining the convex hull of Q (see Sect. 3.2 for further details). The set of spatial skyline points of P may not depend only on CP (Q) in weighted Euclidean distance. For example, in Fig. 3, the point p spatially dominates p$'$ with respect to CP (Q) = $\{q_2, q_3, q_4\}$. The point p is not a weighted spatial skyline point with respect to Q because the point p' spatially dominates it with respect to $Q = \{q_1, q_2, q_3, q_4\}$.
3. A point $p \in P$ inside the convex hull of Q (CH (Q)) may not be a spatial skyline point of P concerning Q in weighted Euclidean distance. For instance, in Fig. 4, the point p' inside CH (Q) is not a weighted spatial skyline point because point p dominates it.
4. Two points $p, p' \in P$ may dominate each other, while the bisector between two points ($B(p,p')$) intersects the interior of the convex hull of Q (CH (Q)). For example, in Fig. 3, even though $B(p,p')$ intersects with the interior of CH (Q), p' dominates p. This statement can be seen again in Fig. 4. The bisector between p and p' intersects with the interior of the convex hull of Q, but the point p dominates the point p'.
5. Each point $p \in P$ whose Voronoi cell overlaps the inside of the convex hull of Q may not be a weighted spatial skyline point. For example, in Fig. 4, the Voronoi cell of p' overlaps the inside of CH (Q), but it is not a weighted spatial skyline point.

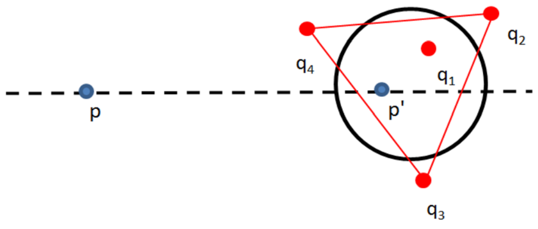


Fig. 3 An example of two weighted point $P = \{p, p'\}$ with $WR(p, p') = 2$ and $Q = \{q_1, q_2, q_3, q_4\}$

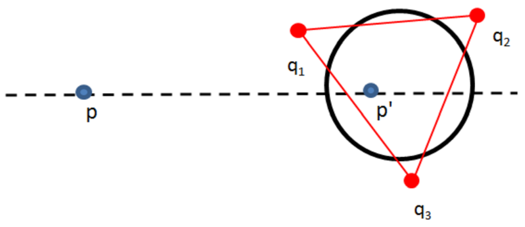


Fig. 4 An example of two weighted point $P = \{p, p'\}$ with $WR(p, p') = 2$ and $Q = \{q_1, q_2, q_3\}$

6. Assume that the independent region $p \in P$ and $q \in Q$ is a sphere centered at q with radius $d_{WE}(p, q)$. Dominator Region of p with respect to Q may not be created because the independent regions of p with respect to Q do not overlap. For instance, the independent regions of point p with $w(p) = 2$ with respect to $Q = \{q_1, q_2, q_3\}$ are shown in Fig. 5. *DR* (p, Q) is the intersection of the regions of point p with respect to Q according to Eq. (6). It is empty in Fig. 5.

As it can be seen, many of the properties used to solve spatial skyline queries do not match the weighted spatial skyline queries. The convex hull or the Voronoi diagram does not reduce the number of tests performed. It is also not possible to use the Dominator Region to accelerate the discovery of the weighted spatial skylines. The weight cannot be selected randomly, and its value significantly affects the results. The weight causes $|Q|$ virtual points to be extracted from each point $p \in P$. Each point p is virtually placed in a different place for each query point $q \in Q$. Figure 6 shows a transformation of the points of P according to each query point. Therefore, many of the techniques used in the unweighted case do not extend to the weighted one.

4 Parallel processing of weighted spatial skyline queries

This section introduces a new MapReduce-based solution for processing weighted spatial skyline queries (MR-WSS). As described in the previous sections, many properties of spatial skylines cannot be used to reduce the number of tests done on weighted spatial skylines. Hence, this section proposes a parallel computing approach to accelerate the process.

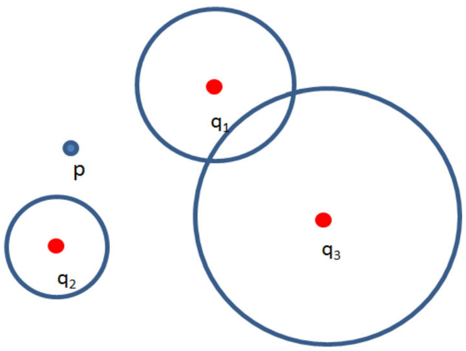


Fig. 5 The independent regions of p with $w(p) = 2$ and $Q = \{q_1, q_2, q_3\}$ in 2-dimensional space

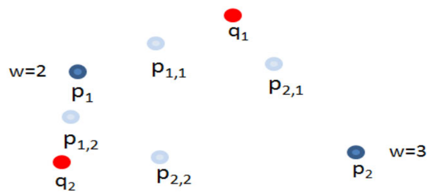


Fig. 6 Virtual transformation of $P = \{p_1, p_2\}$ according to $Q = \{q_1, q_2\}$

4.1 Theoretical foundation

Consider a set of weighted data points P and a set of query points Q. The first property for the weighted spatial skyline set is as follows:

Lemma 1 *Assume that $p \in P$ is a unique point and the point closest to $q \in Q$. The point p is a weighted spatial skyline point.*

Proof We have $p \in P$ is a unique point and the closest point to $q \in Q$, which means:

$$\forall p' \in P, d_{WE}(p, q) < d_{WE}(p', q) \tag{15}$$

Therefore, there is not a weighted point in P that spatially dominates p with respect to Q. Equation (13) implies that point p is a weighted spatial skyline point. □

In particular, if the point p is not unique. There is more than one weighted point with the same minimum distance to the point q. At least one of them is a weighted spatial skyline point, and others are spatially dominated. Therefore, we assume that the closest weighted point to each query point is unique in the remainder of this paper, without loss of generality.

Lemma 1 demonstrates that the presence of some points in the set of weighted spatial skyline points is independent of the position of any other data points. The following Lemma shows the transitivity rule in weighted spatial skyline points.

Lemma 2 *For three weighted points $p, p', p'' \in P$, if p spatially dominates p' concerning Q and p' spatially dominates p'' with respect to Q, then p spatially dominates p'' with respect to Q.*

Proof If p spatially dominates p' with respect to Q i.e.

$$\forall q \in Q,\ d_{WE}(p,\ q) \le d_{WE}(p',\ q) \wedge \exists q' \in Q,\ d_{WE}(p, q') < d_{WE}(p',\ q') \tag{16}$$

If p' spatially dominates p'' with respect to Q i.e.

$$\forall q \in Q,\ d_{WE}(p',\ q) \le d_{WE}(p'',\ q) \wedge \exists q'' \in Q,\ d_{WE}(p',\ q'') < d_{WE}(p'',\ q'') \tag{17}$$

Consider two cases (i) $q' = q''$ (ii) $q' \neq q''$.

(i) $q' = q''$: According to Eqs. (16) and (17):

$$\forall q \in Q,\ d_{WE}(p,\ q) \le d_{WE}(p',\ q) \le d_{WE}(p'',\ q) \textit{and}$$

$$\exists\, q' = q'' \in Q,\ d_{WE}(p,\ q') < d_{WE}(p',\ q') < d_{WE}(p'',\ q')$$

Therefore, p spatially dominates p'' with respect to Q ($p \prec_Q^W p''$).

(ii) $q' \ne q''$: According to Eq. (16), there is a query point $q' \in Q$ that $d_{WE}(p, q') < d_{WE}(p', q')$ and according to Eq. (17), for all query points $q' \in Q$ except $q'' \in Q$, $d_{WE}(p',\ q') \le d_{WE}(p'',\ q')$. Therefore,

$$d_{WE}(p, q') < d_{WE}(p'', q') \tag{18}$$

On the other hand, for all query points except $q' \in Q$, we have $d_{WE}(p,\ q) \le d_{WE}(p',\ q)$ or $d_{WE}(p, q'') \le d_{WE}(p',\ q'')$ according to Eq. (16) and $d_{WE}(p',\ q'') < d_{WE}(p'',\ q'')$ according to Eq. (17). It is clear that:

$$d_{WE}(p,\ q'') < d_{WE}(p'',\ q'') \tag{19}$$

Therefore, by Eqs. (18) and (19), p spatially dominates p'' with respect to Q ($p \prec_Q^W p''$). □

The typical method of calculating weighted spatial skyline is to check the spatial dominance between each pair of data points. We want to exploit the distributed environment to accelerate the computation process. The basic idea is to partition the data points first. Then, it assigns that data to different nodes to compute the weighted spatial skyline points locally. It is notable that according to the role of the query points set in the problem, its cardinality has to be much smaller than the cardinality of the data points set. Hence, each partition receives all query points. Finally, a central site collects the local weighted spatial skyline points to compute the global weighted spatial skyline points.

To solve the problem in a distributed environment, the problem must be decomposable. Assume that op_1 is a query operator on the data points set P and divide P into n subsets. The op_1 is a decomposable operator since it can be computed by another operator (for example, op_2) as given in Eq. (21):

$$P = \bigcup_{i=1}^{n} P_i \tag{20}$$

$$op_1\left(\bigcup_{i=1}^{n} P_i\right) = op_2\left(\bigcup_{i=1}^{n} op_1(P_i)\right) \tag{21}$$

Unfortunately, weighted spatial skyline computation is not a decomposable problem. According to Eq. (13), the weighted point $p \in P$ is a weighted spatial skyline point with respect to Q if and only if there is no other weighted point in P that spatially dominates it. Therefore, we have to check each point with the others, while some are in other partitions.

Lemma 3 *The weighted spatial skyline query is not decomposable.*

Lemma 3 shows that a set of data points cannot be divided into multiple partitions and, then the results of each partition are combined as the final results of a weighted spatial skyline query. Therefore, we need more operation to obtain the final result.

The proposed method computes the set of weighted spatial skyline points in one phase of MapReduce. It obtains the local weighted spatial skyline points in mappers and then distributes the tasks of computing the global weighted spatial skyline points into reducers.

4.1.1 Distributed dominance tests

Let p and p' be two weighted data points in P and $d\_test(p, p', Q)$ denote a function to evaluate whether p spatially dominates p' with respect to Q. If $p \prec_Q^W p'$, $d\_test(p, p', Q)$ returns 'True' else returns 'False'. A dominance test is to perform this function once. The intuitive method to find the weighted spatial skyline points of a set of data points P with respect to Q is to perform a dominance test on each data point pair (p, p') with $p \ne p'$. Let S_{WSS} be the set of weighted spatial skyline points of P with respect to Q and first $S_{WSS} = P$. The data point p' is removed from S_{WSS} if $d\_test(p, p', Q)$ is true for every other data point p in S_{WSS}.

We introduce a dominated subset operation on two sets of data points to design a distributed method to perform the dominance tests on the local weighted spatial skyline points. Let *P1* and *P2* denote two nonempty subsets of P. A dominated subset operation on *P1* and *P2* with respect to Q, denoted $DS_Q(P1, P2)$, is as follows:

$$DS_Q(P1, P2) = \{\, p_j | p_j \in P2 \wedge \exists p_i \in P1, p_i \prec_Q^W p_j \} \tag{22}$$

where *P1* as dominator set and *P2* as dominatee set are known.

The result of $DS_Q(P1, P2)$ is a spatially dominated subset of *P2* with regard to *P1*. The *P2*—$DS_Q(P1, P2)$ is the undominated subset of *P2* with regard to *P1*. $DS_Q(P, P)$ represents the data points in P spatially dominated by at least one point in P. In other words, $DS_Q(P, P)$ consists of the data points which are not in $S_{WSS}(P, Q)$.Therefore,

$$S_{WSS}(P, Q) = P - DS_Q(P, P) \tag{23}$$

Assume that *P1*, *P2*, …, *Pm* are m partitions of a set of data points P ($P = \bigcup_{i=1}^{m} Pi$). The main objective is to find

$S_{WSS}(P, Q)$ from the $S_{WSS}(Pi, Q)$ for i = 1, …, m by performing the dominated subset operations among $S_{WSS}(Pi, Q)$ for i = 1, …, m.

Lemma 4 *Let P1, P2, …, Pm are m partitions of a set of data points P* ($P = \bigcup_{i=1}^{m} Pi$). *Let.*

$$local\_S_{WSS}(P,Q) = \bigcup_{i=1}^{m} S_{WSS}(Pi,Q) \tag{24}$$

For each partition Pi,

$$DS_Q(P,Pi) = DS_Q(local\_S_{WSS}(P,Q),Pi) \tag{25}$$

Proof For each data point p belongs to $DS_Q(local\_S_{WSS}(P,Q),Pi)$, it also belongs to Pi ($p \in Pi$) and it is dominated by a point in $local\_S_{WSS}(P,Q)$. Because the $local\_S_{WSS}(P,Q)$ is a subset of P, p is dominated by a point in P, i.e., $p \in DS_Q(P,Pi)$.

For each data point p in $DS_Q(P, Pi)$, there is a data point in P (e.g., $p' \in P$) that dominates p. Let Pj be the partition containing p'. Consider two cases:

(i) p' is in $S_{WSS}(Pj, Q)$: p is dominated by a data point in $local\_S_{WSS}(P,Q)$ because, according to Eq. (24), $S_{WSS}(Pj, Q)$ is a subset of $local\_S_{WSS}(P,Q)$.

(ii) p' is not in $S_{WSS}(Pj, Q)$: there is a data point p'' in $S_{WSS}(Pj, Q)$ which dominates p' and p (Lemma 2.). Therefore, the data point p is dominated by a point in $S_{WSS}(Pj, Q)$, a subset of $local\_S_{WSS}(P,Q)$.

In both cases, the data point p is in $DS_Q(local\_S_{WSS}(P,Q),Pi)$. □

Lemma 5 *Let P1, P2, …, Pm are m partitions of a set of data points P* ($P = \bigcup_{i=1}^{m} Pi$). *Then,*

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m} \Big(S_{WSS}(Pi,Q) \bigcup_{\substack{j=1..m \\ j \neq i}} DS_Q(S_{WSS}(Pj,Q), S_{WSS}(Pi,Q))\Big) \tag{26}$$

Proof According to Eq. (23): $S_{WSS}(P,Q) = P - DS_Q(P,P)$

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m} Pi - DS_Q\left(P, \bigcup_{i=1}^{m} Pi\right) \tag{27}$$

$$= \bigcup_{i=1}^{m} Pi - \bigcup_{i=1}^{m} DS_Q(P,Pi) \tag{28}$$

$$= \bigcup_{i=1}^{m} (Pi - DS_Q(P,Pi)) \tag{29}$$

□

from Lemma 4:

$$= \bigcup_{i=1}^{m} (Pi - DS_Q(local\_S_{WSS}(P,Q),Pi)) \tag{30}$$

Each Pi consists of the points of the skyline and the points of non-skyline. The points dominated by the points in the skyline are the points of non-skyline. According to Eq. (13), Let S_{WSS} (Pi, Q) be the set of weighted spatial skyline points of the set Pi with respect to Q, and then nonS$_{WSS}$ (Pi, Q) be the set of weighted spatial non-skyline points of the set Pi with respect to Q. Therefore, $S_{WSS}(P, Q)$ is:

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m} ((S_{WSS}(Pi,Q) \cup nonS_{WSS}(Pi,Q))) - DS_Q(local\_S_{WSS}(P,Q), (S_{WSS}(Pi,Q) \cup nonS_{WSS}(Pi,Q))) \tag{31}$$

For any universe U and subsets A, B and C of U, the following identity holds:

$$(A \cup B) - C = (A - C) \cup (B - C) \tag{32}$$

By Eqs. (31) and (32):

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m} ((S_{WSS}(Pi,Q) - DS_Q(local\_S_{WSS}(P,Q), (S_{WSS}(Pi,Q) \cup nonS_{WSS}(Pi,Q)))) \cup (nonS_{WSS}(Pi,Q) - DS_Q(local\_S_{WSS}(P,Q), (S_{WSS}(Pi,Q) \cup nonS_{WSS}(Pi,Q))))) \tag{33}$$

We could consider:

$$DS_Q(local\_S_{WSS}(P,Q), (S_{WSS}(Pi,Q) \cup nonS_{WSS}(Pi,Q))) = DS_Q(local\_S_{WSS}(P,Q), S_{WSS}(Pi,Q)) \cup DS_Q(local\_S_{WSS}(P,Q), nonS_{WSS}(Pi,Q)) \tag{34}$$

By Eq. (24), $S_{WSS}(Pi,Q) \subset local\_S_{WSS}(P,Q) then:$

$$DS_Q(local\_S_{WSS}(P,Q), nonS_{WSS}(Pi,Q)) = nonS_{WSS}(Pi,Q) \tag{35}$$

We substitute Eqs. (35) into (33):

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m} ((S_{WSS}(Pi,Q) - (DS_Q(local\_S_{WSS}(P,Q),S_{WSS}(Pi,Q)) \cup nonS_{WSS}(Pi,Q))) \cup (nonS_{WSS}(Pi,Q) - (DS_Q(local\_S_{WSS}(P,Q),S_{WSS}(Pi,Q)) \cup nonS_{WSS}(Pi,Q)))) \tag{36}$$

Finally,

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m}(S_{WSS}(Pi,Q) - DS_Q(local\_S_{WSS}(P,Q), S_{WSS}(Pi,Q))) \tag{37}$$

The S_{WSS} (Pi, Q) from $local\_$ S_{WSS} (P, Q) must be removed to solve the problem properly. Therefore,

$$S_{WSS}(P,Q) = \bigcup_{i=1}^{m}(S_{WSS}(Pi,Q) - DS_Q(local\_S_{WSS}(P,Q) - S_{WSS}(Pi,Q), S_{WSS}(Pi,Q))) \tag{38}$$

$$= \bigcup_{i=1}^{m}(S_{WSS}(Pi,Q) - \bigcup_{\substack{j=1..m\\ j\neq i}} DS_Q(S_{WSS}(Pj,Q), S_{WSS}(Pi,Q))) \tag{39}$$

We design a distributed algorithm to compute the weighted spatial skyline points according to Lemma 5.

4.2 The proposed strategy

Our approach MR-WSS consists of one MapReduce phase, which receives a set of weighted data points P and query points Q and outputs a set of weighted spatial skyline points of P with respect to Q.

Initially, all data points are divided into m splits, and each split is assigned to a mapper. The cardinality of the set of query points is much less than that of the set of data points, so each mapper receives the set of query points. Each mapper ($mapper_i$) computes the set of weighted spatial skyline points of each split Pi with respect to Q (S_{WSS} (Pi, Q)) and then sends the result to the reducers.

Each reducer ($reducer_i$) uses the set of weighted spatial skyline points of split Pi (S_{WSS}(Pi, Q)) as the dominatee and the set of weighted spatial skyline points of split Pj (S_{WSS}(Pj, Q)) as the dominator (for $j = 1, 2, \ldots, m, j \neq i$). Then, it removes the data points in S_{WSS} (Pi, Q) dominated by at least a data point in S_{WSS} (Pj, Q).

Finally, the results of all the reducers create the final set of weighted spatial skyline points (S_{WSS} (P, Q)). Figure 7 demonstrates the framework of the MR-WSS method.

Each mapper should compute the set of weighted spatial skyline points for a split of data points concerning a set of query points in the proposed method. The main idea is to perform a dominance test on each pair of data points. An adaptation of the traditional methods of computing the local set of weighted spatial skyline points in the mappers is introduced.

An arbitrary point of query points is initially selected (e.g., $q \in Q$), and then the data points of P are sorted according to their weighted distance to the query point q. The closest data point of P to the query point q is chosen (e.g., $p' \in P$) as the first data point of P in the weighted spatial skyline set (due to Lemma 1) and stored in the weighted spatial skyline set (S_{WSS} (P, Q)). Then, each data point $p \in P$ is analyzed in increasing distance order by checking whether it is dominated by any of the weighted spatial skyline points currently detected. If not dominated, it is a weighted spatial skyline point and is added to the weighted spatial skyline set (S_{WSS} (P, Q)). According to lemma 2, each non-skyline point will be dominated by one of the skyline points already detected. Therefore, the final set contains all points of the weighted spatial skyline. Algorithm 1 describes the details of the proposed method.

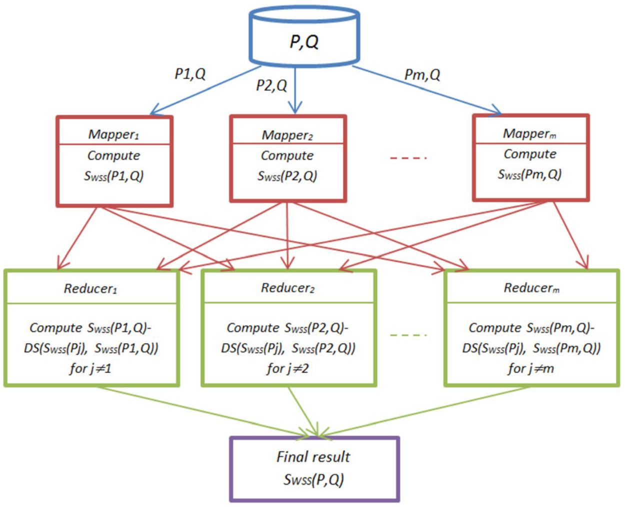


Fig. 7 The framework of the proposed method MR-WSS

A dominance test determines whether the point $p \in P$ is dominated by a point of the currently detected weighted spatial skyline set ($p' \in S_{WSS}$). If a point $q \in Q$ is closer to point p' than to point p, then point p' dominates point p. The algorithm traverses query points Q to compute and compare the weighted distances d_{WE} (p',q) and d_{WE} (p,q). Once a point $q \in Q$ is found with d_{WE} (p',q) < d_{WE} (p,q), it indicates that p' dominates p and the dominance test returns the 'True' value. If for all $q \in Q$ the inequality d_{WE} (p',q) $\geq$ d_{WE} (p,q) holds, p' does not dominate p. If this process is repeated for all $p' \in S_{WSS}$ and none dominates p, then point p is a weighted spatial skyline point. Algorithm 2 shows the dominance test pseudocode.

Algorithm 1 Local_Skyline

```
Input: Pi = {p_1, p_2, ..., p_n}, a set of weighted data points and
       Q = {q_1, q_2, ..., q_m}, a set of query points in d-dimensional
       space.
Output: S_WSS, a set of weighted spatial skyline points from Pi with
        respect to Q.
1: Set $S_{WSS} = \emptyset$;
2: q ← Select an arbitrary query point from Q.
3: Set W={w(p_1), w(p_2), ..., w(p_n)}, weights of data points Pi;
4: for each $p \in P$ do
5:     $d_E(p,q) \leftarrow \sqrt{\sum_{i=1}^{d}(p.x_i - q.x_i)^2}$
6:     $d_{WE}(p,q) \leftarrow \frac{1}{w(p)} d_E(p,q)$
7: end for
8: Sort the data points of Pi in the ascending order according to
   their $d_{WE}(p,q)$;
9: p´ ← Select the closest data point to q;
10: $S_{WSS} \leftarrow S_{WSS} \cup \{p´\}$;
11: for each $p \in P$ do
12:    If Dominance_Test (p, $S_{WSS}$, Q)=False then
13:       $S_{WSS} \leftarrow S_{WSS} \cup \{p\}$;
14:    end if
15: return $S_{WSS}$
```

Algorithm 2 Dominance_Test

```
Input: p, a data point of P and
       S_WSS, a set of weighted spatial skyline points currently
       detected and
       Q, a set of query points.
Output: a boolean variable to inform whether the data point is
        dominated.
1: for each $p´ \in S_{WSS}$ do
2:    for each $q \in Q$ do
3:       Compute $d_{WE}(p´,q)$;
4:       Compute $d_{WE}(p,q)$;
5:       If $d_{WE}(p´,q) < d_{WE}(p,q)$ then
6:          return True
7:       end if
8:    end for
9: end for
10: return False
```

Algorithm 3 Map of MR-WSS

```
Input: Pi, a split of weighted data points P and
       Q, a set of query points.
Output: $LS_{WSS}$-Pi, local weighted spatial skyline points of split Pi.
1: $LS_{WSS}$-Pi ← Local_Skyline (Pi,Q);
2: for j=1 to m do
3:    return (j, (i, $LS_{WSS}$-Pi))
4: end for
```

Algorithm 4 Reduce of MR-WSS

```
Input: $LS_{WSS}$-Pi, local weighted spatial skyline points of split Pi and
       $LS_{WSS}$-Pj, local weighted spatial skyline points of other splits
       and
       Q, a set of query points.
Output: $GS_{WSS}$-Pi, global weighted spatial skyline points found in
        Pi.
1: $GS_{WSS}$-Pi ← $LS_{WSS}$-Pi
2: for each $p \in LS_{WSS}$-Pi do
3:    for each $p´ \in LS_{WSS}$-Pj and $j \neq i$ do
4:       If Dominance_Test (p, p´, Q) then
5:          $GS_{WSS}$-Pi ← $GS_{WSS}$-Pi − {p}
6:       end if
7:    end for
8: end for
9: return $GS_{WSS}$-Pi
```

The proposed method MR-WSS first finds the local skyline points by mappers. Each mapper uses Algorithms 1 and 2 to compute the local skyline points from the assigned split (see the pseudocode in Algorithm 3). The global skyline computation from the local skyline points is done by the dominated subset operations distributed to multiple reducers for parallel processing. Algorithm 4 shows this process.

Complexity analysis Assume that $n = |P|$, $m = |Q|$ and all data points are divided into k splits, and each mapper receives a split. First, the memory and then the execution time requirements of the MR-WSS are analyzed. Each mapper gets the same share of data points and whole query points and computes local weighted spatial skyline points (LS_{WSS}). Suppose that $l = |LS_{WSS}|$ and in the best case $l = m$ and the worst case $l = n/k$. Therefore, the memory requirements are $O(n/k + m + l) = O(n/k + m)$ for each mapper. Each reducer receives local weighted spatial skyline points from the mappers and computes the fraction of global weighted spatial skyline points. The number of global weighted spatial skyline points is much less than the number of data points. Hence, the total memory requirement is $O(n + m)$. In terms of time complexity, the algorithm requires $O(n/k\log(n/k))$ to sort the data points according to the distance to an arbitrary query point q in each mapper. It performs at most $O(nl/k)$ dominance tests takes $O(m)$ time each in the worst case. This process leads to the worst-case time complexity of $O(n/k\log(n/k) + nml/k)$ for each mapper. Each reducer performs at most $O(l^2k)$ dominance tests taking $O(m)$ time in the worst case. It leads to the worst-case time complexity of $O(l^2mk)$ for each reducer. There are cases the number of local skylines is $l = O(m)$ in the best cases. The algorithm could run in $O(n/k\log(n/k))$ time. On the other hand, there exist cases where $l = O(n/k)$, in the worst case, and the time complexity of this algorithm is $O(n^2m/k)$.

Concerning the previous sections, most of the properties and techniques used in spatial skyline queries cannot be used to the weighted spatial skyline query because the nature of the problems is quite different. Therefore, this strategy is introduced to accelerate computations by taking advantage of the benefits of parallel implementation.

5 Experimental results

In this section, the two subsections introduce the experimental settings and the effectiveness evaluations on the strategies applied in the proposed algorithm MR-WSS, respectively.

5.1 Experimental settings

All experiments were performed by an Intel Xeon E7-8890 v4 processor, with a total speed of 2.2 GHz, 28 cores (48 threads) in total, and 512 GB of memory. We executed all the implementations in Spark 2.4.7 and Scala 2.11.12 on Linux 3.10.0.

The actual weighted data sets are not easy to obtain, so experiments have been performed with synthetic data. We consider several data sets *P* with different cardinalities from 2 to 100 K, and three data sets *Q* with 50, 100, and 150 queries as synthetic data sets. Because of the role of query points in the problem, their cardinality is much smaller than the cardinality of *P*. The sets *P* and *Q* were generated randomly in the squared domain [0, 1] × [0, 1], and the weights of data points were also generated within [1, 10]. Figure 8 shows a set *P* with 100 weighted points and a set *Q* with 5 query points that the weighted points are blue, the query points are red, and the obtained weighted skyline points are green. We label the skyline points obtained with their weights in the image. This figure reflects that the weighted spatial skyline points tend to be too close to the query points and, in most cases, have a significant weight.

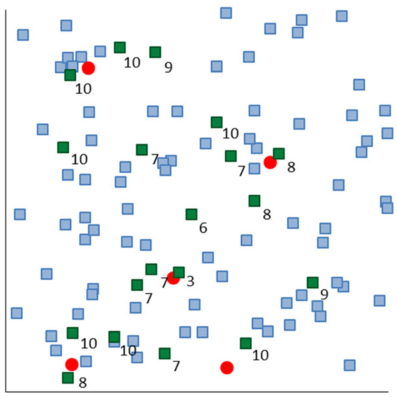


Fig. 8 In blue 50 weighted points, in red 5 query points and in green all the weighted skyline points

5.2 Performance analysis

In this section, we evaluate the performance of the proposed MapReduce-based algorithm on several synthetic data sets. The following comparison algorithms were selected to be implemented in the same experimental environment to assess our proposed algorithm MR-WSS.

DEFT is a serial and base algorithm to obtain weighted spatial skyline points. WDS [13] is another serial algorithm that is an adaptation of [34]. It sorts the interest points according to their distance to an arbitrary query point. Then, it selects the closest point of interest as the first weighted spatial skyline point and checks other points of interest to add to the set of the weighted spatial skyline points. There is no parallel algorithm for weighted spatial skyline points based on the MapReduce framework. PWBF [13] is the only existing parallel solution that uses GPU for weighted spatial skyline queries. Due to the different implementation environments, we only used it to compare speedup.

Figures 9, 10, and 11 provide the information related to the performance of the mentioned algorithms when considering several sets of interest points and query points. We ran the algorithms with 21 pairs of *P* and *Q* sets, repeated ten times for each experimental group, and reported the average value to obtain more reliable results.

We can see the number of weighted spatial skyline points for the different cardinalities of *P* and *Q* in Fig. 9. The output size increases as the number of interest points and query points increases. Based on the results obtained with different sets, between 8 and 31% of the points of P are skyline points. Figure 10 shows the execution times of the presented algorithms. We observe that the proposed algorithm MR-WSS is faster than the DEFT and WDS in all cases. When the number of points of interest is small, the execution time of the three algorithms is almost the same. The reason is that the number of dominance tests performed by the algorithm is small. The execution time of

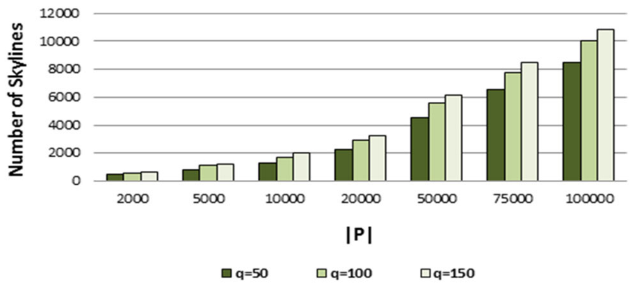
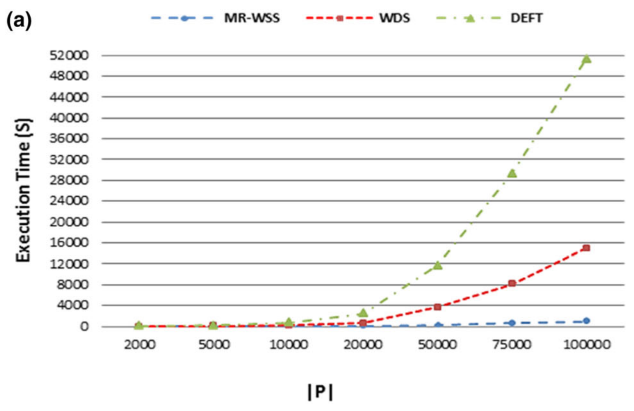
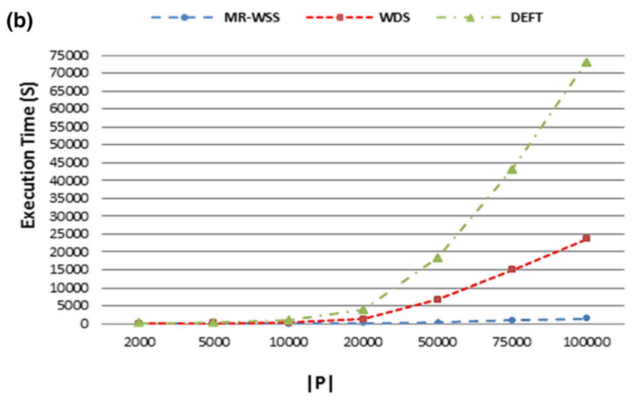


Fig. 9 Output size for different cardinalities of *P* and *Q*





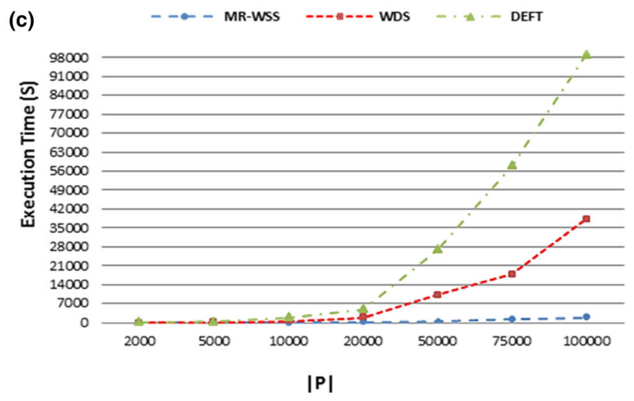
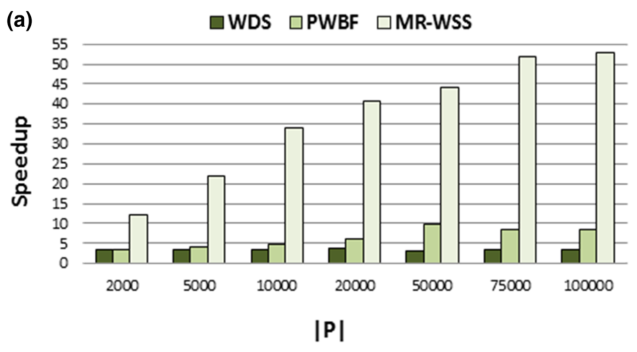
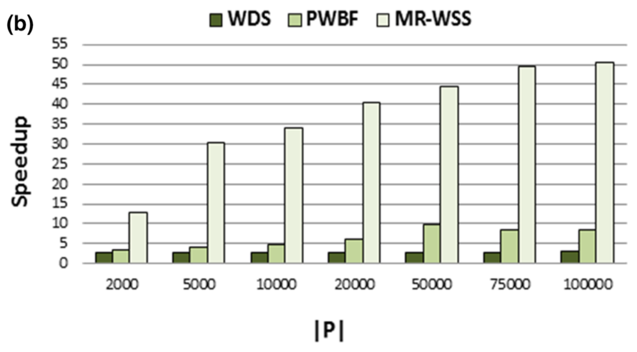


Fig. 10 Execution time at different data sizes of P when **a** $|Q| = 50$, **b** $|Q| = 100$, **c** $|Q| = 150$





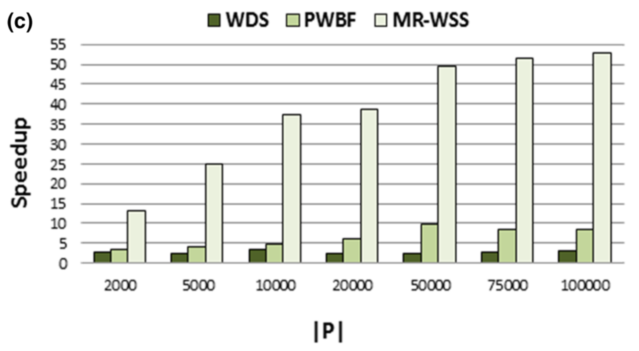


Fig. 11 Speedup MR-WSS vs WDS and PWBF at different data sizes of P when **a** $|Q| = 50$, **b** $|Q| = 100$, **c** $|Q| = 150$

DEFT increases dramatically as the number of points of interest increases because this algorithm performs the dominance test for all pairs of points of interest. According to the experiment results, WDS has a shorter execution time than DEFT in all test cases because it performs less dominance test than DEFT. The proposed algorithm MR-WSS has the least execution time among these algorithms, and its execution time does not increase much as the number of points of interest increases. The reason is that it performs fewer dominance tests based on Lemma 2, similar to WDS. MR-WSS is faster than WDS because WDS sorts all data points serially, but MR-WSS sorts them in parallel by mappers. In addition, WDS does the dominance tests to obtain weighted spatial skyline points in serial, but MR-WSS first computes the local skylines by mappers and then obtains the global skylines by reducers. MR-WSS takes advantage of the benefits of parallel performing of the dominance tests.

To easily visualize how many times MR-WSS is faster than WDS and PWBF, in Fig. 11, we present the speedup of three methods related to DEFT. These speedups vary from 3.3 to 52.8 for $|Q| = 50$, 2.9 to 50.6 for $|Q| = 100$, and 2.7 to 52.9 for $|Q| = 150$. We can see that PWBF always has a higher speedup than WDS, although it does more operations as the number of points of interest increases.

The reason is that PWBF exploits the parallel capabilities of the GPU. According to the experiment results, MR-WSS has the highest speedup among these algorithms. By comparing the complexities of PWBF and MR-WSS, they theoretically have the same time complexity. PWBF has the $O(n^2m/k)$ time complexity when $O(n)$ threads are considered [13]. It is not possible in the real world. Therefore, GPU hardware has memory limitations, and the MapReduce scheme can efficiently deal with large-scale data sets. Finally, the experiment results demonstrate that the proposed method is a scalable and fast algorithm.

6 Conclusions and future work

We have theoretically studied and proposed a novel solution for the weighted spatial skyline query, namely MR-WSS, which can be applied to different decision-making applications. Traditional methods cannot efficiently process the weighted spatial skyline queries because most of their properties are useless for the weighted spatial skyline queries. Therefore, we design a parallel algorithm utilizing the MapReduce framework to improve the performance of the weighted spatial skyline queries. The proposed algorithm MR-WSS prevents the bottleneck of centrally finding the global spatial skyline from the local spatial skylines. It uses an adaptation of the traditional methods to compute the local set of weighted spatial skyline points in the mappers. Then, it distributes a set of dominated subset operations to reducers to find the global weighted spatial skyline points. The experimental results show that MR-WSS is a scalable and fast algorithm because the execution time increases much slower than other algorithms while the data points size increases. In addition, MR-WSS outperforms WDS and PWBF algorithms by a factor between 4 and 13 in most analyzed settings.

In the future, to further improve the performance, we are interested in controlling the load balance of the tasks assigned to the mappers and reducers. Another future work is to extend the proposed method to support the weighted spatial skyline queries over data streams. The proposed algorithm will be adapted to manage resources for virtual machine migration in cloud computing. The expanded solution can improve the performance to cases where more intelligence is required to improve the performance of fog or edge computing in task scheduling based on different parameters such as response time, quality of results, and energy.

Funding The authors have not disclosed any funding.

Data availability The datasets generated during and/or analysed during the current study are not publicly available but are available from the corresponding author on reasonable request.

Declarations

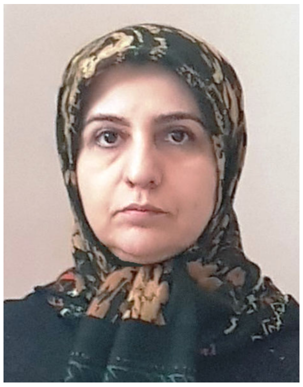
Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings 17th International Conference on Data Engineering, pp. 421–430. IEEE (2001)
2. Bai, M., Jiang, S., Zhang, X., Wang, X.: An efficient skyline query algorithm in the distributed environment. J. Comput. Sci. **58**, 101524 (2022)
3. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. VLDB **7**, 291–302 (2007)
4. Lee, K.C., Zheng, B., Li, H., Lee, W.-C.: Approaching the skyline in Z order. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 279–290 (2007)
5. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. **30**(1), 41–82 (2005)
6. Ryu, H.-C., Jung, S.: MapReduce-based skyline query processing scheme using adaptive two-level grids. Clust. Comput. **20**(4), 3605–3616 (2017)
7. Bartolini, I., Ciaccia, P., Patella, M.: Efficient sort-based skyline evaluation. ACM Trans. Database Syst. **33**(4), 1–49 (2008)
8. Cho, S.-R., Lee, J., Hwang, S.-W., Han, H., Lee, S.-W.: VSkyline: vectorization for efficient skyline computation. ACM SIGMOD Rec. **39**(2), 19–26 (2010)
9. J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with presorting: Theory and optimizations,” in Intelligent Information Processing and Web Mining: Springer, 2005, pp. 595–604.
10. Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. VLDB **5**, 229–240 (2005)
11. Zhang, S., Mamoulis, N., Cheung, D.W.: Scalable skyline computation using object-based space partitioning. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 483–494 (2009)
12. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 751–762. Citeseer (2006)
13. Fort, M., Sellarès, J.A., Valladares, N.: Nearest and farthest spatial skyline queries under multiplicative weighted Euclidean distances. Knowl.-Based Syst. **192**, 105299 (2020)
14. Islam, M.S., Liu, C., Rahayu, W., Anwar, T.: Q+ tree: An efficient quad tree based data indexing for parallelizing dynamic and reverse skylines. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 1291–1300 (2016)
15. Afrati, F.N., Koutris, P., Suciu, D., Ullman, J.D.: Parallel skyline queries. Theory Comput. Syst. **57**(4), 1008–1037 (2015)
16. Cuzzocrea, A., Karras, P., Vlachou, A.: Effective and efficient skyline query processing over attribute-order-preserving-free encrypted data in cloud-enabled databases. Futur. Gener. Comput. Syst. **126**, 237–251 (2022)

17. Ding, L., Wang, S., Song, B.: Efficient k-dominant skyline query over incomplete data using MapReduce. Front. Comp. Sci. **15**(4), 1–14 (2021)
18. Gavagsaz, E.: Parallel computation of probabilistic skyline queries using MapReduce. J. Supercomput. 1–27 (2020)
19. Nasridinov, A., Choi, J.-H., Park, Y.-H.: A two-phase data space partitioning for efficient skyline computation. Clust. Comput. **20**(4), 3617–3628 (2017)
20. Park, Y., Min, J.-K., Shim, K.: Parallel computation of skyline and reverse skyline queries using mapreduce. Proc. VLDB Endow. **6**(14), 2002–2013 (2013)
21. Park, Y., Min, J.-K., Shim, K.: Processing of probabilistic skyline queries using MapReduce. Proc. VLDB Endow. **8**(12), 1406–1417 (2015)
22. Wu, J., Chen, L., Yu, Q., Kuang, L., Wang, Y., Wu, Z.: Selecting skyline services for QoS-aware composition by upgrading MapReduce paradigm. Clust. Comput. **16**(4), 693–706 (2013)
23. Zhang, J., Jiang, X., Ku, W.-S., Qin, X.: Efficient parallel skyline evaluation using MapReduce. IEEE Trans. Parallel Distrib. Syst. **27**(7), 1996–2009 (2015)
24. Wang, W., Zhang, J., Sun, M.-T., Ku, W.-S.: Efficient parallel spatial skyline evaluation using mapreduce. In: Proceedings of the 20th International Conference on Extending Database Technology (2017)
25. Ghosh, P., Sen, S., Cortesi, A.: Skyline computation over multiple points and dimensions. Innov. Syst. Softw. Eng. **17**(2), 141–156 (2021)
26. Maroc, S., Zhang, J.B.: Cloud services security-driven evaluation for multiple tenants. Clust. Comput. **24**(2), 1103–1121 (2021)
27. Chan, C.-Y., Jagadish, H., Tan, K.-L., Tung, A.K., Zhang, Z.: On high dimensional skylines. In: International Conference on Extending Database Technology, pp. 478–495. Springer (2006)
28. Chan, C.-Y., Jagadish, H., Tan, K.-L., Tung, A.K., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp. 503–514 (2006)
29. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: the k most representative skyline operator. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 86–95. IEEE (2007)
30. Li, Z., Peng, Z., Yan, J., Li, T.: Continuous dynamic skyline queries over data stream. J. Comput. Res. Dev. **48**(1), 77–85 (2011)
31. Tao, Y., Xiao, X., Pei, J.: Efficient skyline and top-k retrieval in subspaces. IEEE Trans. Knowl. Data Eng. **19**(8), 1072–1088 (2007)
32. Gao, Y., Liu, Q., Zheng, B., Chen, G.: On efficient reverse skyline query processing. Expert Syst. Appl. **41**(7), 3237–3249 (2014)
33. Sharifzadeh, M., Shahabi, C., Kazemi, L.: Processing spatial skyline queries in both vector spaces and spatial network databases. ACM Trans. Database Syst. **34**(3), 1–45 (2009)
34. Lee, M.-W., Son, W., Ahn, H.-K., Hwang, S.-W.: Spatial skyline queries: exact and approximation algorithms. GeoInformatica **15**(4), 665–697 (2011)
35. Bhattacharya, B., Bishnu, A., Cheong, O., Das, S., Karmakar, A., Snoeyink, J.: Computation of spatial skyline points. Comput. Geom. **93**, 101698 (2021)
36. Aurenhammer, F., Edelsbrunner, H.: An optimal algorithm for constructing the weighted Voronoi diagram in the plane. Pattern Recogn. **17**(2), 251–257 (1984)



Elaheh Gavagsaz received master degree from Islamic Azad University in 2011. She received the Ph.D. in computer software engineering at Science and Research Branch, Islamic Azad University in 2018. Her current research focuses on the big data, cluster computing and distributed systems.